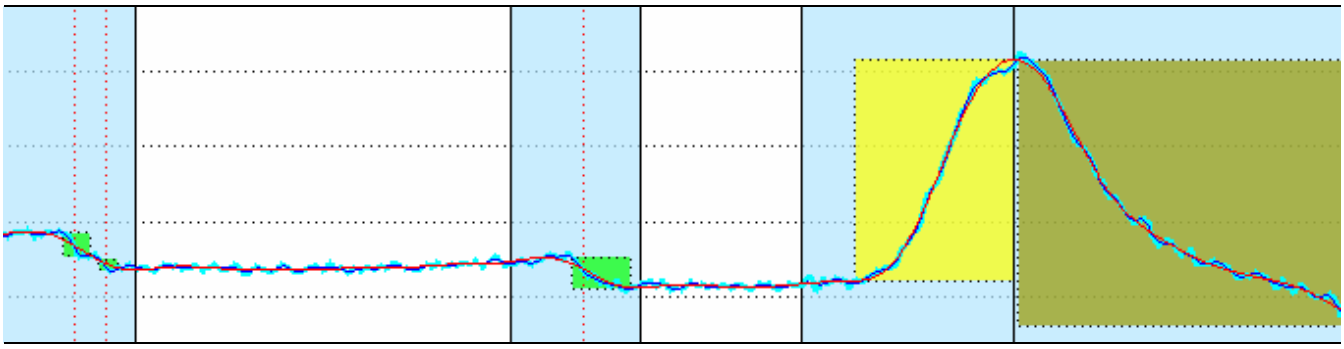
eogui – a Matlab software to analyze electro-oculogram (EOG) recordings

Software written by Maik Hofmann[1], Robert Schleicher[2], Niels Galley[3] & Martin Golz[1]
Manual by Robert Schleicher

[1]University of Applied Sciences Fachhochschule Schmalkalden
[2]Deutsche Telekom Laboratories, TU Berlin
[3]University of Cologne



1 Overview

Eogui is a free Matlab software with a graphical user interface (GUI) to analyze electro-oculogram (EOG) recordings. It detects saccades in the vertical and horizontal channel as well as blinks in the vertical channel of the EOG.
Each event (i.e. blink or saccade) is marked in the raw signal and described with a number of parameters like start, end, amplitude, maximum velocity etc. that can be stored in various formats (*.mat, ASCII etc.). The GUI allows to browse through the raw signal with events marked and to adjust the parameters for analysis. All settings are stored in a configuration file that can be used for the next data set. Batch processing is also possible.

The manual is structured as follows: Chapter 2 tells you how to calibrate and analyze EOG recordings with eogui. This is the main focus of this text. We tried to keep it short so that you can start quickly and only described the most common options. Chapter 3 gives more details on accepted input formats as well as possible output formats. Chapter 4 provides some background information on the underlying algorithm and additional parameters used. Unlike the previous chapters, that chapter is not complete, but rather work in progress. We might provide an updated version at some point.

1.1 Limitations

Before you continue reading, we will briefly describe the capabilities as well as limitations of eogui and this text.

1.1.1 Type of eye movements and signals

eogui was developed to analyze two types of eye movements, namely saccades and blinks that were recorded with the EOG using a separate channel for horizontal (x) and vertical (y) movements (see examples 1 & 2 p. 15). It does not cover any other types of eye movements like smooth pursuit eye movements (see example 3 p.16) or changes in pupil diameter. The latter is not detectable in the EOG due to its physiological roots, which are also not explained in this text, as it is just a manual for the software. If you wish to learn more about these topics, we give you a couple of references at the end, which also address the general limitations of EOG recordings.
While it would in principle be possible to analyze camera-based eye movement data that consist of a separate x and y channel with eogui, we have not done it yet and do not plan to do so in the near future, which leads to the next paragraph.

1.1.2 Technical issues and support

This software comes as is, i.e. with no additional support. So please refrain from sending us respective emails. Based on our experience with using it for the first time, there are two things we would like to point out:

- we included an example EOG file (EOG.mat) in the zip archive. Try whether your eogui installation processes it. If not, it's most likely something with the paths to the eogui files or with missing toolboxes in your Matlab installation. If the example file is accepted, but not your data, try to arrange your data in the same way like the example to let eogui read them. This means a Matlab data matrix named 'data' with 3 colums: TIME XVALUE YVALUE (see also chapter 3.1.1 p.11)
- to meaningful analyze your data with eogui, you need to do the calibration AT THE BEGINNING of each recording as described in the next chapter. If this has not been done, you can try to determine some workable default parameters by guessing distance to screen and the amplitude of certain large saccades you know the subjects were likely to make (e.g. line breaks while reading the instructions on screen), but all results have to be handled with care.

2 Quick guide to eogui

2.1 Calibrating the gaze at the beginning of a recording

The raw unit of EOG values is microvolt, which are amplified and then converted into digital numbers by an A/D converter. However, what you are usually interested in is the change in visual angle that is caused by a saccade, which is measured in angular degrees. As the conversion factor varies, you need to do a calibration at the beginning of each recording to be able to translate AD value changes into angular degrees. For eogui, this is done by letting the subjects look repeatedly at predefined points from left to right and up and down, while the experimenter writes down the distance between the points and the distance from the subjects to the points. Figure one shows the three values that need to be known.

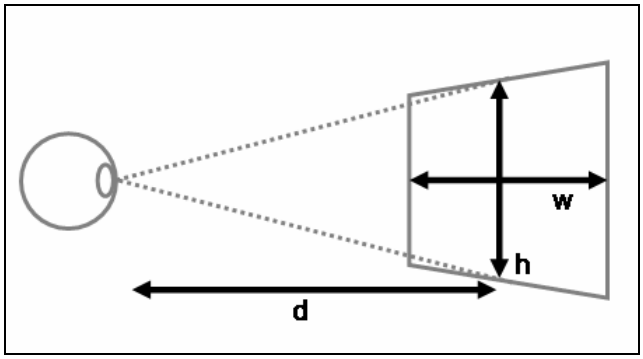


Figure 1: required information to translate AD values of reference saccades into visual angles. d = distance of viewing plane to eye; h= height of target object for vertical saccade; w = width of target object for horizontal saccade

We typically set these points by putting small stickers on the left and right side as well as top and bottom frame of the screen and ask the subject to look to the left sticker, to the right sticker, to the left sticker, to the… (see examples 4&5 p.16)
As the distance between the points is fixed across all subjects, all we need to measure for each run is the distance from the subject's eye to the screen.
You will later need these information to set the parameter for analysis, where you can easily identify these paced saccades in the raw EOG as some kind of square wave signal.

2.2 Analyse EOG recordings

To start eogui, extract the zip file and include it with all subdirectories in your Matlab search path. Then simply type eogui in the Matlab command window and press enter.

2.2.1 Load file and select range

The first step is to select a file with the EOG recordings. The input formats eogui accepts are covered in 3.1 (p. 11), the default is a Matlab data matrix called 'data', which is stored in a Matlab *.mat file. To choose another format, use the file type drop-down box.

2.2.1.1 Select signal range to be analyzed

In case of longer recordings, you may not want to analyze the complete file, but just a certain range. This can be specified here

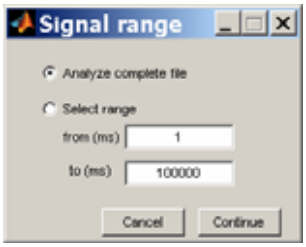


Figure 2: Select signal range

By giving the start an the end time of the recording section to be analyzed. Please note that the unit is milliseconds.

2.2.1.2 Assign channels

In the next step, you have to tell eogui which channel of your recording is the horizontal and which one is the vertical channel in case you did not follow the suggestion to arrange your data matrix such that the horizontal is the second column and the third column corresponds to the vertical channel.

