ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

The Journey Towards Better 3D Gaze Estimation

Developping a data collection protocol for better 3D gaze estimation with an RGBD camera

Semester Thesis

Mathis Kurt Lindner
Department of Information Technology
and Electrical Engineering

Advisors: Paudel Danda
Nikola Popovic
Xi Wang
Supervisor: Prof. Dr. Luc van Gool

June 6, 2023

Abstract

The primary focus of this thesis revolves around the creation of a comprehensive data-set tailored specifically for gaze estimation. In order to ensure the potential for verification, the thesis offers a thorough explanation of the various processes and manipulations involved in its development. The goal was to design a data-set capable of accurately predicting human gaze in three dimensions, while also promoting effortless replication and future expansion. Currently, the availability of publicly accessible data-sets within the realm of human gaze prediction is quite limited. And even when such data-sets do exist, they often only provide eye images and two-dimensional gaze data. Additionally, a significant portion of these data-sets tend to offer solely synthetic labels, further contributing to the scarcity of authentic and diverse data. Recognizing this void, our efforts were focused on bridging this gap. To address these limitations, we established an open-source data-set that is easily accessible to anyone interested in the field. This data-set serves as a valuable resource for researchers and practitioners alike, facilitating their exploration and advancement in gaze estimation. By making the data-set openly available, we encourage collaboration.

Acknowledgements

I would like to express my sincere gratitude and appreciation to all those who have contributed to the completion of this thesis. Their guidance, support, and invaluable contributions have played a significant role in the success of this research endeavor.

First and foremost, I would like to extend my deepest gratitude to my supervisors for their unwavering support and mentorship throughout this journey. Their expertise, guidance, and constructive feedback have been instrumental in shaping the direction and quality of this thesis. I am truly grateful for their patience, encouragement, and belief in my abilities.

I would also like to express my heartfelt thanks to all the individuals who willingly participated in the creation of the data-set for this research. Their invaluable contribution and cooperation have made this possible. Their willingness to share their time and efforts for the collection of data has significantly enriched the quality and diversity of the data-set.

I would like to extend my appreciation to the Computer Vision Laboratory for granting me access to the necessary resources and facilities, which were crucial for the data collection process. Their support has been invaluable in ensuring the smooth progress of this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, the fields of augmented reality and virtual reality have gained significant attention due to their potential to revolutionize various industries and enhance user experiences. One crucial aspect of AR and VR applications is the ability to accurately predict and track the user's gaze in three dimensions. Gaze estimation plays a vital role in understanding user intent, enhancing interaction, and creating immersive experiences in these domains. We are in the middle of this turning point: Apple announced on the 5th of June 2023 that they will release a new Virtual/Augmented Reality Headset that heavily relies on eye tracking. It is very unlikely that they will share the details about the data-set they used to train their model. That is why we went forward with creating a data-set, to allow models to be trained, to accurately predict the human gaze in 3 dimensions. We have created a complete pipeline to easily record new data and label it.

1.1 Focus of this Work

The overarching objective of the project was to develop a comprehensive dataset for the 3D human gaze estimation, and to design a methodology that would facilitate the easy replication and expansion of the data-set. By leveraging this approach, we can easily generate more data in various conditions that can be used for a variety of applications such as developing new assistive technologies.

1.2 Research Problem and Objective

In the realm of the Human Gaze Prediction there are a few data-sets that exists that we discuss in the related work section. To put it vaguely: the Issue at hand is that there are no open source data-sets that connect images of eyes and gaze data in 3D. The existing data-sets about predicting 2D Gaze or they will have synthetic labels instead true 3D Gaze. This is why our objective was to start the an open-source easily replicable data-set that can be done anywhere with the tools we describe in 3.

1.3 Thesis Organization

To achieve a comprehensive overview, we will begin by introducing the concept of gaze estimation and providing some background information to contextualize our research. Next, we will compare and contrast our data set with existing ones in order to highlight the unique contributions that our work brings.

Subsequently, we will delve into the technical details of how we recorded and processed our data, including a discussion of the equipment and techniques that we employed. In doing so, we hope to provide a transparent and replicable methodology that can serve as a foundation for future research.

Furthermore, we will present our first dataset, showcasing its structure, content, and organization. This will include a discussion of the factors that we considered when selecting participants and tasks, as well as the measures that we used to ensure the quality and reliability of our data.

Lastly, we will discuss the limitations of our work, including areas where we could have improved our methodology or dataset. We will conclude by summarizing the key insights that we have gained through our research, and outlining potential avenues for future work in the field of gaze estimation.

Chapter 2

Related Work

2.1 Background

Gaze estimation has been an active area of research for several decades, it is motivated by its potential applications in fields such as human-computer interaction, virtual reality, and assistive technologies. The idea behind gaze estimation is to infer where a person is looking based on their eye movements, head orientation, and other physiological and environmental factors. Early approaches to gaze estimation included the Yarbus eye tracker from the 1960s [12] which was rudimentary, but worked. However, with the advent of more sophisticated sensing technologies, such as eye trackers and depth cameras, researchers have been able to develop more accurate and robust methods for estimating gaze direction. Despite these advances, challenges remain in terms of achieving real-time and naturalistic gaze estimation, particularly in complex and dynamic environments. Nevertheless, the potential benefits of accurate gaze estimation are considerable, and the field continues to attract significant interest.

2.2 Existing Data-sets

In the realm of gaze estimation, there are several methods of recording the eyes, and labeling the targets.
In the review and benchmark of Appearance-based Gaze Estimation With Deep Learning, a few of those methods are presented: Methods include skin electrodes around the eyes, infrared cameras and web cameras [1]. We will focus on comparable data-sets which are the ones that include near-eye infrared cameras which can be on-axis, or off-axis, meaning, some of them occlude parts of one's vision of field.
MagicEyes [11] is a Dataset that has a large variety of subjects involved with an off-axis camera, but unfortunately it is not publicly available.
The Paper OPENEDS2020 is an on-axis dataset that has a per-eye Ground-Truth. [6] These were obtained using a hybrid-model. The labeling is not a 3D true gaze vector data-set.
TüEyeD is a data-set gathered by the University of Tübingen of a considerable size, unfortunately they do not provide the instrincs of the cameras that were used, the fact that they are have such a diverse set of footage makes the prediction even harder. [2] Other approaches, such as with NVGaze include the creation of synthetic data: The images of the eyes are purely animated 3D models [5].
Pupil Labs created a device: the Pupil Invisible with a working prediction algorithm [9]. However, they did not publish their data-set to help others to create their own gaze prediction algorithms. They mention the data-set size and how they validated it but omit the rest.

