Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

Testing the accuracy of the neural network based eye gaze estimation using an eye tracker

Bachelor Thesis

2024
Matej Budoš

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

Testing the accuracy of the neural network based eye gaze estimation using an eye tracker

Bachelor Thesis

Study Programme:	Applied Informatics
Field of Study:	Computer Science
Department:	Department of Computer Science
Supervisor:	prof. Ing. Igor Farkaš, Dr.

Bratislava, 2024
Matej Budoš

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Matej Budoš
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Testing the accuracy of the neural network based eye gaze estimation using an eye tracker
Testovanie presnosti odhadu očného pohľadu založeného na neurónovej sieti pomocou sledovača očí

Anotácia: Plynulosť interakcie medzi človekom a robotom výrazne závisí od schopnosti robota sledovať pohľad očí človeka, ktorý je dobrým indikátorom jeho úmyslov. Takýto systém môže využiť priamo vstupy z kamier v očiach robota, ktoré snímajú scénu a vedia vyextrahovať hlavu a oči človeka sediaceho pred robotom.

Cieľ: 1. Nainštalujte a otestujte fungujúci systém založený na neurónových sieťach na odhad polohy ľudského pohľadu (Herashchenko, 2023) pomocou webovej kamery.
2. Implementujte metódu na mapovanie pozícií očí predpovedaných systémom na súradnice obrazovky.
3. S obrazovkou umiestnenou vertikálne a/alebo horizontálne pred účastníkom otestujte presnosť odhadovaných súradníc obrazovky počas behaviorálneho experimentu.

Literatúra: Admoni H., Scassellati B. (2017) Social Eye Gaze in Human-Robot Interaction: A Review. Journal of Human-Robot Interaction, 25–63. https://doi 10.5898/JHRI.6.1.Admoni
Herashchenko D. (2023). Robot reading human head pose and gaze direction. Bakalárska práca. FMFI UK Bratislava

Vedúci: prof. Ing. Igor Farkaš, Dr.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 12.10.2023

Dátum schválenia: 16.10.2023 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Matej Budoš
Study programme:	Applied Computer Science (Single degree study, bachelor I. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Bachelor´s thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Testing the accuracy of the neural network based eye gaze estimation using an eye tracker

Annotation: The smoothness of the human-robot interaction greatly depends on the robot's ability to follow the human eye gaze, which is a good indicator of human intentions. Such a system can take an input directly from the cameras in the robot's eyes, which capture the scene and can extract the head and eyes of a person sitting in front of the robot.

Aim:
1. Install and test a working system based on neural networks for estimation of human gaze position (Herashchenko, 2023) using a web camera.
2. Implement a method to map the eye gaze positions predicted by the system to the screen coordinates.
3. With the screen placed vertically and/or horizontally in front of the participant, test the accuracy of the predicted screen coordinates during a behavioral experiment.

Literature:
Herashchenko D. (2023). Robot reading human head pose and gaze direction. Bachelor's thesis. FMFI UK Bratislava
Admoni H., Scassellati B. (2017) Social Eye Gaze in Human-Robot Interaction: A Review. Journal of Human-Robot Interaction, 25–63. https://doi 10.5898/JHRI.6.1.Admoni

Supervisor:	prof. Ing. Igor Farkaš, Dr.
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	doc. RNDr. Tatiana Jajcayová, PhD.
Assigned:	12.10.2023
Approved:	16.10.2023

doc. RNDr. Damas Gruska, PhD.
Guarantor of Study Programme

.....
Student

.....
Supervisor

Acknowledgments: I would like to thank my supervisor prof. Ing. Igor Farkaš, Dr. for invaluable advice, patience and time dedicated to our consultations.

Abstrakt

Táto práca skúma potenciál pre praktické využitie systému sledovania očí založeného na zábere tváre, pre aplikácie ako je sledovanie pohľadu na obrazovke alebo interakcia medzi človekom a robotom. Systém využíva konvolučnú neurónovú sieť na odhadnutie smeru pohľadu pomocou obrázkov očí z webovej kamery. Zatiaľ čo neurónová sieť vyzerá byť sľubná pri vyhodnotení na skutočných a syntetických obrázkoch, naše hodnotenie odhaľuje obmedzenia, ktoré bránia jeho používaniu v reálnom svete.

Identifikovali sme nezrovnalosti vo výstupoch siete a jeho náchylnosť na zmeny pozície používateľa napriek tomu, že poloha hlavy je jedným zo vstupov siete. Preskúmali sme techniky mapovania pohľadu, aby sme našli najvhodnejšiu metódu, a implementovali sme filtrovanie dát na zlepšenie celkového výkonu. Zatiaľ čo sledovanie očí na obrazovke umiestnenej zvisle pred používateľom bolo možné použiť na hrubý odhad pohľadu očí, celková chyba presnosti na horizontálnej obrazovke bola približne 15 cm.

Naše zistenia naznačujú, že táto neurónová sieť vyžaduje ďalší vývoj pre aplikáciu v reálnom svete, najmä v oblastiach robustnosti a konzistencie.

Kľúčové slová: sledovanie pohľadu, mapovanie pohľadu, kalibrácia, odhadnutie pohľadu na základe vzhľadu, Pupil Core

Abstract

This thesis explores the potential of applying appearance based eye tracking system for practical use like general eye tracking on screen or human-robot interaction. The systeml utilizes a convolutional neural network to predict gaze direction using web camera eye images. While the neural network shows promise on a combined real-world and synthetic dataset, our evaluation reveals limitations hindering real-world use.

We identified inconsistencies in the network's outputs and its susceptibility to user position variations despite head position being one of network's inputs. We explored gaze mapping techniques to find the most suitable method and implemented data filtering to improve overall performance. While eye tracking on screen placed in front of a user vertically could be used for rough estimate of an eye gaze, the overall accuracy error on the horizontal screen was around 15 cm.

Our findings suggest that this neural network requires further development for real-world applications, regrading the robustness and consistency.

Keywords: eye tracking, gaze mapping, calibration, appearance-based gaze estimation, Pupil Core

Contents

List of Figures

Introduction

Eye tracking is a technology that measures and records the movement and position of the eyes, providing valuable insights into visual attention, cognitive processes, and user interactions. By analyzing where and how long a person looks at different areas of a visual field, researchers can understand the patterns of gaze and attention. This technology has diverse applications, ranging from psychological studies and neuromarketing to enhancing user experience in virtual reality and optimizing website designs. In the realm of human-robot collaboration, eye tracking plays a crucial role by allowing robots to interpret and respond to human gaze patterns, improving coordination and efficiency in shared tasks. The data collected through eye tracking can inspire design improvements, educational tools, and even assistive technologies, making it a versatile tool in both research and practical applications.

This thesis builds upon the work of Dmytro Herashchenko who developed a neural network model that predicts the gaze direction in the form of pitch and yaw angles, without using any additional hardware apart from an RGB web camera. This research aims to explore different gaze mapping techniques, their implementations and comparison in terms of accuracy and robustness to find the most suitable method for eye tracking on screen using this system. To objectively assess the accuracy of different mapping approaches, used head mounted eye tracker Pupil Labs and compared the predicted gaze data with the ground truth points on screen. We can then visualize the strengths and weaknesses of each mapping technique using error heat maps.

Following the evaluation, integrate the most accurate mapping technique in an human-robot interaction scenario and thus test the real-world applicability of our model. This final stage will determine whether this approach of eye tracking is suitable for further experiments and development.

Chapter 1

Theoretical Background

This chapter provides an overview of various eye tracking technologies and methodologies. It covers both appearance-based systems and head-mounted eye tracking devices. The chapter also explores gaze mapping methods and error visualization.

1.1 Gaze estimation

1.1.1 Appearance-based method

Appearance-based system estimates where a person is looking by analyzing their eye images captured by a regular RGB camera. This method typically utilizes machine learning algorithms and/or computer vision techniques to interpret the raw pixel data and identify the gaze direction by recognizing specific features in the eye, such as pupil location and eyelid shape. The primary advantage of appearance-based eye tracking is its ability to function without the need for specialized hardware, relying solely on standard cameras. This includes consumer devices like laptops and smartphones making this method entirely non-intrusive and affordable for everyone. For instance, controlling cursor on the screen with eyes or tracking the gaze on websites for advertisement purposes. While still under development, appearance-based eye tracking holds promise for various applications in human-robot experiments (Jiaqi et al., 2019).

1.1.2 Feature-based method

On the other hand, feature-based estimation focuses on detecting and tracking specific anatomical features of the eye, such as the pupil, iris, and corneal reflections usually refereed to as glint, to determine the gaze direction. This method often involves using infrared light to create high-contrast images that make these features easier to identify and track accurately. Feature-based eye tracking is known for its precision and robustness, especially in controlled environments, making it a popular choice for applications

requiring high accuracy, such as scientific research and medical diagnostics. However, it typically requires specialized hardware, making these devices inaccessible for many consumer applications. As technology advances and hardware becomes more affordable and portable, the accessibility of feature-based eye tracking solutions is expected to increase, potentially expanding its application domains beyond traditional research and medical settings (Sheela and Radhika, 2020).

1.2 WebGazer

One of the most popular appearance-based system is WebGazer. It is an eye tracking library written in JavaScript that uses web cameras to infer the gaze locations in real time. It utilizes clmtrackr (Mathias, 2014) library to detect face landmarks in an image. The landmarks that form eye contours are then used to extract rectangles of the eyes that serve as an input to WebGazer, which then detects location of the pupil based on the eye features like iris being darker than sclera and its ellipsoid shape. Instead of mapping 2D features of the pupil, they represent each eye as a 6×10 image patch to capture more characteristics of the eyes. With more prepossessing, the result is 120-dimensional feature vector that serves as an input into linear regression algorithm.

The thing that sets this system apart from other appearance-based systems is its addition of self-calibration. Chen et al. (2001) show that there is correlation between the cursor and gaze during web browsing, especially in the regions of the page where the user is focused. It also suggests this relationship can be used for gaze mapping on screen coordinates. WebGazer utilizes these facts and alongside with regular calibration, refits the regression with additional cursor data that have a life span of 200ms. That means, in a scenario where cursor is idle, the mapping reverts back to the original state right after the calibration (Papoutsaki et al., 2016).