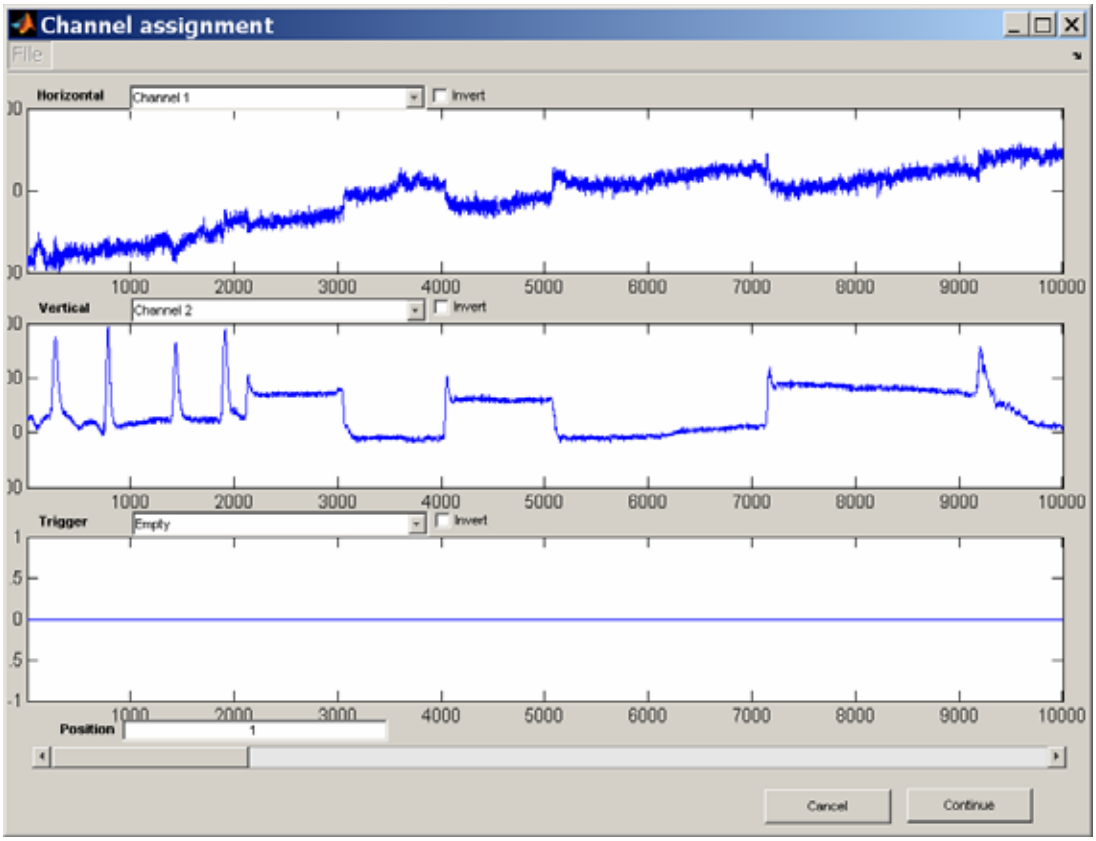


Figure 3: Assign which channel of your data corresponds to the horizontal EOG and which on to the vertical. The third plot shows an optional trigger channel, which is empty in this case.

Figure 3 shows the interface. You can see that the second plot contains a couple of sharp, upwards peaks at the beginning (around ms 1000). That is how eyeblinks look in the vertical EOG. As they are easy to spot, they can serve as an indicator which channel is the vertical one. In case you accidentally switched electrodes while recording the vertical EOG, the peaks will not point upwards, but downwards. Then mark the 'Invert' checkbox to correct this mistake. Blink events must always point upwards, otherwise eogui will not detect them in the raw signal.
There is also a third channel plotted. This is an optional trigger channel that we use from time to time. We do not deal with this in the quick guide.

2.2.2 Setting the parameters for analysis

Once the file is loaded, you need is to set the parameters for extracting blinks and saccades from the raw signal. Eogui includes a GUI wizard that leads you through setting the parameters relevant for analysis. These are basically:

- information to translate the saccadic amplitude in the AD value raw signal into angular degrees (see calibration)
- the amount of noise in you signal
- a reference blink to indicate what events are considered as blinks in your vertical EOG

The last dialogue box of the GUI wizard will give you a summary of the values you just set. All the parameters you entered are stored in a file called 'lastpara.config'.
In case you want to use the parameters of a previous analysis, you can press 'Yes' in the following dialogue window

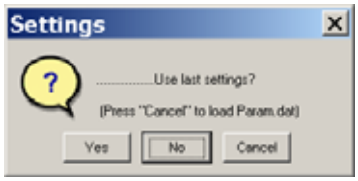


Figure 4: Use settings from last analysis?

The default answer is 'No', meaning that you start setting the parameters anew. As usual, 'Cancel' terminates eogui and lets you return to the Matlab command window. Don't worry about the statement in brackets about the Param.dat, this regards only an old file format we used a lot in the past.[1]
In the following, we assume that you pressed 'No' to determine the parameters for analysis manually.

[1] In case you loaded an EOG recording in the 'Königstein-Format' (Idat.dat), 'Cancel' will lead to another 'File Open'-dialogue, where you can choose the corresponding 'Param.dat'. If you press 'Cancel' in the file selection dialogue, you return to the Matlab command window.

2.2.2.1 Step 1: Determine visual angle

The first step is to enter the viewing distance measurements you did at the beginning of your recording session. The dialogue box looks like this:

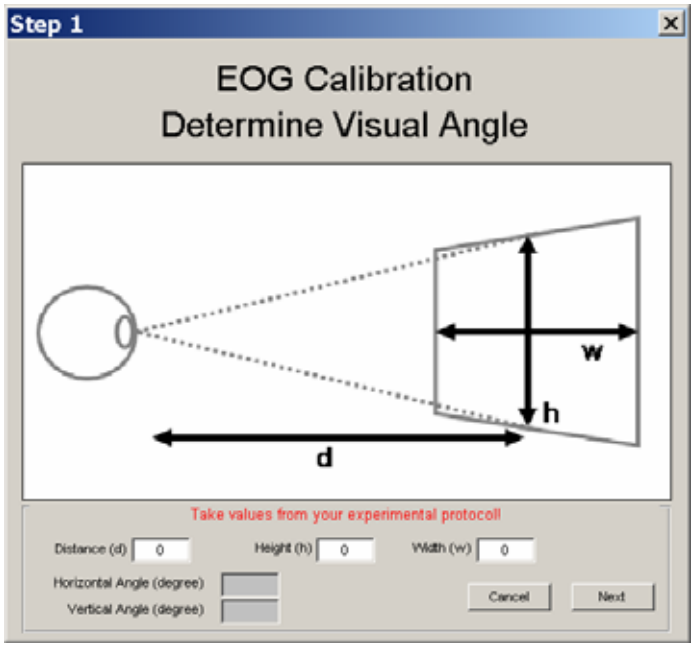


Figure 5: First Step of Calibration: enter viewing distance as well as the distance between the target points for the horizontal and vertical reference saccades called width and height here.

Matlab might require you to press 'Enter' after you entered a number to start calculating the horizontal and vertical angle. Please note that the boxes for the resulting angles are greyed out as you can not enter them directly.

2.2.2.2 Step 2: Mark horizontal reference saccade

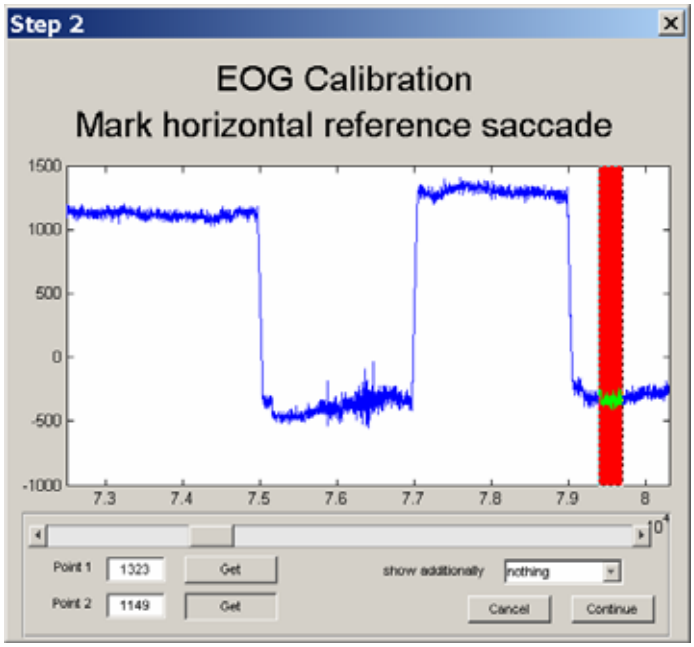


Figure 6: Second Step of Calibration: Mark the parts in the raw signal where the eyes were resting prior and subsequent to one of the horizontal calibration saccades. The saccades are visible as step-like jumps.