2.3 How ours differs

We identified the lack of open-source data-sets connecting eye images and 3D gaze data, with existing data-sets primarily focusing on 2D gaze prediction or providing synthetic labels instead of true 3D gaze. Our Approach includes all the necessary information to create a full data-set including labels, with proper 3D gaze. Some of the sources mentioned above either have only pseudo labels predicted by a network and/or do not include the intrinsic and extrinsic of each camera. Providing this crucial information allows the user of the data-set to apply transfer learning comfortably. More specifically: even after training the model on a device with different extrinsics/instrinsics, using transfer learning will allow for great results, rather than making a data-set unusable.

Chapter 3

Data Collection

The goal of the data collection was to record each eye, locate the user and locate the point where the participants were looking at in 3 dimensions.
The Data Collection Pipeline is composed of two separate steps: The first one is recording and the second one is Post-Processing. The repository enabling the execution of these processes is composed of 2 files that can be run independently on different machines that answer these needs.

3.1 Recording Pipeline

The data-set creation involved the utilization of various devices, as illustrated in 3.1 Although the process took place at the Swiss Federal Institutes of Technology, the code can be adapted to be used in different environments. To initiate the pipeline, one can simply launch the "record.py" file after installing the necessary requirements. The user will be guided through the execution via prompts. This approach ensures a seamless experience for users, even in unfamiliar settings.

3.1.1 Devices

Pupil Labs Invisible Glasses and Phone

To track the eye's of the participant we chose the Pupil Labs Invisible glasses 3.2 that consist of three cameras we call:

1. world camera
2. left eye camera
3. right eye camera

Camera	Resolution	Refresh Rate
World	1088x1080	30Hz
Left/Right Eye	192x192	200Hz
Intel RGB and Depth	640x480	30Hz

Table 3.1: Camera Specifications

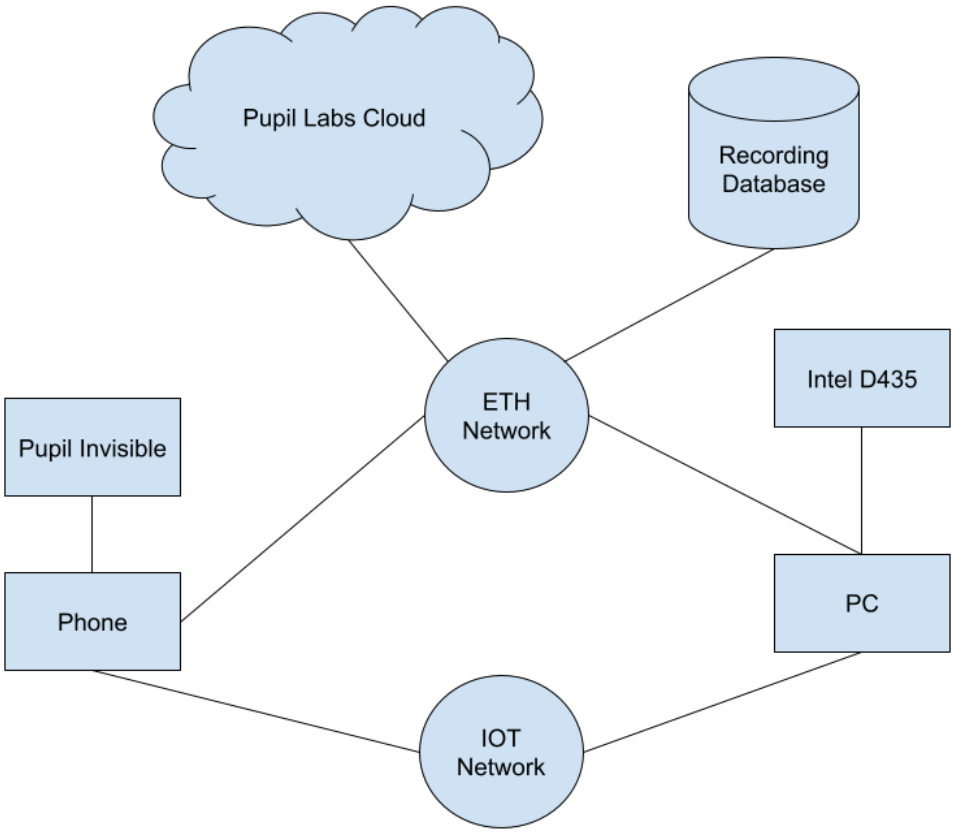


Figure 3.1: Devices Interaction

The world camera face forward, while the other two are infrared cameras that face each eye. The phone connects through USB to the glasses and takes care of recording the three cameras locally. Once the recording is stopped, we can easily upload this data to the Pupil-Cloud, which is the a service provided by pupil-labs which lets us then, transfer the files to the Computer for Post-Processing. The Pupil Invisible also includes IMU data, we are not using this data for localisation but we include it in the data-set.

Depth Camera

To estimate the scale of the 3D reconstruction we needed a depth camera, we go through how this is done in 3.2.4. We chose the The Intel® RealSense™ Depth Camera D435 for this 3.3. The idea is to place it where it can always see the laser to be able to localise the laser's positions anywhere. When recording, we do not process the files from this camera in any way, we have the possibility to save both the depth and the RGB data as a binary file to take care of in a later stage. This recording file also includes details such as the timestamps of when each frame was taken and the intrinsics of the both the depth and RGB modules. As an advantageous byproduct, this helps us reducing the latency of each event trigger thanks to the absence of having to process the videos straight away.

Figure 3.2: The Pupil Invisible Glasses attached to the Phone

Figure 3.3: The Depth Camera attached to a tripod

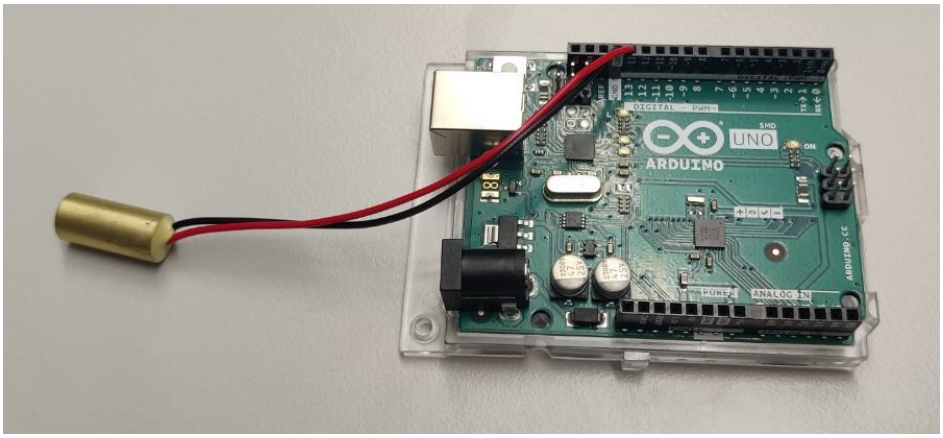


Figure 3.4: The Laser attached to an Arduino board

Computer

To orchestrate the operation, we use a computer that starts the recording on all of the devices.
The Intel® RealSense™ Depth Camera D435 connects to computer directly over a USB port, it communicates thanks to the scripts provided by Intel themselves [3].
The Phone and therefore the Pupil Invisible connects through the network. More precisely, the phone connects to through the IOT sub-network, since the networking rules did not allow otherwise. There were attempts connecting the glasses directly into the computer to ease the synchronisation process between all the devices, but these fell in vain because of driver issues.

