Table of Contents

```
% Date: 29-11-2019             Programmed by: D. Bolger
% Script to detect the onset times of the feedbacks of interest and to
% determine the modality and type of each feedback.
% Script applied in the Brain-IHM project.
% ********************************************************
```

LOAD IN THE *.bdf FILE OF THE CURRENT SUBJECT

Loads in the *.bdf file, ensuring that 74 channels are included so that the ERGO1 and ERGO2 data are loaded.

```
curr_suj = 's01';
Dirbase =
 fullfile(filesep,'Users','bolger','Documents','work','Projects','Project-
PEPs','PEPs_raw_Subjects',curr_suj,filesep);
Savebase =
 fullfile(filesep,'Users','bolger','Documents','work','Projects','Project-
PEPs','PEPs_Preprocess_Subjects',curr_suj,filesep);
case_spec = 'S01';   % Subject 1 videos 1 and 2 are recorded on same
 *.bdf file.

allfiles= dir(Dirbase);
fileIndex = find(~[allfiles.isdir]);
filenum = dir(strcat(Dirbase,'*.bdf'));                            %find all
 the *.bdf files in the current folder
```

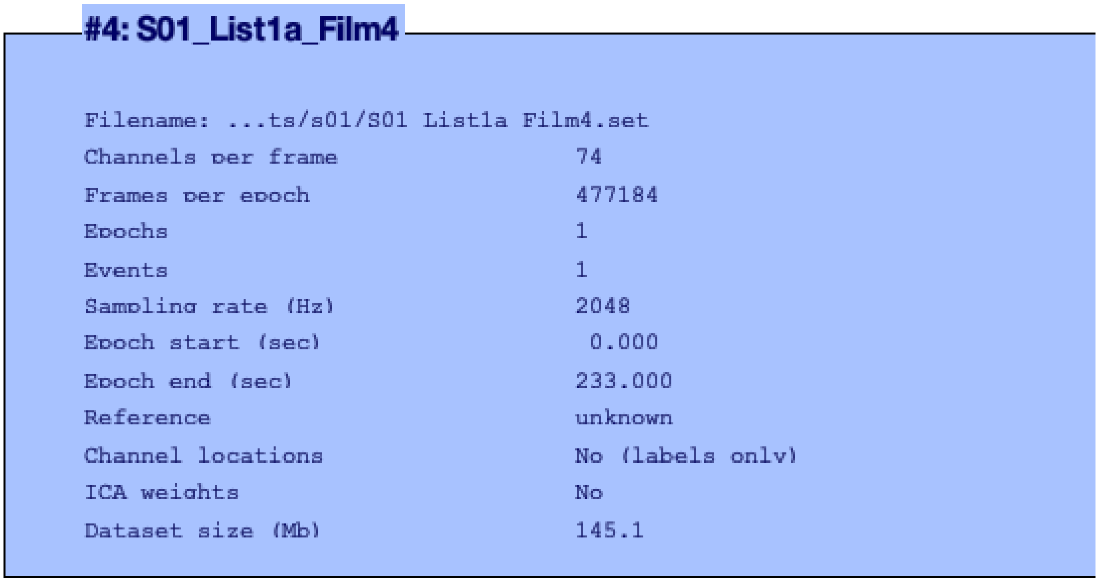
```
filenom = {filenum.name};
ergsig1 = cell(1,length(filenom));
ergsig2 = cell(1,length(filenom));

% OPEN EEGLAB SESSION
[ALLEEG, EEG, CURRENTSET, ALLCOM] = eeglab;
[ALLEEG, EEG] = eeg_store(ALLEEG, EEG, CURRENTSET);
filenoms01 = [1 1 1 1];

for scnt = 1:length(filenoms01)

    % The following three lines is added to resolve a bug occurring
 when
    % opening the *.bdf file.
    x = fileparts( which('sopen') );
    rmpath(x);
    addpath(x,'-begin');

    if strcmp(filenom{1,1}(1:3),case_spec)    % Needed to add this to
 deal with *bdf file of two videos.

        if scnt ==1
            fullDir = strcat(Dirbase,filenom{1,scnt});
            fnom = filenom{1,scnt}(1:end-4);
            EEG = pop_biosig(fullDir, 'channels',[1:76],'blockrange',
[0 275], 'ref', [] ,'refoptions',{'keepref' 'off'} );
            ergsig1{1,scnt} = EEG.data(75,:);
            ergsig2{1,scnt} = EEG.data(76,:);

            EEG = pop_select( EEG,'nochannel',{'Erg1' 'Erg2'});
            EEG = eeg_checkset( EEG );

            [ALLEEG, EEG, CURRENTSET] = pop_newset(ALLEEG, EEG,
 CURRENTSET,'setname',char(fnom),'gui','off'); % Create a new dataset
 for the current raw datafile
            [ALLEEG, EEG, CURRENTSET] = eeg_store(ALLEEG, EEG,
 CURRENTSET);
            EEG = eeg_checkset( EEG );
            EEG =
 pop_saveset( EEG, 'filename',char(fnom),'filepath',Savebase);
  % Saves a copy of the current resampled dataset to the current
 directory
            eeglab redraw


        elseif scnt ==2

            counttemp = 1;
            fullDir = strcat(Dirbase,filenom{1,1});
            fnom = 'S01_List1a_Film2';
            EEG = pop_biosig(fullDir, 'channels',[1:76],'blockrange',
[280 477], 'ref', [] ,'refoptions',{'keepref' 'off'} );
            ergsig1{1,scnt} = EEG.data(75,:);
            ergsig2{1,scnt} = EEG.data(76,:);
```

```
           EEG = pop_select( EEG,'nochannel',{'Erg1' 'Erg2'});
           EEG = eeg_checkset( EEG );

           [ALLEEG, EEG, CURRENTSET] = pop_newset(ALLEEG, EEG,
CURRENTSET,'setname',char(fnom),'gui','off'); % Create a new dataset
for the current raw datafile
           [ALLEEG, EEG, CURRENTSET] = eeg_store(ALLEEG, EEG,
CURRENTSET);
           EEG = eeg_checkset( EEG );
           EEG =
pop_saveset( EEG, 'filename',char(fnom),'filepath',Savebase);
 % Saves a copy of the current resampled dataset to the current
directory
           eeglab redraw

       elseif scnt>2
           counttemp = counttemp+1;
           fullDir = strcat(Dirbase,filenom{1,counttemp});
           fnom = filenom{1,counttemp}(1:end-4);
           EEG = pop_biosig(fullDir, 'channels',[1:76], 'ref',
[] ,'refoptions',{'keepref' 'off'} );


           ergsig1{1,scnt} = EEG.data(75,:);
           ergsig2{1,scnt} = EEG.data(76,:);

           EEG = pop_select( EEG,'nochannel',{'Erg1' 'Erg2'});
           EEG = eeg_checkset( EEG );

           [ALLEEG, EEG, CURRENTSET] = pop_newset(ALLEEG, EEG,
CURRENTSET,'setname',char(fnom),'gui','off'); % Create a new dataset
for the current raw datafile
           [ALLEEG, EEG, CURRENTSET] = eeg_store(ALLEEG, EEG,
CURRENTSET);
           EEG = eeg_checkset( EEG );
           EEG =
pop_saveset( EEG, 'filename',char(fnom),'filepath',Savebase);
 % Saves a copy of the current resampled dataset to the current
directory
           eeglab redraw
       end

   else   % For all other subjects who have 4 *.bdf files each.

       fullDir = strcat(Dirbase,filenom{1,scnt});
       fnom = filenom{1,scnt}(1:end-4);

       % Opening up *.bdf file and saving as a *.set file.
       EEG = pop_biosig(fullDir, 'channels',[1:76], 'ref',
[] ,'refoptions',{'keepref' 'off'} );


       ergsig1{1,scnt} = EEG.data(75,:);
```

```
        ergsig2{1,scnt} = EEG.data(76,:);

        EEG = pop_select( EEG,'nochannel',{'Erg1' 'Erg2'});    %
 The photodiode signals are on ERG1 and ERG2- need to extract these
 channels.
        EEG = eeg_checkset( EEG );

        [ALLEEG, EEG, CURRENTSET] = pop_newset(ALLEEG, EEG,
 CURRENTSET,'setname',char(fnom),'gui','off'); % Create a new dataset
 for the current raw datafile
        [ALLEEG, EEG, CURRENTSET] = eeg_store(ALLEEG, EEG,
 CURRENTSET);
        EEG = eeg_checkset( EEG );
        EEG =
 pop_saveset( EEG, 'filename',char(fnom),'filepath',Savebase);
  % Saves a copy of the current resampled dataset to the current
 directory
        eeglab redraw
    end % end of if


end
```

```
eeglab: options file is ~/eeg_options.m
EEGLAB: adding "Biosig" to the path; subfolders (if any) might be
 missing from the path
EEGLAB: adding "ERPWAVELABv" to the path; subfolders (if any) might be
 missing from the path
EEGLAB: adding "Fileio" to the path; subfolders (if any) might be
 missing from the path
EEGLAB: adding "dipfit" v2.3 (see >> help eegplugin_dipfit)
EEGLAB: adding "erplab" v7.0.0 (see >> help eegplugin_erplab)
EEGLAB: adding "firfilt" v1.6.1 (see >> help eegplugin_firfilt)
EEGLAB: adding "fullRankAveRef" v0.10 (see >> help
 eegplugin_fullRankAveRef)
EEGLAB: adding "limo_eeg-master" v2.0 (see >> help eegplugin_limo)
EEGLAB: adding "loreta" v1.1 (see >> help eegplugin_loreta)
EEGLAB: adding "rERP" v0.4 (see >> help eegplugin_rerp)
Creating a new ALLEEG dataset 1
Warning: line (423: 65152,"V/s",,"",,#obsolete#) not valid
Warning: line (428: 65312,"mHg s-1","","",#obsolete#) not valid
Warning: line (430: 65376,"r.p.m",,"rotations per minute",#obsolete#)
 not valid
Warning: line (431: 65408,"B", "Bel", "relative power
 decibel",#obsolete#) not valid
Warning: line (432: 65440,"dyne s m2 cm-5","Vascular Resistance
 Index","dyne seconds square meter per centimetre to the power of
 5",#obsolete#) not valid
Warning: line (433: 65440,"dyne*s*mÂ²/cm^5","Vascular Resistance
 Index","dyne seconds square meter per centimetre to the power of
 5",#obsolete#) not valid
Warning: line (436: 65504,"T",,"<magnitude> Tesla",#obsolete#) not
 valid
Reading data in BDF format...
```

```
eeg_checkset note: upper time limit (xmax) adjusted so (xmax-
xmin)*srate+1 = number of frames
Importing data events...
eeg_checkset note: creating the original event table (EEG.urevent)
eeg_checkset note: re-creating the original event table (EEG.urevent)
Removing 2 channel(s)...
Creating a new ALLEEG dataset 1
Saving dataset...
Reading data in BDF format...
eeg_checkset note: upper time limit (xmax) adjusted so (xmax-
xmin)*srate+1 = number of frames
Importing data events...
Removing 2 channel(s)...
Creating a new ALLEEG dataset 2
Saving dataset...
Reading data in BDF format...
eeg_checkset note: upper time limit (xmax) adjusted so (xmax-
xmin)*srate+1 = number of frames
Importing data events...
eeg_checkset note: creating the original event table (EEG.urevent)
eeg_checkset note: re-creating the original event table (EEG.urevent)
Removing 2 channel(s)...
Creating a new ALLEEG dataset 3
Saving dataset...
Reading data in BDF format...
eeg_checkset note: upper time limit (xmax) adjusted so (xmax-
xmin)*srate+1 = number of frames
Importing data events...
eeg_checkset note: creating the original event table (EEG.urevent)
eeg_checkset note: re-creating the original event table (EEG.urevent)
Removing 2 channel(s)...
Creating a new ALLEEG dataset 4
Saving dataset...
```



#4: S01_List1a_Film4

```
Filename: ...ts/s01/S01 List1a Film4.set
Channels per frame                 74
Frames per epoch                   477184
Epochs                             1
Events                             1
Sampling rate (Hz)                 2048
Epoch start (sec)                   0.000
Epoch end (sec)                    233.000
Reference                          unknown
Channel locations                  No (labels only)
ICA weights                        No
Dataset size (Mb)                  145.1
```

************************EXTRACT PHOTODIODE ONSETS***************************

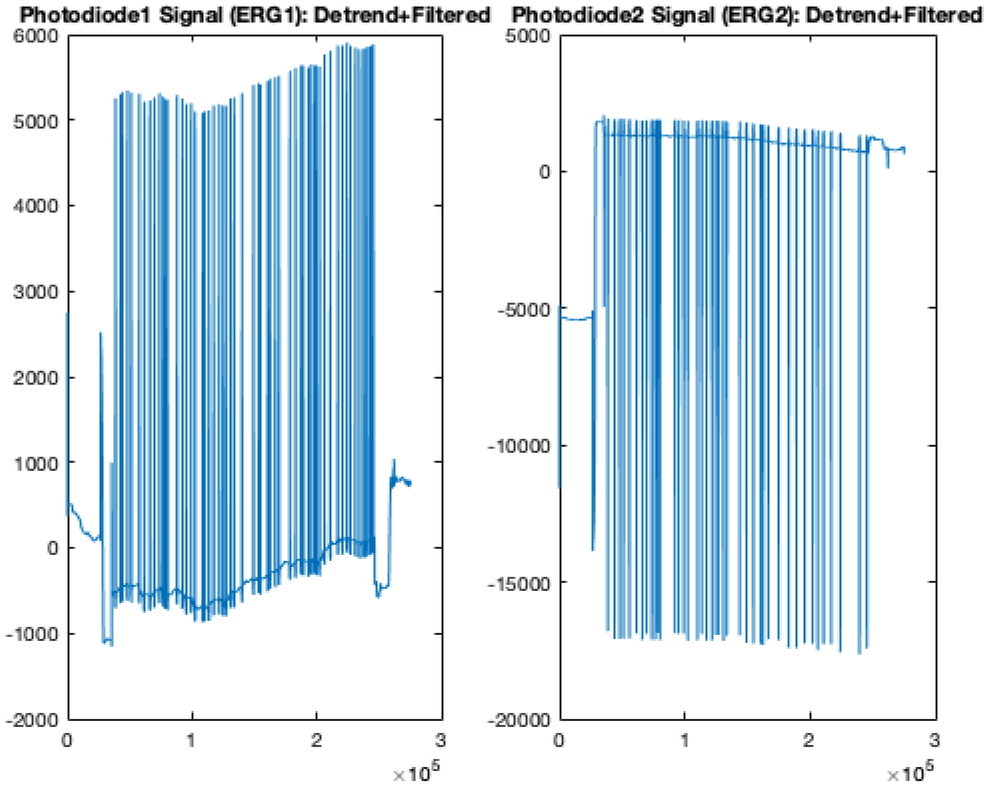
Need to carry out some pre-processing on the photodiode signals to facilitate the detection of their onsets. The idea is to convert the photodiode signals to step functions.

