
MASTER'S THESIS 2021

# Evaluation of an RGB-D-data-based Gaze Tracker for Human-Robot Interaction

Josef Abdul Al, Daniel Siljegovic Persson

ISSN 1650-2884
LU-CS-EX: 2021-41

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2021-41

# Evaluation of an RGB-D-data-based Gaze Tracker for Human-Robot Interaction

# Utvärdering av en RGB-D-databaserad Gaze Tracker för människa-robot interaktion

**Josef Abdul Al, Daniel Siljegovic Persson**

# Evaluation of an RGB-D-data-based Gaze Tracker for Human-Robot Interaction

Josef Abdul Al
elt15jab@student.lu.se

Daniel Siljegovic Persson
cif06dsi@student.lu.se

September 9, 2021

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisor: Elin Anna Topp, Elin_Anna.Topp@cs.lth.se

Examiner: Jacek Malec, jacek.malec@cs.lth.se

**Abstract**

Eye gaze and body language are very important parts in interaction between people. The eye gaze especially is a clear indicator of where a person's attention is directed. For a robot to be able to communicate effectively with humans it is important to find the person's gaze direction. The purpose of this master thesis is to implement and evaluate a gaze tracker for use in human-robot interaction. This is done by using a pipeline of convolutional neural networks to extract facial features from an RGB-D image, and using those features to estimate head pose and gaze direction. The accuracy of the gaze tracker is evaluated using a robot system, and a series of other tests.

**Keywords**: Gaze Tracking, Computer Vision, Machine Learning, Neural Networks, Head Pose Estimation, Human-Robot Interaction

# Acknowledgements

We would like to thank our supervisor Elin Anna Topp for her advice and assistance. We would also like to thank Filip Kronström for his assistance with the robot integration as well as his expertise with regards to the pyrealsense2 library. Last but not least we would like to thank Tom Andersen for keeping us company while working with this master thesis.

# Contents

# Chapter 1
# Introduction

## 1.1 The Field of Gaze Tracking

When people communicate they do not only use speech. They also use body language, facial expressions and eye gaze. The eye gaze especially is a clear indicator of a person's attention [5]. This is useful in a wide range of applications from safety systems in automobiles [18] [29] [37] to uses in assisted living [10].

Outside of the technical uses, eye gaze can also be useful in studying many cognitive processes [11] such as reading comprehension [13]. It can even be used in diagnosing autism in young children [28].

The purpose of our work is to implement and evaluate a gaze tracker for use in human-robot interaction (HRI). Gaze tracking in HRI is a very wide field with technical applications ranging from controlling surgical robots [23] to uses in brain-computer interfaces [33].

The research is not limited to just getting robots to correctly interpret the human gaze. There are other fields of study that research the way humans interact with robots [4] and ways of emulating the human gaze in robot systems [17], in order to bring HRI closer to human-human interaction.

## 1.2 Thesis Objectives

The goal of this thesis is twofold. The first goal is to implement a functional gaze tracker inspired by the article *Real Time Eye Gaze Tracking System using CNN-based Facial Features for Human Attention Measurement* [25] by Oliver Lorenz and Ulrike Thomas. The second goal is to evaluate whether this gaze tracker can be used for HRI, as well as the limitations of this implementation with regards to accuracy. Essentially how well it can predict what object a person is looking at and how closely different target objects can be located to each other without significant false positives.

## 1.3 Research Contribution

Since this thesis is based on an already published article [25] our focus is not so much how to build a gaze tracker but rather what that gaze tracker can be used for and how well it works. There are some differences in our implementation compared to the one described in the article. They are outlined in the theory and implementation chapters, but the main focus is to extend their proposed three step pipeline with a fourth step that actually makes use of the depth data in some practical way. The design outlined by Lorenz and Thomas includes the RGB-D data but does not actually use it for the gaze estimation. We make use of it to identify objects in the room and make predictions on which object a person is looking at. We also extend on their work by using it to do a simple experiment where we control the rotation of a robot using eye gaze.

## 1.4 Method

The work on this thesis consisted of three main parts. The first was to implement the gaze tracker as outlined in the article [25]. This in turn was also a three step process where each of the steps of their proposed pipeline was implemented. These steps are landmark regression where the facial landmarks are found, head pose estimation using said landmarks and finally gaze estimation. Once the gaze tracker was implemented the second step was to extend it with object detection functionality. This allows the the gaze tracker to use depth data to locate the subject as well as other objects in the room and estimate what object the subject is observing. The third and final step was evaluation. Here we did the robot test as well as a series of experiments where we attempted to estimate the accuracy of the tracker in different environments and use case scenarios.

## 1.5 Division of Labor

This work was very much a collaboration between the authors. Considering the pipeline structure of the work we chose to tackle each step of the pipeline together rather than try to divide it up. In retrospect such a division would probably have been possible but that was not clear to us at the time. That being said there was often one person taking the lead role with the other assisting as needed while working on their own task. A detailed description of this can be found in the appendix.

## 1.6 Outline

In Chapter 2 we introduce the theory needed to understand this thesis, as well as related work. In Chapter 3 we describe how we implemented the gaze tracker. In Chapter 4 the evaluation process of the gaze tracker is described, as well as the results of the evaluation, and a discussion of those results. In Chapter 5 our conclusions of this thesis is found. Finally Chapter 6 is the appendix.

# Chapter 2
# Background

## 2.1 Theory

In this Section we introduce some of the theory used in this master thesis. We start with a brief overview in Section 2.1.1, and continue in Section 2.1.2 by introducing Artificial Neural Networks and a particular kind of Neural Network called a Convolutional Neural Network. In Section 2.1.3 we talk about Decision Tree Learning. In Section 2.1.4 and 2.1.5 we introduce some techniques used to estimate head pose and gaze direction, respectively.

### 2.1.1 Overview

Our gaze tracker is a pipeline of processes. The first step is to use a Convolutional Neural Network (CNN) to find facial landmarks. We use the network known as the multi-task cascaded convolutional neural network (MTCNN) [38]. This is only one way of finding facial landmarks. There are of course other approaches that are just as viable. One popular [25] [41] approach seems to be the Dlib-ml library [20].

The landmarks found by the MTCNN are used in both calculating the head pose and in determining where to crop images of the eyes that are used as input to the eye gaze estimation. The landmarks needed in order to estimate eye gaze are also found with the use of a CNN.

When the gaze tracker had been implemented some of the parameters were optimized using a decision tree based approach.

In the following Sections we will present all the theory used in the methods we chose.

### 2.1.2 Artificial Neural Networks

Artificial Neural Networks (ANN) are computing systems that are designed to vaguely resemble the neural networks that an animal brain consists of. An ANN consists of a collection of nodes called "artificial neurons". Each connection between nodes, like synapses in a brain,

can transmit a signal to other nodes. Each node can then process the signal and send it to other connected nodes. The output of a node is produced by first taking the weighted sum of all the inputs to the node, and then adding a bias term. After that the weighted sum is passed through a function called the "activation function".

The idea with ANN:s is to construct a network of nodes to solve classification and regression problems without specifically telling the computer how to solve the problems. This is done by training of the network. Training is done by presenting the network with inputs and desired outputs. The weights of the network are then updated so that the output of the network matches the desired output as closely as possible.

### Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a special kind of artificial neural networks that uses convolution, instead of the usual matrix multiplication, in at least one of the networks layers. CNN:s are usually used when working with images as input to the network. The CNN performs a convolution by taking the dot product between a convolution kernel and a part of the input image, then slides it further to the next part until it has gone through the whole image. It then sends the outputs to the next layer.

One of the biggest advantages with the CNN is that it uses a lot less weights than an ordinary ANN. If we would have used an ordinary ANN we would have one connection from each pixel of the image to each node in the next layer, and as many weights to train as connections. While for the CNN, the size of the image does not affect the number of weights, since we only have as many weights as there are elements in the convolution kernel. Also since the convolution kernel slides through the whole image, many of the pixels share the same weights.

## 2.1.3 Decision Tree Learning

Decision tree learning is a method in machine learning that uses decision trees to find a target value from a group of features. It can be used for both classification problems (classification trees) and regression problems (regression trees). Each feature is represented by a node in the decision tree and depending on the feature value of the element we are trying to find a target value for, we follow the branch corresponding to that feature value. The decision tree model is created from training data, and it is structured so that the closer the node is to the root node, the bigger is the split of the data. Depending on the kind of tree different methods can be used to find the biggest split. Some examples are information gain and reduction in variance.

Information gain is usually used in classification trees. It can be seen as a measure of decrease in entropy. It calculates the entropy before and after each split. The split with the highest decrease in entropy is placed the closest to the root node in the resulting tree model. Entropy for one attribute is calculated by the following formula:

$$E(S) = \sum_{i=1} -p_i \log_2 p_i, \tag{2.1}$$

where $S$ is a node and $p_i$ is the percentage of class $i$ in node S.

The information gain of a split can then be found by subtracting the average entropy after the split from the entropy before the split:

$$InformationGain = E(Before) - \frac{1}{K}\sum_{j=1}^{K} E(j, After), \tag{2.2}$$

where $E(Before)$ is the entropy of the data before the split, $K$ is the number of nodes created after the split, and $E(j, After)$ is the entropy of subset $j$ after the split.

Reduction in variance is usually used for solving regression problems with decision trees. This algorithm finds the best split by calculating the variance of each split. The split that gives the lowest variance is the best, and thus gets placed the closest to the root node.

### 2.1.4 The facial normal

The facial normal is the normal vector that extends from the base of the nose, through the tip of the nose and out. It corresponds well to the direction the head is facing. See Figure 2.1.

The method of estimating the facial normal from an image that we use was developed by Andrew Gee and Roberto Cipolla. It can be found in their article *Determining the gaze of faces in images* [12]. Another approach to estimating the head pose is to view it as the so called Perspective-n-Point problem [22]. That algorithm requires knowledge of the camera parameters. Which makes it hard to use on images captured by an unknown camera.

In the article by Gee and Cipolla, they present two different methods of estimating the facial normal. We implemented the one called "the 3D-method". The method works by first defining a series of landmarks and measurements on the face. The landmarks are the corners of the mouth, the nose and the eyes. These landmarks correspond to the landmarks given as output from the MTCNN. The measurements used are as follows:

- $L_f$, the distance from the midpoint of the mouth to the midpoint between the eyes.
- $L_m$, the distance from the midpoint of the mouth to the nose base (The point along the symmetry axis in the facial plane directly behind the tip of the nose).
- $L_n$, the length of the nose. Measured from tip to base. The base lies along the symmetry axis of the face. The symmetry axis is the imagined line measured by $L_f$ (shown as the dashed line in Figure 2.1).