Laser

A laser to look at was used to locate the participants gaze. This laser also needs to be visible by the depth camera to be able to locate it in 3 dimensions. To be able to include world images, we made the laser blink with an Arduino Uno board 3.4. If the laser is blinking, the participants are able to follow the laser and we can remove the laser from all the images where it is visible in the world camera. Because of a high refresh rate, enough world images are being created by the pupil labs device to use for a prediction.

3.1.2 Recording Database

For the data-set creation, we needed to extract all the images of all the videos created by the 4 cameras. This process requires a considerable amount of space, since the some of the cameras have a high refresh rate. After labeling, only the necessary files are moved to the exports folder to avoid unnecessary storage waste since one raw recording of 3 Minutes produces approximately 20 GB worth of files.

3.1.3 Recording Implementation Overview

To communicate with the Pupil Labs's Phone, to be able to record remotely without the participant having to take care of anything, we used the Pupil-Labs API to send commands to the phone. We had issues because of the restrictiveness of the ETH-Networks, which forced us to use one network to record and the other to be able to connect to the Internet and upload the files to the Pupil Cloud. More specifically when recording, the phone has to be connected to the ETH-IOT network, which does not have access to the internet and else it can be connected to the normal ETH network. This also helps one to synchronize the phone to the NTP servers which reduces the offset between the phone and the PC. Thankfully we were able to start the

recordings using Pupil Labs API [10] that allows one to start and end recordings conveniently. With that in mind we could combine the start of the recording at the same time with Intel's python wrapper for their camera.

3.2 Post-Processing Pipeline

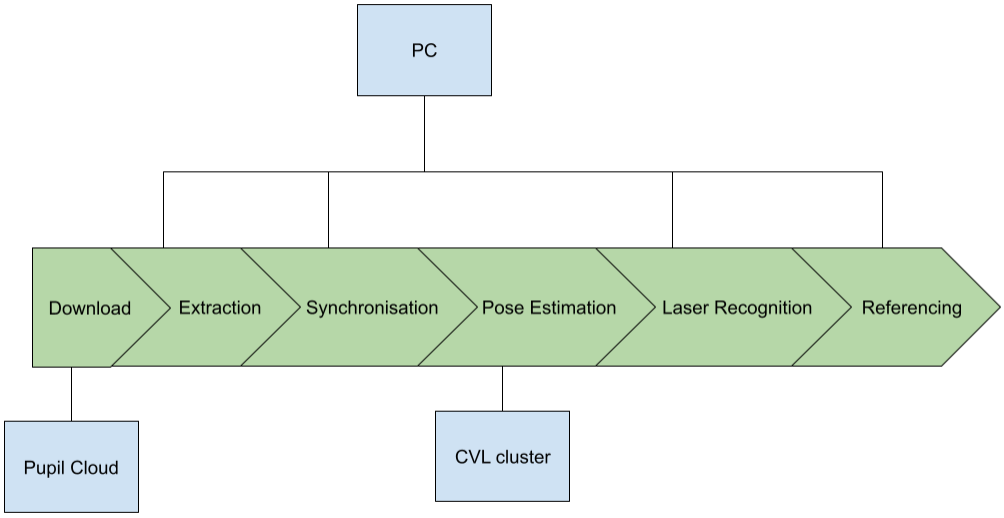


Figure 3.5: Post-Processing Pipeline

The Post-Processing Pipeline consists of gathering the data from each source, meaning all the devices, synchronising them, extracting the necessary information, and finally creating a record of all the files to easily access them.

3.2.1 Download and Extraction

Pupil Cloud

After ending the recording the phone automatically uploads all the files to the pupil-cloud, a script is run to grab all the videos and extract each individual image of each camera to various sub-folders for later processing. We used the ffmpeg library [8] and information about when each image was taken to be able to synchronize the images to each-other and to the other sensors.

Depth Camera Processing

Since we record the video on the Computer, we can directly extract the frames locally. Once recorded we can simulate a replay of the binary file, which lets us save the RGB images as PNG's and the depth information recorded by the depth module as an array, which avoids compression for a more accurate 3 Dimensional localisation of the targets. While we are doing that, we use Intel's functions [3] that lets us align the depth

information with the colored images. This process takes the most amount of time in the post processing pipeline, but is necessary to make sure the depth scale and the laser positions can be accurately pinpointed.

World Camera Processing

The world camera creates a video that is very distorted, thankfully, Pupil Labs include the coefficient to undo this.
The world capturing device can be modeled as a radial camera and it has then the following camera matrix:

$$C = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1.0 \end{bmatrix} \tag{3.1}$$

The distortion coefficient are modeled as such:

$$k = \begin{pmatrix} k1 & k2 & k3 & k4 & k5 & k6 \end{pmatrix} \tag{3.2}$$

These are the tangential distortion coefficients :

$$p = \begin{pmatrix} p1 & p2 \end{pmatrix} \tag{3.3}$$

With these coefficient we can undistort our image as such and then model it as a pinhole model:

$$x'' = x' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 x' y' + p_2 \tag{3.4}$$

$$y'' = y' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + p_1 (r^2 + 2y'^2) + 2p_2 x' y' \tag{3.5}$$

$$\text{where} \quad r^2 = x'^2 + y'^2 \tag{3.6}$$

$$u = f_x * x'' + c_x \tag{3.7}$$

$$v = f_y * y'' + c_y \tag{3.8}$$

We undistorted the images to facilitate the Camera Pose Estimation 3.2.4 using the OpenCV module [4]. The results can be seen in 3.6

