MEET: MODULAR EEG PROCESSING TOOLBOX

J. Šťastný

[1]FPGA Laboratory, Department of Circuit Theory, Faculty of Electrotechnical Engineering, Czech Technical University in Prague, Technická 2, 166 27 Prague 6, Czech Republic

Abstract

The contribution describes an EEG Matlab signal processing toolbox implementing all the operations needed to carry out single-trial off-line EEG classification experiments. Besides application in the author's research the toolbox has been already used by students of our laboratory as a starting point for their theses. The toolbox is designed in a modular way allowing students to simply replace parts of the system by their own implementations.

1 Introduction

All existing Brain Computer Interface (BCI) prototypes suffer from too slow communication channel between a human brain and a computer working with Information Transfer Rate (ITR) lower than 100 bits per minute. One way to improve the ITR is the recognition of more distinct brain states – transferring more bits per state – *high-resolution EEG recognition*. Our research deals with high-resolution movement recognition from the EEG signal; for more information see [5]. To execute all experiments a dedicated software for EEG processing had had to be developed. As our experiments grew and we needed to enlarge our team, we felt that it is necessary to create a set of basic general-purpose tools available for new team members not to start from scratch. As most of new colleagues are students, the envisioned library of functions had to be as simple as possible to allow steep learning curve. Finaly we decided to write a dedicated toolbox for Matlab cross-platform environment – Modular EEg processing Toolbox (MEET).

The development of the whole system started as far as in 1999 in the frame of author's Master's thesis. At these days, there was no satisfactory EEG processing software for Matlab available. In between, many similar EEG processing systems appeared; among others eg the EEGLab toolbox [2] or Bioelectromagnetism Matlab toolbox available at SourceForge. However, none of them is specifically targeted to the needs of our research (eg HMM integration is not present in any of them). Compared to other toolboxes, our system offers extreme simplicity which is important for student's own research work. To keep things simple, the toolbox also does not have any graphical interface to reduce system complexity.

The MEET toolbox supports any existing EEG data format and any experimental protocol/procedure and allows future integration with the being designed real-time EEG processing system. Further, the toolbox architecture allows students to simply extend it.

2 Toolbox Architecture

A configurable architecture with loosely coupled independent software modules was chosen. The whole system has a unified modular architecture reflecting the BCI EEG processing pipeline, see Fig. 1. The modules are autonomous, each module is a set of Matlab scripts exchanging data with other modules via files stored on the hard drive in a predefined directory structure. The names of files and directories are formed in a way to unambigously specify the contents of any file in the system.

The modular architecture of the toolbox allows to simply replace parts of the systems by other algorithms (CSP, ICA for surface filtering and others) without a need to change the other parts of the EEG processing pipeline and simplifies reuse of the code.

