IDIAP RESEARCH REPORT

EYEDIAP DATABASE: DATA DESCRIPTION AND GAZE TRACKING EVALUATION BENCHMARKS

Kenneth Alberto Funes Mora Florent Monay
Jean-Marc Odobez

Idiap-RR-08-2014

Version of SEPTEMBER 18, 2014

Centre du Parc, Rue Marconi 19, P.O. Box 592, CH - 1920 Martigny
T +41 27 721 77 11 F +41 27 721 77 12 info@idiap.ch www.idiap.ch

EYEDIAP Database: Data Description and Gaze Tracking Evaluation Benchmarks

Kenneth Alberto Funes Mora, Florent Monay, Jean-Marc Odobez

Revision 2: 2nd of September 2014

Abstract

The lack of a common benchmark for the evaluation of the gaze estimation task from RGB and RGB-D data is a serious limitation for distinguishing the advantages and disadvantages of the many proposed algorithms found in the literature. The EYEDIAP database intends to overcome this limitation by providing a common framework for the training and evaluation of gaze estimation approaches. In particular, this database has been designed to enable the evaluation of the robustness of algorithms with respect to the main challenges associated to this task: i) Head pose variations; ii) Person variation; iii) Changes in ambient and sensing conditions and iv) Types of target: screen or 3D object. This technical report contains an extended description of the database, we include the processing methodology for the elements provided along with the raw data, the database organization and additional benchmarks we consider relevant to evaluate diverse properties of a given gaze estimator.

1 Introduction

In recent years there has been a growing interest from diverse domains for using and developing tools able to automatically retrieve gaze information. Such tools have an important potential in the development of consumer market applications in human computer interfaces, entertainment, assistance to person with disabilities, marketing, etc. In another direction, gaze is also of high interests in the sociology and psychology research where it is considered as one of the most important cues in non-verbal behavior analysis, as it is involved in many cognitive processes such as discourse regulation or conveying consciously or not the emotional state of an individual. While head pose is sometimes sufficient to obtain visual attention information by introducing methods to cope with the absence of eye information like eye and head pose relationships in gaze shifts [1] or conversation context [2], and make inference about social construct like dominance [3], it is still fundamentally limited for finer analysis of gaze behaviors when several attention target direction become close.

Researchers have thus placed important efforts in designing automatic gaze tracking solutions [4, 5, 6, 7, 8, 9, 10, 11, 12, 13], leading to the development of methods which differ according to their sensing technique and principles. Some -highly intrusive- approaches are based on electro-oculography or contact-lense based coils which restrict their use to the controlled conditions of a laboratory. Video-oculography [4], that is gaze tracking relying on video input and in particular on remote (not head-mounted) cameras, is often preferred as a less intrusive and more flexible alternative. Solutions are available in the market, but most of them require specialized hardware such as calibrated setups of infra-red (IR) light sources and IR cameras [8], making them costly or limited in terms of the applications and conditions under which they properly function.

Natural light based methods are thus the best candidates in terms of availability, cost and potential applications. Nevertheless, gaze estimation from remote standard (RGB) cameras remains a very difficult task. The challenges are many: person variability, head pose variations, eyelids movements, illumination conditions, specular reflections, image resolution and contrast. Important

advances have been made leading to many diverse gaze estimation methods that differ in terms of accuracy, generalization, need for calibration, robustness, head pose invariance, etc.

Two main approach categories can be identified in the literature: model-based and appearance-based methods. Model-based methods leverage on a parametric description of the gaze observations, which includes iris/pupil fitting techniques [14], complex models fully describing the eye image appearance [7], and/or geometric representations of the eyeball [9, 11]. However, these methods normally require data with high-resolution and good contrast to infer the geometric parameters.

On the other side, appearance based methods avoid the fitting and tracking of local features by inferring a mapping from the high-dimensional image data to the low-dimensional space of gaze parameters. This make them a potential solution for low-resolution sensing. This mapping has been modeled in diverse ways, such as neural networks [6], Semi-supervised sparse gaussian processes [15], Support Vector Machines [16] or local linear mappings [10, 17].

Even though the evaluation methodologies employed by researchers have clearly advanced the development of gaze tracking technologies, and have validated their different proposals, it is unlikely to encounter in the literature comparisons evaluated on the same data and conditions. This makes it difficult to clearly identify the advantages and disadvantages of each one of the methods. The main reason is the lack of a standard benchmark under which researchers can evaluate their methods and report their results.

We intend to fill this need by releasing EYEDIAP: a database for gaze estimation from remote RGB and RGB-D (standard vision and depth) cameras. We have designed the recording methodology in order to systematically include, and isolate, most of the variables which affect gaze estimation algorithms based on remote sensing: i) Head pose variations; ii) Person variation; iii) Ambient and sensing condition changes and iv) Types of target: screen or 3D object. We have also defined a set of benchmarks which are intended to evaluate each one of these aspects in an isolated manner. We have also pre-processed the data to extract and provide complementary observations (e.g. head pose) helping researchers to focus on only a subset of the problem, if wanted.

Recently the Columbia gaze data set was released by Smith et al. [18]. This dataset is a promising resource to advance the research on gaze estimation, in particular, for the training and evaluation of appearance based gaze estimation methods [19]. It has been carefully collected and contains a large quantity of participants (56). However it also has limitations: the range of head poses and gaze directions is small; there is no temporal information (video could be valuable for a given algorithm); only the RGB image modality is provided; the data is limited to a discrete set of point targets lying on a plane at a fixed distance, making it not so appropriate for evaluating gaze estimation in more natural interaction [20] potentially involving larger head poses.

We believe the EYEDIAP database is an important contribution to the community, and we therefore encourage researchers to develop gaze estimation algorithms and to report results using this data. This document is an extension of our previous paper [21] and it is organized as follows: Section 2 describes the recording methodology and the different recording sessions included in the dataset. Section 3 is devoted to the pre-processed information which is provided along with the data. Section 4 describes the files organization and format. Section 5 provides a description on how to use this dataset, including the definition of different benchmark protocols for evaluating gaze estimation algorithms. Finally, Section 6 concludes this document.

2 Data collection

In this section we first describe our recording methodology and then describe the different recording sessions constituting the dataset.

2.1 Overview

The recording setup is as shown in Fig. 1. It comprises an RGB-D camera (a Microsoft Kinect), an HD camera, an ensemble of 5 LEDs located within the field of view of both cameras, a 24" flat computer screen and a 4cm diameter ball, which was used as a visual target for some of the recordings. The characteristics and purpose or function of each element are described as follows:

