Instructions for microDetect plug-in

Contents

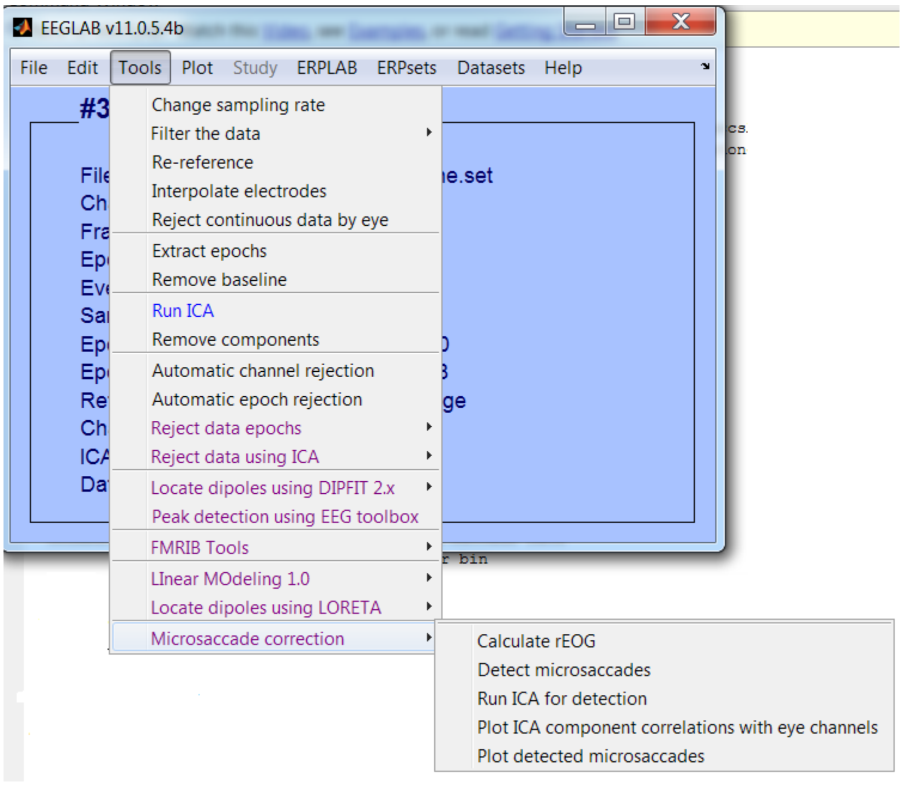
General information

Microsaccades pose significant problems for the analysis of gamma band activity with scalp-recorded EEG, generating an oculomuscular artifact with high power in the gamma band range that projects across the scalp (Yuval-Greenberg, Tomer, Keren, Nelken, & Deouell, 2008). They are too small and too frequent for conventional artifact detection procedures to be sufficient. While simultaneous eye-tracking is the gold-standard method of identifying the timing of microsaccades, appropriate equipment is not always available and may come with its own problems (e.g. power-line artefacts from head-mounted devices). Furthermore, it is rarely the case that simultaneous eye-tracking data is available for previously recorded datasets, such as those that constitute the extensive existing literature on gamma-band activity in EEG. Several methods of detection and correction of microsaccadic artefacts using ICA or linear regression have now been published (Hassler, Barreto, & Gruber, 2011; Keren, Yuval-Greenberg, & Deouell, 2010; Nottage, 2009), but are not commonly available for general use at present. We implement an ICA based approach primarily based on that described by Keren et al. (2010).

Installation instructions

The latest version of the plugin will always be available from here:
https://github.com/craddm/microDetect

Select the “master” branch for the most stable version, or select the “develop” branch for the most up-to-date version. In most cases, the “develop” branch will work at least as well as the “master” version. Download the zip archive using the “Download ZIP” icon on the right hand side of the screen. Unzip the archive to a folder called microDetect in your EEGLAB “plugins” directory. Start or re-start EEGLAB. You will find the new commands through the EEGLAB Tools dropdown menu, under the submenu “Microsaccade correction”.


