Technical Report

Gabriel Anderberg
Karl Hilding
Gustaf Härold
Klara Kastenson
Tai Ta
Michael Yasemi

December 20, 2022

Version 1.0

Status

Reviewed	Gustaf Härold	December 19, 2022
Approved	Martin Skoglund	December 19, 2022

Project Identity

Group E-mail: gusha385@student.liu.se

Homepage: http://www.isy.liu.se/tsrt10/group

Orderer: Martin Skoglund, Linköpings universitet
Phone: +46 13 281890
E-mail: martin.skoglund@liu.se

Customer: Sergi Rotger Griful, Eriksholm Research Centre
Phone: -
E-mail: segr@eriksholm.com

Supervisor: Johanna Wilroth
Phone: +4670-894 48 49
E-mail: johanna.wilroth@liu.se

Course Responsible: Daniel Axehill
Phone: +46 13 28 40 42
E-mail: daniel.axehill@liu.se

Participants of the group

Name	**Responsible**	**E-mail**
Tai Ta	Responsible for the Documentation (DOC)	taita693@student.liu.se
Klara Kastensson	Responsible for the Design (DES)	klaka376@student.liu.se
Karl Hilding	Responsible for the Testing (TEST)	karhi203@student.liu.se
Michael Yasemi	Responsible for the Hardware (HW)	husya078@student.liu.se
Gabriel Anderberg	Responsible for the Software (SW)	gaban592@student.liu.se
Gustaf Härold	Project Leader (PL)	gusha385@student.liu.se

CONTENTS

Oticon
gusha385@student.liu.se

DOCUMENT HISTORY

Version	Date	Changes made	Sign	Reviewer
0.1	2022-12-16	First draft.	All	GA
1.0	2022-12-19	First version.	All	GA, TT, KH

Oticon
gusha385@student.liu.se

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 (2022) 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 will follow the project model LIPS.

1.1 Definition of terms

Table 1 shows the abbreviations used through the document.

Table 1: Definition of terms

Terms	Meaning
sim-env	The simulation environment is a combination of Gazebo with ROS, where instructions from ROS control the Gazebo environment.
GUI	Graphical User Interface, where the user can interact with the program.
SLAM	Simultaneous Localization And Mapping.
EKF	Extended Kalman Filter.
MHT	Multi-Hypothesis Tracking.
IMU	Inertial Measurement Unit.
ISY	The department of electrical engineering at Linköping University.
DP	Decision point.
LIPS	Project model, containing rules, instructions and templates to conduct a project.
User	Refers to the person using the Tobii Pro Glasses.
Speakers	Refers to people talking in the environment of the user.
G2	Tobii Pro Glasses 2.
G3	Tobii Pro Glasses 3.
HA	Hearing aid.
Gazebo	Open source software for simulating robot scenarios in 3D.
ROS	Robot Operating System. Open source software for robotic operating system.

2 SYSTEM OVERVIEW

This section shortly describes the systems' different subsystems and what has been done during the project. More details can be found in Sections 3 to 6. An overview of the system is shown in Figure 1. In the figure, blue boxes show different subsystems, green boxes existing modules, and red boxes show new modules.