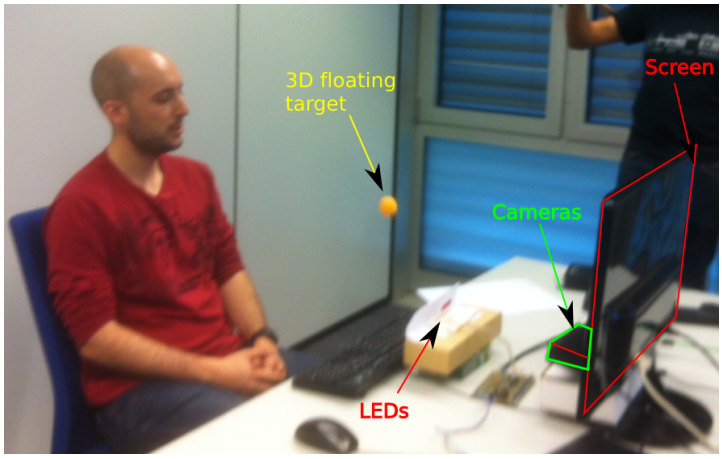


Figure 1: Recording setup

- Microsoft Kinect: this consumer device provides standard video (RGB) and Depth video streams, both at VGA resolution (640×480) and at a 30 frame-per-second acquisition rate.
- HD camera: the Kinect was designed with a large field of view for full body capture, which imposes less restriction on user mobility but is problematic for eye tracking based on VGA resolution. Therefore, we also recorded the scene with a full resolution HD camera (1920x1080) at 25fps. This camera was positioned as close as possible to the Kinect sensor.
- LEDs: we placed 5 LEDs visible by both cameras. The purpose is to synchronize the RGB-D and HD streams using the code displayed by the LEDs (see Section 3.4)
- Flat screen: we used a 24" computer screen to display a visual target (see Section 2.2.1). The effective screen resolution was of 1340×740.
- Small ball: we used a 4cm diameter ball as a visual target with a double purpose: to serve as a visual target in a 3D environment and, to be discriminative in both RGB and depth data such that its 3D position could be precisely tracked (see Section 3.6)

As shown in Fig. 1, the cameras are right below the computer screen, such that the eyes of the participant are observed from below minimizing eyelids occlusions. At each recording, a participant was requested to sit in front of the setup, within the field of view of the cameras, and a distance as needed according to the type of visual target (see Section. 2.2.1). Instructions were then given to gaze at the specified visual target during the recording time. No further requirements were given in terms of speaking or not, type of facial expressions, etc.

We now describe the different recording sessions that were made using this set-up.

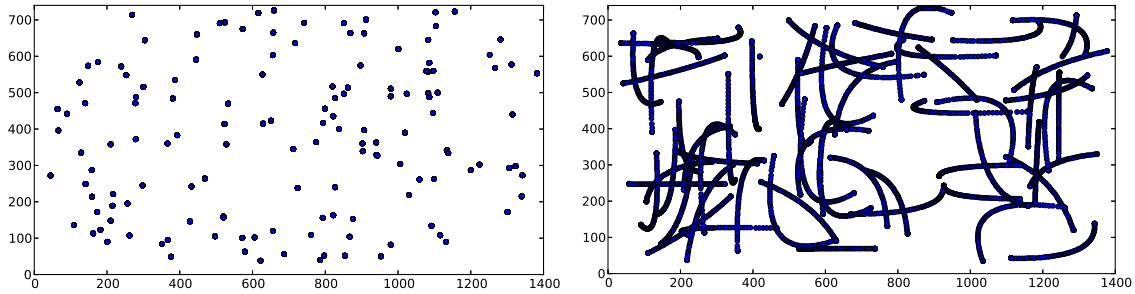
2.2 Recording sessions

In order to evaluate different aspects of gaze estimation algorithms, we designed a set of recording sessions, each one characterized by a combination of four main variables that can affect gaze estimation accuracy and that we describe below: visual target, head pose activity, participant and ambient conditions.

2.2.1 Visual target

The visual target is the object which the participant was requested to gaze at during the recording. In order to be representative of different applications, we included the following cases:

- *Discrete screen target (DS)*: In this case, a small circle was drawn at random locations in the computer screen. The screen coordinates were drawn from a uniform distribution and changed every 1.1 seconds. See Fig. 2a for an illustration.



(a) (b)

Figure 2: Example of screen coordinates for a session using: a) Discrete screen target ; b) Continuous screen target.

- *Continuous screen target (CS)*: The objective of using this target was to have examples of smoother gaze movements. Therefore, similarly to the DS case, a small circle was drawn in the computer screen and programmed to move along a trajectory parameterized by a quadratic Bézier curve. The control points of this curve were drawn from a uniform distribution defined within a smaller window of the screen and whose position were drawn randomly. A new trajectory was redefined every 2 seconds. See Fig. 2b for an example.
- *3D floating target (FT)*: This corresponds to a ball with a 4cm diameter hanging from a thin thread attached to a stick and that was moved within a 3D region between the camera and the participant. In contrast to the screen target, the participant was at a larger distance to the camera to allow for sufficient space for the target mobility. This object is discriminative in both color and depth, allowing to retrieve its position automatically (see Section 3.6).

For the screen based targets, the participants were at a distance of approximately 80-90cm from the recording sensor. For the 3D floating target case, the distance to the recording sensor was around 1.2m.

2.2.2 Head pose activity

In order to evaluate methods in terms of robustness to head pose variations, we requested the participant to keep gazing at the visual target while performing one of the two following head pose activities:

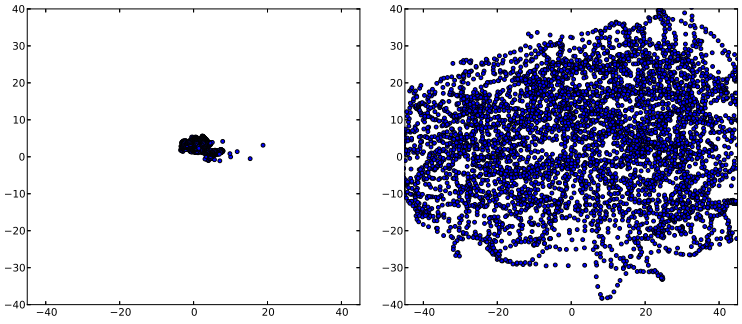
- *Static (S)*: participants were asked to keep an approximately static head pose facing towards the screen, thus being visible by the cameras. Fig. 3a shows examples of distributions over the head pitch and yaw angles recorded in this case.
- *Moving (M)*: participants were told to perform head movements, mostly in terms of rotations, in order to introduce head pose variations. In Fig. 3b we show examples of distributions over the head pitch and yaw angles for this case. As it can be observed, the recorded head poses are rich in terms of variations.

2.2.3 Participants

Our dataset was recorded for 16 different people: 12 males and 4 females. Each participant is assigned an ID from 1 to 16.

2.2.4 Ambient conditions

For participant 12, 13 and 14, some sessions were recorded twice, in different conditions: different day, illumination and distance to the camera. We denote the two possible conditions as A or B.



(a) (b)

Figure 3: Frame by frame head pose observations for session 15-B-FT-H, where H corresponds to the head pose activity, either a) Static or b) Moving. In both cases, the euler angles of the head pose are shown: yaw in the x axis and pitch in the y axis.

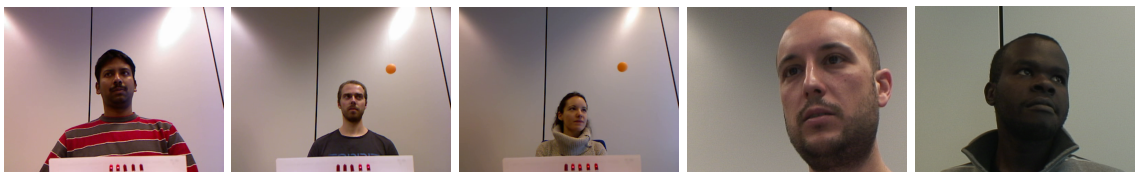
2.2.5 Sessions summary

In total we recorded 94 sessions. Each session will be denoted by the string "P-C-T-H" which refers to the participant id (1-16), the recording conditions C=(A or B), the used target T=(DS, CS or FT) and the head pose H=(S or M) respectively. Examples of the recordings can be seen in Fig. 4.