1.3 Pupil core

Pupil Core by Pupil Labs is a head-mounted eye tracking device, equipped with two IR cameras 1.1 to track movement of both eyes and world camera to capture user's field of view. This device uses the dark pupil technique and 3D model to accurately track eye movements in three dimensions. The dark pupil technique is essentially capturing eye movements using IR camera, creating strong contrast between pupil and iris. It also allows to precisely track pupil dilatation. Tracking using 3D model 1.2, although being less accurate than tracking using 2D gaze positions, retains it's accuracy during subtle head movements and device slippage.

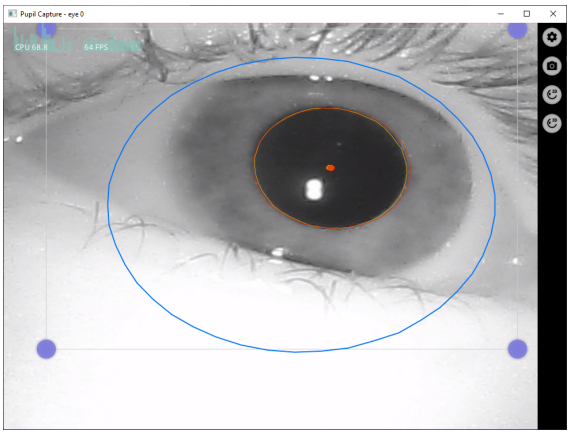


Figure 1.1: IR eye camera

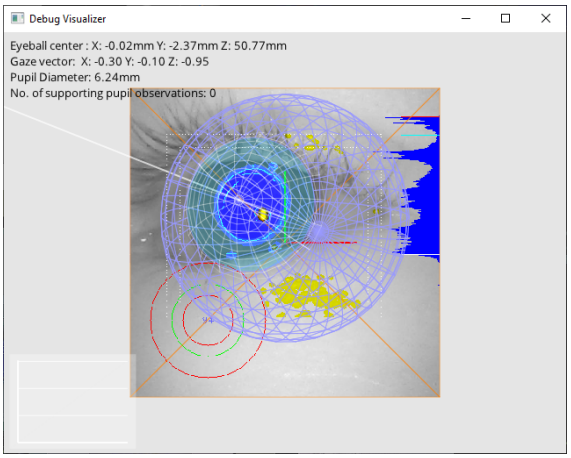


Figure 1.2: 3D eye model

1.3.1 Coordinate system

In Pupil Core, various coordinate systems play crucial roles in understanding and processing visual data. There are three main coordinate systems associated with each camera: 2D Image Space, 2D Normalized Space and 3D Camera Space.

The 2D Image Space is defined within the captured image itself, with its origin at the top left corner and measured in pixels. It includes lens distortion and has boundaries corresponding to the width and height of the image.

The 2D Normalized Space is similar to 2D Image Space but is normalized to unit dimensions, ranging from 0 to 1 for both x and y coordinates. It still incorporates lens distortion effects and maintains the same boundaries as the image space. The main difference is that the origin of the coordinates starts at bottom left corner.

Lastly, the 3D Camera Space is centered at the camera itself and with no lens distortion effects considered. It lacks predefined boundaries and utilizes a Cartesian coordinate system with x, y, and z axes. Notably, the eye model, which shares this space, describes the orientation of the eye relative to the camera. Cartesian coordinates describe the eye position, while spherical coordinates ϕ and θ, provide angular information to express gaze direction. (Kassner et al., 2014)

1.3.2 Pupil Core Network API Client

This Python module serves as a client interface for the Pupil Core Network API, designed to facilitate interactions with Pupil Core. The module offers a range of features, including initiating and stopping recordings, managing plugins like annotation plugin, fixation detector and blink detector. Its main asset is retrieving current eye positions and confidence in real time, providing a convenient way for developers and researchers to integrate Pupil Core functionalities into their Python applications or scripts.

1.3.3 Screen calibration

One of many calibration methods that Pupil Labs provides is screen calibration. Five points (in each corner and the center of the screen) are displayed in a short sequence, where for each point multiple corresponding eye coordinates are collected. The data is then filtered by confidence of pupil detection. Finally, the system fits the data with a polynomial regression and evaluates its accuracy. If needed, the process iterates, removing outliers and refitting the model to a cleaner data subset until a satisfactory accuracy is achieved (PupilLabs, 2024).

1.4 Herashchenko's neural network model

This appearance-based model has been trained on combination of two datasets. The Columbia gaze dataset (Smith et al., 2013) which consists of real-world data and the synthetic Metahuman dataset (Metahuman, 2024) created by Dmytro Herashchenko. The addition of the Metahuman dataset expanded overall training data and managed to improve the accuracy on a test data as shown in table 1.1.

1.4.1 Preprocessing

The whole process of eye tracking starts with preprocessing the image in form of face detection using Batch Face library (Zheng, 2018). This pretrained network locates face landmarks, more importantly eye locations, in order to extract the eyes from the image and use them as an input into the neural network. Another process that takes place is prediction of head positions using another pretraiend network called 6DRepNet (Hempel et al., 2022). These predicted head positions also serve as an input into the neural network however, are only used in the last layer.

1.4.2 The neural network

The network itself operates in a multiple-step process. First, it takes the cropped image of the user's eyes along with predicted head positions which are then fed into a convolutional neural network which is responsible for learning various features of the eyes, most likely, pupil positions and eye contours. In the final layer of the network, the head position information is incorporated into the prediction. This combined approach allows the model to take into account both the visual appearance of the eyes and the user's head pose. Finally, the network outputs the pitch and yaw angles of the user's gaze, essentially pinpointing the direction where they are looking.

Table 1.1: Neural network test dataset evaluation.

Trained / Tested on	Metahuman	Columbia gaze	Combined
Metahuman	MAE ≈ 0.59 $\sigma \approx 0.52$	MAE ≈ 8.12 $\sigma \approx 5.81$	MAE ≈ 1.9 $\sigma \approx 3.79$
Columbia gaze	MAE ≈ 8.42 $\sigma \approx 4.83$	MAE ≈ 1.93 $\sigma \approx 1.5$	MAE ≈ 7.55 $\sigma \approx 5.17$
Combined	MAE ≈ 0.65 $\sigma \approx 0.56$	MAE ≈ 2.88 $\sigma \approx 1.94$	MAE ≈ 1.04 $\sigma \approx 1.25$

1.4.3 Issues with the network

Despite its promising performance on test dataset containing both real-world and synthetic data, this network indicates a notable lack of robustness. The network demonstrates inconsistency when presented with the same images. The images shown in figure 1.3 are nine consecutive frames from camera in great light conditions and eye visibility. Despite this, the network often outputs different yaw and pitch values ranging between [-10.51, -6.49] and [11.58, 14.7] respectively.

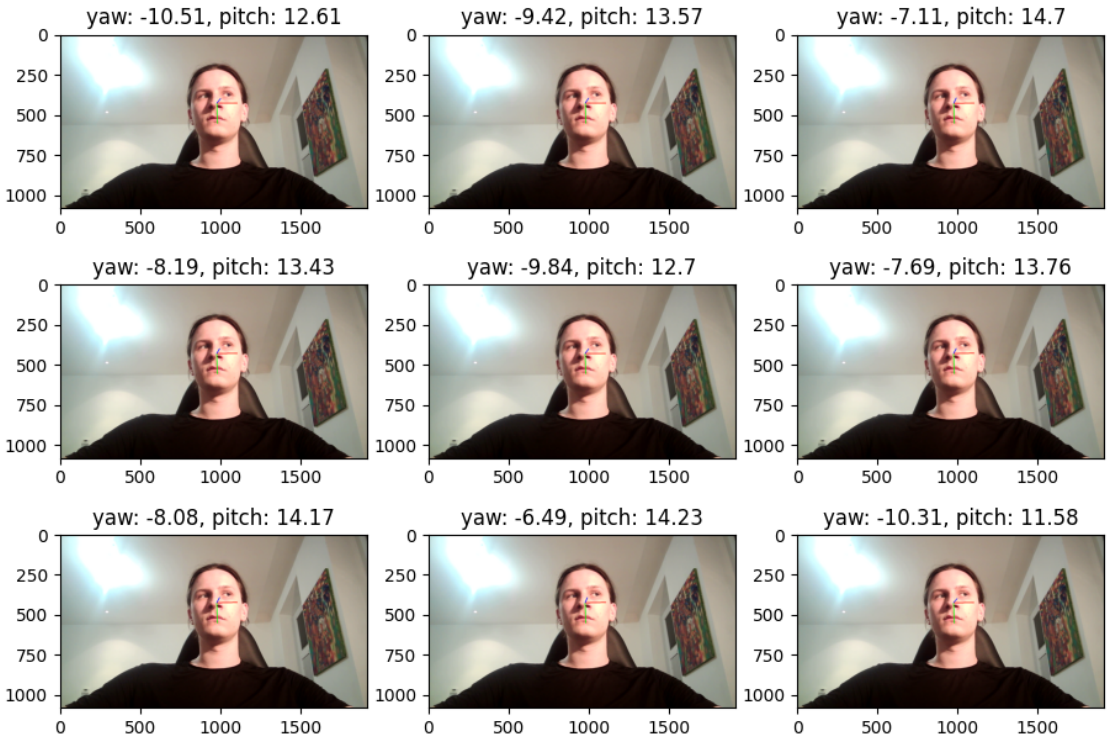
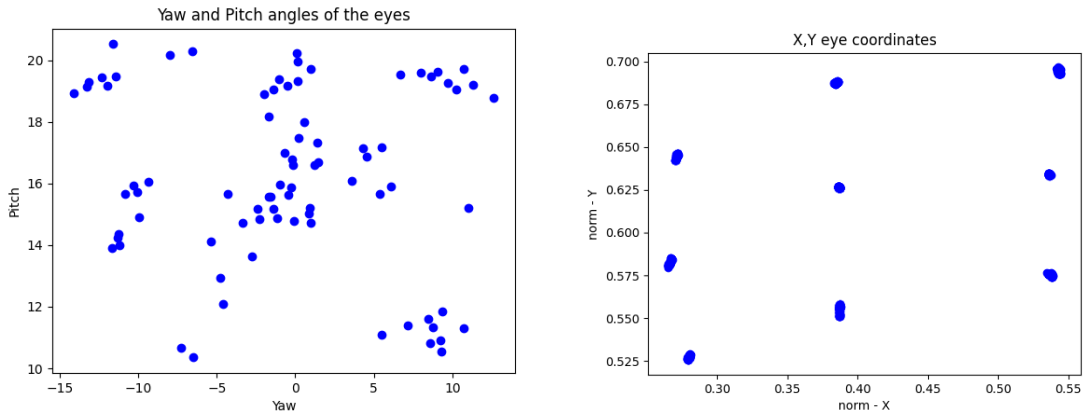


Figure 1.3: Yaw and Pitch values for 9 same images

Moreover, its reliability diminishes when the eyes undergo slight movements, suggesting that its accuracy is compromised in scenarios where precision is required. In a

test where user sits 50 cm from the screen, nine points are displayed successively and for each point eye locations are collected. These eye data should form nine different clusters, each corresponding to one point on the screen. Despite some signs of clustering, many of them tend to blend together as shown in figure 1.4a. For a reference, in figure 1.4b we can see results of the same test on Pupil Core eye tracking device. While it's not expected from the network to perform on same level as the eye tracker these results raise a concern about real life applications of the network.