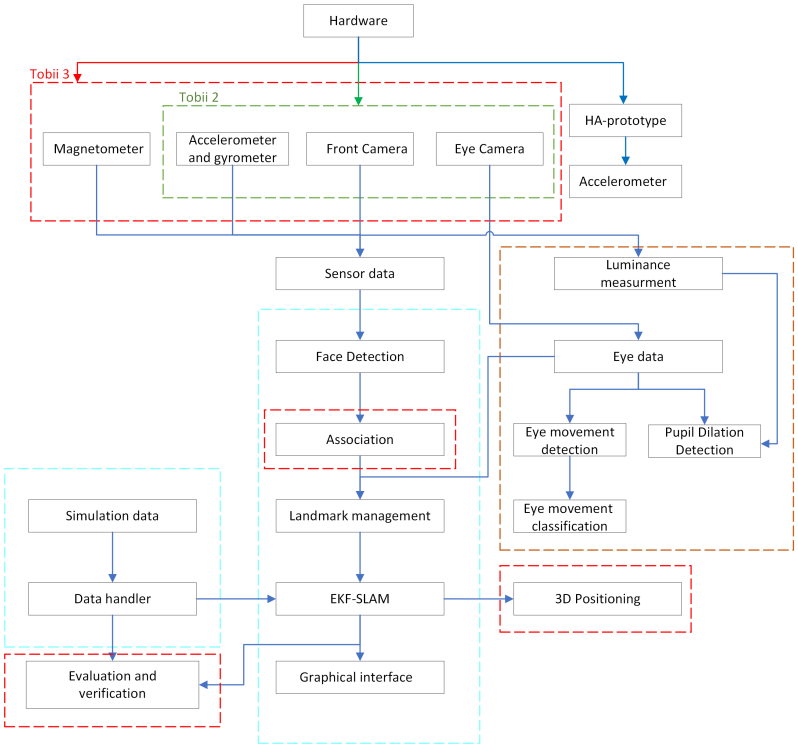


Figure 1: A schematic of the system

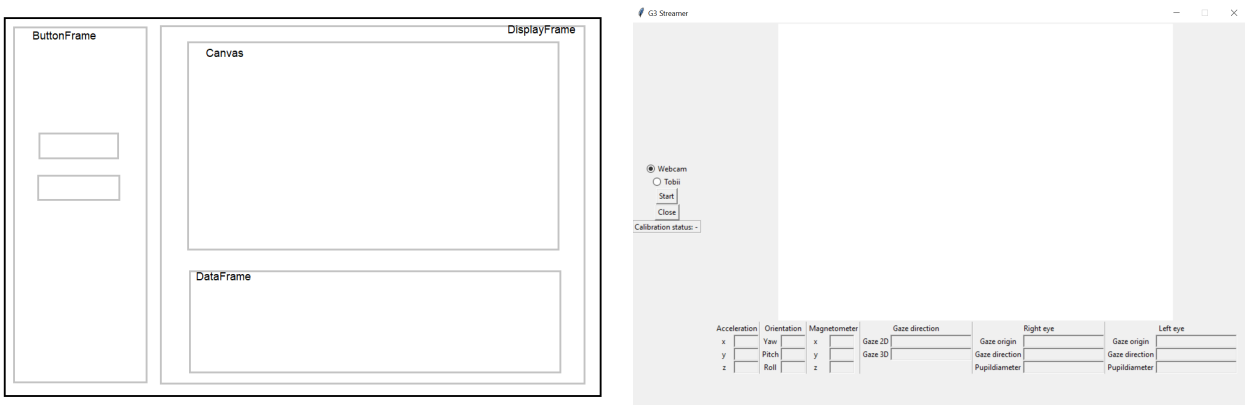
The hardware used in this project includes two versions of Tobii Pro Glasses. The old version (G2) has an IMU, a front-facing camera, and cameras used to record eye data. The new version (G3) has improved versions of these sensors as well as a magnetometer sensor. Development on making everything compatible with G3 has started and a GUI for connecting and streaming data from them has been created.

The target tracking module has been improved and a new Multi-Hypothesis Tracking module has been created. This new module improves on the previous method used to associate measurements from detected faces with confirmed targets.

This year's project has also introduced a number of new eye-tracking modules utilizing eye data from the glasses. These include a module to estimate ambient light in the environment, calculate average saccade frequency, smooth pursuit detection, and detect when the user is trying to listen to something. Eye tracking data is also used to augment the distance perception in the target tracking module.

The simulation environment has also been further developed. It has been extended with new simulation scenarios and the ability to set noise distributions for the accelerometer- and gyrometer-sensors. It can also simulate magnetic field data to simulate the G3:s new magnetometer-sensor.

3 G3 GUI



(a) New GUI structure

(b) The actual GUI

A new GUI for G3 has been constructed and the code has been restructured. The reason for this is that the new Tobii Pro Glasses 3 has a different API compared to Tobii Pro Glasses 2. It is different in syntax but similar semantically but major changes had to be done syntax-wise in python since send and receive is done via WebSockets compared to previous TCP/UDP sockets. Tobii has a repository for python that conveniently handles the low-level connections through WebSockets called g3pylib (v0.3.1). The library is a work in progress so certain functions are yet available for example data streaming over RTSP. This library is built on top of asyncio for asynchronous programming which has a special syntax since tasks have to be defined as "async functions" and executed inside its event loop. The difference in syntax is the reason for the need for a new GUI and a restructure in code. The main gist of how asyncio is used is that it makes sure that all incoming data is processed by the processing unit (a laptop in this case) so that package loss is avoided. Whenever the processing unit isn't able to process the data fast enough, the data is put inside a queue data structure. Each sample will then be processed in the order it came until the queue is empty. This is implemented for receiving IMU and Gaze data. Stream of the front camera on the other hand is done over RTSP. The resulting GUI is shown in the figure above. The code is written in an MVC code pattern which will allow future projects to be easy to maintain.

