Project Report

Creating a Gaze Vector-Labeled Image Dataset Using G3 Glasses

Authors: Gal Ben Ami & Daniel Boianju

Institution: Computer Science department in Bar Ilan University

Supervisor/Mentor: Dr. Reuth Mirsky

Submission Date: <to be filled>

Project Code available here

Table of Contents

Acknowledgments

We would like to express our deepest gratitude to **Reuth Mirsky**, our instructor, for her guidance and support throughout the project. Her technical expertise, insightful feedback, and constant encouragement were invaluable in steering us in the right direction.

We also extend our thanks to **DLR** and **Carlos** for their assistance in making this project. Their technical expertise was used during the creation of our dataset. Their contributions were vital in overcoming key technical challenges, and we are grateful for their collaboration.

Finally, we would also like to thank all the people that were kind enough to participate and be a part of the dataset that we have made.

Abstract

The goal of our project is to establish a methodology for creating a dataset designed to improve the accuracy of gaze detection algorithms. Accurate gaze detection is crucial in numerous applications, ranging from human-computer interaction to cognitive and psychological research, where understanding where a person is looking can provide deep insights into attention, intention, and decision-making processes.

Context and motivation

Developing accurate gaze detection algorithms presents numerous challenges.
Two key challenges that our project seeks to address are:

1. Distinguishing between head pose and actual direction of pupil gaze. This issue, commonly referred to as Head-Gaze Correlation Overfitting*, happens when an algorithm incorrectly infers the gaze direction based on head orientation rather than the precise eye focus. This can lead to errors in scenarios where the head is turned but the eyes are looking elsewhere.
2. Detecting the gaze vector in mobile or dynamic environments. While stationary platforms benefit from stable setups, gaze detection on mobile or dynamic platforms, for instance, cameras located on moving robots, requires advanced software precision and calibration to maintain accuracy in variable conditions.

To tackle these challenges, our project uses the Tobii Pro Glasses 3 (G3), an advanced eye-tracking device designed to capture pupil gaze data from the perspective of the glasses. By transforming this gaze vector into the scene camera's point of view, we generate a dataset that accurately reflects the true gaze direction, while being less dependent on head orientation, which is primarily used for rotational calibration. Additionally, the system's robust software and calibration processes ensure that the gaze data remains precise even in dynamic environments, addressing both head-gaze overfitting and the challenges of mobile gaze detection.

This architecture is not without its challenges in synchronizing two hardware devices. The Kinect camera and the glasses must be temporally aligned, making it particularly difficult to filter the data and output aligned Kinect data with G3 glasses data. Overcoming these obstacles was essential to ensure reliable data output.

* This term is also referenced and described in "*Monocular Free-head 3D Gaze Tracking with Deep Learning and Geometry Constraints" by Haoping Deng and Wangjiang Zhu (2017)*: https://openaccess.thecvf.com/content_ICCV_2017/papers/Zhu_Monocular_Free-Head_3D_ICCV_2017_paper.pdf

