SignalPlant: an open signal processing software platform

To cite this article: F Plesinger et al 2016 Physiol. Meas. 37 N38

View the article online for updates and enhancements.

Related content

- Robust detection of heart beats in multimodal data
  Ikaro Silva, Benjamin Moody, Joachim Behar et al.

- RS slope detection algorithm for extraction of heart rate from noisy, multimodal recordings
  Jan Gieratowski, Kamil Ciuchciski, Iga Grzegorczyk et al.

- Open Ephys: an open-source, plugin-based platform for multichannel electrophysiology
  Joshua H Siegle, Aarón Cuevas López, Yogi A Patel et al.

Recent citations

- Very high-frequency oscillations: Novel biomarkers of the epileptogenic zone
  Milan Brázdil et al

- Ventricular dyssynchrony assessment using ultra-high frequency ECG technique
  Pavel Jurak et al
SignalPlant: an open signal processing software platform

F Plesinger, J Jurco, J Halamek and P Jurak

Institute of Scientific Instruments of the Czech Academy of Sciences, Brno, Czech Republic

E-mail: fplesinger@isibrno.cz

Received 19 October 2015, revised 24 March 2016
Accepted for publication 20 April 2016
Published 31 May 2016

Abstract
The growing technical standard of acquisition systems allows the acquisition of large records, often reaching gigabytes or more in size as is the case with whole-day electroencephalograph (EEG) recordings, for example. Although current 64-bit software for signal processing is able to process (e.g. filter, analyze, etc) such data, visual inspection and labeling will probably suffer from rather long latency during the rendering of large portions of recorded signals. For this reason, we have developed SignalPlant—a standalone application for signal inspection, labeling and processing. The main motivation was to supply investigators with a tool allowing fast and interactive work with large multichannel records produced by EEG, electrocardiograph and similar devices. The rendering latency was compared with EEGLAB and proves significantly faster when displaying an image from a large number of samples (e.g. 163-times faster for $75 \times 10^6$ samples). The presented SignalPlant software is available free and does not depend on any other computation software. Furthermore, it can be extended with plugins by third parties ensuring its adaptability to future research tasks and new data formats.

Keywords: data visualization, software, signal processing, ECG, EEG

Online supplementary data available from stacks.iop.org/PM/37/N38/mmedia

(Some figures may appear in colour only in the online journal)

Original content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
1. Introduction

In this paper, we present SignalPlant (version 1.2.1.20), a new free software research tool for interactive visual inspection, parallel processing and analysis of multichannel records. It is oriented (but not limited) towards biological signals such as ECG or EEG signals and others. SignalPlant registration forms show that it had been installed more than 240 times in 53 countries around the world in both academic and private institutions by March 2016.

Effective research into biological signals such as ECG and EEG is strongly dependent on the software providing the visual inspection, analysis and labeling of recordings. The rapid inspection of recorded signals does not only require quick signal loading and processing, but also requires fast rendering connected to a responsive user interface which becomes an issue when it comes to long, high-density recordings.

Researchers usually use software such as Matlab® (MATLAB 2012), Octave (Eaton et al 2015) or Scilab (Scilab Enterprises 2012) for analysis of recorded signals. Although these tools are perfectly equipped for computational experimentation, they do not allow fast inspection of larger portions of, for example, the 32-channel, 2000 Hz, 24 h EEG recordings used in epilepsy research. Even on a multicore workstation (8-cores/16-thread CPU at 2 GHz) the resultant display latencies will probably exceed acceptable limits for fluent work (Nielsen 1993) which is also the case for the widely used Matlab® toolbox EEGLAB (Delorme and Makeig 2004).

Stand-alone applications faster from the viewpoint of rendering and responsivity, such as SigViewer (Vidaurre et al 2011) and RASCHlab (Schneider et al 2004), have been developed in the past. Although these programs provide an extremely fast pan operation (i.e. movement of the displayed window), the zoom operation to a specific location requires a series of additional menu or button actions, thereby decreasing the efficiency of workflow. Another weakness of RASCHlab is its 32-bit architecture (limiting allocable memory to 2–4 GB (Microsoft 2016a)); SigViewer, on the other hand, provides no evidence of plugin extendibility.

