BCI Programmer Guide

Angeliki-Ilektra Karaiskou, Robert Reichert, Veerle Schepers

June 2019

Contents

1 Introduction

In this programmer guide you can find all the information necessary to change the code to different settings. We will discuss how to change the classifier, the interface and the
If anything remains unclear after going through the guide you can contact the designers of the system by filing a new issue to our GitHub page.

2 Flowchart of the system

The flowchart of our system is abstractly represented in the next image.

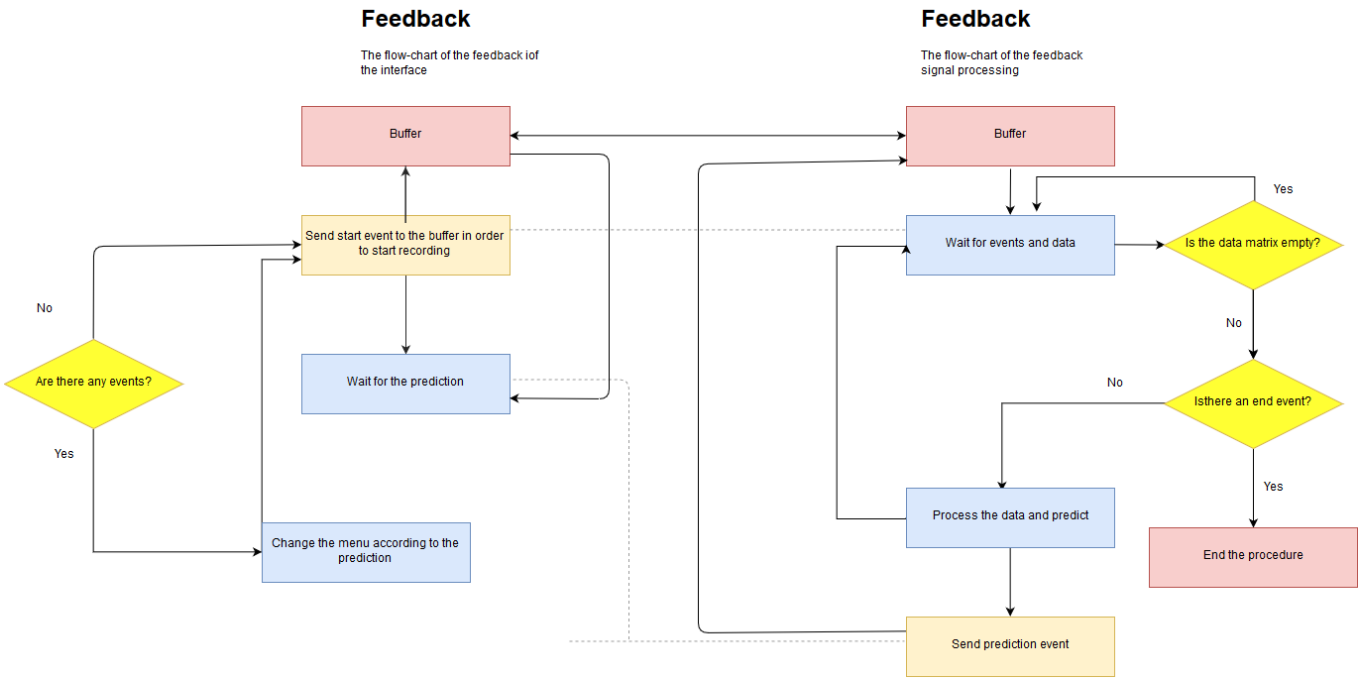


Figure 1: Flowchart of the system

3 Explanation of the code

In this chapter we will go by the code step by step and explain what everything does.

