# Inhoudsopgave

[Inhoudsopgave 1](#_Toc30677672)

[Mijn stageplaats 2](#_Toc30677673)

[1 Mijn stagecontract 3](#_Toc30677674)

[1.1 Stage-opdracht 3](#_Toc30677675)

[1.2 Input 3](#_Toc30677676)

[1.3 Output 3](#_Toc30677677)

[1.4 Criteria om succes te meten 3](#_Toc30677678)

[1.5 Wat zal jouw stage-opdracht niet realiseren? 3](#_Toc30677679)

[1.6 Planning 3](#_Toc30677680)

[2 Stageverslag 4](#_Toc30677681)

[2.1 Verslag stageweek 1 & 2 4](#_Toc30677682)

[3 Tussentijdse evaluatie 7](#_Toc30677683)

[3.1 Verslag 7](#_Toc30677684)

[3.2 Werkpunt 7](#_Toc30677685)

[4 Eindreflectie 8](#_Toc30677686)

# https://www2.howest.be/musicrun/images/logo_howest.jpgMijn stageplaats

Stagebedrijf: HITlab

Adres: Graaf Karel de Goedelaan 32

Stagebegeleider bedrijf: Demanet Jelle

Contactgegevens: [jelle.demanet@gmail.com](mailto:jelle.demanet@gmail.com)

Stagebegeleider Howest: Segers Nathan

Website: http://www.hitlab.be/

# https://www2.howest.be/musicrun/images/logo_howest.jpgMijn stagecontract

Deze gegevens worden ook aangevuld op <https://stage.howest.be>

## Stage-opdracht

Bij het HITlab verwerken ze data van zowel vaste als mobiele eyetrackers, het manueel labelen hiervan is een tijdrovend proces. Het project bestaat daarom uit het automatiseren hiervan. Via AI objectherkenning kunnen Area of Interests automatisch gelabeld worden. De huidige software die het HITlab gebruikt heeft nog andere tekortkomingen, er kunnen geen heatmaps mee gemaakt worden die aan de eisen voldoen. Dit project zal deze functionaliteit dan ook uitbreiden in de vorm van Python code.

## Input

De data komt uit de Tobii Pro Labs software in de vorm van een .tsv data export. Alle software zal zodanig geschreven worden opdat ze kunnen draaien op de aanwezige hardware in het HITlab. De software zal voornamelijk bestaan uit python code en enkele ipython notebooks.

## Output

Een programma geschreven in python dat met behulp van object herkenning eyetracking automatisch data kan labelen alsook heatmaps kan maken. De user zal deze software kunnen bedienen door middel van een grafische interface.

## Criteria om succes te meten

Op het einde van de stage zal ik minstens volgende zaken gemaakt hebben.

Installatie en gebruikershandleiding

Python scripts om:

tsv eyetracking data te parsen

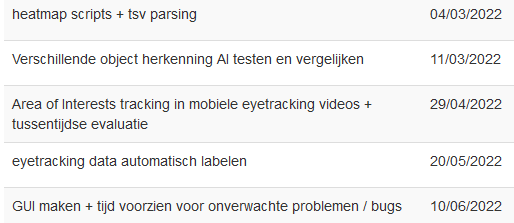
heatmaps te maken en overlayen op video files

eyetracking data automatisch te labelen door object herkenning

## Wat zal jouw stage-opdracht niet realiseren?

Mijn opdracht is voldoende afgebakend.

## Planning



# https://www2.howest.be/musicrun/images/logo_howest.jpgStageverslag

## Verslag stageweek 1 & 2

### Realisaties

Tijdens de eerste week van mijn stage heb mij vooral ingelezen bij HITlab en mijn stageopdracht beter kunnen begrijpen aan de hand van real world data. Ik heb de eerste dagen vooral research gedaan naar hoe eyetracking effectief werkt. Meer daarover in het research hoofdstuk.

De 2de week heb ik al een paar mooie dingen kunnen maken. Ik heb al verschillende miniprojectjes gemaakt die de basis zullen vormen voor mijn stageopdracht.

|  |  |  |  |
| --- | --- | --- | --- |
| Naam | Wat het doet | Waarom ik het heb gemaakt | Werkt het al? |
| Blinkdetector | Detecteert oogknippers en logt die naar de console | Eerste test met de pupil labs core eyetracker en API | ✔️ |
| EyetrackingEffnet | Annoteert objecten in een video van de world camera van de pupil core bril | Word gebruikt door mij om de verschillende efficientnet architecturen te testen op snelheid en accuraatheid | ✔️ |
| EffNet | Gebruikt [google’s automl modellen](https://github.com/google/automl) om object herkenning te doen | Eerste implementatie van efficientnet, dit projectje heb ik gebruikt om vertrouwd te geraken met de werkwijze en de API | ✔️ |
| VasteEyetrackerHeatmap | Creert een heatmap en projecteert deze over de video die de proefpersoon heeft bekeken | Hiermee kan HITlab heatmaps genereren op basis van Tobii Pro Labs tsv exports. Dit zal uitgebreid worden om ook heatmaps van mobiele eyetracking data te maken | ✔️ (hier en daar nog een paar kleine aanpassingen nodig) |
| MovingEyetrackerHeatmap | Dit project zal heatmaps kunnen maken van bewegende eyetracking data | Dit is eigenlijk het einddoel van mijn stage, er is nog heel veel werk aan. | ❌ |