Each session in conditions "A" correspond to 2.5 minutes of recording time, whereas the sessions recorded in conditions B last approximately 3 minutes each. This corresponds to more than 4 hours of data. We summarize all recorded data in Table 1.

Table 1: Summary of the recorded sessions.

Participants	Recorded sessions (the participant index is implicit)
1-11	A-DS-S; A-DS-M; A-CS-S; A-CS-M; A-FT-S; A-FT-M
12-13	B-FT-S; B-FT-M
14-16	A-DS-S; A-DS-M; A-CS-S; A-CS-M; A-FT-S; A-FT-M B-FT-S; B-FT-M



(a) (b) (c) (d) (e)

Figure 4: Examples of the recorded data using: a-c) the RGB-D camera; and d-e) the HD camera, for which the images were cropped to a size of 640 × 480 for display comparison with the VGA resolution data. In these examples the given participant is: a,d) gazing at the screen target with a static head pose; b) gazing at the floating target with a static head pose; c,e) gazing at the floating target while moving the head.

3 Data processing

Besides the raw data itself that we described in Section 2, we also provided additional information that is essential for deriving ground truth measures for evaluation or simply useful to exploit the dataset and run experiments. It comprises the camera(s) calibration information, camera-screen calibration, synchronization, the 3D head pose, the approximate 3D location of the eyes frame, the frame by frame 3D location of the visual target and manual annotations.

A visualization of the provided meta-data is shown in Fig. 5. Below we describe the procedure we employed to estimate these parameters.

3.1 World coordinates system definition

To standarize the definition of all 3D variables in the data, we have defined a common world coordinate system (**WCS**), in which the variables refer to *meters*. It was defined with a fix relative position to the RGB camera of the Kinect, such that the participant is near the $[0,0,0]^\top$ (roughly) and the axis are consistent with the OpenGL standard. If $\mathbf{p}_\kappa \in \mathbb{R}^3$ is a point defined w.r.t. the coordinate system of the Kinect RGB camera, then we have defined the **WCS** such that the equivalent point $\mathbf{p}_W$, w.r.t. the **WCS** is given by $\mathbf{p}_W = \mathbf{R}_W\mathbf{p}_\kappa + \mathbf{t}_W$ where:

$$\mathbf{R}_W = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \mathbf{t}_W = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{1}$$

3.2 RGB-D sensor intrinsic calibration

The Kinect sensor was calibrated using the toolbox from Herrera et al. [22]. This procedure generates the intrinsic parameters for both the depth and RGB cameras together with the extrinsic parameters, i.e the 3D pose between both sensors. In addition it estimates the mapping from depth disparity to actual depth, including a correction for non linear distortions of the depth map.

Please refer to [22] to interpret the RGB-D data, in particular the disparity to depth transformation. We pre-processed the RGB stream to provide a video with corrected non-linear distortions. The provided camera poses are given with respect to the **WCS**.

3.3 3D screen calibration

We provide the calibration between the world coordinate system (3D) and the 2D screen coordinates, such that it is possible to refer a point from one system to the other. The mapping of a point $\mathbf{p} \in \mathbb{R}^3$, defined in the 3D world coordinate system, to the screen coordinates $\mathbf{s} \in \mathbb{R}^2$ is defined in Eq. 2.

$$\mathbf{s} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \end{bmatrix} (\mathbf{R}_\mathbf{s}\mathbf{p} + \mathbf{t}_\mathbf{s}), \tag{2}$$

where a 3D coordinate system $\mathcal{S} = \{\mathbf{R}_\mathbf{s}, \mathbf{t}_\mathbf{s}\}$ has been defined at the coordinates $(0,0)$ of the screen. The values k_x and k_y denote the pixels per *meter*[1] constants along the x and y coordinates respectively. Notice Eq. 2 assumes that $\mathbf{p}$ already lays in the screen plane by ignoring the z component once it is referred to $\mathcal{S}$. The inverse transformation is straightforward: a 3D point $[\mathbf{s}^x/k_x, \mathbf{s}^y/k_y, 0]^\top$ is defined and transformed by $\mathcal{S}^{-1}$.

To infer the parameters of the transformation, i.e. $\mathbf{R}_\mathbf{s}$, $\mathbf{t}_\mathbf{s}$, k_x and k_y we designed a mirror technique: a mirror was located in front of the RGB-D camera, whose 3D plane was found by leveraging on markers observable in the depth map. The mirror was positioned such that the screen was visible by the RGB-D sensor.

We then displayed a colored discriminative target in the screen at coordinates $\mathbf{s}$, and whose reflection by the mirror was visible to the RGB-D sensor. The observed (virtual) target's position is found automatically from color likelihood estimates. Its 3D position is then retrieved from depth

[1] To meters, rather than millimeters, as mentioned in [21]

data. This virtual 3D point is then transformed back to the world coordinate system based on the mirror plane and the law of reflection, to find the true point $\mathbf{p} \in \mathbb{R}^3$.

This process is used to collect a set of pairs $\{(\mathbf{s}, \mathbf{p})\}$. Then we constructed a linear system of equations from Eq. 2 such that the transformation parameters are obtained as the least squares solution from an overdetermined linear system of equations.

Important notice: The parameter values in the *screen_calibration.txt* file (see Section 4.1.1) do **not** correspond to the parameters defined in Eq. 2. Let $\hat{\mathbf{R}}_s$, $\hat{\mathbf{t}}_s$, $\hat{k}_x$ and $\hat{k}_y$ be the parameters provided in the file, then the parameters to be used in Eq. 2 are obtained as follows:

$$\mathbf{R_s} = \hat{\mathbf{R}}_s^\top \mathbf{R}_W^\top; \mathbf{t_s} = -\hat{\mathbf{R}}_s^\top \hat{\mathbf{t}}_s - \hat{\mathbf{R}}_s^\top \mathbf{R}_W^\top \mathbf{t}_W; k_x = \frac{1}{\hat{k}_x}; k_y = \frac{1}{\hat{k}_y} \tag{3}$$

3.4 RGB-D and HD camera synchrony and calibration

In addition to the RGB camera from the Kinect, an HD camera was used to record the scene. The purpose of this is to be able to develop and evaluate algorithms using high resolution data.

In order to use HD data it is necessary to have synchronization with the RGB-D video stream. To be able to achieve synchrony, we used a set of 5 LEDs which were activated in the order determined by the binary Gray Code, such that only one LED turns on or off at each transition. These LEDs were within the field of view of both cameras, and the goal was to post-process the data by aligning the code observed in both cameras. Retrieving the observed code was done by using an Hidden Markov Model (HMM), with transition probabilities according to the Gray Code, and emission probabilities according to noisy visual observations.

Stereo calibration between the two cameras is also desired, as for example, to use the HD video with depth data. To this end we used the standard stereo calibration procedure using a chessboard pattern for cross features. The output from this method was the HD camera pose $\mathcal{P}_{HD} = \{\mathbf{R}_{HD}, \mathbf{t}_{HD}\}$, provided w.r.t. the **WCS**, and the intrinsic parameters $\mathcal{I}_{HD}$ from the pin-hole model. In this manner, a point $\mathbf{p}^H \in \mathbb{R}^3$ defined in the HD camera coordinate system is transformed as $\mathbf{p}^W = \mathbf{R}_{HD}\mathbf{p}^H + \mathbf{t}_{HD}$, where $\mathbf{p}^W$ is the point referred to the **WCS**.

