pybdf Documentation

Release alpha3

Samuele Carcagno

July 16, 2012

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION

Author Samuele Carcagno

pybdf is a pure python library to read BIOSEMI 24-bit BDF files. While being slower than alternative C-based libraries like *BioSig <http://biosig.sourceforge.net/>*, it is very easy to install and use.

CHAPTER

TWO

DOWNLOAD AND INSTALLATION

2.1 Download

The latest release of pybdf can be downloaded from the python package index:

http://pypi.python.org/pypi/pybdf/

For developers: the source code of pybdf is hosted on launchpad:

https://launchpad.net/pybdf

2.2 Installation

2.2.1 Requirements

To install pybdf you will need:

- python >= 3.2
- numpy

On all platforms, after having unpacked the archive you can install pybdf by running:

```
python setup.py install
```

On Windows, you can alternatively use the binary installer, if provided. Note that pybdf has not been extensively tested on Windows.

CHAPTER

THREE

USAGE

To open a bdf file you need to create a bdfRecording object as follows:

```
bdf_rec = bdfRecording('res1.bdf')
```

you can then query the properties of therecording stored in the BDF header using the appropriate functions, which are fully described *here*. Some examples are shown below.

Get the duration of the recording:

```
bdf_rec.recordDuration
```

Get the sampling rate of each channel:

```
bdf_rec.sampRate
```

Get the channel labels:

```
bdf_rec.chanLabels
```

There are two functions to read in the data. The first function reads each channel sequentially:

```
rec = bdf_rec.get_data()
```

the second function reads the channels in parallel, and is thus faster on multicore machines:

```
rec = bdf_rec.get_data_parallel()
```

either function returns the same result, that is a python dictionary with the following fields: - data : an array of floats with dimenions nChannels X nDataPoints - trigChan : an array of integers with the triggers in decimal format - statusChan : an array of integers with the status codes in decimal format For example, to get the value of the first sample of the recording, in the first channel, you can type:

```
rec['data'][0,0]
```

the same sample value, but for the second channel, is stored in:

```
rec['data'][1,0]
```

trigChan contains the triggers for the experimental conditions, in decimal format. The statusChan, on the other hand, contains system codes, like cm in/out-of range, battery low/OK.

Other usage examples are provided in the 'examples' directory inside the pybdf source archive.

Beware that pybdf does not check that you have sufficeint RAM to read all the data in a bdf file. If you try to read a file that is too big for your hardware, you system may become slow or unresponsive. Initially try reading only a small amount of data, and check how much RAM that uses. You can read only a portion of the data by passing the beginning

and end arguments to the get_data() or get_data_parallel() functions. For example, to read the first 10 seconds of the recording, use:

```
rec = bdf_rec.get_data_parallel(beginning=0, end=10)
```

CHAPTER

FOUR

BUGS

Please, report any bugs on Launchpad https://launchpad.net/pybdf

Currently there are problems with the get_data_parallel() function on Windows. Please, use the get_data() function instead, or use Linux.

CHAPTER

FIVE

BENCHMARKS

To give you an idea of the speed of pybdf, here are some rough benchmarks.

Using the get_data_parallel() function:

Channels	Duration (s)	Samp. Rate	File (MB)	CPU	Time (s)
9	931	2048	49.1	Intel Core i7-870	10
9	1457	2048	76.8	Intel Core i7-870	15
41	651	2048	156.4	Intel Core i7-870	21
9	931	2048	49.1	Intel Core2 Quad Q6600	16
9	1457	2048	76.8	Intel Core2 Quad Q6600	24
41	651	2048	156.4	Intel Core2 Quad Q6600	31

CHAPTER

SIX

PYBDF – CLASS TO READ BIOSEMI BDF FILES

This module can be used to read the header and data from 24-bit BIOSEMI BDF files recorded with the ActiveTwo system.

```
>>> bdf_rec = bdfRecording('res1.bdf') #create bdfRecording object
>>> bdf_rec.recordDuration #how many seconds the recording lasts
>>> bdf_rec.sampRate #sampling rate for each channel
>>> #read 10 seconds of data from the first two channels
>>> rec = bdf_rec.get_data(channels=[0, 1], beginning=0, end=10)
>>> rec = bdf_rec.get_data_parallel() #read all data using multiprocess
```

class pybdf.**bdfRecording**(*fileName*)

