Technical Report

Elias William
Rasmus Olofsson
Malva Eveborn
Oskar Ramsberg
Mohammad Alasmi
Adam Roos

December 18, 2023

Version 1.0

Status

Reviewed	Elias William	2023-12-18
Approved	Martin Skoglund	2023-12-18

Oticon
eliwi204@student.liu.se

Project Identity

Group E-mail:	eliwi204@student.liu.se
Homepage:	http://www.isy.liu.se/tsrt10/group
Orderer:	Martin Skoglund, ISY/Eriksholm Research Centre Phone: +46 13 28 18 90 E-mail: martin.skoglund@liu.se
Customer:	Sergi Rotger Griful, Oticon Phone: - E-mail: segr@eriksholm.com
Supervisor:	Johanna Wilroth, ISY Phone: +46 13-28 28 05 E-mail: johanna.wilroth@liu.se
Course Responsible:	Daniel Axehill & Gustaf Hendeby Phone: +46 013-28 40 42 & +46 13-28 58 15 E-mail: daniel.axehill@liu.se & gustaf.hendeby@liu.se

Participants of the group

Name	**Responsible**	**E-mail**
Elias William	Project Leader (PL)	eliwi204@student.liu.se
Rasmus Olofsson	Responsible for software (SW)	rosol028@student.liu.se
Malva Eveborn	Responsible for documentation (DOC)	malev556@student.liu.se
Oskar Ramsberg	Responsible for design (DES)	oskra262@student.liu.se
Mohammad Alasmi	Responsible for hardware (HW)	mohal555@student.liu.se
Adam Roos	Responsible for testing (TEST)	adaro336@student.liu.se

CONTENTS

DOCUMENT HISTORY

Version	Date	Changes made	Sign	Reviewer
0.1	2023-12-14	First draft.	All	EW
1.0	2023-12-18	First version.	MA	EW

1 INTRODUCTION

As a collaboration between Linköping University and Oticon, a company developing hearing aid technology, this project has been an ongoing development process for the recent years. This document describes the progress that has been done in this years (2023) project. The new modules are described, the results discussed and suggestions for further improvements are made. This project is part of the course TSRT10 at Linköping University and follows the project model LIPS.

1.1 Definition of terms

Table 1 lists the abbreviations used throughout the document.

Table 1: Definition of terms.

Terms	Meaning
G3	Tobii Pro Glasses 3.
GUI	Grafical User Interface, where the user can interact with the program.
UI	User Interface, a non-grafical program where the user can interact with the program.
HA	Hearing Aid.
IMU	Inertial Measurement Unit.
EKF	Extended Kalman Filter.
LSL	Lab Streaming Layer.
DOA	Direction of Arrival.
TDOA	Time Difference of Arrival.
RSTP	Real Time Streaming Protocol.

2 SYSTEM OVERVIEW

In this section a general overview of the system will be presented. The main hardware used in the system are two hearing aid devices and a pair of Tobii Pro Glasses 3. The system are separated into different modules. The main modules are:

- Sound Source Tracking
- Beamforming
- Orientation Estimation
- Gaze Tracking
- Face Tracking

Figure 1 shows how the modules are connected. Further explanation of each module can be found in the respective section for each module.

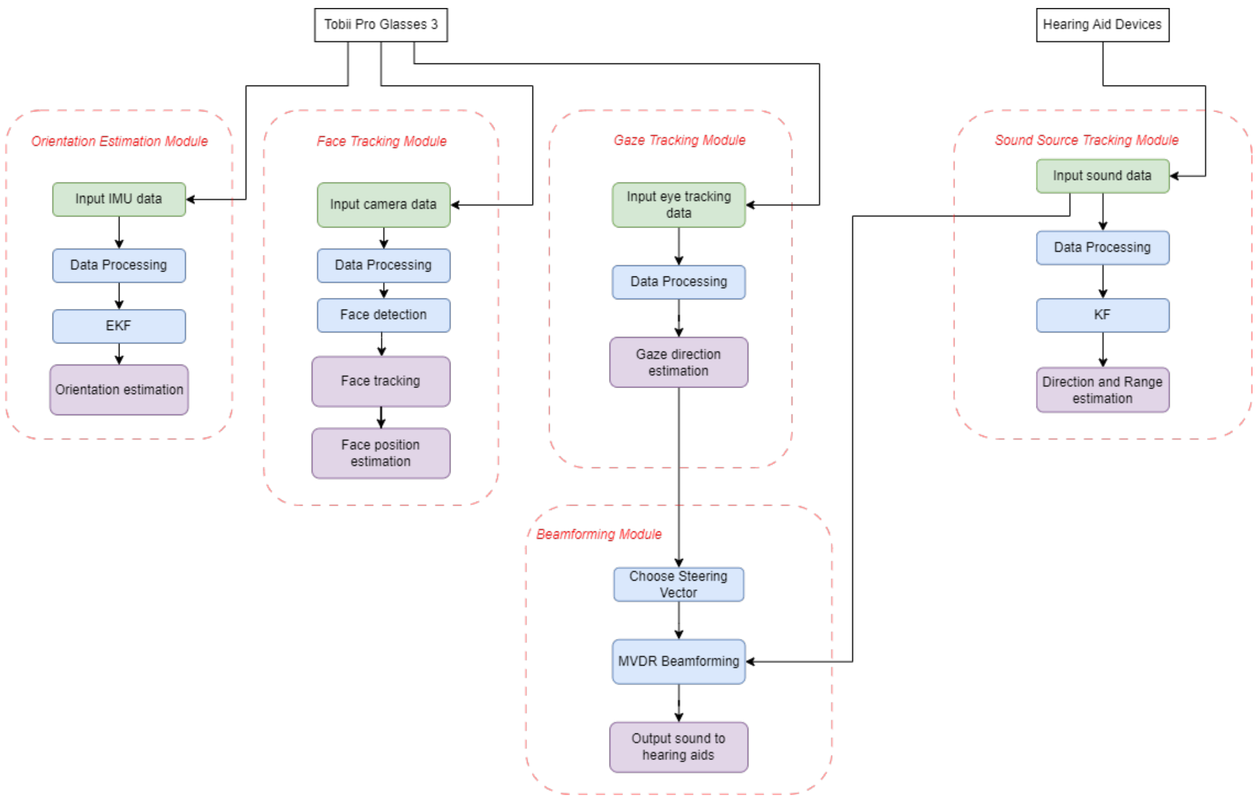

Figure 1: System overview.