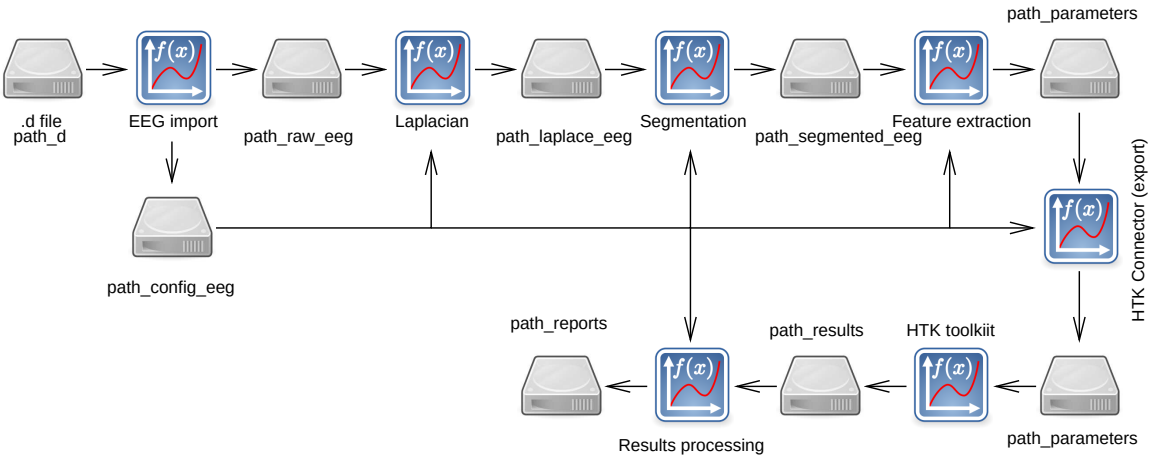

Figure 1: BCI EEG processing pipeline flowchart.

All functions are distributed in the form of a standard Matlab code enabling full visibility into them. Each function also contain a help so as Matlab's standard help command may be used to query function usage. The whole system has one global configuration file `library\configuration.m`. The file is sourced by all the system scripts containing EEG database setup, setup for all the operations, and defining paths to all directories storing intermediate data.

2.1 Initial Setup

The following steps are to be done prior to any work with the toolbox:

1. File `setup_environment.m` is to be configured. This script file adds main `library` directory to the Matlab path so as Matlab may easily find all the needed scripts. Script `setup_environment` is to be run prior to any usage of the toolbox.

2. Then the main EEG data directory is to be defined in the `configuration.m` file. The variable `path_main` is to be set to the root of the EEG database directory structure.

3. Set of experimental subjects, electrodes, and blocks are to be defined by setting up the appropriate variables in the `configuration.m` file.

2.2 EEG Import

One of the requirements on the system was that it shall support any EEG device data format. To fulfill this requirement, data available in the proprietary EEG machine data format are always converted into the Matlab format prior to any other processing. This way only the converter between the native EEG device format and Matlab format is always to be written with any new EEG database.

Currently we support reading of EEG databases in the .d BrainScope EEG format and Alien Technik EEG manufacturer data format. Along with raw EEG, configuration records are generated for each of the subjects using metadata present in the EEG datafiles. The configuration files contain among others sampling frequency, resolution ($LSB/\mu V$), and number of channels along with names of EEG channels and tags. Auxiliary functions `get_fs` and `get_tags` are provided to isolate the user from the real format of the configuration files.

2.3 Surface Filtering

There are a few layers of head tissue in between the brain dipole sources and scalp electrodes – cerebrospinal fluid, head tissues, skull bone, and scalp – resulting in low-frequency band pass spatial filtering of the EEG activity. Due to the conductivity of human head structure the scalp potentials are blurred and any highly localized EEG activity becomes less apparent. This is in contradiction with the need of brain-computer interface experiments requiring EEG activity with as low spatial blurring as possible. The surface filter compensates this blurring to some extent. The discrete Laplacian filter is implemented in our toolbox (function `surface_filtering\filter_all_data`); the Small Surface Laplacian, Large Surface Laplacian, 8-Neighbour Laplacian, Common Average Reference [8], and None filtering are supported including sampling instants realignment [13]. 8-Neighbour Laplacian filter is computed using real coordinates of electrodes on a scalp [10] measured by 3D scanner.

Surface filtering implementation benefits again from the toolbox modular implementation. New types of surface filters were already implemented by our students with ease, including ICA and PCA preprocessing and Common Spatial Pattern extension to the standard Laplacian filtering.

The Laplacian filtering has a dedicated configuration file `library\ surface_filtering_config.m` containing setups of all the filters and also additional symbolic constants allowing to refer to the appropriate filters in a human-readable way.

2.4 Segmentation

The next step in the BCI EEG processing pipeline is segmentation – extraction of the EEG segments containing single trials. Filtered EEG data is segmented based on the tags stored with the original EEG data; the segmentation is performed by the script `segmentation\segmentation.m`. Relevant tag types are defined in the `configuration.m` file. It is usual to use short EEG epochs centered at the time of the movement when movement-related EEG is performed. The segmentation module available in the MEET toolbox support division of the EEG into 10 sec long epochs having the tag indicating desired activity in the fifth second of the epoch.