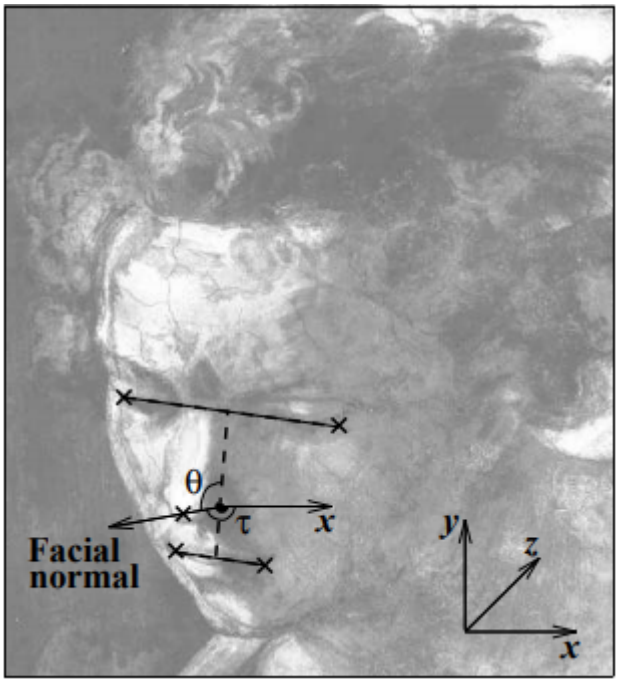



**Figure 2.1:** The image above shows the facial normal, the angle between the symmetry line and the projected facial normal in the image plane, $\theta$, and the angle between the projected facial normal in the image plane and the x-axis of the image plane, $\sigma$.
Image source: *Determining the gaze of faces in images* [12]

The measurements are then used to define the ratios $\boldsymbol{R}_m \equiv \frac{L_m}{L_f}$ and $\boldsymbol{R}_n \equiv \frac{L_n}{L_f}$. Each measurement has a corresponding image measurement that is denoted with lower case l, i.e. $l_f, l_m, l_n$. The facial normal is calculated using the two angles $\tau$ and $\sigma$ and is defined as follows:

$$n = [sin\sigma cos\tau, sin\sigma sin\tau, -cos\sigma] \tag{2.3}$$

The angle $\tau$ is the tilt angle and is defined as the angle between the projection of the normal in the image plane and the x-axis of the image plane. The projected normal is defined as the vector from the nose base to the nose tip in the image. The nose tip is a landmark given by the MTCNN and is therefore easily found. The nose base however is a bit harder. It can not be easily calculated using the landmarks and an assumption must therefore be made as to the symmetry of the face. We found that $\boldsymbol{L}_m = \frac{L_f}{2}$ is a decent assumption. $\boldsymbol{L}_f$ is easily computed using the MTCNN landmarks. The projected normal is then found by calculating the vector between the nose base, which is found using the measurement $\boldsymbol{L}_m$ and the symmetry axis, as well as the nose tip, which is given by the MTCNN. The angle $\tau$ is then calculated using the scalar product of the projected normal and the unit vector $x = [1, 0]$ as follows:

$$\tau = \arccos \frac{proj\_normal \cdot [1, 0]}{|proj\_normal|} \tag{2.4}$$

There is a coordinate system defined by the facial normal and the facial plane. The z-axis of this coordinate system is parallel with the facial normal. The facial plane is the plane that is spanned by the corners of the mouth and the centers of each eye. The facial symmetry axis

lies in this plane. The x-axis is parallel with the line connecting the corners of the mouth. The y-axis runs parallel with the facial symmetry axis (see Figure 2.2). The angle $\sigma$ is defined as the angle between the facial normal, which is the same as the z vector of the facial coordinate system and the image axis $\hat{d}_z$. $\hat{d}_z$ is the z-axis unit vector of the image plane coordinate system, transformed into the facial coordinate system.

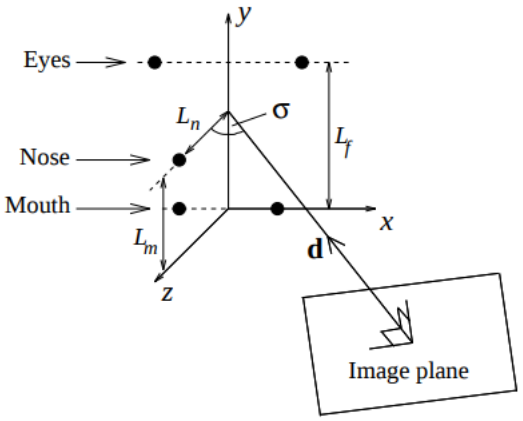



**Figure 2.2:** The facial coordinate system.
Image source: *Determining the gaze of faces in images* [12]

In order to compute $\sigma$ we first define $m_1 \equiv (\frac{l_n}{l_f})^2$ and $m_2 \equiv cos^2\theta$. The angle $\theta$ is defined as the angle between the facial symmetry axis and the projected normal (see Figure 2.1). Since $\sigma$ is the angle between the normal and the image axis z. This means that in order to calculate $\sigma$, we must first calculate the image axis z in the coordinates of the face. It can be calculated as follows:

$$R_n^2(1 - m_2)\hat{d}_z^4 + (m_1 - R_n^2 + 2m_2R_n^2)\hat{d}_z^2 - m_2R_n^2 = 0 \tag{2.5}$$

This equation can be seen as a second order polynomial equation in $\hat{d}_z^2$. It is proven in [12] that there exists only a single root between 0 and 1. This means that $\sigma$ can be computed as:

$$\sigma = cos^{-1}\sqrt{d_z^2} \tag{2.6}$$

With both the tilt angle $\tau$ and the slant angle $\sigma$ computed, the facial normal is computed according to equation (2.3).

### 2.1.5 Eye gaze

The eye gaze estimation is designed to work in conjunction with the head pose estimation, where the final gaze vector is the sum of the eye gaze and facial normal vectors. The head pose vector is weighted in order to adjust its influence on the final gaze vector. This allows for a simple model to be used in estimating the eye gaze. The model is based on three landmarks in the eye. The pupil, the inner corner and the outer corner. Using the two corners, the midpoint of the eye can be found. The vector from the midpoint to the pupil can then be calculated and used as the x- and y-components of eye gaze vector (see Figure 2.3).

When people look straight into the camera small variations in the x- and y-axis can become very large after normalizing the vector. To compensate for this we also add a z-component.

The z-component is $eyewidth \cdot w_z$ where $w_z$ is a scaling factor used to determine the size of the z-component. Since x and y are dependant on how much the pupil shifts away from the eye center they tend to be quite small and having a z-component that is too large will make the eye gaze look almost directly forward with very little horizontal and lateral movement. By having a weight, $w_z$, that is quite small we prevent this from happening.

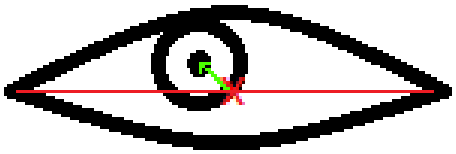


**Figure 2.3:** Eye midpoint marked with red x. Eye gaze vector marked in green.

The method described above works well assuming the eye is completely symmetrical and the midpoint lies perfectly over the pupil when the eye is looking straight ahead. Unfortunately most eyes are not completely symmetrical and the midpoint tends to lie below the pupil (see Figure 2.4).

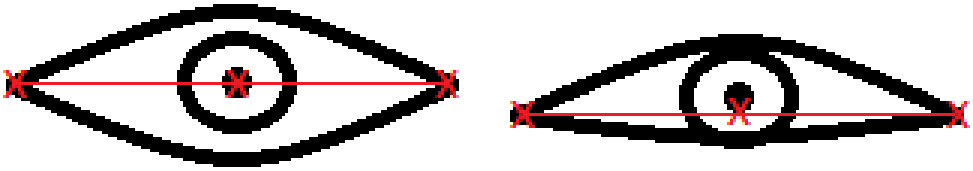


**Figure 2.4:** Comparison between symmetric and asymmetric eye.

To compensate for this we compute a small vector that shifts the eye midpoint towards the center of the eye. This is done by first computing the vector that runs along the line connecting the eye corners. This vector can then be used to find the orthogonal vector in the direction of the eye center. The dot product of two orthogonal vectors is 0. Using this fact, the orthogonal vector of any two dimensional vector $(a, b)$ is defined as $\pm(b, -a)$.

*Proof.*

$$(a, b) \cdot (b, -a) = ab + b(-a) = ab - ba = 0$$

□

Using this shift vector, the eye gaze can then be computed as follows.

$$eye\_gaze = pupil - (eye\_midpoint + w_s \cdot shift\_vector) \tag{2.7}$$

By scaling the shift vector appropriately ($w_s$ in the equation above) the eye gaze algorithm can be calibrated for any eye shape. The final gaze vector is described by the following expression:

$$gaze\_vector = w_f \cdot facial\_normal + eye\_gaze \tag{2.8}$$

The weight $w_f$ consists of the sum of a scalar that acts as a generic weight and the norm of a two-dimensional vector consisting of the x and y components of the eye gaze vector. The idea here is to scale down the effect of the head pose vector when the subject is looking directly into the camera.

This method is fairly straightforward and easy to implement. There are of course more sophisticated methods like the one described by Seonwook Park et al. in their article *Deep Pictorial Gaze Estimation* [27]. They recommend an approach where an intermediate pictorial representation is used to model the eyeball before the gaze is estimated.

While interesting we felt that this approach adds needless complexity to a system that needs to be able to run on video in real-time.

## 2.2 Related Work

Research on gaze estimation has been going on for a long time. Early methods relied on electrodes on the skin around the eye region to detect eye movements [36] or large helmet like devices [6]. Other methods create a 3d model of the eye and estimate the gaze direction using this model [8]. With the development of computer vision, modern gaze trackers can use images captured by cameras to estimate a person's gaze direction.

