The Journal of Open Source Software

Biosiglive: an Open-Source Python Package for Real-time Biosignal Processing

Amedeo Ceglia [1], **Felipe Verdugo** [2], **and Mickael Begon** [1]

1 Institute of Biomedical Engineering, Faculty of Medicine, University of Montreal, Canada **2** Faculty of Music, University of Montreal, Canada

DOI: 10.21105/joss.05091

Software

- Review
- Repository
- Archive

Editor: Kevin M. Moerman

Reviewers:

- @finsberg
- @marcoghislieri

Submitted: 02 December 2022
Published: 07 March 2023

Summary

biosiglive aims to provide a simple and efficient way to access and process biomechanical data in real time. It was conceived as user-friendly software aimed for both non-expert and expert programmers. The library uses interfaces to access data from several sources, such as motion capture software or any Python software development kit (SDK). Some interfaces are already implemented for Vicon Nexus motion capture (Oxford, UK) and Delsys electromyography SDK (EMG) (Boston, USA). That say, any additional interface can be added as custom interface using the abstract class. biosiglive was designed for biosignals, therefore, existing classes represent data collected from standard acquisition systems in biomechanics, such as markers for motion capture or EMG. Methods are available to process in real-time any input signal. Data can be saved in a binary file at each time frame to avoid any data loss in case of system shutdown. Data can also be displayed using the LivePlot class, which is based on PyQtGraph (C++ core) and allows, therefore, fast real-time displaying. Finally, 'biosiglive' was conceived as a flexible real-time data processing and streaming tool adaptable to various set-ups, software, and systems. Therefore, a TCP/IP connection module was implemented to send data to a distant port to be used by any other system.

Statement of Need

Biosignals such as electromyography (EMG) or marker kinematic data are often used to assess human movement in clinical, sports, or artistic contexts. However, the analysis is often time-consuming and requires a good knowledge of programming languages such as MATLAB (Mathworks LCC, Natick, USA) or Python. In the last decade, some open-source tools have emerged to facilitate the analysis of these signals, like biomechanical toolkit (Barre & Armand, 2014), biomechzoo (Dixon et al., 2017), pyomeca (Martinez et al., 2020), or Kinetics toolkit (Chénier, 2021). Since biomechzoo relies on MATLAB, a closed-source software , not every biomechanist can benefit from this tool. The Python environment is, on the other hand, entirely free and allows anyone to use the package without any cost. pyomeca and Kinetics toolkit both provide efficient methods to analyze biosignals, but are designed to work offline. However, real-time use of these signals is often required to provide task feedback to the user (Giggins et al., 2013) or to control a device (Cozean et al., 1988). In this type of use, access to biosignals easily and in real-time is a crucial point. To our knowledge, no tool dedicated to biomechanical data is available to provide real-time access and processing of these signals. Also, there is numerous platforms where data can come from. So, a package able to stream data from any of these sources should help the use of biosignals on a larger scale (in clinical, rehabilitation, pedagogical, sport, and artistic activities). We have developed biosiglive to facilitate the use of biosignals in real-time. It was achieved by pre-implementing standard data processing and data retrieving from several sources such as Nexus software for motion capture and analogical signals. Pre-implemented processing

methods are customizable (i.e. filters cutoff frequency or moving average window size), the user can also develop his/her own method inside the program. Users can also add an interface module to make 'biosiglive' work with the desired acquisition system. Examples are provided to guide the user and documentation is available.

Features

`biosiglive` is divided into five independent modules. The main features are:

- `Processing`: real-time and offline data processing.
- `Interfaces`: interfaces of standard software such as Vicon Nexus (Oxford, UK) or Delsys Trigno Community (Boston, USA).
- `Visualization`: real-time signal visualization,
- `Streaming pipeline`: pipeline to stream, process, disseminate and save data in real time.
- `File I/O`: saving data in binary format at every time frame.

A Biomechanical example: Electromyographic pipeline

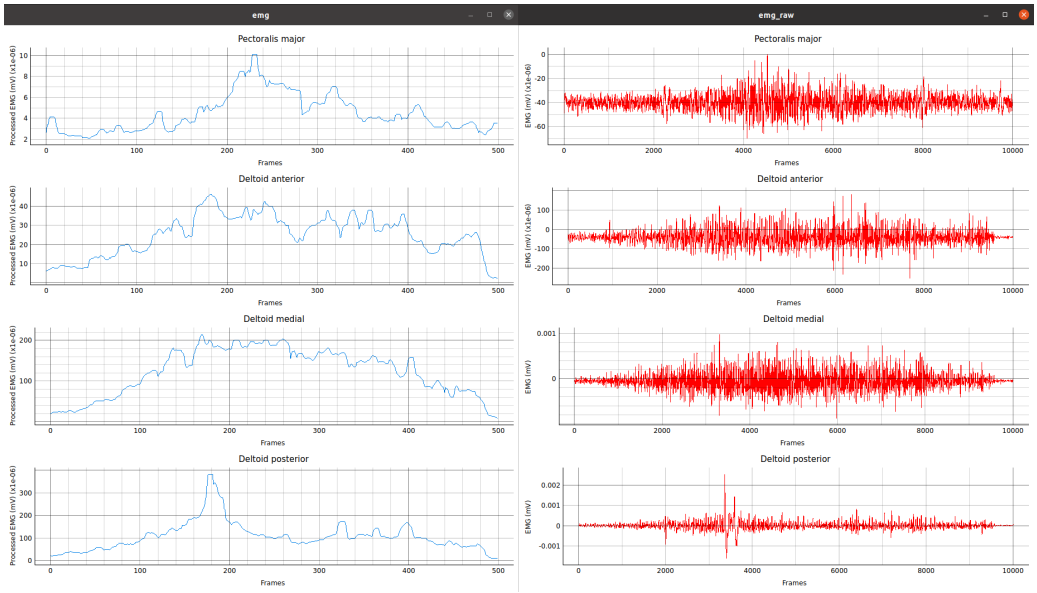
`biosiglive` provides examples for different biomechanical tasks such as getting and processing EMG signals or any generic analog devices from Nexus, compute live cadence from a treadmill, or applying a calibration matrix to raw signals. More advanced examples are available such as computing and showing 3D joint kinematics from a marker set.