3.5 Head pose and eyes tracking

For each participant we created a 3D mesh corresponding to his/hers specific facial shape. This is done by fitting a 3D Morphable Model [23] to depth data using the method described in [24].

Provided the facial template, we tracked the 3D head pose using the Iterative Closest Points (ICP) algorithm. The initialization for ICP corresponds to the estimate from the previous frame, while the overall initialization is obtained from a frontal face detector [25]. The result is the estimated head pose for each frame, given as $\mathbf{p}_t = \{\mathbf{R}_t, \mathbf{t}_t\}$ of a 3D rotation and translation (w.r.t. the **WCS**).

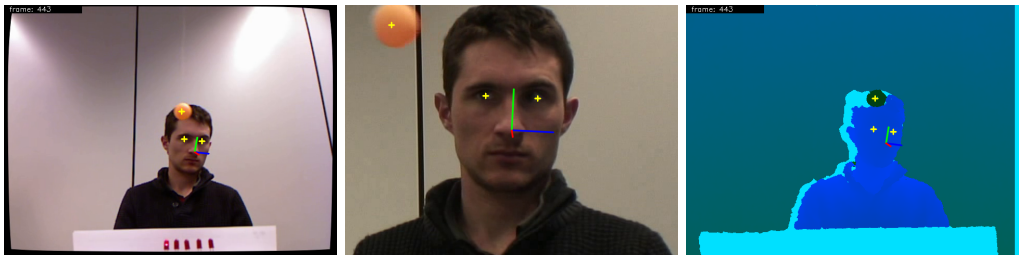
From the 3DMM an approximate location of the eyeballs is predefined and denoted as $\tilde{\mathbf{o}}$. Notice $\tilde{\mathbf{o}}$ is different for each person. In this manner, the 3D eyeball location at time t is given by:

$$\hat{\mathbf{o}}_t = \mathbf{R}_t\tilde{\mathbf{o}} + \mathbf{t}_t \tag{4}$$

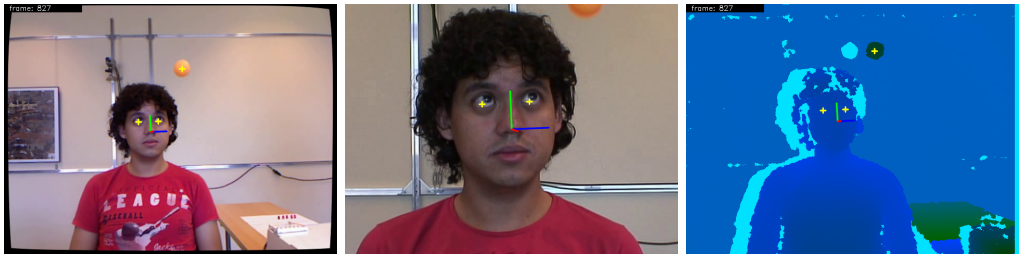
3.6 Floating target tracking

For the recording sessions using a ball as a visual target, we provide the 3D center of the ball at every time step t, denoted as $\mathbf{b}_t \in \mathbb{R}^3$ and defined w.r.t. the **WCS**.

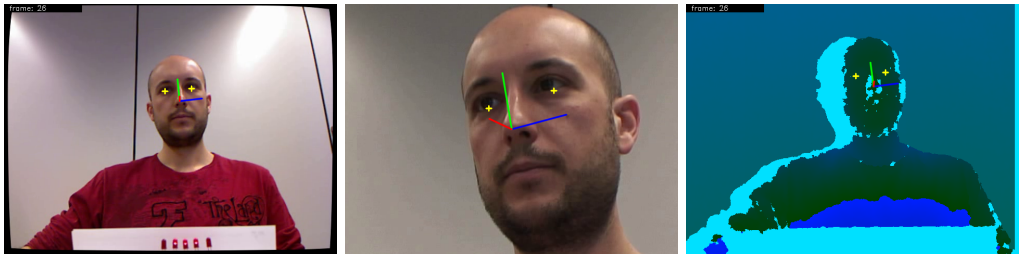
This value was computed as follows: the possible locations are first reduced by depth thresholding; then, the 2D point with the maximum color likelihood is selected as the location of the ball target, where the color distribution of the target was learned from labeling the ball location in one or two images. If the likelihood, within the region of the predefined size of the ball is smaller than a threshold, then the candidate is discarded. Once found, a template 3D mesh, with the size and shape of the target, is rigidly aligned to depth data using ICP. The center of the registered template, i.e. $\mathbf{b}_t$, is used as the gazed visual target 3D position.



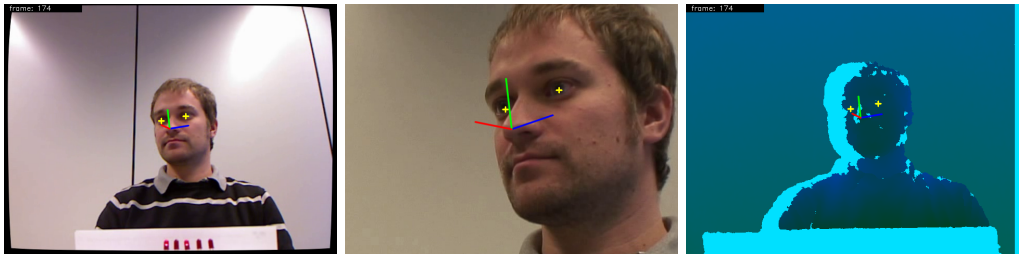
(a)



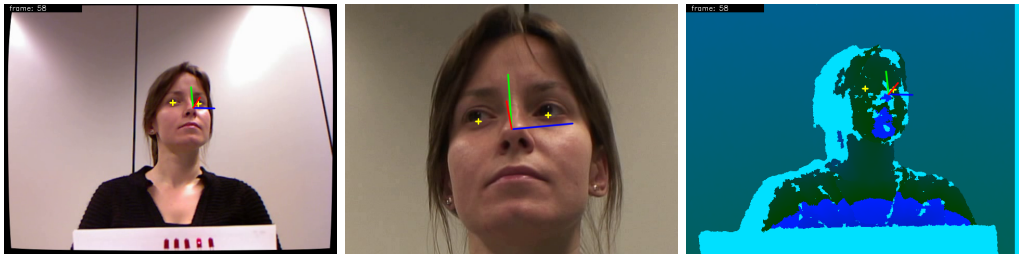
(b)



(c)



(d)



(e)

Figure 5: Examples of the processed data with head pose, eyes and floating target tracking. For each example we show, from left to right, the Kinect RGB frame, the HD frame (a 640×480 cropped region) and the depth frame encoded as an RGB image (as described in Section 4.1.1): a): frame 443 for session 1_A_FT_M; b): frame 827 for session "13_B_FT_S"; c): frame 26 for session 3_A_DS_S; d): frame 174 for session 15_A_CS_M; and e): frame 58 for session 8_A_CS_M.

4 Organization and files

In this section we describe the content of each of the folders as can be retrieved from the distributed database `www.idiap.ch/dataset/eyediap`.

4.1 "Data" folder

This folder contains the 94 recording sessions. Each subfolder is named according to its description, as summarized in Subsection 2.2.5 and each contain a set of files which are described in the following section.