2.5 Feature Extraction

Our previous results [15] showed that the best results are reachable either with a pure FFT linear spectrum or with AR model coefficients combined with Δ parameters. The feature extraction module `00_paramameterization\spectral_analysis.m` currently supports computation of power spectrum, linear spectrum, delta parameters, and AR model coefficients. The parameters are computed using a sliding window technique; window size and step are given to the function as input parameters.

2.6 HTK Connector And Classification

The Hidden Markov Model classification system is an integral part of the whole toolbox; however, it is not implemented in Matlab – instead, Hidden Markov Toolkit [16], is used. HTK connector module is a set of scripts implementing:

Feature conversion – signal features stored in the Matlab format are written down to the disk in a format suitable for HTK.

Classification results processing – the HTK classification system stores classification results in text files. Toolbox then reads all the results, computes mean recognition scores for all subjects and EEG types along with standard deviations of the scores and produces detailed report on the classification results. The report contains the classification scores sorted according to various criteria (classification score per class, overall classification score, rank of electrodes according to the classification score).

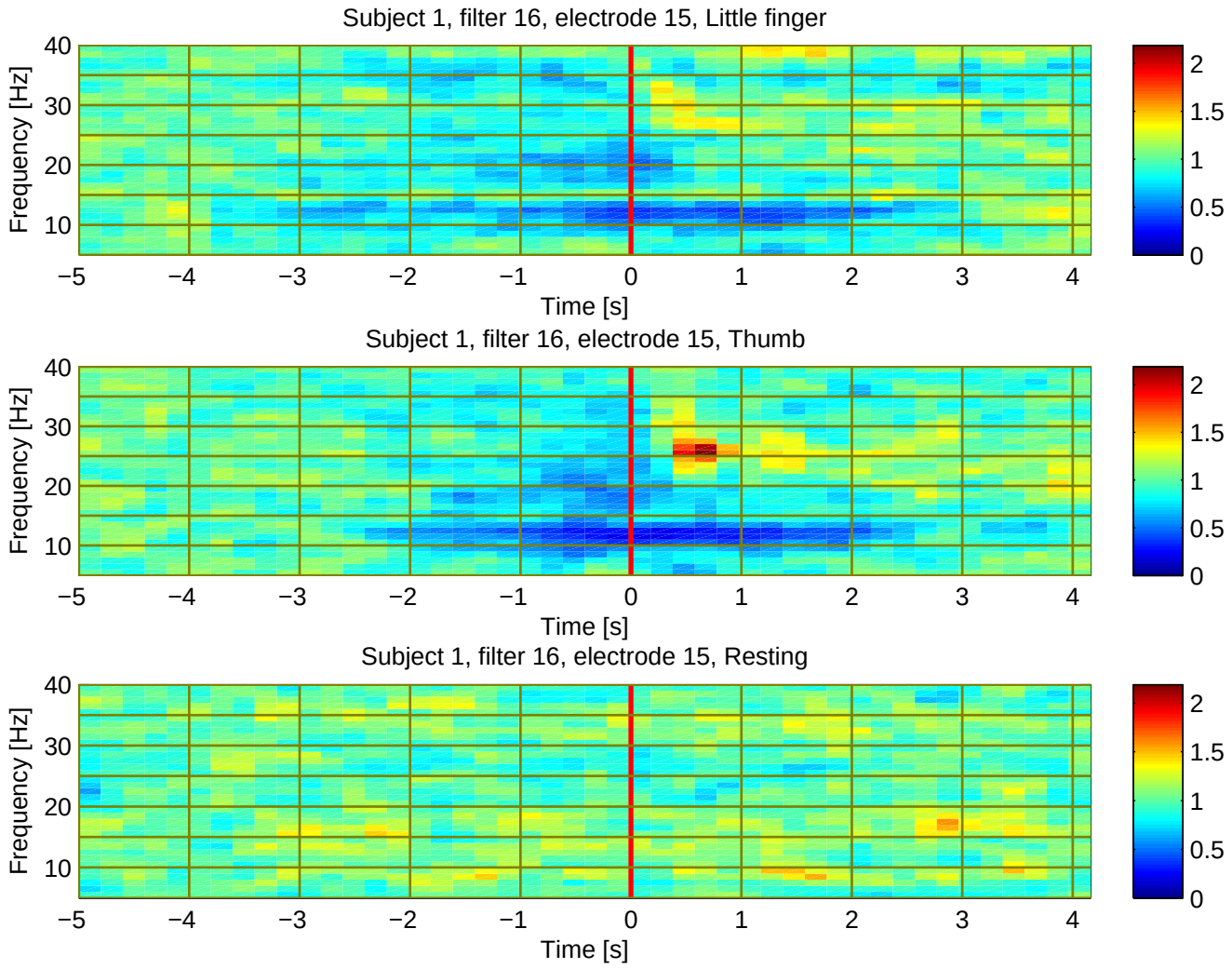

Figure 2: Averaged short-time spectra temporal development of movement-related and resting EEG

HMMs analysis – a dedicated software module is designed to read back Hidden Markov Model definitions as produced by HTK and do an analysis of the model contents. We are mainly interested in average time spent in the model states and envelopes of spectra stored in the emitting states of the models.

Confusion matrices analysis – a set of scripts aggregates all the confusion matrices produced by all the runs of an experiment and computes mean confusion matrix. This matrix gives an information on which type of EEG is interchanged with which by the classifier.

The classification experiment consists of the following steps performed for all subjects, electrodes, and types of parameterization:

1. EEG is parameterized with a selected algorithm.
