Gaze Recording with Tobii Eye Trackers: Usage of **screen based Tobii Eye Trackers** from within a 3rd party program

Tutorial

Simon Maurer - simon.maurer@humdek.unibe.ch
23rd January 2024

Contents

Chapter 1

Introduction

This documents describes how to use the Tobii Eye Tracker Spark and the Tobii Eye Tracker 4C to record gaze data of a subject and (optionally) feed the gaze data to the mouse device. This allows to use the eye tracker to control the mouse pointer position such that the mouse pointer is placed at the screen coordinates where the subject is gazing at. This documentation comes with a set of tools (executable .exe files) that provide this functionality as well as some auxiliary tools that help to calibrate and test the eye tracker.

The project aims at providing a set of executables which allow to use the Eye Trackers in conjunction with third party applications that allow to execute external programs. Specifically, ztree or openseame. The following set of executables are provided:

TobiiCalibrate.exe This program is a simple wrapper for the Tobii calibration tool. It launches the calibration GUI where the subject is led through the calibration process. The calibration data is stored in the current profile of the eye tracker engine. **Attention:** Make sure to to have the Tobii Pro Eye Tracker Manager of version 2.6 or later installed.

Gaze.exe This program uses the Tobii Pro SDK to extract the gaze position on the screen where the subject is looking at. The extracted data is recorded and stored to a file. Optionally, the mouse cursor position is updated to this position such that the mouse cursor is controlled by the gaze of the subject. Instead of using an eye tracker device it is also possible to simply log the mouse coordinates.

`Gaze.exe` runs infinitely until it is terminated by an external command. This should **not** be done with a forced kill (e.g. by executing the command `taskkill /F /IM Gaze.exe` or by killing the task with the task manager) because it prevents the program from terminating gracefully. This as several consequences:

- open files are not closed properly and the data stream is cut off. This can lead to corrupt files.
- if the feature of hiding the mouse pointer is used, the mouse will remain hidden.
- memory is not freed properly.

Instead the program `GazeControls.exe /command TERMINATE` should be used.

The application takes the following two input arguments:

subject A subject code which will be used to prefix the output files.

outputPath The location where the output files will be stored. Note that this can also be configured through the configuration option `DataLogPath` (see Chapter 4).

GazeControl.exe This program allows to interact with Gaze.exe. `GazeControl.exe` accepts the following optional arguments:

`/reset`
Allows to reset the relative timestamp of the gaze data.

`/trialId <ID>`
Sets a trial ID `<ID>` which will be added to each data sample in the output file. **Important**:** Make sure that only integer numbers are used as trial ID.

`/label <LABEL>`
Sets a custom label `<LABEL>` which will be added to each data sample in the output file. Any string is accepted here. Make sure to use quotations marks if the label contains any spaces.

`/command <COMMAND>`
A command allows to activate/deactivate features of `Gaze.exe`. The following commands `<COMMAND>` are supported:

`CUSTOM_CALIBRATE`
uses the Tobii Pro SDK and launches a custom calibration process which allows to calibrate the eye tracker without having to rely on the calibration software provided by Tobii.

`DRIFT_COMPENSATION`
launches a custom drift compensation process to compensate gaze drifts that may occur during experimentation.

`GAZE_RECORDING_DISABLE`
requests Gaze.exe to stop recording gaze data. Gaze.exe will continue to run (and update the mouse pointer if configured accordingly) but no longer store gaze data to the disk.

`GAZE_RECORDING_ENABLE`
requests Gaze.exe to start recording gaze data.

`LOAD`
shows a window with a spinner for a configurable amount of time.

`MOUSE_TRACKING_DISABLE`
requests Gaze.exe to stop updating the mouse pointer by the gaze position.

`MOUSE_TRACKING_ENABLE`
requests Gaze.exe to start updating the mouse pointer by the gaze position.

`RESET_DRIFT_COMPENSATION`
resets the drift compensation computed with the command `DRIFT_COMPENSATION`.

`TERMINATE`
requests Gaze.exe to close gracefully and logs these events to the log file.

`VALIDATE`
uses the Tobii Pro SDK Addon and launches a validation process.

Multiple arguments can be passed to the application but each argument can only be passed once. Passing an argument to an application can be done in command line or by crating a shortcut to the program. Corresponding shortcuts for all available `<COMMAND>`s are provided in the release package.

The application returns the following exit codes:

0: The command was executed successfully.

1: The connection to the named pipe failed and the command was not executed.

2: The command was successfully sent to `Gaze.exe` but failed there.

3: The command was successfully sent to `Gaze.exe` but no reply was received.

4: An unknown error occurred.

ShowMouse.exe This program allows to restore the standard mouse pointer. It might be useful if the program `Gaze.exe` crashes or is closed forcefully such that the mouse pointer is not restored after terminating. The subject might end up with a hidden mouse pointer. A good solution for such a case is to install a shortcut to `ShowMouse.exe` on the desktop in order to execute it with the keyboard.

1.1 Source Code

The necessary code for what is described in this document is hosted on GitHub[1].

Feel free to open new issues or create pull requests.

[1] https://github.com/humdek-unibe-ch/tbi-gaze

Chapter 2

Quick Start

Attention Configure the task manager to be always in the foreground (in the task manager enable "Options->Always on top").
Here is why: The application `Gaze.exe` may open windows that are put to the foreground in a very aggressive manner. This is done in order to cope with experimentation software that uses this same behaviour (e.g. Opensesame with psychopy or expyriment backend). If something goes wrong with '`Gaze.exe` the user could be locked out from the computer because a window keeps blocking access to the system. With the task manager set to "Always on top" there is always an easy way out.

In order to get started with Tobii eye tracking the following installation steps need to be performed:

- Install the Tobii Pro Eye Tracker Manager application.
- Setup the *Tobii Pro Spark* device with the following steps:
  - Connect the *Tobii Pro Spark* device to the computer. Make sure that the device is connected to a powered USB-type A port or, alternatively, use the provided USB-type A to USB-type C adapter and connect the device to a USB-type C port.
  - Launch the Tobii Pro Eye Tracker Manager application
  - Click on the button *Install New Eye Tracker*
  - Select the device *Tobii Pro Spark*
  - Click Install
  - Now the eye tracker device should appear in the list. Click on the device and setup the screen calibration.
  - Perform the device calibration and verify the gaze accuracy through the built-in gaze visualization
- Setup the *Tobii Eye Tracker 4C* device with the following steps:
  - Install the Tobii Experience software. This software comes with the device driver for the *Tobii Eye Tracker 4C*, however, it is only available as a beta version and is no longer maintained.
  - Connect the *Tobii Eye Tracker 4C* device to the computer.
  - Now the eye tracker device should appear in the list of available devices in the *Tobii Pro Eye TRacker Manager* application. Click on the device and setup the screen calibration.
  - Perform the device calibration and verify the gaze accuracy through the built-in gaze visualization
- Install the Gaze toolset provided by the TPF. To install the toolset download[1] the latest version (requires a subscription, see Section 1.1) and extract all files to an installation path of your choosing. The Gaze toolset installation path will be referred to as `<Gaze toolset path>` throughout this document.

[1] http://phhum-g111-nns.unibe.ch:10012/TBI/TBI-tobii_eye_tracker_gaze/tree/master/release

2.1 Quick Start with ztree

In order to get started with a quick experiment you need the following things (*server* refers to the machine running `ztree` and *client* refers to the machines running `zleaf`):

- a ztree installation with a server and one or more clients. To install `ztree` download the latest `ztree` version from the download secetion[2] (requires a license and a login, see Section 2.1.1) and extract the server file `ztree.exe` to the chosen installation path on the server and the client file `zleaf.exe` to the chosen installation path on each client. Throughout this documentation the installation paths of the `ztree` client and server will be referred to as `<zleaf path>` and `<ztree path>`, respectively.
- the ztree sample file. The file is located in `<Gaze toolset path>\sample\template.ztt`. Make sure to change the `Path` variable in `Background` of the sample file such that it points to `<Gaze toolset path>`.

The provided sample `ztree` program performs the following stages:

Initialise Eye Tracker Starts the Gaze.exe application to run in the background during the experiment. Note that a sleep of five seconds is introduced here to give the eye tracker time to initialize.