4 TRACKING AND LOCALIZATION

The tracking and localization problem concern the ability to construct a map of an unknown environment while keeping track of the position of the person wearing the glasses. This is done using the EKF-SLAM, yielding the linearized model shown in equation (1).

$$x_{k+1} = F_k x_k + G_k w_k, \qquad Cov(w_k) = Q_k \tag{1a}$$
$$m_{k+1} = m_k \tag{1b}$$
$$y_k = C_k^x x_k + C_k^m(c_k^{1:I_k}) m_k + e_k, \qquad Cov(e_k) = R_k \tag{1c}$$

where:

$$\begin{cases} x_k & = \text{states} \\ m_k & = \text{landmarks} \\ y_k & = \text{measurements} \\ w_k & = \text{process noise} \\ e_k & = \text{measurement noise} \\ c_k & = \text{association indices} \\ F_k G_k, C_k & = \text{state space matrices} \end{cases}$$

This years project has used the same EKF-SLAM algorithm as previous years in order to update the positions of the user landmarks. However, in order to improve the association between detected faces and existing landmarks a new Multi-Hypothesis Tracker algorithm has been implemented. This replaces the previous Global Nearest Neighbor association algorithm.

Multi-Hypothesis Tracking (MHT) is a method for multi-target tracking that conceptually can be explained as generating all possible association hypotheses and comparing them over time in order to find the most likely hypothesis. Since generating all possible hypotheses is computationally demanding and therefore inefficient, a Hypothesis Oriented MHT (HO-MHT) has been implemented. In a HO-MHT, only the best possible hypotheses are generated using Murtys Algorithm and the ones that would be deleted anyway are ignored [1].

The MHT-algorithm uses relative 3D coordinates between the user and speakers as measurements. These coordinates are estimated from the face detection modules. The first step of the MHT-algorithm is to perform gating on all targets and measurements. Gating is a process where decisions regarding which measurements could reasonably be considered belonging to each target. This was done using elliptical gating, where a measurement is considered valid if it fulfills the following condition.

$$(y_k - \hat{y}_{k|k-1})^T S_k^{-1} (y_k - \hat{y}_{k|k-1}) \leq \gamma_G \tag{2}$$

where:

$$\begin{cases} y_k & = \text{obtained measurement} \\ \hat{y}_{k|k-1} & = \text{estimated measurement} \\ S_k & = \text{innovation covariance} \\ \gamma_G & = \text{gating threshold} \end{cases}$$

The gating threshold γ_G is calculated using the chi-square inverse cumulative distribution function for the desired gating probability. The gating process results in a validation matrix consisting of ones and zeros, where a one means that a measurement could belong to a target. This validation matrix is then used to create an association matrix consisting of the probability that a measurement belongs to a target. These probabilities are used in Murtys Algorithm to generate up to N_h of the most probable association hypotheses and their probabilities are calculated.

The MHT keeps track of the N_h best hypotheses at all times, resulting in N_h different possible associations. This means that the above process must be performed N_h times, yielding up to N_h^2 different hypotheses. Since we only need to have N_h at all times, the best ones are chosen. Then, new positions for each landmark are estimated according to all chosen hypotheses, and the one with the highest probability is used as landmark positions in the EKF-SLAM. A higher number of hypotheses N_h will lead to more accurate results and associations. It will however also make the algorithm more complex and slower. It is therefore important to find a value that gives satisfactory results in a reasonable amount of time.

It should be noted that the algorithm does not perform any form of target clustering, i.e. separating targets with common possible measurements to reduce complexity, between the gating and association step. This was omitted due to time constraints while developing the algorithm, but due to the low amount of faces appearing in a single measurement in performed tests, this was deemed acceptable. A clustering step should however be added to increase the algorithm's speed in complex cases.

5 EYE TRACKING

The new eye-tracking modules are presented below.

5.1 Listening detection

A person exerting effort to focus on an auditory source can be interpreted as if the person has difficulty hearing and there is a need for amplifying the voice. The outcome of this effort is that the pupil diameter will increase [2]. Here this fact has been used to indicate whether the user is exerting effort or not. In doing so it is necessary to determine when the change of the pupil diameter is not the result of exerting effort to focus. In real life, there are many factors affecting pupil diameter but two important factors for pupil diameter change are, one the change in the environment's light and second, the different pupil dilation in the dark and light environment [2] [3].