EEGLAB v11.0.5.4b
File Edit Tools Plot Study ERPLAB ERPsets Datasets Help
Change sampling rate
Filter the data
Re-reference
Interpolate electrodes
Reject continuous data by eye
Extract epochs
Remove baseline
Run ICA
Remove components
Automatic channel rejection
Automatic epoch rejection
Reject data epochs
Reject data using ICA
Locate dipoles using DIPFIT 2.x
Peak detection using EEG toolbox
FMRIB Tools
LInear MOdeling 1.0
Locate dipoles using LORETA
Microsaccade correction
Calculate rEOG
Detect microsaccades
Run ICA for detection
Plot ICA component correlations with eye channels
Plot detected microsaccades

The commands are also accessible from the command line for scripting. After their use through the EEGLAB GUI, the syntax for the commands is accessible from the EEGLAB history function.

Consult the rest of this document for further advice on how to use the plugin, and the help text for the individual commands for an explanation of the options and syntax.

The plug-in has been tested on Matlab 2009a and 2012b, and with EEGLAB versions 11.x and 13.x, on Windows 7 Home Premium 64-bit and Linux (Ubuntu 12.04).

The Signal Processing Toolbox is also required for some elements. To use the matched filter from Keren et al. (2010), download it from http://hcnl.huji.ac.il/Leon/Lab/tools/filtSRP.m and add to a folder on your Matlab path (for example, in the microDetect plugin directory).

Please report bugs using the following website: https://github.com/craddm/microDetect/issues

Matt Craddock, 2013.

matt.craddock@uni-leipzig.de

Suggested workflow

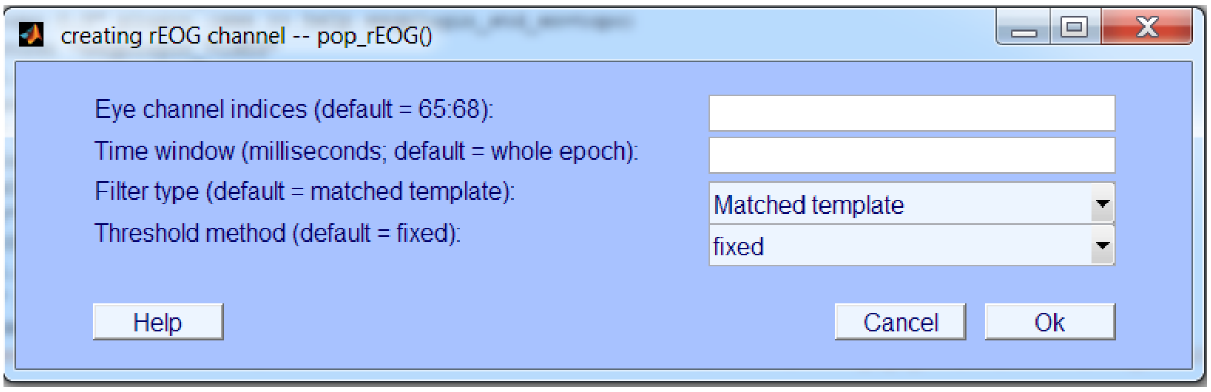
See below for detailed instructions on individual elements of this workflow.

1. Epoch your dataset and run some basic artifact correction. DO NOT BIPOLARIZE EYE CHANNELS. We also do not recommend interpolation of bad channels at this point, since this may interfere with the ICA algorithms. If any channels are interpolated, they should either be removed prior to running ICA or PCA should be used to reduce the rank of the dataset submitted to the ICA; then interpolate those channels afterwards.
2. All conditions of your experiment should be combined into a single epoched dataset. Calculate the radial electro-oculogram (pop_rEOG) on this dataset.
3. Detect microsaccades using pop_detect. If you wish to examine the rate across different conditions of your experiment, split your dataset into the relevant conditions and run the detection on each one individually.
4. Run the ICA on microsaccade centred epochs (pop_saccICA). If you split up your data into different conditions for detection, recombine them for this step.
5. Identify ICA components for removal from the dataset. We have provided a method of examining the correlation between the eye channels and individual ICA components to aid this process (pop_eyeCorrs).