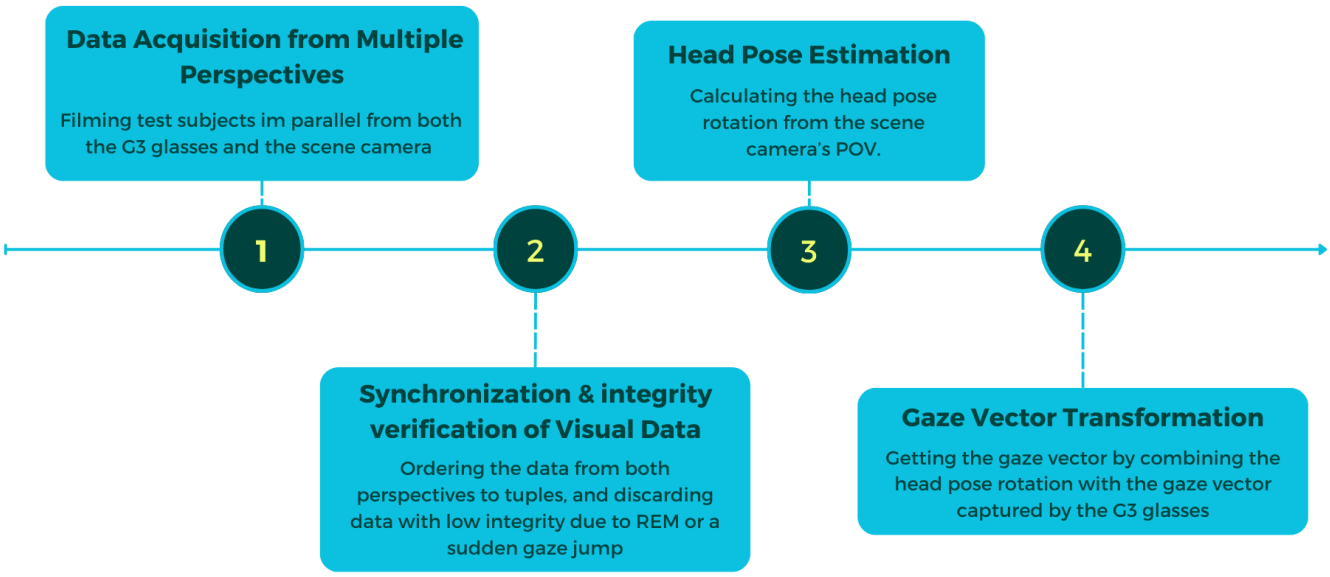
Methodology

This section describes the methodology of the **Dataset Creation Process**.
The process is briefly described in the figure in the lower part of this page.

The dataset is composed of high-resolution RGB images captured using the Azure Kinect camera, and it is generated through a four-stage process:

1. **Data Acquisition from Multiple Perspectives:**
   The first stage involves the simultaneous recording of the subject from two distinct perspectives: the scene camera, represented by the Azure Kinect, and the Tobii Pro G3 glasses, which capture the gaze vector. The combination of these two data sources is crucial for accurately determining the subject's gaze direction relative to the surrounding environment. To facilitate precise tracking, we employed April Tags (similar to QR codes), which serve as reference points for determining the rotation and location of the head within each image frame.

2. **Synchronization & integrity verification of Visual Data:**
   The second stage focuses on synchronizing the captured images from both the Kinect and G3 glasses. This synchronization is achieved through our **Frame Pre-Processor Module**, which aligns the timestamps of each image with the respective start timestamps of the recording devices. This step is vital for ensuring that the data from both sources is temporally aligned, enabling accurate pairing of gaze vectors with corresponding scene images.



3. **Head Pose Estimation:**
   Once the visual data is synchronized, the third stage involves calculating the head pose rotation from the scene camera's point of view. This step utilizes the April Tags embedded in the recorded images to accurately estimate the subject's head orientation relative to the scene. For this stage, we collaborated with DLR, who provided their expertise in extracting and analyzing the rotational data from the April Tags.

4. **Gaze Vector Transformation:**
   The final stage involves combining the calculated head pose rotation with the gaze vector captured by the G3 glasses. By transforming the gaze vector from the glasses' point of view to the scene camera's perspective, we obtain the final labeled dataset. Each RGB image is now associated with a gaze vector that reflects where the subject was looking within the scene.