Calibrate Eye Tracker Runs the command `GazeControl.exe /command CUSTOM_CALIBRATE` to start the custom calibration.

Enable Mouse Tracking Runs the command `GazeControl.exe /command MOUSE_TRACKING_ENABLE`. The gaze of the subject is tracked and transformed to mouse coordinates which allows to control the mouse pointer with the gaze of the subject.

Terminate Runs the command `GazeControl.exe /command TERMINATE` to gracefully shutdown the gaze application and conclude this simple `ztree` program.

The behaviour of the Gaze toolset can be configured with a configuration file (see Chapter 4 for more details). A sample configuration file is provided in `<Gaze toolset path>\sample\config.json` that can be modified to suit different requirements. Copy the configuration file either to `<Gaze toolset path>` or `<zleaf path>` and modify it there.

2.1.1 Requesting Access to the Latest `ztree` Release

To download the latest release of the `ztree` software a login must be requested here[3]. Make a careful note of the terms and conditions.

2.2 Quick Start with opensesame

In order to get started with a quick experiment the following steps are required:

- Install the latest version of opensesame with your preferred installation method.
- As a starting point use the provided opensesame sample file. The file is located in `<Gaze toolset path>\sample\template.osexp`. Make sure to change the `gazeAppPath` variable in `gaze_initialisation` of the sample file such that it points to `<Gaze toolset path>`.

The provided sample `opensesame` program includes the following python scripts:

gaze_start Sets the `gazeAppPath` variable during prepare stage and then starts the gaze application. The application will run in the background during the experiment. Note that the `subject_nr` variable and the `experiment_path` variable are passed as arguments to the application. The former is used to prefix the output file names with the subject number and the latter is used as location to store the output files. Also, note that the log files are stored at the `gazeAppPath`. Further, note that a sleep of three seconds is introduced here to give the eye tracker time to initialize.

[2] https://www.uzh.ch/ztree/ssl-dir/index.php
[3] https://www.uzh.ch/ztree/ssl-dir/index.php?action=obtain

gaze_calibration Runs the command `GazeControl.exe /command CUSTOM_CALIBRATE` to start the custom calibration.

gaze_validation Runs the command `GazeControl.exe /command VALIDATE` to start the validation.

loop Loops over a table of trial IDs

- **gaze_drift_compensation** Runs the command `GazeControl.exe /command DRIFT_COMPENSATION` to start the drift compensation process.
- **set_trial_id** Runs the command `GazeControl.exe /command SET_TRIAL_ID /value <ID>` to communicate a trial ID to the gaze application. Note that `<ID>` is an ID extracted from the loop table.
- **gaze_recording_enable** Runs the command `GazeControl.exe /command GAZE_RECORDING_ENABLE` to start storing gaze data to the disk.
- **loop** Loops over a table of labels
  - **set_label** Runs the command `GazeControl.exe /command SET_LABEL /value <LABEL>` to communicate a label to the gaze application. Note that `<LABEL>` is a label extracted from the loop table.
  - **timeout** Waits for one second.
- **gaze_recording_disable** Runs the command `GazeControl.exe /command GAZE_RECORDING_DISABLE` to stop storing gaze data to the disk.

gaze_stop Runs the command `GazeControl.exe /command TERMINATE` to gracefully shutdown the gaze application.

The behaviour of the Gaze toolset can be configured with a configuration file (see Chapter 4 for more details). A sample configuration file is provided in `<Gaze toolset path>\sample\config.json` that can be modified to suit different requirements. Copy the configuration file either to `<Gaze toolset path>` or `<zleaf path>` and modify it there.

Chapter 3

Use Gaze Toolset in 3rd Applications

This chapter provides instructions of how to use the Gaze toolest in a 3rd party application. The gaze toolest contains several executable applications which can be started from within 3rd party applications to add eye tracker support. The two main pieces is the `CustomCalibration` application which allows to calibrate the eye tracker and the `Gaze` application which allows to record gaze data. The applications `GazeClose`, `GazeRecordingDisable`, `GazeRecordingEnable`, `MouseTrackingDisable`, and `MouseTrackingEnable` only work in conjunction with the `Gaze` application and are used to control the `Gaze` application (refer to Section 1 for an overview).

3.1 Basic Principles of a Screen Based Tobii Eye Tracker

The eye tracker is mounted on the bottom of the screen such that its cameras are able to capture the eyes of the subject. Several infrared LEDs are visible once the device is connected and properly working (see Figure 3.1). This requires the Tobii software to be installed and running.

Figure 3.1: Picture of the Tobii Eye Tracker 4C in operation.

In order for the eye tracker to being able to track the gaze of the subject it must be set up correctly and calibrated. The setup process must only be done once an can be done through the Tobii Pro Eye Tracker Manager application. However, the eye tracker must be calibrated for each individual the gaze data is measured. Hence, the experiment protocol needs to include a calibration step for each subject. A user-friendly calibration tool is included in the Tobii software package and a custom calibration tool `CustomCalibration` is provided in the gaze toolest. Once the eye tracker is calibrated for a subject, everything is ready.

3.2 Execute 3rd Party Applications in `ztree`

This section describes how to execute application from within `ztree`. In a first step, it is useful to define a global variable `Path` in a `ztree` program that points to the location of the toolset.

In order to define a path

1. click on the last table in `Background`
2. choose the menu `Treatment` → `New Program...`
3. define the path of the installation folder of the `Gaze` toolset, i.e. `<Gaze toolset path>` as `Path` variable (e.g. `Path="C:\\Users\\Max Muster\\Documents\\My Experiment\\Gaze\\";`).
   Note that two backslashes are required for each path delimiter because in ztree \ is used as escape character.

When writing the different stages of a ztree program, calls to external applications can be made. This can be achieved as follows:

1. click on the position in your ztree experiment file where you want to include the call to the external program
2. choose the menu Treatment → New External Program...
3. choose Run on z-Leaf
4. add the call to the external program to the field Command Line
   (e.g a call to the Tobii calibration tool: <><Path|-1>TobiiCalibrate.exe
   Note that the Path variable is used to indicate the location of the program. Figure 3.2 shows an example of calling the Tobii calibration tool.

Figure 3.2: External program call definition in ztree. The variable Path must be globally defined.

3.3 Execute 3rd Party Applications in opensesame

Opensesame allows to run python scripts within the experiment protocols and python can be used to start external applications. All python scripts within an opensesame experiment share the same workspace which means that a variable defined in one script will be accessible by all other scripts as well. This can be used to define the installation path of the gaze toolset in order to avoid typing the same path multiple times. To to this append a new item *inline_script* to the experiment and add the following lines (make sure to set the path to where the gaze toolset was installed):

```
gazeToosetPath="C:\\GazeToolset"
print(f"use gaze toolset path {gazeToolsetPath}")
```

For each application to execute from within the experiment use a separate *inline_script*. To run the TobiiCalibration application use the following script:

```
import subprocess

print("start custom calibration")
subprocess.run([f"{gazeToolsetPath}\\TobiiCalibrate.exe]}")
print("Tobii calibration done")
```

To run the Gaze application use a command to run the application in the background instead of waiting for the application to terminate (note that the subject number variable is passed as an argument to the application):

```
import subprocess

print("start gaze process")
subprocess.Popen([f"{gazeToolsetPath}\\Gaze.exe], "/subject", f"{var.get(u'subject_nr')}")
```

To run the GazeControl.exe of the gaze toolset use subprocess.run with the corresponding command argument. For example, to gracefully terminate the Gaze application with GazeControl and the command argument TERMINATE, use the following script:

```
import subprocess

print("stop gaze process")
subprocess.run([f"{gazeToolsetPath}\\GazeControl.exe], "/command", "TERMINATE")
```

3.3.1 Execute C# Library from Python

Attention In order for this to work it is necessary for the python application to access the `GazeControlLibrary.dll` file. If this file has to be accessed remotely (e.g. the file is stored on a server accessible by multiple lab computers), the python application needs to be told to allow this. This can be achieved by copying the configuration file `python.exe.conf` from the sample folder into the same folder where `python.exe` is stored. Usually, `python.exe` lies in the root installation path of opensesame. Note that if multiple opensesame versions are installed, the configuration file needs to be copied to each installed opensesame version.