Now you have to tell eogui how much the signal changed when the test subject performed the required calibration saccades to fixed points from left to right (horizontal). The saccade itself is clearly visible as a step-like jump in the raw signal (see Figure 6), whereas the time when the eyes were focussing on one of the target points is the noisier, flat part in the signal. Choose an segment *before* the saccade by moving the mouse over the signal and keeping the left mouse button pressed. The segment is marked in red and the average signal value of that area will be shown in the text field labelled 'Point 1'. Then use the 'Get'-button next to 'Point 2' to set the value for the signal subsequent to the saccade and mark an area in the signal *after* the saccade. Eogui calculates the difference between these two values. As the corresponding change in visual angel is known (see Step 1 of the calibration proc-

ess), eogui can no determine what change in the AD values of the raw signals corresponds to what change in visual angle measured in degrees. This coefficient is later called AD2deg.

2.2.2.3 Step 3: Mark vertical reference saccade

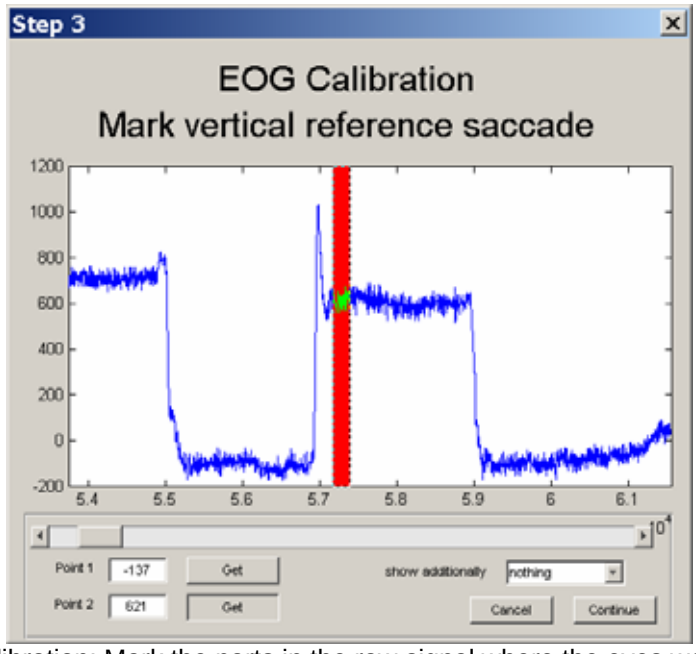


Figure 7: Third Step of Calibration: Mark the parts in the raw signal where the eyes were resting prior and subsequent to one of the vertical calibration saccades. The saccades are visible as step-like jumps.

As for the horizontal reference saccades, you also have to set the values prior and subsequent to a saccade from top to bottom (vertical). Move the mouse over the corresponding signal area while keeping the left mouse button pressed. The average signal value in that area is used for Point 1. Press the 'Get'-Button next to 'Point 2' to mark a section of the signal after the vertical signal jump the same way. As you can see in the example, the end of the vertical saccade is accompanied by a blink, visible as the sharp peak around x=5.7. Mark a section after that blink to obtain valid signal values as it is done in Figure 7. Again, because the change in visual angle is known from step 1, eogui can now determine what change in signal AD values corresponds to how many degrees change in visual angle.

2.2.2.4 Step 4: Determine x noise

Now you have to set the noise level for both channels x and y, i.e. state to what extent signal changes do not represent an eye movement event, but occurred randomly.

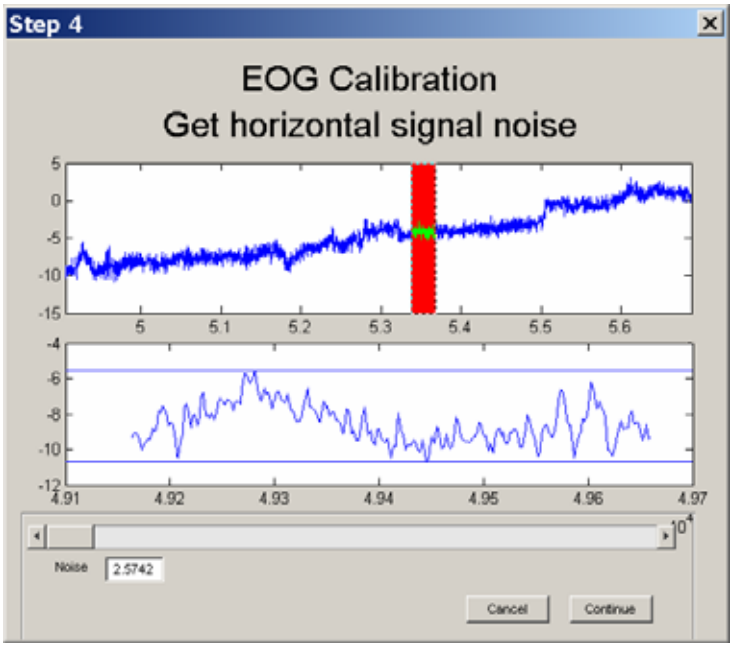


Figure 8: Fourth fifth step of calibration: Mark nose level in the signal. The upper part shows the raw signal, the lower part the first derivative.

As before, you can move the mouse cursor with the left mouse button pressed over a signal area to select a representative part which will be marked in red. The lower part of the dialogue box shows the first derivative of the signal. Alternatively, you can also drag the horizontal lines in that part to adjust the noise level.

2.2.2.5 Step 5: Determine y noise

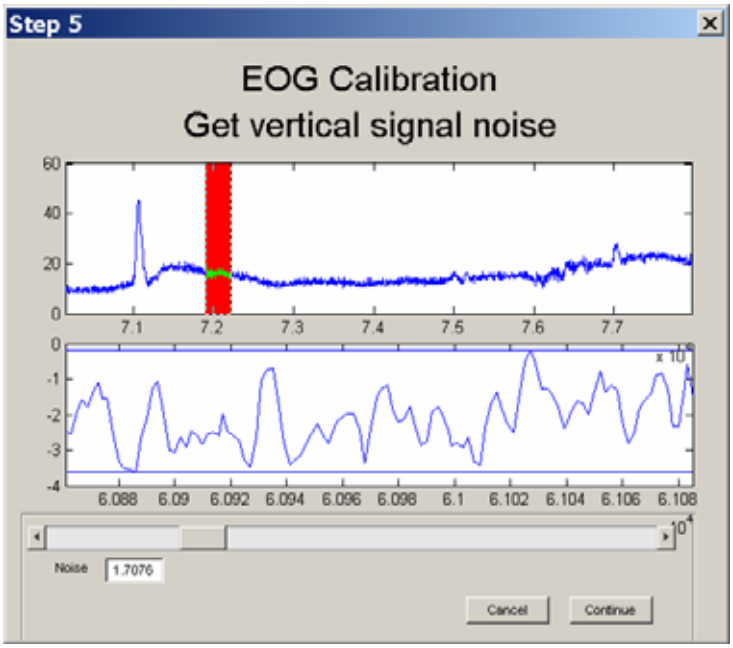


Figure 9: Fourth fifth step of calibration: Mark nose level in the signal. The upper part shows the raw signal, the lower part the first derivative.

The procedure for the vertical (y) channel is the same: either mark an area in the raw signal (upper graph in Figure 9) with the cursor while keeping the left mouse button pressed or adjust the limits in the lower figure displaying the first derivative by dragging the blue horizontal lines with the mouse cursor. Unlike for the horizontal channel, you have to make sure that you do not include a blink (visible as the sharp peak around 7.1 in the upper graph of Figure 9) in your selection, as this event is not noise, but a target of analysis. Blinks are also the scope of step 6 of the calibration.

2.2.2.6 Step 6: Select reference blink

Blinks are visible in the vertical channel of the EOG as sharp peaks. To distinguish them from other signal changes, you need to select an exemplary blink, whose amplitude is the reference for identifying them. So it is advisable not to choose the largest blink you see, but an average one.