The above mentioned software tools did not meet our requirements for efficient work with high-density records (such as the 5–25 kHz, 300–1800 s, 16–30 channel ECG recordings used for ventricle dyssynchrony estimation (Jurak et al 2015a, 2015b) or the 5–25 kHz, 120–180 channel intracranial EEG records for brain connectivity and epilepsy research). For this reason, we have developed SignalPlant, new stand-alone software with the following characteristics:

- fast, interactive inspection of multichannel and high-density recordings
- use of triggering and classification marks for manual or automatic definition of EEG epochs, artifacts or ECG classification, event-related potentials (ERP) or synchronization/desynchronization (ERS/ERD), ECG signal averaging (SAECG), RR analyses, etc
- 64-bit architecture, allowing work with large files (>4 GB)
- non-destructive signal editing, making signal experimentation easier
- a clean graphical user interface (GUI) which is friendly to non-IT personnel
- extendability by plugins, meaning that software functionality as well as input/output formats may be improved by third parties in the future
- a sufficient number of filtering and analysis tools in the default plugin set
- batch processing of multiple records
- independent of commercial computational software

SignalPlant has been developed and tested over a period of two years, allowing us (and collaborating institutions) rapid inspection of high-density data in ultra-high-frequency ECG projects, as well as effective analysis of intracranial EEG recordings for connectivity and
epilepsy research. SignalPlant was publicly released in September 2015, since when it has also begun to be used for analysis of non-biological measurements. We believe the proposed software fills a gap in effective signal inspection and analysis; we also believe it can increase productivity for other researchers, as it has in our case.

2. Software description

SignalPlant software is being developed using the C# programming language and .NET framework. Its program GUI is displayed in figure 1, showing a multimodal record and multichannel FFT analysis plugin window.

2.2. Data formats

SignalPlant is designed to work with several data formats; a generally known subset is shown in table 1. Hierarchical Data Format (HDF5) (The HDF Group 1997) was chosen as the preferred format to store data from SignalPlant. It is also possible to load Matlab files, though Matlab® must be installed on the target computer for this functionality. The widely-used European Data Format (EDF) (Kemp and Olivan 2015) is also supported. EDF supports the use of different sampling frequencies for single channels which is not allowed by SignalPlant. When such a situation occurs, all channels are up-sampled to the highest sampling frequency.

Figure 1. SignalPlant main window showing part of a multimodal record (34 channels, 21 min, 500 Hz recorded by an experimental device for pulse-wave velocity estimation). (A)—area for displaying signals shows blood pressure (BP), heart sounds (AUDIO), ECG channels (V1–V6, where V6 is switched to show filtered layer) and impedance channels (Z1–Z4). Blue verticals are marks produced by SignalPlant QRS detector. (B)—channel list with several channels selected, (C)—navigation bar, (D)—FFT plugin showing frequency analysis of channels V1–V5.
For publishing purposes, the lossless PNG (Randers-Pehrson and Boutell 1999) format is used for exporting displayed signals. A useful feature is export screen to vector graphic formats such as ‘Scalable Vector Graphics’ (SVG) (W3C World Wide Web Consortium 2011) and Adobe® ‘Encapsulated PostScript’ (EPS) (Adobe 1992).

An important feature is the fact that third parties can extend input/output (I/O) capabilities to any other format. Samples on the project website contain a plugin source code for *.csv I/O operations.

### 2.3. Data structure for non-destructive editing

When a record is loaded into SignalPlant, each signal is held in one instance of special object (graph panel—GP) responsible for displaying. When data is modified (e.g. using some filter), the new layer of the signal is added to the parent GP, meaning the ability of non-destructive editing. Any number of such layers can be added, limited only by the available memory. The user can choose which layer of each signal is to be displayed. This feature allows the user to (for example) apply several different filters to the source signal and quickly see the differences. An editable list of signal layers holds information about these layers, such as creation time and creation plugin name (e.g. FIR LP) and settings (e.g. cut-off frequency and order).

Another part of the data structure is a list of marks (i.e. labels). These may be used, for example, for triggering in EEG, as QRS annotation marks in ECG, or as artifact labels. They may be created manually by the user or may be produced automatically by plugins. These marks have several properties (such as ‘Info’, ‘Group’, ‘Validity’, etc) and may have both left and right borders defined. This allows marks to be used for interval selection (e.g. artifact selection in EEG). Marks can be both imported and exported as a text file (with the *.sel extension). This means, for example, that QRS marks detected by SignalPlant can be transferred to another piece of software.