2.1 Hardware

In this section an brief explanation of all the hardware that is used will be presented.

2.1.1 *Tobii Pro Glasses 3*

The Tobii Pro Glasses 3 consists of a facing camera, eye-tracker, microphones and an Inertial Measurement Unit (IMU). The glasses are powered by a battery pack. Additional hardware for the Tobii Pro Glasses 3 are a hard drive with SD-card slot, equipment for Qualisys and calibration tools. An overview of the glasses can be seen in figure 2.

2.1.2 *Hearing Aid*

The two hearing aid devices consists of two microphones and one in-ear earbud each. A phono pre-amplifier of model type "NANO-LP1" is used to amplify the sound from the microphones from the hearing aid devices. Two amplifiers are used, one for each earpiece. The amplifiers consists av two input ports and two output ports each. The input and output from the hearing aids is connected through a sound card of the model "MAYA44 USB+" which have 4-inputs and 4-outputs. The sound card has a USB connection to connect to a computer device where the input signals can be processed. The hearing aid devices are powered by a 1.5 voltage battery through a circuit board.

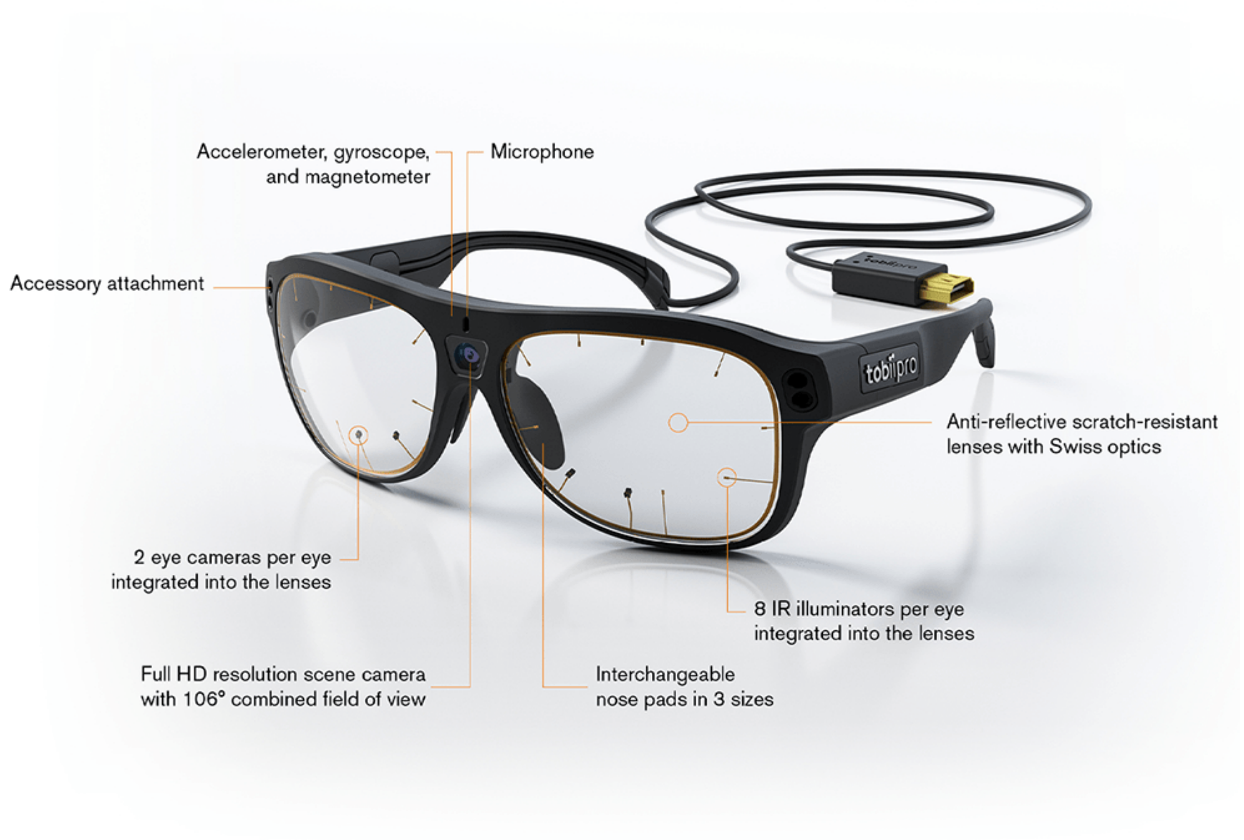
Accelerometer, gyroscope, and magnetometer
Microphone
Accessory attachment
Anti-reflective scratch-resistant lenses with Swiss optics
2 eye cameras per eye integrated into the lenses
8 IR illuminators per eye integrated into the lenses
Full HD resolution scene camera with 106° combined field of view
Interchangeable nose pads in 3 sizes

Figure 2: Tobii Pro Glasses 3 overview [1].

3 SOUND SOURCE TRACKING

This section will focus on the methods used for the Sound Source Tracking module. The module estimates a direction and range to a sound source. An overview of the sound source tracking algorithm can be seen in figure 3. The Source Tracking Module uses the microphones on the left and right hearing aid. The two microphones on the right hearing aid are labeled as one and two and are assumed to be at the same location. This is also assumed for the microphones on the left hearing aid which are labeled as three and four.