Starting a subprocess to send commands to `Gaze.exe` can introduce severe latency to the experiment flow (up to 800ms of latency was observed). To avoid this it is possible to call C# methods directly from within python with the package `pythonnet` (`pip install pythonnet`).

The following example shows how to enable the gaze, set the trial ID 42, set a label `myLabel`, and reset the relative timestamp by calling the method `NamedPipeClient.HandleCommands` from the `GazeControlLibrary.dll` library file.

```
import sys, clr

sys.path.append(r"c:\GazeToolset")
clr.AddReference("GazeControlLibrary")
from GazeControlLibrary import NamedPipeClient

NamedPipeClient.HandleCommands("GAZE_RECORDING_ENABLE", True, 42, "myLabel")
```

The sample file `template.osexp` provides helper functions in the script `gaze_start` in the prepare section to simplify this work. Note the variable `mode` which allows select whether the helper functions will run `GazeControl.exe` with the `subprocess` module (use value `app`) or whether they will perform C# method calls with the `pythonnet` module (use value `lib`).

Chapter 4

Configuration of the Gaze Toolset

The Gaze toolset can be configured to work with different installations. It allows to specify installation paths and gives some control over implemented features (e.g. mouse hiding or data logging). The Listing 4.1 shows the default configuration values and provides an explanation for each value. The configuration file contains detailed descriptions for each parameter. It is recommended to go through all the configuration parameters and read the description in order to get an understanding of the available configuration options.

```
{
  // Allows to define the order and the delimiters between the different calibration result values.
  // The definition is of the form
  //  {0}<delim>{1}<delim> ... <delim>{24}
  // where <delim> can be customized (e.g. '\t' for tab, ',' for comma, etc.) and where the numbers are
  // replaced by the following values
  //  - 0: x-coordinate of the calibration point (normalized value)
  //  - 1: y-coordinate of the calibration point (normalized value)
  //  - 2: x-coordinate of the gaze point of the left eye (normalized value)
  //  - 3: y-coordinate of the gaze point of the left eye (normalized value)
  //  - 4: validity of the gaze data of the left eye
  //  - 5: accuracy of the calibration of the left eye
  //  - 6: x-coordinate of the gaze point of the right eye (normalized value)
  //  - 7: y-coordinate of the gaze point of the right eye (normalized value)
  //  - 8: validity of the gaze data of the right eye
  //  - 9: accuracy of the calibration of the right eye
  // To log all possible values with a comma (i.e. ',') as delimiter use the empty string:
  // "CalibrationLogColumnOrder": "",
  // This configuration value has no effect if "CalibrationLogWriteOutput" is set to false.
  "CalibrationLogColumnOrder": "",

  // Defines the titles of the calibration log value columns. A title for all possible columns must
  // be defined. Titles for values that are removed from the "CalibrationLogColumnOrder" parameter
  // will not be logged but must still be defined here. The index of a title must correspond to the
  // value number of the configuration parameter "CalibrationLogColumnOrder".
  // This configuration value has no effect if "CalibrationLogWriteOutput" is set to false.
  "CalibrationLogColumnTitle": [
    "calibrationPoint_x",
    "calibrationPoint_y",
    "left_gazePoint_x",
    "left_gazePoint_y",
    "left_gazePoint_isValid",
    "left_accuracy",
    "right_gazePoint_x",
    "right_gazePoint_y",
    "right_gazePoint_isValid",
    "right_accuracy"
  ],

  // Defines whether gaze calibration data is written to a log file. If set to false, all the configuration
  // items matching the pattern "CalibrationLog*" are ignored.
  "CalibrationLogWriteOutput": true,
```

```
43
    // Define the calibration points to be shown during the calibration process. Each point is given as
    // a normalize coordinate where [0, 0] is the top left corner and [1, 1] the bottom right corner of
    // the screen. Any number of points is permitted.
    "CalibrationPoints": [
48    [ 0.7, 0.5 ],
      [ 0.3, 0.5 ],
      [ 0.9, 0.9 ],
      [ 0.1, 0.9 ],
      [ 0.5, 0.1 ],
53    [ 0.1, 0.1 ],
      [ 0.9, 0.1 ],
      [ 0.5, 0.9 ]
    ],

58  // Define the calibration accuracy threshold in degrees. If the calibration result has a lower estimated
    // accuracy than the here provided value the participant is asked to restart the calibration.
    "CalibrationAccuracyThreshold": 1.5,

    // Automatic calibration retries due to a missed CalibrationAccuracyThreshold.
63  "CalibrationRetries": 0,

    // In order to detect a fixation with the I-DT algorithm a dispersion threshold is required.
    // Provide an angle in degrees. The fixation detection is only used for drift compensation (for
    // calibration and validation the functions provided by the manufacturer are used).
68  "DriftCompensationDispersionThreshold": 0.5,

    // In order to prevent drift compensation from getting out of hand limit the maximal allowed dispresion.
    // If the drift compensation angle is larger than the here defined degrees, no compensation is applied.
    // Provide an angle in degrees.
73  "DriftCompensationDispersionThresholdMax": 3,

    // Specifies the amount of time (in milliseconds) required to fixate the target during drift compensation.
    "DriftCompensationDurationThreshold": 500,

78  // Specifies the amount of time (in milliseconds) to wait for a fixation point during drift compensation.
    // If the timer elapses drift compensation will be aborted. Use a value of zero for infinite timeout.
    "DriftCompensationTimer": 5000,

    // If set to true the drift compensation window is shown on the drift compensation command.
83  // Otherwise only the drift compensation process is done without showing the window.
    "DriftCompensationWindowShow": true,

    // Specifies the amount of time (in milliseconds) required to fixate the target during validation.
    "ValidationDurationThreshold": 1000,
88
    // Allows to define the order and the delimiters between the different validation result values.
    // The definition is of the form
    //  {0}<delim>{1}<delim> ... <delim>{24}
    // where <delim> can be customized (e.g. '\t' for tab, ',' for comma, etc.) and where the numbers are
93  // replaced by the following values
    //  - 0: x-coordinate of the validation point (normalized value)
    //  - 1: y-coordinate of the validation point (normalized value)
    //  - 2: The accuracy in degrees averaged over all collected points for the left eye.
    //  - 3: The precision (standard deviation) in degrees averaged over all collected points for the left eye
      .
98  //  - 4: The precision (root mean square of sample-to-sample error) in degrees averaged over all collected
    //       points for the left eye.
    //  - 5: The accuracy in degrees averaged over all collected points for the left eye.
    //  - 6: The precision (standard deviation) in degrees averaged over all collected points for the left eye
      .
    //  - 7: The precision (root mean square of sample-to-sample error) in degrees averaged over all collected
103 //       points for the left eye.
    // To log all possible values with a comma (i.e. ',') as delimiter use the empty string:
    // "ValidationLogColumnOrder": "",
    // This configuration value has no effect if "ValidationLogWriteOutput" is set to false.
```

```
  "ValidationLogColumnOrder": "",

  // Defines the titles of the validation log value columns. A title for all possible columns must
  // be defined. Titles for values that are removed from the "ValidationLogColumnOrder" parameter
  // will not be logged but must still be defined here. The index of a title must correspond to the
  // value number of the configuration parameter "ValidationLogColumnOrder".
  // This configuration value has no effect if "ValidationLogWriteOutput" is set to false.
  "ValidationLogColumnTitle": [
    "validationPoint_x",
    "validationPoint_y",
    "left_accuracy",
    "left_precision",
    "left_precision_rms",
    "right_accuracy",
    "right_precision",