The first article to describe an approach using convolutional neural networks (CNNs) in gaze estimation is *Appearance-based gaze estimation in the wild* by X. Zhang et al [40]. The CNNs are used to extract the gaze features. Those features are then used in the gaze estimation. Since the publishing of their article, this approach has been used in several other published papers [41] [21] [9]. This approach is also the basis of the gaze tracker described by Oliver Lorenz and Ulrike Thomas in their article *Real Time Eye Gaze Tracking System using CNN-based Facial Features for Human Attention Measurement* [25], which the gaze tracker described in this master thesis is based on.

The main limitation with the CNN-based gaze trackers is the reliance on training data. In real-world scenarios there are a lot of variables that come in to play. The subjects might have facial features different from what is seen in the training data, the camera might be placed differently and the lighting might also be different just to name a few. A lot of the current research is targeted towards overcoming these limitations.

An approach to creating an automated calibration process to overcome individual differences in appearance is described in *Offset Calibration for Appearance-Based Gaze Estimation via Gaze Decomposition* by Chen, Zhaokang and Shi, Bertram E.

In the article *Gaze Estimation in the Dark with Generative Adversarial Networks* [19] by Kim and Jeong, they use generative adversarial networks (GANs) to overcome the differences in lighting conditions. GANs are also used in the article *PureGaze: Purifying Gaze Feature for Generalizable Gaze Estimation* [7] by Yihua Cheng, et al. Their approach is to purify the input to make it independent of illumination and identity while keeping the gaze features.

There are commercially available products [34] for eye tracking, both appearance based eye trackers, and so called wearable eye trackers. The main advantage of wearable trackers is that they never suffer from pose occlusion like a stationary camera can, which is useful when gaze tracking subjects in dynamic motion [28]. Commercial trackers can offer very good accuracy [39]. They can even measure pupil dilation [42], which has a number of uses in studying the cognitive processes [11].

# Chapter 3

# Approach

## 3.1 Method

In this Section we describe the process of implementing the gaze tracker. The implementation is based on the work done by Oliver Lorenz and Ulrike Thomas, outlined in their article *Real Time Eye Gaze Tracking System using CNN-based Facial Features for Human Attention Measurement* [25]. We use the same methods and algorithms as those described in the article, but not always implemented in the same way. We also removed some smaller components of the pipeline that we found to not make a very big difference in accuracy, such as the kernelized correlation filter and the coordConv [24] layer.

We also expand on their work by combining the gaze tracker with a system for object identification, the purpose of which is to see if the estimated gaze vector is accurate enough to determine what object a person is looking at in a room. This allows us to evaluate the feasibility of using the gaze tracker to facilitate human-robot interaction, as well as evaluating the limits for such use.

## 3.2 Implementation

The gaze tracker consisted of three main parts. Face detection and facial landmark detection, head pose estimation, and finally finding eye features and estimating the eye gaze. An RGB-D image is sent in to a multi-task convolutional neural network (MTCNN) that finds the face and facial landmarks. After that the head pose is estimated, and the eye regions are sent in to another CNN that calculates the coordinates of the pupils and corners of the eyes. Finally the eye gaze is calculated using the relation between the pupils and the eye corners.

### 3.2.1 Face Detection and Facial Landmarks

The multi-task convolutional neural network (MTCNN) was developed by Zhang et al and is described in their article *Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks* [38]. It uses a three step pipeline consisting of one convolutional neural network in each step. The algorithm simultaneously does face detection and facial landmark localization. The landmarks the algorithm finds are both the eyes, the nose and the corners of the mouth (see Figure 3.5a).

We started by evaluating two different implementations of the MTCNN algorithm for use in our system. One is the same implementation used by Lorenz and Thomas [25]. It is an open source implementation publicly available on github [2]. The other is an implementation written by Iván de Paz Centeno [1]. It is also an open source implementation publicly available on github, but this project is also part of the Python Package Index (PyPi), which makes it very easy to install and use.

The two MTCNN implementations differ in one significant way. The version from PyPi is pre-trained and can not easily be retrained on new training data, whereas the version originally used by Lorenz and Thomas [25] can be trained on any training data provided.

When training the MTCNN used by Lorenz and Thomas [25] [2] we used the WIDER Face dataset, proposed by Yang et al. in their article *WIDER FACE: A Face Detection Benchmark* [35], and the dataset proposed by Sun et al. in their articel *Deep Convolutional Network Cascade for Facial Point Detection* [32]. After training the network we found that the pre-trained version was a lot more accurate so we ended up using the pre-trained network.

**Figure 3.1:** Pretrained MTCNN (left) vs manually trained MTCNN (right).

We also considered a third version of the MTCNN. This one was much faster when running it in real time but it was also less accurate so we did not end up changing the network.

### 3.2.2 Head Pose Estimation

To estimate the head pose, the first step was to find the nose base. This was done by using the facial landmarks from the MTCNN to find the middle of the mouth and middle of the eyes, and connecting these two points. We assume that the nose base is at some distance, $R_m$, from the middle of the eyes on that line. We then connected the found nose base and the nose tip, estimated by the MTCNN, to get the normal vector projected onto the image plane. The projected normal is then used to calculate the angles $\tau$ and $\theta$. The variables $m_1 = (\frac{l_n}{l_f})^2$ and $m_2 = cos^2(\theta)$ are defined and used to calculate $d_z^2$ using equation 2.9. Then using $d_z$ we calculate the angle $\sigma$ and use the three angles $\tau$, $\theta$ and $\sigma$ to find the head pose vector (see Figure 3.5b).

### 3.2.3 Eye Gaze Estimation

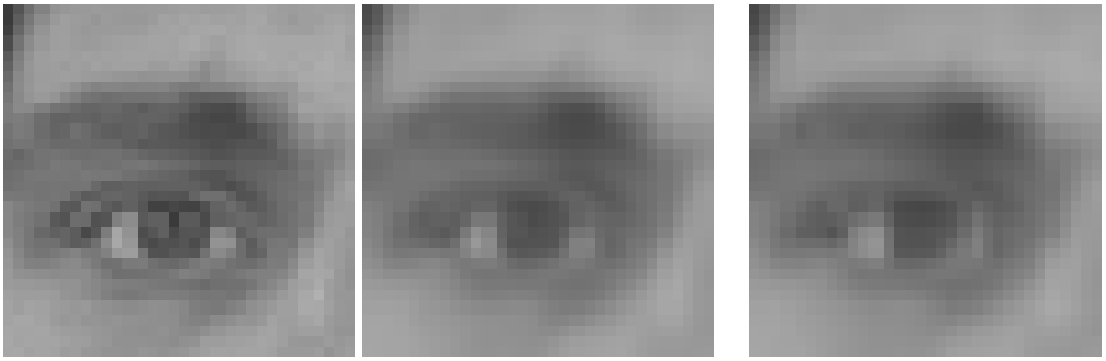
To find the eye gaze we trained a CNN to detect eye corners and pupils from images of eyes. We used a similar network structure to the network used by Lorenz and Thomas [25], but with a few differences. We decided not to use the coordConv layer [24] as we found that it made almost no difference when it came to accuracy. It also had severe compatibility issues with several versions of Tensorflow. We also saw a tendency towards under-fitting when training. To solve this problem we decided to remove the dropout layer.

Lorenz and Thomas used their network to predict the eye corners and relied on the eye landmarks from the MTCNN for the pupils. We use a different MTCNN implementation than they do and we could not detect any indications that the eye landmarks are related to the pupil location. Instead they seem to indicate the center of the eyes with respect to the eye corners. Therefore we chose to include pupil regression as a feature of the CNN. This means that the output vector, as well as the target vector, used in training our network has length 6. The vectors contain the points x and y representing the coordinates for each landmark (pupil, inner corner and outer corner). Since the network outlined by Lorenz and Thomas only detects the corners it has length 4 in both output and target vectors. This change mostly affects the input and output layers of the model. A detailed description of the model we used can be found in the appendix.

We initially started with the Kaggle facial keypoints detection dataset [14] as our training data. We chose this as it was the dataset used by Lorenz and Thomas. Unfortunately the kaggle dataset does not come with annotated testing data and of the roughly 7000 images in the training data, only around 2200 has all of the relevant annotations. Splitting this already limited data into training, validation and testing data gave too little training data to work with. In an attempt to extend the training data we chose to implement an image augmentation scheme.

**Figure 3.2:** Example images from the Kaggle facial keypoints detection dataset.

For each image we cropped 4 new images. This was done in a randomized way so that the images were cropped slightly differently. This gave us the original image as well as the 4 alternate croppings for a total of 5 images. Each of these images was then duplicated, and the duplicate was mirrored. This gave us a total of 10 images. These 10 images where then duplicated and to half of the images, Gaussian blur was applied to bring the images slightly out of focus. The other half were down-up scaled. This means that the image was reduced in height and width by a random percentage in a certain interval, resulting in a smaller image. It was then resized back to the original size. This resulted in slightly pixelated images. Using this strategy we were able to extend the training data by a factor of 20.



**(a)** Original image. **(b)** Blurred image. **(c)** Down-up sampled image.

**Figure 3.3:** Examples of image augmentation.

Despite the extensive image augmentation we were unable to reach an accuracy higher than 80%. We concluded that the training data was still insufficient and decided to include the MUCT Face Database [26]. The MUCT Face Database contains an additional 3755 images. This significantly increased the original 2200 images we got from Kaggle. The aim of the MUCT Face database is to "add more diversity of lighting, age, and ethnicity" than other datasets. This is highly welcomed as the kaggle dataset is fairly uniform with regards to lighting. The image augmentation was also performed on this dataset.

With the addition of the MUCT data we were able to improve the testing accuracy to 98.33%.

Using the eye corners and pupil extracted from the trained CNN we can find the eye gaze vector using the method described in Section 2.1.5.

**Figure 3.4:** Example images from the MUCT Face Database.

### 3.2.4 Putting it together - Gaze Vector

The final gaze vector is the weighted sum of the head pose and the eye gaze vectors, where the eye gaze is the sum of the gaze vector for the left and right eye. See Section 2.1.5 for the details.

When running the gaze tracker in real time, we found that the gaze vector varies slightly for each image frame. To solve this we used a moving average filter on the latest frames in an attempt to stabilize the gaze vector. The filter has a length of 5 and we filter the signal twice. We found this improvement to greatly increase the stability of the vector between frames.