3.2.2 Synchronisation

To make sure the 3 cameras on the pupil invisible are synchronized, when we extracted each frame, we verified the timestamps by creating a visualization that would show us this. More specifically, we turned on a flashlight that would be visible on all three cameras as can be seen in 3.7. We can observe that the left and right eye cameras see the flash at the same time, in the same aligned frame. The world camera only sees it a bit later because it has a different refresh rate. The world camera has a refresh rate of 30 frames per seconds and the eye cameras of 200 frames per seconds, which means that, as long as the world camera is in sync within $\lceil \frac{200}{30} \cdot \frac{1}{2} \rceil = 4$ frames, the timestamps are aligned. In our case it was 3, which is acceptable.
At the start of each recording, we save the starting time of the PC and save the offset with the phone using the peer to peer Network Time Protocol. The offset is also printed in the console when starting the recording to monitor. Moreover, to keep track when the subject is looking at the targets, we save this information with timestamps we called "events" as pupil labs named them on their website. These timestamps can be found in the json file called local_synchronisation.json. An example of this file, can be found in the appendix. A.1.1

(a) Distorted World Image

(b) Undistorted World Image

Figure 3.6: COLMAP's Camera localisation with example images

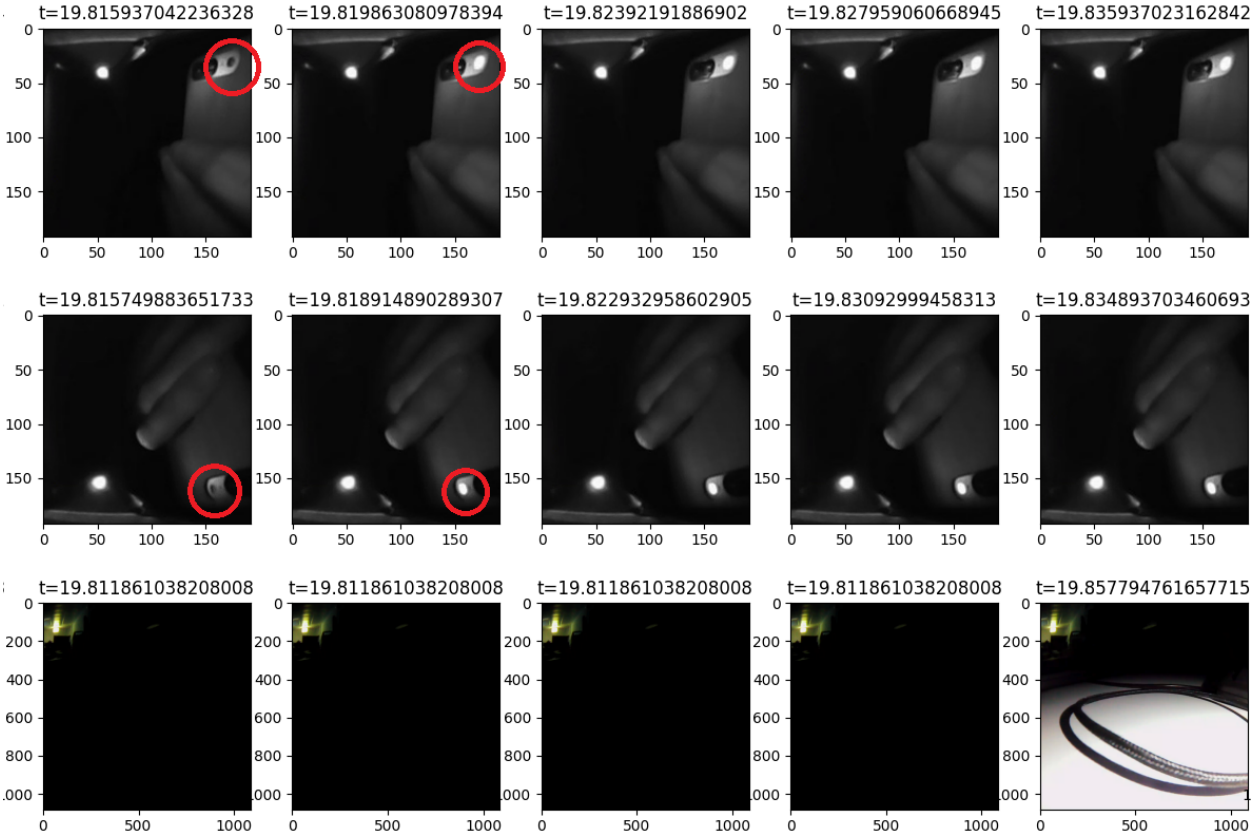


Figure 3.7: Visualization of the Synchronization of the three cameras on the Headset

Camera Name	Model Type	fx	fy	cx	cy
World Camera	Pinhole	766.97	766.49	553.57	543.56
Depth Camera(RGB module)	Pinhole	616.24	616.62	312.24	250.36
Left Eye	Radial	140.87	140.91	105.14	98.20
Right Eye	Radial	140.87	140.45	100.25	97.8

Table 3.2: Intrinsics of the Cameras

3.2.3 Calibration

We verified all the instrinsics given by the manufacturers by using a checkerboard and OpenCV's library [4]. We confirmed that the results between both were nearly identical and were satisfied. The results can be found in 3.2. To ensure reprehensibility, anyone can change the presumed intrisincs for the COLMAP reconstruction in the bash file A.2.1. The images of the world camera that are included in the data-set have been undistorted as mentioned in the world camera processing. The depth camera already comes with undistorted images, but the eye cameras are not. Their distortion coefficients can be found in the data-set.