When the light level in the environment is decreasing the pupil diameter will increase in order to let more light come into the eyes. In order to estimate the brightness of the environment, the RGB values of the video frames by the front camera have been used. According to the international standard recommendation ITU-R BT.709, the luminance is calculated as a weighted sum of R, G, and B [4] illustrated in the equation (3).

$$Luminance = 0.2126 * R + 0.7152 * G + 0.0722 * B \tag{3}$$

Having estimated the environment brightness, it will be easier to tell if the change in the pupil diameter is a result of the change in the environment's light or if it is a consequence of the user having difficulty hearing and paying more attention to the voice source.

Figure 3 shows the decision-making for the listening detection. The listening will be investigated only if the eye is in the fixation or smooth pursuit mode.

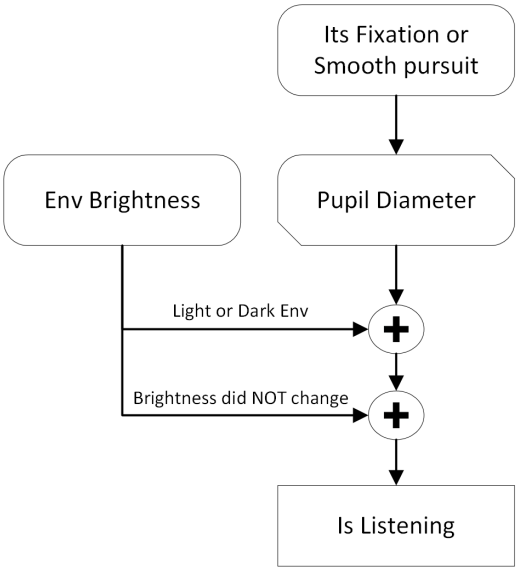


Figure 3: A schematic of Listening Detection

5.2 Eye movement classification

The eye movement classification module uses eye velocity to both determine if a motion is occurring and to classify the type of movement. The possible motions are saccades and smooth pursuit motions. If no motion is detected the eye is assumed to be in a fixation. Two methods/algorithms are used to perform the classification. One method is threshold based and uses solely eye velocity thresholds to detect and determine movements.

The second method utilizes a probabilistic interpretation of the problem where the movement ratio is recorded during a past number of data points [5]. The number of past data points is set to be able to include a whole saccade duration since it is assumed a saccade is preceded and followed by a fixation. The likelihood of a smooth pursuit will be at its highest if movement is detected during all past measurements.

5.3 Average saccade frequency

The average saccade module calculates both the mean saccade frequency over time and the current saccade frequency. The current saccade frequency is the freequency of initiated saccades during a past time window. The idea is to compare the two measures to make conclusions about the user's behavior. For example, if the frequency drops far below the mean, it might indicate that the user is trying to listen or focus.

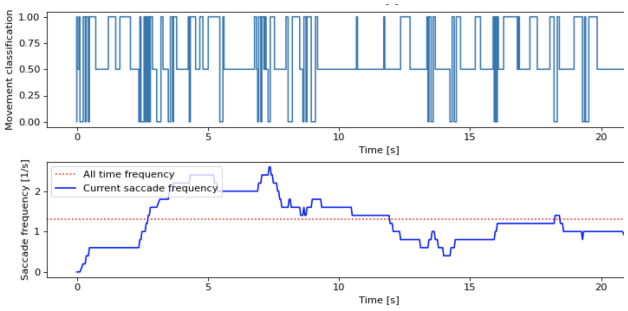
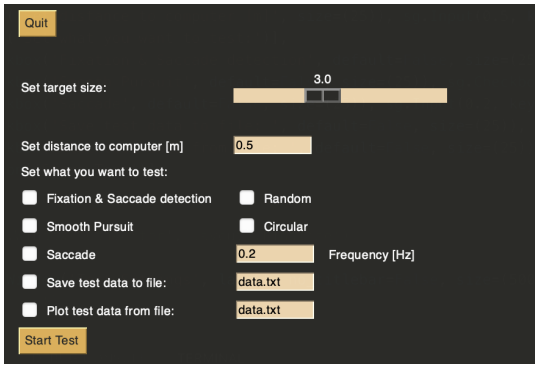


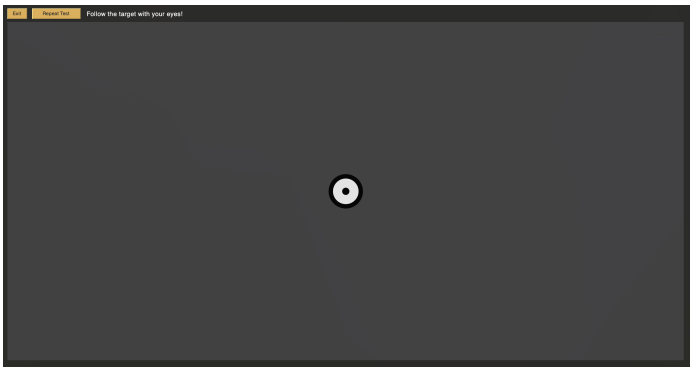
Figure 4: Display of the average saccade frequency module.