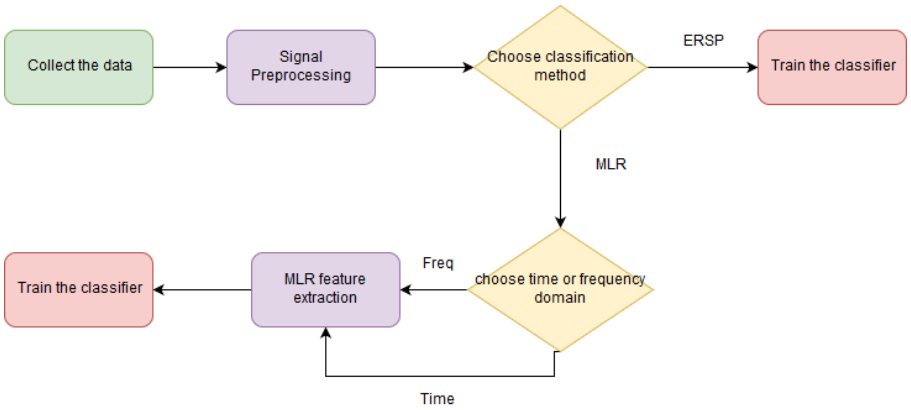
3.1 Explanation of the classification

For the creation of the classifier we collect the events that are sent from the buffer and when the calibration is over we process the data and we train the classifier we are going to use in the calibration phase

3.1.1 Calibration Signal processing

To start the collection of the data we run the script `CalibrationSsvepSigProFinal.m`. Inside the script there is explanation of what each variable does. Here we are providing an overall overview of the script.

- The buffer waits for the appropriate events from the interface and records. The recording time (in milli seconds) is set by changing the variable trialDuration. This is achieved with the function `buffer_waitData`.
- The recorded data then are processed according to the technique asked by the user.
- The user enters the name of the subject using the BCI



analysis pipelin.png

Figure 2: Flowchart of the system

- The user enters the technique used for the training of the classifier between the 'MLR' technique and the 'ERSP' classifier. The default classification technique is 'MLR'.
- If the technique is ERSP then the function `buffer_train_ersp_clsfr` is called to train the ERSP classifier. For more information about this function press help and the name of the function in the command window. One clarification: the variable `freq_band` is defined according to the frequencies used in the calibration.
- For 'MLR' the user has to choose the MLR input. The input can either be the 'freq' spectrum either the 'time' series. The default is 'freq'.
- If the selected method is the MLR then (for both time and freq) the function `SSVEP_sigPrePro_final` is called. The inputs of this function is the recorded data (`data_calibration`), the recorded events (`devents`), the frequency band for the filtering, the channels from the ROI, the technique that was asked ('time' or 'freq') and the information of the buffer (hdr). It returns a structure of the processed data that will be used for the training of the classifier.
- The next step is the feature extraction and the training of the classifier. This is happening by using the function `trainSSVEPclassifier_final`, which requires only one input, the `EEG` structure. The function returns the classifier structure `clssg` and the features (weights and pca coeff) that are required for the testing phase (`features`).
- Then everything is recorded to use them in the feedback phase.

The function `SSVEP_sigPrePro_final`
The detailed explanation of the variables is in the script

- The function `SSVEP_sigPrePro_final` does the basic signal processing and returns: EEG : A structure containg the cleaned data plus the targets and information about the channels that where discarted etc
- At first the data are formed to matrix (from a structure) and are sorted according to their class number.
- Detrend of the data by using the function `detrend`
- Identification of the bad channels (the ones that didn't recieve any signal or were very noisy). This is achieved with the function `idOutliers` from the buffer bci toolbox. Then keep the channels that were discarded in order to discard them also during the feedback phase. For that reason you update the variable `channels2use`.

- Spatial Filtering by using the common average reference.
- Filtering of the data using the frequency band defined in the initial script. The function used for that is again from the buffer bci toolbox (`fftfilter`).
- Identification and discard of the bad trials again by using the function `idOutliers` with different inputs (to see how it works check the help command)
- Extraction of the power spectrum by using the matlab function `fft`. For more information type help fft in the command window.
- Construction of the structure needed for the next steps

The function `trainSSVEPclassifier_final`
In this function occurs the feature extraction and the training of the knn classifier.

- After the normalization of the data the function `pca_func.m` is called to calculate the pca coefficients that have at least 99% power of the total variance. For more information about the procedure check the report and the function (script `pca_func.m`)
- After the mapping of the data, the function `MultiLR.m` is called to calculate the weights that will be used for the projection of the data to construct the feature vectors. For more information about how this procedure is happening check the report and the script of the function.
- **Classification** In order to create the classifier, we use the feature vectors that were created by the weights of MLR. For the training we use the function `fitcknn.m` (press help fitcknn in the command window). To change the options of the classification and so the predictin choice, we can change the following variables inside the classifier. The variables are presenting with their default values:
  - 'NumNeighbors' = `num_neighb` = 3
  - 'NSMethod' = 'exhaustive'
  - Probability distribution of the stimuli 'Prior'= [1 1 1 1 1 1]. With [1 1 1 1 1 1] all the classes have equal probability of being selected. By changing the numbers you give a bias to the classes e.g. if you make it [1 1 1 1 1 2] then class 6 has the biggest probability of being selected. You do this when you see that there is a trend to a class and when a class is never selected. It is something like a cost function
  - 'ScoreTransform' = 'identity'
  - For more information about the options that you can have type in the command window help fitcknn.

