

De huidige problemen rondom oog-besturing in computers en hoe deze opgelost kunnen worden.

Miel De Roeck.

Scriptie voorgedragen tot het bekomen van de graad van
Professionele bachelor in de toegepaste informatica

Promotor: Dhr. S. Verschraege

Co-promotor: Dhr. B. De Lange

Academiejaar: 2024–2025

Eerste examenperiode

Departement IT en Digitale Innovatie .



Woord vooraf

Dit onderzoek heeft als doel te onderzoeken wat de huidige problemen zijn met webcam gebaseerde oogbesturingssystemen en hoe deze opgelost kunnen worden. Er is specifiek gekozen voor webcam gebaseerde technologie omdat deze goedkoper en toegankelijker is. Er is geen derde partij vereist om de systemen op te zetten, en alles kan vanuit huis worden uitgevoerd. Via deze systemen kunnen mensen met een motorische beperking toch nog actief zijn op het web wat in onze samenleving onmisbaar is.

Ik wens nog specifiek mijn promotor Sion Verschraege en co-promotor Bregt De Lange te bedanken voor hun begeleiding en constructieve feedback. Ook wil ik het bedrijf Eleven Ways voor de tijd die zij hebben vrijgemaakt om mijn vragen met betrekking tot bestaande problemen te beantwoorden. Daarnaast bedank ik dokter Tobias Fischer, Hoofddocent aan de Queensland University of Technology, voor zijn advies rondom RT-GENE en blikschatting.

Samenvatting

Dit onderzoek richt zich op het toegankelijker maken van het besturen van een computermuis voor mensen met een motorische beperking. Om dit te doen richt het onderzoek zich op de vraag: wat zijn de bestaande problemen met webcam gebaseerde oogbestuurde software en hoe kunnen deze opgelost worden? Het doel was een werkend prototype te creëren waarmee aangetoond kan worden hoe de precisie van deze technologie verbeterd kan worden om zo het besturen van een computer toegankelijker en gebruiksvriendelijker te maken voor mensen met een motorische beperking.

Om dit doel te realiseren is er tijdens het onderzoek eerst een uitgebreide literatuurstudie verricht naar: de theoretische werking van oogtracerende en -besturende software, hoe deze werking best vertaald wordt naar de programmeertaal Python, de huidige problemen en mogelijke oplossingen van deze problemen.

Naast de literatuurstudie werd er ook een persoonlijk gesprek gevoerd met het bedrijf Eleven Ways. Deze combinatie resulteerde in de identificatie van volgende problemen: de technologie verreisd een hoge graad van precisie; het gebruik ervan vraagt om een kwalitatieve webcam; lichtomstandigheden kunnen een negatieve invloed hebben op de werking; er treden kalibratieproblemen op, zowel aan de softwarematige als aan de praktische kant; en de gebruikersinterface maakt complexere vormen van interactie vaak niet toegankelijk.

Hierna werd er gekozen om verder te bouwen op het bestaande systeem GazeTracking gebouwd door Iamé ([z.d.](#)). De technische verwezenlijking van dit onderzoek implementeert: de python bibliotheek MediaPipe voor gezicht en -oogdetectie, een virtuele kinsteun en [Contrast Limited Adaptive Histogram Equalization \(CLAHE\)](#) om de invloed van belichting te neutraliseren. Dit resulteert in een systeem dat de ogen en de richting van het blikpunt van de gebruiker detecteert. Aan de hand van deze gedetecteerde richting wordt de muis in dezelfde richting gestuurd. Indien de gebruiker knippert, wordt er een klik actie verricht. De volledige implementatie en code van de technische verwezenlijking is te vinden op <https://github.com/Miel-DR2003/Poc-Bachelorproef-2025>.

Na de ontwikkeling van de technische verwezenlijking werd een experiment uitgevoerd met twee proefpersonen, één met een motorische beperking en één zonder. Beide proefpersonen voerden negen verschillende navigatietaken uit in verschillende lichtomstandigheden, zowel met GazeTracking als met de technische verwezenlijking. Gedurende dit experiment werd de gespendeerde tijd per actie verzameld alsook de coördinaten van de muis wanneer een klik actie werd uitge-

voerd. De verzamelde gegevens werden hierna geanalyseerd om de precisie van beide systemen te vergelijken, evenals de invloed van belichting en de gemiddeld gespendeerde tijd per actie.

Het resultaat van dit onderzoek is een werkend systeem dat aantoon hoe de precisie van oogbesturende software verbeterd kan worden. Dit is afgeleid van de analyse waarbij de gemiddelde euclidische afstand tot aan het doel bijna altijd lager lag bij het gebruik van de technische verwezenlijking. De verhoogde precisie gaat echter ten koste van een langere gemiddelde benodigde tijd per actie. Hiernaast bevat het ontwikkelde systeem ook nog enkele beperkingen: het bewegen van de muis naar onder, de gevoeligheid voor lichtomstandigheden en de hoge gemiddelde tijd per actie.

Inhoudsopgave

Lijst van figuren	ix
Lijst van tabellen	x
Lijst van codefragmenten	xii
Lijst van afkortingen	xiii
1 Inleiding	1
1.1 Probleemstelling	1
1.2 Onderzoeks vraag	1
1.3 Onderzoeksdoelstelling	2
1.4 Opzet van deze bachelorproef	2
2 Stand van zaken	3
2.1 Waarom oog-besturing	3
2.2 De onderliggende werking en hoe deze best vertaald worden in Python	4
2.2.1 Theoretische werking	4
2.3 GazeTracking	5
2.4 De huidige problemen met webcam gebaseerde oog-besturing	7
2.4.1 Kalibratie problemen	8
2.4.2 Precisie	8
2.4.3 Lichtgevoelig	9
2.4.4 Andere uitdagingen op het gebied van bruikbaarheid	9
2.5 Literatuur rondom relevante oplossingen	10
2.5.1 Essentiële Python-packages	10
2.5.2 CLAHE	10
2.5.3 Hoofdtracing met een virtuele kinsteen	12
2.6 Onderzochte alternatieven	13
2.6.1 Object detectie	13
2.6.2 Blikpunt detectie	13
2.6.3 kalibratie	14
2.6.4 Python-packages	15
3 Methodologie	16
3.1 Literatuurstudie	16
3.2 Creëren van een gecontroleerde testomgeving	16

3.3	Creëren van een oplossing	17
3.4	Experiment uitvoeren en data verzamelen	17
3.5	Analyse	17
3.6	Conclusie en discussie	18
4	POC	19
4.1	EyeTrackingSystem	20
4.1.1	CLAHE	21
4.1.2	Object detectie	21
4.1.3	Muis beweging	26
5	Experiment	28
5.1	Testomgeving	28
5.2	Opstelling	30
5.3	Proefpersonen	31
5.3.1	Proefpersoon met motorische beperking	31
5.3.2	Proefpersoon zonder motorische beperking	31
5.4	Uitvoeren van het experiment	31
5.4.1	Uitvoering van het experiment met GazeTracking	32
5.4.2	Uitvoering van het experiment met de technische verwezenlijking	35
5.5	Ondervinding uit het experiment	38
5.5.1	Ondervingen van proefpersoon met motorische beperking	38
5.5.2	Ondervingen van proefpersoon zonder motorische beperking	38
6	Analyse	39
6.1	Analyse van precisie	39
6.1.1	Proefpersoon met motorische beperking	39
6.1.2	Proefpersoon zonder motorische beperking	41
6.2	Analyse van heatmaps	42
6.2.1	Heatmaps GazeTracking	42
6.2.2	Heatmaps technische verwezenlijking van dit onderzoek	44
6.3	Analyse van belichting	46
6.4	Analyse van tijd	46
7	Conclusie	48
A	Onderzoeksvoorstel	51
A.1	Inleiding	51
A.2	Literatuurstudie	52
A.2.1	Waarom oog-besturing	52
A.2.2	De onderliggende werking en hoe deze best vertaald worden in python	53

A.2.3 De huidige problemen met webcam gebaseerde oog-besturing	54
A.2.4 Mogelijke oplossingen	56
A.3 Methodologie	57
A.3.1 Literatuurstudie	57
A.3.2 Creëren van een gecontroleerde testomgeving	57
A.3.3 Creëren van een oplossing	58
A.3.4 Experiment uitvoeren en data verzamelen	58
A.3.5 Data analyse	58
A.4 Verwacht resultaat	58
Bibliografie	59

Lijst van figuren

2.1 Visualisatie van de 68 punten topologie	6
2.2 GazeTracking klassendiagram	7
2.3 Mediapipe face mesh	11
2.4 Input en output van het CLAHE model	12
4.1 Klassendiagram van het POC van dit onderzoek	19
4.2 Afbeelding van het systeem zonder kalibratie van de virtuele kinstuur	23
4.3 Afbeelding van het systeem met kalibratie van de virtuele kinstuur met het hoofd op de gekalibreerde positie	24
4.4 Afbeelding van het systeem met kalibratie van de virtuele kinstuur met het hoofd niet op de gekalibreerde positie	24
5.1 Homepagina	29
5.2 Appelen pagina	29
5.3 Peren pagina	30
6.1 Heatmap van de homepagina bij het gebruik van GazeTracking door een persoon zonder motorische beperking	42
6.2 Heatmap van de appelen pagina bij het gebruik van GazeTracking door een persoon zonder motorische beperking	43
6.3 Heatmap van de homepagina bij het gebruik van de technische ver- wezenlijking door een persoon zonder motorische beperking	44
6.4 Heatmap van de appelen pagina bij het gebruik van de technische verwezenlijking door een persoon zonder motorische beperking	45
6.5 Gegroepeerde Staafdiagram met de invloed van belichting op de ge- middelde euclidische afstand	46

Lijst van tabellen

5.1	De verzamelde coördinaten van de muispositie bij het uitvoeren van een klik voor elke navigatietaak met GazeTracking onder verschillende lichtomstandigheden bij een persoon met een motorische beperking	33
5.2	De benodigde tijd (in seconden, afgerond op 0 decimalen) voor elke navigatietaak met GazeTracking onder verschillende lichtomstandigheden bij een persoon met een motorische beperking	34
5.3	De verzamelde coördinaten van de muispositie bij het uitvoeren van een klik voor elke navigatietaak met GazeTracking onder verschillende lichtomstandigheden bij een persoon zonder een motorische beperking	34
5.4	De benodigde tijd (in seconden, afgerond op 0 decimalen) voor elke navigatietaak met GazeTracking onder verschillende lichtomstandigheden bij een persoon zonder een motorische beperking	35
5.5	De verzamelde coördinaten van de muispositie bij het uitvoeren van een klik voor elke navigatietaak met de technische verwezenlijking van dit onderzoek onder verschillende lichtomstandigheden bij een persoon met een motorische beperking	36
5.6	De benodigde tijd (in seconden, afgerond op 0 decimalen) voor elke navigatietaak met de technische verwezenlijking van dit onderzoek onder verschillende lichtomstandigheden bij een persoon met een motorische beperking	36
5.7	De verzamelde coördinaten van de muispositie bij het uitvoeren van een klik voor elke navigatietaak met de technische verwezenlijking van dit onderzoek onder verschillende lichtomstandigheden bij een persoon zonder een motorische beperking	37
5.8	De benodigde tijd (in seconden, afgerond op 0 decimalen) voor elke navigatietaak met de technische verwezenlijking van dit onderzoek onder verschillende lichtomstandigheden bij een persoon zonder een motorische beperking	37
6.1	De euclidische afstand, afgerond op 2 decimalen, van de muispositie bij het gebruik van GazeTracking onder verschillende lichtomstandigheden bij de proefpersoon met een motorische beperking	40

6.2 De euclidische afstand, afgerond op 2 decimalen, van de muispositie bij het gebruik van de technische verwezenlijking van dit onderzoek onder verschillende lichtomstandigheden bij de proefpersoon met een motorische beperking	40
6.3 De euclidische afstand, afgerond op 2 decimalen, van de muispositie bij het gebruik van GazeTracking onder verschillende lichtomstandigheden bij de proefpersoon zonder een motorische beperking	41
6.4 De euclidische afstand, afgerond op 2 decimalen, van de muispositie bij het gebruik van de technische verwezenlijking van dit onderzoek onder verschillende lichtomstandigheden bij de proefpersoon zonder een motorische beperking	41
6.5 De gemiddeld benodigde tijd (in seconden, afgerond op 2 decimalen) voor elk proefpersoon om een actie uit te voeren met GazeTracking en de technische verwezenlijking van dit onderzoek	46

Lijst van codefragmenten

2.1 Het startpunt van de code voor GazeTracking	5
2.2 Code uit GazeTracking waarin getoond wordt hoe het <code>_face_detector</code> attribuut verkregen wordt	5
2.3 Code uit GazeTracking waarin getoond wordt hoe het <code>_predictor</code> attribuut verkregen wordt	6
4.1 Code uit technische verwezeling die bepaalt hoe de muis beweegt op basis van de opgegeven richting	26
5.1 Code uit GazeTracking vooraf aan de implementatie om de muis over te nemen	32
5.2 Code uit GazeTracking na aan de implementatie om de muis over te nemen	32

Lijst van afkortingen

A

AI Artificiële intelligentie. 10, 50

B

BF Bewerkt Frame. 20–26

BGR Blue-Green-Red. 21

C

CLAHE Contrast Limited Adaptive Histogram Equalization. iv, vi, ix, 10, 12, 20, 21, 49, 50

D

DCD Developmental Coordination Disorder. 31, 46

H

HOG Histogram of Oriented Gradients. 5

I

I-DT Dispersion-Threshold Identification. 14

I-VT Velocity-Threshold Identification. 13, 14

L

LB Links boven. 25, 30, 33–37, 40, 41, 43

LM Links midden. 26, 30, 33–37, 40, 41, 47

LO Links onder. 26, 30, 33–37, 40, 41

M

M Midden. 26, 30, 33–37, 40, 41, 43, 45

MB Midden boven. 26, 30, 33–37, 40, 41, 43, 45

ML Machine Learning. 6, 10, 50

MO Midden onder. 26, 33–37, 40, 41, 43

O

OPENCV Open Source Computer Vision. 10

R

RB Rechts boven. 26, 30, 33–37, 40, 41

RM Rechts midden. 26, 30, 33–37, 40, 41

RO Rechts onder. 26, 30, 33–37, 40, 41

S

SR Symbolische Regressie. 14, 49

1

Inleiding

Toegankelijkheid in IT is een onderwerp dat steeds actueler wordt. Dit is niet alleen te zien in de veranderende mentaliteit maar ook in de wetgevingen. Op 28 juni 2025 gaat de ‘European Accessibility Act’ in, deze wetgeving dicteert dat alle e-commerce bedrijven hun diensten toegankelijk moet maken door te voldoen aan de WCAG 2.1 AA richtlijnen (AccessibleEU, 2025). Dit onderzoek wenst verder te bouwen op toegankelijkheid maar focust zich op het besturen van de computermuis.

1.1. Probleemstelling

Werken met een computermuis vereist een precieze motoriek waardoor individuen met motorische beperkingen zoals parkinson, verlammingen,... het vaak lastig hebben met het gebruik hiervan. Dit werd ook al aangetoond in meerdere studies zoals die van TREWIN en PAIN (1999) en Spiller e.a. (2019). Naast hun relevantie voor individuen met motorische beperkingen blijken webcam gebaseerde eye-trackers een waardevol instrument om wetenschappelijk onderzoek te doen vanwege hun brede toegankelijkheid en gebruiksgemak. Doordat ze onafhankelijk kunnen worden ingezet zonder de noodzaak van een specialist, bieden ze een efficiënte en laagdrempelige oplossing voor diverse onderzoeksdoeleinden (Kaduk e.a., 2023). Ondanks hun potentieel zijn er nog steeds verschillende problemen met deze technologie.

1.2. Onderzoeksraag

Wat zijn de bestaande problemen met webcam gebaseerde oogbestuurde software en hoe kunnen deze opgelost worden?

1.3. Onderzoeksdoelstelling

Op het einde van dit onderzoek verwachten we een werkend prototype waarvan duidelijk aangetoond is dat het een verbetering biedt in de precisie van oogbesturende software. Op deze manier maakt dit onderzoek het besturen van een computer toegankelijker en gebruiksvriendelijker te maken voor mensen met motorische beperkingen.

1.4. Opzet van deze bachelorproef

Tijdens dit onderzoek zal eerst een uitgebreide literatuurstudie verricht worden naar de bestaande technologieën en hun problematiek. Naast literatuur zal hier voor ook de hulp ingeschakeld worden van het consultancy bedrijf 'Eleven Ways', zij specialiseren in digitale toegankelijkheid en zullen in de vorm van een persoonlijk gesprek hulp bieden bij het identificeren van de belangrijkste problemen. Hierna zal bovenop de open-source Python library 'GazeTracking' (lamé, [z.d.](#)) de gevonden oplossingen geïmplementeerd worden.

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeks domein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken. Daarbij wordt ook verduidelijkt hoe de gevonden oplossingen bijdragen aan het systeem.

In Hoofdstuk 4 wordt de technische verwezenlijking van dit onderzoek toegelicht.

In Hoofdstuk 5 wordt een experiment uitgevoerd met de eigen technische verwezenlijking en de open-source Python library 'GazeTracking'. Tijdens dit experiment wordt data over precisie en tijd bijgehouden in verschillende lichtomstandigheden.

In Hoofdstuk 6 wordt de data verzameld uit het experiment geanalyseerd om een antwoord te kunnen formuleren op de onderzoeks vragen

In Hoofdstuk 7, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeks vragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2

Stand van zaken

2.1. Waarom oog-besturing

In de studie van Spiller e.a. (2019) namen veertien personen, allemaal gediagnosticeerd met hersenverlamming, deel aan een experiment waarbij de taak was om zo snel mogelijk een doelwit op een computerscherm te activeren. Tijdens dit experiment werden drie verschillende invoerapparaten gebruikt: een computermuis, een touchscreen en een 'Tobii PCEye GO eye tracking mouse'. De verzamelde data werd achteraf geanalyseerd, waaruit geconcludeerd kon worden dat de 'Tobii PCEye GO eye tracking mouse' een positief effect had. Bij het uitvoeren van de taak werden minder fouten gemaakt, maar de tijd om de taak te voltooien was langer. Dit kan erop wijzen dat eindgebruikers voldoende oefening nodig hebben met deze technologie om deze efficiënt te kunnen gebruiken.

In het onderzoek van Lakshmi Pavani e.a. (2018) werd oog-tracerende software gebruikt om de cursor te controleren met de ogen in plaats van met een traditionele computermuis. Voor het onderzoek werd een eigen website gecreëerd waarop de gebruiker moest navigeren. Van de 20 experimenten waren 15 een succes, en werd er een nauwkeurigheid van 75% verkregen. Hieruit blijkt dat oogbesturing een groot voordeel kan bieden voor individuen met een motorische beperking bij webnavigatie.