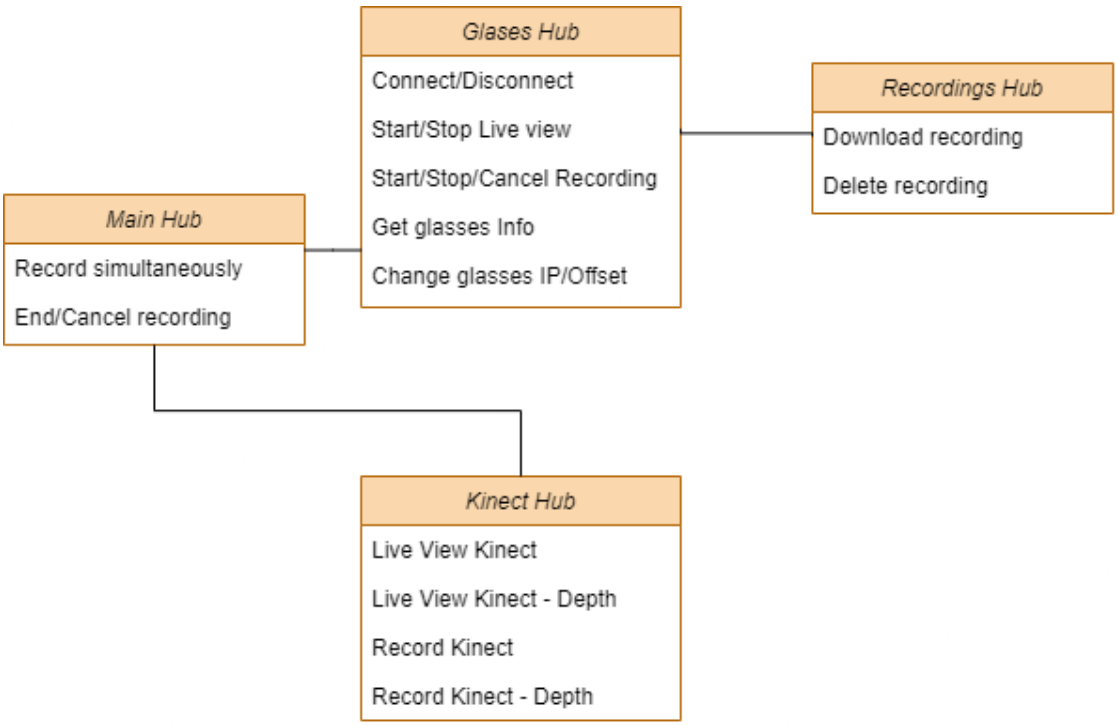
This methodological framework ensures that the dataset is both accurate and reliable, providing a robust foundation for future research in gaze detection. The following sections will provide a detailed breakdown of each step, including the technical processes and algorithms involved.

User Guide

The user guide for usage of the program is explicitly described in the github of the project.

System Architecture

To streamline the process and ensure that this methodology can be easily replicated in future studies, we developed a specialized program to manage the entire workflow of recording and synchronizing data between the Tobii Pro G3 glasses and the Azure Kinect camera. This program encapsulates the methodology for creating our dataset, ensuring consistency and accuracy across all stages of data collection and processing. Our system is implemented in Python and is composed of three connected modules: the **Main Hub**, the **Glasses Hub**, and the **Kinect Hub** (shown in the figure below). These modules work in concert to establish seamless communication with both the Tobii Pro G3 glasses and the Azure Kinect camera, enabling synchronized data recording from both devices.



The figure above shows the functionality of each hub and their high level connections.

Main Hub

The **Main Hub** serves as the central control interface of the system and is the first window that users interact with upon launching the program. It is responsible for orchestrating the operations of the other two hubs, the Glasses Hub and the Kinect Hub. The Main Hub allows for simultaneous recording from both the Kinect and the G3 glasses, based on the parameters predefined in each respective hub. A key feature of the Main Hub is its ability to prevent conflicting operations: if either the Glasses Hub or the Kinect Hub has already initiated a recording session, the Main Hub is restricted from overriding this action. Conversely, if the Main Hub initiates the recording, the individual hubs retain the capability to independently terminate their recording sessions as necessary.

Kinect Hub

The **Kinect Hub** is dedicated to managing the Azure Kinect camera. It provides users with real-time access to the camera's live feed, which can be viewed both as a standard RGB image and as an image overlaid with depth data. This hub not only facilitates simultaneous recording with the G3 glasses when coordinated through the Main Hub, but also operates autonomously, allowing the Kinect to record independently if required. The real-time display and control capabilities make the Kinect Hub a powerful tool for monitoring during data collection.

Glasses Hub

The **Glasses Hub** is tailored specifically for interacting with the Tobii Pro G3 glasses. It enables real-time visualization of the scene captured by the glasses, including the precise point of gaze of the wearer. This hub also manages various technical aspects of the glasses, such as network configurations, including the assignment and management of the device's IPv4 address within the network. Like the Kinect Hub, the Glasses Hub can operate independently, allowing recordings to be made solely through the glasses without requiring Kinect synchronization. This flexibility is crucial for scenarios where gaze data is the primary focus. Additionally, the Glasses Hub ensures that critical processes, such as stopping a recording session, are managed appropriately to prevent data loss or device overuse.