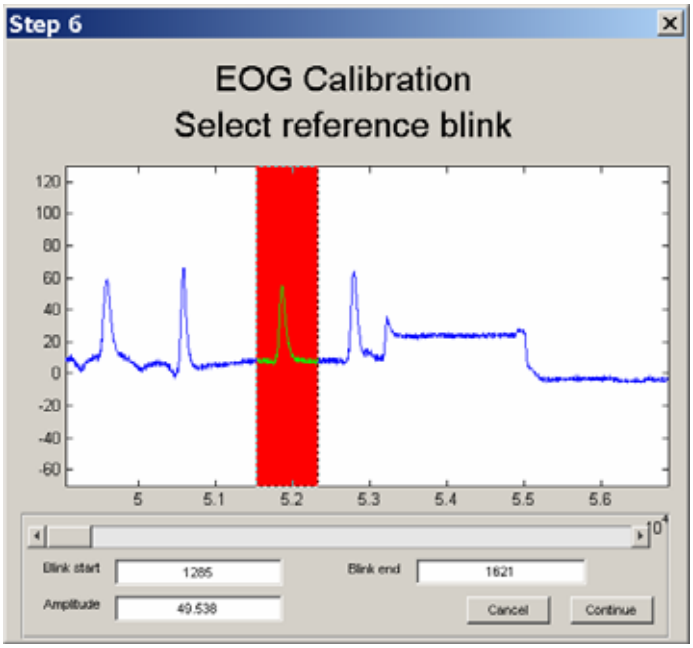


Figure 10: Sixth step of calibration: select a reference blink to let eogui know what thresholds of amplitude and movement speed are minimum for a blink .

Please note that eogui expresses the blink amplitude in angular degrees of saccades, which may not be the 'real' amplitude of a blink. Thus you can only use this information for relative comparisons ("amplitude under condition A larger than in condition B"), not as absolute values.
Even if you chose a rather small blink as the reference, it may still be the case that the blinking behaviour of your subjects changes during the experiment, for example that they start to make smaller and slower blinks due to fatigue. To account for this change, the next dialogue window allows you to set a 'softening factor', which will attenuate the limits that were based on the reference blink.

2.2.2.7 Parameter Summary

Once the calibration is finished, you will see a summary of the parameters you just set and the ones eogui will use for signal analysis. We will briefly describe them from top to botton and left to right. Most of the entries are based on the choices you made during calibration. If you want to revise them, press the 'Get'-button and the corresponding dialogue box from step 1 to 6 of the calibration will pop up.

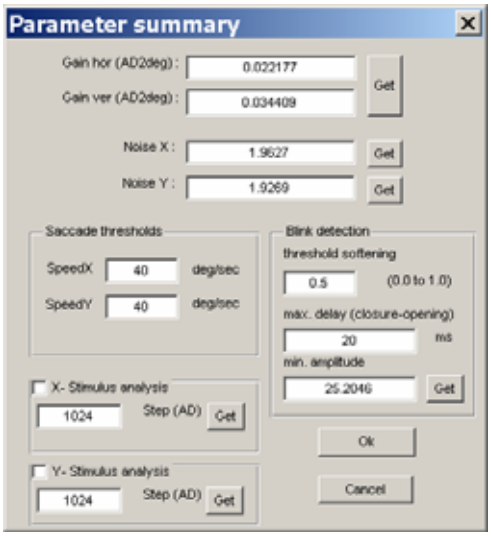


Figure 11: Parameter summary: overview of the parameters set in the calibration process and the values used for analysis.

The two values labelled 'Gain' (horizontal and vertical) are the conversions factors to translate signal changes in AD values into changes in terms of angular degree. These were determined from the information you provided in Step 1-3 of the calibration processes.
The Noise X and Y values are based on your selection in step 4 and 5 of the calibration: Signal changes smaller than these values are not considered as meaningful events (i.e. saccades or blinks), but discarded as irrelevant. If you have the impression that eogui missed some saccades in the later analysis, lower these values.
The box 'Saccade thresholds' sets the minimum speed eogui requires for a signal change to identify it as a saccade. These values are based on our experience. If you have the impression that eogui missed some saccades in the later analysis, lower these values to e.g. 30 degree/second.
The box 'Blink Detection' shows you the minimum amplitude of a blink based on your selection in step 6 of the calibration. It may be the case that the blinking behaviour of your subjects changes during the experiment, for example that they start to make smaller and slower blinks due to fatigue compared to the part of the recording where you took the reference blink from. To account for this change, you can set a 'softening factor', which will attenuate the limits that were based on the reference blink. Softening=1 means no softening, a value of 0.5 means 'take half of the original threshold'. As eogui considers blinks to appear in the vertical EOG like an upward saccade immediately followed by a downwards saccade, this factor also softens the minimum speed for blinks, which is otherwise identical to the saccade thresholds.
Another factor influencing the detection of blinks is the maximum delay allowed between complete closure (the 'upward saccade') and re-opening (the 'downward saccade'). It is by default set to 20ms, but again, tired subjects may show longer delays. You can increase this value here, too.
The boxes for X- and Y- stimuls analysis are irrelevant to you.

2.2.3 Result Browser

After you set all the parameters, the analysis starts, which may take a while based on the length of your recording and the machine you are running eogui on. Once it is done, two new windows pop up: A plot that shows the horizontal and vertical raw signal with identified saccades and blinks marked, and a second smaller window that serves as a control to browse through the plot.

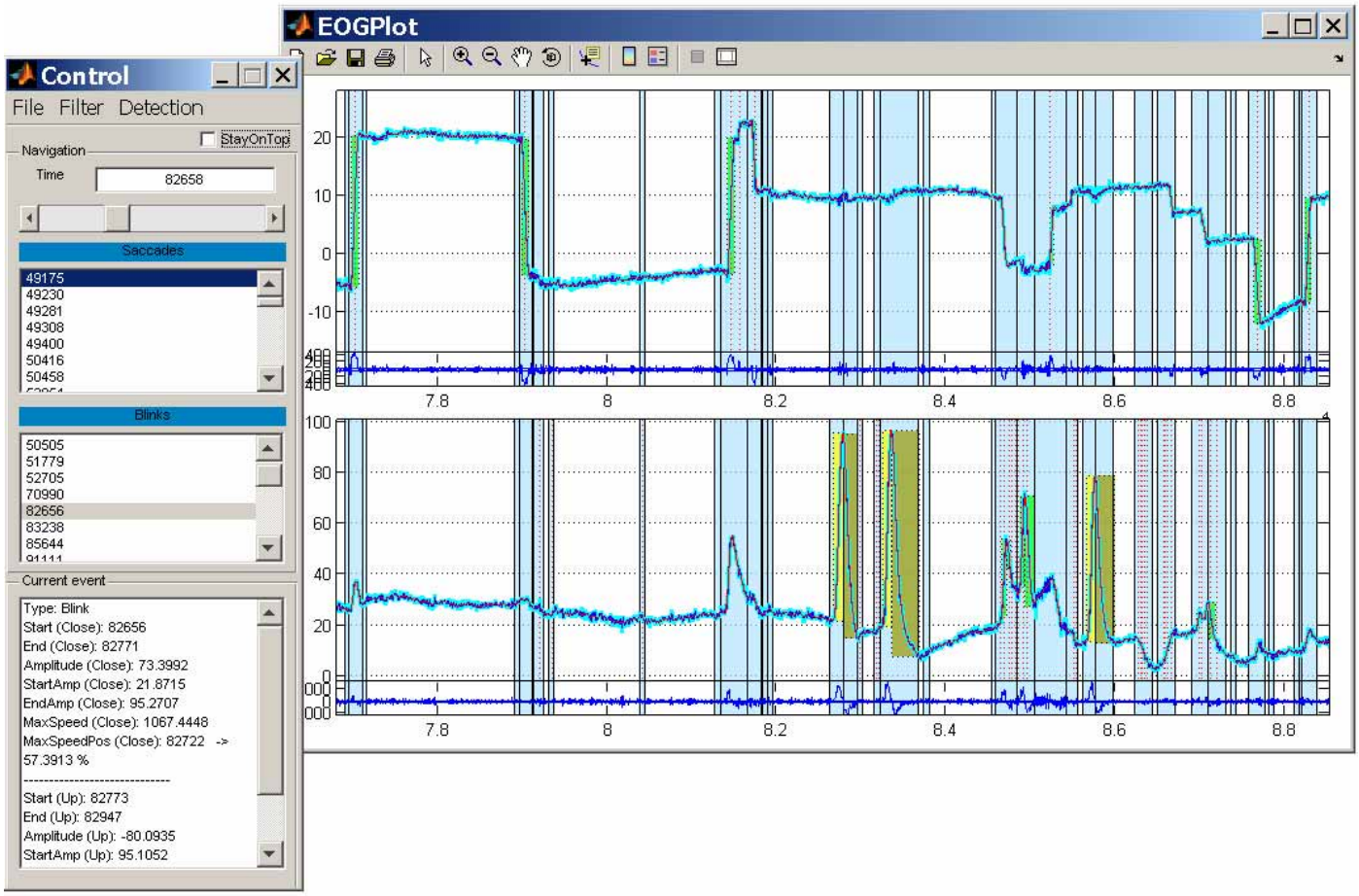


Figure 12: The Control window (left) and the EOGPlot window (right) to inspect the results of eogui's blink and saccade detection. The upper plot shows the horizontal, the lower plot the vertical channel of the EOG. Valid events are marked with yellow (blinks) respective green (saccades) squares in the raw signal. If an event is visible in both channels, it is marked in the signal where the amplitude was larger. For more information please see the text.