5.4 Test application

The testing application was developed to easily conduct tests. It was intended to be used especially to test the average saccade frequency module since it requires controlled circumstances. The user is to follow the target with the eyes. The target moves across the screen in a specific manner and induces fixations, saccades and smooth pursuit motions. The result is then compared to the test data. The application is proved useful when repeatability of tests is necessary.



(a) GUI of the test application.



(b) Test environment.

6 SIMULATION ENVIRONMENT

In this section, the project simulation environment is presented.

6.1 Introduction

Using a simulation environment has many advantages during the development phase. It has the ability to test and analyze systems in a controlled and safe environment. Simulation is also cost- and time-efficient since it can be independent of any hardware during software development. Creating a robust environment testing can be done on more complex scenarios and it has the ability to choose exactly how much noise the signals should have if any.

6.2 Gazebo

Gazebo is a simulation tool that can be used to perform dynamic simulations or generate sensor data, among other things. In Gazebo, a world is a set of models and global parameters such as physical properties. Models are what populate the simulation and can be simple geometric objects such as a cube, or a complex model consisting of many submodels, such as an advanced robot. The accessible models as of now for the project can be found in the left toolbar in the Gazebo environment. The *User* model consists of a body and head where the head is linked to sensors replacing the Tobii glasses.

World plugins let the user control the physical properties of the world, such as ambient lighting or changes to the physics engine. Model plugins allow for the control of the models. Such as giving cube a certain velocity or applying a torque on a link. There are also specific sensor plugins, used to simulate sensors and sensor output. Combining Gazebo with ROS2, where instructions from ROS2 controls the Gazebo environment, creates a powerful simulation tool.

6.3 ROS

The Robot Operating System (ROS) is used as a framework for simulation. It is used for integrating and synchronizing different subsystems in the simulation environment. Together with the simulation software Gazebo it is used to control the user and speakers within the simulation and retrieve data from the modeled sensors. ROS is set up as different packages, each developed to do a certain task. These packages can then be combined to achieve a more complex function. Each package requires files that define requirements, such as other packages it uses, and which executables or scripts are supposed to be included in that particular package. A package is defined in either python or as there is a lot of documentation of this language available for ROS2.

6.4 Launch files

Launch files are used to execute multiple nodes simultaneously. The launch files are different bash scripts to ease the process of starting the simulation. The launch script *launch_2022* was created similarly to the launch script from the previous year. The launch script can be modified to simulate different scenarios and worlds.

6.5 Magnetometer sensor

To be able to investigate further the impact a magnetometer sensor would have on the algorithms used in the project, an attempt to implement a magnetometer sensor in the simulation environment was made. It was possible to simulate magnetometer data in Gazebo. However, due to time constraints, it was impossible to publish the data from Gazebo to ROS, in order to be able to save the data.

The IMU data in the simulation environment is saved via the package Gazebo_ros_pkgs [6]. This package does however not have the functionality to publish magnetometer data from Gazebo to ROS.
An attempt was also made to manually write a plugin to publish magnetometer data from Gazebo to ROS. The attempt mostly followed the steps in this [7] guide. There is also documentation saved in the Oticon Gitlab repository in the branch "magnetic", which can be investigated to see what steps were made. This approach was however dismissed, due to time constraints, but is not deemed impossible. It should however be possible and not particularly difficult to write a plugin that publishes the magnetometer sensor data from Gazebo to ROS.

The last attempt made to publish the sensor data from Gazebo to ROS was using the ROS-package hector_gazebo [8]. The package claims to be able to publish the wanted data but does not provide any information about how it is installed or used. To conclude, this is also an approach that probably would be possible, if one were to investigate thoroughly how to install and use ROS packages.

6.6 Sensor noise

To be able to investigate the impact has on noise in different use-case scenarios, the program set_noise.py was developed. The full documentation of how to use the program is available in the Oticon Gitlab Wiki [9]. Prior to this year, the only noise that was implemented was white (Gaussian) noise with zero mean and a standard deviation of 0.01, and it was the same for all sensors. By using the program set_noise.py, it is now possible to set noise that is Gaussian-, Exponential-, Laplace-, Bi- or Multimodal-distributed, with arbitrary mean and standard deviation for the accelerometer- and gyrometer-sensor. It is also possible to set individual mean and standard-deviation properties of the x-, y- and z-components of each sensor.