**(a)** Face detection and facial landmarks.

**(b)** Head pose estimation vector.

**(c)** Eye corners, pupils (yellow) and final gaze vector (green).

**Figure 3.5:** The three steps of the gaze tracker.

### 3.2.5 Object Detection

In combination with the gaze tracker we also used the YOLOv3-network [30] to detect objects, in order to find which object a person is looking at. This was done by first finding the objects in the room, using the YOLOv3-network. The object detector returns bounding boxes for the different objects. By calculating the center of each bounding box it was then possible to draw a vector from the persons face to the different objects and then compare the angle between each vector and the gaze vector. The vector with the smallest angle between itself and the gaze vector shows the object the person is looking at. To find the vectors to the objects we used the depth data from the RGB-D camera.

The camera is an Intel realsense L515. It has a maximum resolution of 1920 x 1080 pixels at 30 fps and it uses LiDAR to take depth measurements in the image. Intel offers the Intel® RealSense™ SDK 2.0 library [15] as way of interacting with the camera. This library has a python wrapper called pyrealsense2 [16] that we made use of to extract image and depth data from the camera. There is a function called *deproject_pixel_to_point* which allows for the mapping of any point in the image to a real world point with x,y,z-coordinates. This allows us to calculate the vectors needed for predictions.

### 3.2.6 Parameter Optimization

The parameters, $\boldsymbol{R}_m$ and $\boldsymbol{R}_n$, used to construct the head pose vector (see Section 2.1.4) can vary a bit between different individuals. To estimate an appropriate value for these parameters, we compared our head pose vector with the head pose vectors in the Columbia Gaze dataset [31], and adjusted $\boldsymbol{R}_m$ and $\boldsymbol{R}_n$ to minimize the average angle between our head pose vector and the vectors from the dataset. The Columbia Gaze dataset is a dataset with 5880 images of different people with varying head pose and gaze direction.

To get the optimal parameters we used forest_minimize in the scikit optimize python library. forest_minimize uses a decision-tree-based regression model to sequentially evaluate the objective function, defined by the user. After trying a set of parameters, on each run it makes an educated guess on which parameters that will improve the objective function. By doing this it can find the parameters that minimize the objective function.

We also used the same method to find the vector that shifts the middle of the eye, the weight for the added z-component, as well as the weight used when combining the head pose vector with the vector from the eyes to get the final gaze vector (see Section 2.1.5). This time the function that we wanted to minimize was the angle between our predicted gaze vector and the gaze vectors from the Columbia Gaze dataset.

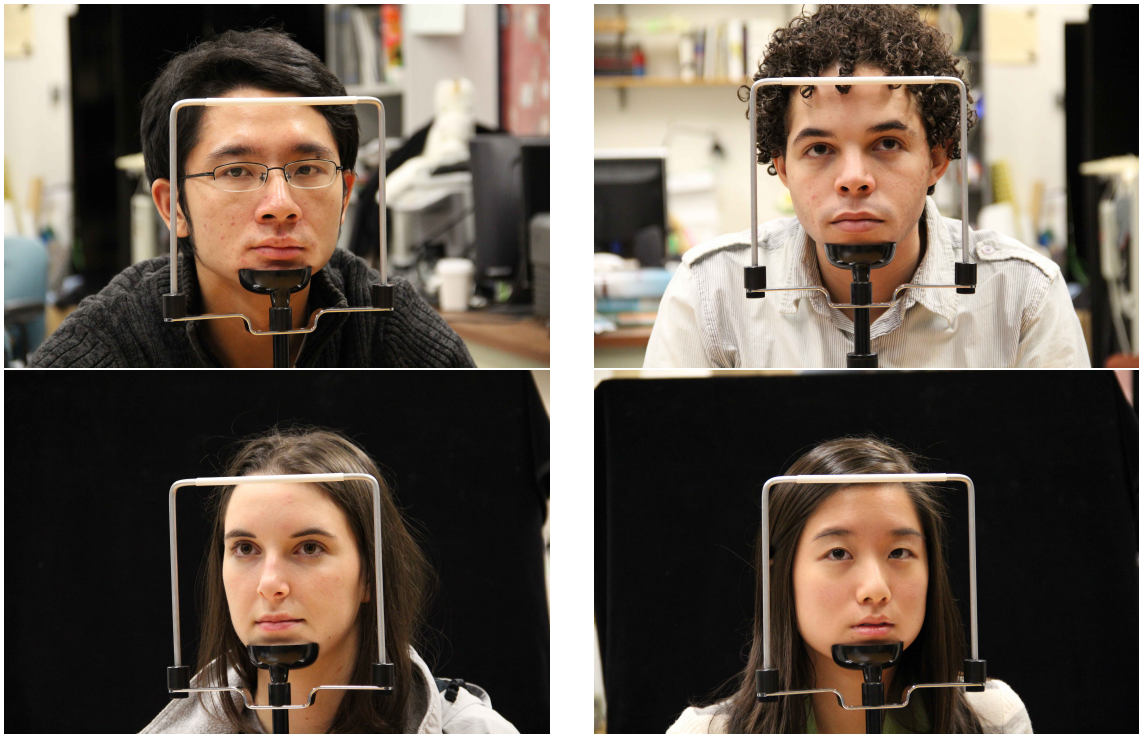


**Figure 3.6:** Example images from the Columbia Gaze dataset.

### 3.2.7 Performance Optimization

The gaze tracker needs to run in real time at a frame rate > 20 fps. This severely limits the amount of time each CNN call and the other computations can take. With 20 fps as the lower bound this means that each gaze prediction must complete in less than 50 ms. In order to hit this target we had to place a few constraints on the functionality of the gaze tracker. The object identification done by the YOLO-net takes roughly 90 seconds on a 1920 x 1088 pixel image (YOLO needs all dimensions to be multiples of 32). Scaled down to 640 x 352 pixels it still takes roughly 20 seconds. This means that the YOLO prediction can only be done once, right after start up. This of course limits any predictions between gaze and the objects in the room to stationary objects.

Another costly operation is the MTCNN call. Using a full size image is not feasible in real time. Instead we do one full size prediction during start up in order to find the bounding box of the face. Using the coordinates of the bounding box, an image 3 times the size of said bounding box is cropped from the full image. This much smaller image is then sent to subsequent calls to the MTCNN. This allows for some head movement in the subject. Currently a full gaze prediction runs in about 40 ms. It is therefore possible to do intermittent full size MTCNN calls. Once per second would allow for slow movement to be manageable.

In order to improve MTCNN performance we evaluated a third MTCNN implementation [3]. While it offered faster predictions, roughly 3 times faster in fact, it did so at the cost of accuracy and we therefore opted not to use it.

# Chapter 4
# Evaluation

In this Section we describe the evaluation process of the gaze tracker. Section 4.1 describes the different experiments that were carried out when evaluating the gaze tracker. In Section 4.2 we show the results of our experiments. In Section 4.3 we discuss the results of the evaluation.

## 4.1 Experiments

The evaluation process consisted of three main experiments. The first experiment was about making a robot rotate in the direction the person was looking. In the second experiment we evaluated how well the gaze tracker can find which object the person is looking at when three objects were standing next to each other, to evaluate its accuracy horizontally. The third and final experiment was to evaluate how well the gaze tracker can find which object the person is looking at when the three objects where at different heights, or in other words, how well it worked vertically.

For experiments two and three we used the RGB-D camera to capture as series of ROS-bags consisting of RGB and depth data. Each sequence consisted of all the possible combinations of head orientation and gaze towards the targets. We then set up the gaze tracker to step through the captured sequences in steps of 10 frames. For each step we recorded the angle between the subjects gaze vector and the intended target of said gaze vector. We recorded 10 instances of every combination of head pose and gaze. For three targets in most experiments this means a total of 3x3x10=90 data points per sequence. The average of each head-pose/gaze combination was then calculated.

### 4.1.1 Robot Rotation

We were assisted by Filip Kronström in integrating our gaze tracker with the robot. The objective was to show that the gaze tracker can be used to control the robots movement to some extent. The experiment we did was very simple. We used the realsense camera to

capture the gaze of a person, and to see if the person was looking to the left or to the right. If the person was looking to the right the robot would rotate to the right, and if the person was looking to the left the robot would rotate to the left. The amount of rotation was fixed and not related to the magnitude of the gaze in any way. It is very much just a simple proof of concept with a lot of room for improvement.

**(a)** Rotation to the right. **(b)** Rotation to the left.

**Figure 4.1:** The images show the "Robot Rotation" experiment when the robot have rotated to the right (a) vs back to the left (b). Note that when the person is looking to its own right the robot rotates to its own right. The same goes for rotation to the left.

### 4.1.2 Objects Placed Horizontally

In this experiment we put three objects next to each other and let the subject look at the different objects in a decided order, with varying head pose. The object where placed approximately 27 cm apart from each other.

We also tried different light conditions in the room. We experimented with a relatively dark room and a much lighter room where we also lit up the subject's face with a flashlight.

We then evaluated the results by determining if the gaze tracker correctly finds which object the subject is looking at, and also by looking how far off the gaze vector is from the object. This was done by looking at the angle between the gaze vector and the vector from the subject's face to the object.

### 4.1.3 Objects Placed Vertically

In this experiment we placed objects on different heights above each other at a distance of either 1m or 1.5 m from the camera and let the subject look at the objects in a decided order, with varying head pose. The subject was, once again, positioned at a distance of 1 m, 1.5 m and 2 m away from the objects. We also experimented with different light conditions in this experiment.

### 4.1.4 Experimental Considerations

We experimented with having both the objects and the subject at different distances from the camera to see the difference in accuracy of the gaze tracker. We first placed the objects at a distance of approximately 1 m away from the camera and the subject was positioned 1 m, 1.5 m and 2 m away from the objects. We then did the same thing but this time the objects where placed at a distance of approximately 1.5 m from the camera. These distances seemed reasonable since we had limited space in the rooms where the experiments were carried out. Also if the subject of the experiment gets too far away from the camera, the resolution of the camera will be insufficient to fully capture the eye features needed for the eye gaze step of the pipeline. If the objects are too close to the camera they will take up most of the image so that the face of the subject will not be as visible.

The algorithm used for estimating the head pose assumes that any effects of perspective i.e. objects in the image closer to the camera appearing bigger are negligible. This limits the minimum distance between the subject and the camera.