2. The randomization procedure is applied to combat the *curse of dimensionality* [7]. Each classification experiment is repeated for 16 times with different (and random) division of EEG realizations between the disjunctive training and testing sets. This helps us to get reliable results independently on the concrete selected training and testing EEG realizations.
3. HMMs are trained (initialization, Baum-Welch reestimation) on the training set.
4. Classification is run on the testing set.
5. The average classification scores and standard deviations are computed for all of the EEG types across all the performed experiments.

The toolbox also supports comparing results of more experiments and the implementation of the classification system allows to run more parallel classifications experiments simultaneously to fully utilize nowadays multicore processors.

2.7 Further Processing

Besides the functions mentioned above, the toolbox also provides other supporting functions to facilitate EEG processing:

- Scalp topographic mapping support (using functions from toolbox [2]).
- Analysis and visualisation of short-time EEG spectra, and other additional functions for EEG spectral and temporal analysis, see Figs. 2 and 3.
- Function to generate a unique name for any picture stored by the toolbox and other infrastructure functions to eg obtain metadata of the EEG records.
- Statistical analysis of the linear spectra using Kruskal-Wallis mean value tests.

3 System Performance

An important criterion we had in mind during software development was the speed of EEG processing. Since EEG processing is time-consuming, Matlab profiler was extensively used and critical parts of the toolbox were rewritten to run as fast as possible. During performance review of the system we made the following observations:

- A common practice used by students – array expansion

$$x = [x\, y] \tag{1}$$

  is extremely slow if the vector x is large. A simple remedy is to allocate all vectors prior to their usage using the *zeros* or *ones* constructs.

- The matrix processing speed depends on if matrix is scanned row- or column-wise. The difference is up to tens of percents in processing time. The root cause is the CPU memory caching. Matlab stores data column wise, so row-wise scanning by the algorithm results into more often trashing of the CPU cache and thus reduced performance. This observation was confirmed by the paper [4] later on.

