Open Neurofeedback Training

Installation Manual v1.0

2017-03-24

Yury Koush, John Ashburner, Evgeny Prilepin, Ronald Sladky, Peter Zeidman, Sergei Bibikov, Frank Scharnowski, Artem Nikonorov, and Dimitri Van De Ville

1 Table of Contents

2 About OpenNFT

OpenNFT is an integrated software package designed for neurofeedback training. It constitutes the core technical framework for developments in this exciting new field of neuroimaging. OpenNFT is based on best practices of Python and Matlab software and incorporates, but is not limited to, the functionality of the SPM and Psychtoolbox software suits. An integrated Python/Matlab framework is specifically selected to address the needs of neurofeedback developers and users with different background, which allows for flexibility in developments and implementations without compromising for speed and functionality. More specifically, the OpenNFT's GUI, synchronization module, and multi-processing core are implemented in Python, whilst computational modules for neurofeedback are implemented in Matlab.

2.1 Download

Download the latest software version and demo data set from our GitHub project page.

2.2 License

OpenNFT Software is open-source and is distributed under GNU GPL v3.0 license.

2.3 Disclaimer

The end user is advised to justify their research or application outcome using pilot data and complementary offline data analyses.

3 Setup Instructions

3.1 Microsoft Windows

Windows	Matlab	Python
7, 8, 10 (32-bit)	2015b (32-bit)	3.4.* (32-bit)
7, 8, 10 (64-bit)	2015b or 2016a	3.4.*
	2016b (recommended)	3.5.* (recommended)

Table 1. Compatibility list for Windows.

3.1.1 All systems

1. Install **Mathworks Matlab** 2015b or higher (2016b recommended)
2. Install a **Python 3** version that is compatible with your Matlab version (see table)

   Download: https://www.python.org/downloads/windows/
3. For convenience, add Python paths (e.g. `c:\Python34\;c:\Python34\Scripts;`) to your system Path environment variable: *System* ▸ *Advanced* ▸ *Environment Variables…*
4. Install **Git SCM**

   Download: https://git-scm.com/downloads
5. Install required **MATLAB toolboxes**
   1. SPM12: http://www.fil.ion.ucl.ac.uk/spm
   2. jsonlab Toolbox: https://www.mathworks.com/matlabcentral/fileexchange/33381-jsonlab--a-toolbox-to-encode-decode-json-files
   3. PsychToolbox: http://psychtoolbox.org/download/

3.1.2 For Python 3.5 setups (with Python virtual environment support)

1. Setup Python 3.5 virtual environments using Windows Command Prompt in your *workspace* directory (i.e., the directory with your neurofeedback projects)

   ```
   cd c:\OpenNFT

   c:\Python35\python -m venv OpenNFT_venv
   ```

   We use *OpenNFT_venv* as convenient name for our virtual environment but you can use any name. If there are multiple python versions installed, you can use the command `python3.5` instead of `python` to ensure that version 3.5 is used when launching Python.

   Activate the new virtual environment by using the corresponding activate script

```
OpenNFT_venv\Scripts\activate.bat
```

Use deactivate.bat to `deactivitate` your active virtual environment.

2. Setup MATLAB Engine API for Python using

```
cd "c:\Program Files\MATLAB\R2016b\extern\engines\python"
python setup.py build --build-base="c:\OpenNFT\OpenNFT_venv\Lib\site-packages\MatlabEngineBuild" install
```

If multiple Matlab installations are present on your system's PATH ensure that the Matlab version that will be used in Python is preferred (i.e., preceding the other versions in the variable definition).

3. Install the required Python packages using the command line tool pip:

```
pip install -U pip setuptools
pip install PyQt5
pip install watchdog
pip install --no-deps git+https://github.com/pyqtgraph/pyqtgraph.git
```

4. Install numpy+MKL (i.e., an optimized numpy version) using a compatible wheel file from, e.g., http://www.silx.org/pub/wheelhouse/

```
pip install http://www.silx.org/pub/wheelhouse/numpy-1.11.3+mkl-cp34-cp34m-win32.whl
```

3.1.3 For Python 3.4 setups (without Python virtual environment support)

1. Setup MATLAB Engine API for Python using Windows Command Prompt (use right click and select *Run as Administrator* or alter access permissions of MATLAB engine folder):

```
cd "c:\Program Files\MATLAB\R2015b\extern\engines\python"
python setup.py install
```

MATLAB Python engine will be installed to your currently active Python environment (e.g., `c:\Python34\Lib\site-packages\matlab\`). In case there are multiple versions installed, ensure that the correct MATLAB and Python versions are installed.

