

How to collect data in the biometrics corner

Sammy Wals

Thank you for your interest in the biometrics corner. Please follow the instructions below to run experiments and start recording data using Eye-Tracking (ET), Galvanic-skin-response (GSR), and/or perform webcam video recordings to do facial expression analysis.

See section 1 for more background information about these biometric research methods, as well as the various links and references to other resources to learn more about these methods.

Booking the biometrics corner: The biometrics corner is automatically booked if you make a DEXLab reservation via the [ResourceBooker](#). Every staff member (incl. PhD candidates) of the SBE can book the DEXLab.

The Biometrics corner provides documentation and a template experiment file using [PsychoPy](#). You can copy and adapt this template to fit your own needs, but please do not change the source files. Copy the file and create a new folder for your experiment. Everything in terms of software necessary for the equipment to work is open source and coded (mostly) in Python.

Please read the documentation and go through some trial experiments and example data before starting data collection for your experiment. The folder [C:\Users\dexlab-sbe\Documents\biometrics_corner](#) contains a template experiment that you can run and some example data files to show what the output will be. You can use the code from [xdf_reader.py](#) in visual studio code (VSC) to go through the example data files and examine the output.

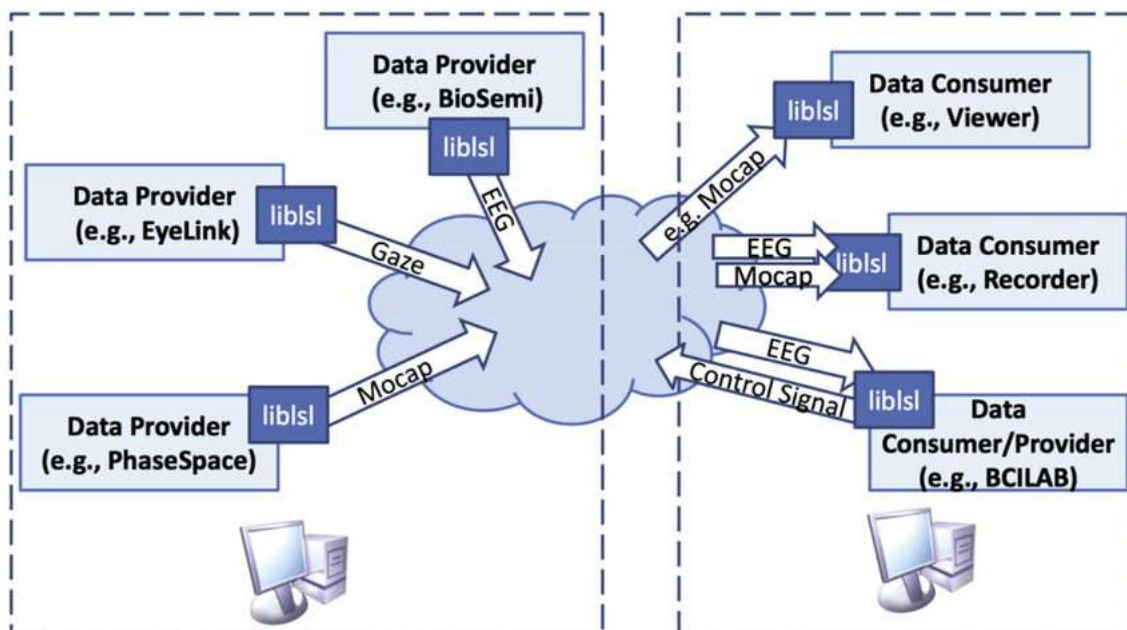
While you can record data from multiple sensors simultaneously with this setup, it may be wise to start with only one sensor due to the increasing complexity of the data and the analysis when using multiple sensors.

If you're stuck, encounter problems, or have any questions regarding the biometrics corner you can contact the current lab manager via e-mail at: sbe-dexlab@maastrichtuniversity.nl

The biometrics corner relies on open-source software, specifically LabstreamingLayer (LSL) to record and synchronize data from multiple biometric sensors. As well as PsychoPy (Pierce, 2007) to program and run experiments, Pyshimmer (Magel, 2023) to work with the Shimmer GSR devices, and Python (Van Rossum & Drake, 2009) to code everything in between.

On a general level, LSL works by streaming data from the different biometric sensors to a network. Then, another program, called the “labrecorder”, can detect and record these signals together in one (XDF) file to synchronize them over time.

To work with LSL, you need to start streaming data from different devices and then use the labrecorder to detect and save these streams in a file on the computer.



When using this open-source software in your publications, please cite the following work:

Peirce, J. W. (2007). PsychoPy - Psychophysics software in Python. *Journal of Neuroscience Methods*, 162 (1-2), 8-13. doi:10.1016/j.jneumeth.2006.11.017

Kothe, C. (2023). LabStreamingLayer. GitHub. <https://github.com/sccn/labstreaminglayer>

Magel, L. (2023). PyShimmer. GitHub. <https://github.com/seemoo-lab/pyshimmer>

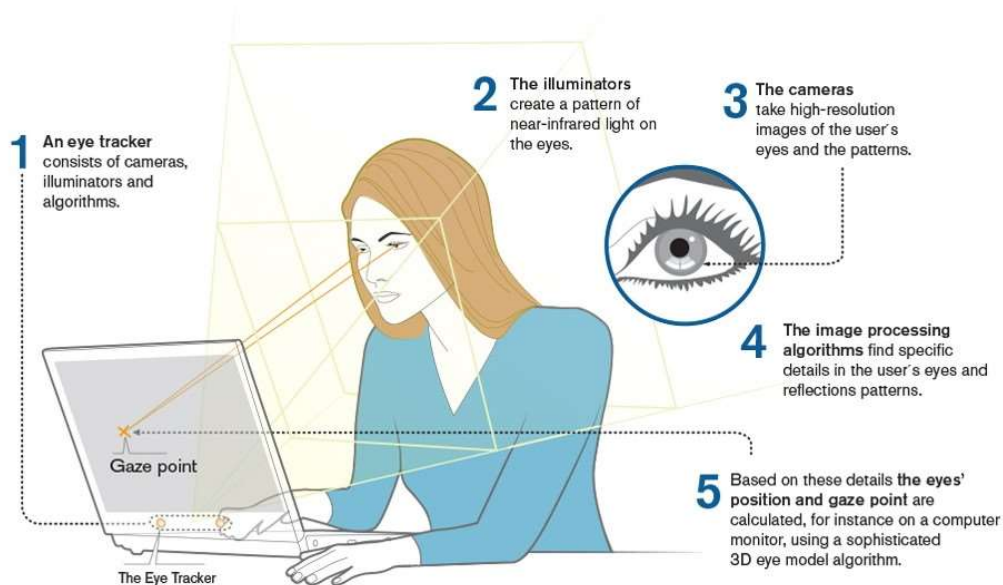
Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace.

Table of contents:

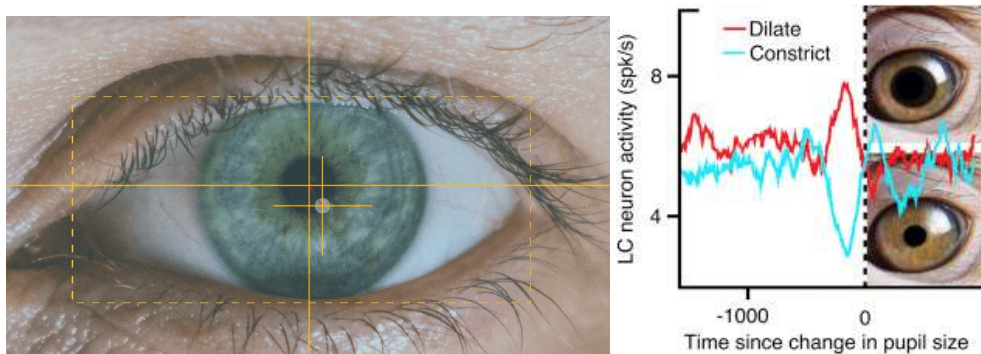
1. Background and information:
 - a. What is eye-tracking?
 - b. What is GSR?
 - c. What is Facial Expression analysis?
2. Quickstart Guide: How to Get Started with Recording Biometric Data
 0. Choosing which biometric sensors to include
 1. Eye-tracking data: calibration and data streaming
 2. GSR data: Bluetooth pairing, physical set up, and data streaming
 3. Facial expressions: webcam video recording and using Facereader by Noldus
 4. Starting up your PsychoPy experiment with event markers
 5. Using the LabRecorder to save and synchronize the data with LSL
 6. Finishing the Experiment and stopping data streaming
 7. Data management: file management and saving the data in the cloud
3. Examining and analyzing the biometric data
 - a. Reading .xdf files with data from multiple sensors and event markers
 - b. Understanding and Preprocessing biometric data and facial expression analysis
4. Final Notes:
 - a. Adapting the template PsychoPy experiment to your own needs
 - b. Collecting GSR data for VR (or other mobile) experiments versus running screen-based experiments in the biometrics corner.

1. Background and information

What is eye-tracking? Eye-tracking is a well-known method to study attention. It works by taking infrared images of the eyes and mapping their movements to a physical display such as a computer screen. The position of the eyes tells us where the participants are looking on the screen (i.e., gaze points in 2D space; defined as X & Y coordinates).



Next to gaze positions, much more information can be drawn from eye tracking data such as fixation durations, which reflect focused attention. And the size of the pupil, which is a very sensitive measure of arousal linked to norepinephrine, see: [Joshi et al., 2016](#).



More information about eye-tracking (ET) can be found here: [What is Eye Tracking and How Does it Work? - iMotions](#)

What is GSR? The acronym GSR stands for galvanic skin response, which refers to small changes in the electrical conductivity of the skin due to activity of the sweat glands (i.e., salt) as people become more aroused (e.g., excited or stressed). An example of how this is used is the case of a lie detector, where people become nervous when they are about to lie. However, the GSR does not measure specific emotions or responses such as lying or happiness, rather it should be interpreted as a general measure of physiological arousal.