    "right_precision_rms"
  ],

  // Defines whether gaze validation data is written to a log file. If set to false, all the configuration
  // items matching the pattern "ValidationLog*" are ignored.
  "ValidationLogWriteOutput": true,

  // Define the validation points to be shown during the validation process. Each point is given as
  // a normalize coordinate where [0, 0] is the top left corner and [1, 1] the bottom right corner of
  // the screen. Any number of points is permitted.
  "ValidationPoints": [
    [ 0.7, 0.5 ],
    [ 0.3, 0.5 ],
    [ 0.9, 0.9 ],
    [ 0.1, 0.9 ],
    [ 0.5, 0.1 ],
    [ 0.1, 0.1 ],
    [ 0.9, 0.1 ],
    [ 0.5, 0.9 ]
  ],

  // Specifies the amount of time (in milliseconds) to wait for a fixation point during validation.
  "ValidationTimer": 3000,

  // Define the validation accuracy threshold in degrees. If the validation result has a lower estimated
  // accuracy than the here provided value the participant is asked to restart the validation.
  "ValidationAccuracyThreshold": 1.5,

  // Define the validation precision threshold in degrees. If the validation result has a lower estimated
  // precision than the here provided value the participant is asked to restart the validation.
  "ValidationPrecisionThreshold": 1.5,

  // Automatic validation retries due to a missed ValidationAccuracyThreshold or
    ValidationPrecisionThreshold.
  "ValidationRetries": 0,

  // Allows to define the order and the delimiters between the different gaze data values.
  // The definition is of the form
  //  {0}<delim>{1}<delim> ... <delim>{24}
  // where <delim> can be customized (e.g. '\t' for tab, ',' for comma, etc.) and where the numbers are
  // replaced by the following values

  //  - 0: timestamp of the gaze data item when it was captured by the tracker (uses DataLogFormatTimeStamp)
  //  - 1: timestamp of the gaze data item when it was received by the system (uses DataLogFormatTimeStamp)
  //  - 2: relative timestamp of the gaze data item (uses DataLogFormatTimeStampRelative)
  //  - 3: the ID of the current trial
  //  - 4: an arbitrary tag to annotate the data sample

  //  - 5: x-coordinate of the drift compensated combined 2d gaze point (normalized value)
  //  - 6: y-coordinate of the drift compensated combined 2d gaze point (normalized value)
  //  - 7: x-coordinate of the raw combined 2d gaze point (normalized value)
```

```
//  - 8: y-coordinate of the raw combined 2d gaze point (normalized value)
//  - 9: flag indicating whether the combined 2d gaze point values are valid
//  - 10: x-coordinate of the drift compensated combined 3d gaze point (mm in UCS)
//  - 11: y-coordinate of the drift compensated combined 3d gaze point (mm in UCS)
//  - 12: y-coordinate of the drift compensated combined 3d gaze point (mm in UCS)
//  - 13: x-coordinate of the raw combined 3d gaze point (mm in UCS)
//  - 14: y-coordinate of the raw combined 3d gaze point (mm in UCS)
//  - 15: z-coordinate of the raw combined 3d gaze point (mm in UCS)
//  - 16: flag indicating whether the combined 3d gaze point values are valid
//  - 17: x-coordinate of the combined 3d gaze origin (mm in UCS)
//  - 18: y-coordinate of the combined 3d gaze origin (mm in UCS)
//  - 19: z-coordinate of the combined 3d gaze origin (mm in UCS)
//  - 20: flag indicating whether the combined 3d gaze origin values are valid
//  - 21: the distance of the gaze origin to the gaze point (mm)
//  - 22: the average pupil diameter of both eyes (mm)
//  - 23: flag indicating whether the averaged pupil diameter is valid

//  - 24: x-coordinate of the raw left 2d gaze point (normalized value)
//  - 25: y-coordinate of the raw left 2d gaze point (normalized value)
//  - 26: flag indicating whether the left 2d gaze point values are valid
//  - 27: x-coordinate of the raw left 3d gaze point (mm in UCS)
//  - 28: y-coordinate of the raw left 3d gaze point (mm in UCS)
//  - 29: z-coordinate of the raw left 3d gaze point (mm in UCS)
//  - 30: flag indicating whether the left 3d gaze point values are valid
//  - 31: x-coordinate of the left 3d gaze origin (mm in UCS)
//  - 32: y-coordinate of the left 3d gaze origin (mm in UCS)
//  - 33: z-coordinate of the left 3d gaze origin (mm in UCS)
//  - 34: flag indicating whether the left 3d gaze origin values are valid
//  - 35: the distance of the left gaze origin to the gaze point (mm)
//  - 36: the pupil diameter of the left eyes (mm)
//  - 37: flag indicating whether the left pupil diameter is valid

//  - 38: x-coordinate of the raw right 2d gaze point (normalized value)
//  - 39: y-coordinate of the raw right 2d gaze point (normalized value)
//  - 40: flag indicating whether the right 2d gaze point values are valid
//  - 41: x-coordinate of the raw right 3d gaze point (mm in UCS)
//  - 42: y-coordinate of the raw right 3d gaze point (mm in UCS)
//  - 43: z-coordinate of the raw right 3d gaze point (mm in UCS)
//  - 44: flag indicating whether the right 3d gaze point values are valid
//  - 45: x-coordinate of the right 3d gaze origin (mm in UCS)
//  - 46: y-coordinate of the right 3d gaze origin (mm in UCS)
//  - 47: z-coordinate of the right 3d gaze origin (mm in UCS)
//  - 48: flag indicating whether the right 3d gaze origin values are valid
//  - 49: the distance of the right gaze origin to the gaze point (mm)
//  - 50: the pupil diameter of the right eyes (mm)
//  - 51: flag indicating whether the right pupil diameter is valid

// To log all possible values with a comma (i.e. ',') as delimiter use the empty string:
// "DataLogColumnOrder": "",
// This configuration value has no effect if "DataLogWriteOutput" is set to false.
"DataLogColumnOrder": "",

// Defines the titles of the gaze data log value columns. A title for all possible columns must be
// defined. Titles for values that are removed from the "DataLogColumnOrder" parameter will not be
// logged but must still be defined here. The index of a title must correspond to the value number
// of the configuration parameter "DataLogColumnOrder".
// This configuration value has no effect if "DataLogWriteOutput" is set to false.
"DataLogColumnTitle": [
  "timestamp",
  "timestamp_received",
  "timestamp_relative",
  "trial_id",
  "tag",

  "combined_gazePoint2dCompensated_x",
  "combined_gazePoint2dCompensated_y",
```

```
    "combined_gazePoint2d_x",
    "combined_gazePoint2d_y",
    "combined_gazePoint2d_isValid",
    "combined_gazePoint3dCompensated_x",
    "combined_gazePoint3dCompensated_y",
    "combined_gazePoint3dCompensated_z",
    "combined_gazePoint3d_x",
    "combined_gazePoint3d_y",
    "combined_gazePoint3d_z",
    "combined_gazePoint3d_isValid",
    "combined_originPoint3d_x",
    "combined_originPoint3d_y",
    "combined_originPoint3d_z",
    "combined_originPoint3d_isValid",
    "combined_gazeDistance",
    "combined_pupilDiameter",
    "combined_pupilDiameter_isValid",

    "left_gazePoint2d_x",
    "left_gazePoint2d_y",
    "left_gazePoint2d_isValid",
    "left_gazePoint3d_x",
    "left_gazePoint3d_y",
    "left_gazePoint3d_z",
    "left_gazePoint3d_isValid",
    "left_gazeOrigin3d_x",
    "left_gazeOrigin3d_y",
    "left_gazeOrigin3d_z",
    "left_gazeOrigin3d_isValid",
    "left_gazeDistance",
    "left_pupilDiameter",
    "left_pupilDiameter_isValid",

    "right_gazePoint2d_x",
    "right_gazePoint2d_y",
    "right_gazePoint2d_isValid",
    "right_gazePoint3d_x",
    "right_gazePoint3d_y",
    "right_gazePoint3d_z",
    "right_gazePoint3d_isValid",
    "right_gazeOrigin3d_x",
    "right_gazeOrigin3d_y",
    "right_gazeOrigin3d_z",
    "right_gazeOrigin3d_isValid",
    "right_gazeDistance",
    "right_pupilDiameter",
    "right_pupilDiameter_isValid"
],

// Number of maximal allowed output data files in the output path. Oldest files are deleted first. To
// keep all files set the value to 0. A value of 1 means that only the output of the current execution
// is kept.
// Note that if multiple clients write to the same folder, this value should be set to at least the
// number of clients.
// This configuration value has no effect if "DataLogWriteOutput" is set to false.
"DataLogCount": 200,

// Defines whether gaze data storing is disabled on Gaze application start. If set to false gaze
// data will be stored to the output gaze file as soon as the device connection is established.