2.2.3.1 The control window

The control window shows you a horizontal slider that scrolls through the plot if you move it. Alternatively, you can enter a specific time in milliseconds you want to jump to. The lower parts of the window list you all detected events, i.e. blinks and saccades with their respective start time. If you select an event in either of the two lists, the plot jumps to that time (probably with a slight delay) and the lowest field of the control window displays you the parameters for that event like start & end time, amplitude, maximum velocity etc. Updating the plot moves it on top of your screen such that it may occlude the control window. If you activate the check box 'StayOnTop', the control window will always be on top of your screen[2].
The menu bar of the control window has the entries:

- File: Save As and Quit. Quit terminates the program without saving the results, Save as lets you save all detected blinks and saccades. Again, various formats for saving are offered. The default is a matlab matrix. More information can be found in the chapter 3.2 (p. 11).
- Filter: This dialogue allows you to define logical filters that are not applied to the raw signal, but filter out the detected saccades. You can do this just as well outside eogui with the statistic software of your choice once you saved all events with eogui.
- Detection: here you can check and revise the parameters you used for analysis. 'Parameters' will show the summary window described in chapter 2.2.2.7 (p.8). If you wish to redo the analysis with adjusted parameters, change them in the summary window, close it, and select Detection → Repeat Analysis

2.2.3.2 The EOGplot window

This window shows you two plots: the upper one is the horizontal EOG channel, the lower one the vertical channel. The cyan lines are the raw signal, the blue lines the slightly low-pass filtered raw signal and the red line the strongly low-passed signal (see chapter 4.1, p.13 for more information on that topic). The blue line at the bottom of each signal is the first derivative, i.e. the speed of the slightly low-pass filtered signal.
Detected saccades are marked with a green square either in the horizontal or vertical channel, depending which component contributed more to the overall amplitude as most saccades are not just horizontal or vertical, but a mixture of both dimensions.
Blinks are marked with a yellow square in the lower, the vertical channel. The closure phase is in light yellow, the re-opening phase in a darker tone. If you zoom in, you may notice a small gap between the two squares. The size of the gap is set by the 'delay' during calibration. As you can also see in the figure, it may be the case that events you would consider as a blink like the one shortly before x=8.2 were not detected as such. If you want eogui to also classify this as a blink, go to Detection→Parameters, and increase the 'threshold softening' factor for blinks or decrease the value for the minimum amplitude and re-run the analysis (Detection→Repeat Analysis).
Once you are satisfied with the detection performance, you can save your data.

2.2.4 Save results

Under File → Save as… you can choose various formats to save the events detected by eogui. The default format is an ASCII files which contains only numbers and has the same number of columns for blinks and saccades. This is achieved by splitting a blink in two lines – one for the closure and one for the reopening. The lines are:
Start Eventtype Duration Amplitude MaximumSpeed MomentOfMaxSpeedInPercentOfDuration ViewingDirection

For more information see section 3.2.1 (p.12)

2.3 Batch processing

So far, you called a Graphical User Interface (GUI) to analyze an EOG recording. In case of longer files (e.g. recordings >45 minutes) and known settings, this may not be necessary as all you want is the result the analysis in a file without browsing through the results. For that, there is `eogbatch.m`.
The basic call is:

```
eogbatch('D:\Recordings\EOG.mat',1,'C:\Results\EOG-events.mat',[1 2])

Syntax:
  eogbatch(inputfname,inputformat,outputfname,outputformat,(param.dat))

Inputs:
inputfname   - file name eog recording
inputformat  - input file format: 1=MAT, 2=VHDR, 3=ATISA, 4=Varioport VPD
5=Königstein IDAT
outputfname  - name of output file (without file extension)
outputformat - format of output file: 1=ASCII - numbers only, 2=ASCII, 3=MAT, 4=XML,
5=xyValues, 6=Köngistein
```

you can also enter several output formats as a vector e.g. `[1 2]`

[2] This functionality requires that you were not prevented from downloading and running the stayontop.dll originally included in the eogui.zip file. Otherwise an error message in the Matlab command window shows up, telling you that this function is not available.

Please note that there needs to be an existing `lastpara.config` before you can do batch processing, which means that you have to have done some kind of calibration with eogui before running eogbatch.

3 In- & Output formats

The previous chapter just mentioned the default settings for in- and output files. We tried to come up with formats that are as generic as possible, but probably these settings do not work for your data. Therefore we will list other formats eogui can read.

3.1 Input formats

Eogui accepts various input data formats. For each input format there is a 'reader', whose files you can see in the corresponding folder in the eogui directory, e.g. @matreader for matlab data files. If you like, you can program a reader for your data yourself, and include the option in the `eogui.m` file (see sections `% Get raw data file` and `% load file` there), but this requires some programming knowledge

3.1.1 Matlab data (*.mat) / ASCII files

The basic format is the already mentioned matlab matrix named 'data' with three columns: TIME (in milliseconds) XVALUE (i.e. horizontal channel) YVALUE (i.e. vertical channel). The included example EOG file (EOG.mat) is in that format. We would always recommend to use this format before wasting too much time trying to import other formats. Most recording software allows you to export your data to ASCII. ASCII data can then easily be imported by Matlab. If your recording software does not export a time stamp for each data point, you can simply create the respective column in Matlab as long as you know the sampling rate of your EOG recordings by:

```
samplingrate=256; %sampling rate in Hertz
mspersample=1000/samplingrate; %milliseconds per sample
mytime=[1:1:size(data,1)]'; %create column  vector of length same as your data
mytime=mytime*mspersample; %adjust time vector to sample rate
data= [mytime data]; %attach the time column to your x and y data already stored in 'data'
```

3.1.2 BrainVision data (*.vhdr)

Brainvision also offers the possibility to convert recordings to ASCII. When doing so, it creates a *.dat file that contain the actual data separated by commas and a *.vhdr file, that contains the file header with information about sampling rate etc. eogui can read these *.vhdr files[3].

3.1.3 Varioport data (*.vpd)

Eogui can also read certain types of data recorded with the Varioport device (Becker Meditec)[4]. Unfortunately it does not handle all of them. While data recorded with the Variograf software 4.67 (vario467.exe) were successfully imported, data from version 4.7 (vario470.exe) did not work. We do not know whether this is due to a change in the software or to some other reason. If you are lucky, eogui reads your Varioport recordings, otherwise this might be one of the cases where the ASCII export and Matlab import (see 3.1.1) might save you time in the end.

3.1.4 ARISTA

One format we needed for a set of experiments.

3.1.5 Königstein data (Idat.dat)

This format is not very common, but as many of our data sets were recorded in it, it is included in the software. The name refers to the programmer of the precursor of eogui. For more information see the history section 4.4 (p. 14).

3.2 Output formats

Eogui offers various output formats. Before describing them in detail, it might be helpful to point out that eogui treats blinks as 2 consecutive saccades in the vertical channel, one upward saccade (=lid closure) shortly followed by a downward saccade (=lid re-opening). This is reflected in the way the data are organized in some of the output formats. If you would like to add a new output format, check section 4.2 (p.13).
Unless explicitly stated otherwise, all x and y values are not AD values, but already converted to degrees. Speed values are in angular degrees per second.

