Heart rate monitoring using PPG signals

AUTHORS:

JULIO CESAR BALLESTEROS BORRERO, 4225791
TIAMUR ALI KHAN, 4247329

SUPERVISORS:

RICHARD HENDRIKS
SEYRAN KHADEMI

BSc graduation project,
Electrical Engineering,
TU Delft

Delft, June 15, 2016

Abstract

This bachelor's thesis discusses the idea and implementation of real-time heart rate monitoring using photoplethysmography (PPG) signals. PPG signals measured from the wrist are often subjected to distortion and noise. Signal processing techniques that tackle these issues were investigated and implemented. The algorithm of choice was JOSS. JOSS consists of sparse signal reconstruction, spectral subtraction and spectral peak tracking. The combination of these techniques is what results in a robust heart rate tracking algorithm. An improvement, in terms of computing power, was made by analyzing general properties of PPG signals. The outcome is a new framework named SMART, which combines the HR tracking power of a robust HR monitoring system, like JOSS, with the speed of faster HR monitoring methods. The overall result is a fast algorithm that computes the heart rate with an average absolute error of 1.42. This result shows that PPG based heart rate monitoring on wearable devices has great potential for fitness and medical purposes.

Contents

Chapter 1

Introduction

The heart rate (HR) is the number of times your heart beats per minute. For the average person this rate is between 60 and 100 beats per minute (BPM) when in rest. Monitoring the HR can give a person insights on their medical condition as well as their fitness level. When performing physical activities, monitoring the HR is useful since it can help the user keeping track of their training load. For example, an optimal HR exists for burning fat. These are the main reasons why major smartwatch producers now have added the HR monitoring feature to their devices.

The HR of a person can be determined in several ways. Two of these are electrocardiography (ECG) and photoplethysmographic (PPG). ECG is more accurate since it is directly recorded from the chest using electrodes. However, ECG requires the wearer to place electrodes on their skin, which is very invasive for physical activities. PPG is less invasive since it only requires two light emitting diodes (LEDs), which can comfortably be placed on the wrist. Therefore, wearable devices that include HR monitoring often use a system based on PPG signals. Although comfortable, the signals received from this sensors are often distorted, noisy and not accurate enough for medical purposes or sport.

The goal of the project is to write software that can accurately monitor real-time HR with PPG sensors placed on the wrist. PPG signals are obtained by using a pulse oximeter embedded in wearable devices. A pulse oximeter illuminates the wearers skin using two LEDs and measures intensity changes in the light reflected from the skin. The two diodes emit light on two different frequencies. The amount of reflected light is a function of the amount of oxygenated blood under the skin and its frequency. By taking the ratio of the reflected power from the two frequencies we obtain what we call a PPG signal. This signal is a periodic signal, with the HR frequency as the most dominant frequency. Under normal conditions this frequency can easily be obtained by doing a frequency or time domain signal analysis. However, the challenge comes from the fact that a lot of noise and distortion will be introduced due to the wearer's movements when exercising. These distortions will be referred to as motion artifacts (MAs). MAs make identification of the HR in both time and frequency domain very challenging.

Many signal processing techniques exist that can identify MAs and remove noise from PPG signals. The techniques to be discussed in this thesis are Adaptive Noise Cancellation (ANC) [4], TROIKA [1] and JOSS [2]. From a literature study it became apparent that JOSS performs the best out of these three. The reason for this is that JOSS is an improvement of TROIKA and that ANC does not use a spectral peak tracking system. Even though JOSS can accurately track the HR, it still uses a lot of computing power which drains the battery of the device. Therefore, all three methods were implemented and studied in order to come to a more robust solution. An improved framework, in terms of computing power, named SMART (Sport Motion Artifact Removal Technique) will be presented in this thesis. This framework combines multiple signal processing techniques and exploits general properties of PPG signals. The low computational load of SMART could open up the possibility of tracking the HR throughout the day for medical purposes. Since our project group consists of six persons, the work has been divided in the following subgroups:

- Subgroup A1: Tiamur Khan and Julio Cesar Ballesteros Borrero
- Subgroup A2: Ziar Khalik and Ahmet Gerçekcioğlu
- Subgroup A3: Bas Generowicz and Xenia Wesdijk

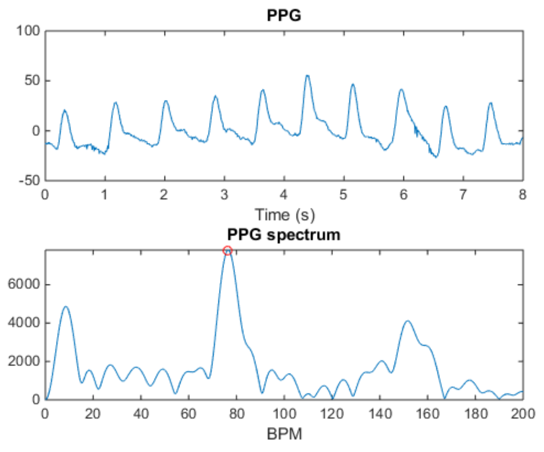
Our subgroup worked mainly on JOSS and SMART, for this reason these techniques will be elaborated in more detail than ANC and TROIKA.