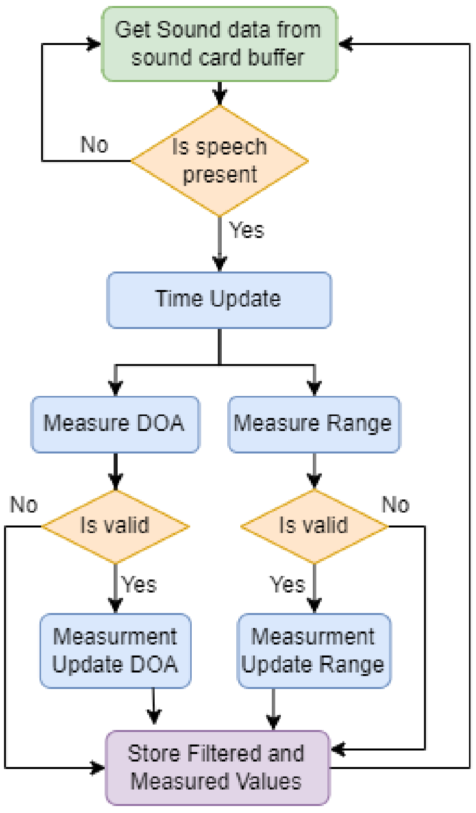

Figure 3: The flowchart of the sound source tracker.

3.1 Voice Detection

The voice detection in the Sound Source Tracking module uses a pretrained deep learning neural network model that can detect speech in a audio signal. The pretraind network "vadnet" is developed by MathWorks, Inc and can be found at the vadnet page at MathWorks.com. The output from the voice detection give a speech probability that is used to determine if speech is present or not. If the speech probability for the signal is higher than the threshold 80% then speech is present in the audio signal. The audio signal is a one second recording from one of the microphones in the hearing aids.

3.2 Time Distance Of Arrival

All the four audio signals from the four microphones, corresponding to one second, are first normalized. This is done by first removing the mean value and then divide the signals by their corresponding standard deviation. Four different cross-correlations are thereafter calculated. The signal from microphone one is cross-correlated with the signal from microphone three, same goes for microphone one with microphone four, microphone two with microphone three and microphone two with microphone four. The cross-correlation between two signals f and g are defined by

$$(f \star g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[m+n],$$

where where n is the lag. Four different time differences when the sound hits the left hearing aid compared to the right hearing aid are thereafter calculated according to

$$TDOA_k = \frac{c}{f_s} \underset{n}{\operatorname{argmax}} (s_i \star s_j)[n],$$

where $c = 343$ is the speed of sound, $f_s = 48000$ is the sampling frequency and n is the lag. $s_i \star s_j$ is the cross correlation between the audio signals s_i and s_j from microphone i and j. The mean value for the four different TDOA values are thereafter calculated to get one TDOA value.

3.3 Direction Of Arrival

Rayleigh's spherical head model in horizontal plane expresses the TDOA with the DOA angle as

$$TDOA = \frac{d}{2}(\theta_{DOA} + \sin\theta_{DOA}), \quad -\frac{\pi}{2} < \theta_{DOA} < \frac{\pi}{2},$$

where θ_{DOA} is the Direction of Arrival (DOA) angle, d is the distance between the microphones and TDOA is the Time Difference of Arrival.

3.4 Range

The range measurements to the sound source utilizes The Inverse Square Law for sound waves. The Inverse Square Law can be expressed as $A_{max} \propto \frac{1}{r^2}$, where r is the range to the sound source and A_{max} is the mean value of the maximum amplitude of the four audio signals. All the four audio signals from the four microphones corresponds to one second. A_{max} at different ranges to the sound source have been tested and a interpolation between these points have been calculated. The interpolation are used to calculate the range to the sound source given A_{max} at a given time.

3.5 Kalman Filter

To improve the quality and smoothness of the estimated DOA and range measurements, a Kalman Filter is applied where the following dynamic system is considered

$$\begin{aligned} x_{k+1} &= Fx_k + Gv_k, \\ y_k &= Hx_k + e_k, \end{aligned}$$

where v_k and e_k are assumed to be zero-mean Gaussian distributed with cross-covariance Q and R respectively, and

$$x_k = \begin{pmatrix} DOA_k \\ Range_k \end{pmatrix}, \quad y_k = \begin{pmatrix} DOA_k \\ Range_k \end{pmatrix}, \quad \hat{x}_{0|0} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \quad \hat{P}_{0|0} = I_2.$$

The motion model parameters are given by

$$F = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad G = \begin{pmatrix} dt & 0 \\ 0 & dt \end{pmatrix}, \quad Q = \begin{pmatrix} 10 & 0 \\ 0 & 0.5 \end{pmatrix}.$$

The sensor model parameters are given by

$$H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad R = \begin{pmatrix} 60 & 0 \\ 0 & 5 \end{pmatrix}.$$

3.5.1 *Outlier Rejection*

Before a measurement update on a new measurement is made, a Mahalanobis-distance test is made to detect and reject noisy or incorrect measurements from e.g. loose cables. The Mahalanobis-distance is calculated using following equation

$$\gamma_{k+1} = (\mathbf{z}_{k+1} - \hat{\mathbf{z}}_{k+1})^T \mathbf{S}^{-1} (\mathbf{z}_{k+1} - \hat{\mathbf{z}}_{k+1}) \sim \chi^2_m,$$

where γ_{k+1} is the Mahalanobis-distance, $\mathbf{z}_{k+1}$ represents the new measurement, $\hat{\mathbf{z}}_{k+1}$ is the predicted measurement, and $\mathbf{S}$ is the covariance matrix of the innovation. A probabilistic threshold T is thereafter applied depending on the type of measurement received. If the Mahalanobis-distance is smaller then the threshold, the new measurement is ignored and no measurement update is preformed. The threshold values used are $T^{DOA} = 10$ and $T^{Range} = 1$.

4 BEAMFORMING

The Beamformer module is based on a Matlab script developed by Poul Hoang at Oticon. The theory behind the beamformer script are explained in the article "Multichannel Speech Enhancement With Own Voice-Based Interfering Speech Suppression for Hearing Assistive Devices" [2]. The original script uses own voice to estimate the steering vectors for the beamformer. This have instead been modified so that the steering vectors are estimated from a recorded scene. The data from the recorded scene contains multiple steering vectors were each steering vector corresponds to a direction. The direction of the beamformer can be controlled live by using the corresponding steering vector for that direction. The direction can be set by e.g. the gaze.