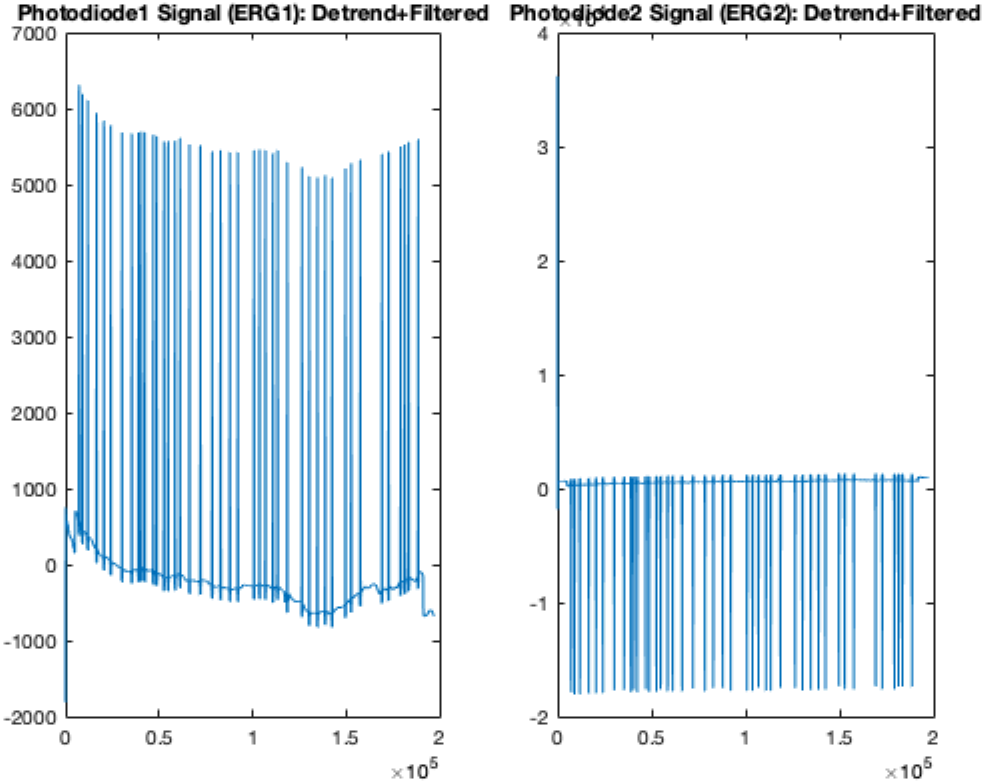
```
allfiles2 = dir(Savebase);                % Finding the number of
 *.set files in the current pre-processed data folder.
filenum2 = dir(strcat(Savebase,'*.set')); % Find all the *.set files
 in the current folder
filenom2 = {filenum2.name};
fb_sessions = filenom2;                   % Contains the name of each
 video type presented.

% Set parameters for photodiode signal pre-processing.
thresh_val = 0;
fs = EEG.srate;                 % Sampling frequency.
[b,a] = butter(2,6./(fs/2));    % Low pass filter with 6Hz cutoff to
 deal with photodiode flickering- get coefficients.

% Initialise variables.
onoffsets_ergo1 = cell(1,size(ergsig1,2));
onoffsets_ergo2 = cell(1,size(ergsig1,2));
onsets_ergo1 = cell(1,size(ergsig1,2));
onsets_ergo2 = cell(1,size(ergsig1,2));
offsets_ergo1 = cell(1,size(ergsig1,2));
offsets_ergo2 = cell(1,size(ergsig1,2));
Times_bloc = cell(length(fb_sessions),1);   % Time vector for each
 video.

for ergcnt = 1:size(ergsig1,2)

    Times_bloc{ergcnt,1} = ALLEEG(ergcnt).times;
    time = Times_bloc{ergcnt,1};
```

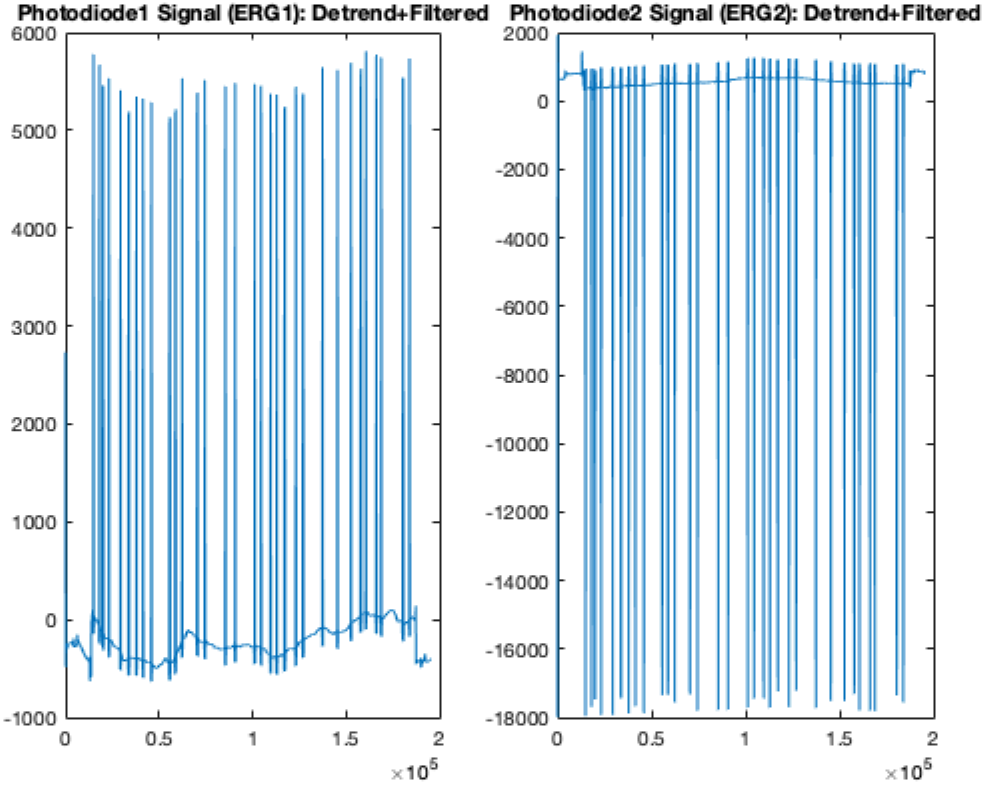
FILTER PHOTODIODE SIGNALS TO REMOVE HIGHER FREQUENCY OSCILLATORY ACTIVITY.

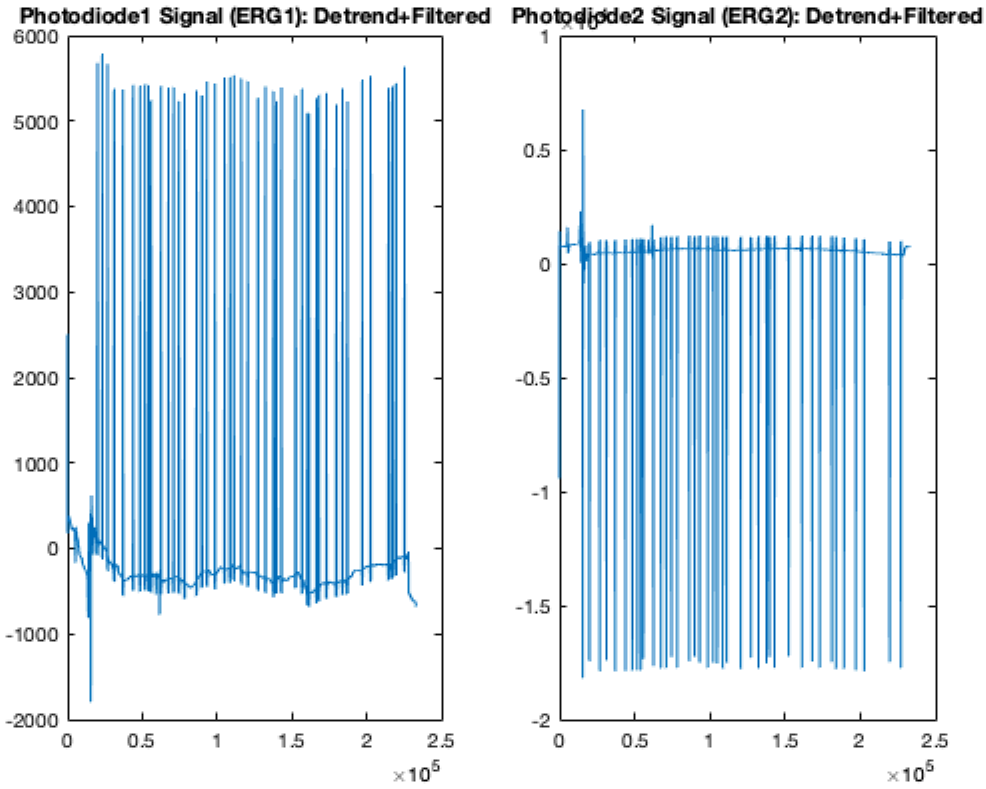
```
    ergsig1_curr = ergsig1{1,ergcnt};
    ergsig2_curr = ergsig2{1,ergcnt};
    ergsig1D = detrend(ergsig1_curr,0);    % Detrend the photodiode1
    ergsig2D = detrend(ergsig2_curr,0);    % Detrend the photodiode2

    ergsig1Dfilt = filtfilt(b,a,double(ergsig1D));
    ergsig2Dfilt = filtfilt(b,a,double(ergsig2D));

    % Plot the detrended and filtered signals
    figure('Name',[fb_sessions{1,ergcnt},': detrended +
 filtered'],'NumberTitle','off');
```

```
   subplot(1,2,1)
   plot(time, ergsig1Dfilt); title('Photodiode1 Signal (ERG1):
Detrend+Filtered');
   subplot(1,2,2)
   plot(time, ergsig2Dfilt); title('Photodiode2 Signal (ERG2):
Detrend+Filtered');
```



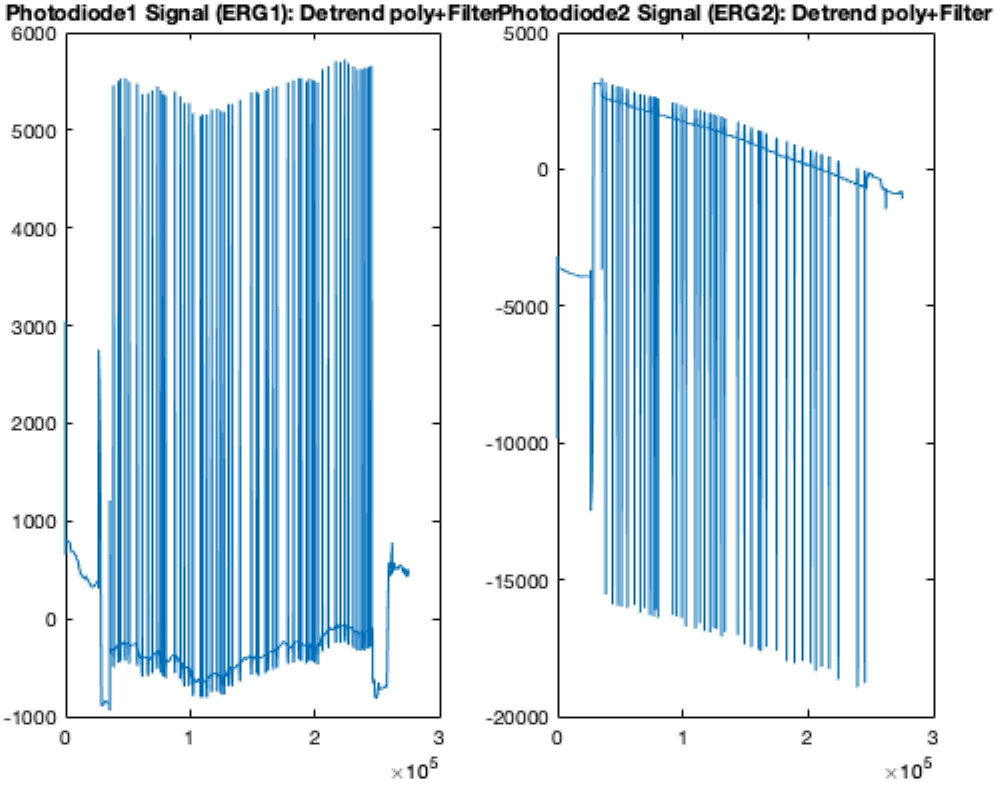

Photodiode1 Signal (ERG1): Detrend+Filtered
7000
6000
5000
4000
3000
2000
1000
0
-1000
-2000
0
0.5
1
1.5
2
×10^5
Photodiode2 Signal (ERG2): Detrend+Filtered
×10^4
4
3
2
1
0
-1
-2
0
0.5
1
1.5
2
×10^5

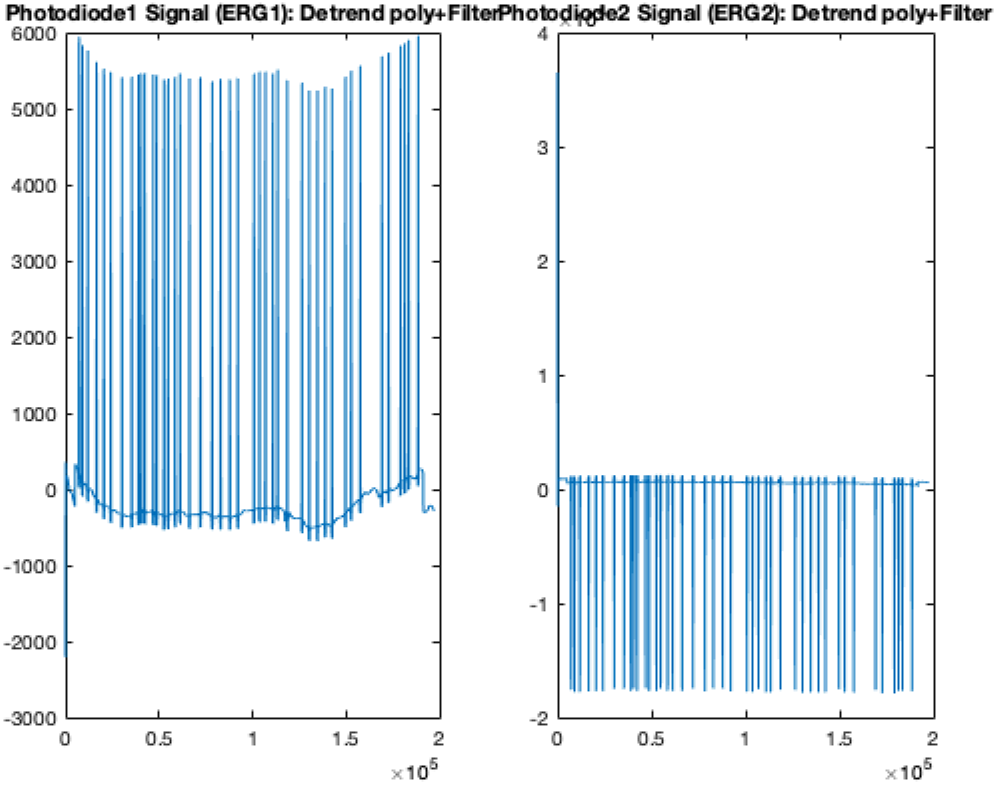

Photodiode1 Signal (ERG1): Detrend+Filtered
6000
5000
4000
3000
2000
1000
0
-1000
0
0.5
1
1.5
2
×10^5
Photodiode2 Signal (ERG2): Detrend+Filtered
2000
0
-2000
-4000
-6000
-8000
-10000
-12000
-14000
-16000
-18000
0
0.5
1
1.5
2
×10^5