In figure 6 a flowchart of the program is shown. The parser can take the arguments "–world", "–quantity", "–value", "–sensor", "–noise", "–weight" and "–offset", see the wiki for full documentation about how to use them properly [9]. Furthermore, the function **change_line_in_file()** changes the necessary lines in the .world the user has provided. In the case that the noise is Gaussian, the properties of the noise are appended to the given .world file. Otherwise, all sensor noise will be removed from the Gazebo simulation, and instead added to the sensor data when it is saved. The function **write_to_file()** simply writes the noise properties as a python dictionary to the file noise.py in the user's python directory, in order to be able to use the information when saving the sensor data.

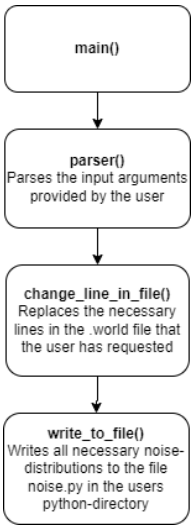


Figure 6: Flowchart of the set_noise.py program.

The most important improvements that can be made to the program that is known at the time of writing are the following. The program should be extended so that it is also possible to set and add noise to the gaze vector in the simulation environment. If it would be made possible to save the magnetometer data, then it should also be possible to add noise to that sensor. The implementation of being able to add noise to the gaze vector and magnetometer data should not be difficult, since what has to be done is almost identical to what the program does when adding noise to the IMU sensors.

Furthermore, the program as of now only consists of 4 functions, including the **main()** function. To be able to easier test and develop new features, as well as increase the maintainability of the program, it would probably benefit a lot from dividing the program into more separate functions, each consisting of less code.

The **parser()** function consists of a lot of code that does not parse the user's input but instead tests if the input is valid. These tests would benefit if they were to be broken out into separate functions, and also extended, as they are not comprehensive.

6.7 Simulation scenarios

To maintain reproducibility in testing when using the simulation environment the simulated scenarios were pre-defined and not altered during the simulation.

The scenarios are defined in the file dynamic_path_controller as different cases that can be switched in between simulations. The controller subscribes to at least one of the /demo/odom_user topics since the odometry message also contains a simulation clock, the positions are not used from the odometry message. Depending on the time, the controller will post different Twist messages to the topics such as user body, user head, and speakers. Planar_move is subscribed to these topics and they control the movement of the user body, the user head, speaker1, speaker2 och speaker3 respectively. A Twist message includes linear velocities in x, y, and z-directions and angular velocities around the x, y, and z-directions. The velocities from the Twist messages are applied to the model's local coordinate system. So positive movement in the x-direction means that the model heads the way it is facing. The planar move node also publishes positional data of several objects. A moving scenario must thus be described as the velocities and acceleration in every direction at any given time. The movement can beneficially be described in the modulus.

gusha385@student.liu.se

7 RESULTS

Below, the most important results of the project are presented.

7.1 Multi-Hypothesis Tracking

Due to time constraints, no new test data was able to be recorded in order to test the new MHT algorithm. It was however used on data from previous years, making it possible to judge its performance. The MHT was able to keep track of and associate measurements from three moving targets over time with good accuracy. An example of a validation test is shown in figure 7. The new algorithm had the same accuracy as the old method. This is expected as the MHT only deals with associating measurements to existing or new targets, meaning that the algorithm for position estimation (EKF-SLAM) was not updated this year. Due to not being able to record new test data more complex tests where the two association methods could be compared were not possible. It is however highly probable that the new method would be better at keeping track of a larger number of people moving in more complex patterns.