2. Install the required Python packages using the command line tool pip:

```
python -m pip install -U pip
pip install watchdog
pip install --no-deps git+https://github.com/pyqtgraph/pyqtgraph.git
```

3. Install numpy+MKL using a compatible wheel file from e.g., http://www.silx.org/pub/wheelhouse/

```
pip install http://www.silx.org/pub/wheelhouse/numpy-1.11.3+mkl-cp34-cp34m-win32.whl
```

4. Install PyQt5 from https://sourceforge.net/projects/pyqt/files/PyQt5/PyQt-5.5.1/PyQt5-5.5.1-gpl-Py3.4-Qt5.5.1-x32.exe/download

3.2 Apple Mac OS X

macOS	Matlab	Python
El Capitan (10.11)	2015b, 2016a	3.4.*
El Capitan (10.11), Sierra (10.12)	2016b	3.5.*

Table 2. Compatibility list for Mac OS X.

3.2.1 All systems

1. Install **Mathworks Matlab** 2015b or higher (2016 recommended)
2. Install a **Python 3** version that is compatible with your Matlab version (see table)

   Download: https://www.python.org/downloads/macos
3. Install **Git SCM**

   Download: https://git-scm.com/download/mac
4. Install required **MATLAB toolboxes**
   1. SPM12: http://www.fil.ion.ucl.ac.uk/spm
   2. jsonlab Toolbox: https://www.mathworks.com/matlabcentral/fileexchange/33381-jsonlab--a-toolbox-to-encode-decode-json-files
   3. PsychToolbox: http://psychtoolbox.org/download/

3.2.2 For Python 3.5 setups (with Python virtual environment support)

1. In Terminal go to your *workspace* directory (i.e., the directory with your neurofeedback projects) and create a Python virtual environment using

   ```
   cd ~/Documents/Work/OpenNFT
   python -m venv OpenNFT_venv
   ```

   We use *OpenNFT_venv* as convenient name for our virtual environment but you can use any name. If there are multiple python versions installed, you can use the command `python3.5` instead of `python` to ensure that version 3.5 is used when launching Python.

   Activate the new virtual environment by using the corresponding activate script

   ```
   source OpenNFT_venv/bin/activate
   ```

   The new active virtual environment is now active in the current terminal window, indicated by, e.g., `(OpenNFT_venv) MacBook:python nfbuser$`
2. Setup MATLAB Engine API for Python using macOS Terminal:

   ```
   cd /Applications/MATLAB_R2016b.app/extern/engines/python/
   python setup.py build install
   ```

   MATLAB Python engine will be installed to your currently active Python virtual environment.

3. Install required Python packages using the command line tool pip:

```
cd ~/Documents/Work/OpenNFT/OpenNFT_venv/bin/

./python -m pip install -U pip

./python pip install watchdog

./python pip install --no-deps
git+https://github.com/pyqtgraph/pyqtgraph.git

./python pip install numpy==1.11.0
```

Please note that the optimized numpy-MKL implementation would be ideal but is currently not (easily) available for macOS.

3.2.3 For Python 3.4 setups (without Python virtual environment support)

1. Setup MATLAB Engine API for Python using macOS Terminal:

```
cd /Applications/MATLAB_R2016a.app/extern/engines/python/
python setup.py install
```

MATLAB Python engine will be installed to your currently active Python environment (e.g., `/Library/Frameworks/Python.framework/Versions/3.4/lib/python3.4/site-packages/`). In case there are multiple versions installed, ensure that the correct MATLAB and Python versions are installed.