Note that many commands add output to a field within the EEG structure, EEG.microS. You should save the EEG set regularly to maintain this information as current.

Calculating the rEOG and calculating thresholds (pop_rEOG)

The REOG is a combination of four eye channels (electrodes at the outer canthi of each eye, and above and below either the left or right eye) re-referenced to Pz (Keren et al., 2010). The REOG is filtered in order to accentuate the biphasic wave that typifies microsaccades, and to determine appropriate measures of the variation in the signal or a threshold for the detection of microsaccades. Note that it is best if your channel locations use names from the extended 10-20 system, since the algorithm will search for Pz automatically; if it does not find Pz, it will default to channel 31 (Pz on Biosemi 64 systems).


creating rEOG channel -- pop_rEOG()

Eye channel indices (default = 65:68):

Time window (milliseconds; default = whole epoch):

Filter type (default = matched template): Matched template

Threshold method (default = fixed): fixed

It is typically best to use a fairly focused time window, although this depends on your epoching. For example, we will typically use our baseline window and post-stimulus window (until at least 500ms after stimulus onset).

Three methods of filtering are available:

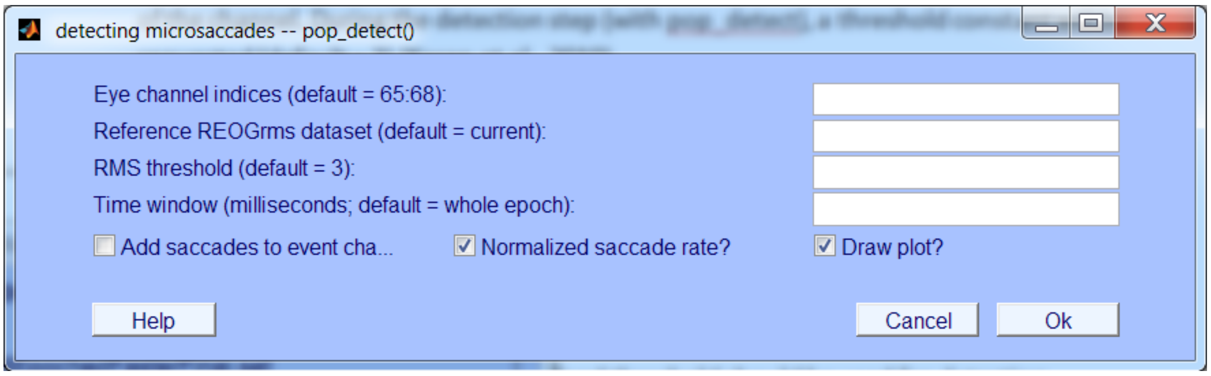
1. The matched template uses the saccade spike filter from (Keren et al., 2010), available from http://hcnl.huji.ac.il/Leon/Lab/tools/filtSRP.m. This convolves the REOG with the average normalized saccade –related potential from 5 subjects as recorded by Keren et al.
2. A 6th order Butterworth IIR filter, bandpassing the signal from 30 Hz to 100 Hz. This requires the Signal Processing Toolbox.
3. The first temporal difference (used by Keren et al., 2010, and Hassler et al., 2011).

Two methods of thresholding are available.

1. Fixed will calculate the root mean square (RMS) of the REOG as a measure of the variability of the channel. During the detection step (with pop_detect), a threshold constant will be requested (default = 3) (Keren et al., 2010).
2. Adaptive instead determines the median-based standard deviation of the REOG and also determines an appropriate thresholding constant. This constant is determined by calculating the amount of microsaccades which should be detected over the course of the whole experiment given an approximate rate of 3/second, iteratively adjusting the constant until the number of detected peaks approximates this amount (Hassler et al., 2011). **Note that this function is still under development.**

Detecting microsaccades (pop_detect)

After calculating the REOG and the various thresholds or measures required, you can proceed to detecting potential microsaccades in the data. The REOG of the dataset is calculated, and using the previously determined thresholds or RMS measures, peaks crossing the threshold are identified as potential microsaccades. Pop_detect will check what methods were used with pop_rEOG automatically, thus ensuring that the same filtering method is used to create the REOG throughout. The plug-in will add events at time points where potential microsaccades are detected, and count and plot the number of microsaccades in time bins of 20ms.


detecting microsaccades -- pop_detect()

Eye channel indices (default = 65:68):

Reference REOGrms dataset (default = current):

RMS threshold (default = 3):

Time window (milliseconds; default = whole epoch):

☐ Add saccades to event cha... ☑ Normalized saccade rate? ☑ Draw plot?