// If set to true data storing must be enabled manually through the application GazeRecordingEnable.
// This configuration value has no effect if "DataLogWriteOutput" is set to false.
"DataLogDisabledOnStartup": false,

// Allows to define the format of how the pupil diameter (in millimetres) will be logged. Use the .NET
// syntax to specify the format:
//  https://docs.microsoft.com/en-us/dotnet/standard/base-types/formatting-types
```

```
// Note that the numbers will be represented differently depending in the localisation settings of the
// windows installation (e.g. 123,4 for DE_CH or 123.4 for EN_US).
// This configuration value has no effect if "DataLogWriteOutput" is set to false.
"DataLogFormatDiameter": "0.000",

// Allows to define the format of how normalized data points will be logged. Use the
// .NET syntax to specify the format:
//  https://docs.microsoft.com/en-us/dotnet/standard/base-types/formatting-types
// Note that the numbers will be represented differently depending in the localisation settings of the
// windows installation (e.g. 123,4 for DE_CH or 123.4 for EN_US).
// This configuration value has no effect if "DataLogWriteOutput" is set to false.
"DataLogFormatNormalizedPoint": "0.000",

// Allows to define the format of how the gaze origin values (in millimetres) will be logged. Use the
// .NET syntax to specify the format:
//  https://docs.microsoft.com/en-us/dotnet/standard/base-types/formatting-types
// Note that the numbers will be represented differently depending in the localisation settings of the
// windows installation (e.g. 123,4 for DE_CH or 123.4 for EN_US).
// This configuration value has no effect if "DataLogWriteOutput" is set to false.
"DataLogFormatOrigin": "0.00",

// Allows to define the format of the timestamp. Use the .NET syntax to specify the format:
//  https://docs.microsoft.com/en-us/dotnet/standard/base-types/formatting-types
// Note that special characters (e.g. ':', '.') need to be escaped with '\\'.
// This configuration value has no effect if "DataLogWriteOutput" is set to false.
"DataLogFormatTimeStamp": "hh\\:mm\\:ss\\.fff",

// Allows to define the format of the relative timestamp in milliseconds. Use the .NET syntax to specify
// the format:
//  https://docs.microsoft.com/en-us/dotnet/standard/base-types/formatting-types
// Note that the numbers will be represented differently depending in the localisation settings of the
// windows installation (e.g. 123,4 for DE_CH or 123.4 for EN_US).
// This configuration value has no effect if "DataLogWriteOutput" is set to false.
"DataLogFormatTimeStampRelative": "0.000",

// Allows to define the format of how validation results will be logged. Use the
// .NET syntax to specify the format:
//  https://docs.microsoft.com/en-us/dotnet/standard/base-types/formatting-types
// Note that the numbers will be represented differently depending in the localisation settings of the
// windows installation (e.g. 123,4 for DE_CH or 123.4 for EN_US).
// This configuration value has no effect if "ValidationLogWriteOutput" is set to false.
"DataLogFormatValidation": "0.000",

// Defines the location of the output file. It must be the path to a folder (not a file). If empty,
// the output file is produced in the directory of the caller (e.g the directory of zleaf.exe).
// This configuration value has no effect if "DataLogWriteOutput" is set to false.
// To avoid confusion with path locations it is recommended to use absolute paths, e.g.:
//  C:\\Users\\Subject\\Documents
"DataLogPath": "",

// Defines whether gaze data is written to a log file. If set to false, all the configuration items
// matching the pattern "DataLog*" are ignored.
"DataLogWriteOutput": true,

// Defines the background color of the canvas where Calibration and validation points are represented.
// To configure the color use either a hex string (with or without alpha value) prefixed with `#` or
// any of the predefined constants from:
//  https: //learn.microsoft.com/en-us/dotnet/api/system.drawing.color?view=net-8.0#properties
"BackgroundColor": "Black",

// Defines the background color of the frame where titles and buttons are represented.
// To configure the color use either a hex string (with or without alpha value) prefixed with `#` or
// any of the predefined constants from:
//  https: //learn.microsoft.com/en-us/dotnet/api/system.drawing.color?view=net-8.0#properties
"FrameColor": "#202124",
```

```
  // Use this parameter to associate the configuration with an experiment. When "Gaze.exe" is
  // executed, a copy of this configuration file is stored at the "DataLogPath" where the parameter
  // "ConfigName" is postfixed to the filename of the copied config file. E.g., by default the following
  // file will be produced at "DataLogPath": <timestamp>_<computer name>_config_experiment_x.json
  // Note that the following characters are not allowed in a file name: <>:"/\|?*
  "ConfigName": "experiment_x",

  // Defines the location of the license files. It must be the path to a folder (not a file).
  // This is only required if the eye tracker device requires an external license file.
  // If an eye tracker does not require a license file either omit this configuration item or
  // set the empty string. Use %S as a placeholder for the device serial number and %A as a
  // placeholder for the device address. The placeholders will be replaced by the actual values
  // of the first device found in the connection list.
  // To avoid confusion with path locations it is recommended to use absolute paths, e.g.:
  //  C:\\Users\\Subject\\Documents\\tobii_licenses
  "LicensePath": "",

  // Specifies the amount of time (in milliseconds) to wait for the eye tracker to become ready while it
  // is in any other state. If the eye tracker is not ready within the specified time the subject will
  // be notified with a pop-up window. This is only relevant for Gaze.exe as the CustomCalibration.exe
  // reacts immediately to lost connections and uses its own GUI to display an error message.
  // Use 0 for immediate reaction.
  "ReadyTimer": 5000,

  // Specifies the amount of time (in milliseconds) to show the custom loading window.
  "LoadingTimer": 1000,

  // Choose the tracker device (1: Tobii Pro SDK, 2: Mouse Tracker).
  // Note that for some eye trackers the Tobii SDK Pro requires a license file to work
  // (see parameter "LicesePath").
  "TrackerDevice": 1,

  // Defines whether the mouse cursor shall be hidden on the calibration window.
  "MouseCalibrationHide": true,

  // Defines whether the mouse cursor shall be controlled by the gaze of the subject during the
  // experiment. If set to true the mouse cursor will be controlled by the gaze of the subject when
  // Gaze.exe is executed and control will be released when GazeClose.exe is executed.
  "MouseControl": false,

  // Defines whether the mouse cursor shall be hidden during the experiment. If set to true the
  // mouse cursor will be hidden when Gaze.exe is executed and restored when GazeClose.exe is executed.
  // This parameter is ignored if "MouseControl" is set to false.
  "MouseControlHide": false,

  // Defines the Path to the standard mouse pointer icon. This is used to restore the mouse pointer.
  // This parameter is ignored if "MouseControl" or "MouseHide" is set to false.
  "MouseStandardIconPath": "C:\\Windows\\Cursors\\aero_arrow.cur",

  // Defines the Tobii installation path. It must be the path to a folder (not a file).
  "TobiiApplicationPath": "C:\\<LocalApplicationData>\\Programs\\TobiiProEyeTrackerManager",

  // The Tobii application to run a calibration.
  "TobiiCalibrate": "TobiiProEyeTrackerManager.exe",

  // The arguments to pass to the calibration application. Use %S as a placeholder for the device
  // serial number and %A as a placeholder for the device address. The placeholders will be replaced
  // by the actual values of the first device found in the connection list.
  "TobiiCalibrateArguments": "--device-sn=%S --mode=usercalibration",

  // If set to true a system tray icon will be shown. Depending on the experimenter software, this
  // may cause to show the taskbar in the foreground. To avoid this disable the systray icon.
  "EnableSystrayIcon": true
}
```