### 2.4. Navigation in data and asynchronous display engine

A major requirement for SignalPlant was a fast inspection of large recordings. Therefore, an agile navigation in data is important. SignalPlant allows several navigation methods using mouse...
in display area (figure 1(A)), mouse in navigation bar (figure 1(C)) or keyboard (all navigation options are described in SignalPlant manual). Here, we were partially inspired by navigation control used in music recording/production software such as Tracktion (Loud Technologies 2007) and Cubase (Steinberg Media Technologies 2016), where the user may navigate in the data using only a mouse (mouse movements or mouse wheel plus the Shift and Control keys). This approach tends to be more user-friendly than the approach used in EEGLAB, RASCHLab, SigViewer or Physionet LightWave (Goldberger et al. 2000), where the user has to click through the menu(s) or repetitively use GUI buttons or sliders to zoom in on a specific detail and back out again. On the other hand, an agile mouse or touch oriented control requires an effective display engine.

In SignalPlant (using the default display plugin), each channel (i.e. signal) is drawn independently of the others and runs its own asynchronous thread (figure 2). When a zoom/pan operation or data update calls for a display refresh, each thread is activated and its priority reconsidered (visible channels on the screen have the highest priority). Next, the minimum and maximum values in the corresponding signal interval are found for each displayed sample on the screen. For long records, precomputed minimum and maximum values are used. Once the arrays of minimal and maximal values (at the physical length of the graph in pixels on the screen) are found, graph redrawing is launched and the asynchronous thread is set back to idle status. Graph redrawing is performed on the main GUI thread, though this consists (mostly) of the quick drawing of minimal and maximal value lines and the filling of the inner area. The channel closest to the mouse or touch pointer is favored over the others and is processed first with the highest priority. This provides the feeling of live navigation in the data.

2.5. Plugins

SignalPlant is extendable by plugins which can be written by third parties. Plugins can be used to increase post-processing functionality (to implement new detection methods, filters, etc), as well as to adapt I/O capabilities to new file formats. Furthermore, plugins can also be used to replace a default display engine without any changes to the application core. They must be compiled in the form of a dynamic-linked library (DLL).

Plugins usually run in a separate window (figure 3) and allow a real-time preview. The ability to save presets is essential for productive work, especially when plugins offer a large number of settings. Presets are available (if this is allowed by the developer of the plugin)

---

**Figure 2.** Flow diagram of display actualization. Data from each channel is placed in one ‘Graph Panel’ component (GP). When a request for a refresh occurs, preparation for display is run asynchronously for all GPs, favoring the one closest to the mouse/touch pointer.
from the plugin window main menu (figure 3—left). They are stored in XML form and can, therefore, be easily transferred between users.

2.6. Selection of channels and marks using expressions (querying)

SignalPlant allows one to do specific tasks with a subset of loaded data (e.g. filtering of specific channels only). We equipped SignalPlant with several ways to select specific channels or marks.

When the user needs to select channels to display or to link them to a plugin, this can be performed with common drag & drop operation. It is, however, useful in some cases (such as EEG records with 100–200 channels) to select channels by an expression that can be written in the text field at the top of the channel list (figure 1(B)) or inside the form for channel selection.

Marks querying is useful when the user needs to work with specific marks only. As marks may have multiple properties defined, they can be used for querying as in figure 3 with the MultiSketch plugin, where only QRS complexes of a specific morphology group were used for image generation. The expression for marks querying is set in the specific window (figure 4) showing the location of the resultant marks (figure 4(B)) based on the current expression (figure 4(A)). Another example of marks querying is shown in supplement S7 (stacks.iop.org/PM/37/N38/mmedia).

2.7. Scripting and batch processing

SignalPlant allows the use of simple scripts for the automation of repetitive tasks or for batch processing of multiple records. Using scripts, it is possible to launch a sequence of commands defined by plugins.
3. Results

3.1. Software and hardware requirements

- Operating system: 64-bit Windows® 7 or newer with .NET framework 4.5 or newer
- Multi-core CPU
- Enough memory (due to parallelism and non-destructive processing capabilities)
- Mouse or touch screen preferred

3.2. Availability, license and installation

The software is available free at https://signalplant.codeplex.com along with sample ECG files and documentation. SignalPlant is available under an MIT license. Because one of SignalPlant’s features is plugin extendability, the source code of a sample I/O plugin (for *.csv import and export) and Visual Studio template (for plugin with GUI) are also available for download.

The current software version (1.2.1.20) does not use any installer. The archive from the project website should be unpacked and is prepared for execution. Simple and quick registration is required when the software is executed for the first time.

The standard plugin set offers 30 plugins with GUI (for filtering, analysis, detection, maintenance, etc) and 12 plugins for file-format extensions. A list of current plugins is available in S1 of the supplementary data (stacks.iop.org/PM/37/N38/mmedia).