(a) Neural network eye positions

(b) Pupil Core eye positions

Figure 1.4: Neural network and Pupil Core comparison

1.5 Regression

Regression is a method used to find correlation between independent and dependent variables. This relationship is expressed through a mathematical function which usually represents a line or a curve that describes this relationship. To find the best fitting function, the least squares method is used. This method evaluates the sum of the squared vertical distances between the line and the points and tries to minimize it.

1.5.1 Linear regression

Linear regression, being the simplest and most direct method, finds correlation between a target and one or more independent variables by simply fitting a line to the data. The main advantage is it's simplicity and computational efficiency. However, it is very sensitive to outliers and can easily overfit on a smaller dataset.

One way to prevent overfitting on a smaller data is to use linear ridge regression. By introducing a small bias to how line is fit to the data we can get long term predictions with significantly less variance.

1.5.2 Polynomial regression

This method is an extension of a linear regression where instead of fitting line to the data we fit n-th degree polynomial. This allows capturing non-linear relationships between variables while being able to fit wider range of data patterns. However, high degrees of polynomial can be prone to overfitting.

1.5.3 Outliers

Outliers or anomalies are data points that deviate significantly from the overall pattern observed in the data. These points can significantly influence the result of a regression analysis especially on smaller datasets. There are many techniques that remove outlier points but for our purposes we will be using Isolation Forest method from library scikit-learn (Pedregosa et al., 2011). This method allows us to locate different regions - forests of points, in our case groupings of yaw and pitch angles that correspond to certain coordinates. We can then remove outliers within that group of points to obtain subset of the dataset containing more accurate data without anomalies as shown in figure 1.5.

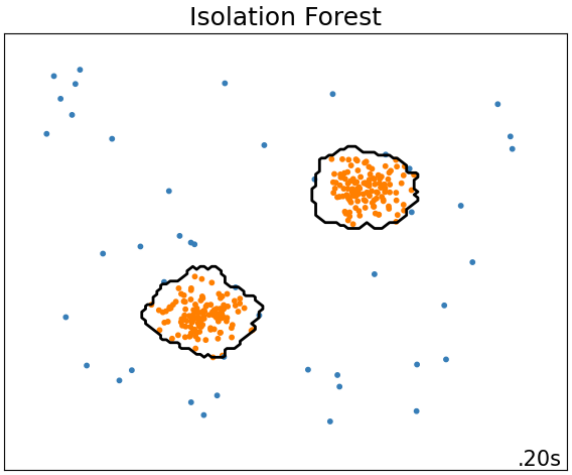


Figure 1.5: Outlier detection (Pedregosa et al., 2011)

1.6 Gaze mapping methods

1.6.1 Interpolation

Interpolation is a very straightforward method which does not require any additional hardware. During a simple calibration choreography, the user records their gaze direction for the outermost bounds of a planar surface (the screen) in four directions. Then by interpolating between maximum and minimum yaw and pitch we are able to

find corresponding x and y coordinates respectively. While being computationally very simple, this method does not take into account various factors like eye physiology, and the geometric relationship between the eye and the screen which makes this technique inferior to other ones mentioned before (Huang et al., 2014) .

1.6.2 2D regression

Regression is one of the most widely used technique when it comes to mapping gaze to screen or world coordinates in general. In context of screen mapping, various gaze locations and their corresponding ground truth points on screen are collected in order to fit a function that predicts screen coordinates from gaze direction. The choice of the regression type mainly depends on the complexity of relationship between dependent and independent variables.

1.6.3 3D geometric model

This method is generally used in head mounted headsets, where infrared cameras capture eye images and map various positions of pupil to 3D model of the eyeballs. One of the examples of mapping this 3D model to screen coordinates can be found in this paper (Mokatren et al., 2022). By adding two RGB cameras to capture both eyes, they acquired corneal image of the scene which could be then mapped to world image from front camera of the headset. The gaze point of the user was found by image matching between the corneal image and the world camera image using a mapping transformation between the cameras.

1.7 Error visualization

This visualization represents the error between predicted and ground truth points using a heat map. The process begins by calculating the Euclidean distance between each pair of ground truth and predicted points. These errors are then interpolated onto a grid spanning the domain of the data - in our case the coordinates of the screen. The grid is defined by dividing the x and y ranges into a specified number of intervals. The heat map is generated using the interpolated error values, with colors indicating the magnitude of error, ranging from cooler colors for lower errors to warmer colors for higher errors as shown in figure 1.6. This visualization provides a spatial understanding of the distribution and intensity of errors across the screen, aiding in the assessment of system performance and identifying regions where improvements may be necessary.

1.7.1 Conversion to centimeters

While pixels quantify the error much more precisely, it is more intuitive for readers to understand measurements in physical units like centimeters. To facilitate this, we convert the pixel error values into centimeters. This conversion is done by using the physical dimensions of the screen and its resolution. For our tests we will be using 68.58cm (27") screen with the resolution of 2560×1440. We can calculate px/cm (pixels per centimeter) by dividing diagonal of the screen in pixels by diagonal in centimeters, resulting in 42.82 px/cm.

Figure 1.6: Error heat map example

Chapter 2

Selection of mapping technique

In this chapter, we will be evaluating three distinct gaze mapping methods: interpolation, linear regression, and polynomial regression. To ensure a controlled environment that isolates the performance of each method, we will use the Pupil Core eye tracking headset instead of Heraschenko's system. The evaluation process will begin with the collection of both eye positions corresponding to calibration points on the screen. Pupil Core also measures confidence of the eye measurement in ranges [0,1]. During the calibration we will be collecting only the eye positions with the average confidence higher than 0.85. This data will then serve as the foundation for each method's attempt to predict gaze locations across a grid overlaid on the screen. Following the predictions, a evaluation will be conducted between the predicted screen coordinates and the actual, ground truth locations of the data points.

2.1 Calibration

We will be mapping the gaze positions on a 27" screen with the width 2560px and height 1440px, placed in front of user at the distance of 50cm. The amount of points displayed will depend on mapping technique. For interpolation, 4 points will be displayed for each of the fours sides of the screen as red points represent in figure 2.1. On the other hand, regression will require 9 points to be shown. Fours points on each side same as interpolation method with the addition of one point in the middle and four points in each of the corner represented by green points2.1. Because of this, we will increase the size of the point dataset and avoid the overfitting, thus improve overall accuracy. In both instances, 9 eye positions are taken for each point displayed, to ensure the robustness of the calibration in case of faulty eye measurement and to find exact corresponding eye position. Despite the fact that Pupil core is a very precise device, Isolation Forest technique will be used to remove 30% of outlier points to further improve precision of the mapping.

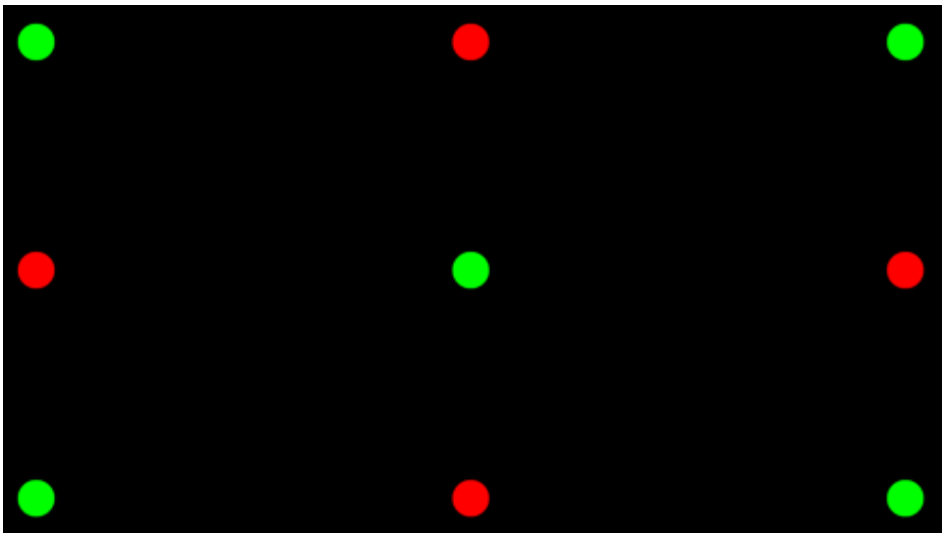


Figure 2.1: Calibration points

2.2 Data collection

After mapping the eye coordinates to the screen, a grid appears for the purpose of measuring the accuracy. This grid consists of five rows (288px) and eight columns (320px), with a point positioned in the center of each rectangle. In centimeters, the rectangle height is 6.75cm and width 7.5cm. These points will appear randomly one by one and for each we will collect 50 predicted screen positions. This data will be saved and used to evaluate the precision of the mapping technique.