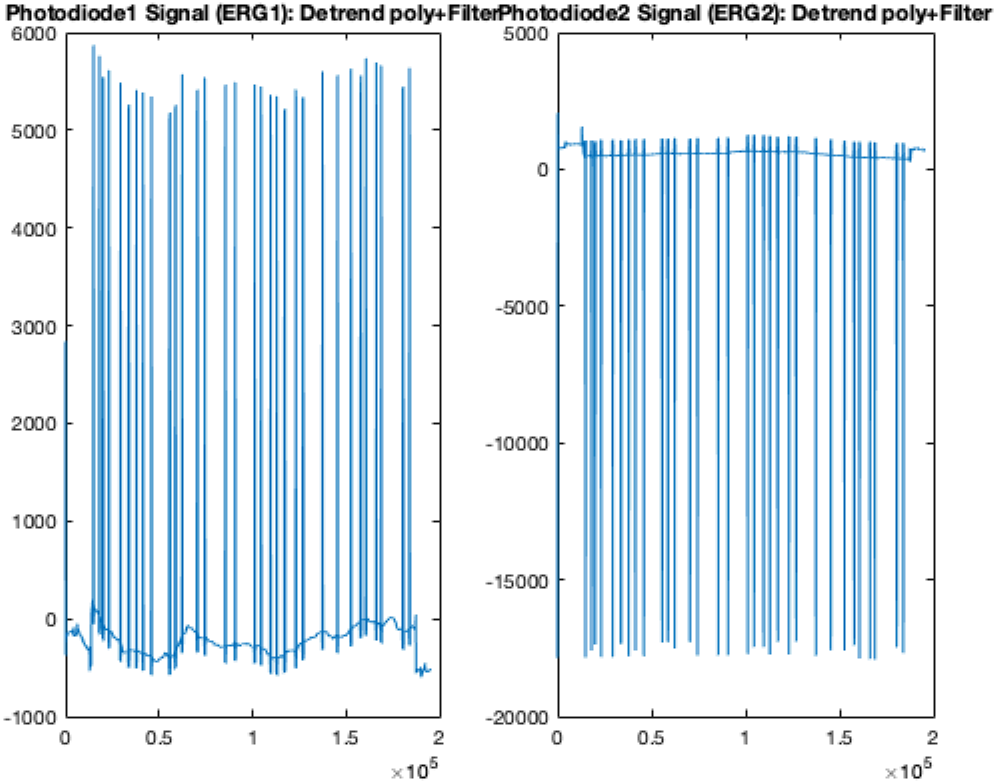


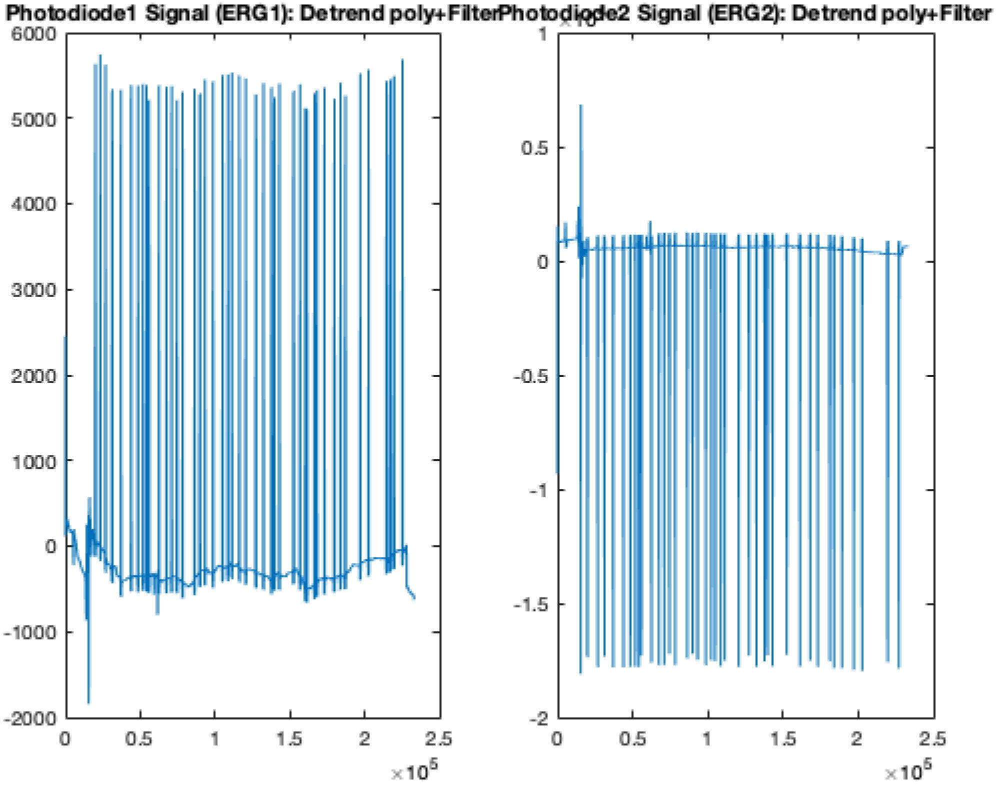
DETREND BY FITTING POLYNOMIAL CURVE.

```
   p1 = polyfit(1:length(ergsig1Dfilt),ergsig1Dfilt,1);  %
Photodiode1
   f1 = polyval(p1,1:length(ergsig1Dfilt));
   ergsig1Dfilt_corr = ergsig1Dfilt - f1;

   p2 = polyfit(1:length(ergsig2Dfilt),ergsig2Dfilt,1);   %
Photodiode2
   f2 = polyval(p2,1:length(ergsig2Dfilt));
   ergsig2Dfilt_corr = ergsig2Dfilt - f2;

   % Plot the detrended using polynomial fitting and filtered signals
   figure('Name',[fb_sessions{1,ergcnt},': detrended poly
+filtered'],'NumberTitle','off');
   subplot(1,2,1)
   plot(time, ergsig1Dfilt_corr); title('Photodiode1 Signal (ERG1):
Detrend poly+Filter');
   subplot(1,2,2)
   plot(time, ergsig2Dfilt_corr); title('Photodiode2 Signal (ERG2):
Detrend poly+Filter');
```


Photodiode1 Signal (ERG1): Detrend poly+Filter
6000
5000
4000
3000
2000
1000
0
-1000
0
1
2
3
×10^5
Photodiode2 Signal (ERG2): Detrend poly+Filter
5000
0
-5000
-10000
-15000
-20000
0
1
2
3
×10^5

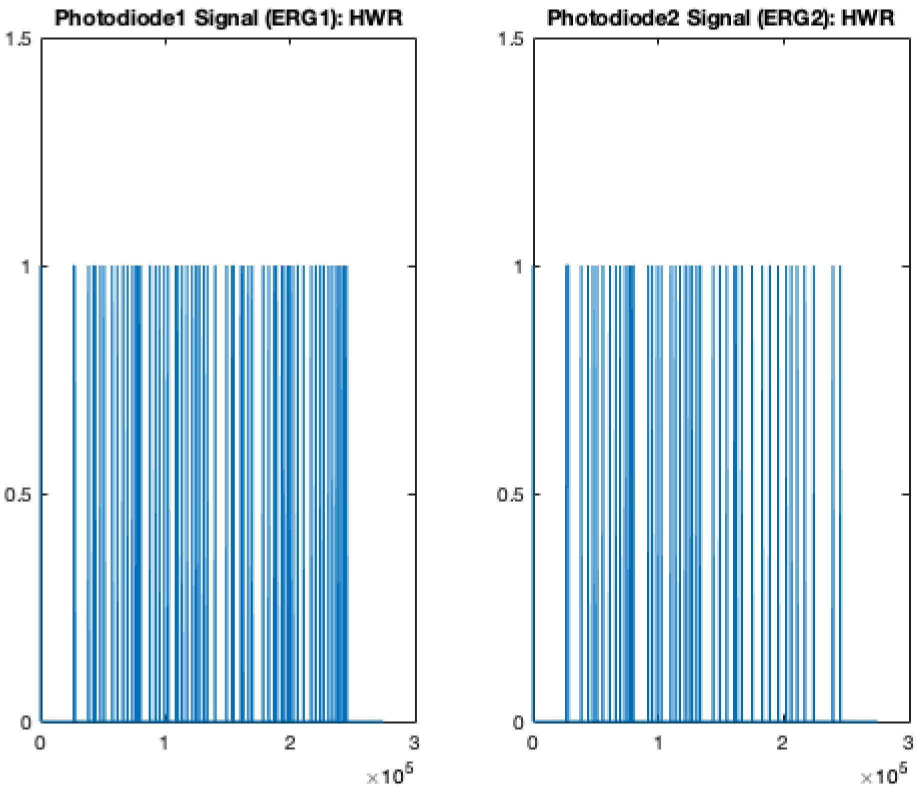

Photodiode1 Signal (ERG1): Detrend poly+Filter
6000
5000
4000
3000
2000
1000
0
-1000
-2000
-3000
0
0.5
1
1.5
2
×10^5
Photodiode2 Signal (ERG2): Detrend poly+Filter
×10^4
4
3
2
1
0
-1
-2
0
0.5
1
1.5
2
×10^5

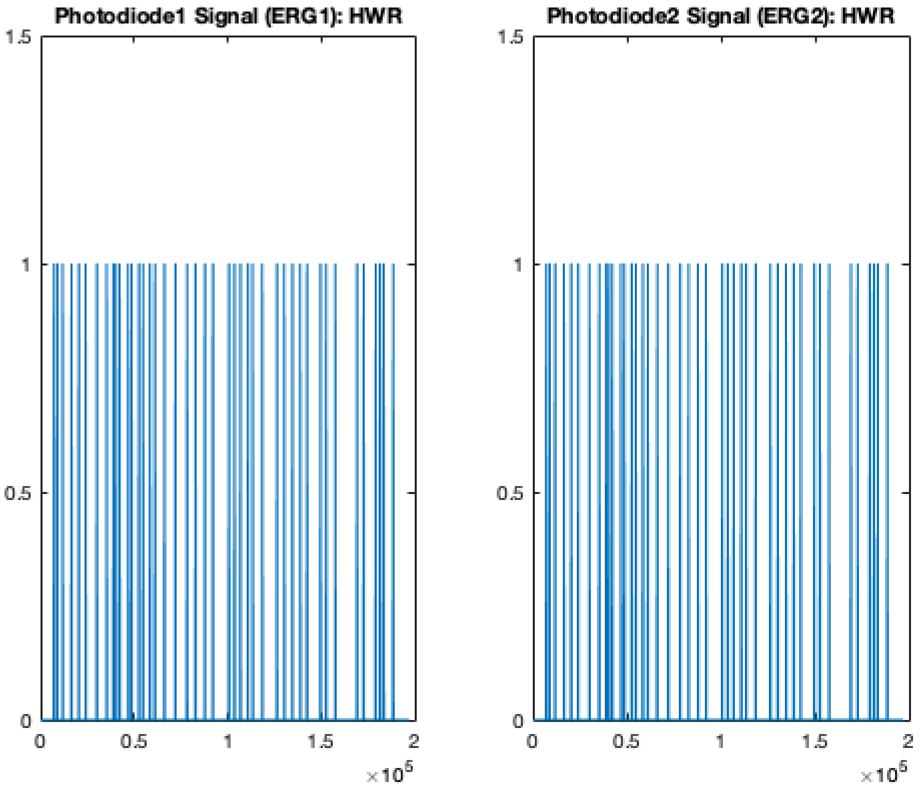

Photodiode1 Signal (ERG1): Detrend poly+Filter
6000
5000
4000
3000
2000
1000
0
-1000
0
0.5
1
1.5
2
$\times 10^5$
Photodiode2 Signal (ERG2): Detrend poly+Filter
5000
0
-5000
-10000
-15000
-20000
0
0.5
1
1.5
2
$\times 10^5$

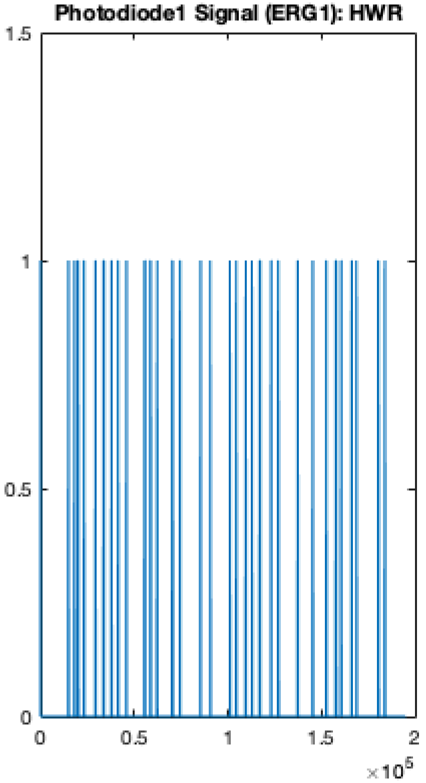

Photodiode1 Signal (ERG1): Detrend poly+Filter
6000
5000
4000
3000
2000
1000
0
-1000
-2000
0
0.5
1
1.5
2
2.5
$\times 10^5$
Photodiode2 Signal (ERG2): Detrend poly+Filter
$\times 10^4$
1
0.5
0
-0.5
-1
-1.5
-2
0
0.5
1
1.5
2
2.5
$\times 10^5$

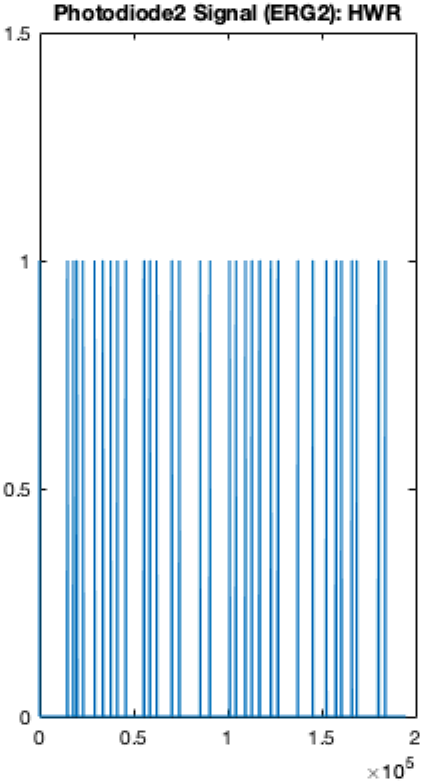
HALF WAVE RECTIFY (HWR) THE FILTERED SIGNAL AND INVERT.