Listing 4.1: Default configuartion values

Note that the configuration file follows the `json` syntax which must not be violated. If the following points are respected, no problem should arise:

- the configuration parameters are enclosed in '{' and '}'.
- all configuration parameters are of the form `"key":value` where `"key"` must not be changed.
- each configuration line ends with a ',' except for the last line where it is omitted.
- the Windows path delimiter '\' must be escaped (i.e. write '\\' when describing a path)
- `json` supports standard data types (e.g. integer, boolean, string). Use the same type as the default value.
- everything following a '//' is considered a comment.

Each executable of the toolset uses the same common configuration file. The configuration file must be named `config.json` and is read from the following places with the indicated priority:

1. in the directory of the caller, i.e. in the execution folder of the ztree client or the opensesame application
2. in the directory of the executables, i.e. in `<Gaze toolset path>`

If no configuration file can be found, the application fails.

Warning: A word of warning when using the mouse hiding feature. This feature hides the mouse when running `Gaze.exe`. If `Gaze.exe` is forcefully closed or crashes, the mouse pointer stays hidden. For such cases the `ShowMouse.exe` utility can be used to restore the mouse pointer.

4.1 Gaze Output File

When running the program `Gaze.exe` an output file can be generated which holds the gaze data provided by the Tobii engine. The output file is saved in the directory specified by `OutputPath` in `config.json`. The name of the output file follows the form

```
<yyyyMMddTHHmmss>_<hostName>_<ConfigName>[_<SubjectNumber>]_gaze[_err-<code>].txt
```

where

- `<yyyyMMddTHHmmss>` is replaced by the timestamp indicating when the file was created (e.g. 20180129T085521 stands for 29.01.2018 08:55:21).
- `<hostName>` is replaced by the name of the machine
- `<ConfigName>` is replaced by the configuration value `"ConfigName"` as specified in the configuration file (see Listing 4.1 for more details)
- `<SubjectNumber>` is replaced by the subject number passed by argument `/subject` to the application. If no argument `/subject` is passed the subject number is omitted in the file name (as well as the prefixed underline character).
- `[_err-<code>]` is either omitted if no error occurred during the experiment or indicates data errors where
  - `<code>` is replaced by an error code which is a binary string where each character can either be `1`, indicating an error or 0, indicating no error (see Section 4.1.3)

A gaze data output file is only generated if the parameter `"DataLogWriteOutput"` in the configuration file is set to `true` (which is the default). The presentation of the data is configurable with the help of the parameters `"DataLogColumnOrder"`, `"DataLogColumnTitle"`, `"DataLogFormatDiameter"`, `"DataLogFormatOrigin"`, and `"DataLogFormatTimeStamp"` (see Listing 4.1 for more details).

A gaze point describes a point on the screen in x and y coordinates (pixel values) the user is gazing at. This coordinate system is called *Active Display Coordinate System (ADCS)* and is illustrated in Figure 4.1.

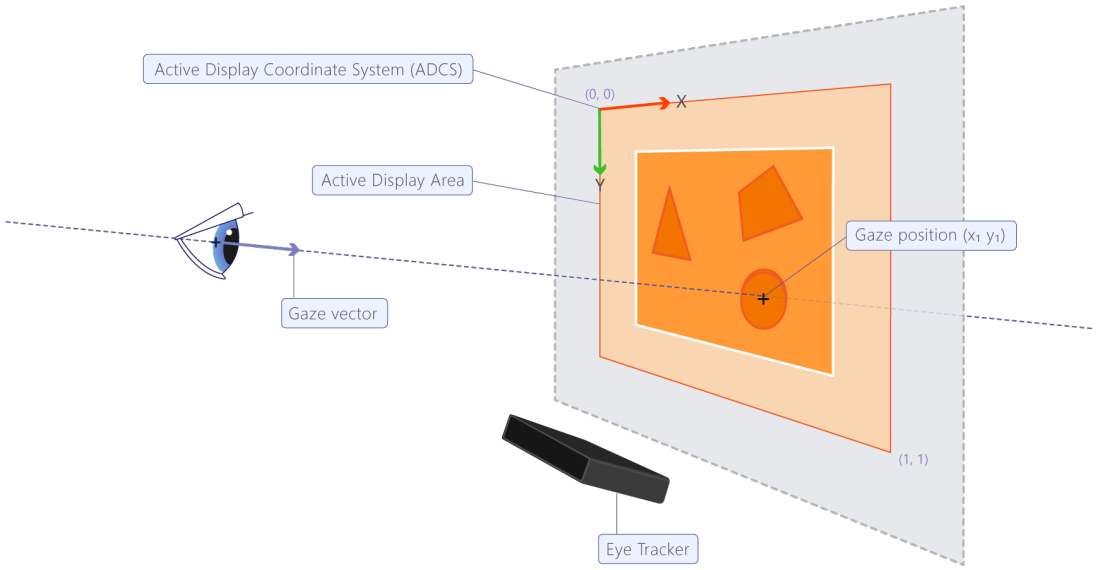

Figure 4.1: The Active Display Coordinate System (ADCS). Figure source: Tobii

By default, the system is configured to log all available data points. However, this can easily be changed through the parameter `"DataLogColumnOrder"`. For example, to only store a minimalistic set of data points use the value `"{0}\t{5}\t{6}"`. This will only store the timestamp of when the gaze point was captured by the eye tracker (data field number 0) and the x and y coordinates of the gaze point (data field numbers 5 and 6, respectively). This produces an output file that is similar to the following:

```
Timestamp   combined_gazePoint2d_x  combined_gazePoint2d_y
12:51:55.409    426 342
12:51:55.420    430 341
12:51:55.431    431 341
...
```

When opening the file with a spreadsheet software such as LibreOffice Calc or Microsoft Office Excel two things need to be considered:

1. The column delimiter needs to be set to the delimiter set in the configuration file (`'\t'` by default).
2. The language needs to be set to the language of the windows system where the configuration file was produced. This is important because numbers are represented differently in different languages and depending on the settings, commas might be interpreted as delimiters when they are not.

When using Tobii Pro SDK, much more values are provided and can be logged to the output file. This includes the pupil diameters of each eye as well as the eye positions in space. The latter uses a coordinate system which is called *User Coordinate System (UCS)*. The UCS is illustrated in Figure 4.2.

In order to log additional data fields, the parameter `"DataLogColumnOrder"` must be modified by adding the required numbers, representing different data fields.

When using the empty string, all possible data fields are logged:

```
"DataLogColumnOrder": ""
```

4.1.1 Gaze Data Fields

The comments above the parameter `"DataLogColumnOrder"` in the configuration file (see Listing 4.1) provide a short description for each available data field. In the following some further information is provided.

The following data field types with individual data fields are provided:

1. timestamps indicating when the data was sampled by the eye tracker

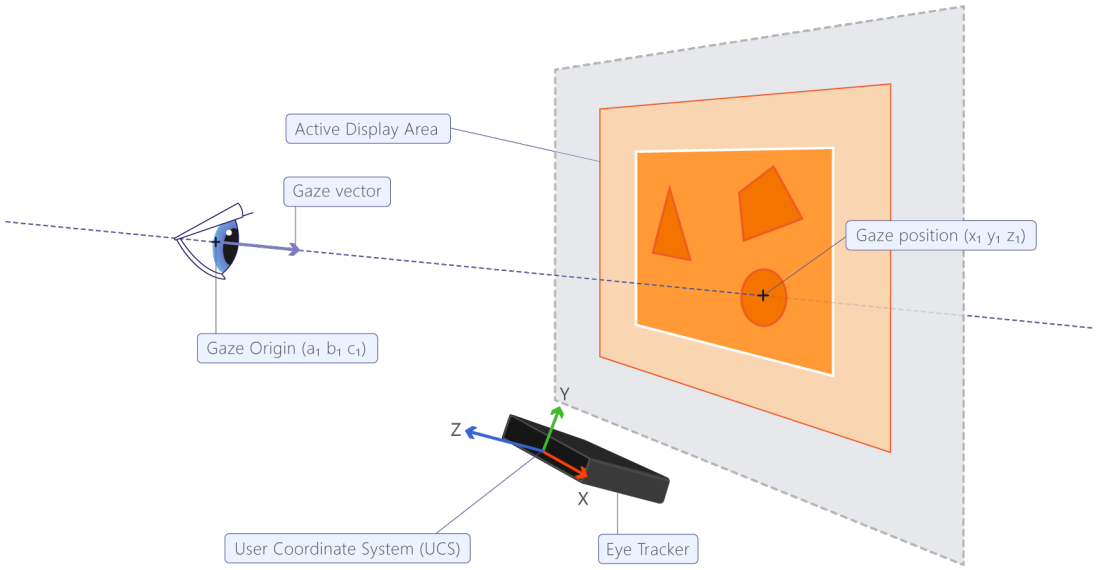

Figure 4.2: The User Coordinate System (UCS). Figure source: Tobii

2. the trial ID and an arbitrary tag to annotate each data sample
3. 2D gaze point ADCS coordinates (see Figure 4.1) as normalized values