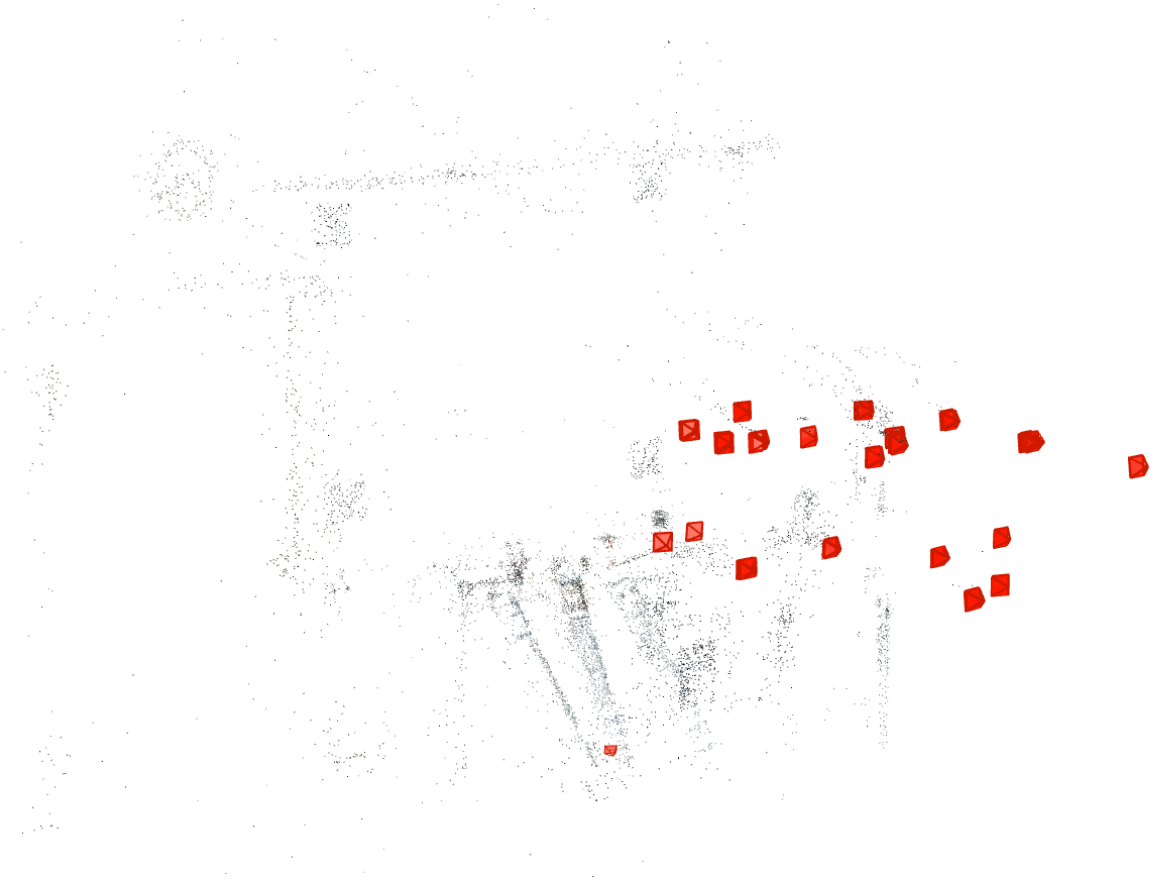
3.2.4 Camera Pose Estimation

To accurately label the gaze in 3D, We need to ensure a good camera pose estimation. For this we chose to use COLMAP [7] in combination with the depth camera to calculate accurately the positions as follows. We first ran COLMAP on a subset of all images: Initially it was 1/10th of every world image was given by the pupil labs device and only one image from the depth camera, since this one was static. Later on, we realised that COLMAP sometimes failed us because the sampled images were blurry, so we changed our approached and only sampled 1/4th of the footage where the participant was static looking at the laser dot, which turned out to work much better. On 3.8 you can observe the depth camera position at the bottom and the world images are the other red squares. The black dots are the sparse points that COLMAP extracted.

To get this result we ran specific commands to make sure the images are registered from the right cameras and giving prior intrinsics, the details of the commands are in the appendix A.2.1. The registration was unfortunately not straight forward: The COLMAP automatic reconstructor did a good job at finding the camera positions and sparse points. But unfortunately this took a very long time since it also creates a dense map among other files which were not needed in our implementation. After skimming through the documentation another option came up. Unfortunately the "vocab-tree-matcher" threw errors that seemed non-solvable, so we chose to replace it by an "exhaustive-matcher", which ended up working perfectly A.2.1.
To verify that COLMAP and the depth camera work together how we intended them to, we created point-clouds from each of these and aligned them. The results can be seen in 3.2.4. You can see the table in both the green and black point-cloud and that they overlap. Unfortunately, since the points are quite sparse, it is difficult to tell how well they are aligned in a 2D projection.

3.2.5 Laser recognition

To recognize the laser in the depth camera frames, we used a series of functions provided by the OpenCV module [4]. More specifically it is about filtering the red dot, contouring the laser and then finding its center of mass. Since we had trouble finding the laser on any surface because the laser is not bright or and large enough for the RGB module of our depth camera, we chose to put up Aruco markers to segment the wall and then made sure to only look for the laser in that space. The aruco Markers detection and the laser recognition with the segmented are can be visualised in 3.10.



(a) COLMAP Camera Localisation

(b) Depth camera perspective

(c) World Camera Perspective

Figure 3.8: COLMAP's Camera localisation with example images

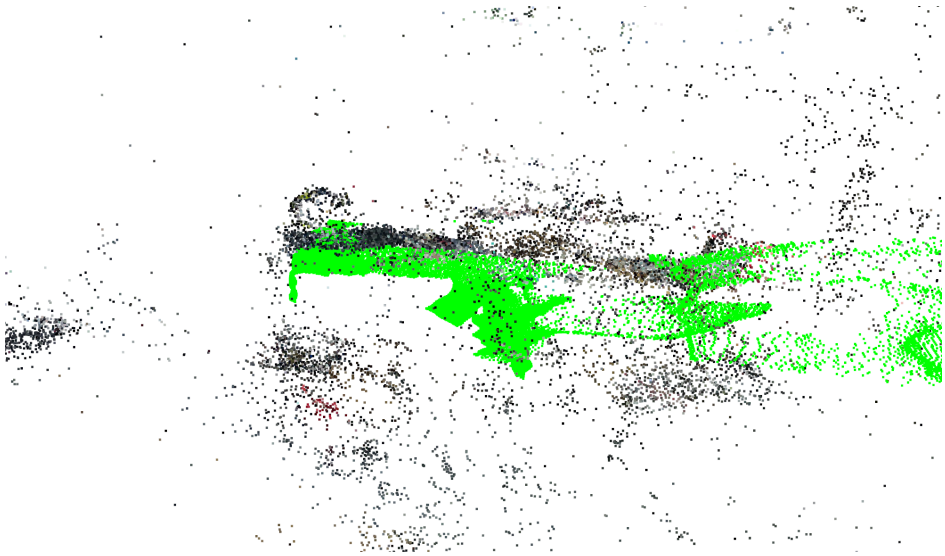
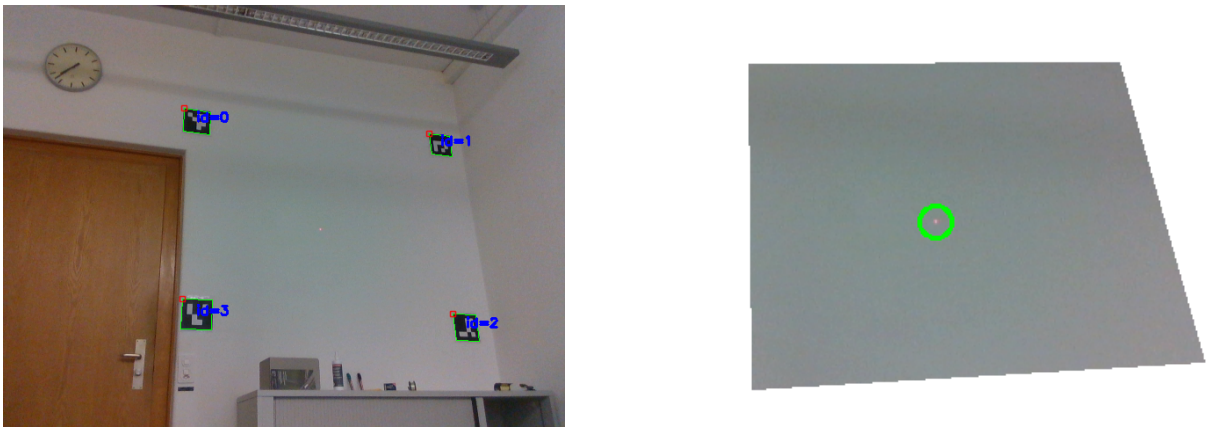