[3] This functionality requires that you were not prevented from downloading and running the vhdrreader.dll originally included in the eogui.zip file. Otherwise an error message in the Matlab command window shows up, telling you that this function is not available.
[4] This functionality requires that you were not prevented from downloading and running the vpdrreader.dll originally included in the eogui.zip file. Otherwise an error message in the Matlab command window shows up, telling you that this function is not available.

3.2.1 ASCII format – numbers only (*.dat)

This is the simplest format and the one we use most frequently. It contains just numbers and has the same number of columns for both types of events, saccades or blinks. This is achieved by splitting a blink in two lines – one for the closure and one for the reopening.
The lines are:
Start Eventtype Duration Amplitude MaximumSpeed MomentOfMaxSpeedInPercentOfDuration ViewingDirection

Notes:
Start is in milliseconds
Eventtype: 0= Saccade, 1=lid closure, 2=lid re-opening
Duration is in milliseconds
Amplitude is in angular degree
Viewing Direction is in nautic degree ((0 for upwards gazes, 90 for gaze to the right, 180 for downwards gazes, 270 for gazes to the left; for blinks this value is meaningless and set to zero) and can only be taken as a coarse information

The delay between lid closure and re-opening can be calculated as:

```
delay = start_reopening – (start_closure + duration_closure);
```

The total duration of a blink is then:

```
Totduration=duration_closure + delay + duration_opening;
```

We usually compute these values after importing the ASCII files in the statistics software SPSS/PASW® and also have SPSS syntax files to convert the two data rows per blink into one row. This syntax is available on request (email to nielsgalley@t-online.de), but please be aware that the variable names and the comments are in German.

3.2.2 ASCII format (*.txt)

This creates 1 line of text for each event, blink or saccade. Saccades are listed first as:
S [(start end) (Xstartvalue Xendvalue Xamplitude) (Ystartvalue Yendvalue Yamplitude) (MomentOfMaximumSpeed MomentOfMaximumSpeedInPercentOfTotalDuration MaximumSpeed)

Blinks come next. The line is basically the concatenation of the lid closure (=an upward saccade) parameters followed by the lid re-opening (=an downward saccade). The parameters are the same as for a single saccade.

B [ParametersOfUpwardSaccade] [ParametersOfDownwardSaccade]

Notes:
The overall saccadic amplitude can be calculated as follows:

```
amp=sqrt(Xamplitude^2 + Yamplitude^2);
```

The delay between lid closure and re-opening can be calculated as:

```
Delay = startUpwardSaccade – endDownwardSaccade;
```

3.2.3 Matlab data structure (*.mat)

Your results will be saved in a *mat file which contains the following Matlab structures: `saccades`, `blinks`, `stimulus1`, `stimulus2` . The latter two are of no relevance, but `saccades` and `blinks` contain as many entries as there were saccades and blinks identified. Each element has more or less the same fields that were shown in the control window in the text box labelled 'Current event', the only difference is that 'end' is written as 'ende' in the structure (the German version of 'end') and '_proz' stands for 'percent'.
So, to get the beginning of the first saccade, you have to type

`saccades(1,1).start`, to get the end, `saccades(1,1).ende`

Use the Matlab array editor to learn about the other fields.

3.2.4 XML data structure (*.xml)

This format is quite similar to the Matlab data structure, except that all events are in one single file with blinks being listed first. Again, the two parts of a blink, the closure and reopening are described as two saccades.

3.2.5 XY Values (*.dat)

This format returns the start and end of each event as well as the maximum speed in terms of AD values, not in angular degrees. The first line contains the column names, the second line the sampling rate, and the next two lines the conversion factor AD2deg to translate horizontal and vertical signal changes AD values into angular degrees.

3.2.6 Königstein-Format

Again, a previous format that is of little use for others. For more information see the history section 4.4 (p. 14).

4 Background information

In this chapter we will describe some of the details you do not need to know to use eogui, but which may help you to understand how eogui works if you for example want to modify parts of its code.
As you might have noticed, eogui was written by Germans. Thus everything is commented neatly, but unfortunately in German. The same holds for many of the variables used. This is due to the fact that we originally did not consider publishing the software. Now, some years later, we are a little wary of changing all the comments & variables to English as we do not know whether it is worth the effort and might introduce some unforeseen errors. We might provide a list of variable descriptions, but this is future work. The same holds for this paragraph, which has to be considered work in progress. It might be extended based on the feedback we get and the time we have...

4.1 Algorithm

Eogui identifies saccades by a change in speed (=1st derivative), which is accompanied by an plateau-like signal shift. An event needs to satisfy both criteria to be labelled as a saccade, which are determined in two iterations (see example 2 p.15 for illustration):
In the first iteration, the raw signal is smoothed by a low-pass filter[5] and the 1st derivative of this filtered signal is used to identify potential signals[6]. In the second iteration, the smoothed signal is filtered again by a stronger low-pass filter[7], and this signal is used to determine the plateau-like signal shift. If both iterations identify an event as a potential saccade, additional plausibility checks are applied before it is finally listed as a valid saccade.
You can set the filters for both iterations, but we clearly discourage you from that unless you really know what you are doing .
Blinks are basically nothing else but a an upward saccade in the vertical signal followed by a downward saccade (see cover figure for an example). As the blink check is done before an event is finally labelled as a saccade, eogui can also identify horizontal saccades during a blink.

4.1.1 Plausibility checks

Eogui performs various plausibility checks before it eventually labels an event that was detect as a potential saccade or blink as such an event. The main checks done in `checkplausi.m` are:

- the saccade must have the same direction in the light low-pass filtered signal (1st iteration) as in the strong low-pass filtered signal
- a saccade has to last at least 10 ms (`plausi.min_dauer`)
- after a saccade there needs to be a plateau-like signal shift for at least 60 ms (`parameter.plausi.plateaudauer`)
- the saccadic amplitude must exceed half the noise level
- consecutive saccades of the same direction which are less than 24 ms apart are merged into 1 (`plausi.vereinigung`)

You can also check an set most parameters in `setupdefaultparameter.m`

4.2 Adding new output formats

If you know how to program in Matlab you can easily add new output formats. Here is some basic information that might help you:
The Save As – dialogue can be found in the function `mSave_Callback` in `eogui.m`. Here you can specifiy the file types offered. The index of the file option the user selected is then handed over to `saveresult.m` (in folder \EOGUI\private\), where a different function for each output file type exists. You can add new formats, but make sure that the index number of the new file format in `mSave_Callback` corresponds to the `switch…case` number in `saveresult.m` .
All parameters eogui determines are stored and passed on in `handles.blinks` and `handles.sakkaden` (please note the German spelling here!)

[5] blue line in output plot
[6] the 1st derivative and the speed thresholds are depicted as separate blue lines below the raw signal in the output plot
[7] red line in the output plot

4.3 Trigger channels

In Figure 3 on page 4 you can see a third plot, the trigger channel. We did not cover it in the quick guide, because this function may vary for different experiments. If you just recorded the EOG, this channel will automatically be set to zero. However, you can use up to 2 trigger channels to mark certain events during the recording session, i.e. the onset of a stimulus by letting the value of in that channel change, e.g. from 0 to 1. If this has been recorded along with your data, you can store it as a fourth and fifth column in the EOG data matrix.
To analyze, use the Parameter summary window (Menu: Detection → Parameter), mark the check box 'X Stimulus Analysis' or 'Y Stimulus Analysis' respectively and press the button 'Get'. You will see a dialogue window that is analogous to the ones used for marking the reference saccades, where you have to indicate what change in signal corresponds to a trigger event.
Once you save your data in the ASCII-numbers only format, the triggers will be saved in a separate file for each trigger channel with the extension *.trg. For more information on that topic please contact Niels Galley vie email (nielsgalley@t-online.de)

4.4 Program history

The main idea to idea described in the section *Algorithm* were first implemented by Achim Königstein in 1989 under the supervision of Prof. Niels Galley (Königstein, 1989) at the University of Cologne as a DOS program which could already deal with recordings of two hours length at a sampling rate of 1000 Hz and process the raw signal online. It was used for around 15 years, e.g. in (Galley, 1998). However, many aspects like the later plausibility checks were programmed as additional Visual Basic scripts that had to be applied subsequent to the initial analysis. In 2004, Maik Hoffmann implemented a revised Matlab-based version, eogui, as his diploma thesis under the supervision of Prof. Niels Galley and Prof. Martin Golz at the University of Applied Sciences Schmalkalden (Hofmann, 2005). Parallel to its development, eogui was used to re-analyze existing data sets recorded in the Königstein format (Schleicher et al., 2008) as well as new recordings (Schleicher, 2009), which helped to improve the first versions of the program and finally led to the current version. In 2011, we decided to make it available online, and translated all GUI texts into English, changed the interaction logic a bit and wrote this manual.