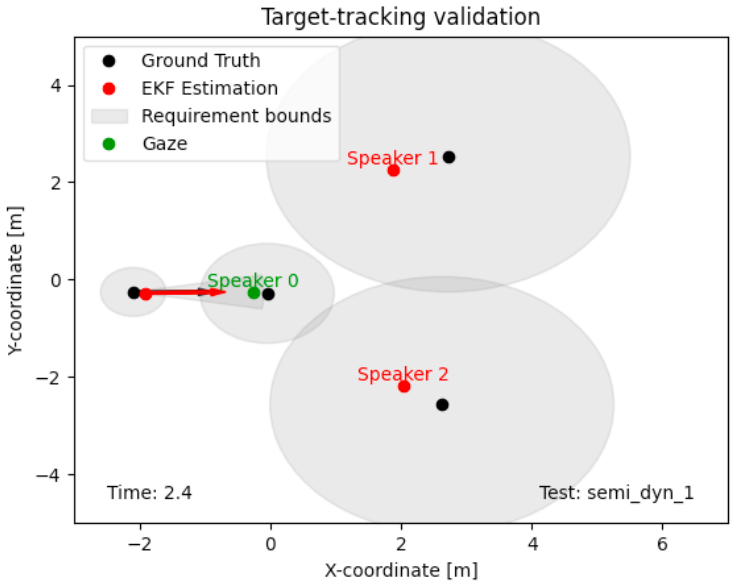


Figure 7: A frame from the validation of the target tracking algorithms

7.2 Simulation Environment

The improvements and new features that have been made to the simulation environment are presented in this section.

7.2.1 *Magnetometer sensor*

A magnetometer sensor is implemented in the simulation environment. It is thus now possible to study the change in the magnetic field in different simulation scenarios. It is however not possible to save the magnetometer data.

7.2.2 *set_noise.py*

The program set_noise.py was implemented to be able to modify the noise of each sensor in the simulation environment. It is now possible to add noise that is Gaussian-, Exponential-, Laplace-, Bimodal- or Multimodal-distributed. The mean and variance of each sensor component can also be individually modified. See figure 8 for an example.

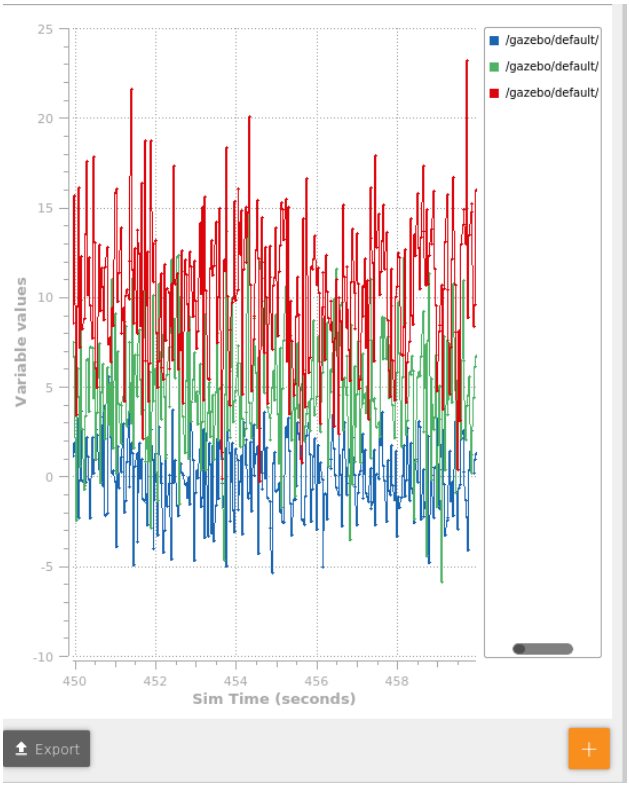


Figure 8: Illustration of noise impact on a sensor in Gazebo.
Blue is the x-component, green is the y-component and red is the z-component.

7.2.3 *New simulation scenarios*

Two new simulation scenarios were implemented in the dynamic_path_controller file that can be launched in the new world as well as the *.world* files created in previous years. One scenario is with only user body movement and the other scenario is with a fixed body but the head is rotating around the z-axis.

7.2.4 *Environment light measurement*

The environment brightness will be measured using the equation (3). This is to estimate if the environment is dark or light and to detect if the environment light has changed. the figure 9 shows one example.

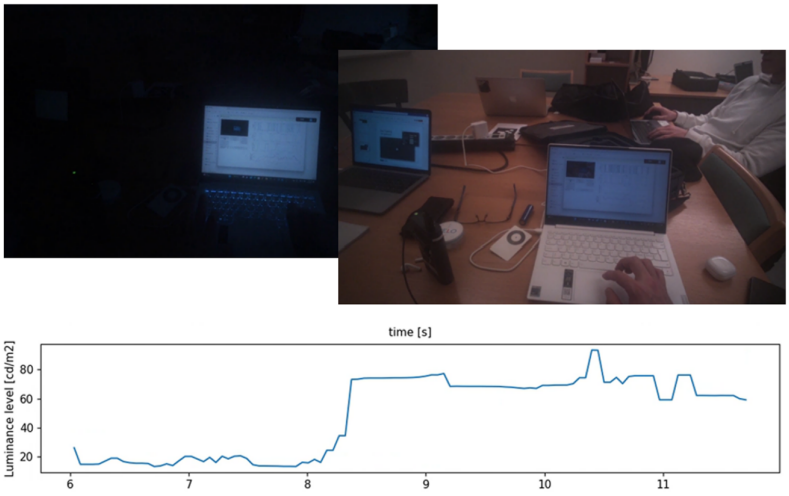


Figure 9: The Dark and light Environment and the result from the environment light measurement.