Recording Hub

The **Recording Hub** manages the recordings stored on the G3 glasses' SD card and is the glasses hub's sub-module. No processing is done in it, and a user will use it only to download and view or to delete recordings in the G3 's storage.

Handling Asynchronous Device Integration

A significant challenge that our system helps deal with is synching the recordings from both devices. This proves to be a challenging task because the Kinect camera operates synchronously, while the G3 glasses communicate asynchronously.

To deal with this challenge, several architectural decisions were made. Each hub operates on a separate thread, with requests to the glasses handled in distinct threads to avoid blocking the user interface (UI) and to enable parallel actions across different hubs. Additionally, when recording with both the glasses and the Kinect, each device stores its recordings alongside a timestamp indicating the start of the recording session in a shared folder.
These timestamps are later used to synchronize and align the data from the glasses and Kinect, resulting in aligned Kinect data with G3 glasses data.

To ensure efficiency and avoid redundancy, each hub within the system follows a singleton design pattern, ensuring that only one instance of each hub exists at any time. This is particularly important for the Glasses Hub, where stopping recordings correctly is essential to prevent the glasses from continuing to record unnecessarily, which could lead to battery drain or excessive storage consumption on the SD card.

The system is implemented in Python, utilizing the **PyQt5** library for the graphical user interface (GUI) and handling asynchronous operations. The use of PyQt5 allowed us to design a responsive and user-friendly interface, despite the challenges described earlier.

Overall, this system architecture provides a reliable framework for dealing with this challenge. Once the G3 glasses and Kinect are set up, the user simply presses the "start recording" button in the main hub, and the system handles the rest. Recording from the Kinect and the glasses in parallel, downloading data and saving it with the starting timestamp of each device. Thus, the data is ready for subsequent data processing steps.

Implementing the methodology

This section describes how the methodology can be implemented using the system that was described in the last chapter.

First stage: Data Acquisition from Multiple Perspectives

- The first step involves launching the **Kinect Hub** and **Glasses Hub** modules to establish a stable connection with their respective devices. The Kinect camera must be physically connected to the computer via a wired connection to ensure consistent data transmission. The Glasses Hub should indicate a successful connection, confirmed by the appropriate status signal within the hub's interface.

- With the G3 glasses successfully connected, the next step is to calibrate the device for the specific user. Calibration is performed by having the subject hold the calibration marker - provided within the G3 glasses bundle - at arm's length. The subject must focus on the central point of the marker and press the "Calibration" button. This step is essential for aligning the gaze tracking system with the user's eye movements, thereby enhancing the accuracy of the gaze vector data that will be recorded.

- Once the calibration is complete, recording can be initiated. It is important to start the recording via the **Main Hub** by pressing the "Start Recording" button. Starting the recording from the Main Hub ensures that both the Kinect camera and the G3 glasses begin recording simultaneously. This is critical for maintaining temporal alignment between the two data streams, allowing the outputs to be correctly routed and stored in the same designated folder.

- After the data collection session is complete, recording should be stopped using the "Stop Recording" button in the Main Hub. Stopping the recording from the Main Hub ensures that both the Kinect and the G3 glasses cease recording simultaneously, although it is possible to stop recordings individually from the Kinect or Glasses Hubs.

After recording, the program automatically requests the glasses to download the newly recorded program and saves it in the "recordings/" directory.

Second stage: Synchronization & integrity verification of Visual Data

This part consists of using the Frame Pre-Processor module.

Before we could utilize the recorded data for further analysis - it was essential to organize the data systematically and to verify its integrity. There were two main challenges in this part. The first is that there was a slight temporal offset between the devices. Thus, generating meaningful pairs of <Kinect image, Gaze data> required a fine level of granularity. The second is that rapid eye movement caused a high level of variance and disturbance in the data. That required filtering the data in cases of high variance.
To achieve this, we developed a **Frame Pre-Processor Module** that processes the Kinect images by attaching the gaze direction data captured by the G3 glasses to each image.