4.5 Related resources and literature

- Prof. Galley gives an annual workshop on eye movement recording techniques with an emphasis on EOG in Cologne. It is usually announced via the DGPs mailing list, the German psychologists' association. For more information contact nielsgalley@t-online.de
- The eye movement mailing list is also useful: https://www.jiscmail.ac.uk/cgi-bin/webadmin?A0=eye-movement
- The Society for Psychophysiological Research has a software repository where you can find various tools to analyze biosignals: http://www.sprweb.org/repository/index.cfm
  eogui might also be available there.

4.5.1 Literature

Recommendations:

(Galley, 2001) gives an extensive overview on physiological foundations of eye movements in German.
(Eggert, 2007) focuses on recording methods and discusses EOG comprehensively.
(Leigh & Kennard, 2004) deals only with saccades, but also clearly describes the limitations of EOG recordings.
(Becker, 1989) provides information on parameters used to describe saccadic events.

Becker, W. (1989). Metrics. In M. E. Goldberg (Ed.), *The neurobiology of saccadic eye movements* (pp. 13-67). Amsterdam: Elsevier.

Eggert, T. (2007). Eye Movement Recordings: Methods. In U. Büttner (Ed.), *Neuro-Ophthalmology* (Developments in Ophthalmology Vol. 40, pp. 15-34). Basel: Karger.

Galley, N. (1998). An enquiry into the relationship between activation and performance using saccadic eye movement parameters. *Ergonomics, 40*, 698-720.

Galley, N. (2001). Physiologische Grundlagen, Meßmethoden und Indikatorfunktion der okulomotorischen Aktivität. In F. Rösler (Ed.), *Enzyklopädie der Psychologie. Biologische Psychologie. Band 4, Grundlagen und Methoden der Psychophysiologie* (pp. 237-316). Göttingen: Hogrefe.

Hofmann, M. (2005). *eogui - ein Computerprogramm zur automatischen Lidschlag- und Sakkadenerkennung in Matlab.* (Diplomarbeit). Schmalkalden: FH Schmalkalden.

Königstein, A. (1989). *Die On-Line-Identifizierung sakkadischer Augenbewegungen aus dem Elektrookulogramm*. Bonn: Fachbereich Informatik der Universität zu Bonn.

Leigh, R. J., & Kennard, C. (2004). Using saccades as a research tool in the clinical neurosciences. *Brain, 127*(3), 460-477.

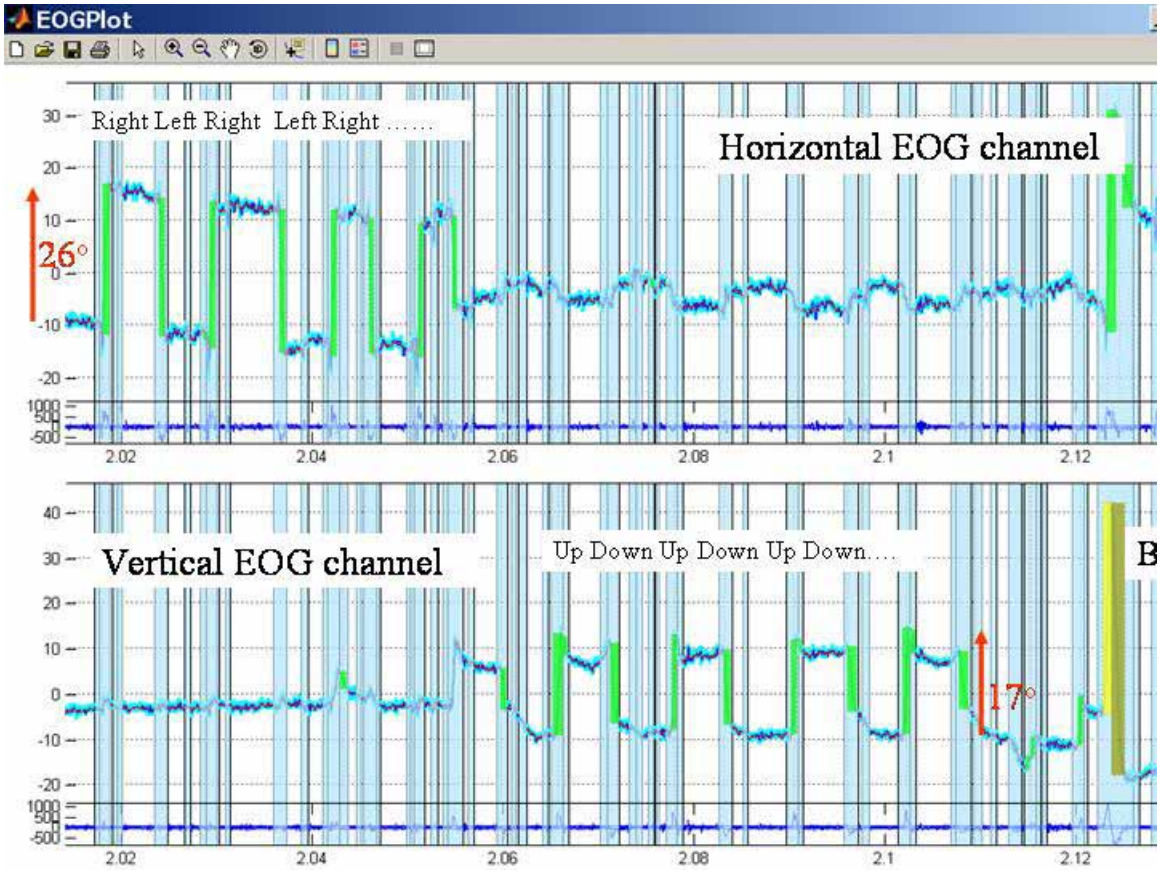
Schleicher, R. (2009). *Emotionen und Peripherphysiologie*. Lengerich: Pabst Science Publishers.

Schleicher, R., Galley, N., Briest, S., & Galley, L. (2008). Blinks and saccades as indicators of fatigue in sleepiness warnings: looking tired? *Ergonomics, 51*(7), 982-1010.