5 ORIENTATION ESTIMATION

This section will describe the methods and techniques used for the orientation module. The main function of the module is estimation of the orientation angles, yaw, pitch and roll. This module primary works on recorded data.

5.1 Sensor Model

Accelerometer, gyroscope, and magnetometer data is collected from the glasses IMU. The accelerometer and gyroscope data are sampled at a frequency of 100 Hz and the magnetometer at a frequncy of 10 Hz. Each sample contains the sample time T_s and the measurement of the x-, y- and z-component for respectively sensor. The magnetometer data comes in a lump of measurement with different sample times.

The sensor fusion model used in the orientation module is considered as

$$x_{k+1} = f(x_k, u_k, w_k)$$

$$y_{k+1} = h(x_k, u_k)$$

$$x_k = q_k, \qquad u_k = \omega_k, \qquad y_k = \begin{pmatrix} y_k^a & y_k^m \end{pmatrix}$$

where q_k is the quaternion representation of a rotation, ω_k, y_k^a and y_k^m is the measured gyroscope, accelerometer and magnetometer data respectively and w_k is the process noise.

5.2 Rotations using Quaternions

The global frame G is the earth's fix right-hand coordinates where the z-axis is parallel to the direction of gravitational acceleration. The body frame B is the local coordinates of the G3 glasses in the testing environment. Since two different frames are considered there needs to be a rotation between the frames. To this a unit length quaternion representation is used. The quaternion representation of a rotation is used as an axis-angle representation of the rotation as

$$q = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix}$$

Where q_0 is scalar and $q_{1,2,3}$ are complex units.

5.3 Calibration

The magnetometer data is calibrated before using the measurements to estimate the rotation angles. The following calibration matrices are used

$$A_{mag,cal} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$b_{mag,cal} = \begin{pmatrix} 41.13 & -119.70 & 11.29 \end{pmatrix}$$

To calibrate the measurements of the magnetometer b_{cal} is subtracted from the measurements and multiplied with A_{cal}.

$$y_{mag,cal} = (y_{mag} - b_{mag,cal})A_{mag,cal}$$

An important thing to keep in mind is to think about what is affecting the magnetometer. All calibration done for the magnetometer were performed indoors, which maybe not is the most suitable environment considering the surrounding materials that can disturb the magnetometer of the glasses. However, being outside or in a magnetic isolated room should be the most optimal environment to calibrate the magnetometer.

The gyroscope is also calibrated in the beginning of the runn the orientation module is run and is done by standing still with the glasses on the head for 20 seconds. Data is collected and the bias is calculated

$$y_{gyr,cal} = (y_{gyr} - b_{gyr,cal})$$

5.4 Extended Kalman Filter

Data from the IMU is streaming continuously containing accelerometer, gyroscope and magnetometer data. When data is obtained an Extended Kalman Filder (EKF) algorithm is used to estimate the rotation represented by the unit quaternion. The EKF algorithm uses the gyroscope data as input to do the time update and the accelerometer and magnetometer to update the measurements.

5.4.1 *Time Update*

The time update function updates the quaternion states according to

$$x_{k+1} = F_k x_k$$

To get the quaternion representation of the orientation to work the quaternion is of unit length by normalize it

$$x_{k+1} = \frac{x_{k+1}}{||x_{k+1}||}$$

The covariance matrix of the time update are calculated as

$$P_{k+1} = F_k P_k F_k^T + G_k R_\omega G_k^T$$

To compute the sensor model parameters the gyroscope input signal ω is used

$$F_k = I_4 \frac{1}{2} S_{\omega,k}(\omega) T_s$$

$$G_k = \frac{T_s}{2} S_{\omega,k}(\omega)$$

$$S_\omega(\omega) = \begin{pmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & -\omega_y \\ \omega_z & \omega_y & -\omega_x & 0 \end{pmatrix}$$

The process noise for the time update is defined as R_ω and is a tuning parameter.

$$R_\omega = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{pmatrix}$$

5.4.2 *Measurement Update*

The measurement update uses data from the accelerometer and the magnetometer. The update functions is described as

$$y_k^a = R_k^{GB}(q)^T g_0 + e_k^a$$

$$y_k^m = R_k^{GB}(q)^T m_0 + e_k^m$$

for the accelerometer and the magnetometer respectively.

To convert from a unit length quaternion to a rotation matrix the R^{GB} matrix is used

$$R^{GB}(q) = \begin{bmatrix} 2q_0^2 - 1 + 2q_1^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & 2q_0^2 - 1 + 2q_2^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & 2q_0^2 - 1 + 2q_3^2 \end{bmatrix}$$

By using the Extended Kalman Filter algorithm the orientation angles can be estimated pretty well. Important to mention is that the covariance matrices, R_ω, R_a and R_m, can be tuned to find a suitable weighting for the estimation of the orientation. The given values for the matrices in this documentation was the one most suitable for the experiments performed during this project.

An Extended Kalman Filter algorithm is applied to filter the measurement. The prediction error ε_{k+1} for the accelerometer and magnetometer are defined as

$$\varepsilon_k^a = y_k - R_k^{GB}(q)^T g_0, \qquad \varepsilon_k^m = y_k - R_k^{GB}(q)^T m_0$$

where g_0 is the nominal gravity vector and m_0 is the magnetic field vector in Linköping. They are expressed as

$$g_0 = (0 \quad 0 \quad 9.81)^T \quad [m/s^2]$$

$$m_0 = 10^{-3} \cdot (15657 \quad 1728 \quad 48954)^T \quad [\mu T]$$

The Kalman gain can calculated by using the inverse of the prediction covariance S_k and the updated covariance estimate P_k

$$K_k = P_k H_k' S_k^{-1}$$

The prediction covariance S_k is computed in the following way by including the updated covariance estimate P_k. The process noise for the measurement update is tuning parameter and defined as R_a and R_m for the accelerometer and magnetometer respectively.

$$S_k = H_k' P_k H_k'^T + R$$

$$R = \begin{cases} R_a & if \quad accelerometer \\ R_m & if \quad magnetometer \end{cases}$$

$$R_a = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad R_m = \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{pmatrix}$$