Chart, treemap chart

Description automatically generated

Figuur - Optimazing Code

Iets wat ik op school nog niet heb moeten doen is code optimaliseren op snelheid. Op figuur 1 zie je hoelang elke functie call er over doet om uitgevoerd te worden. Hieruit kan je dan afleiding waar je je code moet herschrijven.

Ik heb ook al verschillende meetings kunnen meedoen. Bij een daarvan ben ik naar [Waak](https://www.waak.be/nl) geweest, zij zijn bezig om met behulp van eyetracking een test te ontwikkelen die hun medewerkers kan helpen ondersteunen, bijvoorbeeld bij het maken van duidelijkere instructies. De eyetracker die ze daarvoor gebruiken is een mobiele eyetracker van Tobii.

Ik heb ook samengezeten met studenten van UHasselt. Zij gebruiken ook een Tobii eyetracker en hadden daar enkele problemen mee.

### Diagram Description automatically generatedResearch

De eyetrackers waar ik mee werk gebruiken 2 principes om je ogen te tracken, het dark en bright puil effect (figuur 2). Ze zenden infrarood licht in je ogen en meten de weerspiegeling. Afhankelijk van de weerkaatsingshoek wordt bepaald waar je naar kijkt.

Je hebt meerdere soorten oogbewegingen. Voor mijn project ben ik vooral geïnteresseerd in fixaties, dit zijn momenten waarin je ogen op 1 vast punt fixeren om informatie te verwerken. Deze fixatiepunten gebruik ik om heatmaps te plotten.

Figuur 2 – Dark vs Bright pupil effect

Voor heatmaps te maken van data afkomstig van mobiele eyetrackers is er nogal veel wiskunde nodig. Bij een vaste eyetracker heb je geen ‘ego motion’.

Alle datapunten vallen binnen hetzelfde vlak en zijn daar gemakkelijk op te plotten. Het is ook onmogelijk om uit 2d videobeelden de 3d bewegingen van de camera te verkrijgen. Het proces om uit beelden beweging en oriëntatie in te schatten heet [Visual Odometry](https://en.wikipedia.org/wiki/Visual_odometry).

A picture containing text, window, door, tiled

Description automatically generatedEr zijn verschillende methodes om de beweging te verkrijgen, waaronder [optical flow](https://en.wikipedia.org/wiki/Optical_flow). Waarbij de verschuiving van elke pixel gemeten wordt en op die manier een vectorveld kan worden gemaakt (zie figuur 2).

Figuur 3 – Vector field from optical flow

Er is ook veel AI-research over Visual Odometry. De meeste projecten werken met stereocamera’s omdat je dan dieptezicht hebt. De mobiele eyetrackers waar ik mee werk hebben maar 1 camera, dit is een extra moeilijkheid.

Een aanpak die ik wel interessant vind heet [Unsupervised Structure-from-motion (SfM).](https://towardsdatascience.com/unsupervised-structure-from-motion-sfm-demystified-b1f222220ad6) Echter ga ik het eerst proberen met de opencv library van python. Hier zitten functies in de onder andere de transformatie en translatie matrices kunnen bereken van 2 frames. Die matrices kan ik dan toepassen op de datapunten.

### Feedback

Voor mijn eerste project dat met data van een vaste eyetracker heatmaps maakt kreeg ik als feedback dat dit nog niet helemaal accuraat was. Verschillende punten werd fout geprojecteerd op de video. Ondertussen is dat al verbeterd maar het is nog niet perfect.

### Planning voor de komende twee weken

De volgende twee weken ga ik focussen op de heatmaps van een mobiele eyetracker, dit is niet zoals de originele planning maar ik heb ondervonden dat dit een stuk moeilijker is dan gedacht. Daarom wil ik dit deel eerst afhebben voor ik verder ga.

## Verslag stageweek 3 & 4

### Realisaties

Deze 2 weken ben ik erin geslaagd om heatmaps te maken van mobiele eyetracker data. Er is nog een beetje werk de heatmaps er beter te laten uitzien, dit is vooral wat parameters tunen.

A picture containing graphical user interface

Description automatically generated

Figuur 4 - Heatmap overlay of real-world data

Op figuur 4 kun je het resultaat zien. Het proces om de panorama foto maken vanuit de frames in de input video is heel tijdrovend. Om dit te versnellen heb ik de opencv module zelf gecompiled met Nvidia CUDA ondersteuning, hierdoor kan ik mijn grafische kaart gebruiken om complexe berekening te doen en kan de applicatie in realtime draaien.