The reference dataset should be the one on which you calculated the REOG as above. This is useful when you have split the data up into individual conditions, since the thresholds should have been calculated on the complete dataset and thus be the same across all conditions.

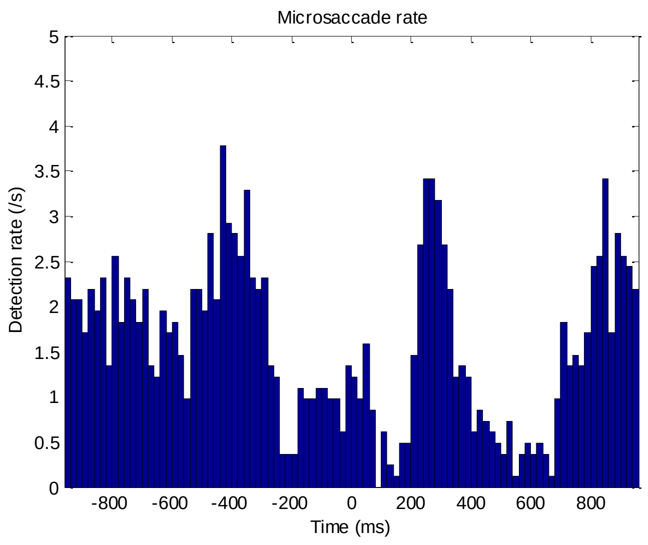
The RMS threshold is a multiplier used to set the threshold; anything more than X times the RMS is considered a suspect peak. **If you specified an adaptive threshold when calculating the REOG initially, this parameter is not necessary and can be ignored.**

The time window does not have to be the same as that used to calculate the REOG, but we would recommend including at least your baseline period and most of the post stimulus onset period.

Tick the **Add saccades to event channel** box to add an event ('sac') at the time each potential microsaccade is detected. This is required for subsequent ICA based correction.

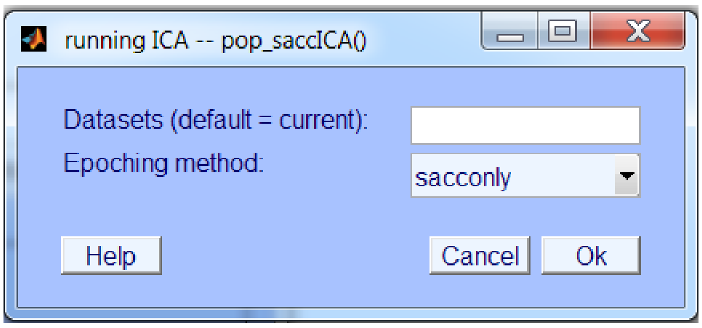
Tick the **Normalized saccade rate?** box to calculate the microsaccade rate per second for each time bin (20ms). Untick the box to simply count the number of microsaccades in each bin.

Tick the **Draw plot?** box to plot the detected microsaccades, either using the normalized rate or raw count as indicated by the other options.



Running ICA (pop_saccICA)

After detecting the microsaccades and adding microsaccade events to the events channel, the next step is to run ICA. The data is re-epoched into microsaccade-locked epochs from -100 ms before to 100 ms after each epoch. The ICA then runs on the microsaccade-locked epochs. The current plug-in runs the Infomax algorithm via pop_runica, attempting to first detect the rank of the data and reduce it using PCA where necessary (i.e. checking if the rank of the data has been reduced by averaging referencing or interpolation, for example). **Currently, only this method is implemented. To use other ICA algorithms or other epoching methods, please use EEGLAB commands.**


running ICA -- pop_saccICA()

Datasets (default = current):

Epoching method: sacconly

If you have several separate datasets in memory (e.g. you have split your dataset up into multiple conditions to run the detection step on each condition separately), enter them above in order to merge them into a single dataset before running ICA.