Hiernaast concludeerde Van der Cruyssen e.a. (2023) dat webcam gebaseerde eye-trackers relevant kunnen zijn voor onderzoeken waarbij er geen extreem hoge precisie vereist is. Tijdens hun onderzoek probeerden ze verschillende voorgaande onderzoeken na te bootsen met behulp van webcam eye-trackers in plaats van eye-trackers uit een laboratorium. Ze merkten op dat de onderzoeken sneller, gemakkelijker en goedkoper konden verlopen. Deze voordelen gingen ten koste van

een lagere data kwaliteit maar het biedt bewijs dat er veel potentieel in de technologie zit. Daarnaast merkte Van der Cruyssen e.a. (2023) ook dat gezien de continue groei van technologie, de toekomst hiervoor veelbelovend is.

2.2. De onderliggende werking en hoe deze best vertaald worden in Python

Oog-tracerende technologie kent verschillende varianten met elk hun eigen onderliggende principes. In deze studie zal de focus liggen op webcam gebaseerde oog-tracing.

2.2.1. Theoretische werking

Het artikel van het bedrijf tobii (Miseviciute, [z.d.](#)), een van de wereldwijde marktleiders in eye-tracking, legt kort uit hoe deze technologie werkt. Eye trackers maken gebruik van sensor technologie die de positie en beweging van de ogen meet. Deze informatie kan dan gebruikt worden om te bepalen waar de ogen naar kijken, dit noemt men ‘the point of gaze’ oftewel het blikpunt. Verschillende vormen van eye-trackers hebben elk hun eigen manier om dit blikpunt te bepalen.

Zoals Jensen ([2022](#)) uitlegt in zijn online blog, is webcam gebaseerde oog-tracing puur video gebaseerd. De webcam moet in staat zijn de ogen en pupillen van de gebruiker te lokaliseren voordat andere acties ondernomen kunnen worden. Dit gebeurd aan de hand van 4 stappen:

1. Detecteren van de locatie van het gezicht.
2. Detecteren van de ogen op het gezicht.
3. Detecteren van de oriëntatie van zowel het linkse als rechtse oog.
4. De oriëntatie van de ogen in kaart brengen op het coördinaten systeem van het scherm.

Het paper geschreven door (Wolf & Seernani, [2023](#)) legt een gelijkaardige maar andere methode uit in hun paper:

1. Verwerking: ‘In deze stap wordt het individu’s blikpunt geschat door een ‘deep learning model’ dat op voorhand getraind werd.’
2. Kalibratie: ‘Gebaseerd op opnames wordt het algoritme gekalibreerd. Deze opnames zijn afkomstig vanuit stap 1’.
3. Projectie van data: ‘Aan de hand van het model dat werd opgebouwd in stap 2 kan het blikpunt geprojecteerd worden op het scherm in de vorm van x,y coördinaten. Dit proces gebeurd voor elk ‘frame’.

De studie van Taore en Dakin (2023) behandelde ook de moeilijkheden met webcam gebaseerde eye-trackers en volgde gelijkaardige stappen als het artikel van (Wolf & Seernani, 2023). De individuele 'frames' worden gebruikt om het gezicht, beide ogen en hun irissen te lokaliseren aan de hand van een detectie model. Taore en Dakin (2023) gebruikte hiervoor 'MediaPipe' maar hier bestaan verschillende technieken voor. Deze gegevens worden hierna omgezet in coördinaten waarop berekeningen uitgevoerd kunnen worden.

2.3. GazeTracking

GazeTracking is een webcam gebaseerde oogvolgsysteem gebouwd met Python door lamé (z.d.). Dit onderzoek zal op dit systeem verder bouwen met de verschillende oplossingen gevonden uit de rest van de literatuurstudie. Verder bouwende op de theorie verkregen uit het vorige deel wordt er uitgelegd hoe dit systeem te werk gaat en de verschillende klassen die gebruikt worden.

Wanneer men het systeem wil gebruiken is het vereist een nieuwe instantie te creëren zoals hieronder getoond wordt. Het systeem verkrijgt een frame van de webcam dankzij cv2, een Python library die verderop in dit onderzoek besproken wordt.

Listing 2.1: Het startpunt van de code voor GazeTracking

```
import cv2
from gaze_tracking import GazeTracking

gaze = GazeTracking()
webcam = cv2.VideoCapture(0)

_, frame = webcam.read()
gaze.refresh(frame)
```

Na het verkrijgen van een frame wordt de `refresh` functie van de klasse 'GazeTracking' aangeroepen. De klasse bevat 2 belangrijke attributen die het systeem zullen helpen met detectie.

_face_detector

Het object, afkomstig uit de dlib-library, is gecreëerd met behulp van een [Histogram of Oriented Gradients \(HOG\)](#) functie, in combinatie met een lineaire classificator, beeldpiramide en detectieschema voor schuifvensters. Wanneer een frame wordt gegeven aan deze detector retourneert het een lijst met rechthoeken waarin gezichten gedetecteerd zijn (King & Phillips, 2014a).

Het object is verkregen via onderstaande code:

Listing 2.2: Code uit GazeTracking waarin getoond wordt hoe het `_face_detector` attribuut verkregen wordt

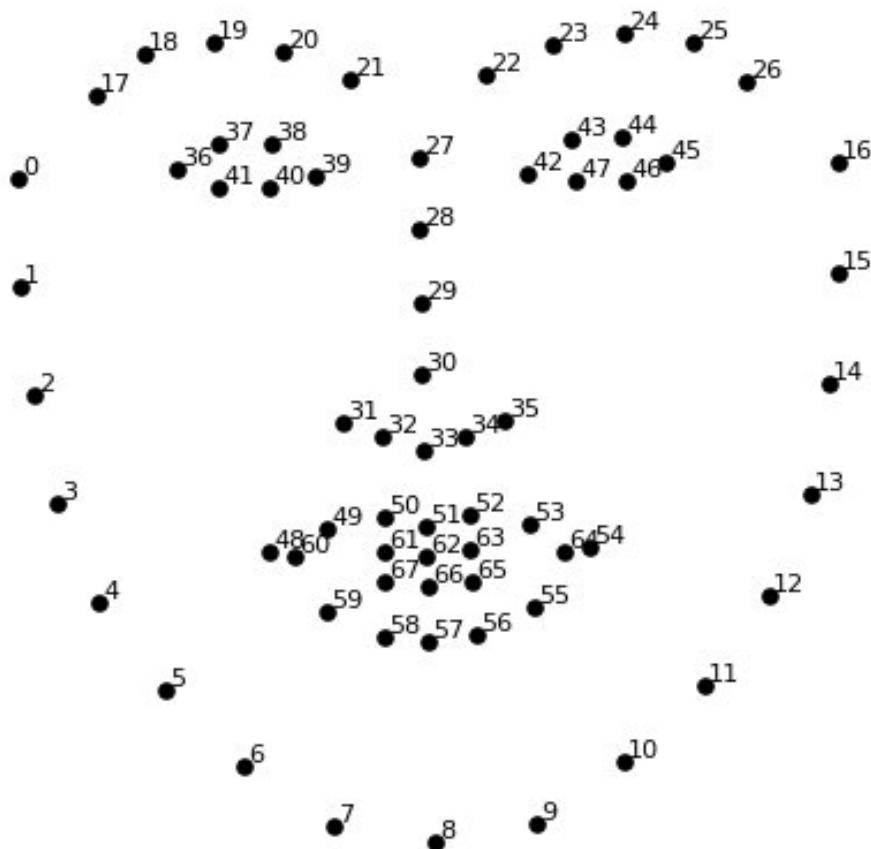
```
self._face_detector = dlib.get_frontal_face_detector()
```

_predictor

De `_predictor` is een object waaraan een afbeelding wordt gegeven, samen met het object dat zich binnen deze afbeelding bevindt. Vervolgens retourneert het een reeks puntlocaties die de houding van het object definiëren (King & Phillips, 2014b). Vooraleer dit object gecreëerd wordt in GazeTracking moet er een pad naar een model gedefinieerd worden. Dit model is vooraf getraind voor **Machine Learning (ML)** projecten. Specifiek bevat het model data met 68 specifieke punten om gezichtskenmerken te detecteren, we spreken van een 68-punten-topologie. Op afbeelding 2.1 is een visuele representatie van deze topologie te zien. Hieronder wordt weergegeven hoe de `_predictor` wordt gecreëerd binnen het GazeTracking-systeem.

Listing 2.3: Code uit GazeTracking waarin getoond wordt hoe het `_predictor` attribuut verkregen wordt

```
cwd = os.path.abspath(os.path.dirname(__file__))
model_path = os.path.abspath(os.path.join(cwd, "trained_models/shape_predictor_68_face_landmarks.dat"))
self._predictor = dlib.shape_predictor(model_path)
```



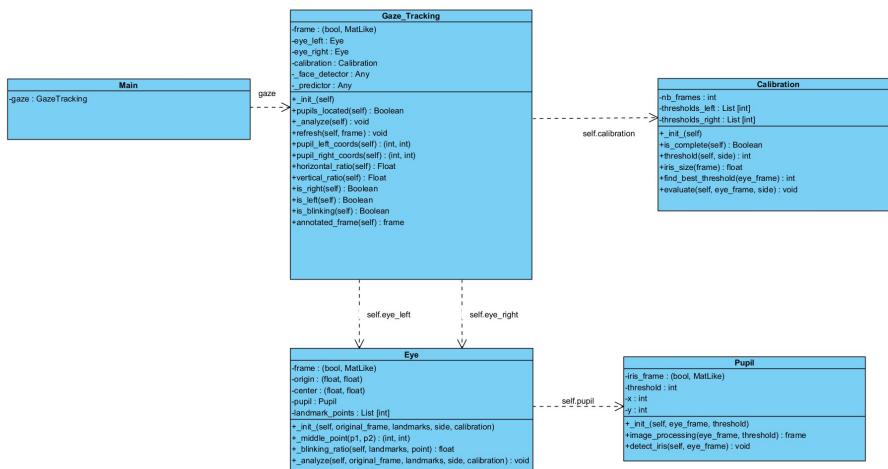
Figuur 2.1: Afbeelding van de 68 punten topologie verkregen uit de studie van Chakraborty e.a. (2019)

De refresh functie zet de variable ‘frame’ van de klasse gaze_tracking op het gege-

ven frame en roept vervolgens de bijhorende analyse functie aan. Binnenin deze functie wordt de eerder besproken _face_detector gebruikt om uit het frame gezichten te detecteren. De gereturneerde waarde van deze actie wordt opgeslagen in een variabele die op zijn beurt wordt meegegeven aan de _predictor samen met het frame. Zoals eerder vermeld zal deze _predictor de verschillende gezichtspunten proberen detecteren en retourneren. Op basis van deze punten worden er voor het linker-en rechteroog een oog-object gemaakt wordt. Aan elk oog-object wordt tevens ook een kalibratie-object meegegeven.

Bij het aanmaken van deze oog-objecten wordt het gegeven frame verder geanalyseerd aan de hand van verschillende wiskundige berekeningen zodat de ogen beter gedetecteerd kunnen worden. Naast deze berekeningen wordt er een kopie van het frame, waarop het oog geïsoleerd is, gestuurd naar het kalibratie object zodat de beste drempelwaarde berekend kan worden. Het is met behulp van deze drempelwaarde en verschillende berekeningen dat het systeem de irissen probeert te detecteren. Elk oog-object bevat een pupil-object dat informatie rondom de coördinaten van de irissen bezit. Uiteindelijk zijn het deze coördinaten waarmee bepaald zal worden, welke richting de gebruiker uitziet en wanneer de gebruiker knippert.

Op afbeelding 2.2 is een klassendiagram te vinden waarop de verschillende klassen met hun attributen en functies te zien zijn.



Figuur 2.2: Klassendiagram van het GazeTracking systeem dat de verschillende klassen met hun attributen en functies beschrijft. Het klassendiagram werd zelf gemaakt met behulp van het Visual Paradigm programma.

2.4. De huidige problemen met webcam gebaseerde oog-besturing

Eleven Ways is een adviesbureau dat digitale toegankelijkheidsdiensten levert aan overheidsorganisaties, productteams en digitale bureaus. In het kader van dit on-

derzoek werd een persoonlijk gesprek gevoerd met hen, waarin zij de problemen aankaartten waarvan zij op de hoogte zijn. Deze waren de volgende:

- een hoge graad van precisie is nodig
- een kwalitatieve webcam is vereist
- goede lichtomstandigheden zijn nodig
- de gebruikersinterface maakt complexere vormen van interactie zoals scrollen, slepen en dubbelklikken vaak niet toegankelijk.

Naast dit gesprek zijn er ook verschillende onderzoeken die de huidige problemen aantonen waarop dieper ingegaan zal worden.

2.4.1. Kalibratie problemen

De studie van Nyström e.a. (2012) legt uit dat kalibratie van de software zowel theoretische als praktische problemen met zich meebrengt. Vanuit theoretisch perspectief blijkt het moeilijk om een correct wiskundig model van de ogen te vinden waarop een functie kan worden uitgevoerd om de blikrichting te bepalen. Deze studie definieert ook 3 mogelijke kalibratie manieren langs de praktische kant:

- 'System-controlled calibration'. Bij deze methode maakt een algoritme de keuze wanneer de ogen gefocust en gericht naar een doelwit zijn. Deze methode heeft als voordeel dat de kalibratie volledig automatisch gebeurd.
- 'Operator-controlled calibration'. De operator bepaalt de kalibratie wanneer hij of zij het gevoel heeft dat de gebruiker gefocust is op het doelwit.
- 'Participant-controlled calibration' Deze derde methode plaatst meer verantwoordelijkheid bij de gebruiker. Wanneer hij of zij het gevoel heeft gefocust te zijn op een doelwit wordt een knop ingeduwd.

De huidige trend is om steeds meer controle te geven aan het systeem wanneer het gaat over kalibratie. Deze methode heeft als voordeel dat de kalibratie snel en makkelijk kan gebeuren. (Nyström e.a., 2012)

2.4.2. Precisie

De studie van Van der Cruyssen e.a. (2023) had als doel om 3 verschillende eye-tracking studies te repliceren met een webcam gebaseerde eye-tracker en concludeerde er een significant verschil was op de data verzameld door een webcam gebaseerde eye-tracker tegenover data verzameld in een laboratorium.

De studie van Kaduk e.a. (2023) vergelijkt een nieuw webcam gebaseerd eye-tracking systeem met dat vanuit een laboratorium. Hun resultaten concludeerden dat het webcam gebaseerd systeem een 0.5° grotere foutmarge zag. Hiernaast ondervond

de studie ook negatieve gevolgen van hoofdbewegingen tijdens het traceren van de ogen.

Een ondervinding die gedeeld wordt door het paper van Wolf en Seernani ([2023](#)). In hun onderzoek vond het team verschillende beperkingen van webcam gebaseerde eye-trackers waaronder: lichtcondities, brillen, lage resolutie camera's en hoofdbewegingen.

Het online artikel van Jensen ([2021](#)) bevestigt dat webcam gebaseerde eye-trackers over het algemeen een lagere precisie hebben. Naast deze bevestiging probeert dit artikel ook in te gaan op hoe andere factoren een negatieve impact kunnen hebben op de kwaliteit: lichtgevoeligheid, kalibratie en de kwaliteit van de webcam. Gezien webcams oorspronkelijk niet ontworpen werden om eye-trackers te zijn kan de hardware een negatieve invloed hebben op de verkregen data.

2.4.3. Lichtgevoelig

De studie van Salari en Bednarik ([2024](#)) onderzocht de invloed van verschillende lichtomstandigheden en hun effect op de nauwkeurigheid van oog-tracerende software. Er werd gekozen om drie verschillende omstandigheden te testen: donker, normaal en fel licht. Negen deelnemers voerde nauwkeurigheidstesten uit onder deze verschillende omstandigheden. Hieruit bleek dat de lichtomstandigheid wel degelijk een invloed had op de nauwkeurigheid; normale of gematigde lichtomstandigheden bleken het beste te werken voor dit soort software. Een belangrijke opmerking uit dit onderzoek was dat de uitvoering van de oogbesturing best gebeurd met dezelfde lichtomstandigheden als tijdens de kalibratie.

Ook in de paper van Wolf en Seernani ([2023](#)) wordt licht genoemd als een mogelijke factor voor een verlaging in kwaliteit van de data. Zij concludeerden dat intens zijlicht de precisie verlaagt. Daarnaast benadrukken Wolf en Seernani ([2023](#)) het belang van consistente lichtomstandigheden. In hun paper raden ze aan de kalibratie te doen in dezelfde lichtomstandigheden als het effectieve gebruik. Indien dit niet gebeurt kunnen er afwijkingen ontstaan.

2.4.4. Andere uitdagingen op het gebied van bruikbaarheid

Langdurig kijken naar computerschermen kan een negatieve impact hebben op de gezondheid van gebruikers. De ogen van de gebruiker raken vermoeid of kunnen droog aanvoelen (Blehm e.a., [2005](#)). Gezien oogbestuurde software afhankelijk is van de ogen van de eindgebruiker kunnen vermoeide ogen lijden tot een verminderde graad van precisie.

Hiernaast blijkt dat het dragen van een bril een negatieve impact kan veroorzaken op de verkregen data. Gebruikers die een bril dragen hebben gemiddeld een lagere precisie dan gebruikers zonder bril. (Wolf & Seernani, [2023](#))

2.5. Literatuur rondom relevante oplossingen

Verschillende studies rondom webcam eye tracking bieden niet alleen beperkingen van de technologie maar ook methodes hoe deze opgelost kunnen worden. Hieronder wordt de theoretische kant van relevante oplossingen en methodes besproken. Hoe deze elementen gebruikt relevant zijn voor dit onderzoek wordt later uitgelegd.

2.5.1. Essentiële Python-packages

Online is een grote hoeveelheid van Python projecten en packages beschikbaar. Hieronder is een oplijsting te vinden van de meest essentiële packages en wat hun functie is.

MediaPipe

MediaPipe bied een reeks bibliotheken en tools aan voor het gebruik van [Artificiële intelligentie \(AI\)](#) en [ML](#) (G. Developers, 2024). Het is een package die zijn waarde in oog tracerende software al bewezen heeft in de studie van Taore en Dakin (2023). Dit onderzoek zal gebruik maken van de gezichtsherkenning module met het 'Face mesh model'. Dit model bied een topologie van 478 gezichtsherkenningspunten. De basisimplementatie van GazeTracking had een 68-punten-topologie voor gezichtsherkenning. Theoretisch gezien zou dit moeten resulteren in een hogere nauwkeurigheid bij het detecteren van gezichten en bijhorende gelaatskenmerken.