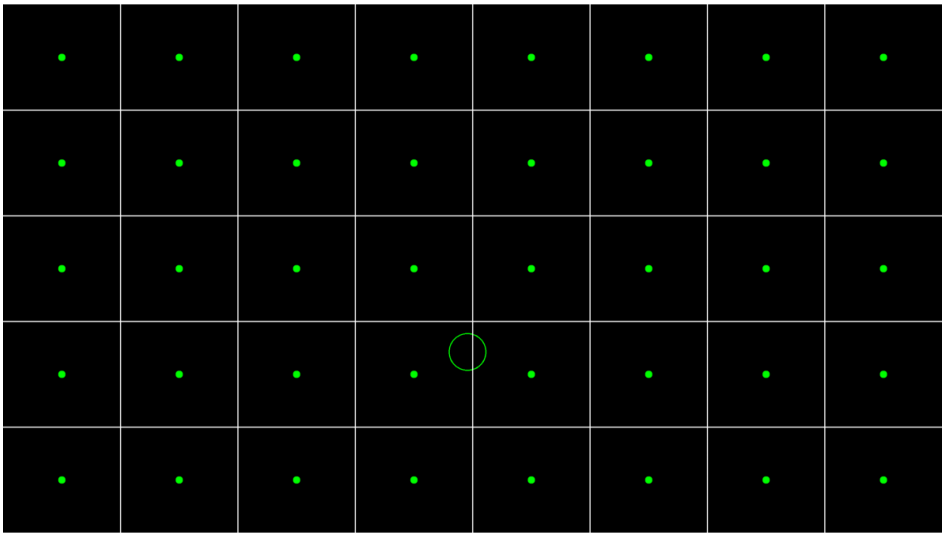


Figure 2.2: Grid used for collecting predicted screen coordinates

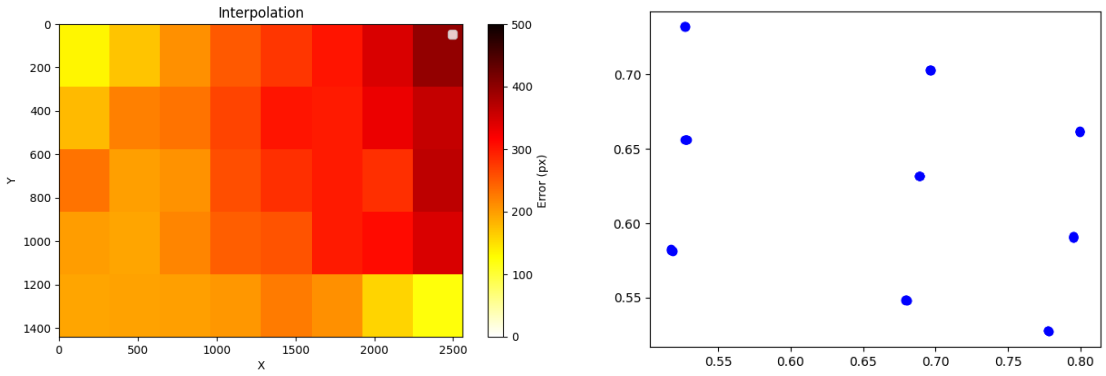
2.3 Data visualization

Finally, after performing 4 independent sessions of data collection, a comparison will be conducted between the predicted screen coordinates and the actual, ground truth locations. The euclidean distance between these two sets of coordinates will be quantified as errors, and to effectively visualise the spatial distribution of these errors heat maps will be generated for each of those methods. By comparing these average errors, we will be able to definitively determine which gaze mapping method offers the most accurate prediction capabilities. Each rectangle on the heat map corresponds to the rectangle on the grid from figure 2.2. The error is measured in pixels and to help the reader to visualize it, an offset of 160 px means that predicted screen coordinate is on a border of the ground truth point's rectangle. We will also convert these error values to centimeters to provide a more intuitive understanding of the errors in real-world terms. Error less than 4cm means the predicted gaze on screen is within the bounds of the rectangle. This comprehensive evaluation will not only shed light on the strengths and weaknesses of each approach but also determine the selection of the most suitable method for future gaze tracking applications.

2.4 Results

2.4.1 Interpolation

As expected Interpolation proved to be the least effective method, with an average error of 230px. We can also notice the peak in inaccuracy in the top right corner of the screen. One reason could be that Interpolation does not take into account that eye surface is not planar. We can also confirm this in figure 2.3b where eye coordinates that correspond to 9 calibration points do not form a rectangular shape. Another reason is the eye camera placement. These camera are not placed right in front of the pupil of the user but rather on the side, resulting in distorted results because of perspective. In figure 2.3b, the coordinates on the left side represent the eye positions closer to camera while coordinates on the right side are pupil positions further away from the camera.

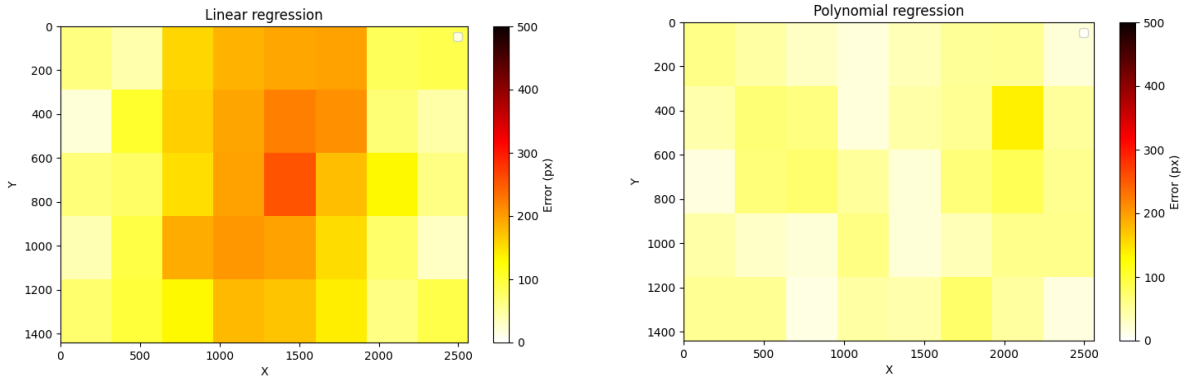


(a) Interpolation error heat map (b) Eye positions during calibration

Figure 2.3: Interpolation method results

2.4.2 Regression methods

Much better results were achieved by using regression methods. Linear regression performed better by around 45% than Interpolation, being more accurate in general. As we can see in figure 2.4a, its main drawback was the accuracy around center of the screen where average error increased to 200px and thus, caused a significant drop in overall accuracy. On the other hand, as shown in figure 2.4b, polynomial regression maintained its consistency across the entire screen and exhibited a very similar error in all parts of the grid which resulted in average error of only 77px.



(a) Linear regression error heat map (b) Polynomial regression error heat map

Figure 2.4: Regression methods results

2.4.3 Consistency test

We also performed consistency test where we fixated on one point in the center of the screen and collected 150 predicted points. These points, originally defined in the ranges [0,2560] for the x-axis and [0,1440] for the y-axis, were normalized to lie between 0 and 1 on both axes. As we can see in figure 2.5, the eye tracker has excellent consistency

with almost no deviation.

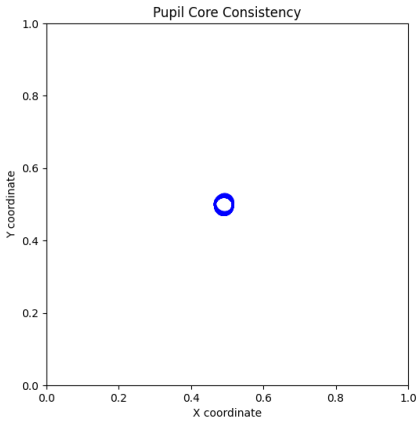


Figure 2.5: Consistency of predicted screen coordinates using Pupil Core eye tracker

2.5 Overview

One important thing to note that in table 2.1 is the maximum error of polynomial and linear regression. Despite much lower average error of polynomial regression, the maximum error was much higher than for linear regression. However when we look at the top 5% of the highest error predictions and average them, the polynomial regression comes on top with an error of 173.64px, while linear regression had an error of 225.03px. What is more, the amount of these points represents less than 0.001% of all collected data. We can safely assume that these high error predictions are either a result of a human error (head movement, blinking) or faulty eye measurement.

Table 2.1: Error overview for each mapping technique in centimeters

Mapping method	Min error	Max error	Average error
Interpolation	2.43	14.44	5.4
Linear regression	0.02	6.06	2.93
Polynomial regression	0.03	8.27	1.8

Chapter 3

Neural network performance

In this chapter we will evaluate the performance of the neural network on vertical screen, point out its issues and try to improve the results. We will also tweak the Isolation Forest method to clean collected eye dataset in order to achieve more accurate mapping.

3.1 Setup

To evaluate the performance of the neural network we will use similar setup as before, with a screen right in front of user and 1080p web camera placed right on top of the screen. Since the network has problems with consistency on frames that should have same output values, we will be collecting 25 points per calibration point instead of 9. This will allow us increase the sensitivity of Isolation Forest and remove 50% of the outlying points to filter out the inconsistencies. The mapping method used will be polynomial regression since it yielded the best results.

3.2 Performance issues

Another downside of this network is its performance. The whole preprocessing, starting from detecting face features, predicting the head position, cropping the eyes and feeding it to neural network in order to predict eye position is computationally challenging since every frame has to go through this process. This significantly slows down the number of predicted eye positions per second. Since we are trying avoid any faulty predictions caused by unwanted head movement, it's essential to keep the data collecting sessions short. For this reason we will be collecting 15 predicted points per grid point to cut the duration of the session. It is important to note that despite reducing the number of collected points, the session will still be 2-3 times longer than with Pupil Core.

3.3 Evaluation

Despite filtering out the inaccurate eye predictions during the calibration, the overall results are less than satisfying. With an average error of 443px and maximum error of 990px, the system struggles to maintain high precision in various areas of the screen. The heat map shown in figure 3.1 underscores the necessity for further improvement.