H' is the derivative of the rotation matrix Q with respect to q. Taking the derivative with respect to each component individually results in four three-dimensional tensors

$$\frac{dQ}{dq_0} = 2\begin{pmatrix} 2q_0 & -q_3 & q_2 \\ q_3 & 2q_0 & -q_1 \\ -q_2 & q_1 & 2q_0 \end{pmatrix}, \quad \frac{dQ}{dq_1} = 2\begin{pmatrix} 2q_1 & q_2 & q_3 \\ q_2 & 0 & -q_0 \\ q_3 & q_0 & 0 \end{pmatrix}$$

$$\frac{dQ}{dq_2} = 2\begin{pmatrix} 0 & q_1 & q_0 \\ q_1 & 2q_2 & q_3 \\ -q_0 & q_3 & 0 \end{pmatrix}, \quad \frac{dQ}{dq_3} = 2\begin{pmatrix} 0 & -q_0 & q_1 \\ q_0 & 0 & q_2 \\ q_1 & q_2 & 2q_3 \end{pmatrix}$$

Each tensor is multiplied with g_0 in the update for the accelerometer and m_0 in the update for the magnetometer and then stacked in a new matrix H' representing the derivative of the rotation matrix R^{GB}.

$$H'_{a,0} = Q'_0\, g_0, \quad H'_{a,1} = Q'_1\, g_0, \quad H'_{a,2} = Q'_2\, g_0, \quad H'_{a,3} = Q'_3\, g_0$$

$$H'_a = \begin{pmatrix} H'_{a,0} & H'_{a,1} & H'_{a,2} & H'_{a,3} \end{pmatrix}$$

$$H'_{m,0} = Q'_0\, m_0, \quad H'_{m,0} = Q'_1\, m_0, \quad H'_{a,2} = Q'_2\, m_0, \quad H'_{a,3} = Q'_3\, m_0$$

$$H'_m = \begin{pmatrix} H'_{m,0} & H'_{m,0} & H'_{a,2} & H'_{a,3} \end{pmatrix}$$

$$H' = \begin{cases} H'_a & if \quad accelerometer \\ H'_m & if \quad magnetometer \end{cases}$$

The measurement update for both the accelerometer and the magnetometer can be written as

$$x_{k+1} = x_k + K_k \varepsilon^a_k$$

$$x_{k+1} = x_k + K_k \varepsilon^m_k$$

$$P_{k+1} = P_k - (K_k H'_k P_k)$$

To get the quaternion representation of the orientation to work, the quaternion is of unit length by normalize it

$$x_{k+1} = \frac{x_{k+1}}{||x_{k+1}||}$$

5.4.3 *Outlier rejection*

Before any measurement update is made in the algorithm, a check of any outlier is made. In the case of updating accelerometer measurements, the criteria for outlier rejection are considering that the norm of the measurement should be within lower bound 8 $[m/s^2]$ and upper bound 11 $[m/s^2]$. Further, it is defined as

$$0.81 \cdot ||g_0|| < \frac{y_{acc}}{||y_{acc}||} < 1.12 \cdot ||g_0|| \qquad [m/s^2]$$

Regarding the update for magnetometer measurements, the criteria for outlier rejection are defined as follows

$$\frac{y_{mag}}{||y_{mag}||} < 1.1 \cdot ||m_0|| \qquad [\mu T]$$

where m_0 is the magnetic field vector in Linköping, presented previous in the documentation, and multiplied with 1.1 to introduce a safety margin or a secure zone, aiming to mitigate the impact of potential disturbances or inaccuracies in the measured data.

6 GAZE TRACKING

The G3 tracks the eyes and streams a 2D gaze point and a 3D gaze point. The 2D gaze point is associated with a point in the scene camera frame and can therefore easily be displayed on the video stream and associated with for example a detected face. The 3D gaze point is estimated by using the direction to which each eye is pointed towards. The 3D gaze data from the glasses comes in cartesian coordinates (x, y, z) given in a coordinate system relative to the user. It is then in the program converted into polar coordinates. The distance to the user r, the horizontal angle θ and the vertical angle ϕ are calculated according to

$$r = \sqrt{x^2 + y^2 + z^2} \tag{1}$$

$$\theta = \cos \frac{x}{r} \tag{2}$$

$$\phi = \tan \frac{x}{z} \tag{3}$$

The horizontal and vertical angles are well estimated. However, due the biology of the eye and the inaccuracy of the eye cameras, the depth estimation is not good at all and should not be trusted.

7 FACE TRACKING

Figure 4 presents and overview of the face tracking module. The face tracking module consist of two submodules, face detection and the actual face tracking.