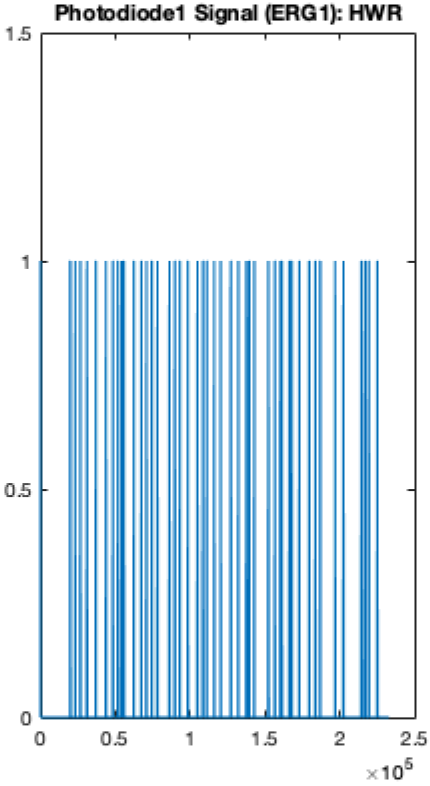
```
   ergsig1_hwr = zeros(size(ergsig1Dfilt_corr));
   ipos = find(ergsig1Dfilt_corr>0);
   ergsig1_hwr(ipos)= ergsig1Dfilt_corr(ipos).*1;

   ergsig2_hwr = zeros(size(ergsig2Dfilt_corr));
   ipos1 = find(ergsig2Dfilt_corr<0);
   ergsig2_hwr(ipos1) = ergsig2Dfilt_corr(ipos1).*-1;


   % Set all activity <= 1/3 of max to 0.
   ergsig1_hwr(ergsig1_hwr<=max(ergsig1_hwr)/3) = 0; ergsig1_hwrz =
ergsig1_hwr;
   ergsig2_hwr(ergsig2_hwr<=max(ergsig2_hwr)/3) = 0; ergsig2_hwrz =
ergsig2_hwr;


   % Turn all trigger signals into step functions.
   ergsig1_hwrst = ergsig1_hwrz;
   ergsig2_hwrst = ergsig2_hwrz;
   ergsig1_hwrst( ergsig1_hwrz>0) = 1;
   ergsig2_hwrst( ergsig2_hwrz>0) = 1;

   % Plot the detrended using polynomial fitting, filtered, hwr and
step-function convertedsignals
   figure('Name',[fb_sessions{1,ergcnt},': detrended, filtered, hwr,
step'],'NumberTitle','off');
   subplot(1,2,1)
   plot(time, ergsig1_hwrst); title('Photodiode1 Signal (ERG1):
HWR'); set(gca,'YLim',[0 1.5]);
   subplot(1,2,2)
   plot(time, ergsig2_hwrst); title('Photodiode2 Signal (ERG2):
HWR');
   set(gca,'YLim',[0 1.5])
```

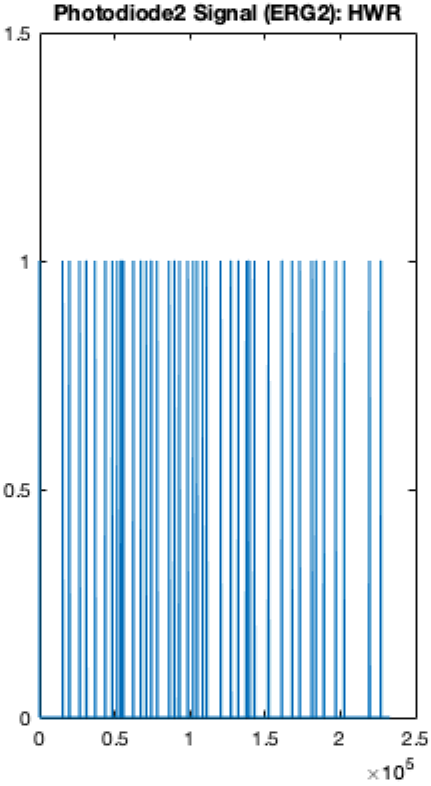

Photodiode1 Signal (ERG1): HWR
1.5
1
0.5
0
0
1
2
3
×10^5
Photodiode2 Signal (ERG2): HWR
1.5
1
0.5
0
0
1
2
3
×10^5


Photodiode1 Signal (ERG1): HWR
1.5
1
0.5
0
0
0.5
1
1.5
2
×10^5
Photodiode2 Signal (ERG2): HWR
1.5
1
0.5
0
0
0.5
1
1.5
2
×10^5


Photodiode1 Signal (ERG1): HWR
1.5
1
0.5
0
0
0.5
1
1.5
2
$\times 10^5$


Photodiode2 Signal (ERG2): HWR
1.5
1
0.5
0
0
0.5
1
1.5
2
$\times 10^5$


Photodiode1 Signal (ERG1): HWR
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
$\times 10^5$


Photodiode2 Signal (ERG2): HWR
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
$\times 10^5$

FIND ONSETS (DIFF(d_hwr==1) AND OFFSETS (diff(D_hwr==-1).

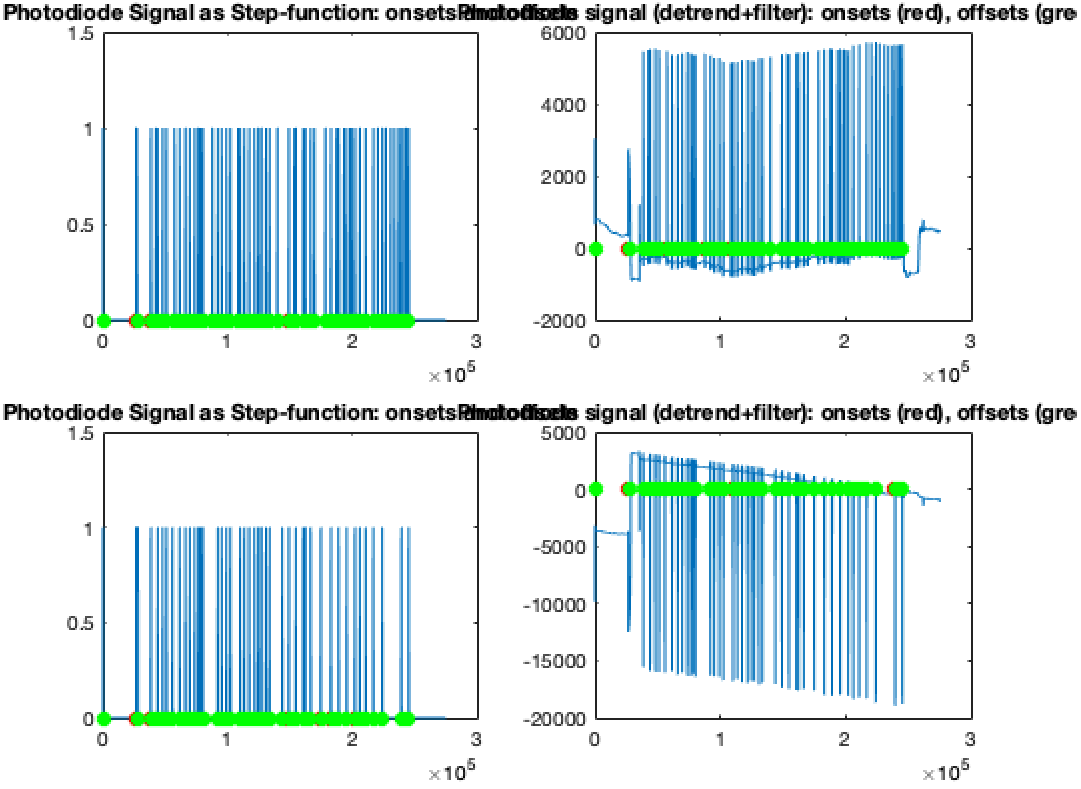
```
   erg1_diff = diff(ergsig1_hwrst);  % 1OD of D_hwr
   erg1_diff = cat(2,erg1_diff,0);
   erg2_diff = diff(ergsig2_hwrst);
   erg2_diff = cat(2,erg2_diff,0);

   [pks_sig1,minima_sig1,locs_pks_sig1,locs_min_sig1]=
CREx_peakfinder(erg1_diff);  %Call of function to locate peaks.
   [pks_sig2,minima_sig2,locs_pks_sig2,locs_min_sig2]=
CREx_peakfinder(erg2_diff);


   if locs_min_sig1(1)<locs_pks_sig1(1)  % If offset precedes an
onset in time for photodiode 1
       onoffsets_sig1 =
[time(locs_pks_sig1);time(locs_min_sig1(2:end))]';
       durs_sig1 = onoffsets_sig1(:,2) - onoffsets_sig1(:,1);       %
Calculate the photodiode signal duration.
       onoffsets_ergo1{1,ergcnt} = cat(2,onoffsets_sig1,durs_sig1); %
Concatenate onset/offset and signal duration information.
       onsets_all1 = nan(size(time));
       offsets_all1 = nan(size(time));
       onsets_all1(erg1_diff == pks_sig1(1)) = 0;
       offsets_all1(erg1_diff== minima_sig1(1)) = 0;
       onsets_ergo1{1,ergcnt} = onsets_all1;
       offsets_ergo1{1,ergcnt} = offsets_all1;


   else
       onoffsets_sig1 = [time(locs_pks_sig1);time(locs_min_sig1)]';
       durs_sig1 = onoffsets_sig1(:,2) - onoffsets_sig1(:,1);
       onoffsets_ergo1{1,ergcnt} = cat(2,onoffsets_sig1,durs_sig1);
       onsets_all1 = nan(size(time));
       offsets_all1 = nan(size(time));
       onsets_all1(erg1_diff == pks_sig1(1)) = 0;
       offsets_all1(erg1_diff== minima_sig1(1)) = 0;
       onsets_ergo1{1,ergcnt} = onsets_all1;
       offsets_ergo1{1,ergcnt} = offsets_all1;
   end


   if locs_min_sig2(1)<locs_pks_sig2(1)  % If offset precedes an
onset in time for photodiode 2

       onoffsets_sig2 =
[time(locs_pks_sig2);time(locs_min_sig2(2:end))]';
       durs_sig2 = onoffsets_sig2(:,2) - onoffsets_sig2(:,1);    %
Calculate the photodiode signal duration.
       onoffsets_ergo2{1,ergcnt} = cat(2,onoffsets_sig2,durs_sig2);
       onsets_all2 = nan(size(time));
```

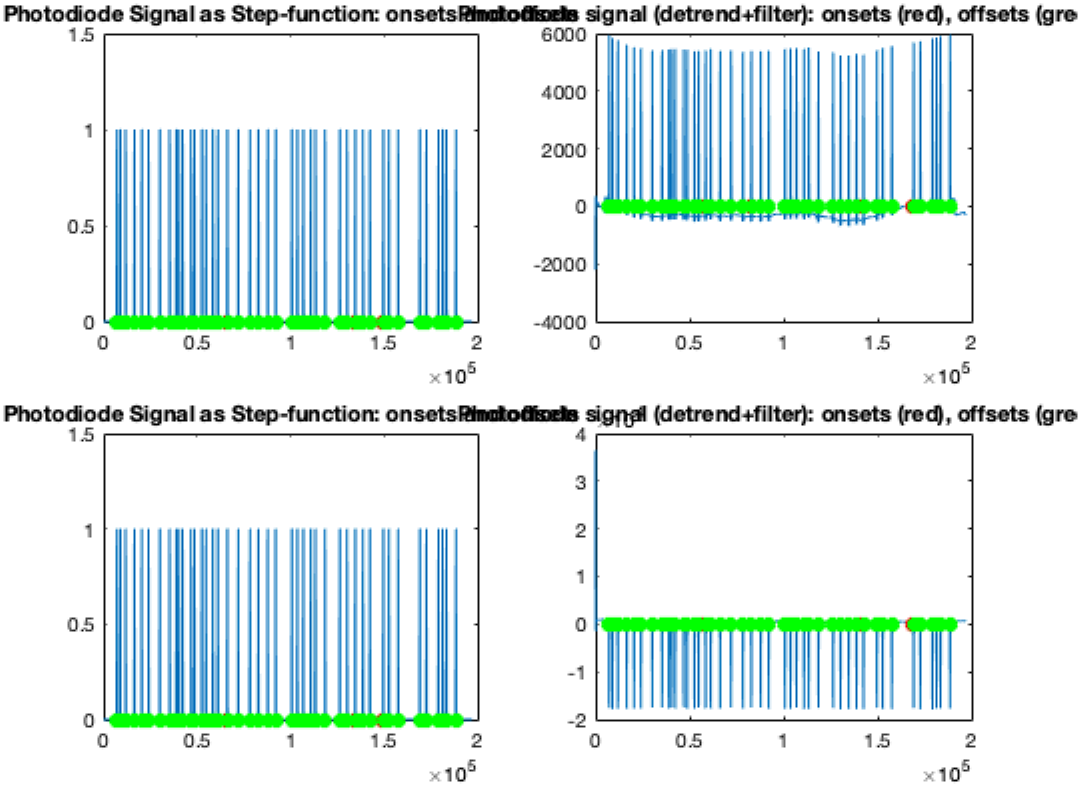
```
        offsets_all2 = nan(size(time));
        onsets_all2(erg2_diff == pks_sig2(1)) = 0;
        offsets_all2(erg2_diff== minima_sig2(1)) = 0;
        onsets_ergo2{1,ergcnt} = onsets_all2;
        offsets_ergo2{1,ergcnt} = offsets_all2;
    else

        onoffsets_sig2 = [time(locs_pks_sig2);time(locs_min_sig2)]';
        durs_sig2 = onoffsets_sig2(:,2) - onoffsets_sig2(:,1);
        onoffsets_ergo2{1,ergcnt} = cat(2,onoffsets_sig2,durs_sig2);
        onsets_all2 = nan(size(time));
        offsets_all2 = nan(size(time));
        onsets_all2(erg2_diff == pks_sig2(1)) = 0;
        offsets_all2(erg2_diff== minima_sig2(1)) = 0;
        onsets_ergo2{1,ergcnt} = onsets_all2;
        offsets_ergo2{1,ergcnt} = offsets_all2;
    end
```

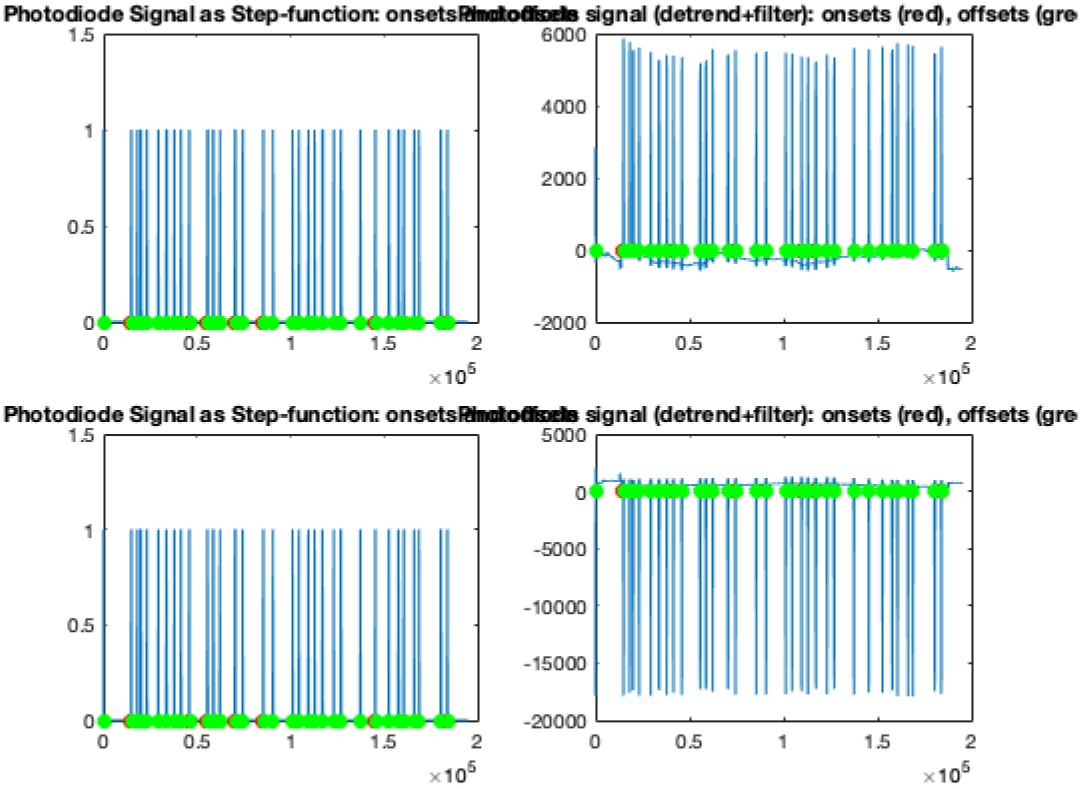
PLOT THE DETRENDED AND FILTERED PHOTODIODE SIGNAL (RIGHT) AND SIGNAL CONVERTED TO STEP FUNCTION (LEFT).