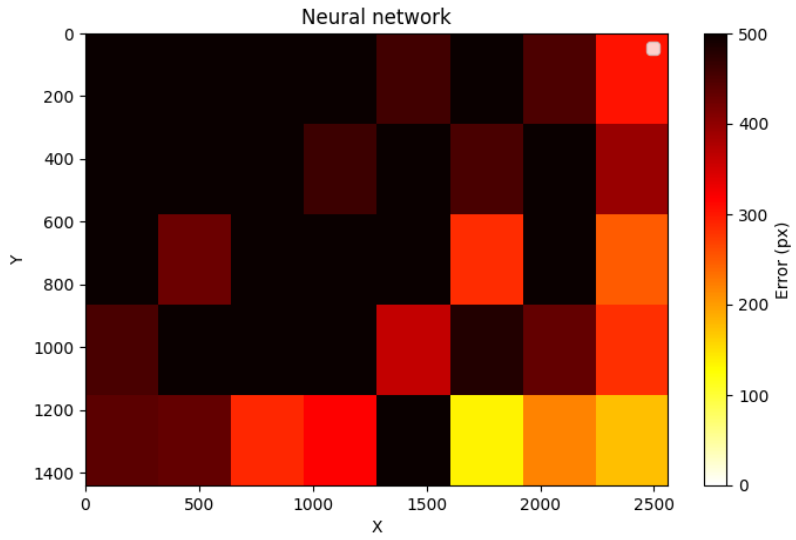
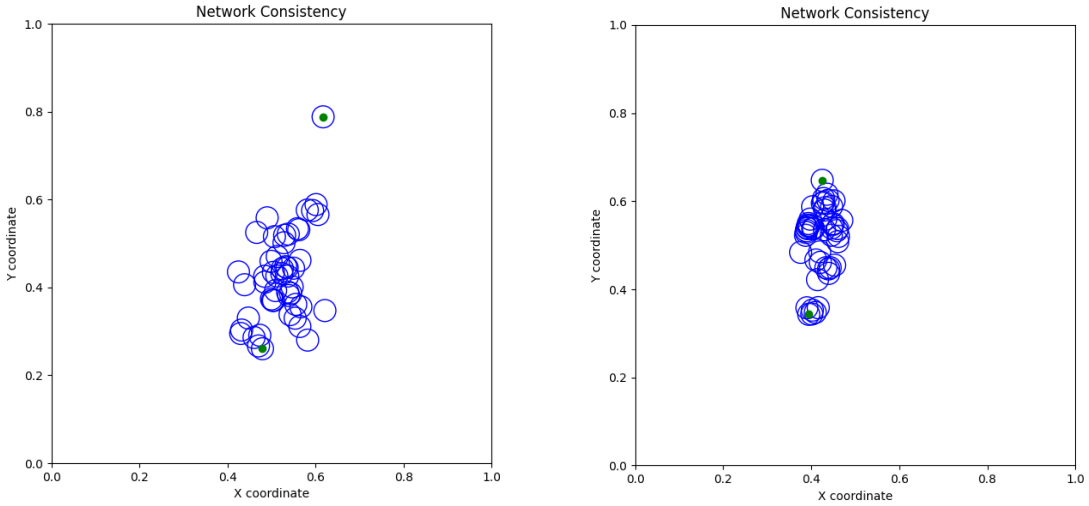


Figure 3.1: Neural network error heat map

3.4 Improvements

3.4.1 Network consistency

One of the things that contributed to improving the accuracy and overall consistency was averaging the last 8 screen predictions. In this test we collected 50 gaze points while fixating on one place on the screen. In figure 3.2a we can see there are some points that deviate from the group. To show the significance of the error we marked two most distant points with green dots. The distance between these points is 837px (19.5cm). In the other figure 3.2b is the result after averaging. We can see the points are much more concentrated with the maximum distance between points being 444px (10.4cm). We managed to improve the consistency by almost 47% and also the overall experience since averaging the predicted points smoothed out the movement of the gaze on screen.



(a) Without averaging (b) With averaging

Figure 3.2: Difference between gaze prediction with averaging and without

3.4.2 Tweaking outliers removal method

One of the issues found with the current implementation is the inefficiency of Isolation Forest. It's sometimes unable to detect the 9 clusters of points, for example in figure 1.3. Another issue is the maximum threshold which can be only set to 50%. However we can afford to reduce the points even more since we are collecting 225 points in total. In figure 3.3 we can see raw eye positions in top left corner and filtered data using Isolation Forest in the top right. Despite removing 50% of the outliers there is still moderately large dispersion of data points in the middle and top. However, when outliers are removed in each group individually, as shown in bottom left, the results are more consistent. On top of that, we can also apply Isolation Forest on all eye data in order to reduce the dispersion even more, reducing the dataset by 66%. This leaves us with 76 points, more than enough to apply Polynomial regression to map them to screen coordinates.

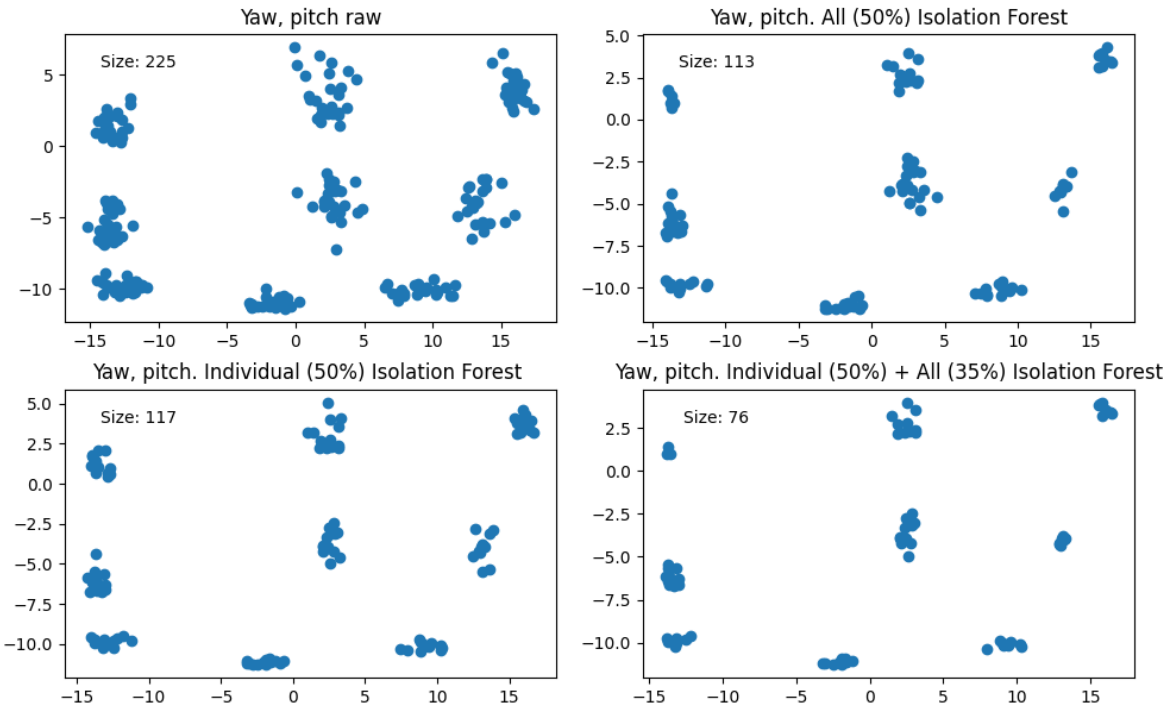


Figure 3.3: Comparison between raw eye data and Isolation Forest

3.5 Results

After implementing the new method for removing outliers and gaze averaging, we managed to reduce the average error to 293px. Also, as we can see in figure 3.4, the inaccuracy spike in top left corner was significantly reduced. While the average error was reduced by nearly 36%, the maximum error remained nearly identical. Although, the top 5% error of the old method was 819.37px compared to new method the average top 5% error is reduced to 713.28px, an improvement of 13%. While we were able to improve the quality of the dataset on which regression is applied, it does not change the fact that this network's output values are inconsistent and it severely affects the overall performance. In table 3.1 we can see side by side comparison between results before and after.

Table 3.1: Error comparison between old and outlier removal method in centimeters

Mapping method	Min error	Max error	Average error
Isolation Forest all	0.27	23.12	10.35
Isolation Forest individual + all	0.06	21.71	6.86

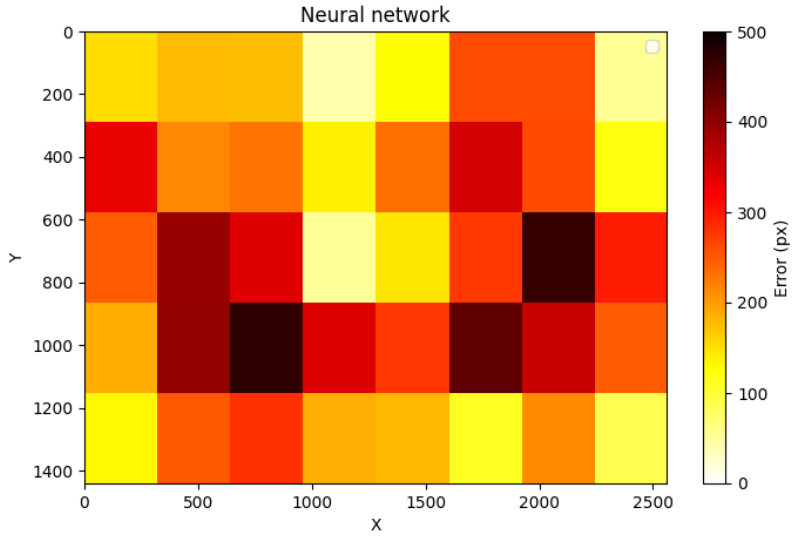


Figure 3.4: Error heat map with improved outlier removal method

Chapter 4

Human-robot interaction

In this chapter we will perform similar evaluation as in previous chapters. However, in this situation there will be screen placed horizontally between the participant and a humanoid robot NICO. When the participant is seated in front of a robot and they interact using the screen between them, collecting eye data during this experiment can provide an insight into human nature during decision making. The aim of this test is to determine whether the eye tracking system is suitable for further implementations in the field of human robot interaction experiments. The example of this scenario is shown in figure 4.1.