We use the shimmer GSR devices to get skin conductance measurements, which you can see on the right here:

This device measures the electrical conductivity between two points on the fingers. The unit of measurement is actually resistance (Ohm), which can be reversed to get to a measure of skin conductivity in Siemens like so ($\text{Siemens} = 1/\text{Ohm}$). These electrical signals are very tiny and they are usually measured in micro Siemens, which is one millionth of a Siemens unit.

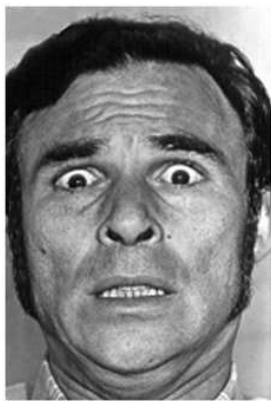
The devices can also be used to measure heart rate using photoplethysmography (PPG), which uses light (“photo-”) to measure changes in the volume of blood; showing the pulses as the heart pumps blood through the arteries. Higher pulse rates indicate arousal. This is however not covered in the guide currently. You can find more information about GSR here:

[Shimmer3 GSR+ Introduction Video - YouTube](#)

[Shimmer3 GSR+ Unit - Shimmer Wearable Sensor Technology \(shimmersensing.com\)](#)

[Galvanic Skin Response \(GSR\): The Complete Pocket Guide - iMotions](#)

What is Facial Expression Analysis? At a basic level, facial expression analysis is the practice of recognizing emotions from faces. Facial expressions are based on “facial action units”, which is the level of activation in specific facial muscles. Such as the Zygomaticus Major muscle that is used for smiling, or the Corrugator Supercilii muscle used for frowning. The activation level in specific muscles can then be used to classify basic emotional states (e.g., angry/sad), but also to get continuous measures of activation in specific muscles over time, such as the level of frowning or smiling. The most well-known facial expression classification model is based on Paul Ekman’s discrete theory of the 6 basic emotions, shown below:



Fearful



Angry



Sad



Happy



Disgusted



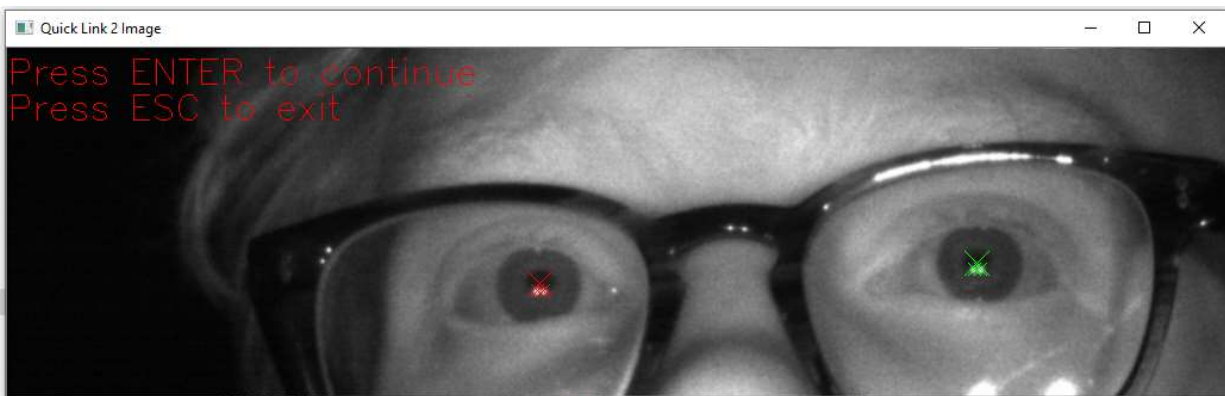
Surprised

While facial expressions used to be classified manually by researchers coding the facial action units, or by measuring the electrical activity in the facial muscles. This can now be done more easily with automated analysis of video recordings using the software by Noldus.

You can find more information about this software at www.noldus.com/facereader

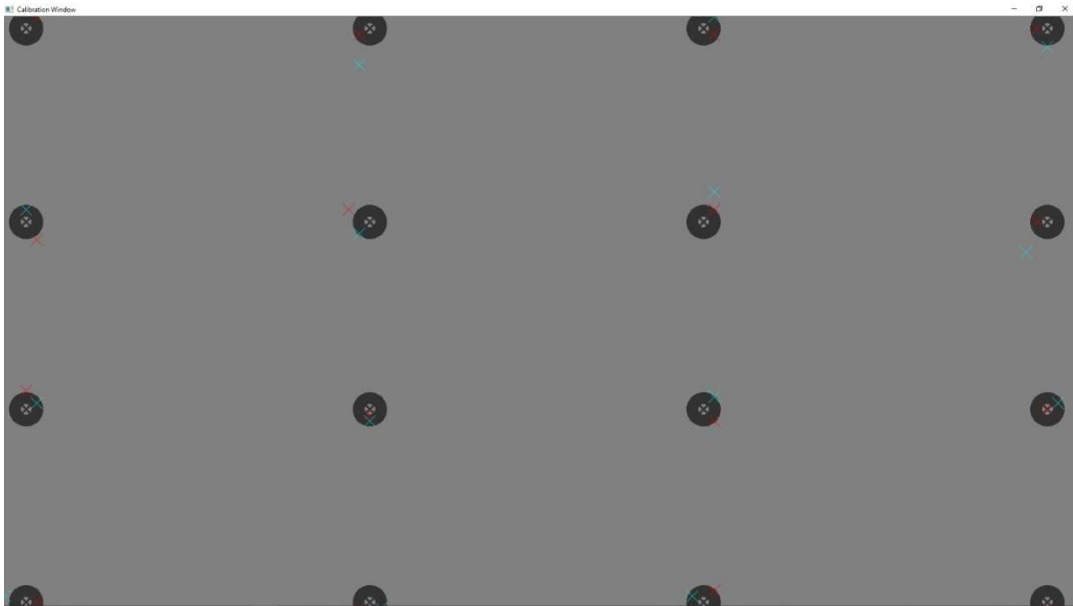
2. Quickstart Guide: How to get Started with Recording Biometric Data

0. Before starting, think about what type of data you want to collect and why. It's possible to collect data from multiple sensors, but this becomes more complicated to analyze and process. Make sure that you have a clear hypothesis and a plan for data collection/analysis beforehand.
1. **If you want to use eye-tracking**, then calibrate the eye-tracker for every individual participant by running the [newcalibration.exe](#) file in the folder and follow the instructions on the screen.
 - a. Make sure the participant is seated in front of the camera with correct height. The participant should sit upright with good posture and 50-70 cm distance from the screen.
 - b. Both LED's on the eye-tracker should turn green if both eyes are visible on the camera.
 - c. Have the participant look at the circles in the calibration procedure. If calibration is successful, the blue and right (left and right eyes gaze points) should overlap with the dots on the screen. Ideally, there would be no distance between the dots and the gaze points, but it is never perfect. You can run the calibration procedure multiple times until you're satisfied. If results are very bad, make sure that the participant is directly in front of the camera with good posture, you can move the screen to fit the participant. Other factors, such as heavy makeup/eyelashes or thick glasses, can influence the results. These are sometimes used as exclusion criteria during participant recruitment.



You can first check whether the eyes of the participant are in the frame by looking at this video that pops up. Both LED's on the eye-tracker should be green and the eyes should be marked in the image. If okay, then press ENTER to continue. The next screen is the calibration screen, move this to the correct monitor and make it full-screen so that it covers the whole display. If okay press ENTER and the calibration starts automatically. The participants should follow the dots that appear on the screen.

When finished, the calibration window should look somewhat like this. The blue and red crosses show the position of the eyes around the calibration dots. Ideally, the crosses should be right on the calibration dots but there is always some deviation. If you are satisfied with the calibration results. You can continue and start streaming data, please exit the calibration with CTRL + C.



- d. When ready to start streaming the ET data. Click the **start_stream_eyetracker.bat** file in the biometrics folder to start streaming data. A CMD window should appear like shown below, showing a LSL connection and some information about the ET device (name and API version). After a moment, the program will start streaming data as shows by the red lights that shown up on the eye-tracking device below the screen.

- **Note:** if the window immediately closes there is an error. Check whether the USB cable is plugged in correctly.
- Make sure that you have calibrated the device before you start streaming data. Otherwise, the data could be invalid/not calibrated correctly.
- The participant should remain good posture and be relatively still during the experiment to get good results, you can check whether the green LED's are still on during the experiment to see whether the eyes are in the frame and the eye-tracker is able to record high-quality data

To quit streaming the eye tracking data, click on the command window that was opened and press CTRL+C on your keyboard to quit the program. The streaming of eye tracking data will stop and the program will ask to confirm. Enter Y for “yes” and press Enter to quit streaming.