Een visuele representatie van de 'Face mesh model' is te zien op afbeelding 2.3.

cv2

cv2 is de Python library van [Open Source Computer Vision \(OPENCV\)](#). OPENCV die de mogelijkheid bied om gemakkelijk met foto's en video's te werken binnenin Python. (Contributors, 2024)

PyAutoGUI

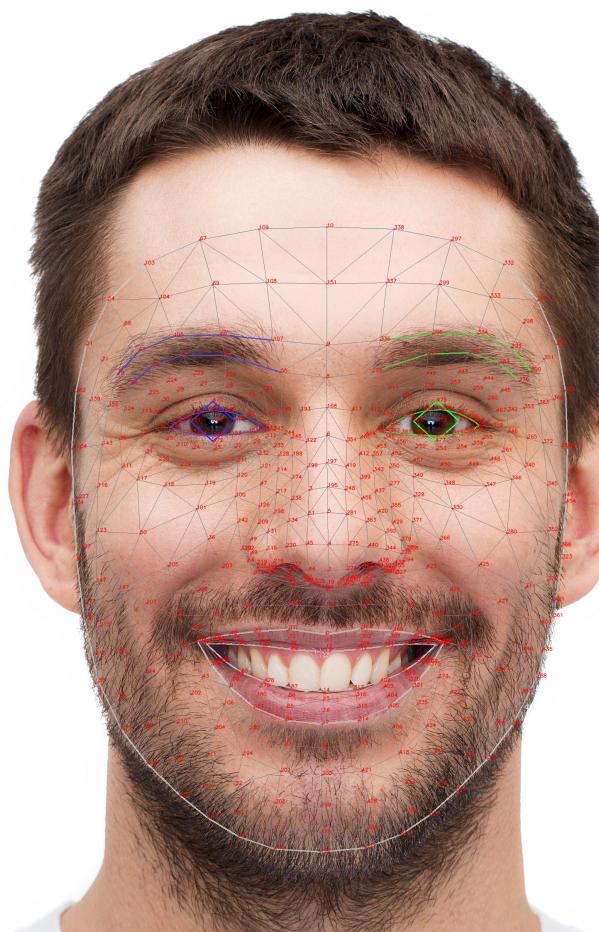
PyAutoGUI bied Python de functionaliteit om de muis over te nemen. Het is deze package dat de muis zal bewegen en de klik functie zal uitvoeren. (Sweigart, z.d.)

NumPy

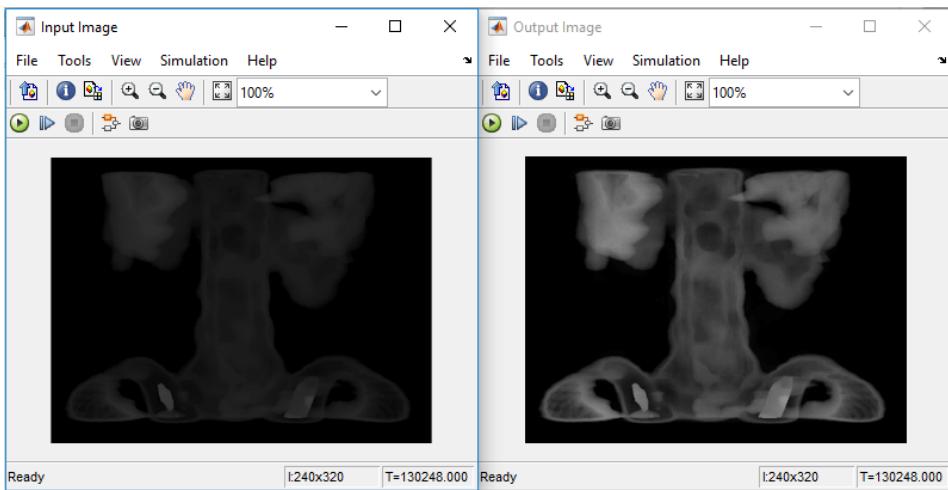
NumPy is een Python package dat essentieel is voor verscheidene wiskundige functies en werken met arrays. (N. Developers, 2024)

2.5.2. CLAHE

CLAHE (Contrast Limited Adaptive Histogram Equalization) is een beeldverwerkings-techniek waarbij het contrast van een afbeelding lokaal verbetert wordt. Het werkt door het invoerbeeld op te verdelen in kleine rechthoekige gebieden van 8x8 (Contributors, 2025). Hierna wordt op elk blok de techniek toegepast om het contrast



Figuur 2.3: Figuur represeneert de topologie van de MediaPipe face mesh, verkregen uit de documentatie van MediaPipe https://ai.google.dev/edge/mediapipe/solutions/vision/face_andmarker



Figuur 2.4: Input en output van het CLAHE model, afbeelding verkregen van het mathworks artikel op 30/04/2025 (MathWorks, 2025)

te verbeteren. Om te voorkomen dat sommige gebieden té veel versterkt worden, wordt een cliplimiet ingesteld. Histogramwaarden boven dit limiet worden afgesneden en herverdeeld. De afzonderlijk bewerkte blokken worden vervolgens op een vloeiente manier samengevoegd via bilineaire interpolatie, zodat er geen harde overgangen ontstaan. (MathWorks, 2025)

Op afbeelding 2.4 toont de input figuur en de output figuur van de CLAHE techniek. Hierop is te zien hoe het contrast verbeterd is zonder oververzadiging van de afbeelding.(MathWorks, 2025)

2.5.3. Hoofdtracing met een virtuele kinsteun

De studie van Kaduk e.a. (2023) maakte gebruik van hoofdtracing. Om dit te doen werd een algoritme met een neuraal netwerk gebruikt. Dit algoritme analyseerde de videobeelden door verschillende lagen van het neuraal netwerk. Hierdoor leert het algoritme de verschillende kenmerken van het menselijk gezicht met als gevolg dat het hoofd getraceerd kan worden en de ogen binnen een bepaald gebied kunnen blijven. Deze functionaliteit is vooral behulpzaam in combinatie met de ‘virtuele kinsteun’ die in dezelfde studie gebruikt werd.

De virtuele kinsteun is een simulatie van een fysieke kinsteun. Aan het begin van de studie koos de gebruiker een comfortabele positie en wanneer de persoon hiervan weg bewoog vroeg het systeem om terug te keren naar de vooraf gekozen positie. Dit aan de hand van instructies en feedback die getoond werden op het scherm (Kaduk e.a., 2023).

2.6. Onderzochte alternatieven

2.6.1. Object detectie

Taore en Dakin (2023) legt in hun paper uit hoe hun onderzoek met het YOLO ('You Only Look Once') object herkenningsmodel de aanhankelijkheid van kalibratie kan verminderen en hoe er rekening kan gehouden worden met een breder spectrum aan hoofd bewegingen. Hun versie van het YOLO model zorgt voor een beter begrip van de ruimtelijke verhoudingen tussen de ogen, irissen en het hoofd. Hierdoor wordt voorspellen van het blikpunt verstrekt. Volgens de studie is het model zelf in staat om onafhankelijke voorspellingen te doen voor zowel het linkse als rechtse oog waardoor de eye-tracking toegankelijker wordt. De resultaten werden vergelijken met 'WebGazer', een populaire eye-tracker, en de precisie van hun webcam gebaseerde eye-tracker lag hoger dan die van WebGazer.

2.6.2. Blikpunt detectie

Om het blikpunt van de gebruiker te bepalen werden verschillende alternatieven onderzocht.

RT-GENE

RT-GENE is een geavanceerde methode voor het inschatten van oogbeweging en het blikpunt van de gebruiker. Gecreëerd door Fischer e.a. (2018), het systeem maakt gebruik van convolutionele neurale netwerken om te bepalen waar de gebruiker naar kijkt. RT-GENE bevat meer dan 120 000 gelabelde beelden van gezichten met bijhorende oog-en hoofdoriëntatie, opgenomen in natuurlijke omstandigheden. Deze data vormt de basis waarop de algoritmes getraind zijn. In essentie is RT-GENE een combinatie van dataset, algoritme en methode om oogbewegingen zo accuraat mogelijk te voorspellen in natuurlijke omstandigheden.

De waarde van het systeem is al bewezen: op het moment van dit onderzoek staat het vierde model op de derde plaats in de rangschikking op de website 'paperswithcode'. (with Code, 2025) RT-GENE ging initieel een cruciale rol spelen in dit onderzoek maar wegens de complexiteit van dit systeem werd eerst een persoonlijk gesprek gevoerd met dokter Tobias Fischer, de bedenker van RT-GENE. Tijdens dit gesprek gaf Tobias het advies dat RT-GENE niet geschikt was voor dit onderzoek en dat er gekeken moest worden naar andere systemen. Tobias is zelf op zoek gegaan naar voorbeeldsystemen en gaf deze ook aan.

I-VT

In het onderzoek van Salvucci en Goldberg (2000) worden vijf verschillende algoritme besproken, een daarvan is het snelheid gebaseerd algoritme **Velocity-Threshold Identification (I-VT)**. I-VT onderscheid fixaties en saccade punten aan de hand van hun punt tot punt snelheid. Salvucci en Goldberg (2000) concluderen in hun onderzoek dat, hoewel het mogelijk is, I-VT en andere snelheidsgebaseerde algoritmen

niet geschikt zijn voor fixatie-identificatie wegens hun ‘sterke fysieke en fysiologische basis’ (Salvucci & Goldberg, 2000).

I-DT

Naast I-VT werd in het onderzoek van Salvucci en Goldberg (2000) ook het algoritme **Dispersion-Threshold Identification (I-DT)** besproken. Deze methode maakt gebruik van het principe dat fixatie punten de neiging hebben dichtbij elkaar te liggen. I-DT detecteert potentiële fixaties met behulp van een schuivend venster. Dit venster omvat verschillende datapunten, waarna de spreiding van deze punten wordt gecontroleerd door het verschil tussen maximale en minimale waarden te berekenen. Als de spreiding boven een vooraf opgestelde drempelwaarde ligt, wordt het venster niet als fixatie beschouwd en schuift het naar rechts. Wanneer de spreiding onder de drempelwaarde blijft, wordt dit geïnterpreteerd als een fixatie.

2.6.3. kalibratie

Om de precisie te verhogen is er onderzoek verricht naar verschillende manieren van kalibratie.

Symbolische regressie

‘**Symbolische Regressie (SR)** is het geautomatiseerd zoeken in een functiemodel, samen met de intrinsieke parameters ervan, dat de schatting van uitvoerwaarden afleidt uit invoerwaarden’ (Hassoumi e.a., 2019). In de studie van Hassoumi e.a. (2019) werd onderzocht of SR gebruikt kon worden om de nauwkeurigheid van kalibratie in oog-tracerende software te verbeteren. In hun studie werden vier verschillende kalibratieprocedures onderzocht en met elkaar vergeleken. Hieruit werd geconcludeerd dat SR een positief effect heeft op de nauwkeurigheid van de kalibratie, al gaat dit ten koste van extra rekentijd.

9-punt kalibratie

De studie van Drewes e.a. (2019) vergeleek een kalibratie systeem waarbij de gebruiker een bewegend object met hun ogen volgden. De resultaten van deze methode werden vergeleken met een 9-punt kalibratiesysteem, waarbij de gebruiker naar 9 vooraf opgestelde punten kijkt om informatie te verzamelen over het blikpunt. Drewes e.a. (2019) concludeerde dat een 9-punt kalibratie de beste nauwkeurigheid oplevert. Dit gaat echter ten koste van de tijdsefficiëntie, aangezien het kalibratieproces iets langer duurt.

De tijdens de kalibratie verkregen informatie kan worden omgezet via polynomiale mappingfuncties. Uit het onderzoek van Cerrolaza e.a. (2008) blijkt dat tweede-graads polynomen de beste balans bieden tussen nauwkeurigheid en rekenefficiëntie bij het vertalen van oogkenmerken naar schermcoördinaten. De bekomen data uit de kalibratie wordt hierbij gemapt op een polynomiaal model.

2.6.4. Python-packages

GazeParser

GazeParser is een ‘open-source’ bibliotheek geschreven in Python. Het bevat een video gebaseerde eye-tracker en data analyse. Om te beoordelen hoe performant GazeParser is werden verschillende experimenten uitgevoerd in de studie van Sogo ([2012](#)) met zowel GazeParser en Eyelink. Eyelink is een commercieel beschikbare eye-tracker. Hieruit werd geconcludeerd dat GazeParser op een betrouwbare manier de latentie en amplitude van [saccade](#) kan meten. Toch trad er bij twee deelnemers incidenteel een fout op bij de detectie van de blikpositie door GazeParser, terwijl Eyelink deze blikpositie wel consistent kon registreren. Dit verschil is waarschijnlijk toe te schrijven aan verschillen in de methoden voor oogpositiedetectie ([Sogo, 2012](#)).

PyTorch

in de studie van Taore en Dakin ([2023](#)) werd PyTorch gebruikt om een neuraal netwerk op te zetten voor blikpunt voorspellingen. Dit deden ze volgens de instructies van Valliappan e.a. ([2020](#)). De resultaten van hun YOLO netwerk werd achteraf vergeleken met de resultaten van dit netwerk.

3

Methodologie

Het onderzoek is opgebouwd in zes fasen, die samen de basis vormen voor het ontwerpen, testen en evalueren van een verbeterde oogbesturing.

3.1. Literatuurstudie

Tijdens de literatuurstudie zal grondig onderzoek gedaan worden naar hoe oogbesturing in zijn werk gaat, de grootste problemen rondom deze technologie en mogelijke oplossingen. Naast bestaande literatuur te onderzoeken wordt er een persoonlijk gesprek gehouden met het Eleven ways, een bedrijf dat specialiseert in digitale toegankelijkheid. Hun expertise zal helpen in het vinden van bestaande problemen die gebruikers van oog-tracerende software ondervinden.

3.2. Creëren van een gecontroleerde testomgeving

In deze fase van het onderzoek, die parallel loopt met de literatuurstudie, wordt een gecontroleerde testomgeving ontwikkeld, geïnspireerd door de methodologie van Lakshmi Pavani e.a. (2018). Deze testomgeving bestaat uit een website die functioneert als een gesimuleerde webshop. Deze omgeving wordt ontwikkeld met behulp van de programmeertalen HTML en CSS. Deze omgeving zal dienen als testplatform voor de evaluatie van de applicatie. Door een uniforme testomgeving te gebruiken, wordt consistentie in de onderzoeksresultaten gewaarborgd. De gesimuleerde webshop zal bestaan uit drie pagina's, waarbij elke pagina is opgedeeld in drie secties: boven, midden en onder. De testomgeving wordt gehost via GitHub Pages. Dit biedt twee belangrijke voordelen. Ten eerste is GitHub Pages gratis, waardoor de kost geen belemmerende factor vormt. Ten tweede is de testomgeving publiek toegankelijk. Hierdoor kunnen externe tools rechtstreeks op de omgeving worden toegepast. Dit is essentieel voor het genereren van de heatmap in de hieropvolgende fasen van dit onderzoek.

3.3. Creëren van een oplossing

Tijdens deze fase van het onderzoek worden oplossingen uitgewerkt voor de problemen die uit de literatuurstudie naar voren kwamen. Voor de implementatie wordt de programmeertaal Python gebruikt, vanwege het brede aanbod aan bibliotheken met bestaande code die de ontwikkeling van dergelijke toepassingen vereenvoudigen. Het systeem wordt parallel aan het onderzoek aangevuld met deze oplossingen, die zowel uit de literatuurstudie als uit eigen inzichten voortkomen. Gedurende deze fase is er een persoonlijk gesprek gevoerd met dokter Tobias Fischer, Hoofddocent aan de Queensland University of Technology. Tobias Fischer bood advies aan rondom blikschutting en RT-GENE. Het resultaat van deze fase wordt uitgebreid toegelicht in Hoofdstuk 4.

3.4. Experiment uitvoeren en data verzamelen

Nadat de technische verwezenlijking van dit onderzoek compleet is, kan deze fase van start gaan. Hierin zal het systeem getest worden door twee proefpersonen: één zonder motorische beperkingen en één met motorische beperkingen. Beide proefpersonen zullen in verschillende lichtomstandigheden navigatietaakjes uitvoeren op de gesimuleerde webshop. In totaal zullen de proefpersonen op negen elementen moeten klikken. Eerst zal er gewerkt worden met GazeTracking, het startpunt van dit onderzoek en het systeem zonder de geïmplementeerde oplossingen, vervolgens wordt er gewerkt met het systeem gecreëerd tijdens de vorige fase. Gedurende dit experiment worden de volgende gegevens verzameld:

- De afwijking van kliks buiten de doelgebieden, gemeten als de Euclidische afstand tot het correcte element.
- De benodigde tijd om per element een klik te voltooien, gemeten in seconden met behulp van een chronometer.
- De invloed van verschillende lichtomstandigheden.

Deze gegevens zullen inzichten bieden in de effectiviteit van geïmplementeerde verbeteringen. Tijdens het uitvoeren van dit experiment zal Hotjar gegevens verzamelen over de muisbewegingen met als doel een heatmap te kunnen genereren.

3.5. Analyse

Deze voorlaatste fase omvat de analyse van de gegevens die in de vorige fase zijn verzameld. De resultaten van beide proefpersonen en systemen worden omgezet in tabellen om een overzichtelijk beeld te creëren. Daarnaast worden de gegevens besproken, met als doel inzicht te geven in wat ze representeren en hoe ze geïnterpreteerd kunnen worden. Hiernaast wordt per proefpersoon ook de door Hotjar gegenereerde heatmap besproken. Deze heatmap biedt inzicht in hoe accuraat

en soepel de muisbewegingen zijn. Ook de gebruikte berekeningen worden verder toegelicht en besproken.

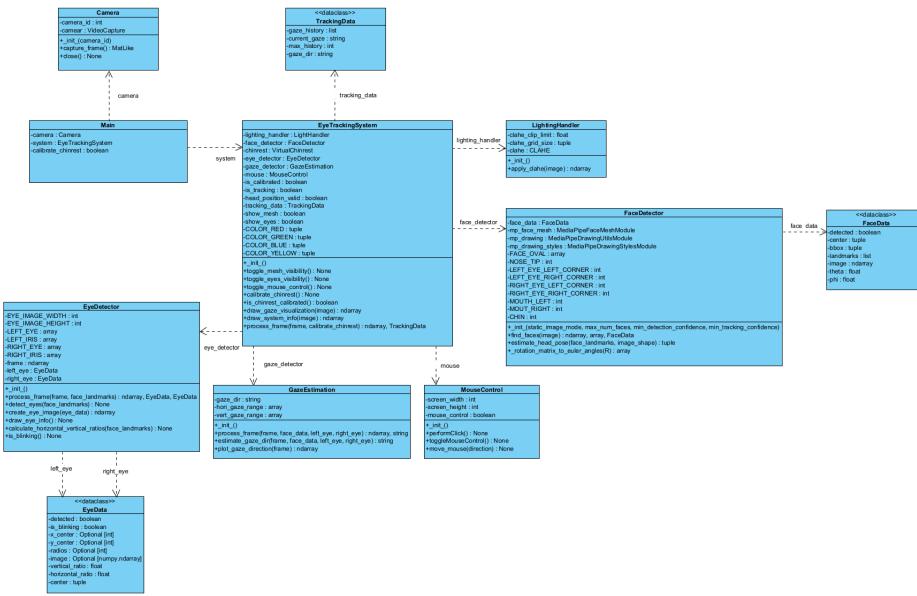
3.6. Conclusie en discussie

Tijdens de laatste fase van dit onderzoek worden de gebruikte technieken en implementaties samengevat. Vervolgens wordt een conclusie geformuleerd op basis van de resultaten uit de analyse fase met als doel een antwoord op te bieden op de hoofdvraag van dit onderzoek. Hiernaast wordt er gereflecteerd over welke factoren invloed hadden op dit onderzoek, en hoe aanpassingen aan deze factoren tot andere resultaten zouden kunnen leiden.