4.1.1 Files per session

In this section we describe the files associated to each session and their interpretation:

- **rgb_vga.mov**: The 640×480 RGB video captured by the Microsoft Kinect device. It has been encoded using MPEG-4. The compression parameters were set to high-quality to reduce to a minimum the information loss.
- **depth.mov**: The 640×480 depth video captured by the Microsoft Kinect device (as captured using the *OpenKinect* library [2]). The Microsoft Kinect provides a depth map where depth values are given as 11 bits integers. Notice the depth is indeed encoded as disparity values. To transform to actual depth (in meters) it is necessary to use the depth camera calibration parameters (see other file descriptions below).

  We encoded the depth map as an RGB image as follows: the 11bits for a given depth value were divided into two bytes, the first 8 less significant bits (LSB) were assigned to the B channel of the RGB image, while the remaining 3 MSB (most significant bits) are taken as a byte, shifted left by 5 bits and then assigned to the G channel. The depth map value is recovered from the corresponding RGB image as follows:

  ```
  unsigned short depth = ((unsigned short) G )<< 3 +(unsigned short)B; ,
  ```

  when using C notation. The provided video was encoded using Zlib to achieve lossless depth compression.
- **rgb_hd.mov**: The 1920×1080 video captured by a high-resolution camera, originally at 25fps. Using the led-based synchronization (see Section 3.4) we generated the provided video which is fully synchronized with the Kinect RGB data (*rgb_vga.mov*) at 30fps.
- **head_pose.txt**: The frame-by-frame head pose tracking states. Obtained from the method described in Section 3.5.
- **eyes_tracking.txt**: The frame-by-frame eyes tracking states. The eyeballs 3D location was derived directly from the head pose track as shown in Eq. 4 whereas their 2D location (with respect to each camera) was obtained by projecting the 3D point into all cameras using the camera calibration parameters.

  Note: the point $\tilde{\mathbf{o}}$ was defined as the *approximate* 3D eyeball center with respect to the head coordinate system. As $\tilde{\mathbf{o}}$ does not lie in the eyeball surface, the 2D projections seem shifted from the (2D) eyes center for not frontal head poses.
- **ball_tracking.txt**: The ball tracking states (if relevant for the given session). The 3D position was obtained using the method described in Section 3.6, while the corresponding 2D position was obtained by projecting the 3D point into all cameras using the camera calibration parameters.

[2] `http://openkinect.org/`

- **screen_coordinates.txt**: The frame-by-frame screen target coordinates (if relevant for the given session). The 2D coordinates correspond to the screen target location as shown to the participant at the given time instant. The corresponding 3D position was derived from the screen-camera calibration (see Section 3.3). **Important notice:** the screen target 3D coordinates provided in this file are actually given with respect to the RGB camera coordinate system. To transform these values to the world coordinate system please apply the transformation defined in Eq. 1, i.e. $\mathbf{p}_W = \mathbf{R}_W \mathbf{p}_{file} + \mathbf{t}_W$.

- **calibration_rgb_vga.txt**: The calibration parameters for the Kinect's RGB camera. These are defined according to the pin-hole camera model in addition to the extrinsic camera parameters (camera 3D pose with respect to the world coordinate system).

- **calibration_rgb_hd.txt**: The calibration parameters for the HD camera. These are interpreted in the same way as for the "rgb_vga" camera.

- **calibration_depth.txt**: The calibration parameters for the Kinect's depth camera. These parameters are to be interpreted in a similar way to the "rgb_vga" camera, but it also includes the mapping parameters between depth map values to actual depth measurements (in meters). The depth mapping is described in [22].

For each tracking file[3] (e.g. eyes_tracking.txt, ball_tracking.txt, ...) the meaning of each value on a line of the file is defined in their respective "header" line at the beginning of the file (see also Section 4.4.1). A row full of zeros (except for the frame index) means the tracking states are not available for the given frame. All 3D parameters are referred to the world coordinate system.

Notice that we have drawn in each video frame its index value at the top-left corner of the frame. This index should coincide with the frame index found in the tracking files and across the videos when these are read simultaneously. This is intended for the user to verify the data being read is correctly synchronized.

4.2 "Metadata" folder

This folder contains a set of parameters which are constant throughout the database. These are the screen calibration parameters (see Section 3.3 in order to find the interpretation of the given values) and the *approximate* eyeballs centers for the left ($\tilde{\mathbf{o}}_{left}$) and right ($\tilde{\mathbf{o}}_{right}$) eye of each participant in the database (there is one file per participant).

4.3 "Example" folder

This folder contains the 3D gaze tracking results for the method proposed in [24] using the training and test sets described in [21]. These results are here provided as an example of usage of the database. A given algorithm should generate a similar output if the goal is to predict the 3D gaze direction, described as a pair of 3D rays describing the line of sight (LoS) of each eye.

The actual evaluation should consider a set not overlapping with the training set. For an example of the evaluation, please refer to Section 4.4.2

4.4 "Scripts" folder

In this folders we provide two main scripts described as follows.

4.4.1 visualize_session.py

This script is used to visualize the data of a given session. We assume python, with the numpy and OpenCV packages are available. To execute it, please run:

```
>> python visualize_session.py session_id
```

[3] and indeed for most files.

Where "session_id" is an integer referring to the session index to visualize. By inspection of the script, it can be observed how to interpret the diverse parameters' files provided along with the raw data. The images from Fig. 4 were obtained from the output of this script in the relevant sessions.

4.4.2 compute_etra_results.py

This script is used to recompute all results reported in [21]. The gaze tracking results are provided with the database within the "Example" folder, as described in Section 4.3. To execute it, please run:

```
>> python compute_etra_results.py
```

We provide this script as an example of how to interpret the output of a gaze tracker and to evaluate its performance by comparisons to the ground truth data. In addition, this provides a clear example on the separations between the training, test and evaluation sets.

Note: there is a slight difference between the output generated by this script and the results shown in [21]. This is due to minor differences between the definition of the evaluation set used in [21] and the exact -time- boundaries of the different sets. Nevertheless, notice the difference to the reported results is minimal.

5 Benchmarks

In this section we describe different evaluation benchmarks for the comparison of gaze estimation methods. We start by providing the framework used as evaluation protocol, including the performance measures, and then list a set of benchmarks used to evaluate the accuracy of a gaze estimation algorithm and its robustness to different variants such as head pose, ambient conditions, etc.

5.1 Evaluation protocol and measures

The goal of this section is to introduce notations to describe an experiment using the EYEDIAP database. We here define what is understood as a gaze estimation algorithm, followed by the definitions of train, test and evaluation sets, and finally define the different performance measures.

5.1.1 Gaze estimation algorithm

A gaze estimation algorithm is denoted as a function $\mathcal{G}$ which, provided a training set $\mathcal{V}$ and test data $\mathcal{T} = \{\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_t, \ldots, \mathbf{I}_T\}$, outputs a set of gaze estimates $\mathcal{G} = \{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_t, \ldots, \mathbf{g}_T\}$ with one to one correspondence to the elements in $\mathcal{T}$. This is shown in Eq. 5:

$$\mathcal{G} = \mathcal{H}(\mathcal{T}|\mathcal{V}) \tag{5}$$

The index t usually represent consecutive frames, such as for algorithms that rely on temporal information, but this is not assumed here. We thus consider methods estimating gaze from a single image as well.