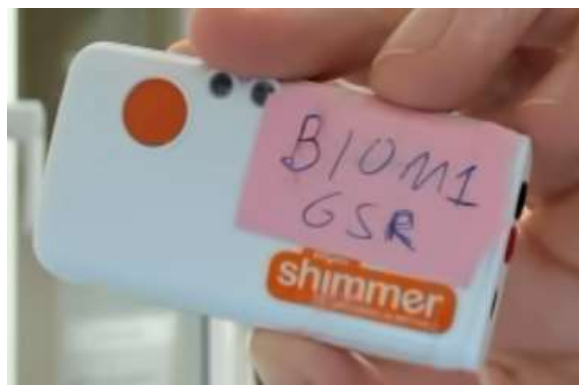

```

C:\Windows\system32\cmd.exe
C:\Users\dexlab-sbe\Documents\biometrics_corner>eyetechlsl stream
stream
Quicklink2 API Version: 2020.1.4.0
Num of devices found: 1
Device ID: 1
Model: AEye 2019.2.3.0
SN: 16783316
Sensor: 2048x1544
Starting device
2023-11-17 13:12:24.550 ( 2.088s) [ A5680039] netinterfaces.cpp:36 INFO| netif '{A6E9B13F-6C31-432A-83E
7-24EFF1E0A025}' (status: 2, multicast: 1
2023-11-17 13:12:24.550 ( 2.088s) [ A5680039] netinterfaces.cpp:36 INFO| netif '{59D61B97-A878-4D8A-AD1
6-A8DB30BF0696}' (status: 2, multicast: 1
2023-11-17 13:12:24.551 ( 2.089s) [ A5680039] netinterfaces.cpp:36 INFO| netif '{9848F02F-A8FE-4B9F-8E8
F-8448EE563A7A}' (status: 1, multicast: 1
2023-11-17 13:12:24.551 ( 2.089s) [ A5680039] netinterfaces.cpp:58 INFO| IPv6 ifindex 14
2023-11-17 13:12:24.551 ( 2.089s) [ A5680039] netinterfaces.cpp:36 INFO| netif '{95148E3F-BD30-412B-89D
2-9CFF2C58A5E7}' (status: 2, multicast: 1
2023-11-17 13:12:24.552 ( 2.090s) [ A5680039] netinterfaces.cpp:36 INFO| netif '{950A1026-5340-11EC-BB5
1-806E6F6E6963}' (status: 1, multicast: 1
2023-11-17 13:12:24.552 ( 2.090s) [ A5680039] netinterfaces.cpp:58 INFO| IPv6 ifindex 1
2023-11-17 13:12:24.552 ( 2.090s) [ A5680039] api_config.cpp:270 INFO| Loaded default config
2023-11-17 13:12:24.553 ( 2.091s) [ A5680039] common.cpp:65 INFO| git:6dc417089a1f73f26589b4f1e6
4def357b183e9c/branch:refs/tags/v1.16.1/build:Release/compiler:MSVC-19.0.24245.0/link:SHARED
2023-11-17 13:12:24.555 ( 2.093s) [ A5680039] udp_server.cpp:82 WARN| Could not bind multicast respo
nder for ff02::1:3d:6fdd:2c17:a643:ffe2:1bd1:3cd2 to interface ::1 (An invalid argument was supplied.)
2023-11-17 13:12:24.555 ( 2.093s) [ A5680039] udp_server.cpp:82 WARN| Could not bind multicast respo
nder for ff05::1:3d:6fdd:2c17:a643:ffe2:1bd1:3cd2 to interface ::1 (An invalid argument was supplied.)

```

2. **If you want to use the Shimmer GSR device**, make sure to use the one that is already paired to the laptop because the code is linked to a specific COM port. The correct shimmer device is labelled BIOM1 with a pink sticker, and the name of the device on the back is 83-C0.

- a. Make sure to use the right device labelled BIOM1, which is already paired to the laptop. Turn it on by using the sliding button on the side; a light should come on.



- b. If it's low on power (red light flashing), charge the shimmer device by plugging into the charging dock, check the device in Consensys to make sure that it's charged and turn on the GSR data collection option. Do not charge the GSR device and while being connected to a participant at the same time. This is a safety hazard because it connects the participant to the charging dock.

- c. Physically attach the GSR sensor to the participant. Plug in the electrodes in the holes on top of the GSR device. The electrodes should be placed like shown below, with the metal part touching the skin on the bottom of the inside of index and ring fingers. Having the electrodes tighter to the skin ensures a better connection with higher quality data, such as reduced data loss, less noise and motion artifacts.

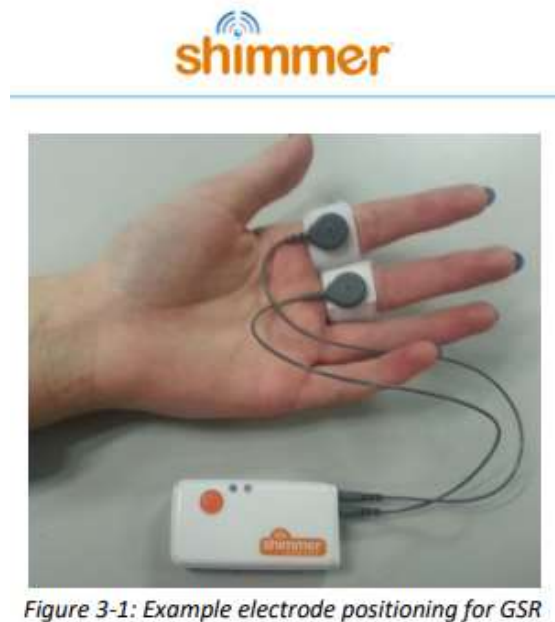
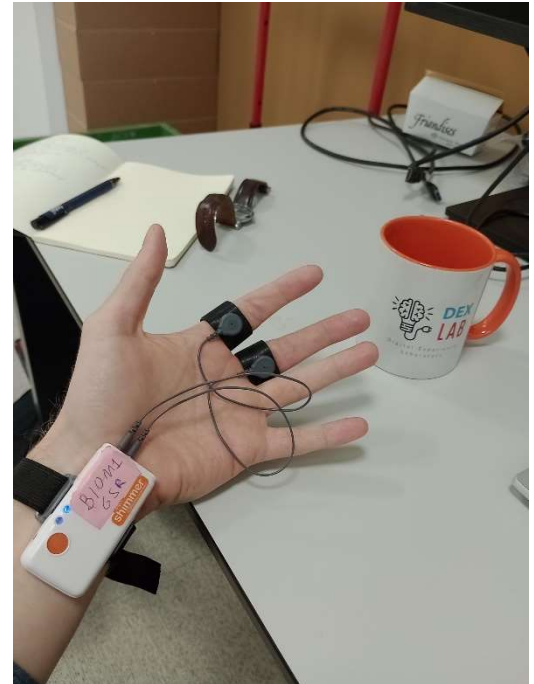


Figure 3-1: Example electrode positioning for GSR

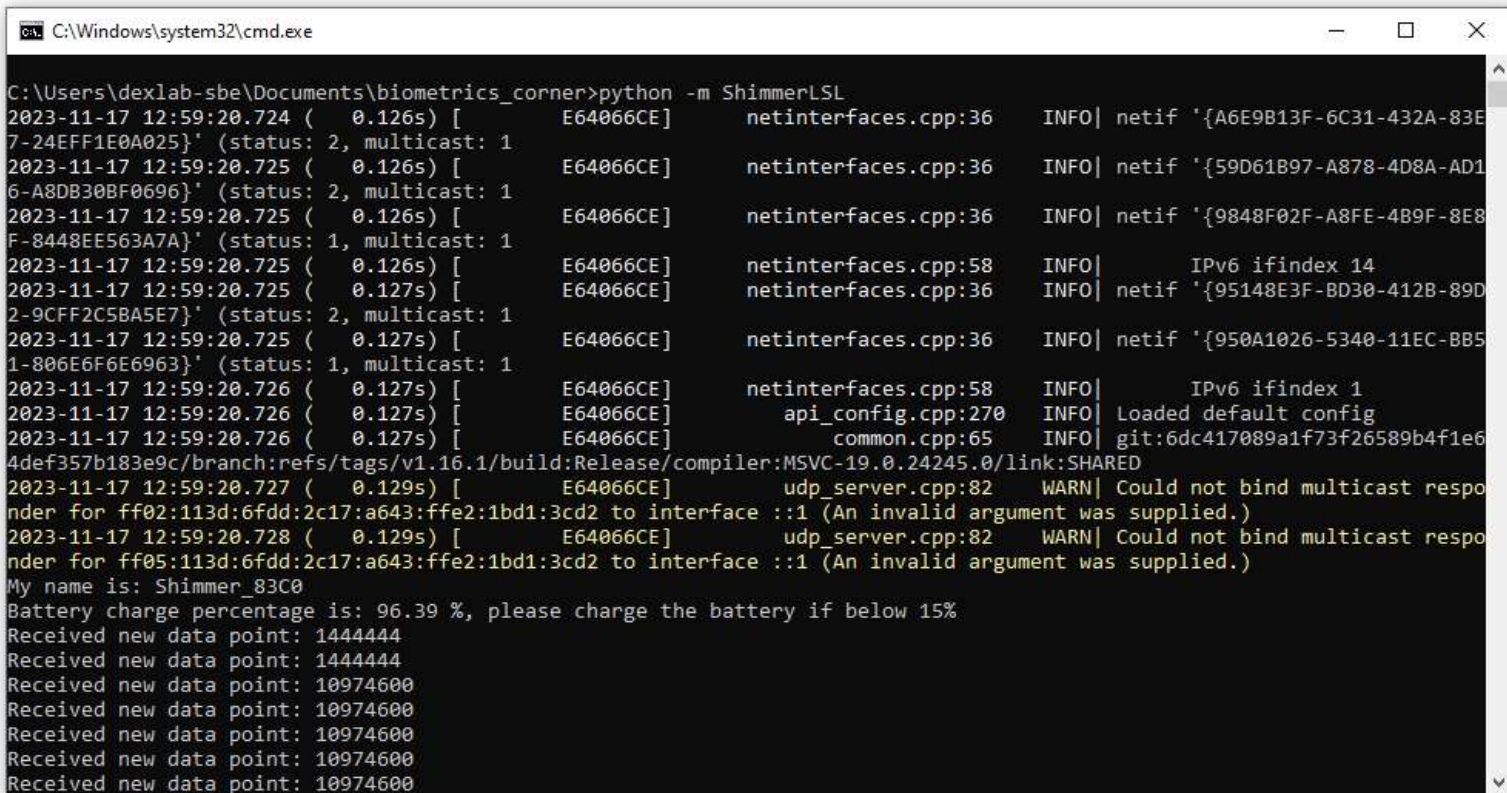


The shimmer device can be attached to the wrist to make the device more easily wearable and mobile if needed. It is generally the most user-friendly to wear the device using the wrist strap that comes with it. The device can be clicked on the wrist strap and device and the sensors should be placed on the inside of the wrist, such that it doesn't get in the way of what the participant is doing. The picture above shows how the GSR device would look setup with the wrist band would on an actual participant.

- d. When ready start streaming the GSR data. Click the **start_stream_GSR.bat** file in the biometrics folder to start streaming data. A CMD window should appear like shown below, showing a LSL connection and some information about the GSR device (name and battery percentage). Check whether the battery is sufficiently charged (e.g., > 20%). After a few seconds, the program will start streaming data as shows by the numbers that show the values per sample (approximately 50 samples per second).