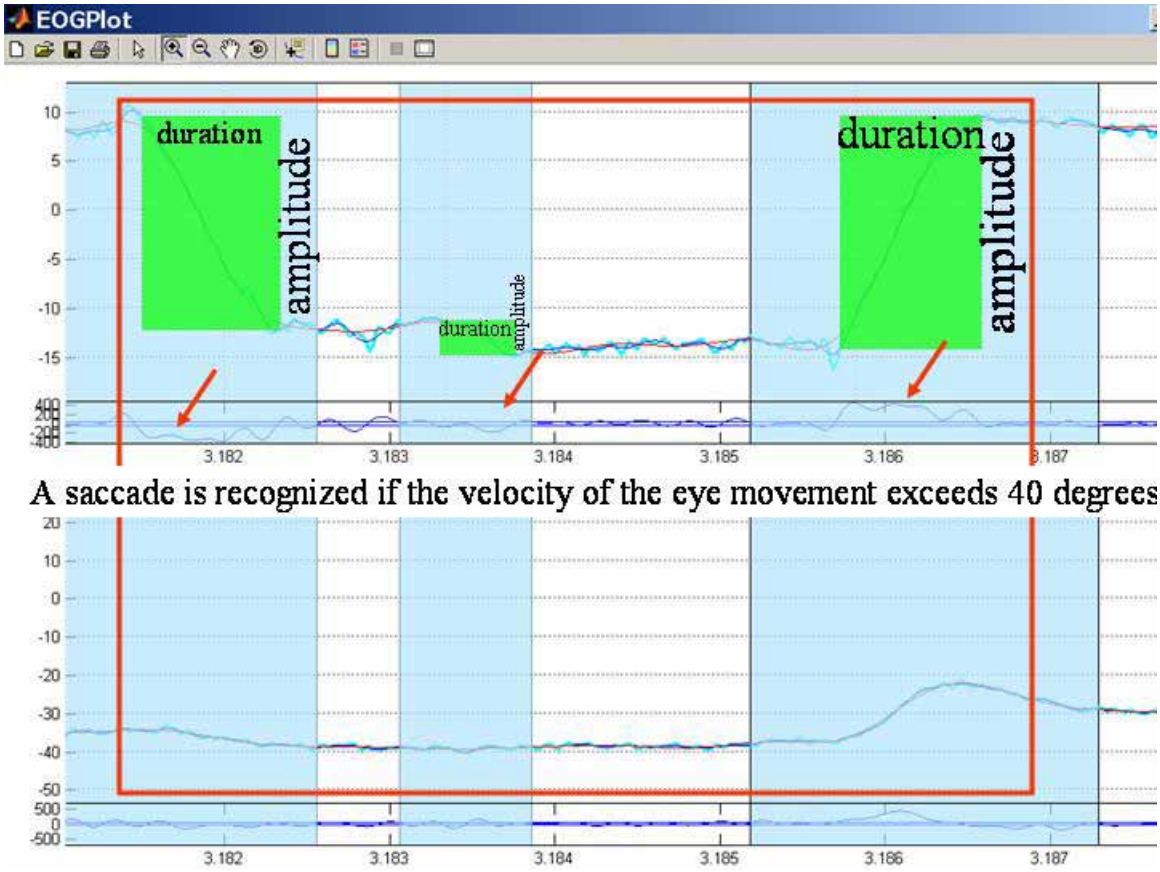
Examples

This appendix shows some screenshots and images to exemplify some of the statements from the manual

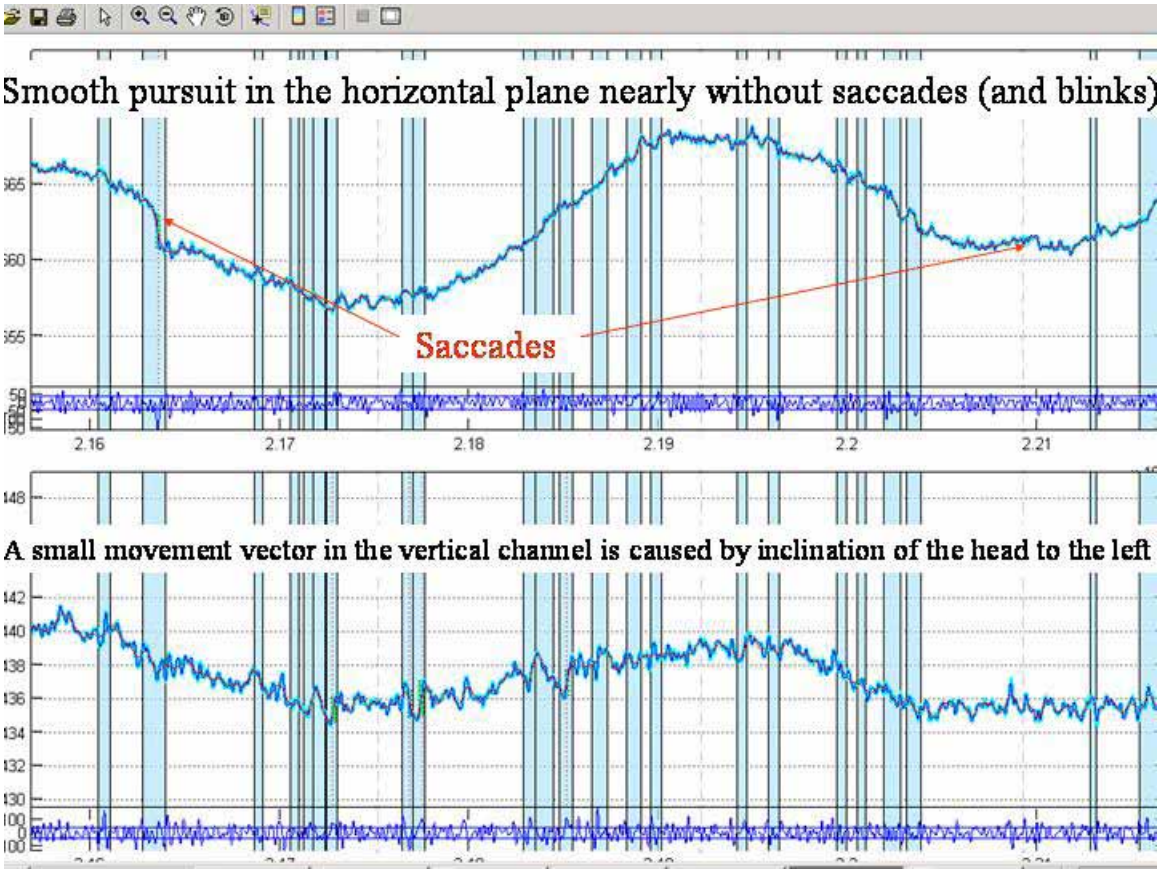
1 Calibration saccades in the EOG



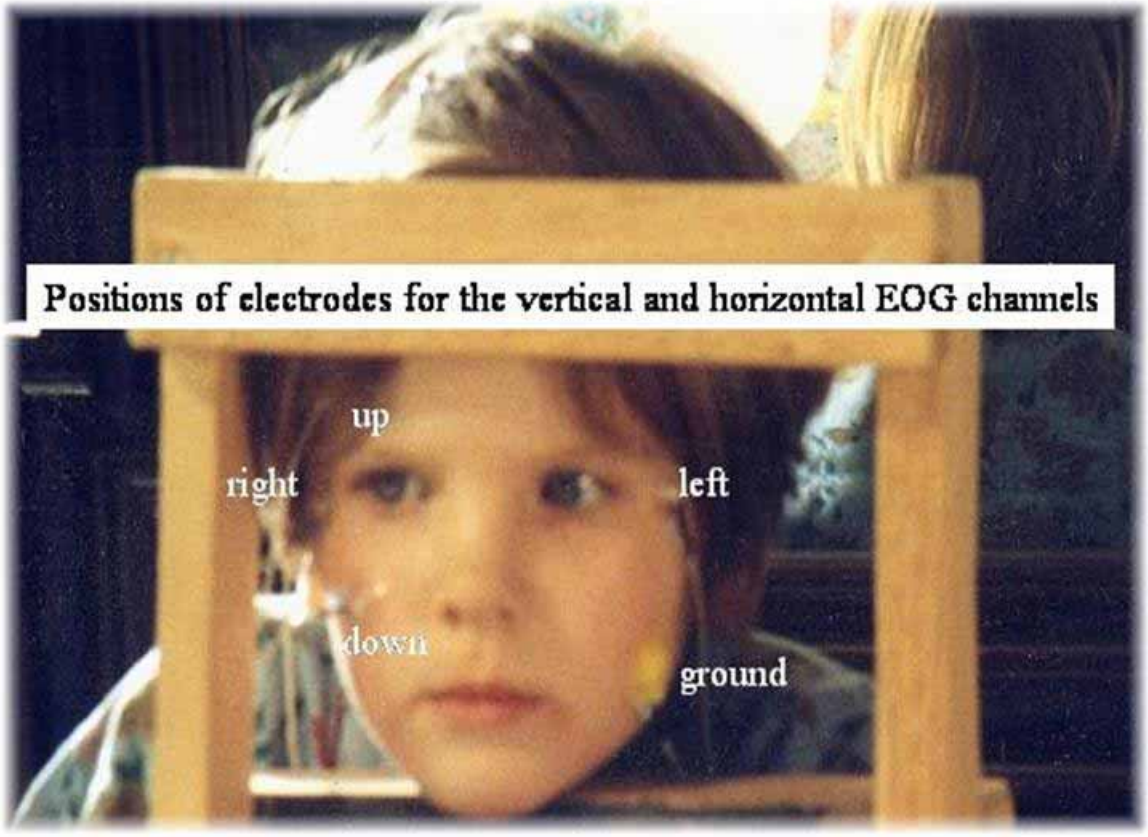
2 Depiction of saccadic amplitude & duration in EOGplot



3 Smooth pursuit eye movements and head movements in the EOG



4 Electrode positions for EOG recordings



5 Calibration procedure example

Panel for the calibration gazes for the vertical and horizontal EOG channels