In Chapter 2 the problem regarding PPG signals is explained in more detail. Chapter 3 establishes the requirements of our software. It is then followed up by Chapter 4, in which a general description of the already existing HR tracking techniques is given. Then in Chapter 5, sparse signal reconstruction (SSR), a key component of both TROIKA and JOSS, will be elaborated. In Chapter 6, information regarding spectral substraction, an important component of JOSS, is explained. Another key component, spectral peak tracking (SPT), is explained in Chapter 7. Chapter 8 discusses the research that has been done about tracking the HR with an ECG signal and quantifying the quality of a PPG signal. In Chapter 9 the general idea of SMART and our implementation of it is explained. Chapter 10 then presents the results of our project group. Chapter 11 and 12 are dedicated to the discussion and conclusion.

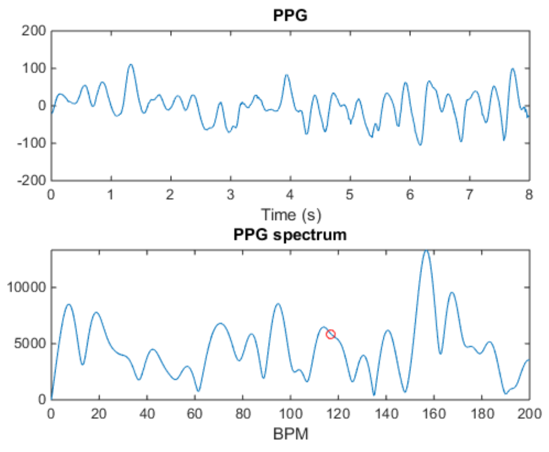
Chapter 2

Problem statement

PPG signals read from the wrist can be heavily distorted when the wearer is in movement. In Figure 2.1a, an 8 second recording of a PPG signal and its digital fourier transform (periodogram) are shown. The true HR is indicated by the circle. It is easy to determine the BPM by looking for the most dominant peak in the frequency spectrum. However, when the PPG signal is distorted due to noise and motion artifacts, this technique is not sufficient. An illustration is given in Figure 2.1b. Many frequencies are present in the periodogram which results in the HR becoming indistinguishable from noise and MAs. Taking the most dominant peak in the periodogram would result into a huge error. Accurately estimating the HR is the major challenge of the project and is also why alternative techniques, other than a simple periodogram, must be explored to tackle this problem.



(a) A PPG signal of good quality and its periodogram, respectively. The red circle indicates the HR, it is found at the most dominant peak.



(b) A PPG signal of bad quality and its periodogram, respectively. The red circle indicates the HR, in this case the HR is difficult to determine.

Figure 2.1: PPG signal analysis

To test and develop our software, twelve datasets were made available [1]. Each dataset includes simultaneously recorded ECG, 2 PPG and 3 acceleration signals (the x, y and z-components). These signals are sampled at a frequency of 125 Hz. The PPG and acceleration signals were recorded from the wearer's wrist and the ECG was recorded from the chest. The individuals ran on a thread mill with varying speeds. The full recording of each dataset lasted around 5 minutes. To emulate real-time heart rate monitoring we read the signal in time windows of T = 8 seconds long with incremental steps of S = 2 seconds. This means that each window overlaps 6 seconds with the next window. Plotting the estimated HR in BPM for all windows is what will be referred to as a "BPM trace". The BPM trace has values for every 2 seconds till the end of the recording session. Also at our disposal was a BPM trace given to us that was determined with the ECG signal. This BPM trace will be referred to as the true BPM trace and will be used as reference. Our goal is to implement an algorithm that estimates the HR for each time window using PPG and acceleration signals, and compare its results to the true BPM trace. An example of a true BPM trace is given in Figure 2.2.

