Pulse Wave Denoise and Heart Rate Calculation Algorithm Documentation

YHY 21/09/2020

The following function documentation is based on the Matlab code. The sampling rate adopted is around 125Hz. Please note that the actual sample rate may change.

Singular Spectrum Analysis (SSA)

SSA is a signal noise reduction algorithm that is often used. The original signal is decomposed into several components, and then the component representing the target signal is selected and recombined into the output signal.

```
function [signalFiltered] = SSA(signal,windowLen,Lc,Rc)
```

Parameters:

- **signal** - the raw signal to be processed
- **windowLen** - the sliding window length
- **Lc** - the starting number of the component to be selected
- **Rc** - the ending number of the component to be selected

Baseline Wander Removal

Polynomial fitting

Polynomial fitting is a simple and fast method for baseline wander removal.

```
function res_signal = MoveBSL(signal,xaxis)
```

Parameters:

- **signal** - the raw signal to be processed
- **xaxis** - the x axis array of the signal, usually is the time

Quadratic Variation Reduction (QVR)

QVR is a novel and strong approach to baseline wander removal for bioelectrical signals. However, it may cause the shape distortion of the signal.

```
function outdata = baseline(indata,theta)
```

Parameters:

- **indata** - the raw signal to be processed
- **theta** - the multiple parameter in the QVR algorithm, usually set from 10 to 100

Heart Rate Calculation through FFT

FFT is a general method to get HR from the pulse wave. To reinforce its accuracy, the previous HR results are used to verify the output and the weighted smoothing method is also applied.

```
function [fs, mf] = WeiVerFreqSA(signal,pri,delta,sr)
function [fs, mf] = VerFreqSA(signal,pri,delta,sr)
function [fs, mf] = FreqSA(signal)
```

Parameters:

- **signal** - the input signal segment
- **pri** - the heart rate (in bpm) of the previous second
- **delta** - the upper limit of heart rate change (in bpm) per second
- **sr** - the sample rate

Outputs:

- **fs** - the single-sided frequency spectrum
- **mf** - the major frequency

For the FFT based HR calculation without weighted smoothing, function *VerFreqSA* can be used. For the FFT based HR calculation without weighted smoothing and HR verification, function *FreqSA* is suitable.

Generally, we use the *WeiVerFreqSA* function.

Sparse Signal Reconstruction (SSR)

SSR is a algorithm that can be utilized to get HR from the sparse pulse wave signal. It is recommended to use the SSR method to obtain HR in the signal after SSA noise reduction. Not only can it obtain accurate results, but it is also affected little by the frequency domain resolution. However, the calculation speed of this algorithm is very slow, and it is difficult to calculate HR in real time on the PC.

The specific SSR method we used is **FOCUSS** (Focal undetermined system solver)

```
function [FocussResult] = OrigFocuss(y,A,iteration)
```

Parameters:

- **y** - the input signal segment
- **A** - the observation matrix (A(m,n) = exp(j*(2*pi*m*n/N)), N is the dimension of column)
- **iteration** - the number of iterations, usually set as 5

Time-domain HR Calculation

The time-domain HR calculation is implemented by simple peak tracking in the pulse wave signal. The peak is detected in a the sliding window, whose range and position is determined by pervious HR results.

```
function [HR,MeanHR,A] = HRcalFun(data)
```

Parameters:

- **data** - the input signal segment

Outputs:

- **HR** - the heart rate results array
- **MeanHR** - the average of heart rate in every second during the whole time
- **A** - the position of detected peaks

Previous Methods

The previous codes written by Luke are also used and tested, including *doubleSlopeProc*, *integralFunc*, *findRPeakFunc* and *calcBPMFunc*. The script *PreviousMethods.m* integrals them, and the heart rate results of the previous methods can be obtained.

Double Slope Processing

Traditional derivative-based peak detection algorithms are highly affected by base-line noises. In order to achieve better detection rates and improve computational efficiency, the original pulse wave signal is converted to the dual-slope signal for peak detection at a later stage.

```
function [outputData,registerWin] = doubleSlopeProc(inputData,fs,registerWin)
```

Parameters:

- **inputData** - one second pulse wave data segment
- **fs** - the sampling frequency
- **registerWin** - a memory buffer to store the last eight samples of the last one second /current one second data segment (input/output parameter)

Integration Process

For each sample in the one second data segment, sum it up with its nearest specified number (intLen – 1) of neighboring samples on the left to obtain a smoother signal for subsequent analysis.

```
function [integralData,intRegister] = integralFunc(inputData,intLen,intRegister)
```

Parameters:

- **inputData** - the input signal
- **intLen** - the number of left neighboring samples to be integrated
- **registerWin** - the last intLen - 1 samples from the previous one second/current one second data segment (input/output parameter)

Find Peak Function

Find peak points of an input signal and update the status vector to indicate state of the object.

```
function [PEAKS,statusVector] =
findRPeakFunc(inputData,fs,judgeSlopeLen,statusVector)
```

Parameters:

- **inputData** - the input signal
- **fs** - the sampling frequency
- **judgeSlopeLen** - number of neighboring sample points on the left and right to compare against when searching a local maxima
- **statusVector** - a one by three vector to store the state of the object for the most recent three seconds. A value of 0 indicates stationary status. A value of 1 indicates walking slowly. A value of 2 indicates walking quickly. A value of 3 indicates running status.

Heart Rate Calculation

Update the latest heart rate every second and store the heart rate information for the most recent five seconds.

```
function bpmVector = calcBPMFunc(PEAKS,fs,bpmVector)
```

Parameters:

- **PEAKS** - a data structure to store peak information of a signal (number of and location of signal peaks)
- **fs** - the sampling frequency
- **bpmVector** - a two by five matrix to store the heart rate information for the most recent five seconds. The first row represents the heart rate (beats per minute) for the most recent five seconds. The second row indicates how the heart rate is obtained, either through calculation (value of 1) or prediction using previous heart rate information (value of 0). The rightmost column corresponds to the latest time and the leftmost column corresponds to the earliest time.