4

POC

In dit hoofdstuk wordt de technische verwezenlijking in detail besproken. De volledige implementatie en code is te vinden op <https://github.com/Miel-DR2003/Poc-Bachelorproef-2023>. Op afbeelding 4.1 is een klassendiagram te zien zodat er snel een overzicht verkregen worden van de verschillende klassen met hun attributen en methodes.



Figuur 4.1: Zelfgemaakte afbeelding van de technische verwezenlijking van dit onderzoek met daarop alle klassen met hun methodes en attributen. Het klassendiagram werd zelf gemaakt met behulp van het Visual Paradigm programma

De technische verwezenlijking is gestart vanaf de Python-bibliotheek GazeTracking, deze werd al eerder besproken in de literatuurstudie. Hierna werden oplossingen die tijdens de literatuurstudie en experimenten werden geïdentificeerd, geïmplementeerd om een oogbesturingssysteem te creëren dat de richting van het

blikpunt van de gebruiker detecteert en op basis daarvan de muis in dezelfde richting beweegt. Hoewel kalibratie uitgebreid werd behandeld in de literatuurstudie, is ervoor gekozen om deze functionaliteit niet te implementeren vanwege de bijbehorende technische complexiteit.

In de rest van dit hoofdstuk wordt elke stap die een videoframe doorloopt tijdens de verwerking uitvoerig toegelicht, met als doel een duidelijk en gestructureerd overzicht te geven van het volledige proces binnen het oogbesturingssysteem.

Op het hoogste niveau van het systeem bevindt zich een lus die continu een frame opvraagt aan de webcam en doorgeeft aan een instantie van de klasse 'EyeTrackingSystem'. Het is dankzij de cv2 bibliotheek dat Python de webcam kan aanspreken en een frame kan verkrijgen.

Wanneer het systeem actief is, wordt het bewerkte frame continu getoond aan de gebruiker. Hierop is namelijk informatie geplaatst dat aantoont of het systeem werkt zoals verwacht. Dit bewerkte frame zal verderop in het onderzoek benoemd worden als **Bewerkt Frame (BF)**.

Binnenin deze lus bevindt zich ook het controle paneel van het systeem, er worden verschillende toetsen gedefinieerd die elk verschillende acties uitvoeren. Hieronder is een overzicht te zien van elke toets en hun actie:

- q: deze toets stopt het systeem.
- m: deze toets schakelt de muisbesturing aan en uit.
- c: deze toets kalibreert de virtuele kinstuur.
- e: deze toets schakelt de zichtbaarheid van de oog detectie op het **BF**.
- t: deze toets schakelt de zichtbaarheid van de, door mediapipe verkregen, face mesh op het **BF**.

4.1. EyeTrackingSystem

Deze klasse dient als het centrale doorgeeflink en besturingscomponent. De vooraf vermelde lus zal een frame doorgeven aan de functie 'process_frame'. Deze start met een kopie te maken van het verkregen frame zodat dit zonder problemen bewerkt kan worden. Hierna spreekt het systeem verschillende functies van verschillende klassen aan. Het frame doorloopt de volgende stappen:

1. Pas **CLAHE** toe.
2. Detecteer het hoofd van de gebruiker.
3. Controleer de virtuele kinstuur.
 - (a) Als de kalibratie nog niet is gebeurd en hierom wordt gevraagd, voer dan de kalibratie van de kinstuur uit.

- (b) Als de kalibratie al is gebeurd, controleer dan of het hoofd zich nog steeds in de correcte positie bevindt.
4. Detecteer de ogen van de gebruiker, dit gebeurt enkel als de vorige stap bepaalt heeft dat het hoofd zich op een correcte positie bevindt.
 5. Detecteer de richting van het blikpunt van de gebruiker, dit gebeurt enkel als het hoofd zich op een correcte positie bevindt.
 6. Beweeg de muis in de juiste richting, dit gebeurt enkel als het hoofd zich op een correcte positie bevindt en de gebruiker muisbesturing aangeschakeld heeft.
 7. De muis voert een klik actie uit, dit gebeurt enkel als alle voorgaande stappen correct doorlopen zijn en het systeem heeft gedetecteerd dat de gebruiker geknippert heeft.

Tussenin elke stap wordt de verkregen informatie op het **BF** geplaatst.

4.1.1. CLAHE

Nadat er een kopie van het verkregen videoframe wordt gemaakt, wordt dit kopie gegeven aan de functie 'apply_clahe' van de klasse 'LightingHandler'. Deze klasse bevat een **CLAHE** element dat verkregen is via de Python bibliotheek cv2. Voordat **CLAHE** kan worden toegepast moet de videoframe omgezet worden naar grijswaarden. Hierna kan de **CLAHE** worden toegepast waarna de videoframe terug wordt omgezet naar **Blue-Green-Red (BGR)** formaat (Contributors, 2025). Het behandelde frame wordt hierna geretourneerd naar het besturingscomponent.

4.1.2. Object detectie

Het oogbesturingssysteem moet op een frame het gezicht en de ogen van een gebruiker kunnen detecteren. Om dit te doen gebruikt dit onderzoek de MediaPipe package (G. Developers, 2024).

Gezichtsdetectie

Het **BF** wordt na de **CLAHE** behandeling doorgegeven aan de functie 'find_faces' van de klasse 'FaceDetector'. Deze klasse bevat het 'face mesh' object verkregen vanuit MediaPipe. Het face mesh object is gecreëerd met de 'refine_landmarks' optie aan waardoor er in totaal 10 gezichtsherkenningspunten bijkomen, 5 per oog. Dit zorgt ervoor dat de irissen beter gedetecteerd kunnen worden en er in totaal 478 gezichtsherkenningspunten gebruikt kunnen worden (Subbiah, 2021). Hiernaast heeft de klasse ook een lijst van indexen uit de gezichtsherkenningspunten die gebaseerd zijn op het onderzoek van Alanazi e.a. (2023).

In de functie 'find_faces' wordt het verkregen videoframe verwerkt door het face mesh object, hieruit krijgt het systeem een resultaat dat bevat of dat er een gezicht

gedetecteerd is. Het gevonden gezicht bevat alle 478 punten. Na het detecteren van het gezicht wordt de functie ‘estimate_head_pose’ aangeroepen. Deze functie schat de gezichtspositie.

Alle verzamelde informatie wordt opgeslagen in het ‘face_data’ object, dit is een instantie van de ‘FaceData’ klasse. De verzamelde informatie bevat:

- een booleaanse waarde die aangeeft of het gezicht gedetecteerd is of niet.
- de x- en y-coördinaten van het midden van het gezicht.
- de x- en y-coördinaten van het minimum- en maximumpunt van het omkaderingsvak dat het gezicht bevat.
- een lijst van alle aanwezige gezichtsherkenningspunten.
- het frame dat verkregen werd toen de functie ‘find_faces’ werd opgeroepen.
- theta en phi, zijnde de hoofdrotatiehoeken die uitgedrukt zijn in radialen.

Nadat het gezicht is gedetecteerd en de nodige informatie is verzameld, werd deze data ook op het frame zelf weergegeven. Na het doorlopen van al deze stappen worden het bewerkte frame en de verkregen resultaten van de gezichtsdetectie teruggestuurd naar het besturingscomponent.

Virtuele kinsteun

Pas na het detecteren van het gezicht kan de virtuele kinsteun geïmplementeerd worden. De werking hiervan werd geïnspireerd door de studie van Kaduk e.a. (2023). Voor deze implementatie wordt de functie ‘process_frame’ van de klasse ‘VirtualChinrest’ aangeroepen. Deze functie heeft twee verschillende werkingen, afhankelijk van of de kalibratie al gebeurd is of niet. Zonder kalibratie van de virtuele kinsteun zal het systeem ook weigeren om de muisbeweging te gebruiken, wanneer dit het geval is ziet het BF eruit zoals op afbeelding 4.2

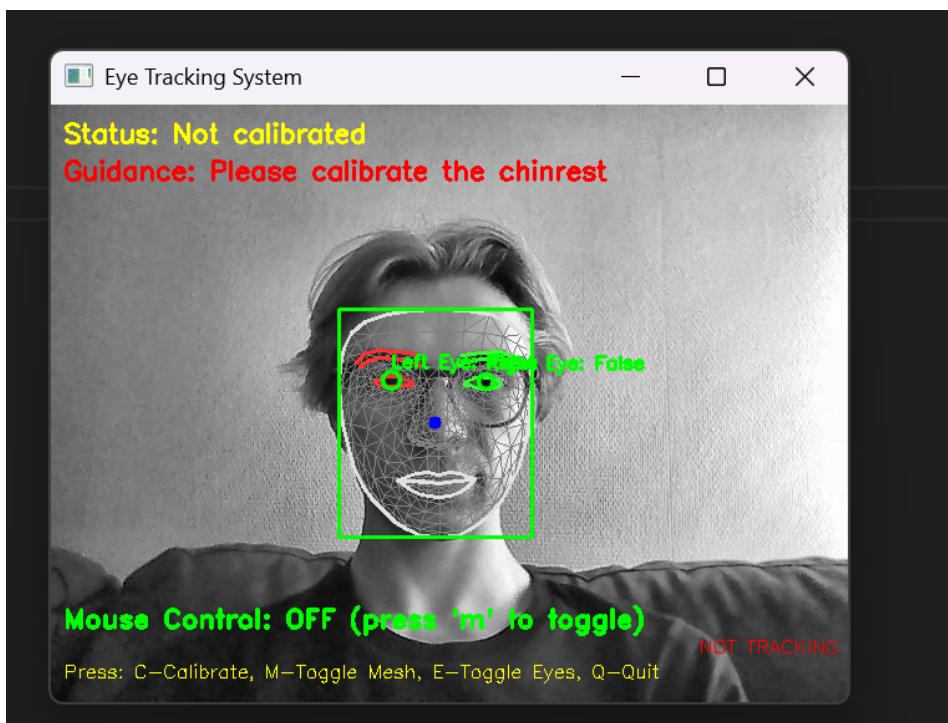
4.1.2.2.1 De kalibratie is nog niet gebeurd

Wanneer de kalibratie nog niet heeft plaatsgevonden maar de gebruiker deze wel aanvraagt dan wordt de functie ‘calibrate’ opgeroepen met de gezichtsdetectie-data uit de vorige stap. Deze functie gebruikt de verkregen data om referentiepunten op te stellen. Om de referentiepunten zo accuraat mogelijk te maken, wordt deze stap meermaals uitgevoerd met verschillende videoframes en wordt het gemiddelde berekend van de x- en y-coördinaten van het gezicht en het gemiddelde van de breedte en hoogte van het gezicht. Ook de relatieve afstand van de camera wordt berekend en bijgehouden.

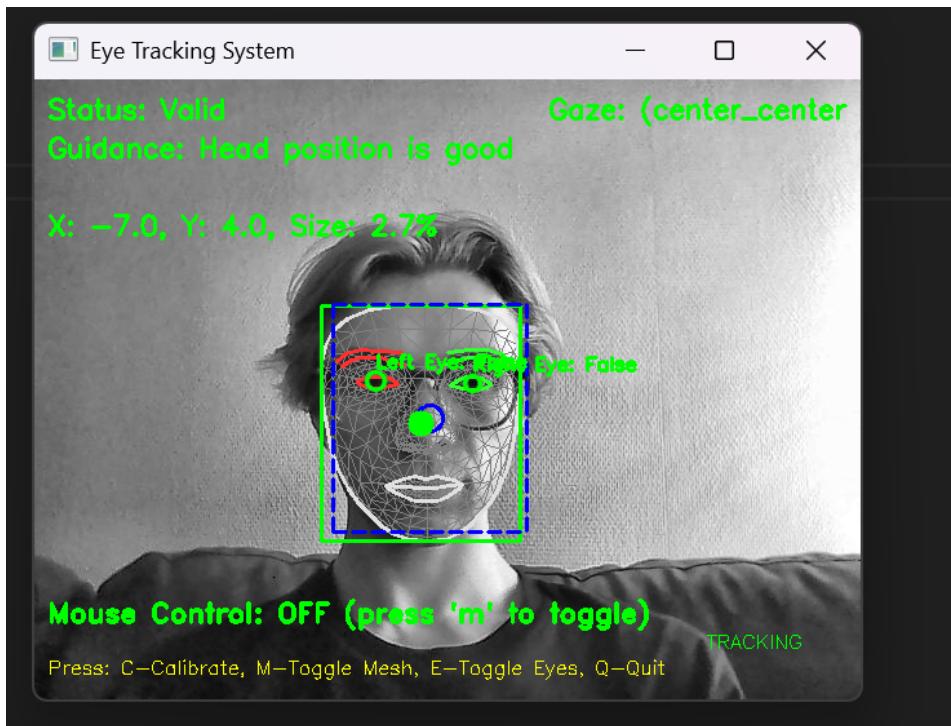
4.1.2.2 De kalibratie is al gebeurd

In het geval dat de kalibratie al gebeurd is, wordt de verkregen gezichtsdata gebruikt om de huidige positie van het gezicht te vergelijken met de gekalibreerde positie. De klasse bevat een tolerantie van 50 pixels, dit betekent dat het midden van het gezicht zich maximaal 50 pixels in elke richting naast de gekalibreerde positie mag bevinden. Op basis van deze data wordt de status op 'valid' of 'invalid' geplaatst. Afhankelijk van de status wordt er verschillende informatie afgebeeld op het BF. Daarnaast zal het systeem zijn werking op deze stap onderbreken totdat het centrum van het gezicht zich opnieuw binnen de tolerantie van de gekalibreerde positie bevindt.

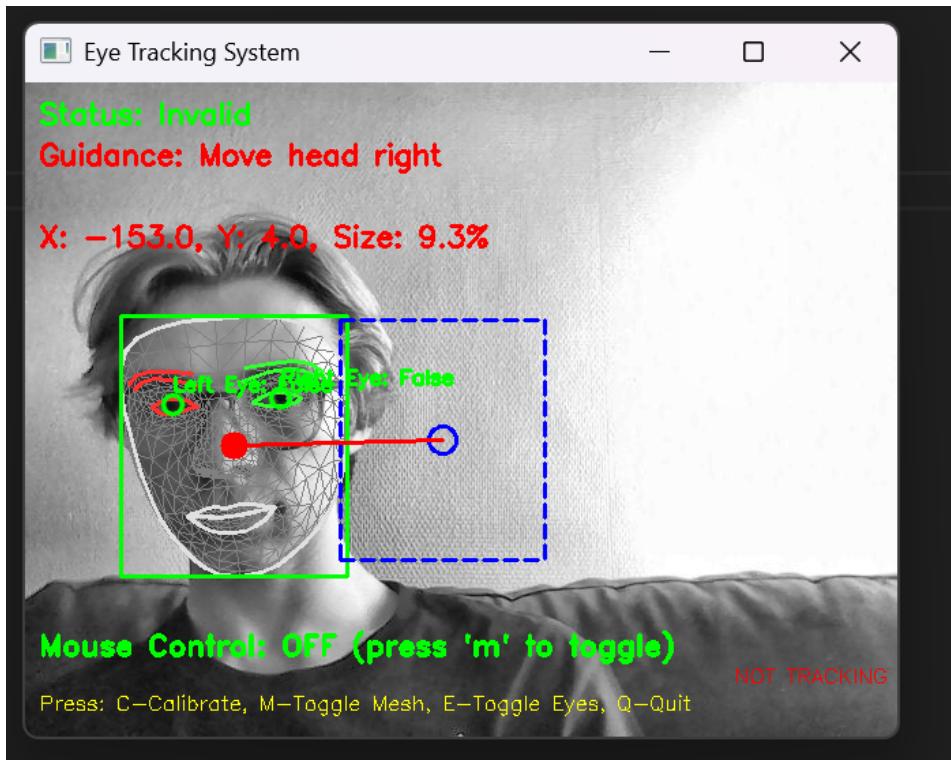
De virtuele kinsteen wordt geplaatst op het gegeven frame zodat de gebruiker kan kijken of het gezicht zich nog steeds op de juiste plaats bevindt. Dit is te zien op afbeelding 4.3. Wanneer dit niet langer het geval is, zal het systeem een lijn tekenen vanaf het gezicht naar het midden van de kinsteen, zodat de gebruiker weet hoe hij zijn hoofd moet bewegen om terug naar de ingestelde positie te komen. Deze werking is te zien op afbeelding 4.4.



Figuur 4.2: Zelfgemaakte afbeelding van de technische verwezenlijking waarop het BF te zien is zonder kalibratie van de virtuele kinsteen



Figuur 4.3: Zelfgemaakte afbeelding van de technische verwezenlijking waarop het BF te zien is met kalibratie van de virtuele kinsteun en het hoofd op de gekalibreerde positie



Figuur 4.4: Zelfgemaakte afbeelding van de technische verwezenlijking waarop het BF te zien is met kalibratie van de virtuele kinsteun en het hoofd niet op de gekalibreerde positie

Uiteindelijk wordt het **BF** en de status van de kinsteen geretourneerd naar het besturingscomponent.

Oogdetectie

De stap waarin de oogdetectie gebeurd enkel als de voorgaande stappen succesvol doorlopen zijn. De kinsteen moet dus de status 'valid' hebben voordat de ogen gedetecteerd kunnen worden. Tijdens deze stap wordt de functie 'process_frame' van de klasse 'EyeDetector' opgeroepen. Deze klasse bevat voor het linker- en rechteroog een instantie van de klasse 'EyeData' waarin volgende informatie wordt opgeslagen:

- een booleaanse waarde die representeert of het oog gedetecteerd is of niet.
- een booleaanse waarde die representeert of het oog knippert of niet.
- de x- en y-coördinaten van het centrum van het oog.
- de radius van het oog.
- een afbeelding van het oog.
- de verticale ratio van het oog.
- de horizontale ratio van het oog.

Om de x-en y-coördinaten van het oog zo accuraat mogelijk te bepalen wordt er gewerkt met de indexen van de irissen vanuit de MediaPipe topologie. De data verkregen uit de gezichtsdetectie wordt gebruikt met de indexen van de irissen met de 'minEnclosingCircle' functie van cv2. Deze functie retourneert de kleinste mogelijke cirkel die alle opgegeven punten omsluit (OpenCV.org, 2018). Met behulp van de verkregen data kan de rest van de nodige informatie berekend worden. Op het **BF** wordt een cirkel getekend rondom het gedetecteerde linker- en rechteroog. Het **BF** wordt samen met de verkregen data over het linker-en rechteroog geretourneerd naar het besturingscomponent.

Blikpunt detectie

Als voorlaatste stap in het systeem wordt alle voorgaand verzamelde data doorgegeven aan de functie 'process_frame' van de klasse 'GazeEstimation'. In deze functie wordt van beide ogen de horizontale en verticale ratio genomen en voor beide het rekenkundig gemiddelde berekent. Deze twee waarden worden vergeleken met vooraf gedefinieerde grenswaarden. Aan de hand van de horizontale ratio wordt bepaald of de gebruiker links, rechts of centraal kijkt. De verticale ratio wordt gebruikt om te bepalen of de gebruiker boven, beneden of centraal kijkt. Deze twee richtingen worden samengevoegd om een van de negen mogelijke combinaties te vormen:

- Links boven (LB)

- Links midden (LM)
- Links onder (LO)
- Rechts boven (RB)
- Rechts midden (RM)
- Rechts onder (RO)
- Midden boven (MB)
- Midden (M)
- Midden onder (MO)

De bepaalde richting wordt geplaatst op het BF en geretourneerd naar het besturingscomponent.