The following example shows how to stream, process, display, and save EMG signals from Nexus software.

```
from biosiglive import LivePlot, save , ViconClient, RealTimeProcessingMethod, PlotType

# Define the system from which you want to get the data.
interface = ViconClient(ip="localhost", system_rate=100)
n_electrodes = 4
raw_emg = None
muscle_names = [
    "Pectoralis major",
    "Deltoid anterior",
    "Deltoid medial",
    "Deltoid posterior"
]

# Add device to Vicon interface
interface.add_device(
    nb_channels=n_electrodes,
    device_type="emg",
    name="emg",
    rate=2000,
    method=RealTimeProcessingMethod.ProcessEmg,
    moving_average_window=600
)

# Add plots
emg_plot = LivePlot(
    name="emg",
    rate=100,
    plot_type=PlotType.Curve,
    nb_subplots=n_electrodes,
```

```
    channel_names=muscle_names
)
emg_plot.init(plot_windows=500, y_labels="Processed EMG (mV)")
emg_raw_plot = LivePlot(
    name="emg_raw",
    rate=100,
    plot_type=PlotType.Curve,
    nb_subplots=n_electrodes,
    channel_names=muscle_names
)
emg_raw_plot.init(plot_windows=10000, colors=(255, 0, 0), y_labels="EMG (mV)")

while True:
    # Get data from Vicon interface and process it.
    raw_emg = interface.get_device_data(device_name="emg")
    emg_proc = interface.devices[0].process()
    # Update plots.
    emg_plot.update(emg_proc[:, -1:])
    emg_raw_plot.update(raw_emg)
    # Add data to the binary file.
    save({"raw_emg": raw_emg, "process_emg":emg_proc[:, -1]}, "emg.bio")
```

The live plot is shown in the following figure:



Live display of processed (left) ad raw (right) EMG signals for a 5-second window.

Research Projects Using `biosiglive`

(Verdugo et al., 2022)

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through the CREATE OPSIDIAN program, the NSERC discovery of M. Begon, and the Pôle lavallois d'enseignement supérieur en arts numériques et économie créative , Appel à projet 2020 - projet eMusicorps.

References

Barre, A., & Armand, S. (2014). Biomechanical ToolKit: Open-source framework to visualize and process biomechanical data. *Computer Methods and Programs in Biomedicine*, *114*(1), 80–87. https://doi.org/10.1016/j.cmpb.2014.01.012

Chénier, F. (2021). Kinetics toolkit: An open-source python package to facilitate research in biomechanics. *Journal of Open Source Software*, *6*(66), 3714. https://doi.org/10.21105/joss.03714

Cozean, C., Pease, W. S., & Hubbell, S. (1988). Biofeedback and functional electric stimulation in stroke rehabilitation. *Archives of Physical Medicine and Rehabilitation*, *69*(6), 401–405.

Dixon, P. C., Loh, J. J., Michaud-Paquette, Y., & Pearsall, D. J. (2017). biomechZoo: An open-source toolbox for the processing, analysis, and visualization of biomechanical movement data. *Computer Methods and Programs in Biomedicine*, *140*, 1–10. https://doi.org/10.1016/j.cmpb.2016.11.007

Giggins, O. M., Persson, U. M., & Caulfield, B. (2013). Biofeedback in rehabilitation. *Journal of Neuroengineering and Rehabilitation*, *10*(1), 1–11. https://doi.org/10.1186/1743-0003-10-60

Martinez, R., Michaud, B., & Begon, M. (2020). Pyomeca: An open-source framework for biomechanical analysis. *Journal of Open Source Software*, *5*(53), 2431. https://doi.org/10.21105/joss.02431

Verdugo, F., Ceglia, A., Frisson, C., Burton, A., Begon, M., Gibet, S., & Wanderley, M. M. (2022). Feeling the Effort of Classical Musicians - A Pipeline from Electromyography to Smartphone Vibration for Live Music Performance. *NIME 2022*. https://doi.org/10.21428/92fbeb44.3ce22588