- **Note:** if the window immediately closes there is an error. Check whether the device is turned on (i.e., colored lights should be on).
- If the battery charge is low (e.g., < 20%), please charge the device by plugging it into the charging dock. **Do not keep the GSR device connected to the participant while charging the device in the charging dock because of safety guidelines.**

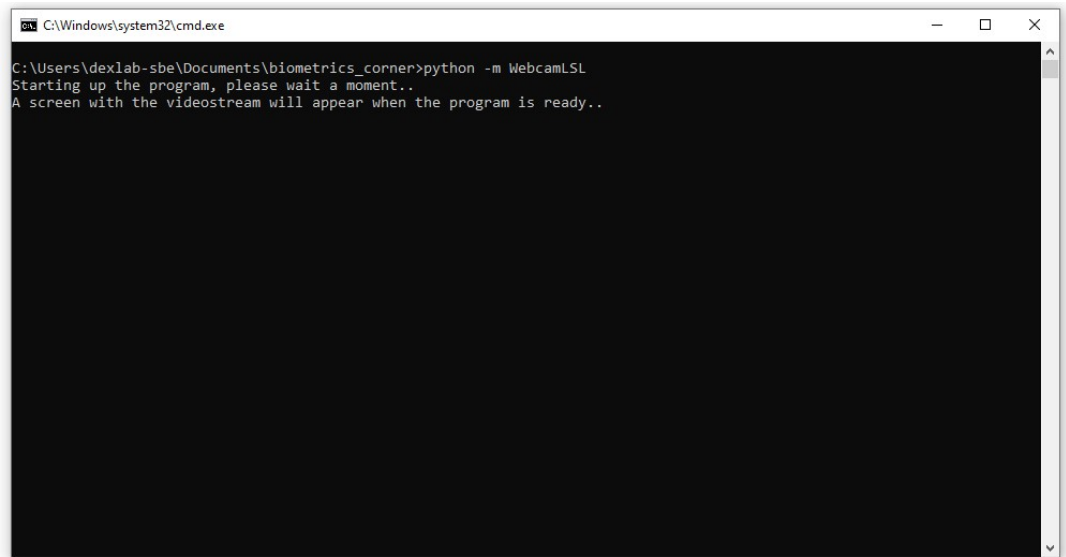
To quit streaming the GSR data, click on the command window that was opened and press CTRL+C on your keyboard to quit the program. The streaming of GSR data will stop and the program will ask to confirm. Enter Y for “yes” and press Enter to quit streaming.



```
C:\Windows\system32\cmd.exe

C:\Users\dexlab-sbe\Documents\biometrics_corner>python -m ShimmerLSL
2023-11-17 12:59:20.724 ( 0.126s) [ E64066CE] netinterfaces.cpp:36 INFO| netif '{A6E9B13F-6C31-432A-83E
7-24EFF1E0A025}' (status: 2, multicast: 1
2023-11-17 12:59:20.725 ( 0.126s) [ E64066CE] netinterfaces.cpp:36 INFO| netif '{59D61B97-A878-4D8A-AD1
6-A8DB30BF0696}' (status: 2, multicast: 1
2023-11-17 12:59:20.725 ( 0.126s) [ E64066CE] netinterfaces.cpp:36 INFO| netif '{9848F02F-A8FE-4B9F-8E8
F-8448EE563A7A}' (status: 1, multicast: 1
2023-11-17 12:59:20.725 ( 0.126s) [ E64066CE] netinterfaces.cpp:58 INFO| IPv6 ifindex 14
2023-11-17 12:59:20.725 ( 0.127s) [ E64066CE] netinterfaces.cpp:36 INFO| netif '{95148E3F-BD30-412B-89D
2-9CFF2C5BA5E7}' (status: 2, multicast: 1
2023-11-17 12:59:20.725 ( 0.127s) [ E64066CE] netinterfaces.cpp:36 INFO| netif '{950A1026-5340-11EC-BB5
1-806E6F6E6963}' (status: 1, multicast: 1
2023-11-17 12:59:20.726 ( 0.127s) [ E64066CE] netinterfaces.cpp:58 INFO| IPv6 ifindex 1
2023-11-17 12:59:20.726 ( 0.127s) [ E64066CE] api_config.cpp:270 INFO| Loaded default config
2023-11-17 12:59:20.726 ( 0.127s) [ E64066CE] common.cpp:65 INFO| git:6dc417089a1f73f26589b4f1e6
4def357b183e9c/branch:refs/tags/v1.16.1/build:Release/compiler:MSVC-19.0.24245.0/link:SHARED
2023-11-17 12:59:20.727 ( 0.129s) [ E64066CE] udp_server.cpp:82 WARN| Could not bind multicast respo
nder for ff02::1:3 to interface ::1 (An invalid argument was supplied.)
2023-11-17 12:59:20.728 ( 0.129s) [ E64066CE] udp_server.cpp:82 WARN| Could not bind multicast respo
nder for ff05::1:3 to interface ::1 (An invalid argument was supplied.)
My name is: Shimmer_83C0
Battery charge percentage is: 96.39 %, please charge the battery if below 15%
Received new data point: 1444444
Received new data point: 1444444
Received new data point: 10974600
Received new data point: 10974600
Received new data point: 10974600
Received new data point: 10974600
Received new data point: 10974600
```


3. **If you want to use the Webcam for facial expression analysis**, please make sure the participant is sitting directly in front of the camera to get a good image of the face.
- a. To start recording data, click the **start_stream_webcam.bat** file in the biometrics folder to start streaming data. A CMD window should appear like shown below, showing a message that ask you to have some patience. After a moment, the program will start streaming data as shows by the webcam video that show up and more information about an LSL stream that has been created.

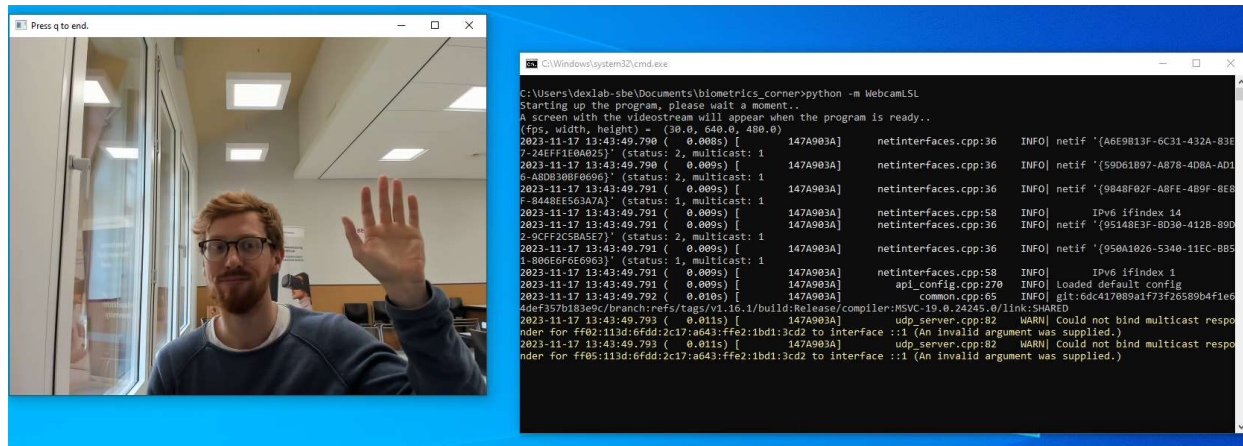


```
C:\Windows\system32\cmd.exe

C:\Users\dexlab-sbe\Documents\biometrics_corner>python -m WebcamLSL
Starting up the program, please wait a moment..
A screen with the videostream will appear when the program is ready..

2023-11-17 13:43:49.790 ( 0.008s) [ 147A983A] netinterfaces.cpp:36 INFO| netif '(A6E9B13F-6C31-432A-83E
7-24E9F1E04025)' (status: 2, multicast: 1
2023-11-17 13:43:49.790 ( 0.009s) [ 147A983A] netinterfaces.cpp:36 INFO| netif '(59061897-A878-4D8A-AD1
6-A80B308F0696)' (status: 2, multicast: 1
2023-11-17 13:43:49.791 ( 0.009s) [ 147A983A] netinterfaces.cpp:36 INFO| netif '(9848F02F-A8FE-4B9F-8EB
F-8448EE563A7A)' (status: 1, multicast: 1
2023-11-17 13:43:49.791 ( 0.009s) [ 147A983A] netinterfaces.cpp:58 INFO| IPv6 ifindex 14
2023-11-17 13:43:49.791 ( 0.009s) [ 147A983A] netinterfaces.cpp:36 INFO| netif '(95148E3F-8030-412B-80D
2-9CFF2C5B8A5E7)' (status: 2, multicast: 1
2023-11-17 13:43:49.791 ( 0.009s) [ 147A983A] netinterfaces.cpp:36 INFO| netif '(950A1026-5340-11EC-B85
1-B0E6F6E0903)' (status: 1, multicast: 1
2023-11-17 13:43:49.791 ( 0.009s) [ 147A983A] netinterfaces.cpp:58 INFO| IPv6 ifindex 1
2023-11-17 13:43:49.792 ( 0.010s) [ 147A983A] api_config.cpp:270 INFO| Loaded default config
2023-11-17 13:43:49.792 ( 0.010s) [ 147A983A] common.cpp:65 INFO| git:6dc417089a1f73f26589b4f1e6
4def357b18389c/branchrefs/tags/v1.16.1/build:Release/compiler:MSVC-19.0.24245.0/link:SHARED
2023-11-17 13:43:49.793 ( 0.011s) [ 147A983A] udp_server.cpp:82 WARN| Could not bind multicast respo
nder for ff02::1:3::6fdd:2c17:a643:ffe2:1bd1:3cd2 to interface ::1 (An invalid argument was supplied.)
2023-11-17 13:43:49.793 ( 0.011s) [ 147A983A] udp_server.cpp:82 WARN| Could not bind multicast respo
nder for ff05::1:3::6fdd:2c17:a643:ffe2:1bd1:3cd2 to interface ::1 (An invalid argument was supplied.)
```

Starting up takes while (e.g., max 1 minute). After a moment, you will see some more information about LSL stream, and a visual that shows the video that is being recorded.