4.1.3. Muis beweging

De laatste stap in het systeem gebeurt enkel als de gebruiker de muisbesturing ingeschakeld heeft en de voorgaande stappen succesvol doorlopen zijn. Tijdens deze stap wordt de functie ‘move_mouse’ van de klasse ‘MouseControl’ aangesproken. Deze functie beweegt de muis aan de hand van de gekregen richting van het blikpunt. Het bewegen van de muis gebeurt met behulp van de PyAutoGUI package (Sweigart, z.d.) en op basis van onderstaande code.

Listing 4.1: Code uit technische verwezeling die bepaalt hoe de muis beweegt op basis van de opgegeven richting

```
match direction:  
    case "left_top":  
        pyautogui.moveRel(-10, -10)  
    case "left_bottom":  
        pyautogui.moveRel(-10, 10)  
    case "left_center":  
        pyautogui.moveRel(-10, 0)  
    case "right_top":  
        pyautogui.moveRel(10, -10)  
    case "right_bottom":  
        pyautogui.moveRel(10, 10)  
    case "right_center":  
        pyautogui.moveRel(10, 0)  
    case "centre_top":  
        pyautogui.moveRel(0, -10)  
    case "centre_bottom":  
        pyautogui.moveRel(0, 10)
```

```
case "centre_centre":  
    pyautogui.moveRel(0, 0)
```

Nadat de muisbeweging voor het frame voltooid is, controleert het systeem of de gebruiker knippert aan de hand van de booleaanse waarde verkregen uit de oog-detectie stap. Indien dit het geval is wordt de functie 'performClick' van de klasse 'MouseControl' gebruikt om de PyAutoGUI package een klikactie te laten uitvoeren met de muis.

5

Experiment

Tijdens dit hoofdstuk van het onderzoek wordt toegelicht hoe het uitgevoerde experiment werd opgesteld en uitgevoerd. Het experiment werd uitgevoerd met twee proefpersonen: één zonder motorische beperkingen en één met motorische beperkingen. Beide personen hadden vooraf aan dit onderzoek al ervaring met het gebruik van een traditionele toetsenbord en muis.

5.1. Testomgeving

Om de consistentie resultaten te garanderen werd een gecontroleerde testomgeving gecreëerd met de programmeertalen HTML en CSS. Deze testomgeving is een gesimuleerde webshop waarop de gebruiker kan navigeren naar een hoofdpagina, een pagina waarop appels gekocht kunnen worden en een pagina waarop peren gekocht kunnen worden. Deze omgeving werd gehost door middel van Github Pages en is te vinden via de volgende link: <https://miel-dr.github.io/controleWebsite/>. Elke pagina bevat een punt bovenaan, een punt in het midden en een punt onderaan. Voor de hoofdpagina liggen deze langs de linkerkant, voor de appelen pagina in het midden en voor de peren pagina liggen deze punten aan de rechterkant van het scherm. De posities van de knoppen zijn zo gekozen dat ze doen denken aan de kalibratiemethode van Cerrolaza e.a. (2008). Dit om een zo duidelijk mogelijk beeld te krijgen van de precisie van het systeem. Hieronder zijn afbeeldingen te vinden van de gesimuleerde webshop.

Meer info

Koop appelen

Koop peren

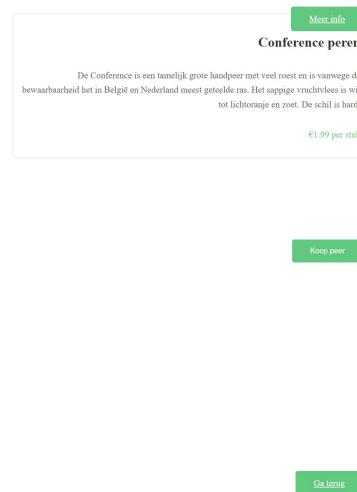
Figuur 5.1: Zelfgemaakte afbeelding van de homepagina van de gesimuleerde webshop.



Koop appelen

Ga terug

Figuur 5.2: Zelfgemaakte afbeelding van de appelen pagina van de gesimuleerde webshop.

**Figuur 5.3**

Zelfgemaakte afbeelding van de peren pagina van de gesimuleerde webshop.

In totaal bevat de testomgeving negen knoppen. De coördinaten van elke knop, afgerond op 2 decimalen, op een scherm met resolutie 2560 x 1600 zijn:

- LB : (x=20 , y=20)
- LM : (x=20 , y=800)
- LO : (x=20 , y=1580)
- MB : (x=1280 , y=20)
- M : (x=1280 , y=800)
- MB : (x=1280 , y=1580)
- RB : (x=2540 , y=20)
- RM : (x=2540 , y=800)
- RO : (x=2540 , y=1580)

5.2. Opstelling

Een laptop met daarop de technische verwezenlijking van dit onderzoek werd op ooghoogte van de proefpersoon geplaatst. Hiernaast was de enige lichtbron de Rexel ActiVita Pod+, een bureaulamp met vijf helderheidsinstellingen. Deze keuze werd gemaakt zodat het experiment in verschillende lichtomstandigheden uitgevoerd kon worden. Na het opzetten van de laptop en de bureaulamp werd de testomgeving geopend. Aan de start van het experiment wordt de muis in het midden van het scherm geplaatst om een zo accuraat mogelijke meting te garanderen.

5.3. Proefpersonen

Het hierboven beschreven experiment zal worden uitgevoerd door twee verschillende proefpersonen: één met een motorische beperking en één zonder.

5.3.1. Proefpersoon met motorische beperking

Het eerste proefpersoon lijdt aan **Developmental Coordination Disorder (DCD)**, ‘een motorische ontwikkelingsstoornis die leidt tot aanzienlijke moeilijkheden bij het verwerven en uitvoeren van zowel grove als fijne motorische vaardigheden’ Association (2013). De persoon in kwestie heeft voornamelijk moeite met fijne motoriek, DCD zorgt ervoor dat bepaalde toetsen of toetsencombinaties op een toetsenbord moeilijk gebruikt kunnen worden. Bijgevolg is blind typen een vaardigheid die moeilijk aangeleerd kon worden en jaren heeft geduurd.

5.3.2. Proefpersoon zonder motorische beperking

Het tweede proefpersoon in dit onderzoek heeft geen motorische beperking en kan gebruik maken van een traditioneel toetsenbord en muis.

5.4. Uitvoeren van het experiment

Aan de proefpersonen werd gevraagd om op de testomgeving navigatietaken uit te voeren met als doel elke knop gebruikt te hebben. Alle metingen zullen uitgevoerd worden in drie verschillende lichtomstandigheden. Om een zo accuraat en consistent mogelijk resultaat te verkrijgen worden andere lichtbronnen verduisterd. De mogelijke lichtomstandigheden met de corresponderende helderheidsinstelling is als volgt:

- Onderbelicht (helderheidsinstelling 1)
- Evenwichtig belicht (helderheidsinstelling 3)
- Overbelicht (helderheidsinstelling 5)

Wanneer een van de gevraagde navigatietaken uitgevoerd wordt, is het aan de gebruiker om via de ogen de muis naar de gewenste positie te bewegen. Nadat de muis in positie is, is het aan de gebruiker om een klik actie uit te voeren door met de ogen te knipperen. Wanneer deze actie uitgevoerd wordt, worden de coördinaten waarop geklikt werd verzameld zodat er in het analyse hoofdstuk de Euclidische afstand tot aan het correcte element berekend kan worden. Hiernaast zal voor elke taak de gespendeerde tijd bijgehouden worden, gemeten aan de hand van een chronometer. Elke navigatietaak heeft drie mogelijke uitkomsten:

- De taak werd correct uitgevoerd en de gegevens konden verzameld worden.
- De taak werd niet correct uitgevoerd omdat het proefpersoon met behulp van het systeem de muis niet tot aan de gewenste knop kon bewegen. In dit geval

konden er geen gegevens worden verzameld en werd de taak beschouwd als mislukt.

- De taak werd niet correct uitgevoerd omdat het proefpersoon met behulp van het systeem de website gesloten heeft. In dit geval konden er geen gegevens worden verzameld en werd de taak beschouwd als mislukt.

Indien het proefpersoon de gewenste pagina niet kon bereiken, kon op deze pagina ook geen metingen uitgevoerd worden gezien er niet correct naar deze pagina genavigeerd kon worden. In dit geval werden de taken op deze pagina ook als mislukt beschouwd. Tijdens het uitvoeren van het experiment wordt in de achtergrond data verzameld over de muisbewegingen door Hotjar, dit heeft als doel een heatmap te kunnen genereren. Deze heatmap zal in het analyse hoofdstuk verder besproken worden.

5.4.1. Uitvoering van het experiment met GazeTracking

GazeTracking biedt standaard enkel de mogelijkheid om de ogen te traceren. Om de vergelijking tussen de 2 systemen zo eerlijk mogelijk te laten verlopen geven we GazeTracking de mogelijkheid om de muis te controleren. Zo kunnen we de precisie testen. Dit wordt gedaan aan de hand van een gelijkaardige methode als de technische verwezenlijking, de implementatie hiervan wordt getoond via onderstaande code. Eerst wordt de originele code getoond, daarna de aangepaste versie.

Listing 5.1: Code uit GazeTracking vooraf aan de implementatie om de muis over te nemen

```
if gaze.is_blinking():
    text = "Blinking"
elif gaze.is_right():
    text = "Looking_right"
elif gaze.is_left():
    text = "Looking_left"
elif gaze.is_center():
    text = "Looking_center"
```

Listing 5.2: Code uit GazeTracking na aan de implementatie om de muis over te nemen

```
if gaze.is_blinking():
    text = "Blinking"
    pyautogui.click()
elif gaze.is_right():
    text = "Looking_right"
    pyautogui.moveRel(10, 0)
elif gaze.is_left():
    text = "Looking_left"
```

```

pyautogui.moveRel(-10, 0)
elif gaze.is_center():
    text = "Looking:center"
    pyautogui.moveRel(0, 0)

```

Data verzameld uit het experiment met GazeTracking van proefpersoon met motorische beperking

In tabel 5.1 zijn de coördinaten te zien die verzameld werden wanneer de proefpersoon met motorische beperking een van de navigatietaken probeerde uit te voeren. Indien de taak als mislukt beschouwd werd konden er geen gegevens verzameld worden, dit wordt aangegeven met het symbool '/'.

Coördinaten per klik	Overbelicht	Evenwichtig belicht	Onderbelicht
LB	(x=/, y=/)	(x=/, y=/)	(x=/, y=/)
LM	(x=275, y=808)	(x=271, y=824)	(x=350, y=812)
LO	(x=/, y=/)	(x=/, y=/)	(x=/, y=/)
MB	(x=/, y=/)	(x=/, y=/)	(x=/, y=/)
M	(x=/, y=/)	(x=/, y=/)	(x=/, y=/)
MO	(x=/, y=/)	(x=/, y=/)	(x=/, y=/)
RB	(x=/, y=/)	(x=/, y=/)	(x=/, y=/)
RM	(x=/, y=/)	(x=, y=)	(x=, y=)
RO	(x=/, y=/)	(x=/, y=/)	(x=/, y=/)

Tabel 5.1: De verzamelde coördinaten van de muispositie bij het uitvoeren van een klik voor elke navigatietaak met GazeTracking onder verschillende lichtomstandigheden bij een persoon met een motorische beperking

In tabel 5.2 is de gespendeerde tijd te zien wanneer de proefpersoon met motorische beperking een van de navigatietaken probeerde uit te voeren. Indien de taak als mislukt beschouwd werd konden er geen gegevens verzameld worden, dit wordt aangegeven met het symbool '/'.

Benodigde tijd per klik	Overbelicht	Evenwichtig belicht	Onderbelicht
LB	/	/	/
LM	21	22	16
LO	/	/	/
MB	/	/	/
M	/	/	/
MO	/	/	/
RB	/	/	/
RM	/	/	/
RO	/	/	/

Tabel 5.2: De benodigde tijd (in seconden, afgerond op 0 decimalen) voor elke navigatietaak met GazeTracking onder verschillende lichtomstandigheden bij een persoon met een motorische beperking

Data verzameld uit het experiment met GazeTracking van proefpersoon zonder motorische beperking

In tabel 5.3 zijn de coördinaten te zien die verzameld werden wanneer het proefpersoon zonder motorische beperking een van de navigatietaken probeerde uit te voeren. Indien de taak als mislukt beschouwd werd konden er geen gegevens verzameld worden, dit wordt aangegeven met het symbool '/'.

Coördinaten per klik	Overbelicht	Evenwichtig belicht	Onderbelicht
LB	(x=/ , y=/)	(x=/ , y=/)	(x=/ , y=/)
LM	(x=290 , y=815)	(x=292 , y=803)	(x=310 , y=801)
LO	(x=/ , y=/)	(x=/ , y=/)	(x=/ , y=/)
MB	(x=/ , y=/)	(x=/ , y=/)	(x=/ , y=/)
M	(x=/ , y=/)	(x=/ , y=/)	(x=/ , y=/)
MO	(x=/ , y=/)	(x=/ , y=/)	(x=/ , y=/)
RB	(x=/ , y=/)	(x=/ , y=/)	(x=/ , y=/)
RM	(x=/ , y=/)	(x= , y=)	(x= , y=)
RO	(x=/ , y=/)	(x=/ , y=/)	(x=/ , y=/)

Tabel 5.3: De verzamelde coördinaten van de muispositie bij het uitvoeren van een klik voor elke navigatietaak met GazeTracking onder verschillende lichtomstandigheden bij een persoon zonder een motorische beperking

In tabel 5.4 is de gespendeerde tijd te zien wanneer het proefpersoon zonder motorische beperking een van de navigatietaken probeerde uit te voeren. Indien de

taak als mislukt beschouwd werd konden er geen gegevens verzameld worden, dit wordt aangegeven met het symbool '/'.

Benodigde tijd per klik	Overbelicht	Evenwichtig belicht	Onderbelicht
LB	/	/	/
LM	40	28	21
LO	/	/	/
MB	/	/	/
M	/	/	/
MO	/	/	/
RB	/	/	/
RM	/	/	/
RO	/	/	/

Tabel 5.4: De benodigde tijd (in seconden, afgerond op 0 decimalen) voor elke navigatietaak met GazeTracking onder verschillende lichtomstandigheden bij een persoon zonder een motorische beperking

5.4.2. Uitvoering van het experiment met de technische verwezenlijking

Vooraleer het experiment met de technische verwezenlijking van dit onderzoek uitgevoerd kan worden is het een vereisten dat de gebruiker de virtuele kinsteun kalibreert op een comfortabele positie. Eens dit gebeurd is kan het experiment starten.

Data verzameld uit het experiment met de technische verwezenlijking van proefpersoon met motorische beperking

In tabel 5.5 zijn de coördinaten te zien die verzameld werden wanneer de proefpersoon met motorische beperking een van de navigatietaken probeerde uit te voeren. Indien de taak als mislukt beschouwd werd konden er geen gegevens verzameld worden, dit wordt aangegeven met het symbool '/'.

Coördinaten per klik	Overbelicht	Evenwichtig belicht	Onderbelicht
LB	(x=247 , y=161)	(x=/ , y=/)	(x=/ , y=/)
LM	(x=256 , y=801)	(x=/ , y=/)	(x=134 , y=806)
LO	(x=301 , y=1322)	(x=/ , y=/)	(x=/ , y=/)
MB	(x=1281 , y=202)	(x=1240 , y=231)	(x=1228 , y=87)
M	(x=1296 , y=795)	(x=/ , y=/)	(x=/ , y=/)
MO	(x=/ , y=/)	(x=/ , y=/)	(x=/ , y=/)
RB	(x=2369 , y=175)	(x=/ , y=/)	(x=/ , y=/)
RM	(x=2519 , y=784)	(x= , y=)	(x= , y=)
RO	(x=/ , y=/)	(x=/ , y=/)	(x=/ , y=/)

Tabel 5.5: De verzamelde coördinaten van de muispositie bij het uitvoeren van een klik voor elke navigatietaak met de technische verwezenlijking van dit onderzoek onder verschillende lichtomstandigheden bij een persoon met een motorische beperking

In tabel 5.6 is de gespendeerde tijd te zien wanneer het proefpersoon met motorische beperking een van de navigatietaken probeerde uit te voeren. Indien de taak als mislukt beschouwd werd konden er geen gegevens verzameld worden, dit wordt aangegeven met het symbool '/'.

Benodigde tijd per klik	Overbelicht	Evenwichtig belicht	Onderbelicht
LB	116	/	/
LM	31	/	34
LO	91	/	/
MB	47	41	81
M	19	/	/
MO	/	/	/
RB	88	/	/
RM	62	/	/
RO	/	/	/

Tabel 5.6: De benodigde tijd (in seconden, afgerond op 0 decimalen) voor elke navigatietaak met de technische verwezenlijking van dit onderzoek onder verschillende lichtomstandigheden bij een persoon met een motorische beperking

Data verzameld uit het experiment met de technische verwezenlijking van proefpersoon zonder motorische beperking

In tabel 5.7 zijn de coördinaten te zien die verzameld werden wanneer de proefpersoon zonder motorische beperking een van de navigatietaken probeerde uit

te voeren. Indien de taak als mislukt beschouwd werd konden er geen gegevens verzameld worden, dit wordt aangegeven met het symbool ‘/’.

Coördinaten per klik	Overbelicht	Evenwichtig belicht	Onderbelicht
LB	(x=238 , y=172)	(x=236 , y=188)	(x=252 , y=294)
LM	(x=/ , y=/)	(x=273 , y=743)	(x=268 , y=804)
LO	(x=/ , y=/)	(x=/ , y=/)	(x=/ , y=/)
MB	(x=/ , y=/)	(x=1221 , y=124)	(x=/ , y=/)
M	(x=/ , y=/)	(x=1251 , y=794)	(x=1304 , y=807)
MO	(x=/ , y=/)	(x=/ , y=/)	(x=/ , y=/)
RB	(x=/ , y=/)	(x=/ , y=/)	(x=/ , y=/)
RM	(x=/ , y=/)	(x= , y=)	(x= , y=)
RO	(x=/ , y=/)	(x=/ , y=/)	(x=/ , y=/)

Tabel 5.7: De verzamelde coördinaten van de muispositie bij het uitvoeren van een klik voor elke navigatietaak met de technische verwezenlijking van dit onderzoek onder verschillende lichtomstandigheden bij een persoon zonder een motorische beperking

In tabel 5.8 is de gespendeerde tijd te zien wanneer de proefpersoon zonder motorische beperking een van de navigatietaken probeerde uit te voeren. Indien de taak als mislukt beschouwd werd konden er geen gegevens verzameld worden, dit wordt aangegeven met het symbool ‘/’.