Figure 3.9: Comparison between Depth Camera pointcloud(green) and COLMAP pointcloud



(a) Detected Aruco Markers

(b) Laser detected within the aruco markers

Figure 3.10: Laser and Aruco Markers detected

3.2.6 3D Gaze estimation

Once we localised the depth camera, we could extract the 3D position of where the participant is looking at, by projecting the laser seen in the RGB module on the depth camera, into the space while using the depth module's information. 3.2.

- Camera intrinsics Cx, Cy, fx, fy
- Depth Camera Tranlation vector $T = \begin{bmatrix} Tx \\ Ty \\ Tz \end{bmatrix}$
- Depth Camera Rotation Matrix R
- Laser Pixel Location $\begin{bmatrix} u_i \\ v_i \end{bmatrix}$

- Depth at pixel i location f_i

Estimate depth scale

To estimate COLMAP's to the true scale, we first inferred the distances from of all the sparse points created by COLMAP that were visible by the depth camera to the center of its estimated position.
Concretly for each sparse point that is visible by the depth camera $\left(x_i, y_i, z_i\right)$

$$d_i = \sqrt{(x_i - Tx)^2 + (y_i - Ty)^2(z_i - Tz)^2)} \tag{3.9}$$

Once we have this distance for each point, we also need the depth that is given by the depth camera. For that we first need to estimate on which pixel the projection of the point in 3D lands.
We will first create the Projection matrix, which is a 3x4 Matrix.

$$P_{3\times4} = \left(R^T|T\right) \tag{3.10}$$

The Instrisic matrix can be created as such:

$$Int = \begin{pmatrix} f_x & 0 & C_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.11}$$

Then we can project the points as such:

$$\begin{pmatrix} \alpha_i \\ \beta_i \\ \gamma_i \end{pmatrix} = Int_{3\times3} \cdot P_{3\times4} \cdot \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \tag{3.12}$$

Then we get our u_i and v_i:

$$u_i = \frac{\alpha_i}{\gamma_i} \tag{3.13}$$

$$v_i = \frac{\beta_i}{\gamma_i} \tag{3.14}$$

We can round both values to their nearest pixels and access these specific values using the depth array. This will give us for each point f_i a specific depth. Finally we can calculate the mean of all the distances d_i found by colmap and all the depths f_i.

$$\text{scale} = \frac{1}{n} \sum_{\forall i} \frac{d_i}{f_i} \tag{3.15}$$

3D Laser position estimation

After finding the scale, we can apply this one to all the sparse points and camera positions calculated by COLMAP. Once these vectors are scaled, it is a matter of finding the positions of the laser in the COLMAP coordinates. This can be done by first using Intel's function that lets us project a pixel in the 3D space of the depth camera coordinates. Then we can apply the transformation that lets us go from the depth camera's coordinates, to the COLMAP coordinates. Thanks to the depth camera localisation we have all the necessary vectors to do that: the translation vectors and the quaternions, which let us calculate the rotation matrix can be inverted and applied to the 3 dimensional points that were given by Intel's function.

3.3 Export

To ensure anonymity of all the participants, we carefully extract only the necessary files for to train a model. This is done using the file generated by Pupil Labs which is called wearer.json. It is not transferred to the exports folder but we use the generated id's and add this information to each full_df.csv which makes each wearer anonymous, yet unique.
The export folders includes the COLMAP poses of the images, the laser positions in 3D and it's projected coordinates to the depth and world camera. We also added the imu information with their timestamps and only the necessary images to reduce the space that is needed A.3.1. This information is referenced in a large data-frame in the root folder of all the exported recordings to access all the images conveniently gaze by gaze from one data-frame.

Chapter 4

Dataset

The first iteration of the data-set includes images of each eye of 20 distinct participants including almost 500,000 images of each of their eyes and where they are looking at. It also includes the location of laser from the depth cameras view and the 3 dimensional location in the space.

4.1 Recording Protocol

Before starting the recording, the participant should properly be introduced to what he is allowed to do and what he is not: The participant should not move too quickly so the images are clear, but you have to make sure that they try to move enough in the space, so COLMAP does a good job at localising all the cameras. Ask the participant to always try to face the wall so that COLMAP always has enough reference points to create the sparse model.
The protocol for the data acquisition is as follows:

1. The participant is standing in front of the wall where the depth camera is facing.
2. The laser pointer is turned on
3. The recording is started
4. The laser pointer will be pointed at a random position, the participant is asked to look at the laser pointer for 3 seconds with a random position and random viewing angle. The operator will ask for a confirmation from the participant before pressing space to trigger an event.
5. Step 5 is repeated 20 times.
6. The recording is stopped.

In case that the participant accidentally does not look at the target, he is asked to mention is to the Recording Operator who will cancel the recording, which will automatically delete both the recording on the phone and on the Computer.

4.2 Subjects

The first iteration of this dataset has been created with a bias towards people that are white and aged between 20-25. To document the ethnicity we used the following reference: categorization of ethnicity. The details can be seen in the table 4.1

(a) Subject (b) Operator

Figure 4.1: The Subject and the Operator when Recording

Sex	Eye Color	Ethnic Group
Male: 16	Brown: 15	White: 18
Female: 4	Blue: 5	Asian: 1
		Middle Eastern: 1

Table 4.1: Participant Characteristics

Figure 4.2: Images from the data-set

Size of export folder	48.32 GB
Total number of images	447,457
Number of Gazes	216,127
Number of recordings	33

Table 4.2: Statistics about Data-set

4.3 Data-Set Statistics

After exporting only the necessary files we ended up producing over 200,000 gaze points 4.2 connected to their respective images, an example of what a batch looks like can be seen in 4.2.