Using the distance to each object, as well as the average error in degrees, we find the radius of a circle around each object. Since the radius is based on the average error this means that the gaze vector will fall within the circle 50% of the time. When the gaze vector falls outside of this circle the tracker will falsely classify which object the subject is looking at if there is another object positioned, in the direction that the gaze vector is pointing, that is closer to the vector than the intended object. This means that the distance needed to any other nearby object in order to avoid significant false positives is twice the radius of the error circle. These are the distances noted as best and worst in the experiments below.

## 4.2 Results

### 4.2.1 Robot Rotation

The experiment was a success. The robot rotated in the direction it was instructed to. As stated earlier, this was mostly a "proof of concept" so not much time was spent to evaluate how well the robot actually performs in this task.

### 4.2.2 Objects Placed Horizontally

#### Camera-Object: 100 cm, Object-Subject: 100 cm, light room

Table 4.1 below shows the average error for the different combinations of head poses and gaze directions when placing the objects 100 cm from the camera and positioning the subject 100 cm behind the objects in a light room. Gaze 1, Gaze 2 and Gaze 3 means that the subject directed their gaze towards object 1, 2 and 3 respectively. Head Pose 1, Head Pose 2 and Head Pose 3 means that the subject directed their head towards object 1, 2 and 3 respectively (see 4.2).

Table 4.2 below shows the worst and best case scenario for the minimum distance the objects can be from each other so that there is a greater than 50% chance that the gaze tracker classifies correctly which object the subject is looking at.

**Figure 4.2:** Object 1 to the left in the image, object 2 in the middle and object 3 to the right.

| | Head Pose 1 | Head Pose 2 | Head Pose 3 |
|---|---|---|---|
| Gaze 1 | 10.00 ° | 14.90 ° | 19.67 ° |
| Gaze 2 | 13.56 ° | 9.94 ° | 13.09 ° |
| Gaze 3 | 19.74 ° | 12.98 ° | 13.60 ° |

**Table 4.1:** Average error, in degrees, for the different head poses and gaze directions when the objects are 1 m from the camera and the subject is 1 m behind the objects.

| Best | Worst |
|---|---|
| 35.05 cm | 74.34 cm |

**Table 4.2:** Minimum distance for the best and worst case.

## Camera-Object: 100 cm, Object-Subject: 150 cm, light room

Table 4.3 below shows the average error for the different combinations of head poses and gaze directions when placing the objects 100 cm from the camera and positioning the subject 150 cm behind the objects in a light room.

Table 4.4 below shows the worst and best case scenario for the minimum distance the objects can be from each other so that there is a greater than 50% chance that the gaze tracker classifies correctly which object the subject is looking at.

**Figure 4.3:** Object 1 to the left in the image, object 2 in the middle and object 3 to the right.

| | Head Pose 1 | Head Pose 2 | Head Pose 3 |
|---|---|---|---|
| Gaze 1 | 10.74 ° | 11.56 ° | 13.72 ° |
| Gaze 2 | 6.21 ° | 7.33 ° | 9.79 ° |
| Gaze 3 | 7.64 ° | 6.53 ° | 6.40 ° |

**Table 4.3:** Average error, in degrees, for the different head poses and gaze directions when the objects are 1 m from the camera and the subject is 1.5 m behind the objects.

| Best | Worst |
|---|---|
| 32.67 cm | 74.43 cm |

**Table 4.4:** Minimum distance for the best and worst case.

## Camera-Object: 100 cm, Object-Subject: 200 cm, light room

Table 4.5 below shows the average error for the different combinations of head poses and gaze directions when placing the objects 100 cm from the camera and positioning the subject 200 cm behind the objects in a light room.

Table 4.6 below shows the worst and best case scenario for the minimum distance the objects can be from each other so that there is a greater than 50% chance that the gaze tracker classifies correctly which object the subject is looking at.

**Figure 4.4:** Object 1 to the left in the image, object 2 in the middle and object 3 to the right.

| | Head Pose 1 | Head Pose 2 | Head Pose 3 |
|---|---|---|---|
| Gaze 1 | 5.70 ° | 7.04 ° | 10.48 ° |
| Gaze 2 | 3.84 ° | 5.78 ° | 5.90 ° |
| Gaze 3 | 3.23 ° | 4.00 ° | 5.81 ° |

**Table 4.5:** Average error, in degrees, for the different head poses and gaze directions when the objects are 1 m from the camera and the subject is 2 m behind the objects.

| Best | Worst |
|---|---|
| 22.80 cm | 74.69 cm |

**Table 4.6:** Minimum distance for the best and worst case.

## Camera-Object: 150 cm, Object-Subject: 100 cm, light room

Table 4.7 below shows the average error for the different combinations of head poses and gaze directions when placing the objects 150 cm from the camera and positioning the subject 100 cm behind the objects in a light room.

Table 4.8 below shows the worst and best case scenario for the minimum distance the objects can be from each other so that there is a greater than 50% chance that the gaze tracker classifies correctly which object the subject is looking at.

**Figure 4.5:** Object 1 to the left in the image, object 2 in the middle and object 3 to the right.

| | Head Pose 1 | Head Pose 2 | Head Pose 3 |
|---|---|---|---|
| Gaze 1 | 10.48 º | 14.47 º | 17.63 º |
| Gaze 2 | 10.35 º | 11.80 º | 13.77 º |
| Gaze 3 | 15.90 º | 14.06 º | 12.11 º |

**Table 4.7:** Average error, in degrees, for the different head poses and gaze directions when the objects are 1.5 m from the camera and the subject is 1 m behind the objects.

| Best | Worst |
|---|---|
| 36.52 cm | 65.85 cm |

**Table 4.8:** Minimum distance for the best and worst case.

## Camera-Object: 150 cm, Object-Subject: 150 cm, light room

Table 4.9 below shows the average error for the different combinations of head poses and gaze directions when placing the objects 150 cm from the camera and positioning the subject 150 cm behind the objects in a light room.

Table 4.10 below shows the worst and best case scenario for the minimum distance the objects can be from each other so that there is a greater than 50% chance that the gaze tracker classifies correctly which object the subject is looking at.

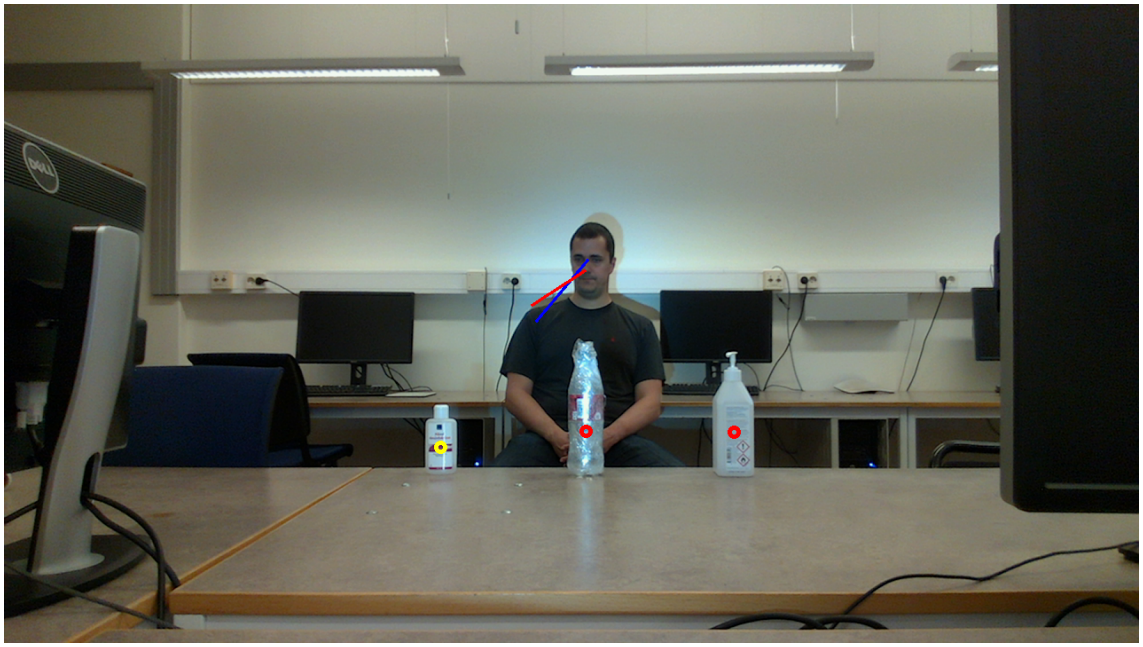


**Figure 4.6:** Object 1 to the left in the image, object 2 in the middle and object 3 to the right.

| | Head Pose 1 | Head Pose 2 | Head Pose 3 |
|---|---|---|---|
| Gaze 1 | 8.11 ° | 13.20 ° | 13.47 ° |
| Gaze 2 | 10.60 ° | 10.49 ° | 8.87 ° |
| Gaze 3 | 8.58 ° | 7.74 ° | 7.91 ° |

**Table 4.9:** Average error, in degrees, for the different head poses and gaze directions when the objects are 1.5 m from the camera and the subject is 1.5 m behind the objects.

| Best | Worst |
|---|---|
| 41.41 cm | 73.01 cm |

**Table 4.10:** Minimum distance for the best and worst case.

## Camera-Object: 150 cm, Object-Subject: 200 cm, light room

Table 4.11 below shows the average error for the different combinations of head poses and gaze directions when placing the objects 150 cm from the camera and positioning the subject 200 cm behind the objects in a light room.

Table 4.12 below shows the worst and best case scenario for the minimum distance the objects can be from each other so that there is a greater than 50% chance that the gaze tracker classifies correctly which object the subject is looking at.

**Figure 4.7:** Object 1 to the left in the image, object 2 in the middle and object 3 to the right.

| | Head Pose 1 | Head Pose 2 | Head Pose 3 |
|---|---|---|---|
| Gaze 1 | 10.12 º | 12.91 º | 11.35 º |
| Gaze 2 | 7.86 º | 10.90 º | 15.72 º |
| Gaze 3 | 8.57 º | 8.62 º | 9.87 º |

**Table 4.11:** Average error, in degrees, for the different head poses and gaze directions when the objects are 1.5 m from the camera and the subject is 2 m behind the objects.

| Best | Worst |
|---|---|
| 55.23 cm | 112.59 cm |

**Table 4.12:** Minimum distance for the best and worst case.

## Camera-Object: 100 cm, Object-Subject: 100 cm, dark room

Table 4.13 below shows the average error for the different combinations of head poses and gaze directions when placing the objects 100 cm from the camera and positioning the subject 100 cm behind the objects in a dark room.