- b. **Note:** this program does not directly send facial expression metrics to LSL because of technical constraints. Instead, the program records a webcam video and streams the video's frame numbers to LSL to synchronize the data. This video recording will need to

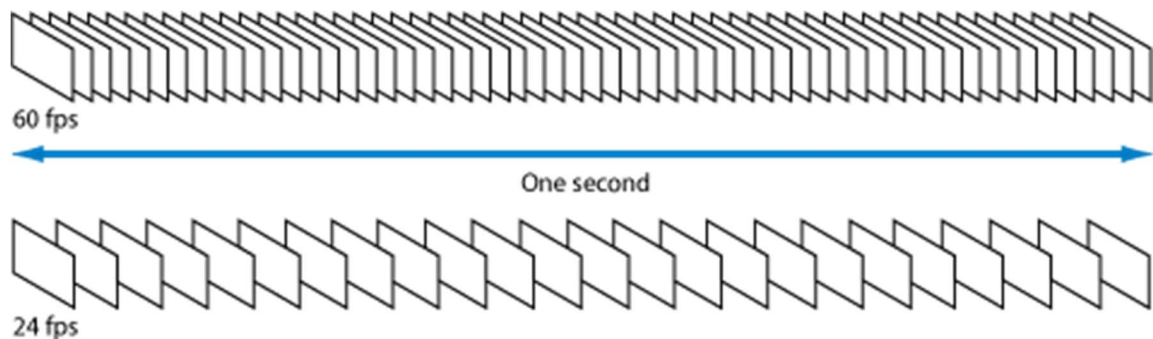
be processed separately with Noldus FaceReader after recording to compute facial expression measures (e.g., degree of frowning & smiling). You can then use the frame numbers to synchronize the data to the other signals that you have recorded.

- c. **To quit streaming the webcam data**, click on the window that shows the video stream that was opened and press Q on your keyboard to quit the program. The streaming of frame markers will stop and the program will save the webcam video in the folder.

- The webcam video recording can be found in this folder:

C:\\Users\\dexlab-sbe\\Documents\\biometrics_corner\\Webcam_recordings

- The recordings are named automatically by the date and time, for example:
 - Camera1_20231102_1905.avi
 - This information is also stored in the header file of LSL data stream. You can connect the data back to the files by using these header files.
- Facial expressions need to be computed at from these recordings per video frame by using Noldus FaceReader software, the exported metrics per video frame can be synchronized to the other data by using the frame numbers that were sent to LSL. See the data analysis section for more information.
- The raw data that was recorded looks like this below, there is a webcam video recorded at 30 frames per second (FPS) stored in the folder and the frame numbers of the individual video frames are recorded in your XDF datafile.



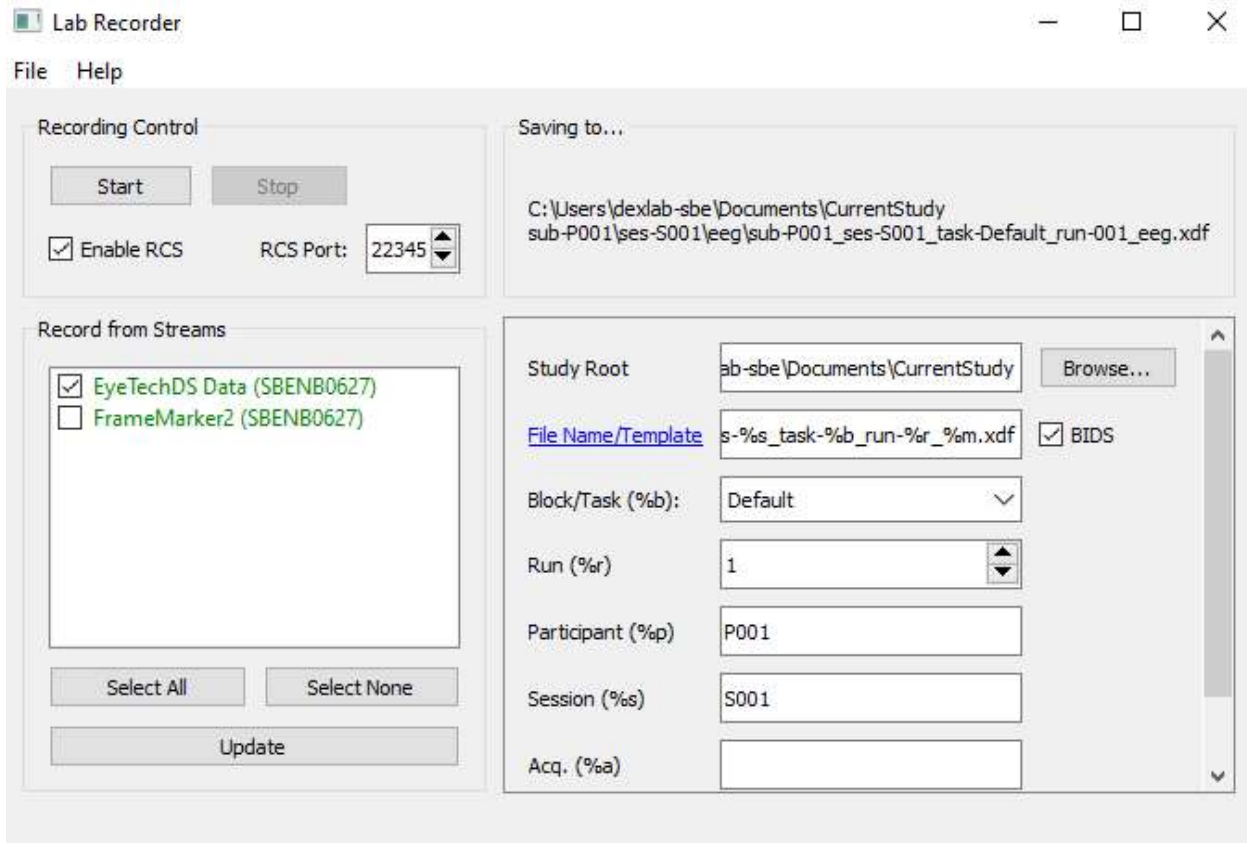
4. **Start up the PsychoPy experiment**, this adds another data stream with event markers
 - a. Click on the file “**Start template experiment.bat**” in the biometrics_corner folder to open the program, please wait a moment and you will see a grey screen like this:



- b. The experiment template has a waiting screen before starting. Do not start the experiment before starting the data recording in the labrecorder program (next step), otherwise no data will be recorded. Make sure that labrecorder is recording the data and that all necessary streams are active before starting the experiment.
 - c. **Note:** this experiment is a simple template and will need to be adapted to fit your own needs. It shows a video of a kangaroo and sends four event markers to LSL for the start and end of the experiment and the same for the video.

You can program your own experiment and specify when event markers should be sent to LSL. Please refer to the documentation and more information on www.PschoPy.org and <https://psychopy.org/coder/index.html> to learn how to program experiments with PsychoPy and make your own experiments.

5. **Start LabRecorder** to record the data from all these different data streams that you started
 - a. Click on the icon “**Start Labrecorder.exe**” in the biometrics_corner folder to open the program, you will find a screen like this:



- b. Select the streams that you want to record by checking the boxes on the left. Make sure to include string markers because these are the event markers that are necessary for data analysis to know when certain events happened/stimuli were shown).
 - If a data stream is not showing in the list, press “update” on the bottom left.
 - c. Specify in the menu on the right what the participant number is and how/where you want to save the data file (i.e., under which name, and in which folder)
 - Click “browse” to select your own folder on where to store the data
 - d. Press “start” to start recording data. The data you selected to include is now recording together into one XDF file. The size of the file should be increasing over time, check whether this is the case.
 - e. Continue with the experiment when the data is recording correctly, have the participant press the spacebar to run the experiment.

6. At the end of the experiment, sending of event markers (string markers) stops automatically

Thank you for your participation.
Please call your experimenter to
end the experiment.

- a. When the experiment is done, first press stop recording in labrecorder, wait a moment for the labrecorder to close the file, and the data is now saved in an XDF file in your specified folder. After the data is saved, please quit all running data streams.
- b. To quit the experiment, press spacebar.
- c. If you are recording eye tracking data, navigate to its CMD window, click on it, and press CTRL+C on your keyboard to quit streaming the data. Enter “Y” to confirm.
- d. If you are recording GSR data, navigate to the CMD window, click on it, and press CTRL+C on your keyboard to quit streaming the data. Enter “Y” to confirm.
- e. If you are recording webcam videos, quit recording by pressing “Q” on the video window. The video is stored in the folder called “webcam recordings” and the frame numbers have been sent to the labrecorder.
 - **Note:** You should run the (proprietary) FaceReader software to convert the recorded video into a set of facial expression measures. You can export the facial expression metrics by frame number from Noldus Facereader.
- f. After quitting all data streams, you can close the labrecorder.

7. Congratulations, you have now recorded biometric data using open source software. Good job.

To manage and store your data

- a. To save space on the lab computer and create a backup, please move your recorded data to your personal surfdrive folder or other method of storage that you prefer.
 - b. Please remove your data from the lab computer when you are done with your experiment to make sure that the computer has enough disk space and speed for future data collection.
8. You can use the example python code [in xdf_reader.py](#) to open the XDF files and examine the data for your own data analysis and preprocessing routines.
9. Profit!