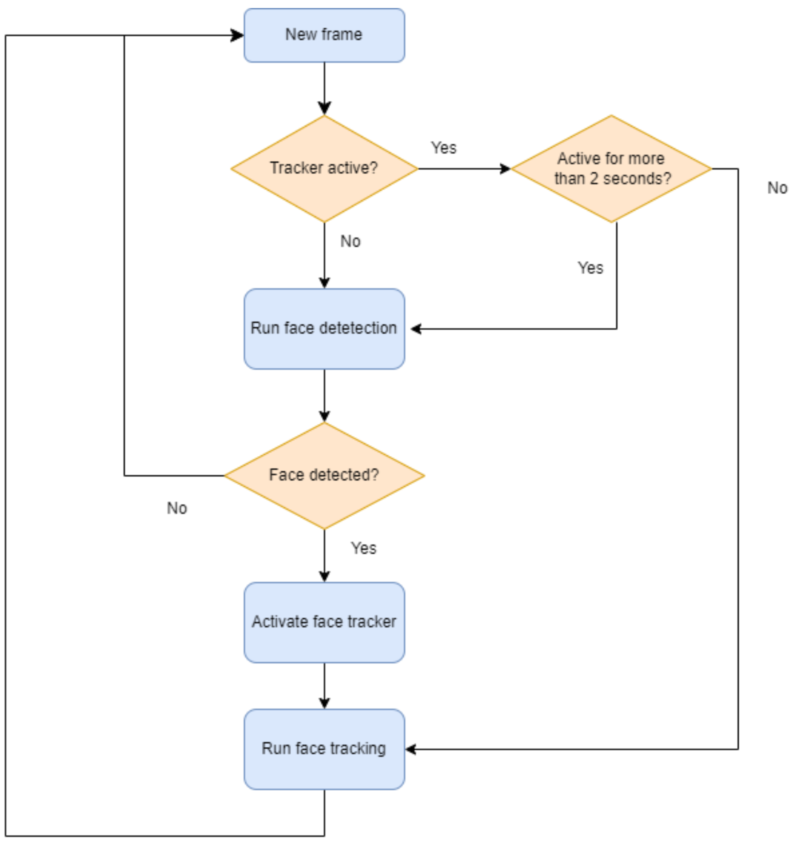

Figure 4: Overview of the face tracking algorithm.

7.1 Face Detection

For a face to be tracked it first needs to be detected. The method used for detecting faces is Haar cascades. The algorithm is already implemented with pre-trained weights in the open-cv library for Python. Detection of faces is run at the beginning of the video stream or video until a face is found. A successful detection of a face generates a bounding box in the video frame coordinates containing the face. It then enters the tracking phase. Since the tracker used in the implementation does not stop tracking a face even when the front of the face is not present in the video frame the detection phase is reactivated every two seconds.

7.2 Face Tracking

Once a face is detected the face tracking algorithm is run. The tracker that is used is a correlation tracker from dlib. The tracker is sufficiently fast to work on real time data for a short while. After some time the buffer of video frames becomes too big and the streaming stops. A con of this tracker is that it does not stop its tracking even when the face is no longer in frame. The face detection is therefore rerun every two seconds to make sure that there is a face present to track.

In the development of the face tracking module several different trackers were tested. Among those tested were trackers implemented in open-cv for Python. These were found to work good for a webcam data stream and recorded data but when used on the rtsp-stream from the glasses the trackers performed slow. The trackers successfully tracked the face but took too long to perform and so the queue of frames got too long and the video lagged circa 30 seconds behind and ultimately crashed entirely. To combat this, an attempt to skip frames was done. This did not make the processing much faster and made the accuracy of the tracking worse. Another attempt was to resize the frame and turn it to grayscale to lower the data to be processed. This change did not improve the performance significantly either.

To estimate the 3D position of a tracked face it is assumed that the bounding box keeps the same proportions relative to the face. Due to lens distortion in the scene camera a the width of a face increases with approximately one pixel per degree. The distance to the face can then be estimated with

$$z_{face} = \frac{p_{face} + |\phi^{t-1}|}{w} \tag{4}$$

where p_{face} is the pixel size of a face at a 1 m distance from the glasses, ϕ^{t-1} is the estimated horizontal angle in degrees at the previous timestamp and w is the current width of the bounding box. The estimated distance is then used get the full estimated position in Cartesian coordinates according to

$$x_{face} = -z_{face}\tan(\frac{FoV_x}{2})(2\frac{p_x}{f_x} - 1) \tag{5}$$

and

$$y_{face} = -z_{face}\tan(\frac{FoV_y}{2})(2\frac{p_y}{f_y} - 1) \tag{6}$$

where $FoV_x = 95$ and $FoV_y = 63$ are camera properties related to the field of view, p_x and p_y are the corresponding pixel for the center of the tracked face, and f_x and f_y are the width and height of the frame. Lastly, the conversion to polar coordinates is done according to Equation 1 - Equation 3.

8 DATA STREAMING

This section explains how the streaming of data from the different sensors to the programs are made and also how the programs communicate with each other.

8.1 Live processing

A GUI for G3 was developed in a previous year's project. The GUI could connect to the glasses, stream and display the data. The initial goal was to develop this GUI further with more functionality. When beginning the development of the GUI there was a lot of problems with stopping the streaming to the glasses in an appropriate way, also the streaming of data from the glasses was very much integrated with the GUI which was not desirable due to modularity and real-time issues. The existing GUI was therefore abandoned and a new UI was developed. The UI is run through the terminal and can be run with several different options. The UI is highly modular and can therefore easily be extended. The UI can run the orientation estimation, face tracking and gaze tracking. The UI can also be used for making a recording on the glasses. More information on how to run the program and the different run options can be found in the User Manual.

The UI is developed using the g3pylib library for Python [3]. g3pylib is a repository developed by Tobii to handle the low level connections though WebSockets which the glasses use to stream data. The library is under development and some functionality might not be available. The library is built on asyncio for asynchronous programming. In the asynchronous programming the tasks are defined as "async functions" and executed in an event loop. The UI makes sure that all incoming data is processed to avoid package loss. If the data is not processed fast enough the data is put into a queue. The data in the queue will then be processed in the order it came until the queue is empty. Streaming of IMU and Gaze data is done over a rudimentary stream that is subscribed by the asyncrounous functions. The scene camera frames are streamed over Real Time Streaming Protocol (RSTP). For more details about how the streams work and what data is available take a look at the documentation of g3pylib and also the developers guid available for Tobii Pro [3] [4].

To be able to interact with the program through the terminal in this asynchrounous environment, a python library called *aioconsole* is used. This enables asyncrounous input from the terminal.

8.1.1 *Known issues G3 live processing*

There is a number of issues concerning the streaming of data that is worth mentioning in its own section. Working with asynchronous functions can be intimidating and hard to both grasp and debug. One of the biggest problems during development was to correctly stop the streams from the glasses. If this is not done correctly, the battery will drain faster. The glasses also stopped working at the end of the project with not being able to stream data. This may or may not be because of bad handling when stopping the streams, the group is divided in the matter. For all sakes and purposes, try to stop the streams correctly at all times. There is some error exception in the existing code, but error can still happen. Learn more about this on the wiki page at the Oticon GitLab repository [5].

8.2 Offline processing

A UI for running the implemented functionality in recorded data was also developed. The offline UI loops through the data, simulating a live data-stream from the glasses. This approach of implementing allows for the same classes for, for example, face tracking to be run on both live data and recorded data with no alternations in the data processing codes.