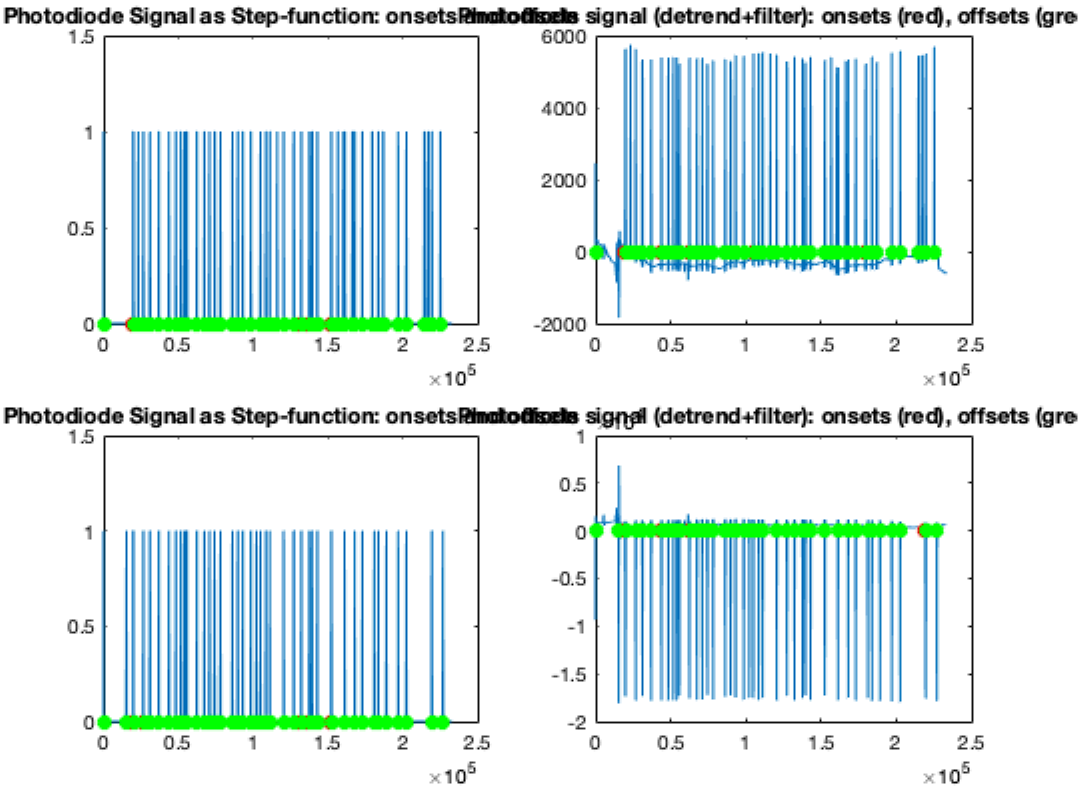
```
   figure('Name',fb_sessions{1,ergcnt},'NumberTitle','off');
   subplot(2,2,1)
   plot(time,ergsig1_hwrst);
   hold on
   plot(time,onsets_ergo1{1,ergcnt},'or','MarkerFaceColor','r')
   hold on
   plot(time,offsets_ergo1{1,ergcnt},'og','MarkerFaceColor','g');
   set(gca,'YLim',[0 1.5])
   title('Photodiode Signal as Step-function: onsets and offsets');
   subplot(2,2,2)
   plot(time,ergsig1Dfilt_corr)
   hold on
   plot(time,onsets_ergo1{1,ergcnt},'or','MarkerFaceColor','r');
   hold on
   plot(time,offsets_ergo1{1,ergcnt},'og','MarkerFaceColor','g');
   title('Photodiode signal (detrend+filter): onsets (red), offsets
(green)');

   subplot(2,2,3)
   plot(time,ergsig2_hwrst);
   hold on
   plot(time,onsets_ergo2{1,ergcnt},'or','MarkerFaceColor','r')
   hold on
   plot(time,offsets_ergo2{1,ergcnt},'og','MarkerFaceColor','g');
   set(gca,'YLim',[0 1.5])
   title('Photodiode Signal as Step-function: onsets and offsets');
   subplot(2,2,4)
   plot(time,ergsig2Dfilt_corr)
   hold on
```

```
    plot(time,onsets_ergo2{1,ergcnt},'or','MarkerFaceColor','r');
    hold on
    plot(time,offsets_ergo2{1,ergcnt},'og','MarkerFaceColor','g');
    title('Photodiode signal (detrend+filter): onsets (red), offsets
(green)');
```




Photodiode Signal as Step-function: onsets and offsets
Photodiode signal (detrend+filter): onsets (red), offsets (gre
1.5
1
0.5
0
0
0.5
1
1.5
2
×10^5
6000
4000
2000
0
-2000
-4000
Photodiode Signal as Step-function: onsets and offsets
Photodiode signal (detrend+filter): onsets (red), offsets (gre
×10^4
4
3
2
1
0
-1
-2

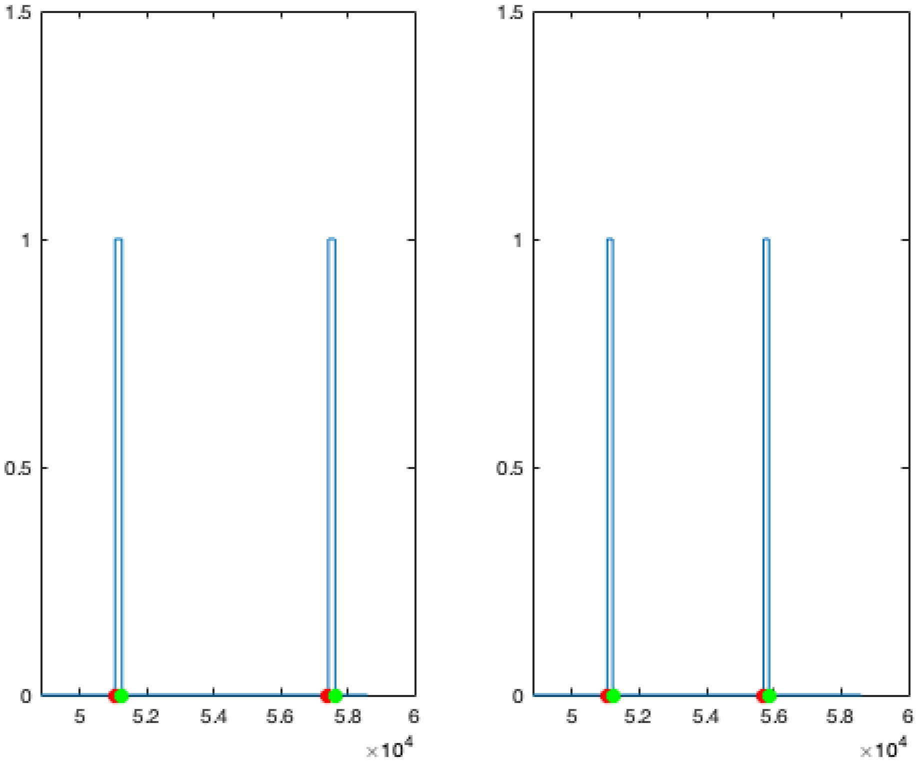

Photodiode Signal as Step-function: onsets and offsets
Photodiode signal (detrend+filter): onsets (red), offsets (gre
1.5
1
0.5
0
0
0.5
1
1.5
2
×10^5
6000
4000
2000
0
-2000
Photodiode Signal as Step-function: onsets and offsets
Photodiode signal (detrend+filter): onsets (red), offsets (gre
5000
0
-5000
-10000
-15000
-20000

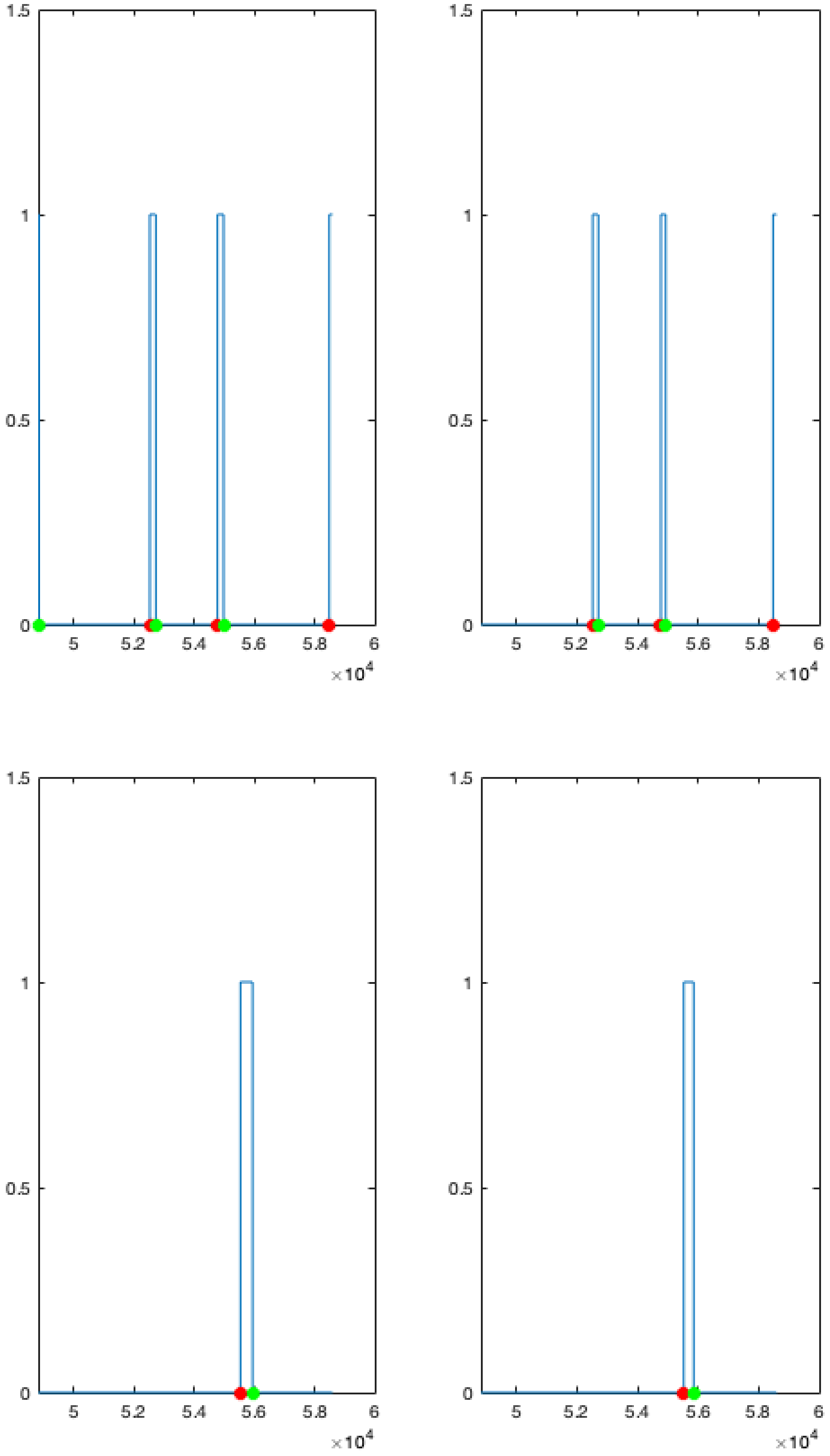


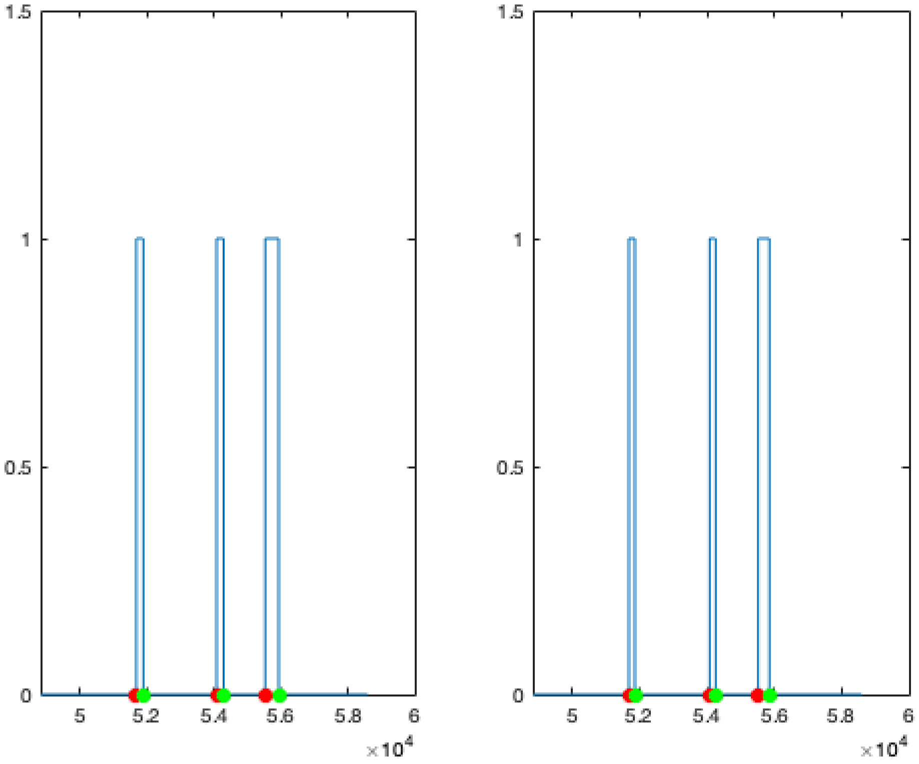
PLOT THE DETAIL OF THE PHOTODIODE SIGNALS TO SHOW DETECTED ONSETS AND OFFSETS.

```
     indxs = 100000:120000;  % Indices to be included in plot.

     figure('Name',fb_sessions{1,ergcnt},'NumberTitle','off');
     subplot(1,2,1)
     plot(time(indxs),ergsig1_hwrst(indxs));
     hold on
     plot(time(indxs),onsets_ergo1{1,ergcnt}
(indxs),'or','MarkerFaceColor','r')
     hold on
     plot(time(indxs),offsets_ergo1{1,ergcnt}
(indxs),'og','MarkerFaceColor','g')
     set(gca,'YLim',[0 1.5])

     subplot(1,2,2)
     plot(time(indxs),ergsig2_hwrst(indxs));
     hold on
     plot(time(indxs),onsets_ergo2{1,ergcnt}
(indxs),'or','MarkerFaceColor','r')
     hold on
```

```
        plot(time(indxs),offsets_ergo2{1,ergcnt}
(indxs),'og','MarkerFaceColor','g')
        set(gca,'YLim',[0 1.5])
```