Algorithm Overview

The algorithm operates in the following manner:

1. **Data Loading and Synchronization:**
   All gaze direction data is first loaded into priority queues, where entries are ordered by their earliest timestamp. Given that there is a slight temporal offset between the start times of the Kinect and the G3 glasses, the gaze data is synchronized relative to the Kinect's start time to ensure temporal alignment.
2. **Kinect Recording Scanning:**
   The algorithm then scans the Kinect recording folder to identify the exact timestamps at which the Kinect captured images. This enables the system to correlate the image data with the corresponding gaze direction data accurately.
3. **Timestamp Comparison and Data Handling:**
   During each iteration, the algorithm identifies the earliest timestamp across all data points and processes it by creating an instance of the `FrameData` class (described in the following page). This class is responsible for handling the modification and storage of the data at the current timestamp.
4. **Data Saving and Organization:**
   When a new Kinect image is processed, the algorithm saves the image alongside the relevant gaze direction and any other pertinent data in a new folder. This process continues until all images have been paired with their corresponding gaze data, resulting in a well-organized dataset ready for further analysis.

Verifying data integrity and temporal proximity - The `FrameData` Class

As mentioned in the previous page, the `FrameData` class plays a role in addressing the challenges of data integrity and filtering within the algorithm. As the algorithm processes Kinect images and attempts to match them with the corresponding gaze direction data, the `FrameData` class ensures that only the most accurate and temporally aligned samples are retained.
This class serves two critical functions within the data processing pipeline:

- **Memory State Management:**
  It updates the module's "memory state," tracking the most recent images and data added to the module at each timestamp. This ensures that all relevant data is available when needed for processing.
- **Data Organization and Integrity Check:**
  Upon receiving a new Kinect image, the class saves it along with all relevant files - such as gaze data and depth information—into a new directory.
  The class also implements stringent data filtering criteria to maintain the integrity of the dataset:
  - **Temporal Proximity:** If the most recent gaze sample is more than a few milliseconds old, or if there is no sample available, the data is deemed unreliable and discarded.
  - **Gaze Jump Detection:** If a recent gaze sample deviates significantly from the mean of a sliding window of samples, it may indicate a sudden and large gaze shift just before the Kinect image was captured. Given the slight imperfections in synchronization, such a discrepancy could lead to inaccurate data pairing and is therefore filtered out.
  - **"Jitter "Analysis (REM):** High variance in the recent window of gaze samples suggests significant eye movement ("jitter"), potentially compromising the accuracy of the most recent sample. Such data is excluded to ensure the dataset's quality.

By discarding any images and gaze samples that fail to meet these criteria, we ensure that only the most accurate and reliable data is retained. This careful curation of the dataset simplifies the subsequent application of the head position transformation, making the overall process more efficient and the results more reliable.

Third stage: Head Pose Estimation

Following the alignment of the Kinect images with the gaze vectors derived from the Tobii Pro G3 glasses, the next step involves estimating the rotation of the subject's head pose.

To achieve this, we employ a method that utilizes AprilTags, which were placed during the Kinect capture. AprilTags are fiducial markers, each with a unique identifier, and in our setup, they were printed at a standard size of 50x50 mm. The AprilTags serve as reliable reference points within the captured images, allowing us to determine the spatial orientation of the subject's head with respect to the scene camera.

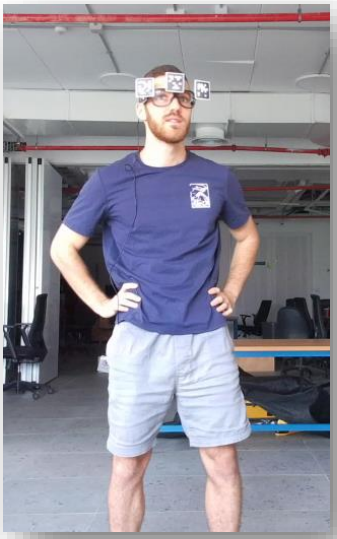
The process begins by detecting the AprilTags within each image. The size of the AprilTag as it appears in the image, along with its position, is then used to calculate its rotation relative to the scene camera.

To enhance the robustness of our head pose estimation, we utilized five AprilTags affixed to the glasses worn by the subject during filming. The use of multiple tags increases the likelihood of detecting at least one tag in each captured frame, thereby improving the reliability of the head pose estimation.