Benodigde tijd per klik	Overbelicht	Evenwichtig belicht	Onderbelicht
LB	79	20	52
LM	/	31	18
LO	/	/	/
MB	/	165	/
M	/	21	21
MO	/	/	/
RB	/	/	/
RM	/	/	/
RO	/	/	/

Tabel 5.8: De benodigde tijde (in seconden, afgerond op 0 decimalen) voor elke navigatietaak met de technische verwezenlijking van dit onderzoek onder verschillende lichtomstandigheden bij een persoon zonder een motorische beperking

5.5. Ondervinding uit het experiment

Tijdens het uitvoeren van het experiment ondervonden beide proefpersonen verschillende moeilijkheden bij het gebruik van zowel GazeTracking als de in dit onderzoek gecreëerde technische verwezenlijking.

5.5.1. Ondervingen van proefpersoon met motorische beperking

De proefpersoon met een motorische beperking gaf aan moeilijk te kunnen zien waar de muis zich op het scherm bevond. Dit wegens twee verschillende redenen. Ten eerste bewoog de muis niet met dezelfde snelheid als de ogen, waardoor het blikpunt van de gebruiker zich reeds verder op het scherm bevond dan de muis. Ten tweede moest de proefpersoon soms zo ver naar links, rechts, boven of onder kijken om de muis voldoende in de gewenste richting te verplaatsen, dat het blikpunt zich buiten het scherm bevond.

5.5.2. Ondervingen van proefpersoon zonder motorische beperking

De proefpersoon zonder motorische beperking gaf dezelfde ondervindingen aan als de proefpersoon met motorische beperking. Daarnaast benoemde deze persoon nog enkele andere bevindingen. Zo werd aangegeven dat het bewegen in een rechte lijn naar links en rechts goed verliep. Dit gold ook voor centraal kijken om de muis te doen stoppen. De muis naar boven doen bewegen werd als moeilijk, maar doenbaar ervaren. Het bewegen van de muis naar beneden werd door de proefpersoon echter als onmogelijk beschouwd.

6

Analyse

De voorlaatste fase van dit onderzoek omvat de analyse van de verzamelde data. Om de precisie te beoordelen wordt de Euclidische afstand van de verzamelde coördinaten tot aan de knoppen uit de testomgeving berekent. Daarnaast worden de gegenereerde heatmaps geanalyseerd. Naast de precisie richt dit onderzoek zich ook op de analyse van de bestede tijd. Hiervoor wordt per systeem de gemiddeld bestede tijd berekent en vergeleken. De analyse van precisie en tijd gebeurd voor alle 3 de lichtomstandigheden.

Indien er tijdens het experiment geen gegevens verzameld konden worden, kan er ook geen analyse over de bijhorende taak gebeuren. Dit wordt dan aangeduid met het symbool '/'. Deze worden dus ook niet meegenomen in gemiddelde berekeningen.

6.1. Analyse van precisie

Om de precisie en de afwijking van kliks van het systeem te meten maken we gebruik van de Euclidische afstand. In de wiskunde stelt dit de afstand tussen twee punten voor en wordt berekent door middel van formule 6.1. De formule werd verkregen uit (WisFaq, 2005).

$$d = \sqrt{(x_{\text{doel}} - x_{\text{realiteit}})^2 + (y_{\text{doel}} - y_{\text{realiteit}})^2} \quad (6.1)$$

6.1.1. Proefpersoon met motorische beperking

Van de uit het experiment verzamelde coördinaten wordt de euclidische afstand tot aan hun doel berekent. Een overzicht hiervan is te zien in de tabellen 6.1 en 6.2.

Euclidische afstand	Overbelicht	Evenwichtig belicht	Onderbelicht
LB	/	/	/
LM	255,13	252,14	330,22
LO	/	/	/
MB	/	/	/
M	/	/	/
MO	/	/	/
RB	/	/	/
RM	/	/	/
RO	/	/	/
Gemiddelde afstand	255,13	252,14	330,22

Tabel 6.1: De euclidische afstand, afgerond op 2 decimalen, van de muispositie bij het gebruik van GazeTracking onder verschillende lichtomstandigheden bij de proefpersoon met een motorische beperking

Euclidische afstand	Overbelicht	Evenwichtig belicht	Onderbelicht
LB	267,23	/	/
LM	236	/	114,16
LO	381,48	/	/
MB	182	214,76	84,81
M	16,76	/	/
MO	/	/	/
RB	230,79	/	/
RM	26,40	/	/
RO	/	/	/
Gemiddelde afstand	195,57	214,76	99,49

Tabel 6.2: De euclidische afstand, afgerond op 2 decimalen, van de muispositie bij het gebruik van de technische verwezenlijking van dit onderzoek onder verschillende lichtomstandigheden bij de proefpersoon met een motorische beperking

Uit tabellen 6.1 en 6.2 kan afgeleid worden dat de gemiddelde euclidische afstand onder alle lichtomstandigheden lager ligt bij de technische verwezenlijking van dit onderzoek. Hiervoor kunnen twee verklaringen zijn: ofwel presteert de technische verwezenlijking daadwerkelijk beter wanneer het aankomt op precisie, ofwel ontstaat er een vertekend beeld doordat er met dit systeem meer acties konden worden uitgevoerd, wat resulteerde in een groter aantal metingen.

6.1.2. Proefpersoon zonder motorische beperking

Van de uit het experiment verzamelde coördinaten wordt de euclidische afstand tot aan hun doel berekent. Een overzicht hiervan is te zien in de tabellen 6.3 en 6.4.

Euclidische afstand	Overbelicht	Evenwichtig belicht	Onderbelicht
LB	/	/	/
LM	270,42	272,02	290,00
LO	/	/	/
MB	/	/	/
M	/	/	/
MO	/	/	/
RB	/	/	/
RM	/	/	/
RO	/	/	/
Gemiddelde afstand	270,42	272,02	290,00

Tabel 6.3: De euclidische afstand, afgerond op 2 decimalen, van de muispositie bij het gebruik van GazeTracking onder verschillende lichtomstandigheden bij de proefpersoon zonder een motorische beperking

Euclidische afstand	Overbelicht	Evenwichtig belicht	Onderbelicht
LB	265,76	273,64	359,02
LM	/	259,34	248,03
LO	/	/	/
MB	/	119,57	/
M	/	29,61	/
MO	/	/	/
RB	/	/	/
RM	/	/	/
RO	/	/	/
Gemiddelde afstand	265,76	170,54	303,53

Tabel 6.4: De euclidische afstand, afgerond op 2 decimalen, van de muispositie bij het gebruik van de technische verwezenlijking van dit onderzoek onder verschillende lichtomstandigheden bij de proefpersoon zonder een motorische beperking

Uit tabellen 6.3 en 6.4 blijkt dat de gemiddelde euclidische afstand lager ligt bij de technische verwezenlijking van dit onderzoek bij overbelichting en een evenwich-

tige belichting. Indien er sprake is van onderbelichting ligt de gemiddelde euclidische afstand van GazeTracking lager. Hiervoor zijn twee mogelijke verklaringen: ofwel presteert de technische verwijzing daadwerkelijk beter op het vlak van precisie onder overbelichte en evenwichtig belichte omstandigheden, ofwel ontstaat een vertekend beeld doordat dit systeem meer acties toeliet, wat resulteerde in een groter aantal metingen.

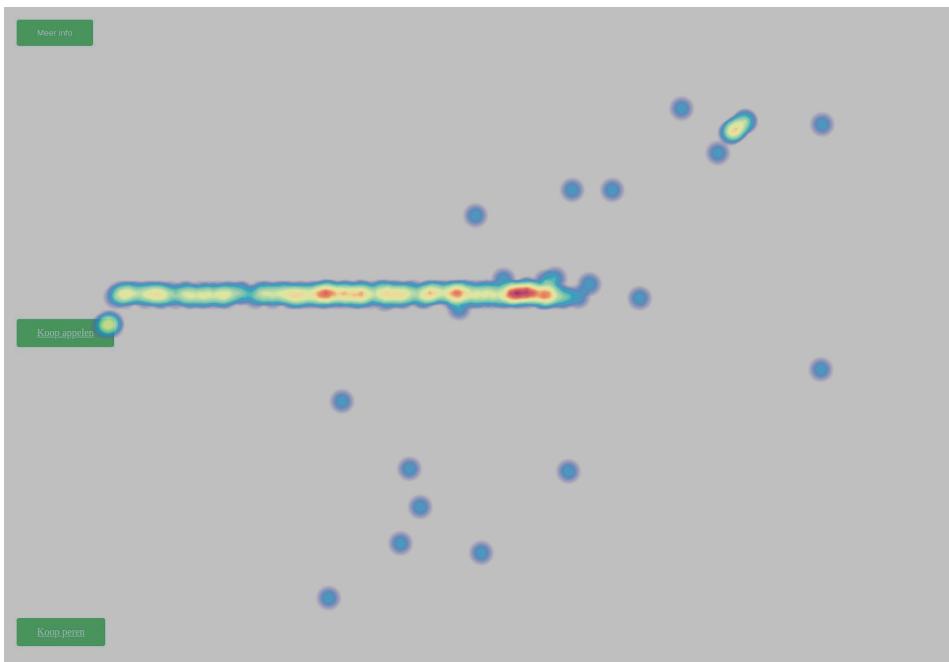
Opvallend is dat de technische verwijzing bij de proefpersoon met een motorische beperking zich in eender welke belichting vertoonde als het accurater systeem. Bij de proefpersoon zonder motorische beperking was dit enkel het geval bij overbelichte en evenwichtig belichte omstandigheden. Deze resultaten suggereren dat er mogelijks meer metingen met meer proefpersonen verricht moeten worden om beide systemen op een betrouwbare manier te kunnen evalueren.

6.2. Analyse van heatmaps

Elke heatmap die gegenereerd is, werd verkregen via de online tool Hotjar. Wegens technische problemen konden er geen heatmaps gegenereerd worden van het experiment met de proefpersoon met motorische beperking. Gezien er tijdens het experiment met de proefpersoon zonder motorische beperking geen metingen konden uitgevoerd worden op de peren pagina, is ook hier geen heatmap van.

6.2.1. Heatmaps GazeTracking

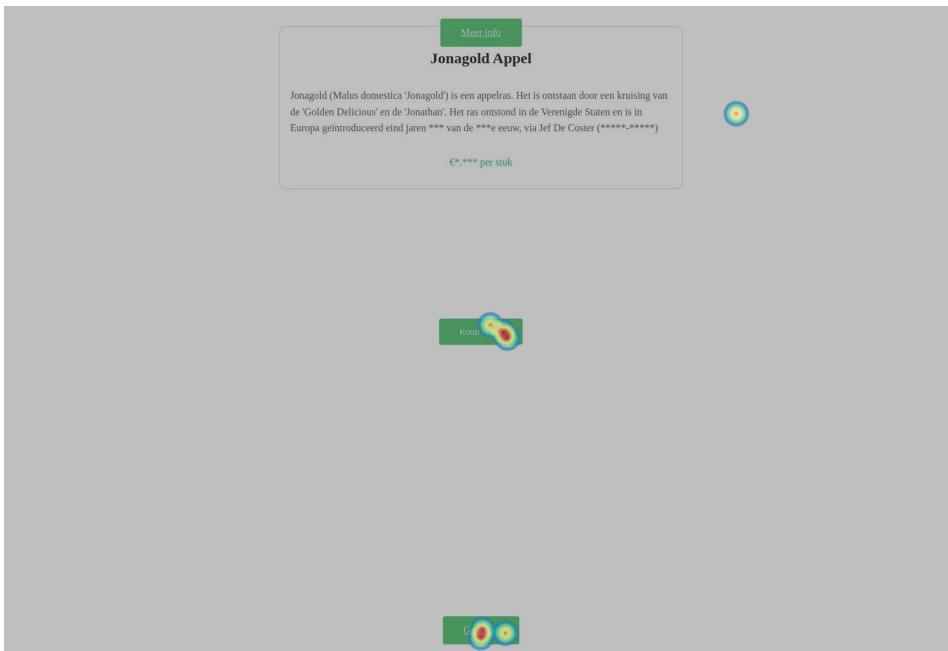
Homepagina



Figuur 6.1: Heatmap van de homepagina gegeneerd tijdens het experiment met de proefpersoon zonder motorische beperking en GazeTracking. Heatmap gegenereerd door Hotjar

Op afbeelding 6.1 is te zien dat GazeTracking de muis voornamelijk in een rechte lijn naar links beweegt. Deze observatie wordt ook bevestigd door de gegevens verzameld in tabel 5.3 waaruit af te leiden is dat enkel LB geraakt werd. Daarnaast is op de afbeelding te zien dat de muis op enkele punten naar boven of beneden bewogen is. Tijdens het experiment met GazeTracking werd deze beweging echter niet door het systeem zelf veroorzaakt, maar door bewegingen op de testomgeving voorafgaand aan het experiment.

Appelen pagina

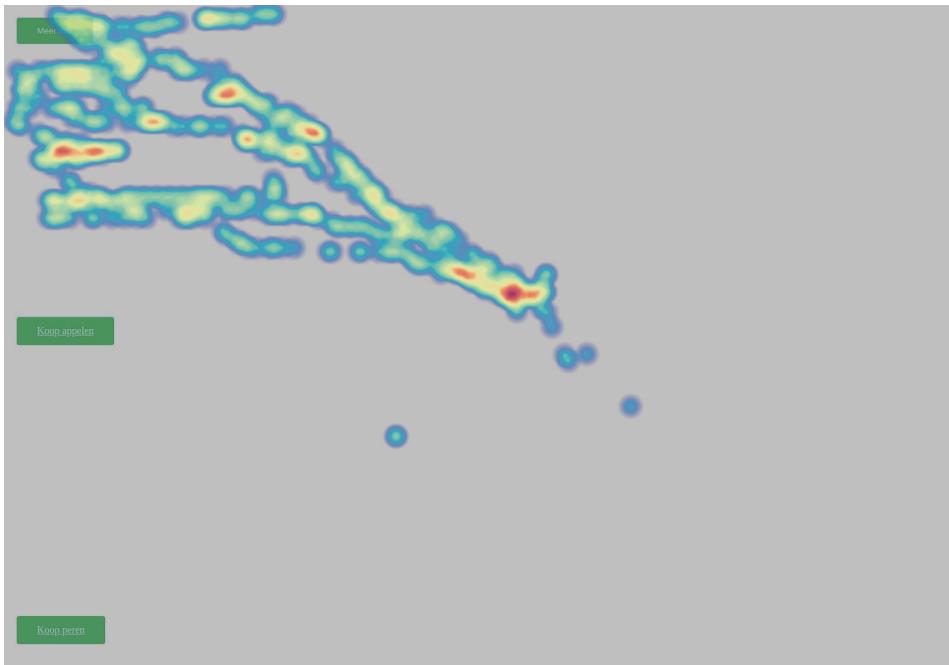


Figuur 6.2: Heatmap van de appelen pagina gegeneerd tijdens het experiment met de proefpersoon zonder motorische beperking en GazeTracking. Heatmap gegenereerd door Hotjar

Op de afbeelding 6.2 is relatief weinig beweging te zien. Het lijkt erop dat voornamelijk op de knoppen M en MB geklikt werd. Uit tabel 5.3 blijkt echter dat deze twee knoppen niet daadwerkelijk zijn geraakt. Deze discrepantie is te verklaren door het feit dat tijdens het voorbereiden van het experiment de knop M geraakt wanneer de omgeving klaargezet werd. Daarnaast werd de knop MO gebruikt om terug te keren naar de homepagina, nadat het experiment op deze pagina afgerond was.

6.2.2. Heatmaps technische verwezenlijking van dit onderzoek

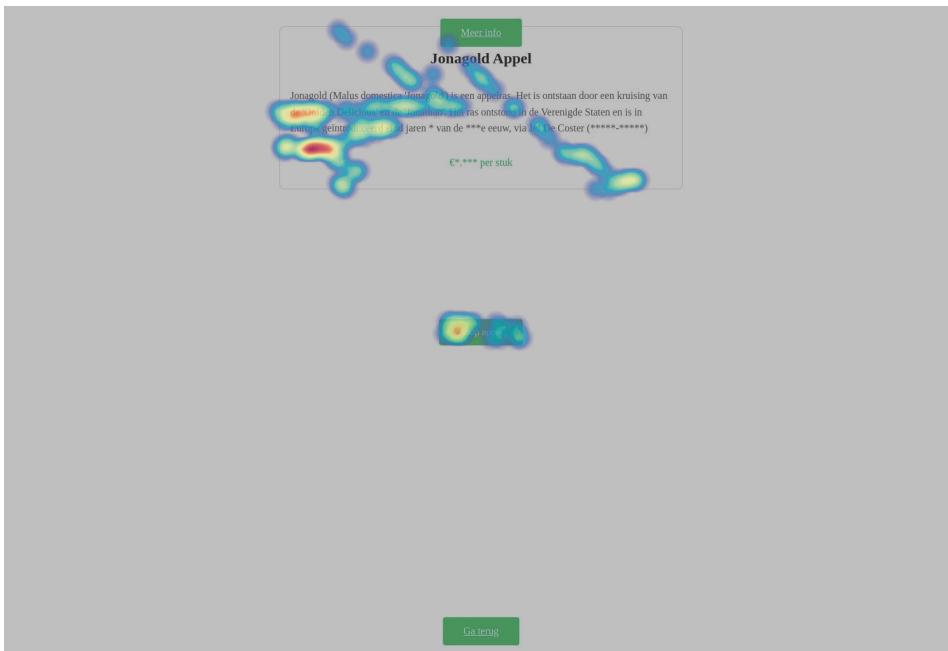
Homepagina



Figuur 6.3: Heatmap van de homepagina gegeneerd tijdens het experiment met de proefpersoon zonder motorische beperking en de technische verwezenlijking. Heatmap gegenereerd door Hotjar

Op afbeelding 6.3 is muisbeweging te zien in zowel een rechte lijn naar links als dia-gonaal omhoog. Hiernaast worden ook de ondervindingen van de proefpersoon, namelijk dat naar beneden bewegen niet mogelijk was, bevestigd. Dit blijkt uit het ontbreken van enige beweging in de onderste helft van de pagina op de heatmap.