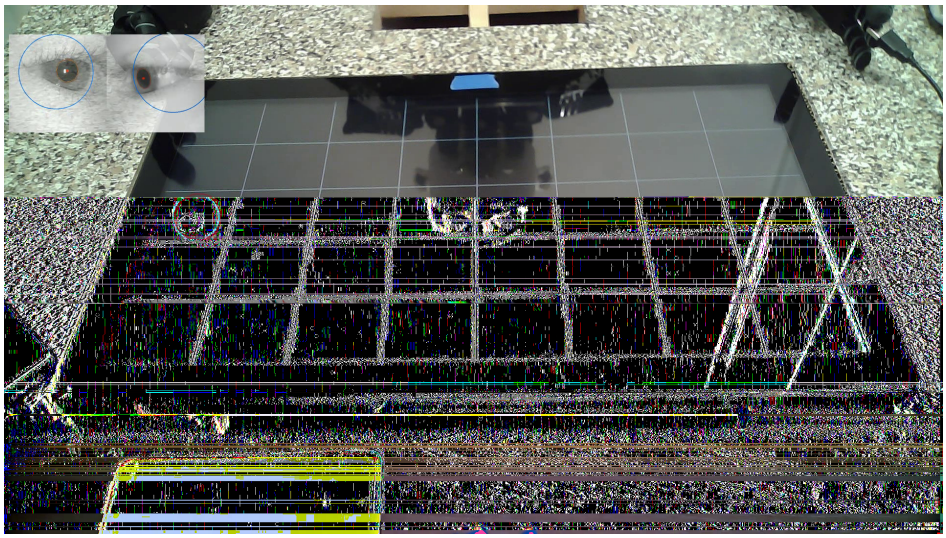


Figure 4.1: The horizontal touchscreen placed between the robot and the participant

4.1 Neural network

In the previous test we placed a camera on top of the screen fulfilling the role of web cam. While it is possible to use NICO's cameras as eyes that would capture the frames of the participant and feed them into the network, it is mentioned in Herashchenko

(2023) that the system has severe problems with predicting eye gaze when the participant is looking down. To avoid this problem, we placed the camera in front of NICO on the table. This allowed us to obtain better angle of the pupil. The mapping method used remained the same (Polynomial Regression), with improved outlier removal method.

4.1.1 Results

As we can see in figure 4.2, the results are rather unsatisfactory. The error in most of the grid points exceeded 500px, going as high as 1454px. When we compare this to previous results with screen in front of participant, it is apparent that angle of the camera and visibility of the eyes plays a huge role in accuracy of the predictions.

Another factor, although not as impacting is the head movement. While in the first test the head was leaned on the chair, in this instance the participant has to keep their head still. Since the whole process of data collection takes around 10 minutes, there will always be some involuntarily head movement.

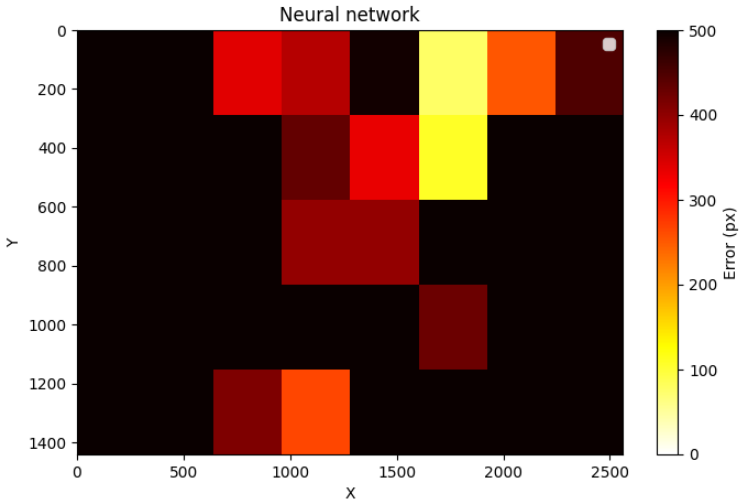


Figure 4.2: Error heat map for neural network on horizontal screen

4.2 Pupil Core

As shown in figure 4.3, there are small inaccuracy spikes in bottom area of the screen. This happened because eye tracker wasn't able to detect full circle of the pupil when participant was looking down. The view might have been obstructed with eye lid or eyelashes causing the drop in detection confidence. The proof of this are several instances of predicted screen coordinates with the error of 810px. These inaccuracies hindered the overall performance which resulted in average error of 108.40px.

Despite this increase in error, the Pupil Core eye tracker remains a very effective tool for gaze tracking applications. The average error of 108.40px, while higher than before, is still within acceptable limits for most practical uses. This level of accuracy can still provide valuable insights in participant behavior analysis, interface testing, and other research scenarios. Furthermore, the system's ability to consistently detect gaze direction under most conditions outweighs the occasional inaccuracies caused by extreme gaze angles or obstructions like eyelids and eyelashes.

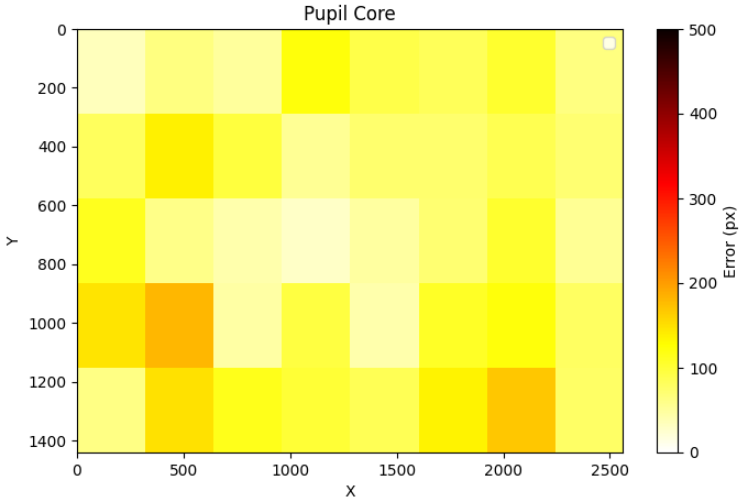


Figure 4.3: Error heat map for Pupil Core on horizontal screen

4.3 Overview

The error analysis in this chapter demonstrates that camera placement significantly impacts the accuracy of eye tracking systems, particularly for appearance-based models as shown in table 4.1. This change more than doubled the average error of predicted eye gaze. Pupil Core exhibited some limitations in tracking eyes looking downwards, its overall error increased from 77px to only 108.4px.

While we are not trying to compare these two eye tracking approaches, it is apparent that this neural network system has a lot of room for an improvement.

Table 4.1: Error overview on horizontal screen in centimeters

Estimation method	Min error	Max error	Average error
Pupil core	0.07	18.93	2.53
Neural network	0.73	34	15.28

Conclusion

This thesis explored the possibility of using an appearance-based gaze estimation model based on neural network developed by Herashchenko (2023) for real world applications in human-robot interaction. The system utilizes a convolutional neural network to predict gaze direction using user eye images captured by a standard webcam.

We explored various gaze mapping techniques and data filtering method. We compared the performance of interpolation, linear regression, and polynomial regression for mapping the eye positions onto a screen coordinates. Our evaluation revealed that polynomial regression yielded the most accurate results, achieving an average error of 77 pixels (1.8cm) when used with a Pupil Labs eye tracker for calibration.

While the neural network demonstrated promising performance on a dataset containing both real-world and synthetic data, our evaluation revealed limitations that hinder its practical applicability. Firstly, the network exhibited inconsistency in its output, especially for consecutive frames containing the same eye image. This inconsistency significantly compromises the system's reliability for tasks requiring precise gaze tracking. Secondly, the system's performance is highly susceptible to variations in user posture and head orientation.

To fight this inconsistency, we implemented an outlier removal method using Isolation Forest to reduce the influence of noisy or inaccurate eye position data. This approach demonstrably improved the overall accuracy of the network when combined with polynomial regression for gaze mapping.

With these improvements we managed to reduce the average error on vertical screen from 443px to 293px (6.8cm), rendering this model suitable for rough estimation for vertical screen tracking. On the other hand, the results on a horizontal screen where user was looking down were unsatisfactory, with the average error of 650px (15cm). This highlights the limitations of appearance-based eye tracking using webcams, particularly in scenarios where eye visibility is not perfect.

In conclusion, while this appearance-based model offers a glimpse into the potential of webcam eye tracking, its current limitations render it unsuitable for real world applications in human-robot interaction. Tracking the positions of the pupils on table/horizontal screen proves to be a difficult task where standard rgb cameras are not satisfactory. We suggest incorporating the infrared camera which highlights the pupil

and might offer better results. Future research efforts should also focus on improving the network's overall consistency.

Bibliography

Chen, M., Anderson, J., and Sohn, M. (2001). What can a mouse cursor tell us more? correlation of eye/mouse movements on web browsing. *Proceedings of CHI Extended Abstracts*, pages 281–282.

Hempel, T., Abdelrahman, A. A., and Al-Hamadi, A. (2022). 6d rotation representation for unconstrained head pose estimation. In *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE.

Herashchenko, D. (2023). Robot reading human head pose and gaze direction. Bachelor's thesis, Comenius University Bratislava.

Huang, J.-B., Cai, Q., Liu, Z., Ahuja, N., and Zhang, Z. (2014). Towards accurate and robust cross-ratio based gaze trackers through learning from simulation. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 75–82.

Jiaqi, J., Zhou, X., Chan, S., and Chen, S. (2019). Appearance-based gaze tracking: A brief review. In *Intelligent Robotics and Applications*, pages 629–640.

Kassner, M., Patera, W., and Bulling, A. (2014). Pupil: An open source platform for pervasive eye tracking and mobile gaze-based interaction.

Mathias, A. (2014). clmtrackr: Javascript library for precise tracking of facial features via constrained local models `https://github.com/auduno/clmtrackr?tab=readme-ov-file`. Accessed: 2024-05-30.

Metahuman (2024). Metahuman high-fidelity digital humans made easy. `https://www.unrealengine.com/en-US/metahuman`. Accessed: 2024-05-30.

Mokatren, M., Kuflik, T., and Shimshoni, I. (2022). 3D gaze estimation using rgb-ir cameras. *Sensors*, 23:381.

Papoutsaki, A., Sangkloy, P., Laskey, J., Daskalova, N., Huang, J., and Hays, J. (2016). Webgazer: Scalable webcam eye tracking using user interactions. In *Proceedings of*

the 25th International Joint Conference on Artificial Intelligence (IJCAI), pages 3839–3845. AAAI.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

PupilLabs, G. (2024). Pupil labs github. `https://github.com/pupil-labs/pupil/tree/master`. Accessed: 2024-05-30.

Sheela, S. and Radhika, K. R. (2020). Feature based methods for eye gaze tracking. In *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 1101–1107.

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

Zheng, E. (2018). Batch face for modern research `https://github.com/elliottzheng/batch-face?tab=readme-ov-file`. Accessed: 2024-05-30.