Table 4.14 below shows the worst and best case scenario for the minimum distance the objects can be from each other so that there is a greater than 50% chance that the gaze tracker classifies correctly which object the subject is looking at.

**Figure 4.8:** Object 1 to the left in the image, object 2 in the middle and object 3 to the right.

| | Head Pose 1 | Head Pose 2 | Head Pose 3 |
|---|---|---|---|
| Gaze 1 | 7.57 ° | 7.14 ° | 12.90 ° |
| Gaze 2 | 12.57 ° | 9.91 ° | 11.80 ° |
| Gaze 3 | 13.44 ° | 11.70 ° | 8.58 ° |

**Table 4.13:** Average error, in degrees, for the different head poses and gaze directions when the objects are 1 m from the camera and the subject is 1 m behind the objects in a dark room.

| Best | Worst |
|---|---|
| 25.94 cm | 49.51 cm |

**Table 4.14:** Minimum distance for the best and worst case.

## Camera-Object: 100 cm, Object-Subject: 150 cm, dark room

Table 4.15 below shows the average error for the different combinations of head poses and gaze directions when placing the objects 100 cm from the camera and positioning the subject 150 cm behind the objects in a dark room.

Table 4.16 below shows the worst and best case scenario for the minimum distance the objects can be from each other so that there is a greater than 50% chance that the gaze tracker classifies correctly which object the subject is looking at.

**Figure 4.9:** Object 1 to the left in the image, object 2 in the middle and object 3 to the right.

| | Head Pose 1 | Head Pose 2 | Head Pose 3 |
|---|---|---|---|
| Gaze 1 | 11.86 ° | 7.32 ° | 7.29 ° |
| Gaze 2 | 17.21 ° | 11.77 ° | 9.04 ° |
| Gaze 3 | 11.79 ° | 18.57 ° | 13.34 ° |

**Table 4.15:** Average error, in degrees, for the different head poses and gaze directions when the objects are 1 m from the camera and the subject is 1.5 m behind the objects in a dark room.

| Best | Worst |
|---|---|
| 39.17 cm | 92.92 cm |

**Table 4.16:** Minimum distance for the best and worst case.

## Camera-Object: 100 cm, Object-Subject: 200 cm, dark room

Table 4.17 below shows the average error for the different combinations of head poses and gaze directions when placing the objects 100 cm from the camera and positioning the subject 200 cm behind the objects in a dark room.

Table 4.18 below shows the worst and best case scenario for the minimum distance the objects can be from each other so that there is a greater than 50% chance that the gaze tracker classifies correctly which object the subject is looking at.

**Figure 4.10:** Object 1 to the left in the image, object 2 in the middle and object 3 to the right.

| | Head Pose 1 | Head Pose 2 | Head Pose 3 |
|---|---|---|---|
| Gaze 1 | 10.52 ° | 8.27 ° | 3.90 ° |
| Gaze 2 | 16.51 ° | 12.51 ° | 6.67 ° |
| Gaze 3 | 21.74 ° | 2.53 ° | 4.97 ° |

**Table 4.17:** Average error, in degrees, for the different head poses and gaze directions when the objects are 1 m from the camera and the subject is 2 m behind the objects in a dark room.

| Best | Worst |
|---|---|
| 35.10 cm | 160.99 cm |

**Table 4.18:** Minimum distance for the best and worst case.

### 4.2.3 Objects Placed Vertically

#### Camera-Object: 150 cm, Object-Subject: 150 cm, light room

Table 4.19 below shows the average error for the different combinations of head poses and gaze directions when placing the objects, vertically, 150 cm from the camera and positioning the subject 150 cm behind the objects in a light room.

Table 4.20 below shows the worst and best case scenario for the minimum distance the objects can be from each other so that there is a greater than 50% chance that the gaze tracker classifies correctly which object the subject is looking at.

The room where this experiment was carried out was one of the computer rooms in the basement of the E-building and we were rather limited in our setup. We therefore opted to only use two objects.

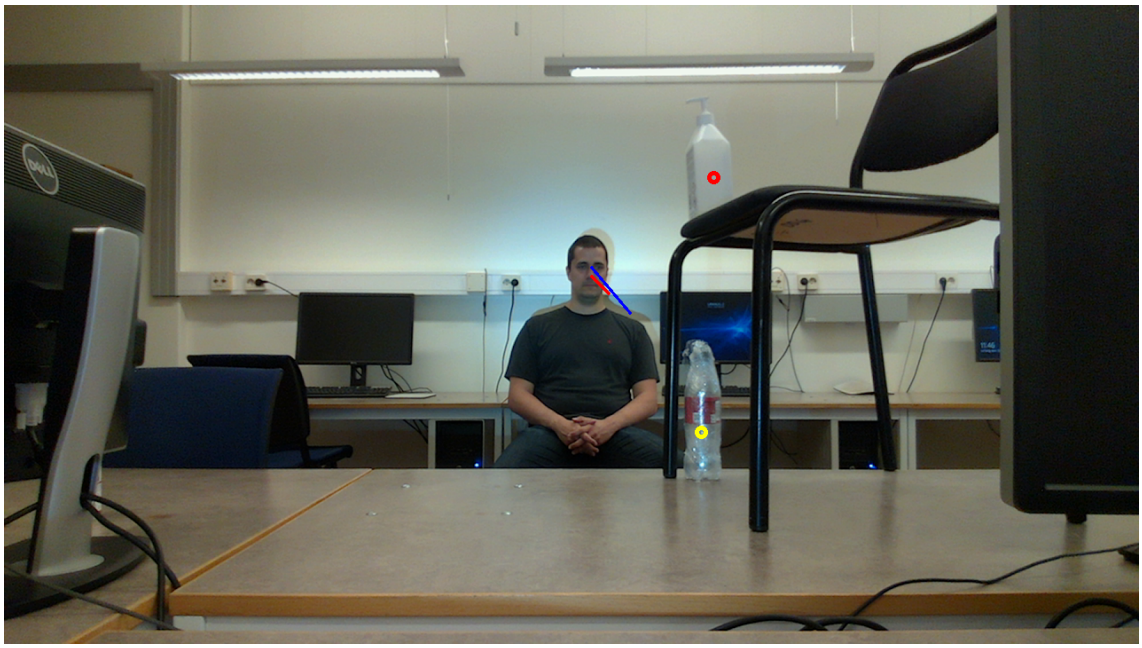


**Figure 4.11:** Object 1 is standing on the table and object 2 is the bottle on the chair.

| | Head Pose 1 | Head Pose 2 |
|---|---|---|
| Gaze 1 | 3.47° | 9.22° |
| Gaze 2 | 7.54° | 7.69° |

**Table 4.19:** Average error, in degrees

| Best | Worst |
|---|---|
| 18.19 cm | 48.68 cm |

**Table 4.20:** Minimum distance for the best and worst case.

## Camera-Object: 100 cm, Object-Subject: 100 cm, dark room

Table 4.21 below shows the average error for the different combinations of head poses and gaze directions when placing the objects, vertically, 100 cm from the camera and positioning the subject 100 cm behind the objects in a dark room.

Table 4.22 below shows the worst and best case scenario for the minimum distance the objects can be from each other so that there is a greater than 50% chance that the gaze tracker classifies correctly which object the subject is looking at.

The subsequent experiments were carried out on the fourth floor on the E-building. Here we had more accessories to work with so we were able to incorporate all three objects in each experiment. Because of the close proximity between objects 2 and 3 we chose to only evaluate two head poses. Facing object 1 and facing object 3.

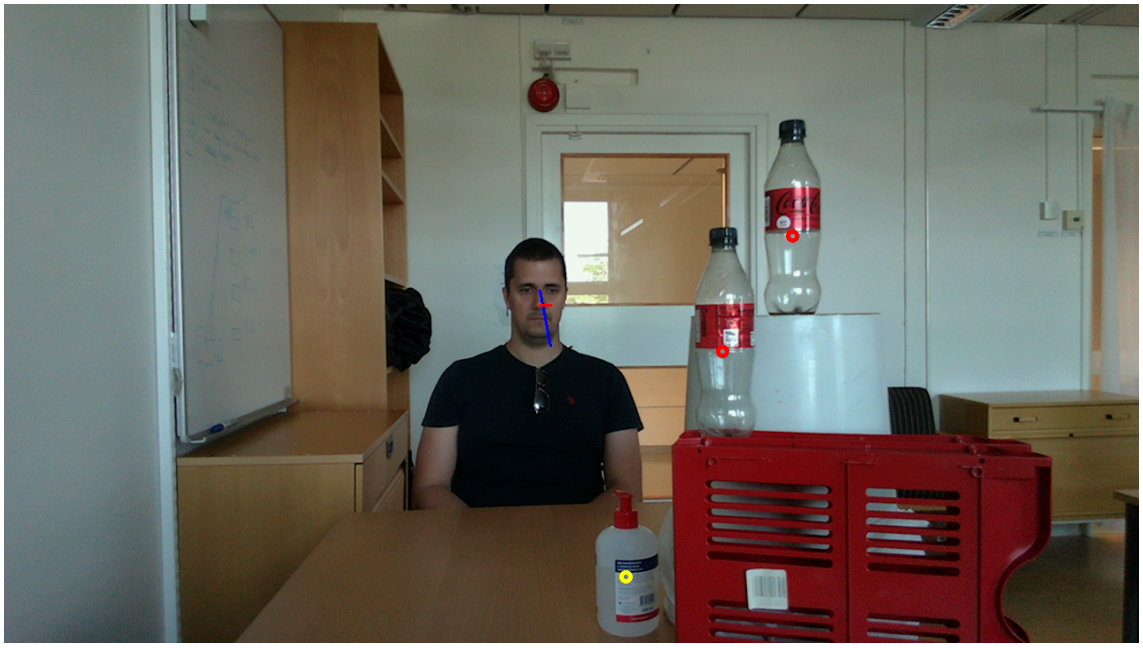


**Figure 4.12:** Object 1 is standing on the table and object 2 is the middle bottle and object 3 is the one on top of the trash can.

| | Head Pose 1 | Head Pose 2 |
|---|---|---|
| Gaze 1 | 11.88 ° | 10.24 ° |
| Gaze 2 | 14.14 ° | 14.41 ° |
| Gaze 3 | 6.27 ° | 7.13 ° |

**Table 4.21:** Average error, in degrees.

| Best | Worst |
|---|---|
| 21.96 cm | 51.40 cm |

**Table 4.22:** Minimum distance for the best and worst case.

## Camera-Object: 100 cm, Object-Subject: 150 cm, dark room

Table 4.23 below shows the average error for the different combinations of head poses and gaze directions when placing the objects, vertically, 100 cm from the camera and positioning the subject 150 cm behind the objects in a dark room.