1.5
1
0.5
0
5
5.2
5.4
5.6
5.8
6
×10^4
1.5
1
0.5
0
5
5.2
5.4
5.6
5.8
6
×10^4
1.5
1
0.5
0
5
5.2
5.4
5.6
5.8
6
×10^4
1.5
1
0.5
0
5
5.2
5.4
5.6
5.8
6
×10^4



```
end
```

CATEGORIZE THE FEEDBACK ONSET TIMES BASED ON DURATIONS

Based on the calculated durations, determine the feedback modality (verbal, gestual) and type (congruent, incongruent).

```
triginfo_file =
 fullfile(filesep,'Users','bolger','Documents','work','Projects','Project-
PEPs','Brain_IHM_Lists.xlsx');  %Path to excel file with trigger info.

% Initialize variables.
onsets_all = cell(1,size(ergsig1,2));
fbtypes_all = cell(1,size(ergsig1,2));
lats_all = cell(1,size(ergsig1,2));
onset_data = cell(1,size(ergsig1,2));
Allergo_onsets = cell(1,size(ergsig1,2));

bloc_num = length(filenom2);
durs_erg1onset = cell(1,bloc_num);
durs_erg2onset = cell(1, bloc_num);
ideg1=cell(1,bloc_num);
ideg2=cell(1,bloc_num);
outlier_eg1=cell(1,bloc_num);
```

```
outlier_eg2=cell(1,bloc_num);
trialnumbers = cell(1,bloc_num);
fb_table = cell(1,bloc_num);

trignames = {'Gestual-cong' 'Verbal-cong' 'Gestual-incong' 'Verbal-
incong'}; % The 4 conditions.

for bcnt = 1:bloc_num

    % Read in information on Lists, corresponding videos and trigger-
types
    % from xlsx file.
    ix = strfind(filenom2{1,bcnt},'_');
    sheetnom = filenom2{1,bcnt}(ix(1)+1:ix(2)-1);
    XIn =
 readtable(triginfo_file,'FileType','spreadsheet', 'sheet',sheetnom, 'ReadVariable
        'TreatAsEmpty','NA', 'Range','A1:F5');
    display(XIn);   %print to screen a table showing the properties of
 current list.
    fprintf('The current film is:\t%s\n', char(XIn{bcnt,1}));   %print
 to screen the current film
```

```
Warning: Variable names were modified to make them valid MATLAB
 identifiers. The
original names are saved in the VariableDescriptions property.

XIn =

  4×5 table

                     Film_Type               ERG1_Congruent
 ERG2_Congruent    ERG1_Incongruent    ERG2_Incongruent

             __________________________    ______________

 ______________    ________________    ________________


    Film1    'Curare_Human_Congruent'         200                  160
               NaN                   NaN
    Film2    'Perf_Agent_Congruent'           200                  160
               NaN                   NaN
    Film3    'Reg_Agent_Incongruent'          200                  160
               400                   320
    Film4    'Curare_Human_Incongruent'       200                  160
               400                   320

The current film is: Curare_Human_Congruent

Warning: Variable names were modified to make them valid MATLAB
 identifiers. The
original names are saved in the VariableDescriptions property.

XIn =

  4×5 table
```

```
                     Film_Type            ERG1_Congruent
 ERG2_Congruent    ERG1_Incongruent    ERG2_Incongruent
             __________________________    ______________

 ______________    ________________    ________________

    Film1    'Curare_Human_Congruent'          200               160
               NaN                 NaN
    Film2    'Perf_Agent_Congruent'            200               160
               NaN                 NaN
    Film3    'Reg_Agent_Incongruent'           200               160
               400                 320
    Film4    'Curare_Human_Incongruent'        200               160
               400                 320

The current film is: Perf_Agent_Congruent

Warning: Variable names were modified to make them valid MATLAB
 identifiers. The
original names are saved in the VariableDescriptions property.

XIn =

  4×5 table

                     Film_Type            ERG1_Congruent
 ERG2_Congruent    ERG1_Incongruent    ERG2_Incongruent
             __________________________    ______________

 ______________    ________________    ________________

    Film1    'Curare_Human_Congruent'          200               160
               NaN                 NaN
    Film2    'Perf_Agent_Congruent'            200               160
               NaN                 NaN
    Film3    'Reg_Agent_Incongruent'           200               160
               400                 320
    Film4    'Curare_Human_Incongruent'        200               160
               400                 320

The current film is: Reg_Agent_Incongruent

Warning: Variable names were modified to make them valid MATLAB
 identifiers. The
original names are saved in the VariableDescriptions property.

XIn =

  4×5 table

                     Film_Type            ERG1_Congruent
 ERG2_Congruent    ERG1_Incongruent    ERG2_Incongruent
             __________________________    ______________

 ______________    ________________    ________________
```

```
    Film1    'Curare_Human_Congruent'          200                160
               NaN                   NaN
    Film2    'Perf_Agent_Congruent'            200                160
               NaN                   NaN
    Film3    'Reg_Agent_Incongruent'           200                160
               400                   320
    Film4    'Curare_Human_Incongruent'        200                160
               400                   320

The current film is: Curare_Human_Incongruent
```

DETECT THE TRIGGERS BASED ON VIDEO-TYPE CORRESPONDING TO THE CURRENT BLOCK.

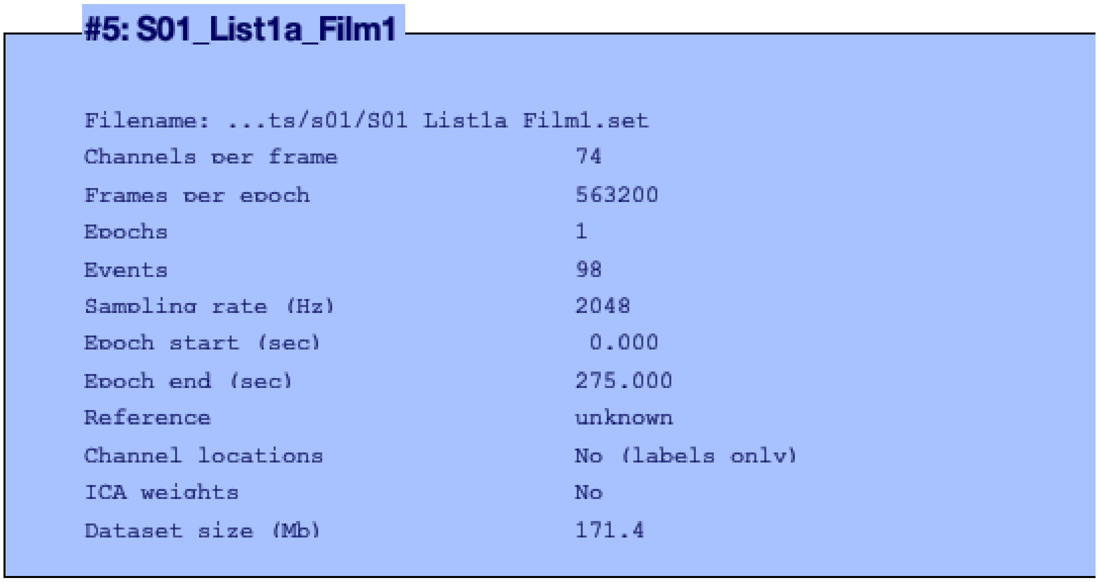
```
    alltrigs = XIn{bcnt,2:5};
    itrig = ~isnan(XIn{bcnt,2:5});
    trigs_curr = alltrigs(itrig);
    trignom_curr = trignames(itrig);

    EG1durs = onoffsets_ergo1{1,bcnt}(:,3);  %durations of photodiode
 signal for ergo1
    EG2durs = onoffsets_ergo2{1,bcnt}(:,3);  %durations of photodiode
 signal for ergo2
    if length(trigs_curr)==2
        ideg1{1,bcnt} = EG1durs<=160 | EG1durs>=320;
        ideg2{1,bcnt} = EG2durs<=80  | EG2durs>=200;
    elseif length(trigs_curr)==4
        ideg1{1,bcnt} = EG1durs<=160 | EG1durs>=460;
        ideg2{1,bcnt} = EG2durs<=80  | EG2durs>=400;
    end
    durs_erg1onset{1,bcnt} = EG1durs;
    durs_erg2onset{1,bcnt} = EG2durs;
    outlier_eg1{1,bcnt} = EG1durs(ideg1{1,bcnt});
    outlier_eg2{1,bcnt} = EG2durs(ideg2{1,bcnt});
    %     onoffsets_ergo1{:,bcnt} = onoffsets_ergo1{:,bcnt}
(~ideg1{1,bcnt});
    %     onoffsets_ergo2{:,bcnt} = onoffsets_ergo2{:,bcnt}
(~ideg2{1,bcnt});

    %ERGO1 (gestual) to begin with...
    if length(trigs_curr)<=2
        trig_erg1incong = 400;
        trig_erg2incong = 320;
    else
        trig_erg1incong = trigs_curr(3);
        trig_erg2incong = trigs_curr(4);
    end

    erg1incong_diff = arrayfun(@(eg1) abs(eg1-trig_erg1incong),
 durs_erg1onset{1,bcnt});
```

```
   erg1cong_diff = arrayfun(@(eg1) abs(eg1-trigs_curr(1)),
durs_erg1onset{1,bcnt});
   [erg1min, ierg1] = min([erg1cong_diff, erg1incong_diff],[],2);
 % 1=congruent; 2=incongruent


   %ERGO2 (verbal)...
   erg2incong_diff = arrayfun(@(eg2) abs(eg2-trig_erg2incong),
durs_erg2onset{1,bcnt});
   erg2cong_diff = arrayfun(@(eg2) abs(eg2-trigs_curr(2)),
durs_erg2onset{1,bcnt});
   [erg2min, ierg2] = min([erg2cong_diff, erg2incong_diff], [],2);
 % 1=congruent; 2=incongruent

   % Determine the latencies for the ERGO1 and ERGO2 onsets
   time = Times_bloc{bcnt,1};
   erg1onsets = onoffsets_ergo1{:,bcnt}(:,1);
   erg1onsets_cong = erg1onsets(ierg1==1);   %ergo1 congruent onset
times
   erg1onsets_incong = erg1onsets(ierg1==2); %ergo1 incongruent onset
times

   erg2onsets = onoffsets_ergo2{:,bcnt}(:,1);
   erg2onsets_cong = erg2onsets(ierg2==1);   %ergo2 congruent onset
times
   erg2onsets_incong = erg2onsets(ierg2==2); %ergo2 incongruent onset
times

   if isempty(erg1onsets_incong) || isempty(erg2onsets_incong)

       dumerg1_cong = ones(length(erg1onsets_cong),1);
       dumerg2_cong = ones(length(erg2onsets_cong),1).*2;
       Allergs = [erg1onsets_cong; erg2onsets_cong];
       Alldums = [dumerg1_cong; dumerg2_cong];
       all_idegs = [ideg1{1,bcnt};ideg2{1,bcnt}];
       dum_erg = [Allergs, Alldums];
       [onsets_all{1,bcnt},idx] = sort(dum_erg(:,1),'ascend');
       idum_all = Alldums(idx,:);  %1=vis_cong, 2=aud_cong
       idegs_all = all_idegs(idx,:);
   else

       dumerg1_cong = ones(length(erg1onsets_cong),1);
       dumerg2_cong = ones(length(erg2onsets_cong),1).*2;
       dumerg1_incong = ones(length(erg1onsets_incong),1).*3;
       dumerg2_incong = ones(length(erg2onsets_incong),1).*4;
       Allergs = [erg1onsets_cong; erg2onsets_cong;
erg1onsets_incong; erg2onsets_incong];
       Alldums = [dumerg1_cong; dumerg2_cong; dumerg1_incong;
dumerg2_incong];
       all_idegs = [ideg1{1,bcnt};ideg2{1,bcnt}];
       dum_erg = [Allergs, Alldums];
       [onsets_all{1,bcnt},idx] = sort(dum_erg(:,1),'ascend');
       idum_all = Alldums(idx,:);  %1=vis_cong, 2=aud_cong,
3=vis_incong, 4=aud_incong
```

```
        idegs_all = all_idegs(idx,:);


    end


    tdiff = arrayfun(@(tin) abs(tin-
onsets_all{1,bcnt}),time, 'UniformOutput', false);
    idx_t = cellfun(@(l) sum(l==0), tdiff);
    Allergo_onsets = time(idx_t==1);
    fbtype = cell(length(Allergo_onsets),1);
    lats = find(idx_t);

    %Assign trigger names to the onset times
    for i = 1:length(Allergo_onsets)

        if idegs_all(i)==1
            fbtype{i,1}=strcat(trignames{1,idum_all(i)},'-outlier');
        elseif ~ismember(trignames{1,idum_all(i)}, trignom_curr)
            fbtype{i,1} = strcat(trignames{1,idum_all(i)},'-outlier');
        else
            fbtype{i,1} = trignames{1,idum_all(i)};
        end

    end

    fbtypes_all{1,bcnt} = fbtype;
    lats_all{1,bcnt} = Allergo_onsets;

    onset_data{1,bcnt}.video = string(fb_sessions{1,bcnt});
    onset_data{1,bcnt}.onset_times = onsets_all{1,bcnt}';
    onset_data{1,bcnt}.latencies = lats_all{1,bcnt}';
    onset_data{1,bcnt}.types = fbtypes_all{1,bcnt}';

    for cntr = 1:length(lats_all{1,bcnt})
        ALLEEG(bcnt).event(cntr).type = string(fbtypes_all{1,bcnt}
(cntr));
        ALLEEG(bcnt).event(cntr).latency = lats(cntr);
        ALLEEG(bcnt).event(cntr).urevent = cntr;
        ALLEEG(bcnt).urevent(cntr).type = string(fbtypes_all{1,bcnt}
(cntr));
        ALLEEG(bcnt).urevent(cntr).latency = lats(cntr);
    end

    [ALLEEG, ~, CURRENTSET] = pop_newset(ALLEEG, ALLEEG(bcnt),
 CURRENTSET,'setname',char(fb_sessions{1,bcnt}),'gui','off');
    [ALLEEG, EEG] = eeg_store(ALLEEG, ALLEEG(bcnt), CURRENTSET);
    EEG = eeg_checkset( EEG );
    EEG =
 pop_saveset( EEG, 'filename',char(fb_sessions{1,bcnt}),'filepath',Savebase);
    eeglab redraw
```