The output of the gaze estimation method depends on the application. Here we consider two very common cases:

- *3D gaze ray*: The output of the gaze estimation algorithm is a 3D ray, known as the 3D line-of-sight (LoS). Therefore, it includes the two following elements: the origin of the gaze ray $\mathbf{o} \in \mathbb{R}^3$ and the *unitary* vector $\mathbf{v} \in \mathbb{R}^3$, such that $\mathbf{g} = \{\mathbf{o}, \mathbf{v}\}$.
- *Screen coordinates*: The output of the gaze algorithm is measured in computer screen coordinates $\mathbf{g} = \{\mathbf{s}\}$, such that $\mathbf{s} \in \mathbb{R}^2$ (pixels), as in the case of screen based applications for HCI.

Note that provided the screen calibration information it is possible to infer screen coordinates from the 3D gaze ray by first computing its intersection to the screen plane and then using Eq. 2 to transform from the 3D intersection point to screen coordinates. Alternatively, from the gazed screen coordinates and head pose, we can compute the equivalent 3D LoS. In addition, a given algorithm may or not output the gaze estimation for each eye separately.

5.1.2 Training data

The training data consist of pairs of data (images) and associated ground truth information such that the training set is defined as $\mathcal{V} = \{(\hat{\mathbf{I}}, \hat{\mathbf{g}})_{\hat{t}}, \hat{t} = 1, \ldots, N\}$. Here we consider different ways to collect these samples:

- *Temporal*: the training data $\mathcal{V}$ is assumed to be temporal, and often corresponds to a section of a recording session.
- *Structured*: the training data is collected in a structured manner in order to fulfill a specific requirement of the gaze estimation algorithm. This can be, for instance, to obtain a predefined number of points in a screen with specific $\hat{\mathbf{g}}$ values.

Notice that it is possible to build a structured training set from a temporal one assuming there is sufficient data in the latter. For the rest of the discussion, it is therefore assumed a training set is always provided as temporal. It is up to the user to decide how to use the given samples.

As ground truth data, we define the two types associated to the considered visual target:

- *3D visual target*: In this case $\hat{\mathbf{g}} := \hat{\mathbf{p}}$ where $\hat{\mathbf{p}} \in \mathbb{R}^3$ corresponds to the 3D location of the gazed point.
- *Screen coordinates*: In this case $\hat{\mathbf{g}} := \hat{\mathbf{s}}$ where $\hat{\mathbf{s}} \in \mathbb{R}^2$ corresponds to the 2D screen coordinates the participant is gazing at. Notice the corresponding $\hat{\mathbf{p}}$ can be computed from $\hat{\mathbf{s}}$ as described in Section 3.3.

5.1.3 Test data

The test data $\mathcal{T}$ here will be assumed to be, similarly to the training case, a temporal section of a recording session. The corresponding range of frame indices within the recording session is $[t_0, t_1]$.

5.1.4 Evaluation data

Given the test data, we define an evaluation set $\hat{\mathcal{E}}$ that comprises the frames on which the performance of an algorithm is computed. This allows to remove the few frames for which, either the data or the ground truth is corrupted, as listed below:

- *Blinking and distractions*: these are samples which were manually labeled as outliers, because the person is blinking or is not gazing the visual target.
- *Extreme head poses*: in such cases the visibility of the eyes is compromised, e.g. an eye is occluded by the nose given a extreme yaw. These samples can be automatically removed from the evaluation by thresholding the head pose euler angles.
- *Train and test set intersection*: in the few cases when the test set $\mathcal{T}$ video range overlaps the range of the training set $\mathcal{V}$, then the samples which were used to construct $\mathcal{V}$ have to be taken out from $\mathcal{T}$ when defining $\hat{\mathcal{E}}$.

Interpolation based methods. Many methods, and especially appearance based approaches, are capable of estimating gaze only within the convex hull of the training data gaze directions as these methods are based on interpolation. In our dataset, such condition can not be always guaranteed in the data collection, particularly for the floating target cases or for the screen target cases in which there are large head pose variations.

To allow fair comparison for such methods, when defining the evaluation set, the user can also discard test samples for which the ground truth is outside the convex hull of the gaze directions of the training data. Even though in real conditions the gaze estimation algorithm does not have access to the test data ground truth, we can assume the system has control on the methodology used to collect the training data. Nevertheless, the users of the database need to report the used evaluation set, such that objective comparisons can later be made against other methods.

Note that comparing the performance of methods on both the convex hull and in the full range will be interesting to distinguish algorithms which are capable of extrapolating gaze estimation (typically, those that rely on a geometrical model) and those that can not.

Finally, note also that the interpolation issue might be true for other variables as well, such as head pose, but here we only considered gaze directions.

5.1.5 Performance measures

In this section we define a set of performance measures to compare different algorithms. For an index t in the evaluation set $\hat{\mathcal{E}}$, with estimated gaze direction $(\mathbf{o}_t, \mathbf{v}_t)$ or screen coordinates $\mathbf{s}_t$, we considered the following error measures:

- *3D distance error* ${\epsilon^d}_t$*:*
  This error is defined for the 3D gaze estimation tasks. It conveys how close the estimated 3D gaze ray passes by the visual target 3D position $\hat{\mathbf{p}}_t$, as provided by Eq. 6.

$$ {\epsilon^d}_t = \min_v \|(\mathbf{o}_t + v\mathbf{v}_t) - \hat{\mathbf{p}}_t\|_2 , \tag{6} $$

  where the gaze ray has been defined in parametric form using $v \in [0, \infty[$ and $\|.\|_2$ defines the euclidean norm in the 3D space, or the vector's magnitude.

- *Angular error* ${\epsilon^\circ}_t$*:*
  This is a normalization alternative to ${\epsilon^d}_t$, where we measure the error in terms of directional error expressed in degrees:

$$ {\epsilon^\circ}_t = \arcsin\left(\frac{{\epsilon^d}_t}{\|\hat{\mathbf{p}}_t - \hat{\mathbf{o}}_t\|_2}\right), \tag{7} $$

  where we used the provided eye location $\hat{\mathbf{o}}_t$ for the purpose of standard comparisons. Alternatively, the angular gaze estimation error can be computed also as:

$$ {\epsilon^\circ}_t = \arccos\left(\mathbf{v}_t \cdot \hat{\mathbf{v}}_t\right), \tag{8} $$

  where $\hat{\mathbf{v}}_t$ denotes the ground truth 3D gaze vector, a unitary vector pointing from $\hat{\mathbf{o}}_t$ to $\hat{\mathbf{p}}_t$

- *Screen pixel error* ${\epsilon^{\mathbf{s}}}_t$*:* This error is defined for the gazed screen pixel coordinates prediction task and it is given in Eq. 9.

$$ {\epsilon^{\mathbf{s}}}_t = \|\mathbf{s}_t - \hat{\mathbf{s}}_t\|_2 \tag{9} $$

  Notice that, provided the screen-camera calibration (cf. Section 3.3), it is possible to compute an angular error from screen coordinates predictions by using as gaze ray origin the provided eye location $\hat{\mathbf{o}}_t$ (cf. Section 3.5).

Using the above errors, we can then compute statistics (usually the mean) on the evaluation set $\hat{\mathcal{E}}$. The default ones are the mean distance error $\epsilon^d = \frac{1}{|\hat{\mathcal{E}}|}\sum_{t\in\hat{\mathcal{E}}} {\epsilon^d}_t$, the mean angular error $\epsilon^\circ = \frac{1}{|\hat{\mathcal{E}}|}\sum_{t\in\hat{\mathcal{E}}} {\epsilon^\circ}_t$, and the mean screen pixel error $\epsilon^{\mathbf{s}} = \frac{1}{|\hat{\mathcal{E}}|}\sum_{t\in\hat{\mathcal{E}}} {\epsilon^{\mathbf{s}}}_t$.