Table 4.24 below shows the worst and best case scenario for the minimum distance the objects can be from each other so that there is a greater than 50% chance that the gaze tracker classifies correctly which object the subject is looking at.

**Figure 4.13:** Object 1 is standing on the table and object 2 is the middle bottle and object 3 is the one on top of the trash can.

| | Head Pose 1 | Head Pose 2 |
|---|---|---|
| Gaze 1 | 14.09 º | 15.55 º |
| Gaze 2 | 15.92 º | 13.82 º |
| Gaze 3 | 9.90 º | 13.60 º |

**Table 4.23:** Average error, in degrees.

| Best | Worst |
|---|---|
| 52.37 cm | 82.57 cm |

**Table 4.24:** Minimum distance for the best and worst case.

## Camera-Object: 100 cm, Object-Subject: 200 cm, dark room

Table 4.25 below shows the average error for the different combinations of head poses and gaze directions when placing the objects, vertically, 100 cm from the camera and positioning the subject 200 cm behind the objects in a dark room.

Table 4.26 below shows the worst and best case scenario for the minimum distance the objects can be from each other so that there is a greater than 50% chance that the gaze tracker classifies correctly which object the subject is looking at.

**Figure 4.14:** Object 1 is standing on the table and object 2 is the middle bottle and object 3 is the one on top of the trash can.

| | Head Pose 1 | Head Pose 2 |
|---|---|---|
| Gaze 1 | 8.09° | 6.88° |
| Gaze 2 | 12.65° | 12.43° |
| Gaze 3 | 13.55 ° | 13.00 ° |

**Table 4.25:** Average error, in degrees

| Best | Worst |
|---|---|
| 48.24 cm | 96.43 cm |

**Table 4.26:** Minimum distance for the best and worst case.

## 4.3 Discussion

As can be seen from the results the gaze tracker seems to perform the best when the gaze direction is close to the head pose direction. It also seems like the gaze tracker works the best when the subject is positioned at a medium range away from the objects. If the subject is too far away the CNNs seem to have a problem to find correct coordinates for the different facial landmarks, probably due to image resolution. If the subject is too close to the objects some combinations of gaze- and head pose directions become unnatural resulting in bad accuracy for the gaze tracker.

For the experiments with the objects placed vertically it seems like the error of the gaze tracker is lower for objects that are positioned further down. This is probably because the offset of the eye mid is set too high for the subject performing the experiments. How to find more optimal parameters is something that can be looked deeper into in future studies.

The light conditions do not seem to affect the gaze tracker as much as expected. There is no big difference between the experiments in the darker and the lighter room.

Also both the head pose and the gaze vector seem to have a tendency to point slightly to the subject's right. We were able to narrow this problem down to mainly stemming from the eye gaze component. Days were spent trying to find what caused this error but, due to time limitations, we were forced to move on, leaving the problem unsolved for now.

Looking at the result it is clear that placing the objects roughly 27 cm apart is only viable in the best cases. The worst result we got was on the experiment with the objects placed 100 cm from the camera and the subject positioned 200 cm from the objects, in the dark room. In this experiment with an average angular error of 21.74°, 161 cm would be needed between the objects for them to be classified correctly most of the time. This was for the head oriented to the far left object in the image with the gaze oriented towards the far right object. This clearly shows the effect of the stationary error that angles the gaze towards the subjects right (which is left in the image from the viewers perspective). This is in line with our own experiences doing the experiments where we saw a fairly high number of false positives in many of the more unnatural use cases.

Using this gaze tracker for human-robot interaction has some limitations. For example, the fact that we only track the gaze of the first face found by the MTCNN can become a problem when there are multiple people in the image. It is possible that the first face that the MTCNN finds does not belong to the subject that is supposed to use the gaze tracker. This could be solved by training a neural network to detect a specific subject's face. That way one could find the correct subject we want to track the gaze of and crop out their face and send that image to the MTCNN.

Another problem that could occur when it comes to human-robot interaction is if we want to have a robot interact with an object a subject is looking at and there are multiple objects that are too close to each other. As could be seen in the results Section, the objects can not be too close to each other if we want to classify correctly which object the subject is looking at.

# Chapter 5
# Conclusions

The gaze tracker can, to some extent, be used to control a robot system, as proven by the experiments described in chapter 4. The gaze tracker works relatively well when the subject is close enough to the camera and the object the subject is looking at is not too close to the subject. This seems reasonable as having the object too close tends to force slightly unnatural positions where the difference between head pose and gaze is large. Also the light conditions tested in the performed experiments affected the gaze tracker far less than we expected.

The eye model presented in this thesis could most likely be improved. To just calculate the difference in the eye mid and the pupil and drawing a vector between these two points feels a bit simplistic, and using a more sophisticated approach could probably improve the gaze tracker quite a bit.

For use in future research applications we recommend the following improvements:

- Allow for bigger head movement in the subject. As it is now the subject cannot move their head too much from the initial starting position. This choice was made in order to improve the performance.

- Fix the stationary error in the gaze estimation. As previously mentioned we devoted a fair amount of time to trying to fix this but were unsuccessful within the time constraints we were bound by.

- Support multiple subjects. The gaze tracker can currently only handle one subject at a time but it is fairly simple to extend the code to handle multiple subjects. The main difficulty comes from making sure the image cropping needed for the performance improvements is handled correctly for each subject.

- The gaze tracker currently makes use of several calibration parameters. The default parameters were calculated with the use of the Columbia Gaze Dataset. Even though

this dataset is fairly diverse it is probably not representative of the entire human population as it only contains data from 56 people. Therefore some type of automated calibration sequence might be useful.

We intend to implement as much of this as time allows before handing over the final code version.

While the accuracy of the tracker is not spectacular, we are satisfied with our results. The combination of gaze tracking and spatial awareness that the realsense camera offers is not often found in commercial eye trackers. Tobii [34] which is the leading company in commercial gaze tracking currently offers two different solutions for gaze tracking. The first is gaze tracking as a computer interface where the gaze is used as an input device similar to a mouse. This is not very interesting in the context of HRI. The second is their wearable Tobii Pro Glasses 3. This is the product they market towards research. While this is a very different technical solution and it is a little like comparing apples to oranges we feel it is important to note that this solution does not offer any spatial awareness. The glasses have a built in regular RGB camera. This does not easily allow for the estimation of where other objects are in relation to the wearer in the same way our gaze tracker does. This limits the ways in which you can use it for HRI. It is for instance difficult to tell the robot to go to a specific point in the room if you have no easy way of knowing where that point is located.

While the accuracy of the tracker could of course be higher we still feel that it offers decent accuracy in natural use cases where eye gaze and head pose align. We therefore hope it will be a valuable tool in future research and look forward to seeing the results.

# References


[1] Source code repository: MTCNN. `https://github.com/ipazc/mtcnn`. Accessed: 2021-02.

[2] Source code repository: MTCNN-Tensorflow. `https://github.com/AITTSMD/MTCNN-Tensorflow`. Accessed: 2021-02.

[3] Source code repository: tf-MTCNN. `https://github.com/blaueck/tf-mtcnn`. Accessed: 2021-06.

[4] Henny Admoni, Anca Dragan, Siddhartha S. Srinivasa, and Brian Scassellati. Deliberate delays during robot-to-human handovers improve compliance with gaze communication. In *2014 9th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 49–56, 2014.

[5] Andrew P Bayliss Alexandra Frischen and Steven P Tipper. Gaze cueing of attention: visual attention, social cognition and individual differences. *Psychological Bulletin*, 133(4):694–724, 2007.

[6] R.S. Allison, M. Eizenman, and B.S.K. Cheung. Combined head and eye tracking system for dynamic testing of the vestibular system. *IEEE Transactions on Biomedical Engineering*, 43(11):1073–1082, 1996.

[7] Yihua Cheng, Yiwei Bao, and Feng Lu. Puregaze: Purifying gaze feature for generalizable gaze estimation, 2021.

[8] Yihua Cheng, Haofei Wang, Yiwei Bao, and Feng Lu. Appearance-based gaze estimation with deep learning: A review and benchmark. *arXiv preprint arXiv:2104.12668*, 2021.

[9] Haoping Deng and Wangjiang Zhu. Monocular free-head 3d gaze tracking with deep learning and geometry constraints. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3162–3171, 2017.

[10] Philipe A. Dias, Damiano Malafronte, Henry Medeiros, and Francesca Odone. Gaze estimation for assisted living environments. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 279–288, 2020.

[11] Maria K. Eckstein, Belén Guerra-Carrillo, Alison T. Miller Singley, and Silvia A. Bunge. Beyond eye gaze: What else can eyetracking reveal about cognition and cognitive development? *Developmental Cognitive Neuroscience*, 25:69–91, 2017. Sensitive periods across development.

[12] Andrew Gee and Roberto Cipolla. Determining the gaze of faces in images. *Image and Vision Computing*, 12(10):639–647, 1994.

[13] Aline Godfroid, Frank Boers, and Alex Housen. An eye for words: Gauging the role of attention in incidental l2 vocabulary acquisition by means of eye-tracking. *Studies in Second Language Acquisition*, 35(3):483–517, 2013.

[14] Kaggle Inc. Facial keypoints detection. `https://www.kaggle.com/c/facial-keypoints-detection`. Accessed: 2021-04.

[15] Intel. Intel® realsense™ sdk 2.0. `https://github.com/IntelRealSense/librealsense`. Accessed: 2021-06.

[16] Intel. Python wrapper for intel realsense sdk 2.0. `https://pypi.org/project/pyrealsense2/`. Accessed: 2021-06.

[17] L. Itti, N. Dhavale, and F. Pighin. Realistic avatar eye and head animation using a neurobiological model of visual attention. In *Proceedings of the SPIE - The International Society for Optical Engineering*, volume 5200, pages 64 – 78, Dept. of Comput. Sci., Univ. of Southern California, Los Angeles, CA, USA, 2003.

[18] Qiang Ji and Xiaojie Yang. Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. *Real-Time Imaging*, 8(5):357–377, 2002.