To verify the integrity of the randomness that we tried to propose, we supply a heat-map of the laser positions that were recorded from the depth camera's perspective. Keep in mind that this is not projected to a plane: this explains why the heat-map may look like a parallelogram and not a rectangle. One laser position is 400 points. Each unit, is the laser-point, recognized by the depth camera, at a certain position, for 1/200th of a second.4.3

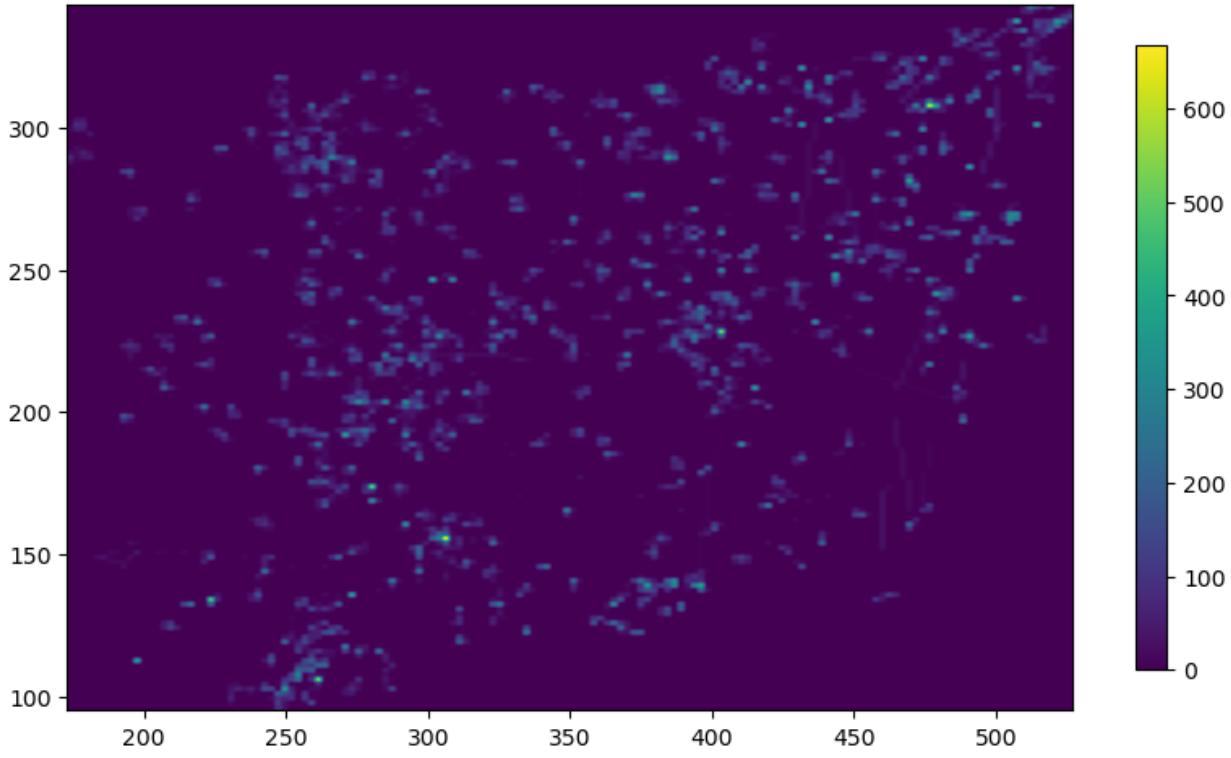


Figure 4.3: Heatmap of the detected laser

Chapter 5

Conclusion

In this thesis, we addressed the need for a comprehensive data-set for gaze estimation in the emerging fields of augmented and virtual reality. We developed a complete pipeline to record and label new data. Our overarching objective was to create an open-source and easily extendable data-set that lets one create applications that accurately predicts the human gaze in 3D.

5.1 Data Collection

The Data-collection Pipeline we have created is fully functioning and always ready to collect more data, The goal that was set, was achieved. It is easy to use and fully automated. If someone has the same devices at hand, it is a matter of plug and play. One of the key strengths of our pipeline is its flexibility and extensibility. We have provided tools and protocols that make it easy to replicate and build upon our work, and we encourage others to use it and extend the data-set.

5.2 Data-Set

To demonstrate the capabilities of our Gaze Data Acquisition we have also included a data-set that is meaningful enough to start training on a Deep Neural Network. Unfortunately, the fact that the laser was not powerful enough to be easily seen anywhere in a scene made our work more difficult and restricted us in our collection. The world camera images cannot be used for a prediction because we had to use a white wall to make it all possible. Nevertheless, our data-set includes a true 3 Dimensional gaze with all the intrinsics that are necessary to train a model. Overall, we believe that our data-set represents an important step forward in the field of gaze estimation by making it true 3D gaze and open-source.

5.3 Future Work

The purpose of this Gaze Data Acquisition Pipeline and the data-set was to enable future work as described thoroughly. We wish to see this work be extended to a larger space and have more participants to be included, to have a more diverse data-set. Moreover, we wish that the extension of the data-set stays open-source to let everyone in on the research involving the human gaze. We originally had issues with the laser detection, we had to carefully choose the room to record our data-set, but this can be fixed with a better RGB-D camera and a more powerful laser. This would make the pipeline applicable to every space imaginable. Once this issue is fixed, recording in different rooms of different size and different lighting conditions can make the data-set more diverse.

To help with the laser recognition, the information given by pupil labs can be helpful: instead of looking for the laser in the whole image, it would be possible to restrain the search to only a a certain radius around the gaze computed by pupil labs.
The option to make the laser blink, would allow one give the world images to a network, to learn new predictions to make upon them and not just the 3 dimensional vectors. This could be interesting for many applications.

Appendix A

The First Appendix

A.1 Files Examples

A.1.1 synchronisation json

Listing A.1: local_synchronisation.json


```
{
    "system_start_time": 1685610133053114837,
    "offset": -70390000,
    "Event: 0": 1685610137564849073,
    "Event: 1": 1685610142956872181,
    "Event: 2": 1685610149519215103,
    "Event: 3": 1685610155824216520,
    "Event: 4": 1685610161964813694,
    "Event: 5": 1685610168836737341,
    "Event: 6": 1685610176036805063,
    "Event: 7": 1685610182348682162,
    "Event: 8": 1685610190423257950,
    "Event: 9": 1685610197364670176,
    "Event: 10": 1685610204236555199,
    "Event: 11": 1685610212476554843,
    "Event: 12": 1685610219037166190,
    "Event: 13": 1685610226108464300,
    "Event: 14": 1685610233471721812,
    "Event: 15": 1685610240396419609,
    "Event: 16": 1685610247836372482,
    "Event: 17": 1685610255148387669,
    "Event: 18": 1685610262396327711,
    "Event: 19": 1685610268908294826
}
```