3.2 Explanation of the prediction

For the creation of the classifier we collect the events that are sent from the buffer and when the calibration is over we process the data and we train the classifier we are going to use in the calibration phase
To start the collection of the feedback data we run the script `FeedbackSsvepSigProFinal.m`. Inside the script there is explanation of what each variable does. Here we are providing an overall overview of the script. The script is quite similar with the `CalibrationSsvepSigProFinal.m`. According to the technique that used in the calibration phase, the assigned procedure is followed. Nothing is changing here, so for more information look at the commented sections in the scripts. The feedback scripts are `FeedbackSsvepSigProFinal.m`, `applySSVEPclassifier`$_{final.m}$,

`We use one script for the interface during the calibration and several scripts for the interface while running the BCI. We will first go over the calibration interface script, then we look at the initialization script of the running phase and then we will go over the scripts used for the different menus.`

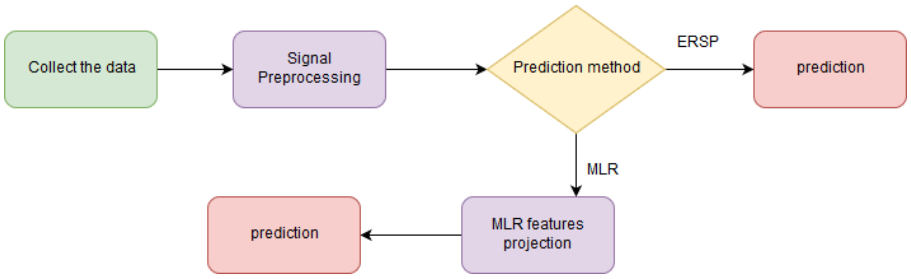


Figure 3: Flowchart of the system

3.2.1 Calibration interface

Below you will find the explanation of each sub-part of the calibration interface script.

- Clear the workspace and close all screens
  This does noting more than closing all open matlab screens (e.g. images etc.) and clear the workspace and command window.
- Recording settings
  Here you find the recording settings. We used 20 runs to get the best result of the classifier. For this we took into account that we needed enough data without reducing the subjects attention. We record for 4 seconds because we found that this gave the optimal results. Instructions to change these settings are found in section 3.3.2.
- Connect to the buffer
  In this part the buffer is connected which makes it able to send events to the buffer, which later can be used to classify the data.
- Psychtoolbox settings
  We choose to use psychtoolbox to achieve better temporal resolution. In this part the computer screens are detected and their specifications are gathered with which the interface window is created. The background color for the interface window is also set here.
- BLOCK FREQUENCY SETTINGS
  The frequencies with which the different blocks are flickering is set here. A zero means that the block is off for the period between one refreshing of the screen, a one means the block is on for this period. The according frequencies can be computed with the formula

$$f = \frac{\#zeros \ + \ \#ones}{screen\ refresh\ rate} \tag{1}$$

  We choose to use frequencies in the Beta range and the higher part of the Alpha range because they seem to result in the strongest brain signals. Instruction to change the frequency settings can be found in section 3.3.1.

  The lower part of this section is used to repeat each frequency signal to achieve one length at which all frequencies end with a full cycle. This is necessary for longer recording where the flickering cycle has to be run multiple times. When a block would not end with a full cycle this would change the overall frequency of the block.

- block position text settings
  The first part of this section is used to set the position and size of the blocks by filling in the horizontal and vertical start and end position for each of the blocks. Instruction to change the position settings can be found in section 3.3.1. The part below is used to set the position of the text within each block. We choose the text positions to be on the side furthest away from the other blocks to reduce interference. In the last part the text itself is set for each of the blocks individually.

- Set instructions
  This section of the code is used to insert the instruction texts.

- time settings
  Here the time each instruction is on the screen is set, as well as the time to the subject should keep his eyes open or closed to record noise. The time we record noise is matched to the time the subject is attending to the stimuli.