Class for dealing with BIOSEMI 24-bit BDF files. A bdfRecording object is created with the following syntax:

```
>>> bdf_rec = bdfRecording('bdf_file.bdf')
```

This reads the BDF header, but not the data. You need to use the get_data() or get_data_parallel() methods to read the data. The full documentation of the BDF file format can be found here: http://www.biosemi.com/faq/file_format.htm

idCode [str] Identification code

subjId [str] Local subject identification

recId [str] Local recording identification

startDate [str] Recording start date

startTime [str] Recording start time

nBytes [int] Number of bytes occupied by the bdf header

versionDataFormat [str] Version of data format

nDataRecords [int] Number of data records "-1" if unknown

recordDuration [float] Duration of a data record, in seconds

nChannels [int] Number of channels in data record

chanLabels [list of str] Channel labels

transducer [list of str] Transducer type

physDim [str] Physical dimension of channels

physMin [list of int] Physical minimum in units of physical dimension

physMax [list of int] Physical maximum in units of physical dimension

digMin [list of int] Digital minimum

digMax [list of int] Digital maximum

prefilt [list of str] Prefiltering

nSampRec [list of int] Number of samples in each data record

reserved [list of str] Reserved

scaleFactor [list of floats] Scaling factor for digital to physical dimension

sampRate [list of int] Recording sampling rate

statusChanIdx [int] Index of the status channel

nDataChannels [int] Number of data channels containing data (rather than trigger codes)

dataChanLabels [list of str] Labels of the channels containing data (rather than trigger codes)

get_data(*beginning=0*, *end=None*, *channels=None*, *trig=True*, *status=True*, *norm_trig=True*, *norm_status=True*)

Read the data from a bdfRecording object

beginning [int] Start time of data chunk to read (seconds).

end [int] End time of data chunk to read (seconds).

channels [list of integers or strings] Channels to read. Both channel numbers, or channel names are accepted. Note that channel numbers are indexed starting from *zero*.

trig [boolean] If True, return the channel containing the triggers

status [boolean] If True, return the channel containing the status codes

norm_trig [boolean] If True, the trigger channel will only signal *changes* between one trigger status to the next. A trigger value that is equal to the previous one will be set to zero

norm_status [boolean] If True, the status channel will only signal *changes* between one status code to the next. A code value that is equal to the previous one will be set to zero

rec [a dictionary with three keys]

- data : an array of floats with dimenions nChannels X nDataPoints
- trigChan : an array of integers with the triggers in decimal format
- statusChan : an array of integers with the status codes in decimal format
- chanLabels : a list containing the labels of the channels that were read, in the same order they are inserted in the data matrix

```
>>> x = bdfRecording('res1.bdf')
>>> rec = x.get_data(channels=[0, 2], beginning=0, end=10)
```

get_data_parallel(*beginning=0*, *end=None*, *channels=None*, *trig=True*, *status=True*, *norm_trig=True*, *norm_status=True*)

Read the data from a bdfRecording object using the multiprocessing module to exploit multicore machines.

beginning [int] Start time of data chunk to read (seconds).

end [int] End time of data chunk to read (seconds).

channels [list of integers or strings] Channels to read. Both channel numbers, or channel names are accepted. Note that channel numbers are indexed starting from *zero*.

trig [boolean] If True, return the channel containing the triggers

status [boolean] If True, return the channel containing the status codes

norm_trig [boolean] If True, the trigger channel will only signal *changes* between one trigger status to the next. A trigger value that is equal to the previous one will be set to zero

norm_status [boolean] If True, the status channel will only signal *changes* between one status code to the next. A code value that is equal to the previous one will be set to zero

rec [a dictionary with three keys]

- data : an array of floats with dimenions nChannels X nDataPoints
- trigChan : an array of integers with the triggers in decimal format
- statusChan : an array of integers with the status codes in decimal format
- chanLabels : a list containing the labels of the channels that were read, in the same order they are inserted in the data matrix

```
>>> x = bdfRecording('res1.bdf')
>>> rec = x.get_data_parallel(channels=[0, 2], beginning=0, end=10)
```

CHAPTER

SEVEN

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

p

INDEX