A.2 Programm Code

A.2.1 COLMAP localisation

Listing A.2: COLMAP commands

```
#!/bin/bash
#first argument should be the ws path
EM_WS_PATH=$1

echo "running Exhaustive Matcher on $EM_WS_PATH"
COLMAP_OUT_PATH=$EM_WS_PATH/exhaustive_matcher_out
IMAGES_PATH=$EM_WS_PATH/all_images
DATABASE_PATH=$EM_WS_PATH/database.db
#extract images from world camera
colmap feature_extractor \
   --database_path $DATABASE_PATH \
   --image_path $IMAGES_PATH \
   --image_list_path $EM_WS_PATH/world_images.txt \
   --ImageReader.single_camera_per_image false \
   --ImageReader.single_camera_per_folder true \
   --ImageReader.camera_model PINHOLE \
   --ImageReader.camera_params "766.9718099563071, 766.4976087911597,
   553.5783632028212, 543.5611312035519"

colmap exhaustive_matcher \
   --database_path $DATABASE_PATH

mkdir $COLMAP_OUT_PATH
mkdir $COLMAP_OUT_PATH/only_world
mkdir $COLMAP_OUT_PATH/only_world/sparse
#localise cameras and sparse points from the world camera
colmap mapper \
    --database_path $DATABASE_PATH \
    --image_path $IMAGES_PATH \
    --output_path $COLMAP_OUT_PATH/only_world/sparse \
    --image_list_path $EM_WS_PATH/world_images.txt
##################################################################
colmap feature_extractor \
   --database_path $DATABASE_PATH \
   --image_path $IMAGES_PATH \
   --image_list_path $EM_WS_PATH/depth_images.txt \
   --ImageReader.single_camera_per_image false \
   --ImageReader.single_camera_per_folder true \
   --ImageReader.camera_model PINHOLE \
   --ImageReader.camera_params "616.2463989257812, 616.6265258789062,
   312.24853515625, 250.3607940673828"

colmap exhaustive_matcher \
   --database_path $DATABASE_PATH

mkdir $COLMAP_OUT_PATH/world_and_depth
```

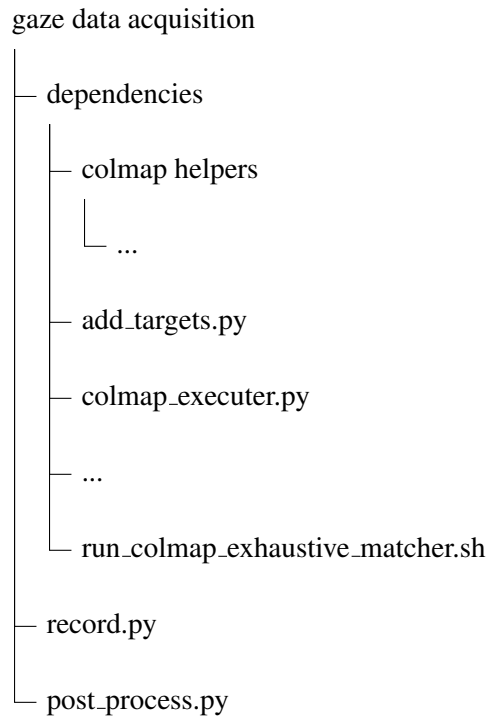
```
mkdir $COLMAP_OUT_PATH/world_and_depth/sparse
#localise the depth image in the model created with the world images
colmap mapper \
    --database_path $DATABASE_PATH \
    --input_path $COLMAP_OUT_PATH/only_world/sparse/0 \
    --image_path $IMAGES_PATH \
    --output_path $COLMAP_OUT_PATH/world_and_depth/sparse \
    --image_list_path $EM_WS_PATH/depth_images.txt \
    --Mapper.ba_refine_extra_params 0 \
    --Mapper.ba_refine_focal_length 0 \

colmap bundle_adjuster \
    --input_path $COLMAP_OUT_PATH/world_and_depth/sparse \
    --output_path $COLMAP_OUT_PATH/world_and_depth/sparse \
    --BundleAdjustment.refine_extra_params 0 \
    --BundleAdjustment.refine_focal_length 0  \
    --BundleAdjustment.refine_extrinsics 0

#create text file that says that the exhaustive matching is done
touch $EM_WS_PATH/exhaustive_matching_done.txt
```

A.3 Full Structures

A.3.1 Exports Structure

- exports
  - all_dfs.csv
  - recording xxxx-xxxx-xxxx-xxxx
    - world
      - 0.png
      - ...
    - left
      - 0.png
      - ...
    - right
      - 0.png
      - ...
    - depth png
      - 0.png
      - ...
    - colmap_exports
      - cameras.txt
      - images.txt
      - points.txt
    - imu.csv
    - fulldf.csv
  - recording yyyy-yyyy-yyyy-yyyy
    - ...

A.3.2 Code Structure



Bibliography

[1] Yihua Cheng, Haofei Wang, Yiwei Bao, and Feng Lu. Appearance-based gaze estimation with deep learning: A review and benchmark, 2021.

[2] Wolfgang Fuhl, Gjergji Kasneci, and Enkelejda Kasneci. TEyeD: Over 20 million real-world eye images with pupil, eyelid, and iris 2d and 3d segmentations, 2d and 3d landmarks, 3d eyeball, gaze vector, and eye movement types. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, oct 2021.

[3] Intel. Intel® realsense™ sdk 2.0. `https://github.com/IntelRealSense/librealsense`.

[4] Itseez. Open source computer vision library. `https://github.com/itseez/opencv`, 2015.

[5] Joohwan Kim, Michael Stengel, Alexander Majercik, Shalini De Mello, David Dunn, Samuli Laine, Morgan McGuire, and David Luebke. Nvgaze: An anatomically-informed dataset for low-latency, near-eye gaze estimation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–12, New York, NY, USA, 2019. Association for Computing Machinery.

[6] Cristina Palmero, Abhishek Sharma, Karsten Behrendt, Kapil Krishnakumar, Oleg V. Komogortsev, and Sachin S. Talathi. Openeds2020: Open eyes dataset, 2020.

[7] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] Suramya Tomar. Converting video formats with ffmpeg. *Linux Journal*, 2006(146):10, 2006.

[9] Marc Tonsen, Chris Kay Baumann, and Kai Dierkes. A high-level description and performance evaluation of pupil invisible, 2020.

[10] Marc Tonsen, Chris Kay Baumann, and Kai Dierkes. A high-level description and performance evaluation of pupil invisible. *arXiv preprint arXiv:2009.00508*, 2020.

[11] Zhengyang Wu, Srivignesh Rajendran, Tarrence van As, Joelle Zimmermann, Vijay Badrinarayanan, and Andrew Rabinovich. Magiceyes: A large scale eye gaze estimation dataset for mixed reality, 2020.

[12] Alfred L. Yarbus. Eye movements and vision. *Eye movements and vision*, 1967.