3.3. SignalPlant extendability

SignalPlant functionality can be extended by new plugins, written in C# and compiled into DLL. Each DLL may contain multiple plugins as is the case with the standard plugin set. New plugins are loaded during SignalPlant initialization and they can be executed from the ‘Plugins’ folder. The project website contains source codes for several plugins.

3.4. Documentation

SignalPlant is accompanied by a help file describing the deployment process and program elements, as well as the basic plugin set. Short tutorial movies are also available from the project website at https://signalplant.codeplex.com.

---

Figure 4. Marks querying window showing 217 marks satisfying querying conditions for mark group (#GROUP) and specific time interval (#TIME). (A)—area for querying expression, (B)—live preview showing location of marks selected by expression, (C)—buttons with common querying conditions.
4. Discussion

Comparison of the presented software and existing tools is shown in Table 2. Interactive signal inspection is not achieved in EEGLAB (where the user must hit the ‘forward/backward’ buttons to navigate); in the other software, the user can navigate in the signal more intuitively by dragging the signal (SignalPlant, SigViewer) or using the scroll bar (RASCHLab). Instant magnification of specific area (using the mouse wheel or by selection to window) can be performed only in SignalPlant. Other applications may distract users; they have to change scale using other dialog windows (EEGLAB, RASCHLab) or using a sequence of button clicks (SigViewer). 64-bit architecture is needed to analyze large records and only RASCHLab does not meet this requirement. The dependence on Matlab® is also compared because it may influence the research project budget. The weakness of SignalPlant is its platform dependency on a .NET framework which is (in full) currently available in the Windows® operating system only. Added functionality without changing application code (i.e. plugins) is available with RASCHLab and SignalPlant. The difference lies in user comfort. For example, selecting a signal for frequency analysis in RASCHLab is performed by redefining variable(s) using a dialog window, while in SignalPlant the user merely drags a signal (or signals) into the plugin window (there are also other options). The result (in the case of frequency analysis) is immediately shown and reacts instantly to changes made in the plugin window or to a new ruler position in the main program window.

Display latency (Figure 5) was compared for SignalPlant 1.2.1.20 and EEGLAB 13.5.4b, running with Matlab® R2012a. Both applications were changed to store rendering latencies. The testing computer was an HP Z820 workstation with CPU Xeon E5-2650 at 2 GHz (8 cores/16 threads) and 64 GB of RAM. The dimensions of the area displaying signals were 2155 × 1056 pixels and the graphic adapter used was an NVIDIA Quadro K4000. The operating system was Windows® 7 Professional. In Figure 5, it is shown that SignalPlant is usually able to refresh high-density data more efficiently than EEGLAB. The low rendering latency allows fluent navigation in the signals. Accelerated rendering, such as OpenGL (OpenGL Architecture Review Board 1993), was not used as the refresh delays achieved are sufficient for fluent work and are not dependent on a powerful graphic processing unit (GPU). On the other hand, we should reconsider this in the future as accelerated graphics can take part of the current CPU load. A comparison of SignalPlant running on different hardware is presented in supplement S2.

We also tested one of the in-built plugins—a finite-impulse-response filter (FIR). This and other filtering plugins process signals using multiple CPU threads. This is also the case when signals are filtered using FIR with EEGLAB (resp. Matlab®). FIR filter (100th order) was

<table>
<thead>
<tr>
<th>Software</th>
<th>User-friendly inspection</th>
<th>64-bit</th>
<th>Matlab® independent</th>
<th>Multiplatform</th>
<th>Plugins extendability</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEGLAB</td>
<td>—</td>
<td>Yes</td>
<td>—</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RASCHLab</td>
<td>Pan only</td>
<td>—</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SigViewer</td>
<td>Pan only</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>—</td>
</tr>
<tr>
<td>SignalPlant</td>
<td>Pan and zoom</td>
<td>Yes</td>
<td>Yes</td>
<td>—</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 2. Comparison of SignalPlant, EEGLAB (Delorme and Makeig 2004), RASCHlab (Schneider et al 2004) and SigViewer (Vidaurre et al 2011).
set to band-pass mode (20–80 Hz). The average filtering time (20 trials) of the testing record (259 channels EEG file, length of 250 s, 1 kHz sampling) was $2.29 \pm 0.08$ s for EEGLAB, while SignalPlant computed the same record in $4.93 \pm 0.77$ s. This shows that Matlab® implementation of the FIR filter is more effective than SignalPlant implementation.