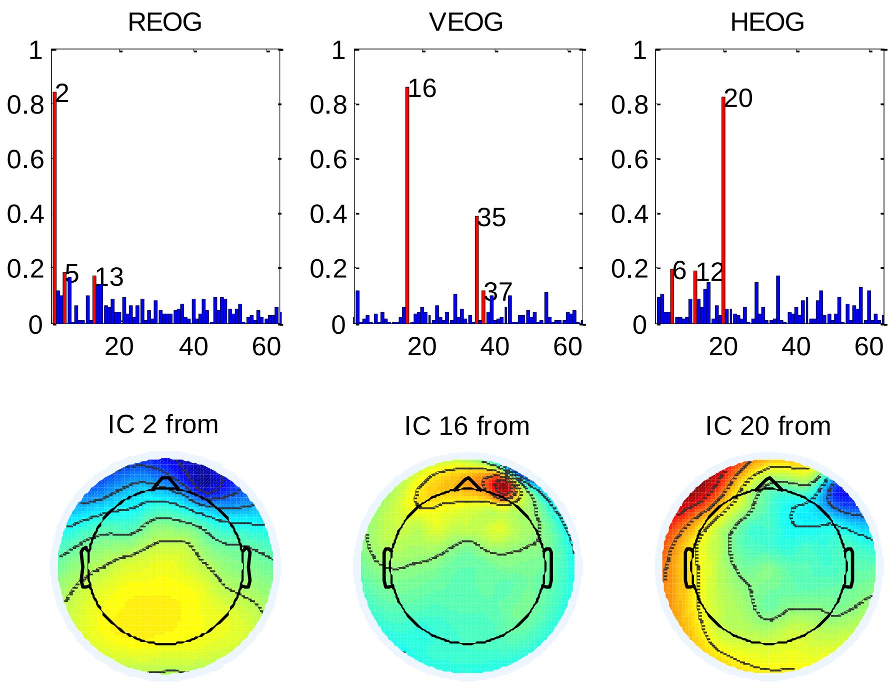
Two epoching methods will be available:

1. Sacconly re-epochs the data set to -100-100ms around each detected microsaccade. This only short, saccade locked epochs are submitted to ICA.
2. Addsacs will also create short epochs around each microsaccade but will then append these epochs to the end of the full dataset before submitting the data to ICA. **Not yet implemented**.

Note that in practice, we have found little difference between the artifactual components returned by either method; however, the latter method may in some cases provide better separation between of artifactual and brain-generated components.

Identifying artifactual components (pop_eyeCorrs)

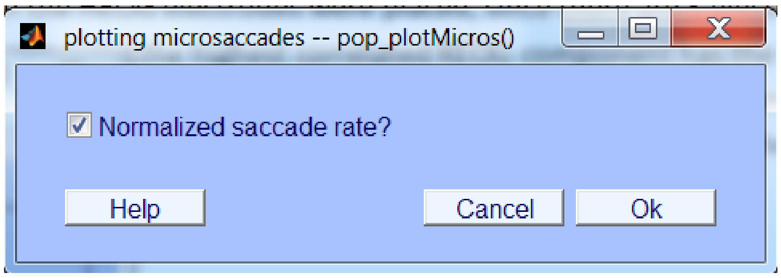
Identifying the relevant microsaccade related components can be difficult on occasion. It is best done through inspection of the topography and timecourse of the saccade related components. To help in this process, pop_eyeCorrs calculates the correlation of each ICA component's time course with the REOG channel and with bipolarized VEOG (vertical) and HEOG (horizontal) eye channels. It plots the correlation as a bar chart, highlighting the three highest correlated components for each of the REOG, VEOG, and HEOG in red and numbering them accordingly. The topography of the highest correlated component is also plotted.