Pupil Labs calibration tutorial

World Camera

1. Set the resolution to 1920x1080 and the refresh rate to 30fps.
2. Utilize the slide bar to adjust the exposure time for your region of interest.
3. If necessary, make fine-tuned adjustments in the "Image Post Processing" section.

Interact with the system using the following key commands:

- Press "C" to initiate/terminate the calibration process.
- Use "T" to test accuracy by focusing on an object and clicking on it in the world camera window.
- Press "R" to start/stop the recording process.

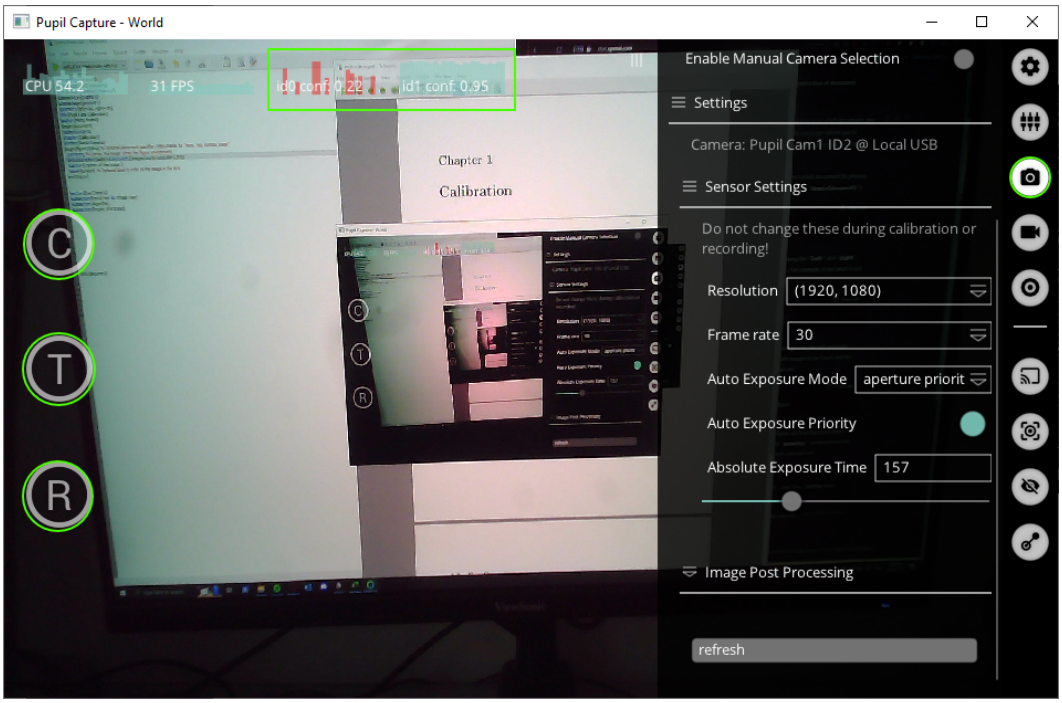
`Id0` (left eye) & `Id1` (right eye) confidences are represented on a scale of 0 to 1 and can be found on top of world camera screen as shown in picture.

Eye Camera

To ensure precise pupil tracking with optimal image quality, it's crucial to configure the eye cameras correctly. Follow the instructions below for each eye individually, within their respective windows.

While maintaining identical settings for both eyes regarding resolution and framerate, consider the following factors, such as camera angles and consistent lighting in the room, for the best results:

- The Pupil Core provides eye camera extenders to enhance the angle of pupil capture.
- If needed, you can flip eye image in the general settings. However, note that image orientation doesn't affect the eyetracking algorithm's performance.



Pupil Capture world camera

Resolution & refresh rate

Set the resolution to the highest possible quality, in this case, 400x400, and the framerate to 120. The increased framerate is particularly beneficial for accurately tracking fast or sudden eye movements, resulting in fewer drops in confidence levels.

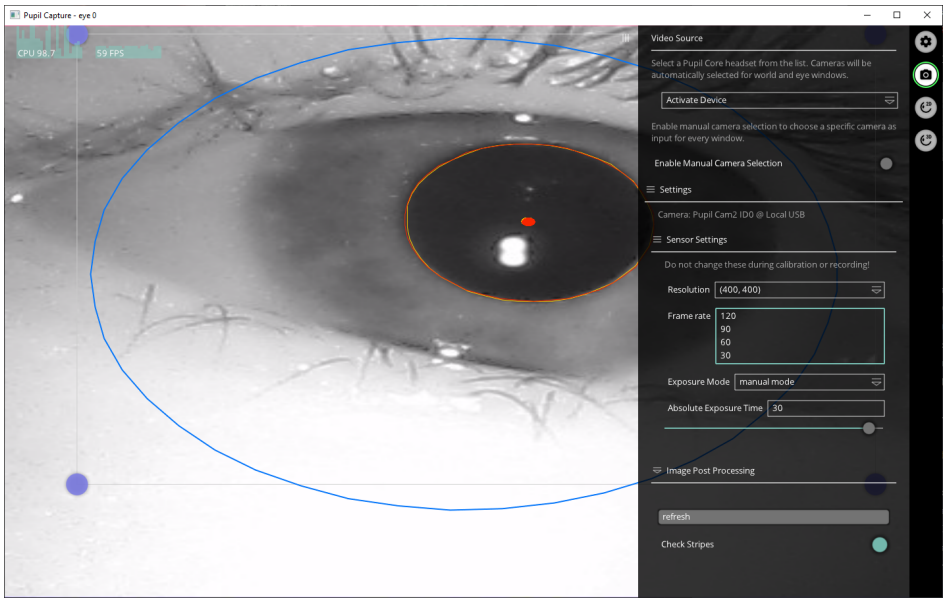
You can also adjust the "Absolute Exposure Time" to higher values, enhancing the luminance of the eye. This adjustment is useful when tracking in environments with inconsistent lighting conditions.

2D eye model

The algorithm employed by this system aims to detect the dark pupil within the infrared-illuminated eye camera image. Notably, it is independent of corneal reflections, rendering it effective even for individuals wearing contact lenses.

The eye camera offers two distinct modes, both accessible through the general settings: Algorithm Mode and ROI (Region of Interest) Mode. These modes are instrumental in refining the pupil tracking algorithm for heightened precision.

Remember to update these settings each time the eyetracking device is used by a different individual or when the surrounding conditions undergo changes.

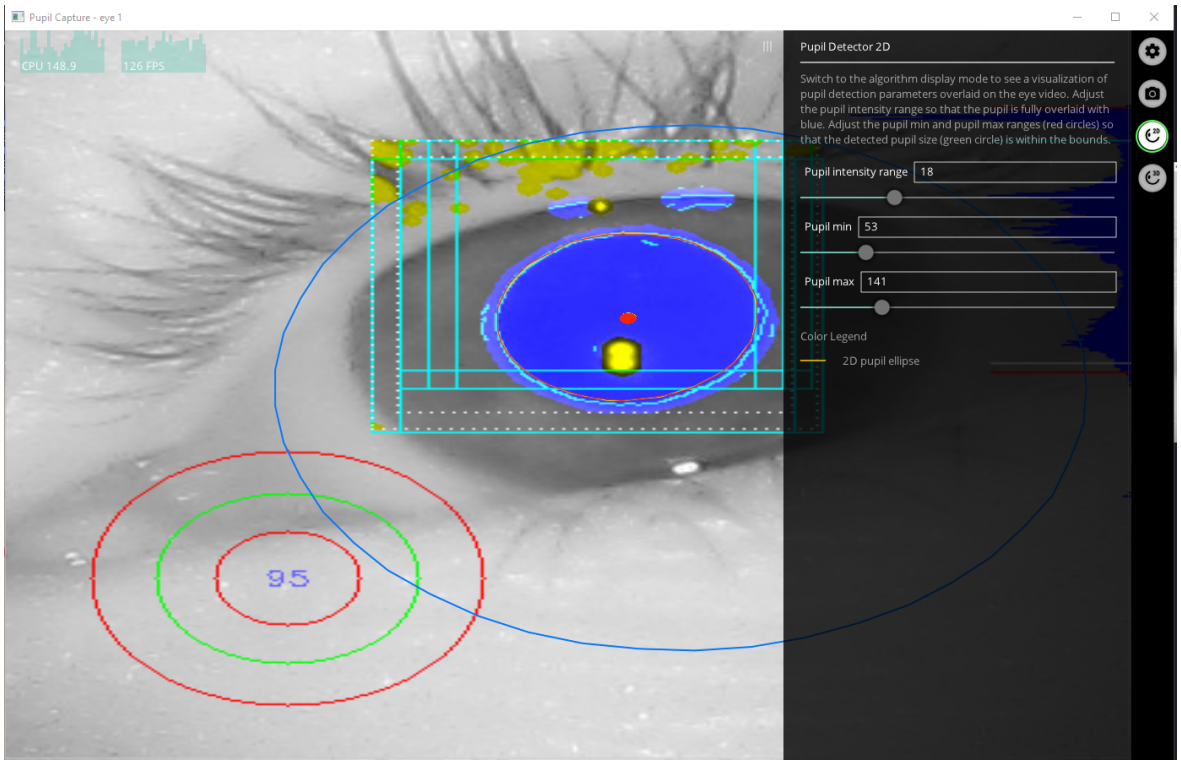


Pupil Capture eye-resolution settings

Algorithm

The blue overlay of the pupil must cover the area across all eye positions. The intensity range of the pupil overlay relies on the image's sharpness and the contrast between the pupil and the iris. For enhanced precision, consider elevating the absolute exposure time 4.3.