   Mouse Tracker

   - the x and y coordinates of the mouse pointer

   Tobii Pro SDK

   - raw x and y coordinates of the left and the right eye
   - the drift compensated x and y coordinates which are computed by the drift compensation process using the I-DT algorithm for fixation detection and then comparing the averaged gaze samples to an oracle gaze to computing the drift of the gaze
   - the average x and y coordinates which are computed from raw data as defined by Equation 4.1

$$\text{val} = \begin{cases} \text{NaN} & \text{if val\_right and val\_left are not valid} \\ \text{val\_left} & \text{if val\_right is not valid} \\ \text{val\_right} & \text{if val\_left is not valid} \\ \frac{\text{val\_left}+\text{val\_right}}{2} & \text{otherwise} \end{cases} \tag{4.1}$$

4. 3D gaze point UCS coordinates (see Figure 4.2) in millimeters

   Mouse Tracker not applicable

   Tobii Pro SDK

   - raw x, y, and z coordinates of the left and the right eye
   - the drift compensated x, y, and z coordinates which are computed by the drift compensation process using the I-DT algorithm for fixation detection and then comparing the averaged gaze samples to an oracle gaze to computing the drift of the gaze
   - the average x and y coordinates which are computed from raw data as defined by Equation 4.1

5. gaze origin UCS coordinates (see Figure 4.2) in millimetres

   Mouse Tracker not applicable

   Tobii Pro SDK

   - raw x, y, and z coordinates of the left and the right eye
   - the average x, y, and z coordinates which are computed from raw data as defined by Equation 4.1
   - the distance from the gaze origin to the left and the right gaze point, respectively which are computed from raw data as defined by Equation 4.2

$$\text{dist} = \sqrt{x^2 + y^2 + z^2} \tag{4.2}$$

- the average distance which is computed from the distance of the left and the right eye as defined by Equation 4.1

6. pupil diameters in millimetres

   Mouse Tracker not applicable

   Tobii Pro SDK
   - raw diameters of the left and the right eye
   - the average diameter which is computed from raw data as defined by Equation 4.1

7. data validity indicators as true or false

   Mouse Tracker not supported

   Tobii Pro SDK
   - separate values that indicate whether gaze points, gaze origins, and pupil diameters are valid for combined values, the left and the right eye, respectively

Notice: The time resolution of the mouse tracker device is rather poor (15 milliseconds). Therefore it can happen that multiple mouse events are logged with the same timestamp.

4.1.2 Gaze Data Timestamps and Annotations

Gaze data have three different timestamps:

timestamp The system time when the data point was captured by the device.

timestamp_received The system time when the data point was received by the system.

Timestamp_relative The number of milliseconds since the start of the application. Note that this timestamp can be reset with the command `GazeControl.exe /reset`.

The latency of the gaze data can computed with Equation 4.3.

$$latency = timestamp\_received - timestamp \tag{4.3}$$

Gaze data can be annotated with a `label` and a `trial ID`. To do this use the command `GazeControl.exe /label <label>` and `GazeControl.exe /trialID <ID of the trial>`, respectively. A label can be any arbitrary string (make sure to use quotations marks in the command if the label has spaces). The trial ID must be an integer number. The label and trial ID annotations are assigned to gaze data points synchronized with the timestamp when the data was captured. This means that when a label or a trial ID is set through a command, the gaze data is not annotated immediately. Rather, the annotations are delayed such that the timestamps are synchronized with the timestamps of the data capture.

4.1.3 Gaze Data Error Code

The data error code that is postfixed to the output file name is a binary string where each character indicates whether a specific error has occurred (indicated with the number `1`) or not (indicated with the number `0`). Multiple errors can occur at the same time, hence, the position of a `1` in the error code indicates the specific error. If all positions of the error code are 0, the errors are not postfixed to the output file name.

The following list describes the individual errors that can occur during an experiment:

code `10`
This code indicates that during the experiment the eye tracker device stopped tracking. This means that in the output file gaze data is potentially missing. This can be caused by a malfunctioning of the eye tracker device or simply because the device was disconnected. The exact time instances of such an occurrence is logged to the log file.

code `01`
This code indicates that the system had to fall back to the Mouse Tracker. This error occurs if the system was configured to use the Tobii Pro SDK but was not able to do so. This usually happens when the license file is not accessible. Check whether the license files are accessible by the system and whether the license path is correctly defined in parameter `"LicensePath"`.

4.2 Calibration Output File

When running the program `GazeControl.exe /command CUSTOM_CALIBRATE` an output file can be generated which holds the calibration results provided by the Tobii engine. The output file is saved in the directory specified by `OutputPath` in `config.json`. The name of the output file follows the form

```
<yyyyMMddTHHmmss>_<hostName>_<ConfigName>[_<SubjectNumber>]_calibration[_err-<code>].txt
```

where

- `<yyyyMMddTHHmmss>` is replaced by the timestamp indicating when the file was created (e.g. 20180129T085521 stands for 29.01.2018 08:55:21).
- `<hostName>` is replaced by the name of the machine
- `<ConfigName>` is replaced by the configuration value `"ConfigName"` as specified in the configuration file (see Listing 4.1 for more details)
- `<SubjectNumber>` is replaced by the subject number passed by argument `/subject` to the application. If no argument `/subject` is passed the subject number is omitted in the file name (as well as the prefixed underline character).
- `[_err-<code>]` is either omitted if no error occurred during the experiment or indicates data errors where
  - `<code>` is replaced by an error code which is a binary string where each character can either be 1, indicating an error or 0, indicating no error (see Section 4.2.2)

A calibration output file is only generated if the parameter `"CalibrationLogWriteOutput"` in the configuration file is set to `true` (which is the default).

The configuration of the calibration data fields to be stored to the output file follows the same logic as described with the gaze output in Section 4.1, except that the presentation of the data is defined through parameters `"CalibrationLogColumnOrder"`, `"CalibrationColumnTitle"`, `"DataLogFormatNormalizedPoint"`, and `"DataLogFormatTimeStamp"` (see Listing 4.1 for more details).

4.2.1 Calibration Data Fields

The comments above the parameter `"CalibrationLogColumnOrder"` in the configuration file (see Listing 4.1) provide a short description for each available data field. In the following some further information is provided.

The following three data field types with individual data fields are provided:

1. calibration point ADCS coordinates (see Figure 4.1) as normalized values

   Mouse Tracker not applicable
   Tobii Pro SDK
   - x and y coordinates of the calibration point

2. gaze point ADCS coordinates (see Figure 4.1) as normalized values

   Mouse Tracker not applicable
   Tobii Pro SDK
   - x and y coordinates of the gaze point of the left and right eye

3. boolean data validity indicators

   Mouse Tracker not supported
   Tobii Pro SDK
   - separate values that indicate whether gaze points are valid for the left and the right eye, respectively

4. calibration accuracy estimation

   Mouse Tracker not supported
   Tobii Pro SDK
   - separate values provide an estimated accuracy for the left and the right eye, respectively

4.2.2 Calibration Data Error Code

The error code that is postfixed to the calibration output file name is a binary string where each character indicates whether a specific error has occurred (indicated with the number `1`) or not (indicated with the number `0`). Multiple errors can occur at the same time, hence, the position of a `1` in the error code indicates the specific error. If all positions of the error code are 0, the errors are not postfixed to the output file name.

The following list describes the individual errors that can occur during an experiment:

code `10`
: This code indicates that during the calibration the eye tracker device stopped tracking. This means that the calibration had to be redone in order to be valid. This can be caused by a malfunctioning of the eye tracker device or simply because the device was disconnected. The exact time instances of such an occurrence is logged to the log file.

code `01`
: This code indicates that the system was not able to start the calibration because the device does not support calibration. This error occurs if the configuration option `TrackerDevice` was set to a device with no calibration support.