3. Examining and Analyzing the data

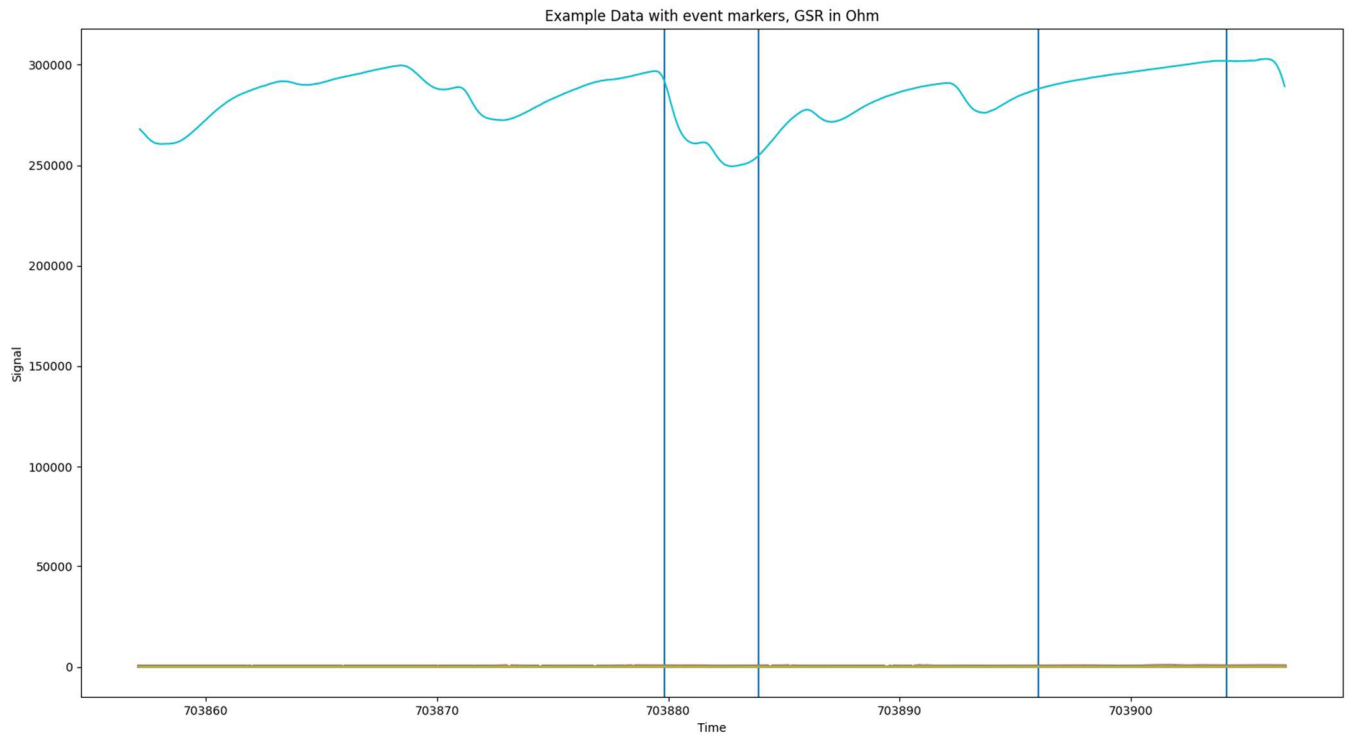
A full description of biometric data analysis is beyond the scope of this guide because there are many possibilities since every individual project has different outcomes and interests. However, the guide below provides some basic examples on how to read and view the data that was collected from the sensors. In addition, the guide provides some starting points for the analysis of eye-tracking data, GSR data, and Facial expressions. More information about analysis of eye-tracking, GSR, and facial expressions can be found online and in the documentation attached.

In general, most biometric data needs to have some preprocessing done to be useful in data analysis. There are many sources of noise or artifacts that you might not be interested in. For instance, you may need to correct for baseline differences in pupil size between participants. Some people simply have larger pupils than others, but the variation in the signal (over time) within a participant is usually the most interesting information. Another example of noise in biometric data comes from participant movement that may result in missing data or artifacts. To solve these issues, the raw biometric data needs to be preprocessed e.g., to account for individual differences (baseline), interpolate missing data, and filter out (movement) artifacts.

The event markers are needed to link the biometric data back to the experimental task. They show the “events” in your experiment, such as when a specific element was shown or when a specific task was performed. You can compare the signals corresponding to these different events to test your hypotheses. For example, you might find that your GSR measure shows a peak of arousal after the start of certain events (e.g., pictures) that indicates an arousing effect. Alternatively, you can define your task as a longer event that has a starting- and ending event marker. You can then examine what happened during the task that the participant performed.

The section below shows a couple of visual examples and some guidelines for analysis. The figures show what the data looks like out coming of the lab recorder. We also present some starting points for data preprocessing and examples of responses from example data sets. The code that was used for this example is in the file `xdf_reader.py`

GSR data

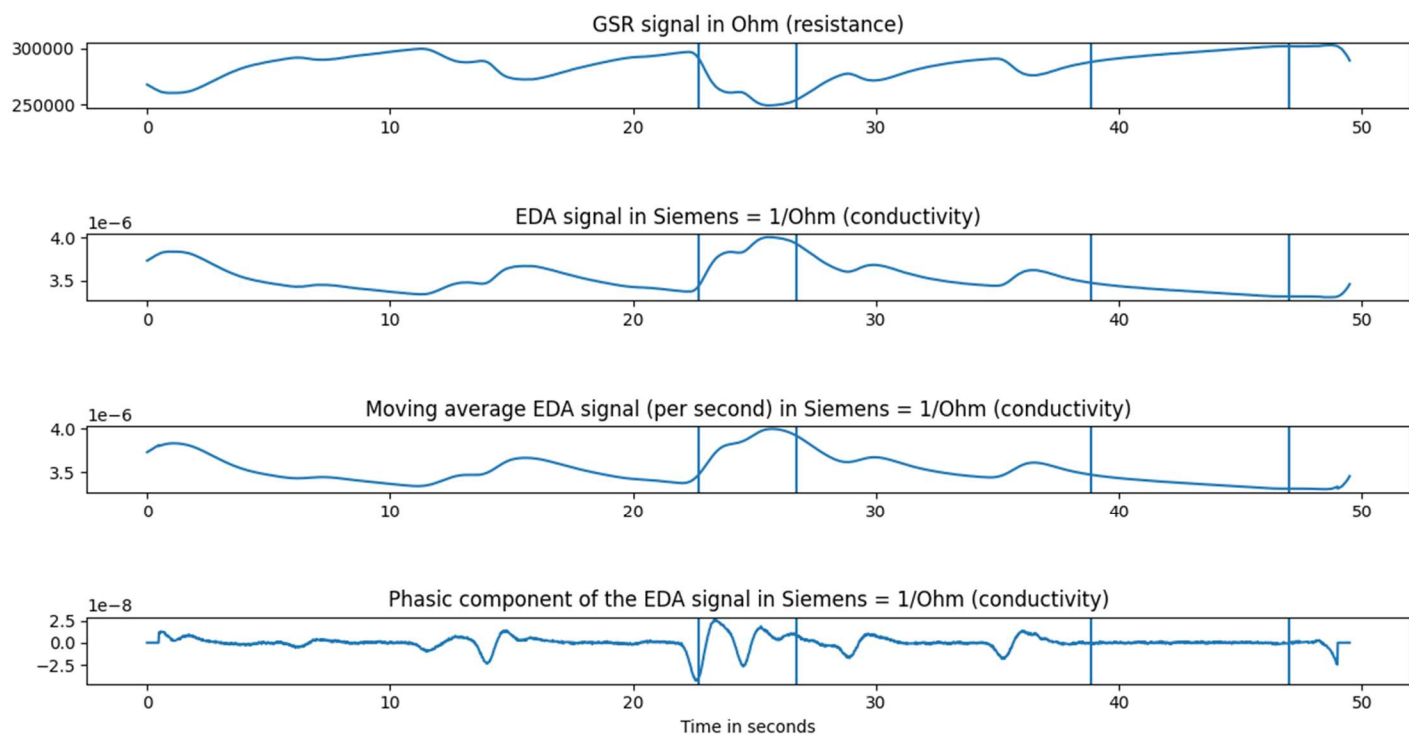


The figure above shows example data recorded over time, showing GSR data (in [Ohm](#)) and the event markers as the vertical bars. Eye-tracking data is also in there but it's squashed to the bottom because it is measured on different scales (E.g., pupil size in millimeters, or gaze position in Pixels). The first event marker on the left shows the start of the experiment, the participant was quite excited when the experiment began. There is a clear GSR response showing increased conductivity of the skin because of sweat gland activity (arousal).

You can observe changes in the GSR signal over time, but the GSR signal is relatively “slow” meaning that it takes about a second for an event to generate a GSR (sweat gland) response. Moreover, arousal levels take some time to recover from previous highs, similar to moods.

This arousal response shown in the figure above is a negative drop because it is measured in Ohm, which is a unit of electrical resistance (i.e., the opposite/inverse of conductivity). When there is more excitement, the skin becomes more conductive and resistance should go down. Skin conductance (G, measured in Siemens) is then the opposite of resistance, and it can be calculated from skin resistance (Ω) using the formula $G = 1 / \Omega$.

The GSR signal also needs preprocessing and the signal can be divided into the Tonic (slow) and Phasic (fast) component. One simple approach to get the tonic component is to use a moving average (e.g., per second) as a measure of tonic arousal. In this example, we calculated the moving average with a window of 52 samples, corresponding to one second of data with the 52 Hz GSR sampling rate. Subtracting this moving average from the original signal then gives you the phasic component (the quick peaks and dips); the panel shown at the bottom highlights some quick EDA responses that occur over 1 to 3 seconds.