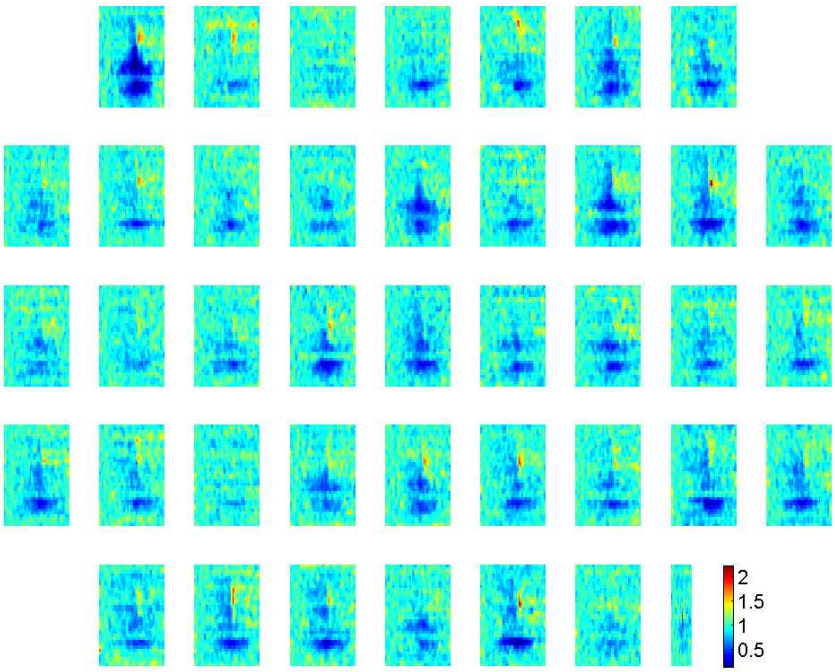

Figure 3: Averaged short-time spectra temporal development of movement-related EEG, scalp montage.

The concept of exchanging data between modules using the disk files was proven as very handy. It allows easily to fully separate data, pictures, and logs from scripts which facilitates backing up of the toolbox and experimental results. The only drawback of the chosen architecture is a need for fairly large disk space (up to ten gigabytes per database if all the parameters and intermediate results are archived), but this is not an issue as disk space is quite cheap nowadays.

4 Conclusions, Future Steps

The whole system was originally developed in the frame of author's own research. The toolbox was used by nearly all of our publications – [5, 6, 13, 14, 9, 3, 11] to cite only few selected ones. Developed code was reused by or served as a starting point for nine master's thesis and four PhD theses. Compared to other EEG toolboxes, students do not have to face huge amount of Matlab code and still obtain already debugged and running code which is simple to extend. The whole toolbox has also a pedagogic dimension as students can have a look at the functions and see how are the algorithms implemented. Both factors were proved as encouraging from students' points of view. Last, but not least, reusing the existing code helps to suppress the "Not invented here" syndrome [1] in the students preparing them better for their future career.

The strictly modular architecture with well defined boundaries between modules was proven as highly advantageous. Even beginners are able to quickly add new functions to the system without a risk of introducing bugs into already running code, also the learning curve of the system is quite steep. In addition to this, it is easily possible to introduce new modules to the system and distribute them amongst students, even if they have an older release of the whole system. The modular concept was reused during the development of our real-time EEG processing software, see [12]; the RTP real time network transfer protocol was used for communication between modules instead of the disk files. The flexibility of the MEET toolbox was also proven by its succesfull application in more distant EEG processing fields during work on research published in [9] (ssMRP HMM classification) and [11] (EEG-based biometric identification). Finally, the modular architecture allows to maintain system integrity in the environment where developers are fluctuating (students usually leave after finishing Master's theses).