The offline processing does allow for the data of the magnetometer to be simulated to come at the right time relative to the accelerometer and gyroscope data. In the data stream from the glasses the magnetometer data has a frequency of 10 Hz but comes in batches every second to to the recording unit. The data accelerometer and gyroscope data on the other hand comes at the right time. When replaying the data in the offline script some processing can be done to the data to simulate the magnetometer to come at the right time in relation to the accelerometer and gyroscope data. This allows for a better estimation of the orientation of the glasses, see section 5.

8.3 Data streaming between programs

The data stream and processing from the Tobii Pro G3 are made in Python with the g3pylib API from Tobii [3]. Hearing aid streaming and processing is made in Matlab. This causes a problem when combining the systems since there are different programs needing to communicate with each other for different purposes e.g controlling the beamformer with gaze information. This is solved with a program called Lab Streaming Layer (LSL) [6]. This is an easy to use solution for sending and receiving data from different parts of the network. This project used LSL to send the 2D gaze information from the Tobii Pro 3 glasses to the hearing aid during live streaming. The gaze is then used to determine the direction of the beamformer.

Another use case was also implemented, but not tested. The hearing aid estimates a position of a sound object and the face tracking algorithm used by the camera of the glasses does this also. These measurement could be fused by sending the DOA measurement to the Tobii Pro 3 glasses. An illustrative image of the communication can be seen in 5.

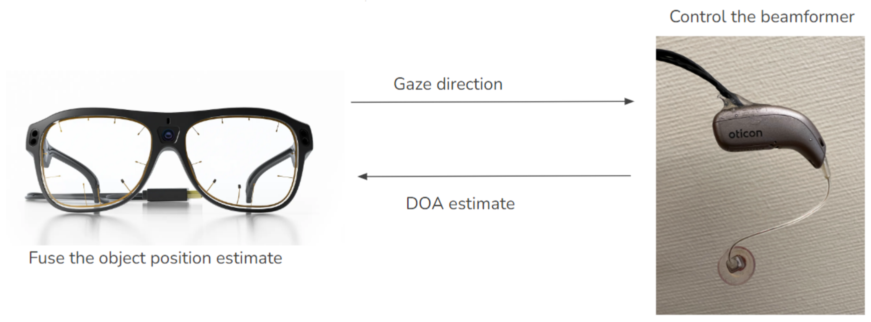

Figure 5: Data streaming between the systems

9 COMBINATION OF SYSTEMS

In this section it is described how the systems combines and works together.

9.1 Gaze controlled beamformer

As mentioned in 8 the use of LSL enabled communication between the programs running the Tobii Pro Glasses 3 and the hearing aids. By sending the gaze information as below.

$$gaze2d = [x, y] \tag{7}$$

where x is the normalized distance in the horizontal direction of the camera frame meaning $x = 0$ is the left edge of the camera frame and $x = 1$ is the right edge. The same goes for y but in the vertical frame.
The information is sent over LSL at the same frequency as the gaze information is received at the glasses, which is 100 Hz in our case. There is a 50 Hz Tobii Pro Glasses 3 version and this would not change anything, the gaze information would then be sent at 50 Hz instead as it is automatically done in the asynchronous receiving function for the gaze data.
The beamformer checks the LSL stream in every iteration of its main loop, which is not a contant frequency loop since it is using buffered audio data in a nested way. Anyhow the frequency of this audio loop is way higher than the LSL stream which means that no new information about the gaze is available in the majority of times that it checks. This is handled by making the LSL receiving function non-blocking and not updating the direction of the beamformer if no new value has been received by the glasses.

10 RESULTS

The most important result of the project are presented below.

10.1 Sound Source Tracking

The most important results for the Sound Source Tracking module are:

- Being able to recognize an active sound source of interest with automatic speech recognition (ASR) within 1 seconds.
- Estimate the direction of a moving sound source (a human voice).
- Estimate the distance to a moving sound source (a human voice).
- Being able to control outlier rejection.
- Estimate a 2D position to a sound source (a human voice) by combining the DOA and range measurements.
- Do the Sound Source Tracking live and on recorded data.

The tests for the range and DOA estimates can be seen in figure 6. The hardware used for the hearing aid devices are very sensitive and can often result in extremely high and noisy measurements. The problems that often occurs is that the connectors from the hearing aids are a bit loose and the cables connecting the battery to the hearing aids may deteriorate.

10.2 Noise Reduction Module

The most important results for the Noise Reduction Module are:

- Able to filter sound using a monaural beamformer where both microphones on a hearing aid are used to amplify speech in a given direction from the user.
- Able to filter sound using a binaural beamformer where one microphone on each hearing aid are used to amplify speech in a given direction from the user. See Table 2 for results.

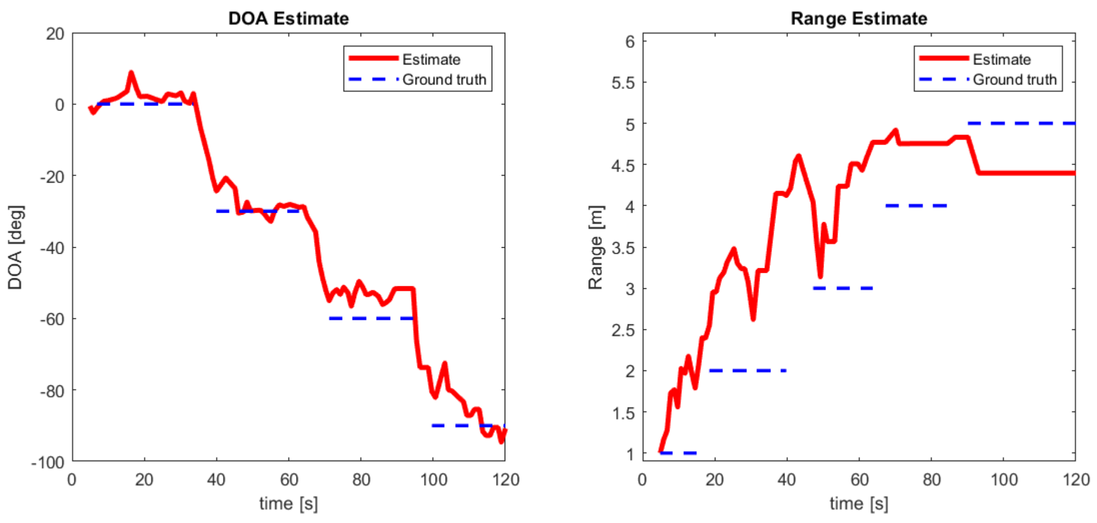

Figure 6: Tests for the range and DOA estimates.