4.1. Use of other signal types

Although SignalPlant was developed primarily for work with biological signals, and first and foremost for EEG and ECG processing, other signal types may also be processed. This applies, for example, to the time-frequency analysis of sound recordings (S3 of the supplementary data (stacks.iop.org/PM/37/N38/mmedia)) or statistical analysis of cryogenic experiments (S6 of the supplementary data (stacks.iop.org/PM/37/N38/mmedia)). The availability of a specific I/O plugin may prove a limitation. In the case of biological signals, the BIOSIG project (Vidaurre et al 2011) may be used for data conversion to, for example, the EDF format. Otherwise, the *.CSV format may be used, though this is not recommended due to its text-based nature leading to excessive data consumption with large files. Another limitation is the need for a fixed sampling frequency for all signals in a record, as well as all signals being of the same length.

Figure 5. Comparison of display latency for SignalPlant (solid line) and EEGLab (dashed line). The X-axis represents the number of samples rendered per channel. The results show lower latencies of SignalPlant, allowing fluent display control even with large number of displayed samples. SignalPlant displays channels in a multi-thread, asynchronous manner; the channel closest to the mouse (touch) pointer is drawn first (with a latency of tens of milliseconds in all tested cases) while other channels are rendered asynchronously. Here, we present the SignalPlant latency of the slowest channel. (A)—Each sample is drawn. (B)—displayed samples are optimized as mentioned in chapter 2.4. Each evaluated delay is the average value of 20 trials excepting last two EEGLAB values.
4.2. SignalPlant memory requirements

The number of channels and layers in SignalPlant is limited only by the available memory (technically, the maximal numbers of channels as well as maximal number of layers in each channel is $2^{31}$ (Microsoft 2016)). Because SignalPlant is a 64-bit application, it can allocate a maximum of 8 GB–4 TB of memory depending on the OS version (Microsoft 2016a). SignalPlant uses 32-bit floating point datatype to store signal values and, therefore, each signal layer is limited to hold $2^{31}$ samples. SignalPlant loads all data into the memory, which is one of the conditions for fast display refresh. Data miniatures are precomputed, consuming additional memory. Therefore, our approach consumes a slightly larger amount of memory than the real size of the data (in case of HDF5 files). As an example, a record with 30 channels and 32-bit quantization, 25 kHz sampling rate, 40 min long is saved in a non-compressed HDF5 file of a size of 7.14 GB. After loading and the precomputing of miniatures, the memory consumed by SignalPlant is 7.7 GB of RAM. If the user merely investigates this record, no other memory will be allocated, though if the user decides to filter all signals, a larger amount of memory will be allocated during multi-thread filtering (the exact amount is dependent on available system resources and is driven by the .NET framework). When the filtering process ends and .NET framework clears unused memory blocks allocated during processing, the amount of memory consumed by SignalPlant is 15.37 GB. This number reflects the fact that each channel now contains an additional layer with a filtered signal.

In case of loading EDF files, users should expect that the amount of requested memory will be higher than the file size. It is caused by use of a 32-bit floating point datatype (while EDF is usually 16-bit), request of constant sampling frequency over all channels (while signals in EDF may differ in sampling) and some amount of memory needed for data miniature. As an example, a Physionet (Goldberger et al. 2000) record ‘s32701-3247-02-13-00-43’ downloaded as a 2.11 GB EDF file will consume 4.56 GB of RAM (4.33 GB for data and 240 MB for data miniatures and SignalPlant data structure). However, when the amount of available memory is an issue, the user can select which signals and which data area will be loaded from the EDF record into SignalPlant.

5. Conclusion

We have presented SignalPlant 1.2.1.20 in this paper. It is a free, stand-alone software application for researchers. It has an interactive GUI allowing fast visual inspection and is aimed at the processing and analysis of large multichannel records. Its main highlights are display speed, interactive plugins with real-time preview, 64-bit and multi-threaded architecture, function and file-format extendability by plugins, non-destructive processing and clean GUI.

Acknowledgments

Development was supported by the projects GACR P103/11/0933, MEYS CR (LO1212) and GACR P102/12/2034.

References

Steinberg Media Technologies 2016 Cubase Elements 8 (www.steinberg.net/en/products/cubase/line_up/cubase_elements_8.html)


MATLAB 2012 Version R2012a (Natick, MA: The MathWorks Inc.)


Nielsen J 1993 Usability Engineering (San Francisco, CA: Morgan Kaufmann Publishers)


Scilab Enterprises 2012 Scilab: Free and Open Source Software for Numerical Computation (France: Orsay)

The HDF Group 1997 Hierarchical Data Format, version 5 (www.hdfgroup.org/HDF5/)


W3C World Wide Web Consortium 2011 Scalable Vector Graphics (SVG) 1.1 2nd edn vol 1 (www.w3.org/TR/SVG/)


N48