4.3 Validation Output File

When running the program `GazeControl.exe /command VALIDATE` an output file can be generated which holds the validation results provided by the Tobii engine. The output file is saved in the directory specified by `OutputPath` in `config.json`. The name of the output file follows the form

```
<yyyyMMddTHHmmss>_<hostName>_<ConfigName>[_<SubjectNumber>]_validation[_err-<code>].txt
```

where

- `<yyyyMMddTHHmmss>` is replaced by the timestamp indicating when the file was created (e.g. 20180129T085521 stands for 29.01.2018 08:55:21).
- `<hostName>` is replaced by the name of the machine
- `<ConfigName>` is replaced by the configuration value `"ConfigName"` as specified in the configuration file (see Listing 4.1 for more details)
- `<SubjectNumber>` is replaced by the subject number passed by argument `/subject` to the application. If no argument `/subject` is passed the subject number is omitted in the file name (as well as the prefixed underline character).
- `[_err-<code>]` is either omitted if no error occurred during the experiment or indicates data errors where
  - `<code>` is replaced by an error code which is a binary string where each character can either be `1`, indicating an error or 0, indicating no error (see Section 4.2.2)

A validation output file is only generated if the parameter `ValidationLogWriteOutput"` in the configuration file is set to `true` (which is the default).

The configuration of the validation data fields to be stored to the output file follows the same logic as described with the gaze output in Section 4.1, except that the presentation of the data is defined through parameters `"ValidationLogColumnOrder"` and `"ValidationColumnTitle"` (see Listing 4.1 for more details).

4.3.1 Validation Data Fields

The comments above the parameter `"ValidationLogColumnOrder"` in the configuration file (see Listing 4.1) provide a short description for each available data field. In the following some further information is provided.

The following three data fields are provided for the left and the right eye:

1. validation point ADCS coordinates (see Figure 4.1) as normalized values

Mouse Tracker not applicable

Tobii Pro SDK

- x and y coordinates of the validation point

2. accuracy of the gaze points.

   Mouse Tracker not applicable

   Tobii Pro SDK

   - systematic error of the measured gaze point with respect to the target gaze point in degrees

3. precision of the gaze points

   Mouse Tracker not applicable

   Tobii Pro SDK

   - averaged standard deviation over all collected points in degrees

4. precision RMS

   Mouse Tracker not supported

   Tobii Pro SDK

   - averaged root mean square of sample-to-sample error over all collected points in degrees

4.3.2 Validation Data Error Code

The error code that is postfixed to the validation output file name is a binary string where each character indicates whether a specific error has occurred (indicated with the number `1`) or not (indicated with the number `0`). Multiple errors can occur at the same time, hence, the position of a `1` in the error code indicates the specific error. If all positions of the error code are 0, the errors are not postfixed to the output file name.

The following list describes the individual errors that can occur during an experiment:

code `10`
: This code indicates that during the validation the eye tracker device stopped tracking. This means that the validation had to be redone in order to be valid. This can be caused by a malfunctioning of the eye tracker device or simply because the device was disconnected. The exact time instances of such an occurrence is logged to the log file.

code `01`
: This code indicates that the system was not able to start the validation because the device does not support validation. This error occurs if the configuration option `TrackerDevice` was set to a device with no validation support.

4.4 Configuration File Dump

For each experiment where the utility `Gaze.exe` is executed, a dump of the configuration file is produced. This allows to associate a set of configuration values to an experiment, reuse the same configuration file should the experiment be repeated, and provides transparency of how the output data was produced. Note that in this configuration file all comments are omitted and the formatting (indentations, white spaces, carriage return) is removed. To reformat the file or display the file as a tree structure, online tools, such as the Online JSON Viewer[1], can be used.

The name of the dumped configuration file is of the form

```
<yyyyMMddTHHmmss>_<hostName>_<ConfigName>_config[_err-<code>-<code>].txt
```

where

- `<yyyyMMddTHHmmss>` is replaced by the timestamp indicating when the file was created (e.g. 20180129T085521 stands for 29.01.2018 08:55:21)

[1] http://jsonviewer.stack.hu/

- `<hostName>` is replaced by the name of the machine
- `<ConfigName>` is replaced by the configuration value `"ConfigName"` as specified in the configuration file (see Listing 4.1 for more details)
- `[_err-<code>-<code>]` is either omitted if no error occurred during the experiment or indicates data errors where
  - `<code>` is replaced by an error code which is a binary string where each character can either be `1`, indicating an error or `0`, indicating no error (see Section 4.4.1)

In addition to the configuration data used during the experimentation the dumped configuration file includes information on the screen area in UCS coordinate system (see Figure 4.2). The Listing 4.2 shows an example of the screen area data as it can be found in the dumped configuration file.

```
{
  "ScreenArea": {
    "Width": 597.5177001953125,
    "Height": 336.10369873046875,
    "Center": [
      0.1185302734375,
      173.82257080078125,
      56.42921829223633
    ],
    "TopLeft": [
      -298.64031982421875,
      331.7396545410156,
      113.90633392333984
    ],
    "TopRight": [
      298.87738037109375,
      331.7396545410156,
      113.90633392333984
    ],
    "BottomLeft": [
      -298.64031982421875,
      15.905486106872559,
      -1.0478993654251099
    ],
    "BottomRight": [
      298.87738037109375,
      15.905486106872559,
      -1.0478993654251099
    ]
  }
}
```

Listing 4.2: Screen Area Data in Dumped Config

4.4.1 Configuration Error Code

The configuration error codes that are postfixed to the dumped configuration file name are binary strings where each character indicates whether a specific error has occurred (indicated with the number `1`) or not (indicated with the number `0`). Multiple errors can occur at the same time, hence, the position of a `1` in the error code indicates the specific error. If all positions of all error codes are 0, the errors are not postfixed to the dumped configuration file.

The following list describes the individual errors of the first error code that addresses general problems with the configuration file:

code `100`
: The system ignores the configuration file and falls back to the default configuration values. This happens if an invalid configuration file is provided that cannot be parsed. Verify the syntax of the configuration file and make sure that the key names are not modified and no additional keys are added to the file. Note that it is perfectly valid to not provide a configuration file which causes the system to use the default values without creating an error.

code 010
The system uses the current location (e.g. <zleaf path>) to store the output files. This happens if the provided path in the configuration file (parameter "DataLogPath") is invalid or non-existent. Verify that the provided path exists and that no invalid characters are used (do not use <>:"/\|?).

code 001
This happens if no name was provided or the provided name in the configuration file (parameter "ConfigName") is invalid. The configuration name is mandatory and the application failed due to this error. Verify that in the provided name no invalid characters are used (do not use <>:"/\|?).

The following list describes the individual errors of the second error code that specifically addresses problems concerning the formatting of the output file:

code 010000
The system falls back to the default column order. This happens if the parameter "DataLogColumnOrder" is invalid. Make sure that only existing data filed numbers are used and that the format string is valid. More information on format strings is provided on the MSDN page about Composite Formatting[2].

code 001000
The column titles are not printed to the output file. This happens if the value of parameter "DataLogColumnTitle" is not valid. Make sure that a title is provided for all possible data fields (not only the ones that are displayed).

code 000100
The system falls back to the default format for the timestamp. This happens if the value provided for the parameter "DataLogFormatTimeStamp" is invalid. Refer to the MSDN page about Formatting Types[3] for more information.

code 000010
The system falls back to the default format for the gaze origin coordinates. This happens if the value provided for the parameter "DataLogFormatOrigin" is invalid. Refer to the MSDN page about Formatting Types[3] for more information.

code 000001
The system falls back to the default format for the diameter of the pupil. This happens if the value provided for the parameter "DataLogFormatDiameter" is invalid. Refer to the MSDN page about Formatting Types[4] for more information.

code 100000
The system falls back to the default format for the normalized coordination point. This happens if the value provided for the parameter "DataLogNormalizedPoint" is invalid. Refer to the MSDN page about Formatting Types[5] for more information.

4.5 Log File

The app Gaze.exe writes continuously to a log file with a name of the form <hostName>_gaze.log and GazeControl.exe writes to <hostName>_control.log, respectively. The prefix <hostName> is replaced by the name of the machine. This allows to track the eye tracker events that happened throughout a session within two log files. The log files are produced in a folder log at the root directory of the application which is making the calls to the executables (e.g. at the location of zleaf.exe: <zleaf path>).

[2] https://docs.microsoft.com/en-us/dotnet/standard/base-types/composite-formatting
[3] https://docs.microsoft.com/en-us/dotnet/standard/base-types/formatting-types
[4] https://docs.microsoft.com/en-us/dotnet/standard/base-types/formatting-types
[5] https://docs.microsoft.com/en-us/dotnet/standard/base-types/formatting-types