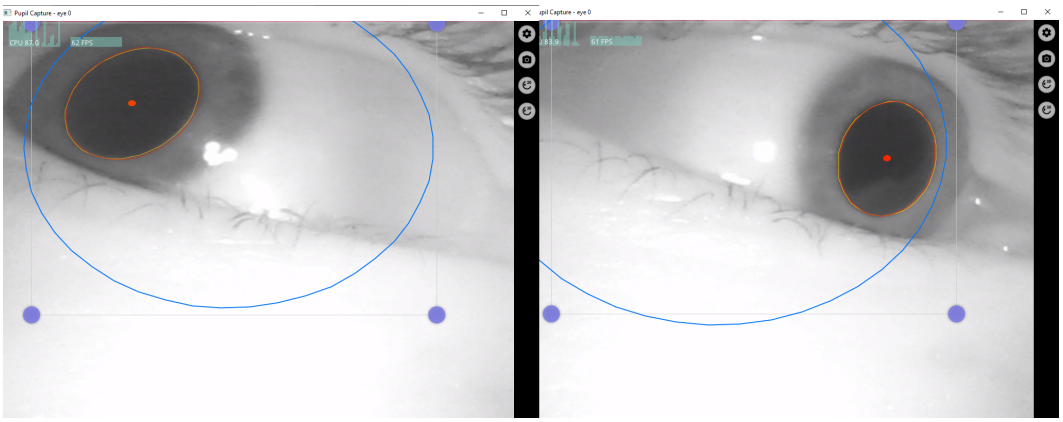
The green circular line represents the current size of the pupil, while the red circular lines represent the minimum and maximum pupil sizes. Make sure both these values are properly configured, as it heavily relies that the pupil size consistently falls within these designated ranges.



Pupil Capture eye-resolution settings

Region of interest

Set the region of interest borders like shown below, so it covers whole range of motion of the pupil. This significantlly improves eyetracking quality when person has longer eyelashes or uses mascara.



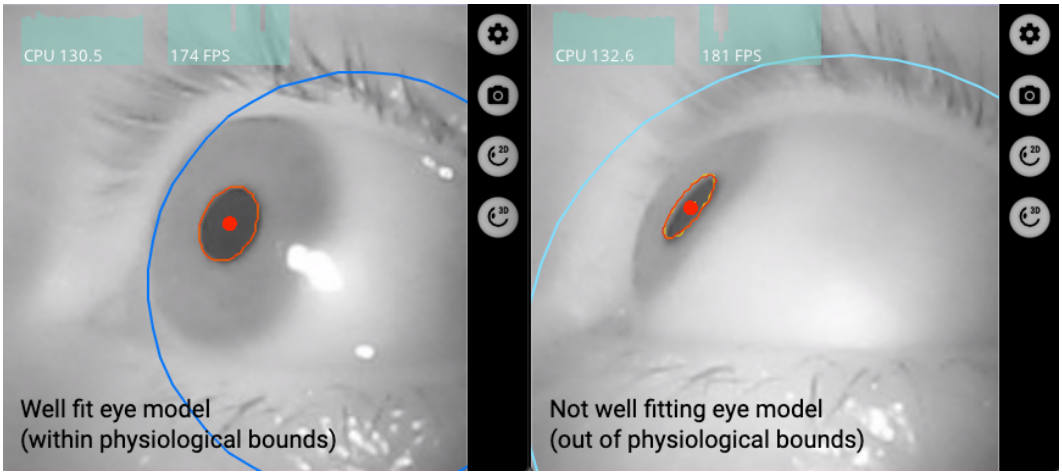
ROI settings example

3D eye model - pyed3d

The blue circle encircling the eye illustrates the fitting of the 3D model to eye movements. Ensure the color remains consistently dark blue throughout the entire range of

pupil motion. In the 3D section, there's an option to reset the 3D model and initiate the fitting process anew. Additionally, you can freeze the model to prevent it from re-fitting. This frozen model proves valuable when head movement is unnecessary for the experiment, as Pupil Labs employs the model to rectify instances of slippage [1] or intentional head movements.

Fitting 3D model



Even though the detected pupil ellipses fit the pupil well on the right-side picture, the model is outside of the physiological ranges. This can lead to errors in gaze-direction and pupil-size estimates. You can improve such a model by looking into different directions.

Calibration - Natural Features

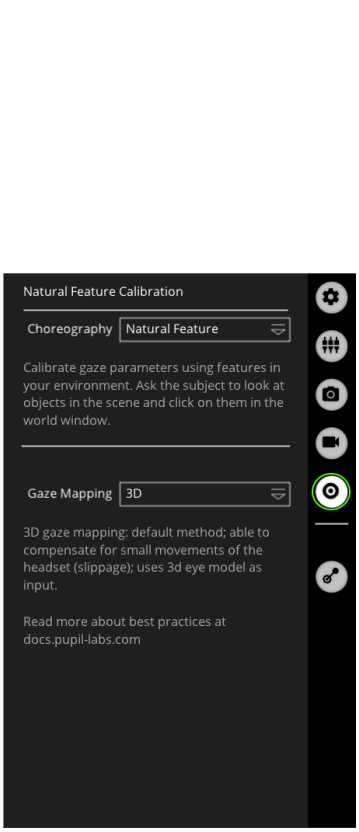
1. Select "Natural features" for choreography and set gaze mapping to 3D. Note that 3D mapping offers precise tracking even during head movement or slippage, although it is less accurate than 2D gaze mapping.

2. Activate the "Accuracy Visualizer" in the plugin menu. This tool calculates accuracy post-calibration and displays the calibrated area of interest in the Pupil Capture window along with angular precision. This visualization can aid in configuring the area of interest. For instance, you can position objects within the field of view of the world camera, such as coins placed at the corners or along the edges of the camera's view.

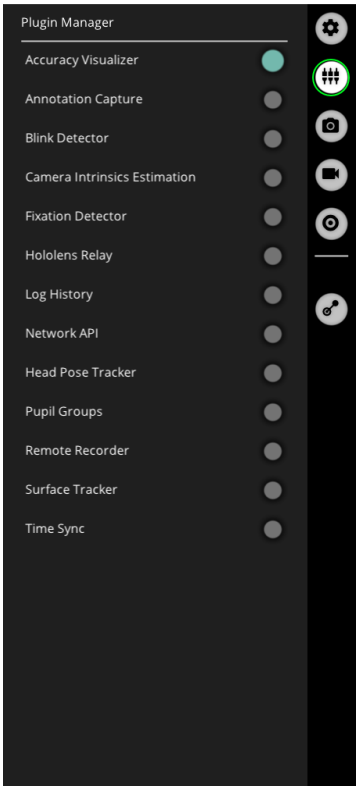
3. Initiate the calibration process by looking at the object and clicking on it in the Pupil Capture window. This step is repeated for all designated points be-

[1] Slippage refers to the headset slipping on the participant's head

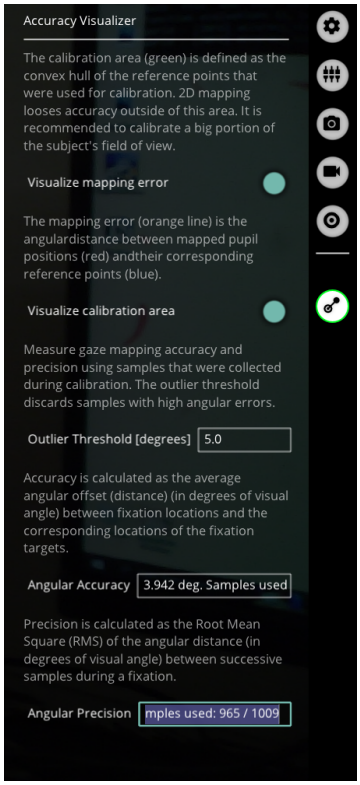
fore concluding the calibration. It's crucial to minimize head movement during calibration to ensure accuracy.



Calibration method



Plugin selection



Accuracy vizualizer plugin

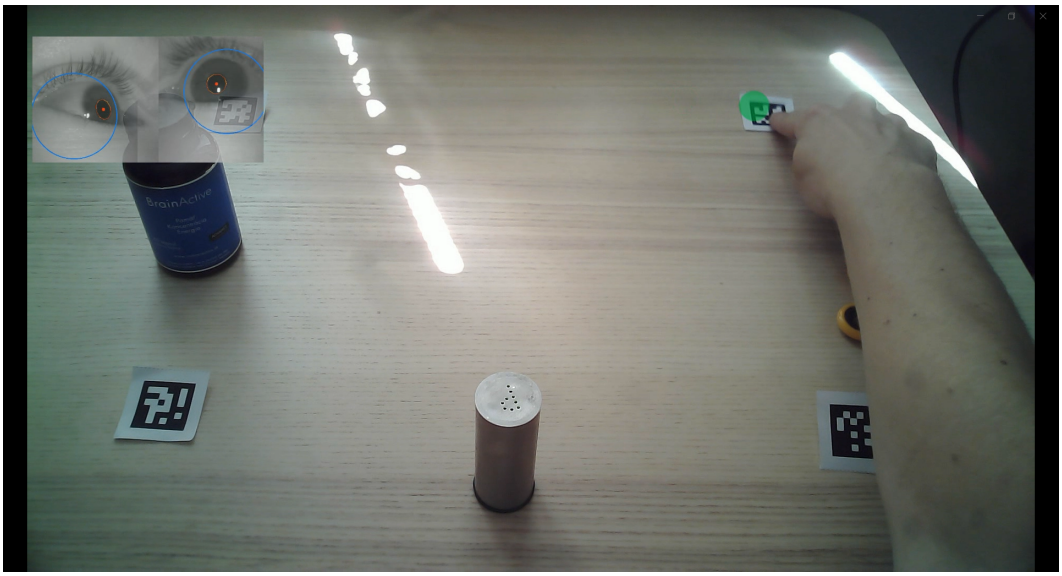
Results

Right after the calibration process, Pupil Capture presents the calibration result. Aim for an angular accuracy of approximately 2°, which signifies the angular deviation between the object and gaze direction. Angular precision indicated the angular gap between successive samples captured during a fixation in the calibration process. Essentially, it quantifies the stability of the pupils during calibration when samples (obtained by clicking on the object in the Pupil Capture window) were collected.

Pupil Capture discards samples that exceed the accuracy threshold established in the accuracy visualizer window. If over 20% of samples from angular precision are discarded, Pupil Capture shows the message "An unexpectedly large amount of pupil data ($> 20\%$) was dismissed due to low confidence." This message indicates a significant drop in id_ confidence at certain points during sample collection.

Consider examining the fit of the 3D model and the level of id_ confidence, particularly in more extreme pupil positions. When calibrating on a table, it's possible that eyelashes obstruct the pupils when looking at the lower part of the calibration area. To

address this, attempt a more direct angle on the table or adjust the calibration points to a higher position.



Natural features experiment on table. Four pieces paper were used as calibration points.