At a minimum, the visibility of one AprilTag is sufficient to estimate the head pose rotation. This is because the known angles between the AprilTag and the subject's head allows us to infer the head's orientation based on the tag's position and angle in the image.

The figure on the right shows the locations and types of April tags that we have used, to find the head pose of the person wearing the glasses.
The April tags used are located in this repository:
https://github.com/AprilRobotics/apriltag-imgs

The figure on the right illustrates how the G3 glasses with the April tags appear from the perspective of the scene camera.



Fourth stage: Gaze Vector Transformation

After successfully gathering and organizing the data into cohesive image/gaze direction packages, we are left with a comprehensive dataset consisting of images captured from the perspective of the Kinect, gaze vectors derived from the Tobii Pro G3 glasses, and the corresponding rotation and position of the subject’s head in three-dimensional space.

The ultimate objective of this process is to enable an algorithm to analyze an image from its perspective and accurately estimate the direction in which the person in front of it is looking. To achieve this, it is crucial that the gaze direction is expressed in the same coordinate system as the Kinect camera's perspective.

This requirement underscores the importance of determining the subject's head position, as the rotation and orientation of the head—described through a linear transformation—are key to accurately transforming the gaze vectors. Specifically, the head's rotation provides the exact linear transformation needed to convert the gaze vector from the perspective of the glasses to that of the Kinect camera.

It is important to note that because rotations around different axes are not commutative, the order in which these transformations are applied is critical. This non-commutative nature of rotational transformations necessitates a careful and precise calculation of the sequence in which each axis' rotation is applied to ensure that the final transformed gaze vector accurately represents the subject’s gaze from the Kinect's viewpoint.

The task of performing these transformations falls to the Frame Post-Processor Module. This module is responsible for calculating the necessary transformation for each image based on the rotation values of the head position. Once the appropriate

transformation is determined, it is applied to the corresponding gaze vector, effectively translating it into the Kinect’s frame of reference.

With the completion of this transformation process, the dataset is fully prepared and ready for its intended application. The final dataset, now consisting of images with accurately aligned gaze vectors in the Kinect's perspective, is suitable for use as training data for machine learning algorithms, such as Convolutional Neural Networks (CNNs).

Discussion

The successful execution of this project demonstrates the viability of the proposed methodology for generating a precise and robust dataset tailored for training gaze detection models. However, despite the positive outcomes, the methodology presents several limitations that warrant careful consideration.

Synchronization Challenges

A critical aspect of this methodology is the synchronization of the glasses and the client device, which automatically connect to the same NTP (Network Time Protocol) server, ensuring that their internal clocks are closely aligned. Nevertheless, even with this synchronization, the process of initiating the Kinect recording and immediately saving the current NTP timestamp introduces a delay of a few milliseconds. This seemingly minor delay can have significant implications, particularly when dealing with small and rapid eye movements.

To mitigate this issue, data suspected of compromised accuracy due to these delays are filtered out. While the delay appears to be relatively consistent within recordings, it likely varies across different machines, complicating the identification of optimal parameters for this filtering process. Determining the best approach for this filtering is a non-trivial task and represents a critical challenge for the robustness of the dataset.

Head Positioning Complexities

Accurately determining the head direction of the individual wearing the glasses emerged as one of the most conceptually challenging aspects of the project. The need to account for three degrees of freedom in head rotation added to the complexity. The chosen solution involved attaching AprilTags to the glasses and using Kinect's imaging and depth data to determine their orientation.

However, this approach is not without its drawbacks. The AprilTags, being attached in a non-rigid manner, may shift relative to the glasses, leading to minor inaccuracies in determining the glasses' orientation. This inaccuracy potentially compromises the precision of the gaze data collected. Additionally, these april tags appear in the image, blocking some of the subject's facial features, introducing biases into the dataset, as will be discussed in the following sections.

