sensormotion Documentation

Release 1.0.0

Simon Ho

Apr 20, 2018

Contents:

Python package for analyzing sensor-collected human motion data (e.g. physical activity levels, gait dynamics).

Dedicated accelerometer devices, such as those made by Actigraph, usually bundle software for the analysis of the sensor data. In my work I often collect sensor data from smartphones and have not been able to find any comparable analysis software.

This Python package allows the user to extract human motion data, such as gait/walking dynamics, directly from accelerometer signals. Additionally, the package allows for the calculation of physical activity (PA) or moderate-to-vigorous physical activity (MVPA) counts, similar to activity count data offered by companies like Actigraph.

CHAPTER 1

Installation

You can install this package using `pip`:

```
pip install sensormotion
```

CHAPTER 2

Requirements

This package has the following dependencies, most of which are just Python packages:

- **Python 3.x**
  - The easiest way to install Python is using the Anaconda distribution, as it also includes the other dependencies listed below
  - Python 2.x has not been tested, so backwards compatibility is not guaranteed
- **numpy**
  - Included with Anaconda. Otherwise, install using pip (`pip install numpy`)
- **scipy**
  - Included with Anaconda. Otherwise, install using pip (`pip install scipy`)
- **matplotlib**
  - Included with Anaconda. Otherwise, install using pip (`pip install matplotlib`)

CHAPTER 3

Usage

Here is example workflow for extracting step-based metrics from raw vertical acceleration data:

Import the package:

```
import sensormotion as sm
```

If you have a vertical acceleration signal `x`, and its corresponding time signal `t`, we can begin by filtering the signal using a low-pass filter:

```
b, a = sm.signal.build_filter(frequency=10,
                              sample_rate=100,
                              filter_type='low',
                              filter_order=4)

x_filtered = sm.signal.filter_signal(b, a, signal=x)
```

Next, we can detect the peaks (or valleys) in the filtered signal, which gives us the time and value of each detection. Optionally, we can include a plot of the signal and detected peaks/valleys:

```
peak_times, peak_values = sm.peak.find_peaks(time=t, signal=x_filtered,
                                             peak_type='valley',
                                             min_val=0.6, min_dist=30,
                                             plot=True)
```

From the detected peaks, we can then calculate step metrics like cadence and step time:

```
cadence = sm.gait.cadence(time=t, peak_times=peak_times, time_units='ms')
step_mean, step_sd, step_cov = sm.gait.step_time(peak_times=peak_times)
```

I would recommend looking over the documentation to see other functionalities of the package.

CHAPTER 4

Contribution

I work on this package in my spare time, on an "as needed" basis for my research projects. However, pull requests for bug fixes and new features are always welcome!

Please see the develop branch for the development version of the package, and check out the issues page for bug reports and feature requests.

CHAPTER 5

Getting Help

You can find the full documentation for the package here

Python's built-in help function will show documentation for any module or function: `help(sm.gait.step_time)`

You're encouraged to post questions, bug reports, or feature requests as an issue

Alternatively, ask questions on Gitter

CHAPTER 6

Index & Modules

- genindex
- modindex
- search