We currently plan to develop more tight interaction between the HTK subsystem and the toolbox to support voting and thresholding of classification results. We also still have to rewrite some parts of the code to use column based scanning instead of per-row scanning to speed up matrix access. A support for lossless EEG compression to save disk space is to be added as well. Last, but not least, the future development will be targeted to continuous EEG processing support; we have already formed a team and have started design of real-time EEG processing software which will implement our algorithms.

The whole toolbox is available on request at the author of the article.

5 Acknowledgement

This work has been supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS10/178/OHK3/2T/13. I also would like to thank to Jaromír Doležal for his work done on converting the HTK scripting part to support the latest HTK release. The conversion script for EEG import uses a proprietary Matlab function `dread` for import of .d files originally developed by Jiří Svoboda and then further extended by Tomáš Bořil.

References

[1] Alistair Cockburn. *Agile Software Development: The Cooperative Game*. Addison-Wesley Professional, 2nd edition, 2001.

[2] A. Delorme. EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Journal of Neuroscience Methods*, 134(1):9–21, March 2004.

[3] J. Doležal, J. Šťastný, and P. Sovka. Recording and recognition of movement related EEG signal. Applied Electronics, pages 95–98, Pilsen, Czech Republic, September 2009.

[4] Pascal Getreuer. Writing Fast Matlab Code. Technical report, 2009.

[5] Jakub Šťastný and Pavel Sovka. High-Resolution Movement EEG Classification. *Computational Intelligence and Neuroscience*, 2007. Article ID 54925, 12 pages, doi:10.1155/2007/54925.

[6] Jaromír Doležal, Jakub Šťastný, and Pavel Sovka. Recognition of direction of finger movement from eeg signal using markov models. In *Proceedings of the 3rd European Medical & Biological Engineering Conference - EMBEC05. Prague, Czech Republic*, volume 11 of *IFMBE Proceedings*, pages 1492–1 – 1492–6, Nov. 20-25, 2005.

[7] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1137–1145, 1995.

[8] Dennis J. McFarland, Lynn M. McCane, Stephen V. David, and Jonathan R. Wolpaw. Spatial filter selection for EEG-based communication. *Electroencephalography and clinical Neurophysiology*, (103):386–394, 1997.

[9] K. Nazarpour, J. Šťastný, and R. C. Miall. ssmrp state detection for brain computer interfacing using hidden markov models. In Proceedings of IEEE/SP 15th Workshop on the Statistical Signal Processing, pages pg. 29–32,, Cardiff, August 2009.

[10] Gert Pfurtscheller. *Digital Biosignal Processing*, chapter 17, pages 459–480. Elsevier, 2nd edition, 1991.

[11] J. Šťastný. Brain-computer interface with an automatic user identification, utility model no. 19972, application no. 2009-21380, accepted on the 24th of August, 2009, industrial property office.

[12] Jakub Šťastný, Jaromír Doležal, Vladimír Černý, and Jan Kubový. Design of a modular brain-computer interface. Applied Electronics, pages 312 – 322, Pilsen, Czech Republic, September 2010.

[13] Jakub Šťastný and Pavel Sovka. The 3D approach to the surface Laplacian filtration with integrated sampling error compensation. *Elsevier Signal Processing magazine*, (1):51 – 60, 2007.

[14] Jakub Šťastný, Pavel Sovka, and Andrej Stančák. EEG signal classification: Introduction to the problem. *Radioengineering*, 12(3):51–55, 2003.

[15] Jakub Šťastný, Jiří Zejbrdlich, and Pavel Sovka. Optimal parameterization selection for the brain-computer interface. In *Proceedings of the 4th WSEAS International Conference of Applications of Electrical Engineering*, WSEAS, pages 300–304, 2005.

[16] Stephen J. Young. *HTK reference manual*. Cambridge university engineering department, 1993.

Jakub Šťastný
stastnj1@seznam.cz, FPGA Laboratory, Department of Circuit Theory, Faculty of Electrotechnical Engineering, Czech Technical University in Prague, Technicka 2, Prague 6 166 27, Czech Republic