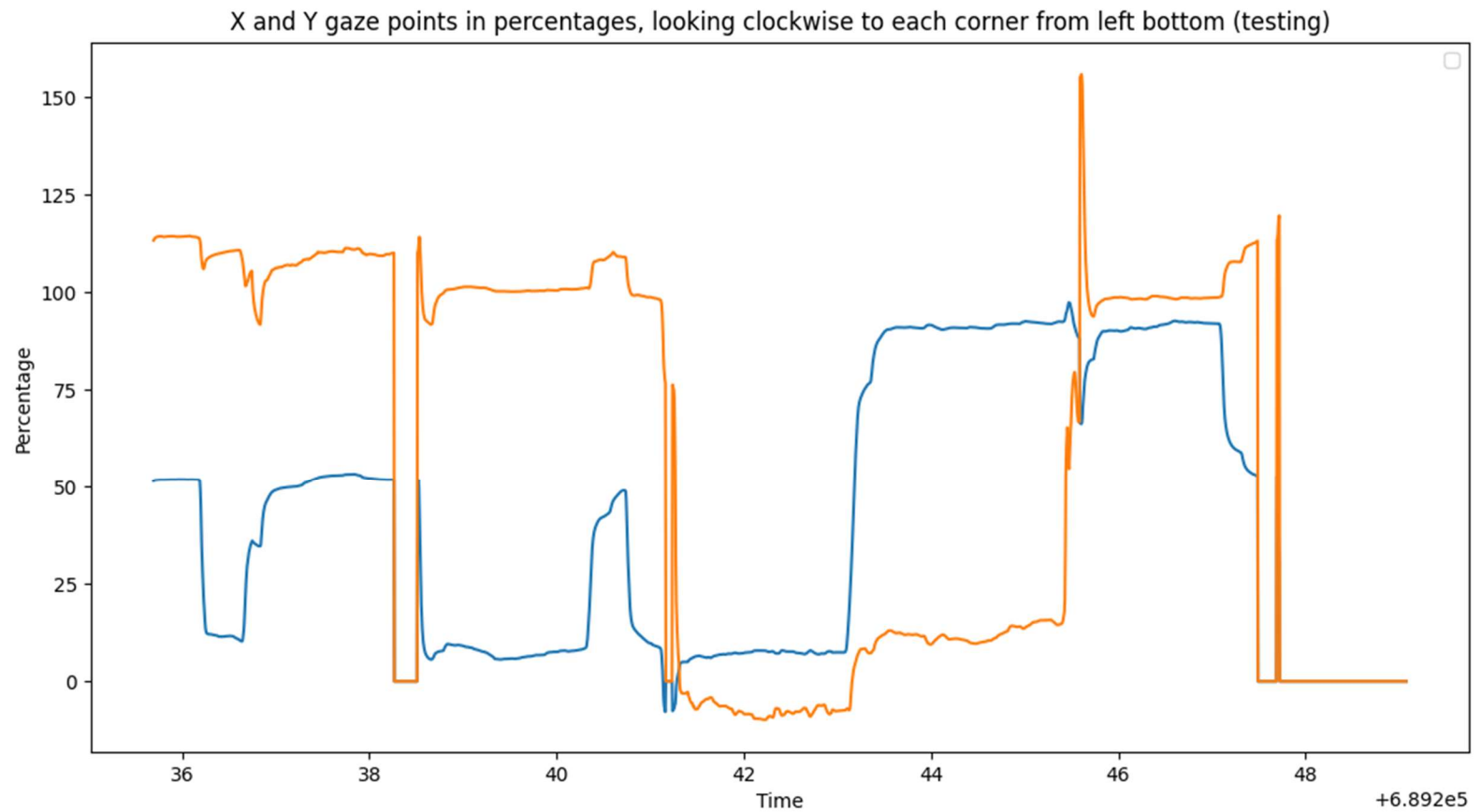


An effect shown by GSR data could then be a “phasic” (short-term) response to an event shown by a short period increased conductivity, or by an increased “tonic” (average) level of arousal differing between experimental tasks (e.g., high and low task load) for example.

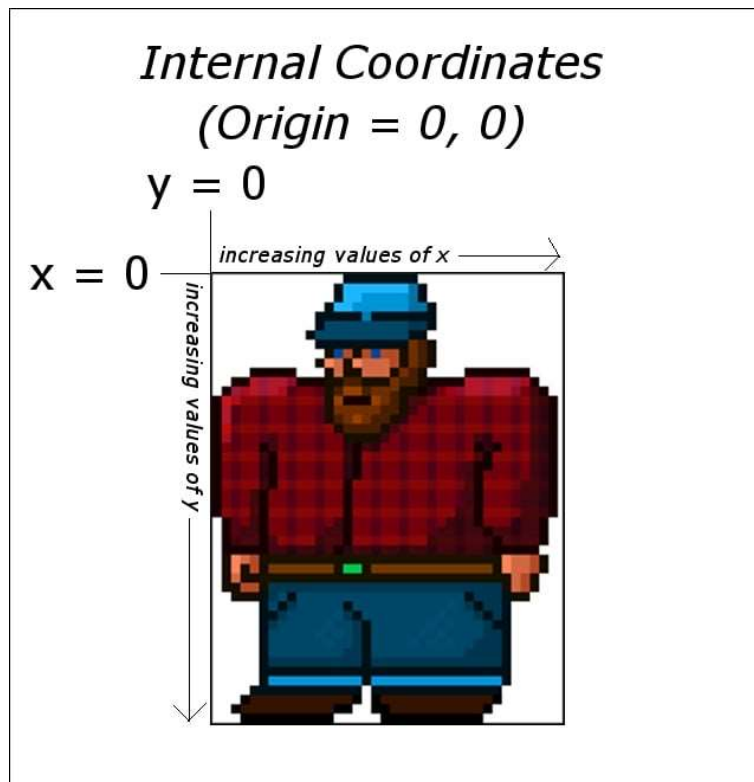
There are many different (and valid) ways to approach this problem, so a full discussion of GSR data preprocessing and analysis is beyond the scope of this guide. This is only a very basic overview to get started, but some more examples of GSR data artifacts can be found here:

https://connect.tobii.com/s/article/galvanic-skin-response-gsr?language=en_US

Eye tracking data

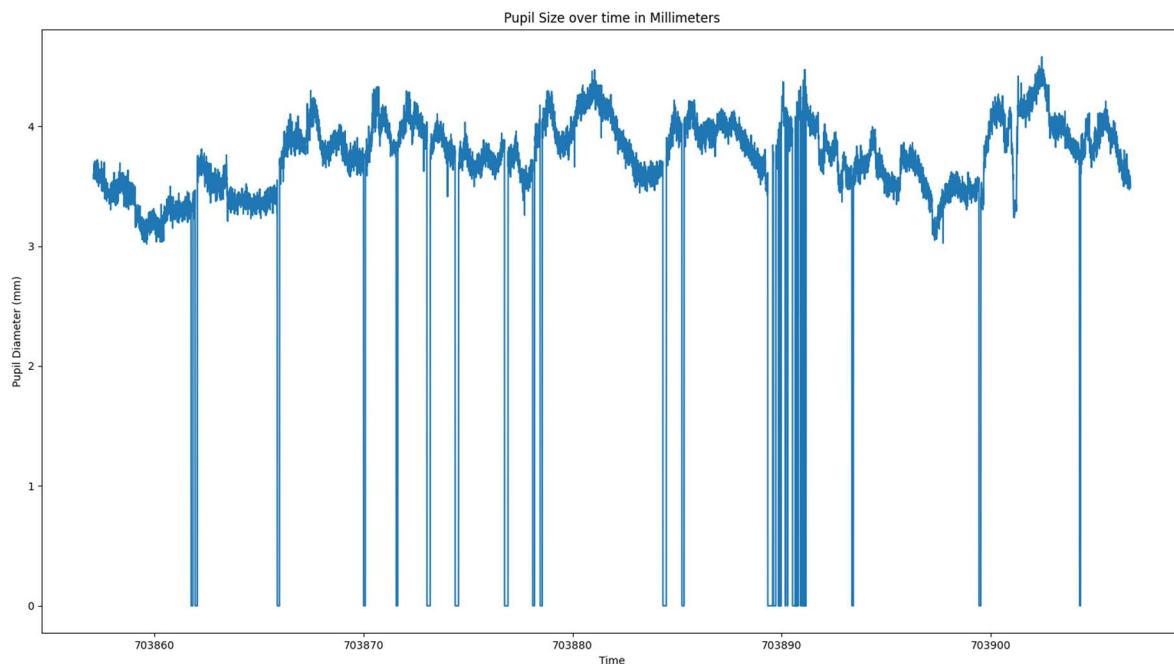


This figure above shows an example of different data, namely my own eyes as they move clockwise across the screen from left bottom, top left, top right, to bottom right. This is an example of gaze points in eye-tracking. The blue (orange) line shows the X (Y) coordinates. The units are in percentages width (height) of the screen (from 0% to 100%).



The coordinate system might also be a bit confusing, but see the example on the left here. The origin is the top-left (0,0) moving down and to the right gives higher values. The eye-tracking data has a number of different channels (total of 29 channels, see the list below for a full overview). Most of these channels are redundant (duplicates), the gaze points and pupil size data is the most interesting.

The figure below for an example of pupil size data that was also collected from the eye-tracker (different data).



As you can see, there is some variation over time. These can be due to arousal (similar to GSR) but also because of luminance. The pupil contracts with increasing light and enlarges with darkness. You can see some weird vertical bars, these are instances of missing data (e.g., due to blinks). Most biometric data needs preprocessing before data analysis, e.g., to remove blinks and increase the signal to noise ratio.

The list below provides a full list of all 29 data channels that are streamed from the eye-tracker.

[App-EyeTechDS](#) / [eyetechsl](#) / [docs](#) / [sample_data.json](#) 



[itayplav](#) Added CLI for streaming Gaze/VideoRaw data & viewing the V

Code

Blame

31 lines (31 loc) · 1.07 KB

```
1  {
2      "LeftEye_Found": 1,
3      "LeftEye_Calibrated": 1,
4      "LeftEye_PupilDiameter": 3.5422542095184326,
5      "LeftEye_Pupil_X": 1160.5,
6      "LeftEye_Pupil_Y": 312,
7      "LeftEye_Glint0_X": 1152.760009765625,
8      "LeftEye_Glint0_Y": 317.32000732421875,
9      "LeftEye_Glint1_X": 1160.25,
10     "LeftEye_Glint1_Y": 317,
11     "LeftEye_GazePoint_X": 49.20669937133789,
12     "LeftEye_GazePoint_Y": 123.4298324584961,
13     "RightEye_Found": 1,
14     "RightEye_Calibrated": 1,
15     "RightEye_PupilDiameter": 3.8945322036743164,
16     "RightEye_Pupil_X": 760.1875,
17     "RightEye_Pupil_Y": 354.6875,
18     "RightEye_Glint0_X": 755.5,
19     "RightEye_Glint0_Y": 359,
20     "RightEye_Glint1_X": 762.6571655273438,
21     "RightEye_Glint1_Y": 359.028564453125,
22     "RightEye_GazePoint_X": 50.30195617675781,
23     "RightEye_GazePoint_Y": 111.88605499267578,
24     "WeightedGazePoint_Valid": 1,
25     "WeightedGazePoint_X": 49.754329681396484,
26     "WeightedGazePoint_Y": 117.65794372558594,
27     "WeightedGazePoint_LeftWeight": 0.5,
28     "WeightedGazePoint_RightWeight": 0.5,
29     "Focus": 6.897058963775635,
30     "Distance": 74.89541625976562
31 }
```

Finally, for eye-tracking data, eye movements are often classified into fixations and saccades. Fixations are moments where the eyes are focused on something in space and the eyes remain relatively still. In contrast, saccades are quick eye movements where people switch their attention to something else. This classification of raw eye-tracking data provides some useful statistics such as fixation durations and fixation counts that are often used in publications. The rationale is that the more (fixation count) and longer (fixation duration) people view something, the more interested they are in it.