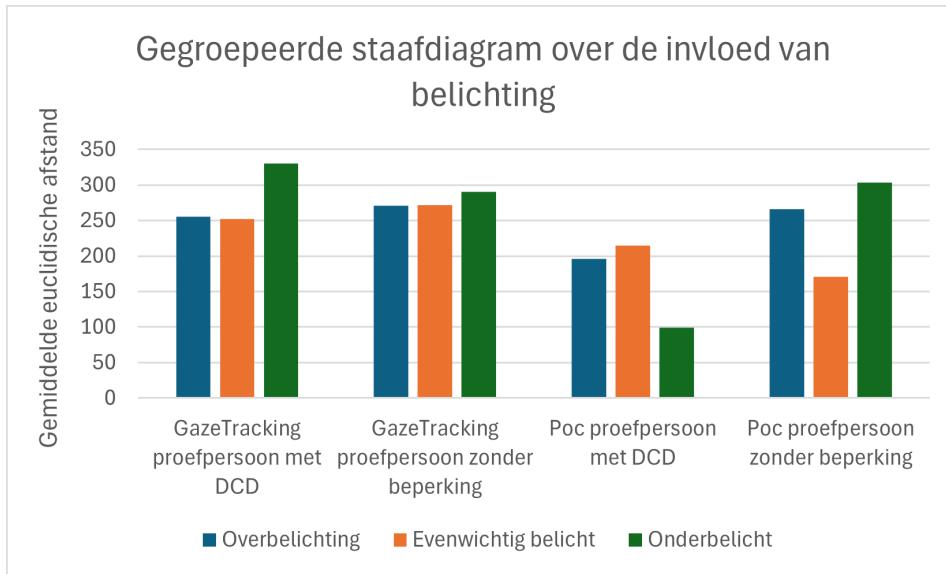
Appelen pagina



Figuur 6.4: Heatmap van de appelen pagina gegeneerd tijdens het experiment met de proefpersoon zonder motorische beperking en de technische verwezenlijking. Heatmap gegenereerd door Hotjar

Op afbeelding 6.4 is te zien dat de proefpersoon de knop M succesvol heeft geraakt en de muis vervolgens omhoog bewoog richting de knop MB. Opvallend is echter dat er aanzienlijke bewegingen rondom de knop MB plaatsvonden voordat deze daadwerkelijk werd geraakt. Dit fenomeen is ook te zien in tabel 5.8, waaruit blijkt dat de proefpersoon 165 seconden nodig had om deze knop te raken. De ontdekkingen van de proefpersoon waarin aangegeven werd dat de muis omhoog bewegen mogelijk is maar moeilijk is, worden bevestigd door deze heatmap.

6.3. Analyse van belichting



Figuur 6.5: Gegroepeerde Staafdiagram waarop te zien is wat de invloed van belichting is op de gemiddelde euclidische afstand bij gebruik van GazeTracking en de technische verwezenlijking van dit onderzoek. De diagram werd gecreëerd met behulp van Microsoft Excel

Afbeelding 6.5 toont een gegroepeerde staafdiagram waarin de eerder berekend gemiddelde euclidische afstand tot het doel per proefpersoon en systeem wordt weergegeven. Hieruit blijkt dat de technische verwezenlijking van dit onderzoek gevoeliger is voor variaties in belichting. Hiervoor kunnen twee verklaringen worden aangehaald: ofwel presteert het systeem daadwerkelijk minder goed dan GazeTracking wat betreft consistentie onder verschillende lichtomstandigheden, ofwel ontstaat er een vertekend beeld doordat er met dit systeem meer acties kunnen worden uitgevoerd, wat resulterde in een grotere hoeveelheid metingen.

6.4. Analyse van tijd

Om de benodigde tijd te analyseren wordt het aantal seconden, in eender welke lichtomstandigheid, dat een proefpersoon gemiddeld nodig had om een actie uit te voeren berekent. Dit is te zien in tabel 6.5

Gemiddeld gespendeerde tijd	Proefpersoon met DCD	Proefpersoon zonder beperking
GazeTracking	19,67	29,67
Technische verwezenlijking	61,00	50,88

Tabel 6.5: De gemiddeld benodigde tijd (in seconden, afgerond op 2 decimalen) voor elk proefpersoon om een actie uit te voeren met GazeTracking en de technische verwezenlijking van dit onderzoek

Uit tabel 6.5 blijkt dat de gebruiker gemiddeld aanzienlijk langer doet over een actie bij de technische verwezenlijking dan bij GazeTracking. Beide systemen waren ingesteld om de muis met dezelfde snelheid te bewegen dus dat is geen mogelijke verklaring voor dit resultaat.

De vertraging bij de technische verwezenlijking kan mogelijk verklaard worden door het feit dat het systeem in meerdere richtingen kan bewegen dan GazeTracking, waardoor de gebruiker vaker moet bijsturen.

Een alternatieve verklaring kan worden afgeleid uit de data van tabellen 5.1 en 5.3 in combinatie met de ondervindingen van de proefpersonen: GazeTracking interpreert de blikrichting vrijwel altijd als links, wat resulteert in hoge consistentie in de linkse richting en bijgevolg een snel bereik van LM.

7

Conclusie

Webcam gebaseerde oogtraceren en -besturing is een technologie die het gezicht en de ogen van de gebruiker detecteert. Vervolgens wordt de verzamelde informatie gebruikt om het blikpunt van de ogen te bepalen.

Het doel van dit onderzoek was om te bepalen wat de bestaande problemen zijn met deze technologie en hoe deze opgelost kunnen worden. Aan de hand van een uitgebreide literatuurstudie, een technische verwezenlijking, een experiment, een analyse en een persoonlijk gesprek met het bedrijf Eleven Ways zijn verschillende problemen geïdentificeerd worden en bijhorend mogelijke oplossingen.

Het experiment in dit onderzoek werd uitgevoerd met twee proefpersonen. Om echter tot een betrouwbaar resultaat waar objectieve conclusies uit te verkrijgen zijn, zouden meer metingen met een grotere groep proefpersonen noodzakelijk zijn. In deze conclusie wordt besproken wat geleerd kan worden uit de behaalde resultaten, in combinatie met de literatuurstudie.

Het eerste probleem, namelijk de precisie, werd geïdentificeerd in zowel de literatuurstudie als het gesprek met Eleven Ways. Dit probleem bestond uit twee deelproblemen: ten eerste de precisie waarmee de ogen en het blikpunt worden gedetecteerd, bijgevolg ook de precisie van muisbewegingen. De eerste stap ondernomen om dit probleem op te lossen was het implementeren van MediaPipe. Dankzij de topologie van 478 gezichtsherkenningspunten konden de ogen met een veel hogere precisie gedetecteerd worden.

Deze verbetering is bevestigd in de analyse van precisie, waarin bleek dat de gemiddelde euclidische afstand bij gebruik van de technische verwezenlijking vrijwel altijd lager lag dan bij gebruik van GazeTracking. De enige uitzondering hierop was bij het proefpersoon zonder motorische beperking in een onderbelichte situatie. Een tweede deelprobleem was het negatieve effect van hoofdbewegingen. Om dit tegen te gaan nam dit onderzoek inspiratie uit de studie van (Kaduk e.a., 2023). Een virtuele kinsteun werd geïmplementeerd zodat het gecreëerde systeem stopt met

traceren zodra het hoofd zich niet meer op de vooraf ingestelde positie bevindt. De combinatie van MediaPipe en de virtuele kinsteen heeft geleid tot een significante verbetering van de precisie.

Uit tabel 6.5 blijkt echter dat deze verhoogde graad van precisie ten kosten gaat van een langere gemiddelde benodigde tijd per actie. In een toekomstig onderzoek zou er onderzocht kunnen worden hoe deze benodigde tijd kan worden gereduceerd, zonder vermindering van de precisie.

Een tweede probleem dat werd vastgesteld was de invloed van lichtomstandigheden. In meerdere studies werd belichting genoemd als een negatieve factor op de werking van de technologie. Om dit probleem op te lossen heeft dit onderzoek CLAHE geïmplementeerd. Voordat de ogen werden gedetecteerd, werd het verkregen frame bewerkt met de CLAHE techniek.

Wanneer gekeken wordt naar de grafiek op afbeelding 6.5 blijkt echter dat deze techniek geen merkbare invloed had op de consistentie van de technische verwijzenlijking onder verschillende lichtomstandigheden. Sterker nog, het tegenovergestelde blijkt het geval: GazeTracking presteerde consistentier dan de technische verwijzenlijking onder de verschillende lichtomstandigheden. De oorzaak van deze inconsistentie is niet eenduidig vast te stellen. Mogelijks is er een vertekend beeld ontstaan bij de technische verwijzenlijking omdat er met dit systeem een groter aantal metingen kon uitgevoerd worden. Hierdoor was er meer data om te analyseren.

Dit onderzoek kon dan ook geen effectieve oplossing bieden voor het belichtingsprobleem.

Tijdens de literatuurstudie werd ook kalibratie geïdentificeerd als een probleem binnenin de huidige technologie. Zowel de software als de praktische kant rondom kalibratie brengt verschillende problemen met zich mee. Dit onderzoek heeft twee mogelijke oplossingen hiervoor onderzocht: SR uit de studie van Hassoumi e.a. (2019) en de 9-punt kalibratie van Drewes e.a. (2019). Beide methodes bieden mogelijke oplossingen voor het kalibratie probleem maar wegens de technische complexiteit is er in dit onderzoek gekozen om geen kalibratie te implementeren.

Dit onderzoek bied dus geen technisch uitgewerkte oplossing voor het kalibratie probleem.

Tijdens het persoonlijk gesprek met het bedrijf Eleven ways kwam ook het probleem rondom de gebruikersinterface aan bod. Complexere vormen van interactie zoals scrollen, slepen en dubbelklikken zijn vaak niet toegankelijk in bestaande systemen. Dit onderzoek bied geen uitwerking rondom scrollen of slepen maar de functionaliteit van dubbelklikken is wel gerealiseerd. Met behulp van PyAutoGUI verkrijgt de technische verwijzenlijking de mogelijkheid om niet alleen de muis te bewegen maar ook om klik acties uit te voeren. Indien de gebruiker knippert zal het systeem een klikactie uitvoeren. Bijgevolg zal er een dubbelklik actie uitgevoerd worden indien de gebruiker tweemaal snel achter elkaar knippert.

Aan het begin van dit onderzoek werd verwacht dat een werkend prototype zou worden ontwikkeld dat verbeteringen biedt in de precisie van oogbesturende software. Deze verwachtingen zijn gerealiseerd: uit de analyse blijkt dat de precisie verbeterd is door de implementatie van MediaPipe en een virtuele kinstuur. Het resultaat is een systeem dat het besturen van een computer toegankelijker en gebruiksvriendelijker maakt voor mensen met motorische beperkingen.

Desondanks kent de technische verwezenlijking van dit onderzoek nog verschillende beperkingen: het bewegen van de muis naar onder, de gevoelighed voor lichtomstandigheden en de hoge gemiddelde tijd per actie. In toekomstig onderzoek zouden deze beperkingen mogelijk kunnen worden verminderd door het implementeren van een kalibratieproces, het toepassen van **AI** en **ML**, of het gebruik van alternatieve technieken ter vervanging van **CLAHE** om de invloed van belichting te minimaliseren.

A

Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

Samenvatting

Het internet en computers zijn niet meer weg te denken uit ons dagelijks leven; ze zijn een vereisten in heel veel jobs en bieden voor velen een vorm van ontspanning. Helaas is het besturen van een computer niet voor iedereen even toegankelijk, met name voor mensen met motorische beperkingen. Voor deze groep is het gebruik van een traditioneel toetsenbord en muis vaak niet gebruiksvriendelijk of zelfs onmogelijk. Oog-besturende software biedt voor deze individuen een manier om de traditionele computermuis over te nemen en makkelijker hun computer te besturen. De technologie bevat echter ook foutenmarge die het gebruik ervan hinderen. Dit bacheloronderzoek richt zich daarom op de vraag hoe we deze software kunnen verbeteren door de bestaande problemen te studeren en op te lossen, specifiek zal er gefocust worden op webcam gebaseerde eye-trackers. Op deze manier hoopt dit onderzoek besturing van een computer toegankelijker en gebruiksvriendelijker te maken voor mensen met motorische beperkingen.

A.1. Inleiding

Werken met een computermuis vereist een precieze motoriek waardoor individuen met motorische beperkingen zoals parkinson, verlammingen,... het vaak lastig hebben met het gebruik hiervan. Dit werd ook al aangetoond in meerdere studies zoals die van TREWIN en PAIN (1999) en Spiller e.a. (2019). Naast hun relevantie voor individuen met motorische beperkingen blijken webcam gebaseerde eye-trackers een waardevol instrument voor wetenschappelijk onderzoek te doen vanwege hun

brede toegankelijkheid en gebruiksgemak. Doordat ze onafhankelijk kunnen worden ingezet zonder de noodzaak van een specialist, bieden ze een efficiënte en laagdrempelige oplossing voor diverse onderzoeksdoeleinden Kaduk e.a. (2023). Ondanks hun potentieel zijn er nog steeds verschillende problemen met deze technologie waardoor men zich afvragen hoe we deze tools verbeteren? Om deze hoofdvraag te beantwoorden, wordt eerst onderzocht welke problemen momenteel met deze tools bestaan, hoe de onderliggende technologie precies functioneert en welke python libraries zijn geschikt voor deze technologie?

Tijdens dit onderzoek zal eerst een uitgebreide literatuurstudie verricht worden naar de bestaande technologieën en hun problematiek. Naast literatuur zal hier voor ook de hulp ingeschakeld worden van het consultancy bedrijf 'Eleven Ways', zij specialiseren in digitale toegankelijkheid en zullen in de vorm van een persoonlijk gesprek hulp bieden bij het identificeren van de belangrijkste problemen. Hierna zal bovenop de open-source python library 'GazeTracking' de gevonden oplossingen geïmplementeerd worden.

A.2. Literatuurstudie

A.2.1. Waarom oog-besturing

In de studie van Spiller e.a. (2019) namen veertien personen, allemaal gediagnosticeerd met hersenverlamming, deel aan een experiment waarbij de taak was om zo snel mogelijk een doelwit op een computerscherm te activeren. Tijdens dit experiment werden drie verschillende invoerapparaten gebruikt: een computermuis, een touchscreen en een 'Tobii PCEye GO eye tracking mouse'. De verzamelde data werd achteraf geanalyseerd, waaruit geconcludeerd kon worden dat de 'Tobii PCEye GO eye tracking mouse' een positief effect had. Bij het uitvoeren van de taak werden minder fouten gemaakt, maar de tijd om de taak te voltooien was langer. Dit kan erop wijzen dat eindgebruikers voldoende oefening nodig hebben met deze technologie om deze efficiënt te kunnen gebruiken.

In het onderzoek van Lakshmi Pavani e.a. (2018) werd oog-tracerende software gebruikt om de cursor te controleren met de ogen in plaats van met een traditionele computermuis. Voor het onderzoek werd een eigen website gecreëerd waarop de gebruiker moest navigeren. Van de 20 experimenten waren 15 een succes, en werd er een nauwkeurigheid van 75% verkregen. Hieruit blijkt dat oogbesturing een groot voordeel kan bieden voor individuen met een motorische beperking bij webnavigatie.

Hiernaast concludeerde Van der Cruyssen e.a. (2023) dat webcam gebaseerde eyetrackers relevant kunnen zijn voor onderzoeken waarbij er geen extreem hoge precisie vereist is. Tijdens hun onderzoek probeerden ze verschillende voorgaande

onderzoeken na te bootsen met behulp van webcam eye-trackers in plaats van eye-trackers uit een laboratorium. Ze merkten op dat de onderzoeken sneller, gemakkelijker en goedkoper konden verlopen. Deze voordelen gingen ten kosten van een lagere data kwaliteit maar het biedt bewijs dat er veel potentieel in de technologie zit. Daarnaast merkte Van der Cruyssen e.a. (2023) ook dat gezien de continue groei van technologie, de toekomst hiervoor veelbelovend is.

A.2.2. De onderliggende werking en hoe deze best vertaald worden in python

Oog-tracerende technologie kent verschillende varianten met elk hun eigen onderliggende principes. In deze studie zal de focus liggen op webcam gebaseerde oog-tracing.

Theoretische werking

Het artikel van het bedrijf tobii (Miseviciute, [z.d.](#)), een van de wereldwijde marktleiders in eye-tracking, legt kort uit hoe eye-tracking juist werkt. Eye trackers maken gebruik van sensor technologie die de positie en beweging van de ogen meet. Deze informatie kan dan gebruikt worden om te bepalen waar de ogen naar kijken, dit noemt men ‘the point of gaze’ ofterwel het blikpunt. Verschillende vormen van eye-trackers hebben elks hun eigen manier om dit blikpunt te bepalen.

Zoals Jensen (2022) uitlegt in zijn online blog, is webcam gebaseerde oog-tracing puur video gebaseerd. De webcam moet in staat zijn de ogen en pupillen van de gebruik te lokaliseren voordat andere acties ondernomen kunnen worden. Dit gebeurd aan de hand van 4 stappen:

1. Detecteren van de locatie van het gezicht.
2. Detecteren van de ogen op het gezicht.
3. Detecteren van de oriëntatie van zowel het linkse als rechtse oog.
4. De oriëntatie van de ogen in kaart brengen op het coördinaten systeem van het scherm.

Het paper geschreven door (Wolf & Seernani, 2023) legt een gelijkaardige maar andere methode uit in hun paper:

1. Verwerking: ‘In deze stap wordt het individu’s blikpunt geschat door een ‘deep learning model’ dat op voorhand getraind werd.’
2. Kalibratie: ‘Gebaseerd op opnames wordt het algoritme gekalibreerd. Deze opnames zijn afkomstig vanuit stap 1’.
3. Projectie van data: ‘Aan de hand van het model dat werd opgebouwd in stap 2 kan het blikpunt geprojecteerd worden op het scherm in de vorm van x,y coördinaten. Dit proces gebeurd voor elk ‘frame’.

De studie van Taore en Dakin (2023) adresseerde ook de moeilijkheden met webcam gebaseerd eye trackers en gebruikte gelijkaardige stappen als het paper van (Wolf & Seernani, 2023). De individuele 'frames' worden gebruikt om het gezicht, beide ogen en hun irissen te lokaliseren aan de hand van een detectie model. Taore en Dakin (2023) gebruikte hiervoor 'Mediapipe' maar hier bestaan verschillende technieken voor. Deze gegevens worden hierna omgezet in coördinaten waarop berekeningen uitgevoerd kunnen worden.

Bestaande python packages

Er zijn reeds verschillende Python pakketten beschikbaar voor eye-tracking. Hier worden enkele van deze kort besproken, zodat tijdens de ontwikkeling van een eigen oplossing een gefundeerde keuze kan worden gemaakt over welke het meest geschikt is.

1. GazeParser: GazeParser is een 'open-source' bibliotheek geschreven in Python. Het bevat een video gebaseerde eye-tracker en data analyse. Om te beoordelen hoe performant GazeParser is werden verschillende experimenten uitgevoerd met zowel GazeParser en Eyelink. Eyelink is een commercieel beschikbare eye-tracker. Hieruit werd geconcludeerd dat GazeParser op een betrouwbare manier de latentie en amplitude van saccades, snelle oogbewegingen, kan meten. Toch trad er bij twee deelnemers incidenteel een fout op bij de detectie van de blikpositie door GazeParser, terwijl Eyelink deze blikpositie wel consistent kon registreren. Dit verschil is waarschijnlijk toe te schrijven aan verschillen in de methoden voor oogpositiesdetectie (Sogo, 2012).
2. PyTorch: in de studie van Taore en Dakin (2023) werd PyTorch gebruikt om een neuraal netwerk op te zetten voor blikpunt voorspellingen. Dit deden ze volgens de instructies van Valliappan e.a. (2020). De resultaten van hun YOLO netwerk werd achteraf vergeleken met de resultaten van dit netwerk.