- Being able to control the direction of the beamformer live through e.g. the gaze.

	$SNR_{seg,left}$	$SNR_{glob,left}$	$SNR_{seg,right}$	$SNR_{glob,right}$
Before	-2.3 dB	-1.3 dB	-0.2 dB	2.3 dB
After	0 dB	5.5 dB	0 dB	5.6 dB

Table 2: SNR test, binaural, segmented mean and global of interferer test for beamformer

10.3 Orientation Module

The most important results for the Orientation Module are:

- Able to estimate the orientation of the head with an error within 20%.
- Being able to estimate yaw, pitch, and roll angles given either live or saved data. Example of a sequence when all the orientation angles is rotated of a user's head is illustrated in Figure 7.

10.4 Gaze Tracking

The most important results for the gaze tracking are:

- Estimating the 3D-position of the gaze of the user.
- Associating the gaze with a point in the video frame.

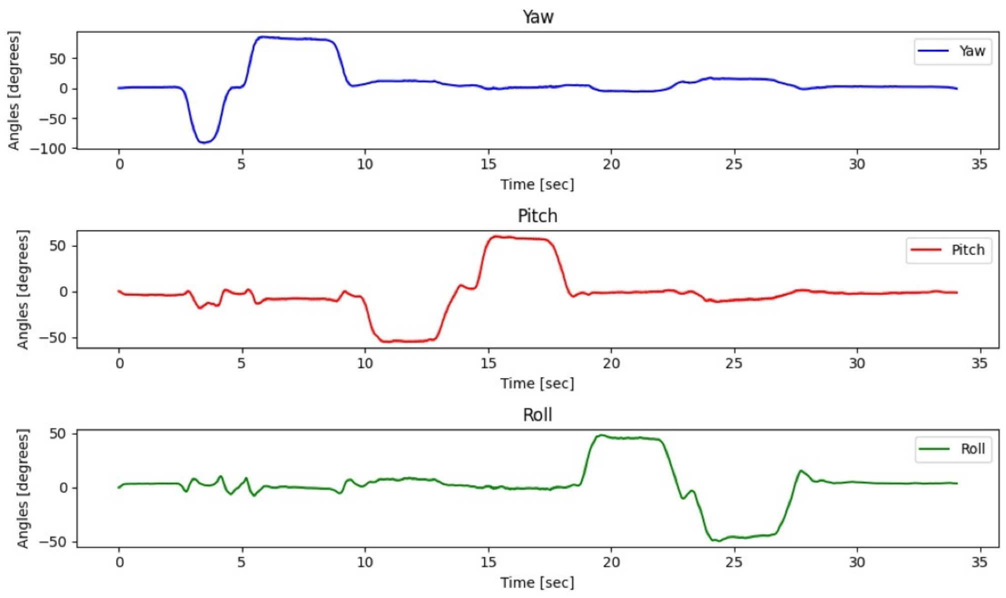

Figure 7: Three plots showing the orientation of the yaw, pitch and roll angles of a user's head during a sequence of 35 seconds.

10.5 Face Tracking

The most important results for the face tracking are:

- Detection and tracking of a face.
- 3D-positioning of a face within 30% of ground truth in the range 1-5 m.

10.6 Data streaming

The most important results for the data streaming are:

- Live datastreamer for the Tobii Pro 3 glasses.
- Offline datastreamer for the Tobii Pro 3 glasses.
- Communication between the HA and Glasses using LSL.

11 FUTURE IMPROVEMENTS

In order to further develop the project the following improvements are suggested.

11.1 Multi Object Tracking

The face tracking and the sound source tracking is only implemented for tracking a single target. To extend this to several faces and sound source would allow for example listening selection between the two faces.

11.2 SLAM

By combining the sound source tracker, face tracker and the orientation module, it is possible simultaneously localise and map (SLAM) the orientation of the users head, as well as map where the sound object/face relative to the user.

11.3 Automated/selective Beamformer

By combining e.g. the slam estimate of the sound object and eye tracker it is possible to make the beamformer automated to be fixed at a sound source instead of only the gaze direction.

11.4 Sensor models

Trying different sensor models for the sound source tracker that e.g. use more advanced head transfer functions would make it possible to get an estimate of the vertical angle.

11.5 GUI

Create a GUI that combines the control and analysis of data from both hearing aids and Tobii Pro G3. This could be done by for example streaming more information through LSL and receiving it at the GUI program. Some issues with this would be that LSL is not meant for streaming at such high frequencies as audio, and video stream over LSL can be slow.

REFERENCES

[1] T. AB, “Tobii pro glasses 3,” https://corporate.tobii.com/sv/media/mediabank, [Online; accessed September 19, 2023].

[2] Z.-H. T. S. M. I. J. J. Poul Hoang, Jan Mark de Haan, “Multichannel speech enhancement with own voice-based interfering speech suppression for hearing assistive devices,” 2022.

[3] “Tobii g3pylib api for streaming data,” https://github.com/tobiipro/g3pylib, [Online; 2023].

[4] “Tobii pro glasses 3 document downloads,” https://www.tobii.com/products/eye-trackers/wearables/tobii-pro-glasses-3#download, [Online; 2023].

[5] “Oticon gitlab,” https://gitlab.liu.se/tsrt10/2023/oticon, [Online; 2023].

[6] “Lab streaming layer,” https://labstreaminglayer.org/#/, [Online; accessed November 28, 2023].