2. Install required Python packages using the command line tool pip:

```
python -m pip install -U pip
pip install watchdog
pip install --no-deps git+https://github.com/pyqtgraph/pyqtgraph.git
pin install numpy==1.11.0
```

Please note that the optimized numpy-MKL implementation would be ideal but is currently not (easily) available for macOS.

3. Install Qt5 using the package manager brew (https://brew.sh/):

```
brew install qt5
```

4. Download PyQt5 from https://pypi.python.org/pypi/pyqt5-macos-built, extract the archive and install the package using Terminal:

```
cd ~/Downloads/dist/pyqt5-macos-built-5.5.0
python setup.py install
```

4 Launching OpenNFT

You can launch OpenNFT GUI in different ways: using the command line, using a desktop shortcut to OpenNFT.py with a proper Python version set as default program, or using a software development environment such as PyCharm.

4.1.1 Using the command line application on Windows

1. Open cmd
2. Activate your **OpenNFT_venv** virtual environment by using the corresponding activate script

   ```
   cd c:\OpenNFT

   OpenNFT_venv\Scripts\activate.bat
   ```

3. Start OpenNFT

   ```
   python OpenNFT.py
   ```

4. Use `OpenNFT_venv\Scripts\deactivate.bat` to deactivate your active virtual environment at the end of the session.

4.1.2 Using the macOS terminal application

1. In Terminal go to your *workspace* directory (i.e., the directory with your neurofeedback projects) and activate the Python virtual environment by using the corresponding activate script

   ```
   cd ~/Documents/Work/OpenNFT

   source OpenNFT_venv/bin/activate
   ```

2. Start OpenNFT

   ```
   python OpenNFT.py
   ```

5. Use `OpenNFT_venv/bin/deactivate` to deactivate your active virtual environment at the end of the session.

5 Testing OpenNFT

There are two ways to test the OpenNFT functionality without concurrent real-time data acquisition. In offline mode, data is read from a folder that already contains the whole data set from a previous measurement. In online mode, the files are expected to arrive sequentially in the Watch Folder, as it happens during scanning.

5.1 Online OpenNFT mode using simulated fMRI data export

1. Open `testRTexp.py` and set the source and destination data folders to perform the delayed copying of the fMRI data files. The simulated TR can be specified by the `sleep()` function in seconds.
2. Launch OpenNFT, review the parameters and, in particular, uncheck the *Offline mode* checkbox before pressing Setup. Start the framework. It is now waiting for the first data file in the specified destination folder, i.e., the *MRI Watch Folder*.
3. Launch the testRTexp.py using `python Path\To\tests\testRTexp.py`

Processing time in the right upper OpenNFT corner displays the time it takes to process each scan, which must be less than your simulated TR to satisfy the real-time property of the neurofeedback pipeline.

5.2 Offline OpenNFT mode using already acquired fMRI data

- Launch OpenNFT, review the parameters and, in particular, check the *Offline mode* checkbox before pressing Setup. Press the Start button. OpenNFT is now processing the data in the *MRI Watch folder* as fast as possible.

6 Troubleshooting

6.1 Caveats

6.1.1 Configuration files (ini files) as operating system dependent

All settings in the ini files, for example and most importantly the path definitions follow the conventions of your host operating system. E.g., use `\` as file separator in Windows and `/` in a Unix-based system such as macOS.

6.1.2 DCM-based neurofeedback is based on DCM10

The currently implemented version of DCM neurofeedback is based on Koush et al., 2013 and 2015 publications where DCM10 as implemented in SPM8 (`spm_dcm_estimate_rt()`, `spm_nlsi_GN_rt()`) was used. Other versions of DCM use different estimation methods and might fail to reproduce the same results and would require an additional testing.

6.1.3 SPM preprocessing is optimized for real-time applications

Note the differences between the real-time modifications of the SPM12 inbuilt preprocessing functions (`spm_realign_rt()`, `spm_reslice_rt()`) and their SPM12 analogues. Applied modifications have a sufficient quality level for real-time applications, but they are not necessarily matching your local SPM setup.