A.2.3. De huidige problemen met webcam gebaseerde oog-besturing

Er is een persoonlijk gesprek gevoerd met het bedrijf Eleven Ways waarin zij de problemen aankaarten waar zij van op de hoogte zijn. Deze waren de volgende: een hoge graad van precisie, een kwalitatieve webcam is vereist, men heeft goede lichtomstandigheden nodig en de gebruikersinterface maakt complexere vormen van interactie zoals scrollen, slepen en dubbelklikken vaak niet toegankelijk. Naast dit gesprek zijn er ook verschillende onderzoeken die de huidige problemen aantonen waarop dieper ingegaan zal worden.

Kalibratie problemen

De studie van Nyström e.a. (2012) legt uit dat kalibratie van de software zowel theoretische als praktische problemen met zich meebrengt. Vanuit theoretisch perspectief blijkt het moeilijk om een correct wiskundig model van de ogen te vinden

waarop een functie kan worden uitgevoerd om de blikrichting te bepalen. Deze studie definieert ook 3 mogelijke kalibratie manieren langs de praktische kant:

- ‘System-controlled calibration’. Bij deze methode maakt een algoritme de keuze wanneer de ogen gefocust en gericht naar een doelwit zijn. Deze methode heeft als voordeel dat de kalibratie volledig automatisch gebeurd.
- ‘Operator-controlled calibration’. De operator bepaalt de kalibratie wanneer hij of zij het gevoel heeft dat de gebruiker gefocust is op het doelwit.
- ‘Participant-controlled calibration’ Deze derde methode plaatst meer verantwoordelijkheid bij de gebruiker. Wanneer hij of zij het gevoel heeft gefocust te zijn op een doelwit wordt een knop ingeduwd.

De huidige trend is om steeds meer controle te geven aan het systeem wanneer het gaat over kalibratie. Deze methode heeft als voordeel dat de kalibratie snel en makkelijk kan gebeuren. (Nyström e.a., 2012)

Precisie

De studie van Van der Cruyssen e.a. (2023) had als doel om 3 verschillende eye-tracking studies te repliceren met een webcam gebaseerde eye-tracker en concludeerde er een significant verschil was op de data verzameld door een webcam gebaseerde eye-tracker tegenover data verzameld in een laboratorium.

De studie van Kaduk e.a. (2023) vergelijkt een nieuw webcam gebaseerd eye-tracking systeem met dat vanuit een laboratorium. Hun resultaten concludeerden dat het webcam gebaseerd systeem een 0.5° grotere foutmarge zag. Hiernaast ondervond de studie ook negatieve gevolgen van hoofdbewegingen tijdens het traceren van de ogen.

Een ondervinding die gedeeld wordt door het paper van Wolf en Seernani (2023). In hun onderzoek vond het team verschillende beperkingen van webcam gebaseerde eye-trackers waaronder: lichtcondities, brillen, lage resolutie camera's en hoofdbewegingen.

Het online artikel van Jensen (2021) bevestigt dat webcam gebaseerde eye-trackers over het algemeen een lagere precisie hebben. Naast deze bevestiging probeert dit artikel ook in te gaan op hoe andere factoren een negatieve impact kunnen hebben op de kwaliteit: lichtgevoeligheid, kalibratie en de kwaliteit van de webcam. Gezien webcams oorspronkelijk niet ontworpen werden om eye-trackers te zijn kan de hardware een negatieve invloed hebben op de verkregen data.

Lichtgevoelig

De studie van Salari en Bednarik (2024) onderzocht de invloed van verschillende lichtomstandigheden en hun effect op de nauwkeurigheid van oog-tracerende software. Er werd gekozen om drie verschillende omstandigheden te testen: donker, normaal en fel licht. Negen deelnemers voerde nauwkeurigheidstesten uit onder deze verschillende omstandigheden. Hieruit bleek dat de lichtomstandigheid

wel degelijk een invloed had op de nauwkeurigheid, normale of gematigde lichtomstandigheden bleken het beste te werken voor dit soort software. Een belangrijke opmerking uit dit onderzoek was dat de uitvoering van de oog-besturing best gebeurd met dezelfde lichtomstandigheden als tijdens de kalibratie.

Ook in de paper van Wolf en Seernani ([2023](#)) wordt licht genoemd als een mogelijke factor voor een verlaging in kwaliteit van de data. Zij concludeerden dat intens zijlicht de precisie verlaagt. Daarnaast benadrukken Wolf en Seernani ([2023](#)) het belang van consistente lichtomstandigheden. In hun paper raden ze aan de kalibratie te doen in dezelfde lichtomstandigheden als het effectieve gebruik. Indien dit niet gebeurd kunnen er afwijkingen ontstaan.

Andere uitdagingen op het gebied van bruikbaarheid

Langdurig kijken naar computerschermen kan negatieve gevolgen hebben voor men hun gezondheid. De ogen van de gebruiker raken vermoeid of kunnen droog aanvoelen (Blehm e.a., [2005](#)). Gezien oogbestuurde software afhankelijk is van de ogen van de eindgebruiker kunnen vermoeide ogen lijden tot een verminderde graad van precisie.

Hiernaast blijkt dat het dragen van een bril een negatieve impact kan veroorzaken op de verkregen data. Gebruikers die een bril dragen hebben gemiddeld een lagere precisie dan gebruikers zonder bril. (Wolf & Seernani, [2023](#))

A.2.4. Mogelijke oplossingen

Verschillende studies rondom webcam eye tracking bieden niet alleen beperkingen van de technologie maar ook methodes hoe deze opgelost kunnen worden.

Kalibratie

De studie van Hassoumi e.a. ([2019](#)) biedt een mogelijke oplossing voor het kalibratie probleem

YOLO object herkenningsmodel

Taore en Dakin ([2023](#)) legt in hun paper uit hoe hun onderzoek met het YOLO ('You Only Look Once') object herkenningsmodel de aanhankelijkheid van kalibratie kan verminderen en hoe er rekening kan gehouden worden met een breder spectrum aan hoofd bewegingen. Hun versie van het YOLO model zorgt voor een beter begrip van de ruimtelijke verhoudingen tussen de ogen, irissen en het hoofd. Hierdoor wordt voorspellen van het blikpunt verstrekt. Volgens de studie is het model zelf in staat om onafhankelijke voorspellingen te doen voor zowel het linkse als rechtse oog waardoor de eye-tracking toegankelijker wordt. De resultaten werden vergelijken met 'WebGazer', een populaire eye-tracker, en de precisie van hun webcam gebaseerde eye-tracker lag hoger dan die van WebGazer.

Hoofdtraceren met een virtuele kinsteen

De studie van Kaduk e.a. (2023) maakte gebruik van hoofdtraceren. Om dit te doen werd een algoritme met een neuraal netwerk gebruikt. Dit algoritme analyseerde de videobeelden door verschillende lagen van het neuraal netwerk. Hierdoor leert het algoritme de verschillende kenmerken van het menselijk gezicht met als gevolg dat het hoofd getraceerd kan worden en de ogen binnen een bepaald gebied kunnen blijven. Deze functionaliteit is vooral behulpzaam in combinatie met de ‘virtuele kinsteen’ die in dezelfde studie gebruikt werd.

De virtuele kinsteen is een simulatie van een fysieke kinsteen. Aan het begin van de studie koos de gebruiker een comfortabele positie en wanneer de persoon hiervan weg bewoog vroeg het systeem om terug te keren naar de vooraf gekozen positie. Dit aan de hand van instructies en feedback die getoond werden op het scherm (Kaduk e.a., 2023).

A.3. Methodologie

Het onderzoek is opgebouwd in vijf fasen, die samen de basis vormen voor het ontwerpen, testen en evalueren van een verbeterde oog-besturing.

A.3.1. Literatuurstudie

Tijdens de literatuurstudie zal grondig onderzoek gedaan worden naar hoe oog-besturing in zijn werk gaat, de grootste problemen rondom deze technologie en mogelijke oplossingen.

A.3.2. Creëren van een gecontroleerde testomgeving

In deze fase van het onderzoek, die parallel loopt met de literatuurstudie, wordt een gecontroleerde testomgeving ontwikkeld, geïnspireerd door de methodologie van Lakshmi Pavani e.a. (2018). Deze testomgeving bestaat uit een website die functioneert als een gesimuleerde webshop. Deze omgeving wordt ontwikkeld met behulp van de programmeertaal HTML en CSS. Deze omgeving zal dienen als testplatform voor de evaluatie van de applicatie. Door een uniforme testomgeving te gebruiken, wordt consistentie in de onderzoeksresultaten gewaarborgd.

Naast deze omgeving zal gekozen worden om zelf een oog-besturing applicatie te ontwikkelen in de programmeertaal Python. Er wordt gekozen voor deze taal wegens de grote hoeveelheid aan bibliotheken met bestaande code die het creëren van zo een applicatie vereenvoudigen. Dit systeem zal parallel met de rest van het onderzoek aangevuld worden met de verschillende oplossingen voor de gevonden problemen.

A.3.3. Creëren van een oplossing

Tijdens deze fase van het onderzoek zullen oplossingen uitgedacht worden voor de gevonden problemen uit de vorige fase. Deze oplossingen zullen voortkomen uit zowel de literatuurstudie als eigen inbreng. Nadat een oplossing is uitgedacht zal deze geïmplementeerd worden in de oog-besturing software.

A.3.4. Experiment uitvoeren en data verzamelen

Nadat dit product verwezenlijkt is kan de voorlaatste fase van start gaan. Hierin zal de extensie getest worden door twee proefpersonen: één zonder motorische beperkingen en één met motorische beperkingen. Beide proefpersonen voeren navigatietaaknissen uit op de gesimuleerde webshop, eerst met behulp van de oog-besturing zonder de geïmplementeerde oplossingen en vervolgens met de geïmplementeerde oplossingen. Tijdens deze experimenten worden de volgende gegevens verzameld:

- De afwijking van kliks buiten de doelgebieden, gemeten als de afstand tot het correcte klikbare element.
- De benodigde tijd om een klik te voltooien.
- De invloed van verschillende lichtomstandigheden.
- Het gebruik van complexere handelingen zoals scrollen.

Deze gegevens bieden inzicht in de effectiviteit van de verbeteringen.

A.3.5. Data analyse

De laatste fase omvat de analyse van de verzamelde data. Om te bepalen of kliks gecentreerd zijn op de juiste elementen, wordt een heatmap gegenereerd die de klikpatronen visueel weergeeft. Daarnaast wordt de gemiddelde tijd berekend die beide proefpersonen nodig hebben om een klick uit te voeren. Hiernaast zullen de metingen uitgevoerd worden in verschillende lichtomstandigheden. Deze resultaten worden met elkaar vergeleken om de impact van de toevoegingen te beoordelen en eventuele verbeterpunten te identificeren.

A.4. Verwacht resultaat

Het verwachte resultaat van dit onderzoek is dat de nauwkeurigheid in eender welke lichtomstandigheid waarmee de individu met motorische beperking de website navigeert hoger ligt dan als deze de initiële oog-besturing gebruikt. Niet alleen de nauwkeurigheid zou hoger liggen maar ook de mate waarin het mogelijk is om complexere interacties uit te voeren zoals scrollen en slepen.

Bibliografie

- AccessibleEU. (2025). The EAA comes into effect in June 2025. Are you ready? https://accessible-eu-centre.ec.europa.eu/content-corner/news/eaa-comes-effect-june-2025-are-you-ready-2025-01-31_en
- Alanazi, F., Ushaw, G., & Morgan, G. (2023). Improving Detection of DeepFakes through Facial Region Analysis in Images. *Electronics*, 13(1), 126. <https://doi.org/10.3390/electronics13010126>
- Association, A. P. (2013). *Diagnostic and Statistical Manual of Mental Disorders* (5de ed.) [DSM-5]. American Psychiatric Publishing.
- Blehm, C., Vishnu, S., Khattak, A., Mitra, S., & Yee, R. W. (2005). Computer Vision Syndrome: A Review. *Survey of Ophthalmology*, 50(3), 253–262. <https://doi.org/https://doi.org/10.1016/j.survophthal.2005.02.008>
- Cerrolaza, J. J., Villanueva, A., & Cabeza, R. (2008). Taxonomic study of polynomial regressions applied to the calibration of video-oculographic systems, 259. <https://doi.org/10.1145/1344471.1344530>
- Chakraborty, P., Roy, D., Rahman, M. Z., & Rahman, S. (2019). Eye Gaze Controlled Virtual Keyboard. *International Journal of Recent Technology and Engineering (IJRTE)*, 8(4), 3264–3269. <https://doi.org/10.35940/ijrte.D8049.118419>
- Contributors, O. (2024). Introduction to OpenCV. <https://docs.opencv.org/4.10.0/d1/dfb/intro.html>
- Contributors, O. (2025). Histogram Equalization — OpenCV Documentation. https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html
- Developers, G. (2024). MediaPipe Solutions Guide. <https://ai.google.dev/edge/mmediapipe/solutions/guide>
- Developers, N. (2024). NumPy: The Fundamental Package for Scientific Computing with Python. <https://numpy.org/>
- Drewes, H., Pfeuffer, K., & Alt, F. (2019). Time- and space-efficient eye tracker calibration, 1–8. <https://doi.org/https://doi.org/10.1145/3314111.3319818>
- Fischer, T., Chang, H. J., & Demiris, Y. (2018). RT-GENE: Real-Time Eye Gaze Estimation in Natural Environments, 339–357. https://doi.org/10.1007/978-3-030-01249-6_21
- Hassoumi, A., Peysakhovich, V., & Hurter, C. (2019). Improving eye-tracking calibration accuracy using symbolic regression (J. Tang, Red.). *PLOS ONE*, 14(3), e0213675. <https://doi.org/10.1371/journal.pone.0213675>

- Jensen, O. B. (2021). Webcam Eye Tracking vs. an Eye Tracker [Pros and Cons]. <https://imotions.com/blog/learning/best-practice/webcam-eye-tracking-vs-an-eye-tracker/>
- Jensen, O. B. (2022). Webcam Eye Tracking vs. an Eye Tracker [Pros and Cons]. <https://imotions.com/blog/learning/best-practice/webcam-eye-tracking-vs-an-eye-tracker/#how-does-webcam-eye-tracking-work>
- Kaduk, T., Goeke, C., Finger, H., & König, P. (2023). Webcam eye tracking close to laboratory standards: Comparing a new webcam-based system and the EyeLink 1000. *Behavior Research Methods*, 56(5), 5002–5022. <https://doi.org/https://doi.org/10.3758/s13428-023-02237-8>
- King, D. E., & Phillips, R. D. (2014a). Example: Finding faces in images with dlib [Ge- raadpleegd op 14/04/2025].
- King, D. E., & Phillips, R. D. (2014b). Example: Finding faces in images with dlib [Ge- raadpleegd op 14/04/2025].
- Lakshmi Pavani, M., Bhanu Prakash, A. V., Shwetha Koushik, M. S., Amudha, J., & Jyotsna, C. (2018, december). Navigation Through Eye-Tracking for Human-Computer Interface. In *Information and Communication Technology for Intelligent Systems* (pp. 575–586). Springer Singapore. https://doi.org/10.1007/978-981-13-1747-7_56
- Iamé, A. (z.d.). GazeTracking. *Github*. <https://github.com/antoinelame/GazeTracking>
- MathWorks. (2025). Contrast Limited Adaptive Histogram Equalization. <https://nl.mathworks.com/help/visionhdl/ug/contrast-adaptive-histogram-equalization.html>
- Miseviciute, I. (z.d.). How do eye trackers work? <https://www.tobii.com/resource-center/learn-articles/how-do-eye-trackers-work>
- Nyström, M., Andersson, R., Holmqvist, K., & van de Weijer, J. (2012). The influence of calibration method and eye physiology on eyetracking data quality. *Behavior Research Methods*, 45, 272–288. <https://doi.org/https://doi.org/10.3758/s13428-012-0247-4>
- OpenCV.org. (2018). Contour Features — OpenCV-Python Tutorials 1 documentation. https://docs.opencv.org/3.4/dd/d49/tutorial_py_contour_features.html
- Salari, M., & Bednarik, R. (2024). Investigating the Impact of Illumination Change on the Accuracy of Head-Mounted Eye Trackers: A Protocol and Initial Results, 204–210. <https://doi.org/https://doi.org/10.1145/3686215.3688383>
- Salvucci, D. D., & Goldberg, J. H. (2000). Identifying fixations and saccades in eye-tracking protocols, 71–78. <https://doi.org/10.1145/355017.355028>
- Sogo, H. (2012). GazeParser: an open-source and multiplatform library for low-cost eye tracking and analysis. *Behavior Research Methods*, 45(3), 684–695. <https://doi.org/https://doi.org/10.3758/s13428-012-0286-x>

- Spiller, M. G., Audi, M., & Bracciali, L. M. P. (2019). Motor performance of children and adolescents with cerebral palsy during the execution of computer tasks with different peripherals. *Revista CEFAC*, 21(4). <https://doi.org/10.1590/1982-0216/20192140319>
- Subbiah, S. (2021). Facial Landmarks Detection using MediaPipe Face Mesh in TensorFlow.js. <https://selvamsubbiah.com/mediapipe-facemesh-in-tensorflow-js/>
- Sweigart, A. (z.d.). PyAutoGUI Documentation. <https://pyautogui.readthedocs.io/en/latest/>
- Taore, A., & Dakin, S. C. (2023). Enhancing Eye Tracking with a YOLO Inspired Network. <https://doi.org/https://doi.org/10.31219/osf.io/eqtak>
- TREWIN, S., & PAIN, H. (1999). Keyboard and mouse errors due to motor disabilities. *International Journal of Human-Computer Studies*, 50(2), 109–144. <https://doi.org/10.1006/ijhc.1998.0238>
- Valliappan, N., Dai, N., Steinberg, E., He, J., Rogers, K., Ramachandran, V., Xu, P., Shojaeizadeh, M., Guo, L., Kohlhoff, K., & Navalpakkam, V. (2020). Accelerating eye movement research via accurate and affordable smartphone eye tracking. *Nature Communications*, 11(1). <https://doi.org/https://doi.org/10.1038/s41467-020-18360-5>
- Van der Cruyssen, I., Ben-Shakhar, G., Pertzov, Y., Guy, N., Cabooter, Q., Gunschera, L. J., & Verschuere, B. (2023). The validation of online webcam-based eye-tracking: The replication of the cascade effect, the novelty preference, and the visual world paradigm. *Behavior Research Methods*, 56(5), 4836–4849. <https://doi.org/https://doi.org/10.3758/s13428-023-02221-2>
- WisFaq. (2005). Euclidische en rechthoekige afstand. [https://www.wisfaq.nl/show3archive.asp?id=38464&j=2005#:~:text=De%20Euclidische%20afstand%20is%20inderdaad,+\(y-v\)%5E2](https://www.wisfaq.nl/show3archive.asp?id=38464&j=2005#:~:text=De%20Euclidische%20afstand%20is%20inderdaad,+(y-v)%5E2).
- with Code, P. (2025). Gaze Estimation on MPII Gaze | Papers with Code. <https://paperswithcode.com/sota/gaze-estimation-on-mpii-gaze?p=rt-gene-real-time-eye-gaze-estimation-in?style=square>
- Wolf, K., & Seernani, D. (2023). Webcam Based Eye Tracking -Whitepaper v3. <https://doi.org/10.13140/RG.2.2.22875.54562>