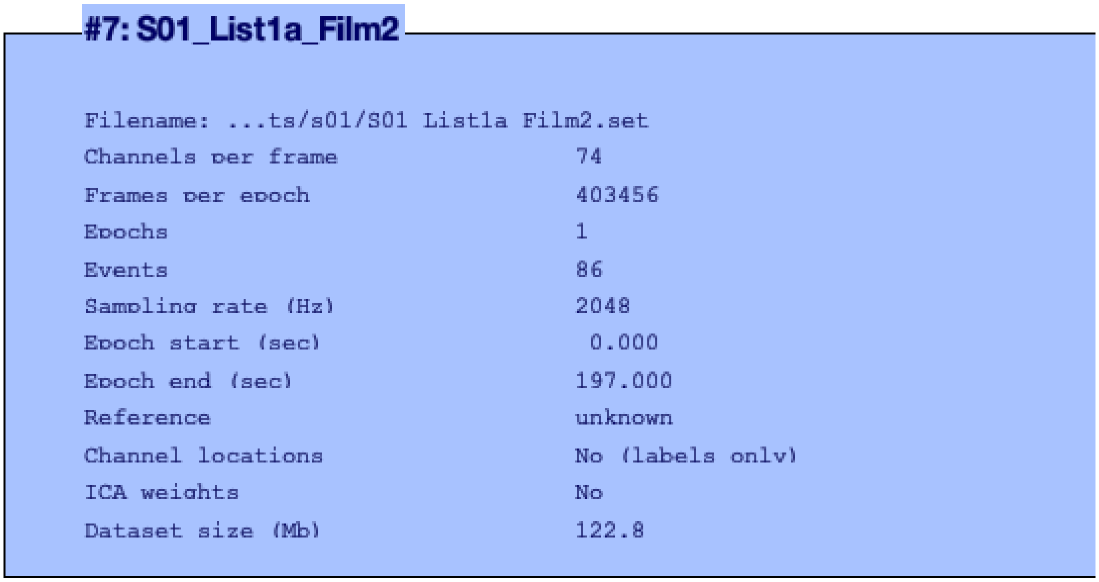
```
Warning: converting all event types to strings
Creating a new ALLEEG dataset 5
```

```
Warning: converting all event types to strings
Saving dataset...
```



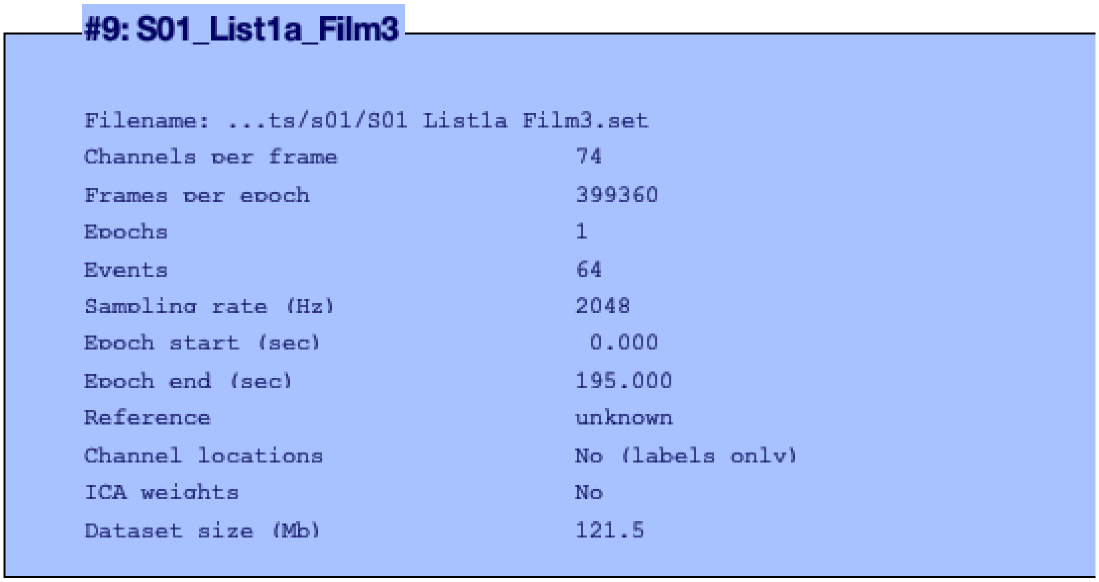
```
#5: S01_List1a_Film1

Filename: ...ts/s01/S01 List1a Film1.set
Channels per frame                  74
Frames per epoch                    563200
Epochs                              1
Events                              98
Sampling rate (Hz)                  2048
Epoch start (sec)                    0.000
Epoch end (sec)                     275.000
Reference                           unknown
Channel locations                   No (labels only)
ICA weights                         No
Dataset size (Mb)                   171.4
```

```
Warning: converting all event types to strings
Creating a new ALLEEG dataset 7
Warning: converting all event types to strings
Saving dataset...
```



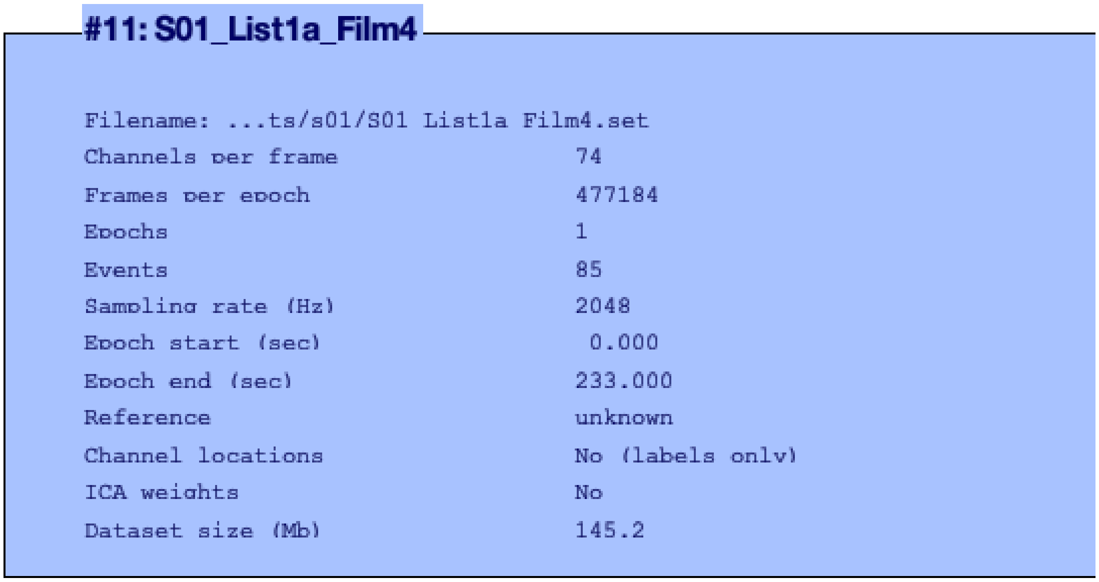
```
#7: S01_List1a_Film2

Filename: ...ts/s01/S01 List1a Film2.set
Channels per frame                  74
Frames per epoch                    403456
Epochs                              1
Events                              86
Sampling rate (Hz)                  2048
Epoch start (sec)                    0.000
Epoch end (sec)                     197.000
Reference                           unknown
Channel locations                   No (labels only)
ICA weights                         No
Dataset size (Mb)                   122.8
```

```
Warning: converting all event types to strings
Creating a new ALLEEG dataset 9
Warning: converting all event types to strings
Saving dataset...
```



```
#9: S01_List1a_Film3

Filename: ...ts/s01/S01 List1a Film3.set
Channels per frame                 74
Frames per epoch                   399360
Epochs                             1
Events                             64
Sampling rate (Hz)                 2048
Epoch start (sec)                   0.000
Epoch end (sec)                    195.000
Reference                          unknown
Channel locations                  No (labels only)
ICA weights                        No
Dataset size (Mb)                  121.5
```

```
Warning: converting all event types to strings
Creating a new ALLEEG dataset 11
Warning: converting all event types to strings
Saving dataset...
```



```
#11: S01_List1a_Film4

Filename: ...ts/s01/S01 List1a Film4.set
Channels per frame                 74
Frames per epoch                   477184
Epochs                             1
Events                             85
Sampling rate (Hz)                 2048
Epoch start (sec)                   0.000
Epoch end (sec)                    233.000
Reference                          unknown
Channel locations                  No (labels only)
ICA weights                        No
Dataset size (Mb)                  145.2
```

CALCULATE ITI AND DETERMINE FEEDBACKS FALLING INTO THE SAME TRIAL.

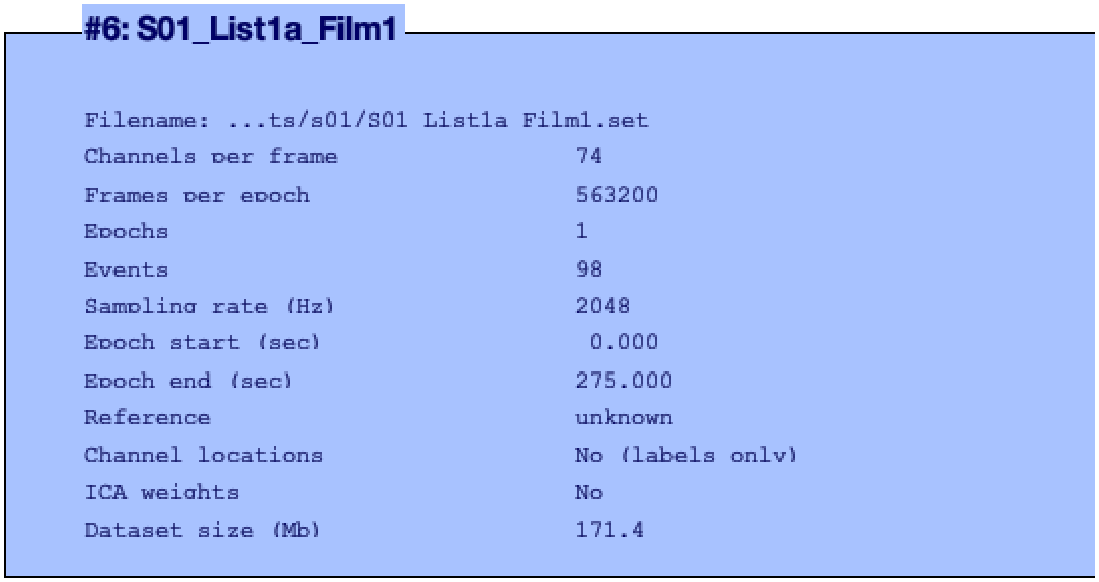
```
   min_dist = 0.1;   %define the minimum distance between two
feedbacks. Below this value, the feedbacks are considered as being in
the same trial.
   diff_lat = diff([ALLEEG(bcnt).event.latency]);
```

```
    difflat_sec = diff_lat./ALLEEG(bcnt).srate;
    trl_dummy = zeros(length(difflat_sec)+1,1);
    trlcounter = 1;

    for trlcnt = 1:length(ALLEEG(bcnt).event)

        if trlcnt==1
            t_end=trlcounter+1;
        end

        if t_end<=length(ALLEEG(bcnt).event)

            diff_curr = (ALLEEG(bcnt).event(t_end).latency-
ALLEEG(bcnt).event(trlcounter).latency)/ALLEEG(bcnt).srate;

            if diff_curr<=min_dist
                trl_dummy(trlcounter:t_end)=trlcnt;
                trlcounter = t_end+1;
                t_end=t_end+2;

            elseif diff_curr> min_dist
                trl_dummy(trlcounter)=trlcnt;
                trlcounter = trlcounter+1;
                t_end=t_end+1;
            end

        elseif t_end>length(ALLEEG(bcnt).event) && diff_curr<=min_dist

            trl_dummy(end) = trl_dummy(end-1);

        elseif t_end>length(ALLEEG(bcnt).event) && diff_curr> min_dist

            trl_dummy(end)=trl_dummy(end-1)+1;
        end

        trl_dummy(end)=trl_dummy(end-1)+1;

    end

    trialnumbers{1,bcnt}=trl_dummy;
```

ADD TRIAL NUMBER INFORMATION TO THE EVENT FIELD OF THE EEG STRUCTURE (EEGLAB based).

```
    for icnt = 1:length([ALLEEG(bcnt).event])

        ALLEEG(bcnt).event(icnt).trialnum = trl_dummy(icnt);

    end
    [ALLEEG, ~, CURRENTSET] = pop_newset(ALLEEG, ALLEEG(bcnt),
 CURRENTSET,'setname',char(fb_sessions{1,bcnt}),'gui','off');
    [ALLEEG, EEG] = eeg_store(ALLEEG, ALLEEG(bcnt), CURRENTSET);
    EEG = eeg_checkset( EEG );
    EEG =
 pop_saveset( EEG, 'filename',char(fb_sessions{1,bcnt}),'filepath',Savebase);
    eeglab redraw
```

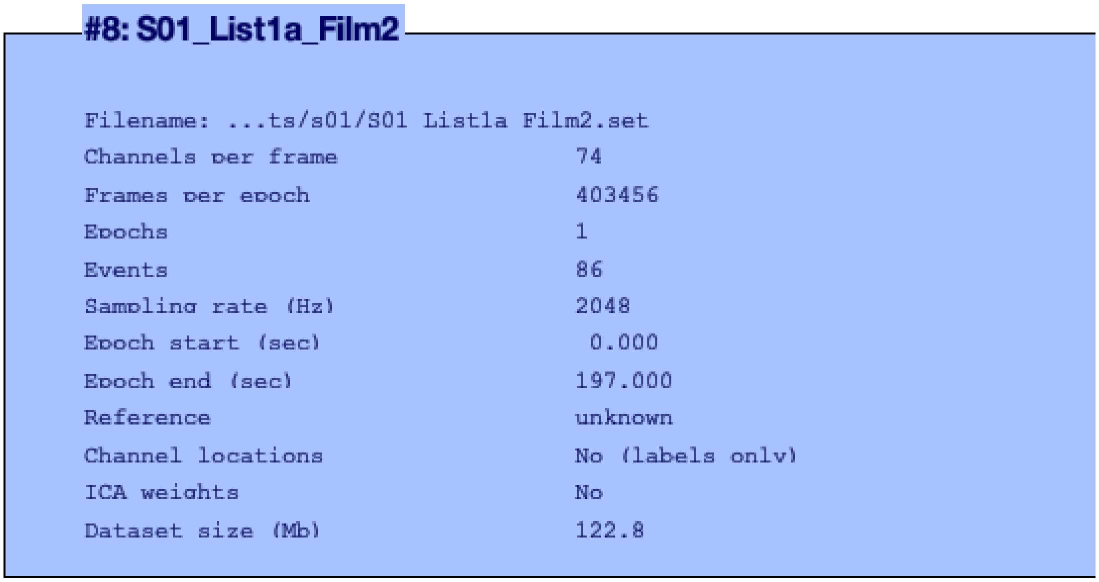
```
Warning: converting all event types to strings
Creating a new ALLEEG dataset 6
Warning: converting all event types to strings
Saving dataset...
```