- Give instructions
  All settings are set above and now we can start running the script. We start with showing the instruction that the calibration starts. Then the for loop is used to run over each of the different stimuli as well as the noise recording sections. The while loop is used to show the flickering stimuli. The script runs through this while loop for the above set recording time. Each time the screen refreshes you will enter the loop again and each block will be turned on or off according to the above set frequencies.

- End calibration
  Finally we present a text on the screen which indicates to the subject that the calibration phase is finished after which the screen is cleaned.

3.2.2 Feedback initialization interface

We use an initialization script for the feedback phase to make it easier to change certain settings. In this way you only have to change the settings in this scripts, rather then in all 5 scripts created for the different menus. This is however not possible for all settings, which will be discussed in the menu part.

The initialization script for the feedback interface starts in the same way as the calibration interface script and remains the same till time settings. Therefore we will skip these steps. Only one more step is than done which we will explain below.

- Start the main menu of the BCI
  Here only one line of code is run which opens the main menu of the BCI.

3.2.3 Feedback menu interfaces

The only thing that changes between the scripts is which frequency blocks are visible, which texts are shown on the blocks and which events are send to the buffer. This does not change the structure of the scripts and therefore we will only discuss the sub-sections of the main menu script, which contain one extra step at the beginning.

- Psychtoolbox
  This is the only part that is not present in the other menus. We run this initialization of the psychotoolbox settings again because an error occurs on several windows systems if we don't run it here.

- Text settings
  Here the text and positioning of the texts are set for each of the blocks. The text positioning settings are also here rather then in the initialization script because the texts have different lengths which could cause troubles.

- Settings for the feedback loop
  This part of the script is used to be able to receive feedback from the buffer. Without this we could not make any predictions.

- Run flickering blocks loop
  Just as in the calibration script the flickering blocks are turned off and on based on the before set frequencies. The only thing that is different is that a period of showing nothing on the screen is added to indicate when the system is predicting which target was focused on. This is also added because this causes a small time delay which changes the specific frequencies of the blocks.

- Feedback
  The first part of the script looks at the current running time to decide when the acquisition period is over. Then in different scripts the data is analyzed and a prediction is made. This prediction is received and used in the switch loop. Here, based on the prediction the program decides to perform the task related to the stimuli the subject was focusing on. If no prediction could be made, we stay in the same menu and start over again to try and get a better prediction.

- end of menu
  In the last part of these scripts we send an end event to the buffer to indicate that this feedback run is completed. Only when the start signal is send again the buffer will start to collect data again.

3.3 Changing the interface

In this section you will find instructions on how to change the stimuli and the interface.

3.3.1 Stimuli

Certain settings, like colors, size and position can be changed. Below you will find the explanation on how to change this.

Background color We choose to use a black background because it creates the greatest contrast with the white stimuli block, which accordingly gives the strongest brain signals. It is however possible to change the background to a different color. The background color during calibration and using the BCI should be matched to get the best results. Thus, the colors need to be changed in both the calibration interface as in feedback interface menu.

1. Open *Calibration.m*
2. Scroll down to the comment *Psychtoolbox settings*. In the original code this is line number 33.
3. Replace

   ```
   bg = 0;
   ```

   with

   ```
   bg = [R, G, B];
   ```

   where $0 < R, G, B, \leq 255$
   You can now change the amount of red, green and blue present in the background by changing the value of $R, G,$ and B respectively. Lower values mean less of this color is shown in the image.
4. Open *FeedbackInterface.m*
5. Repeat step 2 and 3 in this menu. Make sure to set the background color exactly the same as in the calibration menu to achieve the best results.

stimuli color We choose to use white stimuli on a black background because this gives the greatest contrast and accordingly the strongest brain signals. It is however possible to change the stimuli to different colors. The stimuli colors during calibration and using the BCI should be matched to get the best results. Thus, the colors need to be changed in both the calibration interface as in each of the individual menus used in the feedback phase.

1. Open *Calibration.m*
2. Scroll down to the comment *Flickering blocks screen*. In the original code this is line number 230.
3. For each block you will find the following line of code where # represent a number for each block.