[19] Jung-Hwa Kim and Jin-Woo Jeong. Gaze estimation in the dark with generative adversarial networks. In *ACM Symposium on Eye Tracking Research and Applications*, ETRA '20 Adjunct, New York, NY, USA, 2020. Association for Computing Machinery.

[20] Davis King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 07 2009.

[21] Kyle Krafka, Aditya Khosla, Petr Kellnhofer, Harini Kannan, Suchendra Bhandarkar, Wojciech Matusik, and Antonio Torralba. Eye tracking for everyone. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2176–2184, 2016.

[22] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81, 02 2009.

[23] Peng Li, Xuebin Hou, Xingguang Duan, Hiuman Yip, Guoli Song, and Yunhui Liu. Appearance-based gaze estimator for natural interaction control of surgical robots. *IEEE Access*, 7:25095–25110, 2019.

[24] Rosanne Liu, J. Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alexander Sergeev, and J. Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *NeurIPS*, 2018.

[25] Oliver Lorenz and Ulrike Thomas. Real time eye gaze tracking system using cnn-based facial features for human attention measurement. *14th International Conference on Computer Vision Theory and Applications*, pages 598–606, 01 2019.

[26] S. Milborrow, J. Morkel, and F. Nicolls. The MUCT Landmarked Face Database. *Pattern Recognition Association of South Africa*, 2010. `http://www.milbo.org/muct`.

[27] Seonwook Park, Adrian Spurr, and Otmar Hilliges. *Deep Pictorial Gaze Estimation*, pages 741–757. Springer International Publishing, 09 2018.

[28] Lorenzo Piccardi, Basilio Noris, Olivier Barbey, Aude Billard, Giuseppina Schiavone, Flavio Keller, and Claes von Hofsten. Wearcam: A head mounted wireless camera for monitoring gaze attention and for the diagnosis of developmental disorders in young children. In *RO-MAN 2007 - The 16th IEEE International Symposium on Robot and Human Interactive Communication*, pages 594–598, 2007.

[29] Akshay Rangesh, Bowen Zhang, and Mohan M. Trivedi. Driver gaze estimation in the real world: Overcoming the eyeglass challenge. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1054–1059, 2020.

[30] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[31] B.A. Smith, Q. Yin, S.K. Feiner, and S.K. Nayar. Gaze Locking: Passive Eye Contact Detection for Human?Object Interaction. In *ACM Symposium on User Interface Software and Technology (UIST)*, pages 271–280, Oct 2013.

[32] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep convolutional network cascade for facial point detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3476–3483, 2013.

[33] Haofei Wang, Xujiong Dong, Zhaokang Chen, and Bertram E. Shi. Hybrid gaze/eeg brain computer interface for robot arm control on a pick and place task. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1476–1479, 2015.

[34] Tobii pro. `https://www.tobiipro.com/`. Accessed: 2021-05-24.

[35] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[36] Laurence R Young and David Sheena. Survey of eye movement recording methods. *Behavior research methods & instrumentation*, 7(5):397–429, 1975.

[37] Zehui Yu, Xiehe Huang, Xiubao Zhang, Haifeng Shen, Qun Li, Weihong Deng, Jian Tang, Yi Yang, and Jieping Ye. A multi-modal approach for driver gaze prediction to remove identity bias. In *Proceedings of the 2020 International Conference on Multimodal Interaction*, ICMI '20, page 768–776, New York, NY, USA, 2020. Association for Computing Machinery.

[38] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23, 04 2016.

[39] Xucong Zhang, Yusuke Sugano, and A. Bulling. Evaluation of appearance-based methods and implications for gaze-based applications. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019.

[40] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. Appearance-based gaze estimation in the wild. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4511–4520, 2015.

[41] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. MPIIGaze: Real-world dataset and deep appearance-based gaze estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP, 11 2017.

[42] Is pupil size calculations possible with tobii eye trackers? `https://www.tobiipro.com/learn-and-support/learn/eye-tracking-essentials/is-pupil-size-calculations-possible-with-tobii-eye-trackers/`. Accessed: 2021-05-24.

# Chapter 6
# Appendix

## 6.1 Eye Gaze CNN

The CNN used for the eye gaze feature extraction has the structure detailed below.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 32, 32, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 32, 32, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 16, 16, 32)        128
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
batch_normalization_1 (Batch (None, 8, 8, 64)          256
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
batch_normalization_2 (Batch (None, 4, 4, 128)         512
_________________________________________________________________
```

```
conv2d_3 (Conv2D)              (None, 4, 4, 128)         147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
batch_normalization_3 (Batch (None, 2, 2, 128)         512
_________________________________________________________________
dense (Dense)                  (None, 2, 2, 512)         66048
_________________________________________________________________
dense_1 (Dense)                (None, 2, 2, 512)         262656
_________________________________________________________________
flatten (Flatten)              (None, 2048)              0
_________________________________________________________________
dense_2 (Dense)                (None, 6)                 12294
=================================================================
Total params: 582,662
Trainable params: 581,958
Non-trainable params: 704
_________________________________________________________________
```

## 6.2 Division of Labor - Tables

The following tables outline the division of labor used in this thesis work. Tasks that were done together have both authors listed as lead. In those cases names are listed alphabetically by first name. Some part of both the implementation and the report were cut as they either did not work or were not needed. In those cases it is marked as (cut) in the table.

| Task | Lead | Assist |
|---|---|---|
| Configuration of andaconda env. and remote desktop on exjobb14. | Daniel | |
| Configuration and training of the initial MTCNN. | Daniel, Josef | |
| Evaluation of the inital MTCNN. | Daniel, Josef | |
| Researching alternative MTCNN implementations. | Daniel | |
| Implementation and evaluation of the second MTCNN. | Daniel, Josef | |
| Test implementation of the KCF (cut) | Josef | |
| Head pose estimation, symmetry method (cut). | Daniel, Josef | |
| Head pose estimation, 3d-method. | Daniel, Josef | |
| CSV-data processing for eye gaze training. | Daniel | Josef |
| Image augmentation for eye gaze training. | Josef | Daniel |
| Eye gaze CNN design. | Daniel, Josef | |
| Eye gaze model improvements. | Daniel | |
| Final gaze vector example implementation | Daniel, Josef | |
| Test implementation of the third MTCNN. | Josef | |
| Performance optimization. | Daniel | |
| Columbia gaze data processing. | Josef | |
| Implementation of script for calibration parameter estimation. | Josef | Daniel |
| YOLO test implementation. | Josef | |
| YOLO integration. | Daniel | |
| Realsense camera test implementation. | Josef | |
| Object detection using pyrealsense2 library. | Daniel | |
| Evaluation and troubleshooting. | Daniel, Josef | |
| Experiments. | Daniel, Josef | |
| Experiment evaluation. | Daniel, Josef | |

**Table 6.1:** Division of labor for the gaze tracker implementation.

| Task | Lead | Assist |
|---|---|---|
| Abstract | Josef | Daniel |
| The Field of Gaze Tracking | Daniel | Josef |
| Goal | Daniel | |
| Division of Labor | Daniel, Josef | |
| Overview | Daniel | |
| Kernelized Correlation Filter (cut) | Josef | |
| Artificial Neural Networks | Josef | |
| Decision Tree Learning | Josef | |
| The facial normal | Daniel | |
| Eye gaze | Daniel | |
| Related Work | Daniel | Josef |
| Face Detection and Facial Landmarks | Daniel, Josef | |
| Head Pose Estimation | Josef | |
| Eye Gaze Estimation | Daniel | Josef |
| Putting it together - Gaze Vector | Daniel | |
| Object Detection | Daniel | Josef |
| Parameter Optimization | Josef | |
| Performance Optimization | Daniel | |
| Experiments | Josef | |
| Results | Daniel, Josef | |
| Discussion | Daniel, Josef | |
| Conclusions | Daniel, Josef | |

**Table 6.2:** Division of labor for writing the report.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2021-06-21

**EXAMENSARBETE** Evaluation of an RGB-D data based Gaze Tracker for Human-Robot Interaction
**STUDENTER** Josef Abdul Al, Daniel Siljegovic Persson
**HANDLEDARE** Elin Anna Topp (LTH)
**EXAMINATOR** Jacek Malec (LTH)

# Finna den mänskliga blicken med hjälp av AI

POPULÄRVETENSKAPLIG SAMMANFATTNING **Josef Abdul Al, Daniel Siljegovic Persson**

Med hjälp av en djupseende kamera lyckades vi konstruera en blickspårare som kan förutspå vilket objekt i rummet en person tittar på. Vi använde även blickspåraren till att styra en robot.

Blickspårning är ett stort forskningsområde med tillämpningar i allt ifrån säkerhetssystem till att studera de kognitiva processerna hos små barn.

I vårt examensarbete försökte vi konstruera en blickspårare som är tillräckligt exakt för att kunna förutspå vilket objekt en person tittar på. Tanken var att den skulle användas vid framtida forskning inom människa-robot interaktion. Av denna anledning så konstruerades en enkel demonstration där en robot styrs att rotera antingen åt vänster eller höger beroende på vilket håll en person tittar.

Blickspårningen är en process i flera steg där varje steg leder in till nästa. Det första steget tar en vanlig färgbild och identifierar huvudets position samt ögonen, näsan och munnen. Detta görs med hjälp av ett neuralt faltningsnät. Det är en typ av neuralt nät som lämpar sig väl då information vill utvinnas ur bilder. Steg två är att ta punkterna i ansiktet och använda dessa för att beräkna huvudets orientering. Det tredje och sista steget är att använda ytterligare ett neuralt faltningsnät för att identifiera pupillen samt inre och yttre ögonvrå. Med hjälp av de punkterna i ögat så kan ögats riktning uppskattas. Genom att kombinera huvudets position och riktningen på ögonen så kan personens blick beräknas.

Djupsensorn i kameran gör det möjligt att bestämma punkter i rummet för både ansikte och de olika objekten. Genom att beräkna vinkeln mellan blickriktningen och vektorerna från ansiktet till de olika objekten så går det att uppskatta vilket objekt personen tittar på.

Vi gjorde en serie experiment där vi filmade en sekvens med kameran där en person med varierande huvudposition tittar på olika objekt. För vissa positioner och avstånd lyckades vi få avvikelser så små som 3.5°.

Vår förhoppning är att blickspåraren blir ett värdefullt redskap i den framtida forskningen.