A simple algorithm to classify saccades and fixations is the [velocity threshold algorithm](#). As the name implies, it has a certain velocity threshold for classification purposes. When the eyes remain still for a fixation, the velocity is (near) zero and when they move (saccade) the velocity is higher than zero. When the velocity is above the threshold, big eye movements are classified as a saccades and periods of small movements are classified as fixations. This algorithm can be applied but there are many other options.

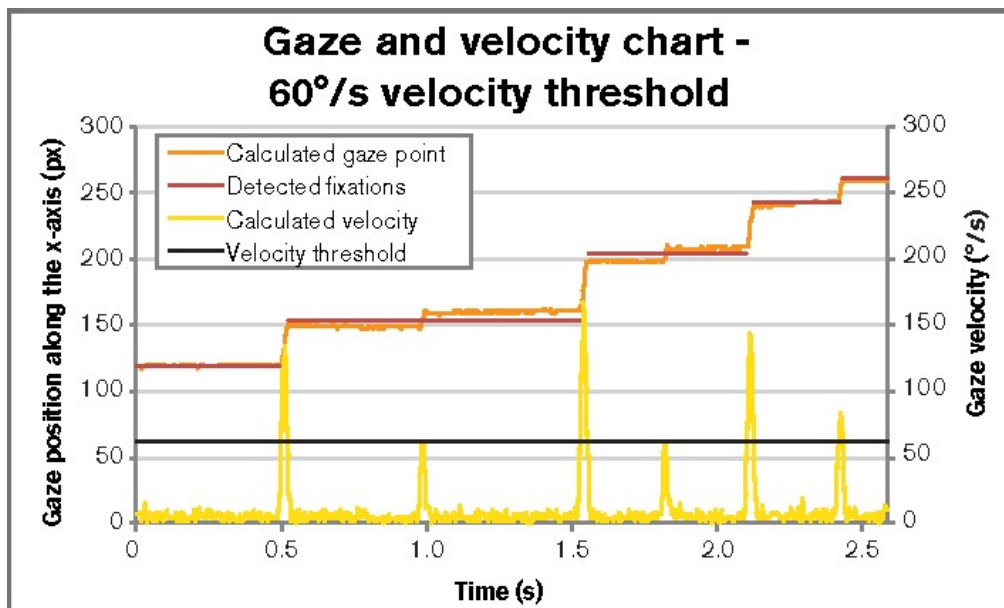


Figure 8. Chart showing the eye movement for one participant along the

There are also more complicated cases in fixation classification such as smooth pursuit, when people are focusing/fixating on an object that moves. To keep things simple, I would advise to use still images to start with, otherwise you can rely on the gaze points (X, Y coordinates) in videos instead of using fixations/saccades. Next to fixation durations and counts, there are many other metrics that can be computed from eye-tracking data including but not limited to:

- Fixation sequences (ordering of fixations across objects/areas 1st, 2nd, 3rd etc.)
- Attentional exploration (spread out) versus exploitation (focused); e.g., coefficient K
- Attentional synchrony (similarity in the attention of a group of individuals)
- Pupil size (a measure of arousal)
- Areas of interest (AOI's), seen or not (binary), time to first fixation (salience), time spent
- Heat maps of attention (visual representation)
- Many other options

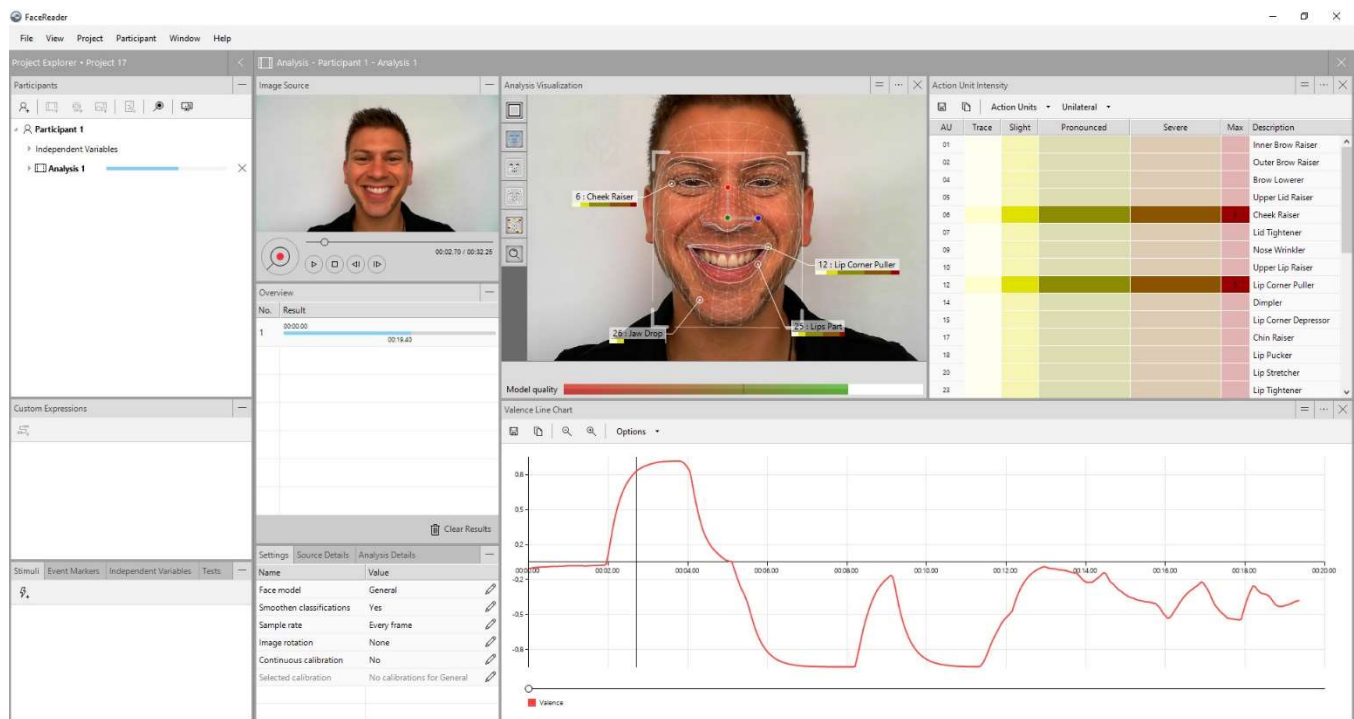
See more ET data examples here: <https://imotions.com/blog/learning/10-terms-metrics-eye-tracking/>

Facial Expression Analysis with Noldus Facereader

Finally, converting the webcam video recording to facial expression metrics with facereader is relatively easy using the user-friendly software from Noldus, see the demo video here for [example](#).

In this example you can see the graphical user interface (GUI) of the Noldus Facereader program. You can select which metrics you want to record in the GUI and you can replay the video to see how the emotional responses are scored over time. If you press “analyze”, it work automatically.

The window on the right here shows the activation level in the various “facial action units” upon which the facial expression metrics of the Noldus software are based. You can choose whether to use the aggregated metrics such as “Happy” or “Suprise” or whether to use the raw facial action units.



After your analysis in Noldus Facereader, you can export the facial expression metrics per video frame from the Noldus program. This can then be linked back to your other data by using the frame numbers.

The analysis is similar to ones before, but the data can be quite noisy and it is difficult to get a strong and explicit emotional reaction from your participants. Common practice in the literature is to aggregate the data across participants, by looking at which percentage of the participants shows a “happy response” per point in time for example. This practice can improve the signal-to-noise ratio.

For more information, see Noldus documentation and the white papers in the documentation folder.

Alternatively, there is also free and open-source software available to do facial expression analysis, such as LibreFace which you can find here: [LibreFace: An Open-Source Toolkit for Deep Facial Expression Analysis \(arxiv.org\)](#) and here [GitHub - ihp-lab/LibreFace: \[WACV 2024\] LibreFace: An Open-Source Toolkit for Deep Facial Expression Analysis](#)

4. Final Notes:

- A. You can use and adapt the PsychoPy experiment template to your own needs, the template is very basic; it presents some stimuli (video/pictures), runs a questionnaire, and sends markers to LSL for marking the different blocks in the data. This will need to be adapted to your own needs. Please copy this code and create your own folder instead of directly changing the source code. More information about PsychoPy can be found here: <https://www.psychopy.org/> and <https://psychopy.org/coder/index.html> to learn how to program experiments with PsychoPy and make your own experiments. The important thing to remember is that you need to include event markers to link your experiment to the biometric data. Examples of how to do this are shown in the template experiment.
- B. If you want to use GSR in combination with virtual reality or augmented reality experiments, this can be done more easily with the Consensys software (i.e., licensed shimmer firmware). The other equipment is not compatible with VR experiments. When using the GSR like this, you can simply start recording straight from the Consensys software (installed on the desktop).
- This solution is preferable for running VR (or mobile) experiments instead of screen-based experiments, but you would need to organize the markers in another way (e.g., you can simply have a separate data file for every run/block). Data can be recorded straight from the Consensys software which has a relatively simple to use graphical user interface (GUI) and you can name the data files accordingly.
 - See Shimmer Documentation on Consensys for more detail.