7.2.5 *General improvements*

The script startup_simulator.bash has been further developed so that it now has program flags so that the user can choose which simulation they want to run with one command. There is also a new script added to the project called *installscript.bash* located in the folder install. This script makes it possible to install all dependencies needed for the software with only one command.

8 FUTURE IMPROVEMENTS

In order to continue the development of the system, the following improvements are suggested.

8.1 G3 GUI

Additional functions that existed for G2 could now be added to the code. The GUI will provide an easy-to-use platform to control and display data. Orientation estimation with a magnetometer could be added to this platform.

8.2 Tracking and Localization

The following things could be implemented and improved upon regarding target tracking and localization.

8.2.1 *Perform more tests on the MHT*

As stated in Section 7.1, more complex tests should be performed in order to validate the new MHT algorithm. These tests could consist of a larger number of people in the frame or having people move in irregular patterns close to each other.

8.2.2 *Add clustering to MHT*

As stated in Section 4, the MHT algorithm does not utilize any clustering that could reduce computation time. This should be added in order to improve real-time performance in the tracking algorithm.

8.2.3 *Combine the EKF-SLAM and Target Tracking modules*

The current system performs the target tracking and EKF-SLAM as two largely independent systems with different states describing the position of the faces. Combining these two modules into one would reduce the algorithm complexity and reduce the chance of errors.

8.3 Eye tracking

The gaze angles for each individual eye should have the possibility to be represented in a global coordinate system instead of a body fixed. This will remove the problem of movements being detected when the orientation of the user is changing, without changing the point of gaze.

8.4 Simulation Environment

The simulation data should be compatible with the EKF-algorithm to be able to verify the simulation environment and the simulated sensors. In the save_data folder, all data from the simulation is saved as .json files but must be transformed to fit and replica the data output from the glasses. A new script must be done to accomplish this.
The magnetometer data as of now is not saved when launching the simulation, this must be fixed if the simulation environment should be compatible with the G3 glasses.

REFERENCES

[1] G. Hendeby and R. Karlsson, "Lecture notes on multi target tracking: multi-hypothesis tracking," April 2019. [Online]. Available: https://www.control.isy.liu.se/student/graduate/targettracking/file/le5.pdf

[2] P. Książek, A. Zekveld, D. Wendt, L. Fiedler, T. Lunner, and S. Kramer, "Effect of speech-to-noise ratio and luminance on a range of current and potential pupil response measures to assess listening effort," *Trends in Hearing*, 2021.

[3] B. Pfleging, D. K. Fekety, A. Schmidt, and A. L. Kun, "A model relating pupil diameter to mental workload and lighting conditions," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 5776–5788. [Online]. Available: https://doi.org/10.1145/2858036.2858117

[4] M. Benavides, J. Mailier, A.-L. Hantson, G. Muñoz, A. Vargas, J. Van Impe, and A. Vande Wouwer, "Design and test of a low-cost rgb sensor for online measurement of microalgae concentration within a photo-bioreactor," *Sensors*, vol. 15, no. 3, pp. 4766–4780, 2015. [Online]. Available: https://www.mdpi.com/1424-8220/15/3/4766

[5] E. K. Thiago Santini, Wolfgang Fuhl and T. Kübler, "Bayesian identification of saccades, smooth pursuits, and fixations," Mars 2016. [Online]. Available: https://www.hci.uni-tuebingen.de/assets/pdf/publications/TWTE032016.pdf

[6] gazebosim. (2022) Gazebo ros pkgs. [Online]. Available: https://github.com/ros-simulation/gazebo_ros_pkgs

[7] O. S. R. Foundation. (2022) Intermediate: Control plugin. [Online]. Available: http://classic.gazebosim.org/tutorials?tut=guided_i5&cat=

[8] (2022) Hector gazebo. [Online]. Available: https://github.com/tu-darmstadt-ros-pkg/hector_gazebo

[9] Oticon. (2022) Oticon gitlab wiki. [Online]. Available: https://gitlab.liu.se/tsrt10/2022/oticon/-/wikis/Oticon-Wiki