6.1.4 Spatial orientation of input data

Generally, you are advised to carefully check the spatial orientation of all the images provided to the software. Our software is independent of data spatial orientation, which is often a function of acquisition parameters. Unfortunately, most Phillips MR scanner setups do not provide adequate image header information when performing a real-time export. Additionally, Phillips real-time data may require a 180-degree flip to match the proper EPI template.

6.1.5 Optimize signal processing settings

The configuration of the optimal signal processing settings depends on the experimental design and acquisition parameters. E.g., see Koush et al. 2012 for the setup of a Kalman filter.

6.2 Runtime errors and troubleshooting

6.2.1 Real-time exported files not found in watch folder

Please check that the real-time export is properly set up on your scanner and the host computer that is used for OpenNFT. If files are exported correctly, review if First File Path is set to the correct destination and the MRI Watch Folder is accessible.

- Press 'Review Parameters' and check the status of *First File Path*. If you pressed the Setup button and the field is empty indicates that you might have used an invalid syntax to specify the *First File Name*. Valid formats are:
    - Explicit file names that indicate the first file of the real-time export. Examples:
        - `001_000007_000001.dcm`
        - `001_000007_000001` (file extension will be added based on the MRI Data Type)
    - Template based mechanisms to fill parts of the filename with variables that are defined in the GUI. `{variable name}` defines a variable as specified by the caption in the OpenNFT GUI (e.g., Subject ID), `{#}` refers to the iteration/volume number, and `{…:06}` can be used to set a number format (e.g, 6 digits, zero fill, i.e., 000001). Variable names are case insensitive and spaces are ignored. Examples:
        - `001_{Image Series No:06}_{#:06}.dcm`
        - `{Project Name}/{Subject ID}/{NR Run No:03}_{Image Series No:06}_{#:06}.dcm`

      This means users can easily adapt the file locations / file names to their scanner environment
- Check the status feedback:
    - *MRI Watch folder exists* indicates that the MRI watch folder was found. *MRI Watch folder does not exist* might indicate an error.
      However, this is not necessarily always the case, given that the folder will be created during image export in certain real-time setups (e.g., Philips DRIN dumper creates a run folder for each new export, e.g., c:\drin\0001, 0002, etc.)
    - *First file does not exist* indicates that OpenNFT has not located the first file of the image series during setup. This is desired in normal online mode operations, as the file export has not yet started.
      On the other hand, *First file exists* shows that the folder is not empty and might indicate that the wrong folder is used (e.g., the previous run). However, in offline mode, which can be used for offline testing, it is expected that the first file is already available.

6.2.2 Undefined function 'spm_select' for input arguments of type 'char'

Make sure that SPM is installed and MATLAB is able to locate it in your system path. To test if the correct SPM version is found use `which spm` in a MATLAB window.

6.2.3 Undefined function or variable 'bwperim'

Make sure that you have installed MATLAB's Image Processing toolbox.

6.2.4 Undefined function or variable 'zscore'

Make sure that you have installed MATLAB's Statistics and Machine Learning toolbox.

6.2.5 Error when loading DICOM files

There is a known bug in the current implementation of MATLAB's dicominfo.m. We used the following modifications to fix the problem:

Line #336

`personName = struct([]);` changed to
`personName = repmat(makePerson(pnParts),[1,numel(splitRawData)]);`

Line #353

`%personName = makePerson(pnParts);` changed to
`personName(p) = makePerson(pnParts);`

6.2.6 Error when starting Matlab processes on macOS

We observed problems when starting Matlab instances on macOS from within OpenNFT, either during startup or using the *Initialize* button. The easiest way to fix this problem is to independently start and share the required Matlab instances (main, PTB, and SPM instances) using the macOS command line:

```
/Applications/MATLAB_R2016b.app/bin/matlab -desktop -r
"matlab.engine.shareEngine('MATLAB_NFB_00001')"
```

```
/Applications/MATLAB_R2016b.app/bin/matlab -desktop -r
"matlab.engine.shareEngine('MATLAB_NFB_PTB_00001')"
```

```
/Applications/MATLAB_R2016b.app/bin/matlab -desktop -r
"matlab.engine.shareEngine('MATLAB_NFB_SPM_00001')"
```