Sensitivity: in addition to the prediction accuracy, we can also report additional performance measure, like the sensitivity which can be used to compare two experiments for which the obtained corresponding errors are $\epsilon^{\circ 1}$ for the baseline conditions, and $\epsilon^{\circ 2}$ for a more challenging condition. This measure is defined as:

$$ \mathcal{R}(\epsilon^{\circ 2}|\epsilon^{\circ 1}) = \min\left(0, \frac{\epsilon^{\circ 2} - \epsilon^{\circ 1}}{\epsilon^{\circ 1}}\right), \tag{10} $$

where it is expected that the second experiment is more difficult than the first one (and thus normally ${\epsilon^\circ}_2 > {\epsilon^\circ}_1$). $\mathcal{R}$ is thus intended to measure the robustness of an algorithm. Ideally $\mathcal{R} \to 0$.

5.2 Predefined benchmarks

In order to allow comparisons between algorithms and their merit under different experimental conditions, we have defined a set of protocols that differ mainly in the data used from our database to define the train, test and evaluation sets[4].

Notice this dataset has two main types of visual targets: 3D floating target (FT) and screen target (CS or DS). Therefore, the evaluation protocols are defined independently of the visual target, which once specified, lead to the definition of the actual recording sessions to be used.

5.2.1 Benchmark 1: Gaze estimation accuracy

In this protocol, we evaluate the accuracy of a gaze estimation algorithm $\mathcal{H}$ under minimal variation of all parameters which are not gaze. Therefore, for a recording session Σ where the only variation is in the gaze direction itself (e.g. 1_A_DS_S), we define the training set $\mathcal{V}$ as the first temporal half of Σ and the test set $\mathcal{T}$ as the second temporal half of Σ, such that $\mathcal{V}$ and $\mathcal{T}$ do not overlap.

The goal then consist on obtaining the mean gaze angular error ϵ° from the defined sets $\mathcal{V}$, $\mathcal{T}$ and $\hat{\mathcal{E}}$, where $\hat{\mathcal{E}}$ is a subset of samples from $\mathcal{T}$, as discussed in Section 5.1.4. The relevant sessions depend on the visual target type as follows:

- *Screen target*: Includes all recording sessions with static head pose (S) and the screen target (either CS or DS). This makes 14 sessions in total (1 per participant), see Table 1.
- *3D floating target*: Includes all sessions with static head pose (S) and the 3D floating target (FT). Notice this case includes sessions from both ambient conditions A and B. This makes a total of 19 sessions.

The process of training, testing and evaluation is repeated for all relevant recording sessions and the mean angular error ϵ° averaged among all sessions is reported.

This is the methodology used in [21] for which the algorithm was evaluated using the floating target (FT) recording sessions and using the Kinect data only.

5.2.2 Benchmark 2: Head pose invariance

In this case the objective is to measure how much does the gaze estimation accuracy decays when the participant perform changes in head pose. Two experiments are conducted per participant k, where the relevant recording sessions are defined from the desired visual target:

- Experiment 1: In this experiment, an evaluation of the gaze estimation accuracy is conducted for a static (S) head pose, such that the obtained mean angular error is $\epsilon^{\circ S}$. We denote the used recording session as Σ_S. This step follow closely the procedure defined in Protocol 1.
- Experiment 2: In this case, we evaluate the $\mathcal{H}$ in the presence of head pose variations (M) obtaining a mean angular error of $\epsilon^{\circ M}$. Notice that, for a recording session Σ_S used for "Experiment 1", there is an equivalent recording session Σ_M with the same configuration (participant, ambient conditions and visual target) except that it includes the case of head pose variations (e.g. Σ_S ="1_A_DS_S" and Σ_M ="1_A_DS_M"). Then, for this experiment, let the training set $\mathcal{V}$ be the same used for Experiment 1, whereas the test set $\mathcal{T}$ is Σ_M. From the obtained $\epsilon^{\circ M}$ we can then compute the sensitivity as $\mathcal{R}(\epsilon^{\circ M}|\epsilon^{\circ S})$.

The errors $\epsilon^{\circ S}$ and $\epsilon^{\circ M}$ are computed, together with the $\mathcal{R}(\epsilon^{\circ M}|\epsilon^{\circ S})$ value for every pair (S and M) of relevant recording sessions (the user can decide to evaluate on only the floating visual target, the screen targets or both). As final result, the values $\epsilon^{\circ S}$, $\epsilon^{\circ M}$ and $\mathcal{R}(\epsilon^{\circ M}|\epsilon^{\circ S})$ are averaged among all experimental pair.

[4]Researchers are free to define their own evaluation protocol using the provided dataset, but we strongly encourage the use of the benchmarks defined here.

5.2.3 Benchmark 3: Person independence

In this benchmark the goal is to evaluate how well does a method $\mathcal{H}$ generalize to unseen users. The set of relevant sessions are the same as for Benchmark 1 (Gaze estimation accuracy) but we discard the sessions from ambient conditions B (to avoid training and evaluating on the same user). Notice also that we can do a separate experiment per type of visual target (CS, DS or FT). Then two experiments will be conducted for a participant k as follows:

- Experiment 1: This follow exactly the methodology defined for Benchmark 1. The output is the gaze estimation error $\epsilon^{\circ k}$ obtained when there is a person specific training and the evaluation is done within the same recording session Σ_k (the session for user k). Notice that, as in Benchmark 1, the sets $\mathcal{V}_k$ and $\mathcal{T}_k$ are disjoint.
- Experiment 2: This experiment follow a leave-one-person-out scheme. Therefore, for user k, we define the training set from the sets from all other users as $\mathcal{V} = \cup_{j \neq k} \mathcal{V}_j$. Notice $\mathcal{V}_j$ corresponding to the training data from session Σ_j, where all other parameters (head pose, ambient conditions and visual target) are the same for session Σ_k, except that it is the session corresponding to the participant j. The test set is the one specific to user k, i.e. the second half from session Σ_k. The resulting mean angular error is $\epsilon^{\circ \setminus k}$. From this we can compute the robustness of the method as $\mathcal{R}(\epsilon^{\circ \setminus k} | \epsilon^{\circ k})$.

Both experiments are conducted for all participants, the values $\epsilon^{\circ k}$, $\epsilon^{\circ \setminus k}$ and $\mathcal{R}(\epsilon^{\circ \setminus k} | \epsilon^{\circ k})$ are computed and their average are reported.

5.2.4 Benchmark 4: Ambient conditions invariance

Finally, in this case the goal is to study the generalization of a gaze estimation algorithm $\mathcal{H}$ to different ambient conditions. To this end, four experiments are conducted for participants 12, 13 and 14 for which recording sessions under different set-up and illumination conditions exist. Notice only the floating target data is available for this task. For each participant k, the experiments are:

- Experiment 1: let $\mathcal{V}$ be the first half of session k-A-FT-S and $\mathcal{T}$ be the second half of session k-A-FT-S. The obtained mean angular gaze estimation error is $\epsilon^{\circ A}$.
- Experiment 2: the first half of session k-A-FT-S is again used as training set $\mathcal{V}$, but now the test condition is changed by using the second half of session k-B-FT-S as test set $\mathcal{T}$. The obtained mean angular error is $\epsilon^{\circ B|A}$.
- Experiment 3: we conduct similar experiments, but now in the reverse order. That is, the first half of session k-B-FT-S is assigned to the training set $\mathcal{V}$, and the second half of session k-B-FT-S is assigned to the test set $\mathcal{T}$. The obtained mean angular error is $\epsilon^{\circ B}$.
- Experiment 4: the first half of session k-B-FT-S is assigned to $\mathcal{V}$ and the A-condition data is used for testing, i.e. the second temporal half of session k-A-FT-S is used as test data $\mathcal{T}$. The obtained mean angular error is $\epsilon^{\circ A|B}$.