Text

Description automatically generated

Figuur - Speed difference CPU & GPU

Ik heb ook een projectmap gemaakt voor de uiteindelijke applicaties met daarin volgende python klasses:

Text

Description automatically generated

Figuur - folder structure

ConverterGPU verwerkt de ruwe data naar een universeel formaat dat geschikt is om op de panorama te plotten, Converter doet het zelfde maar dan op de CPU, zoals je in figuur 5 kunt zien is de GPU sneller.

De converter klasses erven van Datahandler, die zal bij initialisatie de ruwe eyetracking data verwerken.

HeatmakMaker is de overkoepelende klasse van de converter klasses, deze maakt met de data van de converters de heatmaps

GUI klasses is de klasse waar de gebruikersinterface in gebouwd zal worden.

Er zullen nog klasses bijkomen om analyses op de data te doen, zoals Area of Interests maken.

### Research

Deze 2 weken heb ik vooral research gedaan naar gpu programming en de sift en surf algoritmes.

SIFT (Scale Invariant Feature Transform) kan features in afbeelding detecteren, beschrijven en herkennen.

Een goede beschrijving van hoe het werkt kun je [hier](https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/) vinden.

Ik gebruik sift in de Converter klasse om per frame van de eyetracker video keypoints te localiseren en deze te matchen met de panorama foto. Vanuit de 2 lijsten met keypoints bereken ik de transformatiematrix M waarmee ik elke willekeurig punt in het frame kan transformeren naar de panorama foto. Deze matrix pas ik dan toe op de data van de eyetracker.

Dit proces is echter traag en vergt veel rekenkracht. In opencv kun je ook SURF gebruiken, dit is een gelijkaardig aan sift, alleen werkt het veel sneller en kan draaien op de GPU wat het nog sneller maakt.

Een probleem met de GPU gebruiken is de architectuur van de computer. De CPU en GPU hebben alle 2 een apart geheugen en communiceren via een PCI-E bus. Deze communicatie zorgt voor veel overhead. Elk frame moet eerst geüpload worden naar de GPU, daar verwerkt worden en dan terug gedownload worden naar de CPU. Ik heb dit onderzocht en het blijkt dat de GPU gebruiken toch de betere manier is, de extra overhead en de omslachtige manier van code (zie figuur 7) wegen niet op tegen de rekenkracht van de GPU (zie figuur 5).

A screenshot of a computer

Description automatically generated with medium confidence

Figuur - Code to acces GPU

Ik heb ook een idee om mijn algoritme helemaal te veranderen om het nog sneller en efficiënter te doen draaien.

In plaats van elk frame apart te verwerken zou ik frames per batch op de GPU willen plaatsen, gelijkaardig aan een neuraal netwerk trainen, dit is helaas iets dat in python moeilijk te implementeren valt. Python is te high level hiervoor. Daarom zou ik [CUDA-C](https://developer.nvidia.com/blog/easy-introduction-cuda-c-and-c/) willen gebruiken, dat is een C taal voor grafische kaarten. Dit zal eerst verder onderzocht moeten worden.

### Feedback

/

### Planning voor de komende twee weken

Ondertussen zit ik terug op schema, de volgende weken ga ik mogelijkheden bouwen om de eyetracker data te analyseren en te onderzoeken. Er staan ook een paar workshops gepland om de projecten van het HITlab voor te stellen.

# Tussentijdse evaluatie

*Halverwege de stageperiode zit je samen met jouw stagebegeleider. Je bespreekt de voorbije periode. Gebruik onderstaande vragen om jouw stage te evalueren.*

* *Hoe is het inwerken in de stage-opdracht verlopen?*
* *Welke technische aspecten verliepen moeilijker dan verwacht?*
* *Kon je zich snel inwerken in de technologie? Waar is nog meer research nodig?*
* *Kon de vooropgestelde timing behouden blijven?*
* *Verloopt de communicatie vlot? Kan je met jouw vragen vlot bij iemand terecht?*
* *Krijg je voldoende feedback?*
* *Is er ruimte voor uitbreidingen aan jouw stage-opdracht?*

## Verslag

<Maak van jouw tussentijdse evaluatie een kort verslag.>

## Werkpunt

<Spreek met jouw stagebegeleider minimaal één werkpunt af waar je in de komende periode aan zal werken. Dat kan ook over niet-technische aspecten gaan (voorbeelden: meer aan de planning houden, duidelijker communiceren, sneller feedback vragen, meer open staan voor suggesties, beter verslaggeving, etc.). Sta ook even stil hoe je dat zal verbeteren.>

# Eindreflectie

<Bespreek de eindrealisatie:

* Welke aspecten van jouw stage-opdracht zijn afgerond?
* Welke zaken zijn niet gerealiseerd?
* Welke zaken zijn anders verlopen dan verwacht?
* Wat heb je bijgeleerd?
* Zijn er extra zaken bij jouw takenpakket bijgekomen?
* Schrijf hier ook een dankwoord aan jouw stagebedrijf>