#6: S01_List1a_Film1

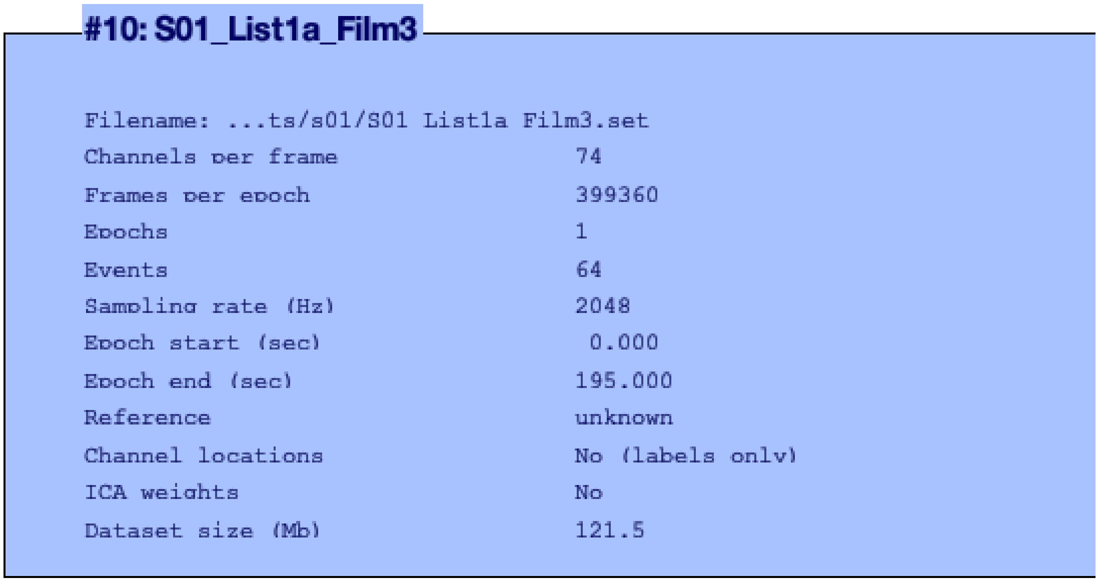
```
Filename: ...ts/s01/S01 List1a Film1.set
Channels per frame                  74
Frames per epoch                    563200
Epochs                              1
Events                              98
Sampling rate (Hz)                  2048
Epoch start (sec)                    0.000
Epoch end (sec)                     275.000
Reference                           unknown
Channel locations                   No (labels only)
ICA weights                         No
Dataset size (Mb)                   171.4
```

```
Warning: converting all event types to strings
Creating a new ALLEEG dataset 8
Warning: converting all event types to strings
Saving dataset...
```



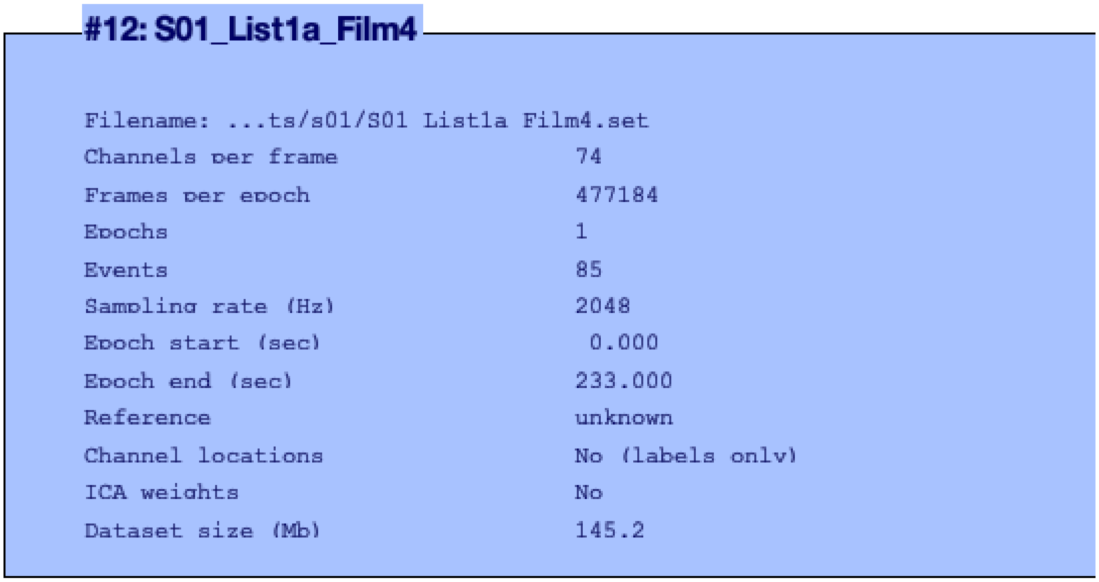
```
#8: S01_List1a_Film2

Filename: ...ts/s01/S01_List1a_Film2.set
Channels per frame                 74
Frames per epoch                   403456
Epochs                             1
Events                             86
Sampling rate (Hz)                 2048
Epoch start (sec)                   0.000
Epoch end (sec)                    197.000
Reference                          unknown
Channel locations                  No (labels only)
ICA weights                        No
Dataset size (Mb)                  122.8
```

```
Warning: converting all event types to strings
Creating a new ALLEEG dataset 10
Warning: converting all event types to strings
Saving dataset...
```



```
#10: S01_List1a_Film3

Filename: ...ts/s01/S01_List1a_Film3.set
Channels per frame                 74
Frames per epoch                   399360
Epochs                             1
Events                             64
Sampling rate (Hz)                 2048
Epoch start (sec)                   0.000
Epoch end (sec)                    195.000
Reference                          unknown
Channel locations                  No (labels only)
ICA weights                        No
Dataset size (Mb)                  121.5
```

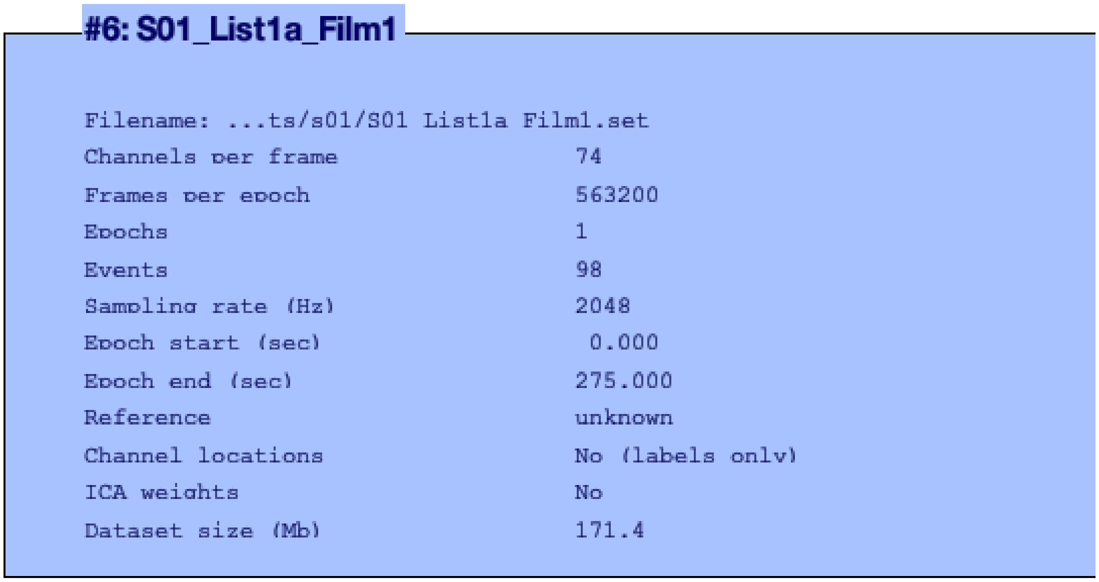
```
Warning: converting all event types to strings
Creating a new ALLEEG dataset 12
Warning: converting all event types to strings
Saving dataset...
```



```
#12: S01_List1a_Film4

Filename: ...ts/s01/S01 List1a Film4.set
Channels per frame                  74
Frames per epoch                    477184
Epochs                              1
Events                              85
Sampling rate (Hz)                  2048
Epoch start (sec)                    0.000
Epoch end (sec)                     233.000
Reference                           unknown
Channel locations                   No (labels only)
ICA weights                         No
Dataset size (Mb)                   145.2
```

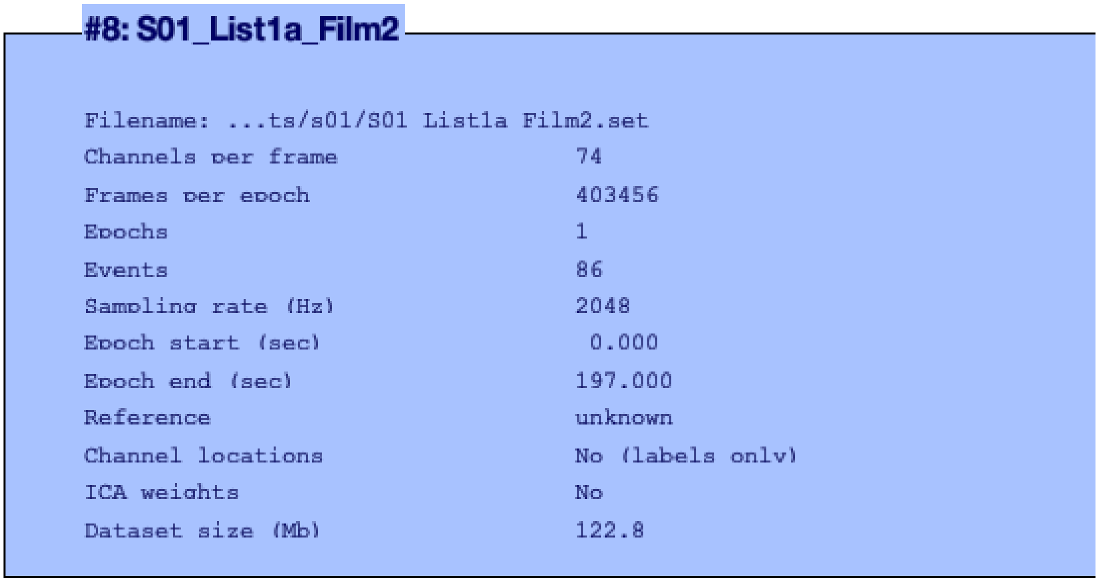
SAVE ALL THE INFORMATION INTO A MAT FILE AS A TABLE FOR EACH FILM

```
Trial_numbers = trl_dummy;
Feedbacks = {EEG.event.type}';
Times = [ALLEEG(bcnt).event.latency]'./ALLEEG(bcnt).srate;
ITI = cat(1,difflat_sec',nan);

fb_table{1,bcnt} = table(Times, Feedbacks, Trial_numbers, ITI);

save(fullfile(Savebase,'feedback_summary.mat'),'fb_table');

eeglab redraw
```



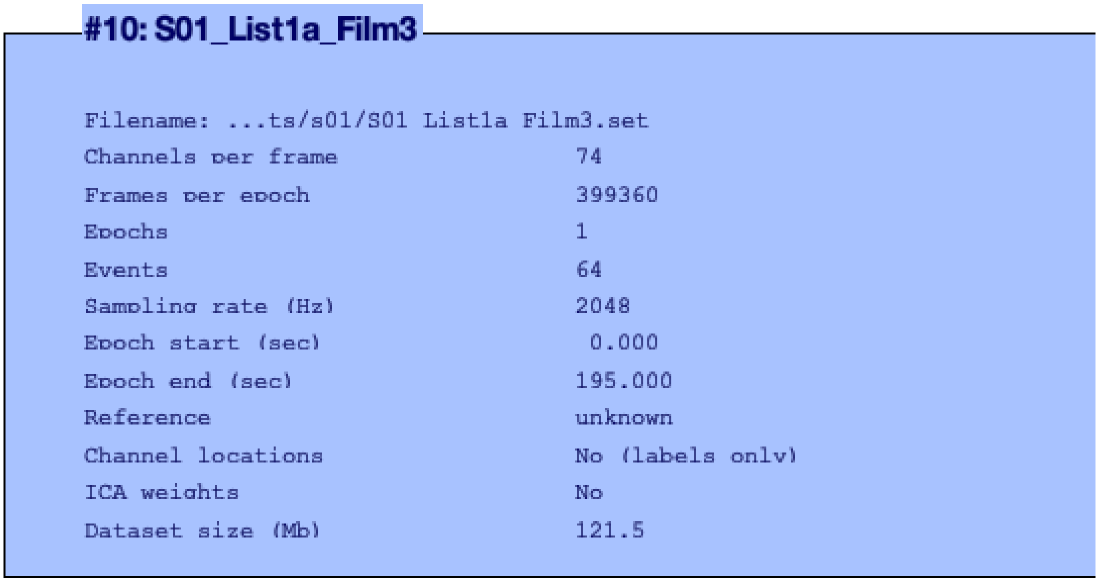
#6: S01_List1a_Film1

```
Filename: ...ts/s01/S01 List1a Film1.set
Channels per frame                  74
Frames per epoch                    563200
Epochs                              1
Events                              98
Sampling rate (Hz)                  2048
Epoch start (sec)                    0.000
Epoch end (sec)                     275.000
Reference                           unknown
Channel locations                   No (labels only)
ICA weights                         No
Dataset size (Mb)                   171.4
```



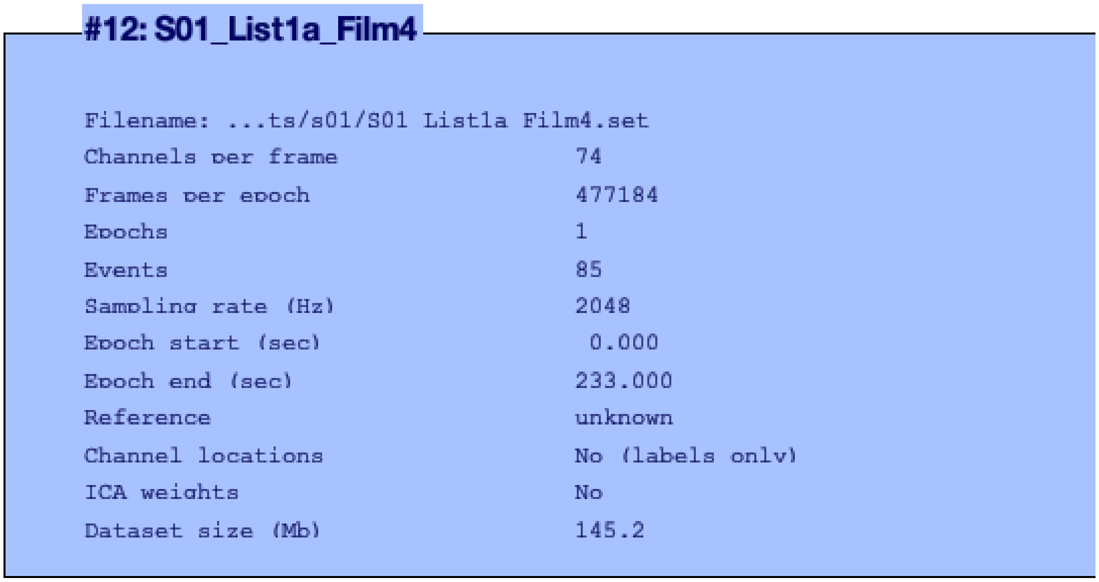
#8: S01_List1a_Film2

```
Filename: ...ts/s01/S01 List1a Film2.set
Channels per frame                  74
Frames per epoch                    403456
Epochs                              1
Events                              86
Sampling rate (Hz)                  2048
Epoch start (sec)                    0.000
Epoch end (sec)                     197.000
Reference                           unknown
Channel locations                   No (labels only)
ICA weights                         No
Dataset size (Mb)                   122.8
```



#10: S01_List1a_Film3

```
Filename: ...ts/s01/S01 List1a Film3.set
Channels per frame                  74
Frames per epoch                    399360
Epochs                              1
Events                              64
Sampling rate (Hz)                  2048
Epoch start (sec)                    0.000
Epoch end (sec)                     195.000
Reference                           unknown
Channel locations                   No (labels only)
ICA weights                         No
Dataset size (Mb)                   121.5
```



#12: S01_List1a_Film4

```
Filename: ...ts/s01/S01 List1a Film4.set
Channels per frame                  74
Frames per epoch                    477184
Epochs                              1
Events                              85
Sampling rate (Hz)                  2048
Epoch start (sec)                    0.000
Epoch end (sec)                     233.000
Reference                           unknown
Channel locations                   No (labels only)
ICA weights                         No
Dataset size (Mb)                   145.2
```

```
end
```

Published with MATLAB® R2018b