Alternative methods were considered but found to be either cumbersome or insufficiently accurate. For instance:

- **Using the Built-in IMU:** The glasses' built-in IMU (Inertial Measurement Unit) consists of a gyroscope, magnetometer, and accelerometer, theoretically capable of determining the glasses' orientation in 3D space, assuming a known starting position. However, the gyroscope's poor accuracy rendered this solution impractical.
- **Utilizing Room-Positioned AprilTags:** Another approach involved placing AprilTags around the room to triangulate the glasses' position using the built-in front-facing camera. However, this method presented two significant challenges: first, the complexity of the necessary calculations and the precision required in placing the AprilTags; and second, the limitation it imposed on the recording environment, restricting the range of images that could be generated for the dataset.
- **Leveraging Pre-existing Tools for Head Position Estimation:** Another potential solution considered was the utilization of an existing machine learning algorithm specifically trained to estimate head position and orientation. This approach would allow us to combine this head position data with our gaze data to accurately infer the direction of the subject's gaze. While this method could have been the most convenient, we ultimately decided against it. Our primary goal was to develop a fully self-reliant system, minimizing the introduction of potential inaccuracies inherent in pre-trained algorithms. By avoiding dependence on external machine learning models, we ensured that the integrity and precision of our data remain under our direct control throughout the entire process.

Dataset Biases

One of the most notable limitations of this project is the introduction of biases in the dataset. Since all participants were required to wear the glasses to capture their gaze direction, the dataset consists solely of images featuring individuals wearing glasses. This uniformity could introduce a bias in the model trained on this dataset, potentially impacting its generalization to real-world scenarios.

Several potential solutions were considered to address this bias:

- **Supplementing the Dataset:** One option was to augment the dataset with images not captured using the glasses and with gaze directions calculated through different methods. However, this approach would contradict the project's core principles, potentially undermining its benefits of ease and accuracy.
- **Digital Removal of Glasses:** Another solution involved digitally removing the glasses from some or all images. While existing algorithms can achieve this with a reasonable degree of reliability, there remains a risk that these modifications could introduce artifacts, potentially biasing the neural network's learning process. Nonetheless, since the images are only subject to human visual inspection, the impact of such artifacts may be minimal.

In summary, while this methodology has proven effective for generating a dataset suitable for gaze detection model training, the outlined limitations must be addressed to enhance the dataset's reliability and generalizability. Future work should focus on refining synchronization techniques, improving head positioning accuracy, and mitigating dataset biases to fully realize the potential of this approach.

Conclusion

In conclusion, our project developed methodology for creating a dataset meant for improving the accuracy of gaze detection algorithms. Using the Tobii Pro G3 glasses and Azure Kinect camera, we were able to generate a more precise dataset that also helps mitigate the challenge of Head-Gaze Correlation Overfitting.

Challenges such as synchronization delays, head positioning complexities, and potential dataset biases highlight areas for future refinement. Addressing these limitations will further enhance the reliability of the dataset and of the methodology, contributing to advancements in gaze detection research.

References

- Kinect SDK written by Microsoft used in the project: https://learn.microsoft.com/he-il/azure/Kinect-dk/
- G3 glasses site, developer guide and the user manual that we used: https://connect.tobii.com/s/glasses3-get-started?language=en_US https://go.tobii.com/tobii-pro-glasses-3-developer-guide https://go.tobii.com/tobii-pro-glasses-3-user-manual
- pyKinectAzure – the python library used for handling the communication with the Kinect: https://github.com/ibaiGorordo/pyKinectAzure
- g3pylib – the python library used for handling the communication with the g3 glasses: https://github.com/tobiipro/g3pylib
- April tags that we used with the glasses: https://github.com/Aprilrobotics/apriltag-imgs
- References consulted during the development of this project:
    - A Free-head, Simple Calibration, Gaze Tracking System That Enables Gaze-Based Interaction by Takehiko Ohno and Naoki Mukawa: https://www.researchgate.net/publication/220811142_A_free-head_simple_calibration_gaze_tracking_system_that_enables_gaze-based_interaction
    - Monocular Free-head 3D Gaze Tracking with Deep Learning and Geometry Constraints by Haoping Deng and Wangjiang Zhu.: https://openaccess.thecvf.com/content_ICCV_2017/papers/Zhu_Monocular_Free-Head_3D_ICCV_2017_paper.pdf
    - Deep Learning-Based Gaze Detection System for Automobile Drivers Using a NIR Camera Sensor by Rizwan Ali Naqvi, Muhammad Arsalan, Ganbayar Batchuluun, Hyo Sik Yoon and Kang Ryoung Park: https://www.mdpi.com/1424-8220/18/2/456