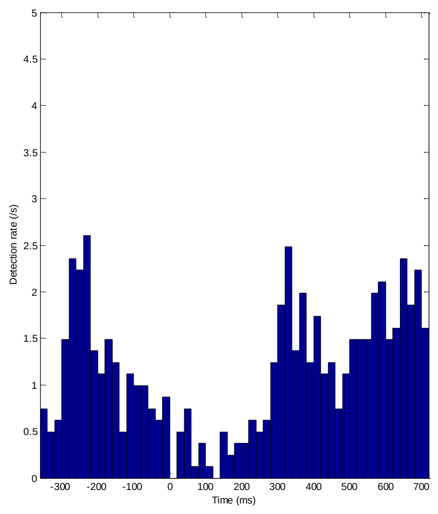
In the above example, IC 2 has the highest correlation with the REOG channel, and the topography exhibits a very typical pattern for a microsaccade component with high activity around Pz and parietal electrodes and around the frontal, eye electrodes. The strongest peak frontal peak is typically centred on the eye at which the HEOG electrodes were placed, since there are 3 electrodes around that eye. In our experience, removing the highest correlated REOG component has the largest impact on microsaccades. IC 16 correlates highest with the VEOG eye channel, and thus may capture blinks or vertical eye movements as well as microsaccades, while IC 20 correlates highest with HEOG, thus probably capture left-to-right or right-to-left eye movements.

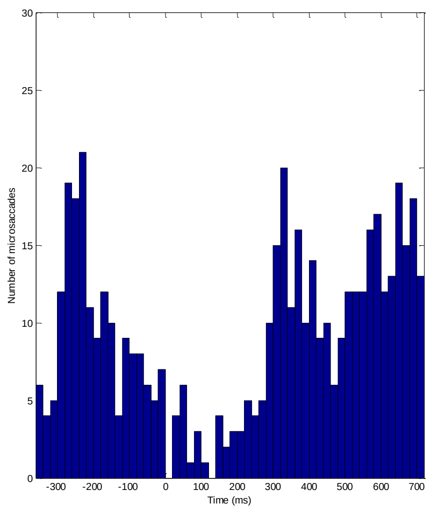
Plotting previously detected microsaccades (pop_plotMicros)

If you wish to plot previously detected microsaccades without having to re-run the detection step, use pop_plotMicros. This also allows the plotting of either microsaccade rates or microsaccade counts using the checkbox indicated in the figure.


plotting microsaccades -- pop_plotMicros()

[✓] Normalized saccade rate?





Command reference list

pop_rEOG

Creates the radial electrooculogram (rEOG) used to define the threshold for detection of miniature eye movements from EEG eye channels. Returns both the rEOG and the RMS of the rEOG, or the median based standard deviation and an adaptively determined thresholding constant.

```
 Usage:
 >>[EEGOUT] = pop_rEOG(EEG,'key1',value,'key2',value...)
```

Inputs:

EEG - EEGLAB data structure
eyechans - vector of eye channel indices (default = 65:68)
window - [mintime maxtime] (ms) e.g.[-500 700] default = whole epoch
filt - Filtering type to use:
 1 - matched filter. Requires filtSRP.m: (http://hcnl.huji.ac.il/Leon/Lab/tools/filtSRP.m)
 2 - Butterworth filter, 6th order, bandpass from 30-100 Hz (requires Signal Processing Toolbox)
 3 - First derivative.
method - Determine the RMS of rEOG for subsequent use with a fixed threshold, or the median-based standard deviation and adaptively determine a thresholding constant.
 1 - fixed (determine RMS only)
 2 - adaptive (threshold + median-based stdev)

Outputs:

EEGOUT - EEGLAB data structure with additional rEOG channel

pop_rEOG adds several fields to the EEG.microS structure:

EEG.microS.threshold = whether adaptive or fixed threshold should be used for detection
EEG.microS.REOGfilt = which filtering method was used (1st derive, butterworth, or matched)
EEG.microS.REOGrms = the rms of the channel if fixed threshold will be used
EEG.microS.REOGstdmed = standard deviation based on medians if adaptive threshold used
EEG.microS.adaptThresh = the adaptively determined constant used for subsequent detection.

pop_detect

pop_eyeCorrs

pop_saccICA

pop_plotMicros