For each participant $\epsilon^{\circ A}$, $\epsilon^{\circ B|A}$, $\epsilon^{\circ B}$ and $\epsilon^{\circ A|B}$ are computed, together with $\epsilon^{\circ} = (\epsilon^{\circ A} + \epsilon^{\circ B})/2$ and $\mathcal{R} = (\mathcal{R}(\epsilon^{\circ B|A} | \epsilon^{\circ A}) + \mathcal{R}(\epsilon^{\circ A|B} | \epsilon^{\circ B}))/2$. As final result the average of ϵ° and $\mathcal{R}$, among the 3 participants, is reported.

6 Conclusion

In this document we have presented an extended description of the EYEDIAP database, originally described in our paper [21]. This dataset was designed to provide a common framework for the training and evaluation of gaze estimation algorithms from remote RGB and RGB-D cameras.

The database description presented in this document includes the data collection methodology, the set-up and the sessions that were recorded. The recorded sessions were designed such that we systematically include and isolate most variables which affect gaze estimation algorithms.

The processing of the data has also been described to give a better understanding of the meta-data which is provided along with the raw data. In addition, we have also described the files composing the database to make easier their interpretation.

Finally, we have described in detail a set of bechmarks which we recommend to use when evaluating the performance of a gaze estimation algorithm such that, through standarization, we enable a direct comparison between diverse methods.

We believe this database is of high value to researchers as it will help to advance the development of gaze estimation technologies under less constrained conditions.

References

[1] S. Sheikhi and J.-M. Odobez, "Investigating the midline effect for visual focus of attention recognition," in *ACM Int Conf. on Multimodal Interaction (ICMI)*, oct 2012.

[2] S. Ba and J.-M. Odobez, "Multi-party focus of attention recognition in meetings from head pose and multimodal contextual cues," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, (Las-Vegas), march 2008.

[3] D. Jayagopi, S. Ba, J.-M. Odobez, and D. Gatica-Perez, "Predicting two facets of social verticality in meetings from five-minute time slices and nonverbal cues," in *Proc. Int. Conf. on Multimodal Interfaces (ICMI), Special Session on Social Signal Processing*, (Chania), Oct. 2008.

[4] D. W. Hansen and Q. Ji, "In the eye of the beholder: a survey of models for eyes and gaze.," *IEEE Trans. on Patt. Anal. and Machine Intelligence*, vol. 32, pp. 478–500, Mar. 2010.

[5] P. Majaranta, H. Aoki, M. Donegan, D. W. Hansen, and J. P. Hansen, *Gaze Interaction and Applications of Eye Tracking: Advances in Assistive Technologies*. Hershey, PA: Information Science Reference - Imprint of: IGI Publishing, 1st ed., 2011.

[6] S. Baluja and D. Pomerleau, "Non-Intrusive Gaze Tracking Using Artificial Neural Networks," tech. rep., CMU, 1994.

[7] T. Moriyama and J. Cohn, "Meticulously detailed eye model and its application to analysis of facial image," *Int. Conf. on Systems, Man and Cybernetics*, vol. 1, pp. 629–634, 2004.

[8] E. D. Guestrin and M. Eizenman, "General theory of remote gaze estimation using the pupil center and corneal reflections.," *Trans. on bio-medical engineering*, June 2006.

[9] H. Yamazoe, A. Utsumi, T. Yonezawa, and S. Abe, "Remote gaze estimation with a single camera based on facial-feature tracking without special calibration actions," in *Proceedings of the 2008 symposium on Eye tracking research & applications*, vol. 1 of *ETRA '08*, (New York, NY, USA), pp. 245–250, ACM, 2008.

[10] Y. Sugano, Y. Matsushita, Y. Sato, and H. Koike, "An incremental learning method for unconstrained gaze estimation," in *ECCV*, pp. 656–667, Springer, 2008.

[11] T. Ishikawa, S. Baker, I. Matthews, and T. Kanade, "Passive Driver Gaze Tracking with Active Appearance Models," in *Proceedings of the 11th World Congress on Intelligent Transportation Systems*, pp. 1–12, Oct. 2004.

[12] K. A. Funes Mora and J.-M. Odobez, "Person Independent 3D Gaze Estimation From Remote RGB-D Cameras," in *International Conference on Image Processing*, Sept. 2013.

[13] H. Salam, R. Seguier, and N. Stoiber, "Integrating head pose to a 3d multi-texture approach for gaze detection," *International Journal of Multimedia & Its Applications*, vol. 5, pp. 1–22, Aug. 2013.

[14] D. Winfield and D. Parkhurst, "Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches," *Computer Vision and Pattern Recognition Workshops*, 2005.

[15] O. Williams, A. Blake, and R. Cipolla, "Sparse and semi-supervised visual mapping with the S3GP," in *Computer Vision and Pattern Recognition*, 2006.

[16] B. Noris, J. Keller, and A. Billard, "A wearable gaze tracking system for children in unconstrained environments," *Computer Vision and Image Understanding*, pp. 1–27, 2010.

[17] F. Lu, Y. Sugano, O. Takahiro, and Y. Sato, "Inferring Human Gaze from Appearance via Adaptive Linear Regression," in *ICCV*, (Barcelona, Spain), 2011.

[18] B. A. Smith, Q. Yin, S. K. Feiner, and S. K. Nayar, "Gaze Locking: Passive Eye Contact Detection for Human-object Interaction," in *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, (New York, NY, USA), pp. 271–280, ACM, 2013.

[19] T. Schneider, B. Schauerte, and R. Stiefelhagen, "Manifold Alignment for Person Independent Appearance-based Gaze Estimation," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR)*, (Stockholm, Sweden), IEEE, Aug. 2014.

[20] K. A. Funes Mora, L. S. Nguyen, D. Gatica-Perez, and J.-M. Odobez, "A Semi-Automated System for Accurate Gaze Coding in Natural Dyadic Interactions," in *Int Conf. on Multimodal Interaction*, (Sydney), Dec. 2013.

[21] K. A. Funes Mora, F. Monay, and J.-M. Odobez, "EYEDIAP: A Database for the Development and Evaluation of Gaze Estimation Algorithms from RGB and RGB-D Cameras," in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '14, (Safety Harbor, FL, USA), pp. 255–258, ACM, 2014.

[22] D. Herrera C., J. Kannala, and J. Heikkilä, "Joint Depth and Color Camera Calibration with Distortion Correction," *TPAMI*, vol. 34, no. 10, pp. 2058–2064, 2012.

[23] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter, "A 3D Face Model for Pose and Illumination Invariant Face Recognition," in *AVSS*, (Genova, Italy), 2009.

[24] K. A. Funes Mora and J.-M. Odobez, "Gaze Estimation From Multimodal Kinect Data," in *Computer Vision and Pattern Recognition Workshops*, pp. 25–30, June 2012.

[25] P. Viola and M. Jones, "Robust Real-time Object Detection," in *International Journal of Computer Vision*, 2001.