   ```
   Screen('FillRect', window, freq_long(#,k), pos_block#);
   ```

4. You can either change the brightness of the block, or change it completely to a different color.

   - Changing the brightness
     Replace

     ```
     freq_long(#,k)
     ```

     with

     ```
     x*freq_long(#,k)
     ```

     where $0 < x \leq 1$.
     The brightness of the block will become darker for lower values of x.
   - Changing the color
     Replace

     ```
     freq_long(#,k)
     ```

     with

     ```
     [R*freq_long(#,k), G*freq_long(#,k), B*freq_long(#,k)]
     ```

     where $0 < R, G, B \leq 255$
     You can now change the amount of red, green and blue present in each stimuli by changing the value of $R, G,$ and B respectively. Lower values mean less of this color is shown in the image.

5. Make the same changes in the following menus:

   - *FB_Menu01_Main.m*
   - *FB_Menu02_TV.m*
   - *FB_Menu03_Call.m*
   - *FB_Menu04_Move.m*
   - *FB_Menu05_SOS.m*

   You will see that not all blocks are present in each of the menus, thus make sure you change the settings such that they match the block with the same number.

Stimuli size We choose to use the stimuli blocks to be certain size and position based on our screen-size and the optimal size of the stimuli as found in the literature. It is however possible to change the stimuli size and position. The stimuli size and position during calibration and using the BCI should be matched to get the best results. Thus, the size and position need to be changed in both the calibration interface and the feedback interface menu.

1. Open *Calibration.m*
2. Scroll down to the comment *block position settings*. In the original code this is line number 84.
3. For each block you will find the following line of code where # represent a number for each block.

   ```
   pos_block# = [x1 y1 x2 y2];
   ```

   Where $x1$ and $x2$ represent the horizontal star-t and end coordinates, and $y1$ and $y2$ the vertical start- and end coordinates of the block.
4. Change the values according to the position and size you prefer.
5. Open *FeedbackInterface.m*
6. Repeat step 2 to 4 in this menu. Make sure to set the coordinates exactly the same as in the calibration menu to achieve the best results.

Stimuli frequencies We choose to use frequencies in the Beta range and the higher part of the Alpha range because they seem to result in the strongest brain signals. It is however possible to change the flickering frequencies. The frequencies during calibration and using the BCI should be matched to get the best results. Thus, the frequencies need to be changed in both the calibration interface and the feedback interface menu.

1. Open *Calibration.m*
2. Scroll down to the comment *BLOCK FREQUENCY SETTINGS*. In the original code this is line number 48.
3. For each block you will find the following line of code where # represent a number for each block.

   ```
   freq# = [...];
   ```

   Where ... are different amounts of zero's and one's that represent the cycles of turning the block off and on respectively.
4. Change the amount of zeros and ones to change the frequencies.
   The frequencies can be computed with the formula

   $$f = \frac{\#zeros \ + \ \#ones}{screen\ refresh\ rate} \tag{2}$$

5. to find the screen refresh first rate run the section under the comment *Psychtoolbox settings*. In the original code this is line number 33.
6. Then type round(1/ifi) in the command window to get the refresh rate.
7. Open *FeedbackInterface.m*
8. Repeat step 2 and 3 in this menu. Make sure to set the frequencies exactly the same as in the calibration menu to achieve the best results.

3.3.2 Recording settings

It is possible to change the amount of runs you use during the calibration phase as well as how long you record before predicting at which stimuli the subject is looking. Below you will find the instructions on how to change this.

Calibration runs We choose to use 20 runs in the calibration phase to compute the classifier based on the results we got and the literature. It is however possible to change the amount of runs according to your preference. More runs can result in better classifications, but making the runs to long can also reduce the alertness of the subjects which will cause the result to get worse.

1. Open *Calibration.m*
2. Scroll down to the comment *Recording settings*. In the original code this is line number 6.
3. You will find the following line of code

   ```
   runs = 20;
   ```

4. Change the amount of runs by replacing the number 20 with the number of runs you want to have.

Calibration runs We choose to record the brain signals for 4 seconds before predicting to which stimuli the subjects is paying attention based on the literature and the maximum performance of our system. It is however possible to change the recording time according to your preference. The recording time during calibration and using the BCI should be matched to get the best results. Thus, the recording time need to be changed in both the calibration interface and the feedback interface menu.

1. Open *Calibration.m*
2. Scroll down to the comment *Recording settings*. In the original code this is line number 6.
3. You will find the following line of code

   ```
   sec = 4;
   ```

4. Change the recording time by replacing the number 4 with the number of seconds you want to record.
5. Open *FeedbackInterface.m*
6. Repeat step 2 to 4 in this menu. Make sure to set the recording time exactly the same as in the calibration menu to achieve the best results.