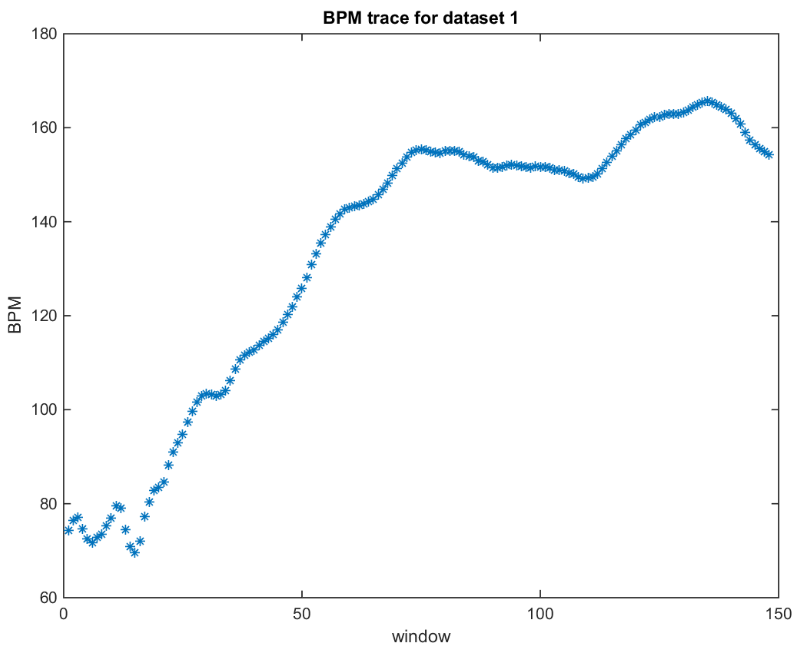


Figure 2.2: The true BPM trace of dataset 1. The HR is increasing since the individual is running on a thread mill. ECG is used as a reference to compare HRs calculated by PPG.

Chapter 3

Programme of requirements

Heart rate (HR) monitoring on wearable devices is typically done by using photoplethysmographic (PPG) signals measured from the wrist. However, PPG signals measured from the wrist are often distorted due to the movement of the hand. This makes PPG based HR monitoring challenging. This challenge motivates the development of an algorithm that can handle such distortions and can robustly estimate the HR. The full system was developed by the project group. The system is aimed towards manufacturers of wearable devices that include HR monitoring as a feature for their devices.

The requirements that are set for the system are based on the given datasets. These datasets consist of different individuals running on a thread mill for approximately 5 minutes. We are given 12 datasets in total.

3.1 Functional requirements

[1.1] The system must output the HR in BPM (Beats Per Minute).
[1.2] The system must be implementable on wearable devices that are used for HR monitoring.
[1.3] The system must give accurate HR readings during rest and exercising.

3.2 System requirements

[2.1] The average absolute error on each dataset must be below 10 BPM.
[2.2] The average absolute error over all data sets must be below 5 BPM.
[2.3] The system must be able to calculate the heart rate within 2 seconds on a wearable device.

3.3 Development of manufacturing methodologies

[3.1] The algorithm is developed in MATLAB, which has a rich set of toolboxes and functions that allow for easy implementation of the system.

3.4 Business strategies, marketing and sales opportunities

[4.1] The system will be marketed in a business to business model to smart-watch manufacturers.
[4.2] Licensing will be used to generate revenue.

Chapter 4

Signal processing techniques

In order to meet the system requirements set by Chapter 3, we investigated signal processing techniques and frameworks that deal with HR monitoring. An initial investigation made clear that for every HR monitoring system, two functions are critical for robust performance: noise cancellation (NC) and spectral peak tracking (SPT). NC is used to remove noise and MAs from the measured PPG signal. Furthermore, NC is well researched in the field of signal processing. The basic concept of NC for HR monitoring is illustrated in Figure 4.1.

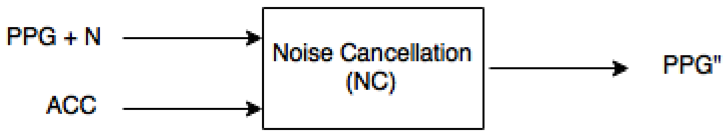


Figure 4.1: The basic concept of NC is depicted. Here noise (N) is removed from the measured signal (PPG+N) by making use of acceleration data (ACC). The output is a processed PPG signal that is free of noise and MA (PPG”).

Herein the acceleration data, measured from an accelerometer, is used to identify MAs in the measured PPG signal. MAs are then removed as to obtain an estimation of the original PPG signal. NC can be done by methods like adaptive noise cancellation (ANC) [4], independent singular spectrum analysis (SSA) [1], empirical mode decomposition (EMD) [11] and spectral subtraction (SS) [2]. The other key function, SPT, can also be done in multiple ways. Nevertheless, only our implementation of SPT will be discussed. Reason for this being that the core principle of all SPT algorithms are the same. In the following sections an introduction will be given for the different HR monitoring methods that our project group has investigated and implemented. The contributions of each subgroup will be made clear.

4.1 Adaptive Noise Cancellation (ANC)

ANC is a form of noice cancellation. Unlike fixed filters, the parameters of adaptive filters are adaptive. This means that they change according to the characteristics of the system. This is advantageous as no prior knowledge of the signal or noise is needed for this method to work. Two types of adaptive filters were investigated by subgroup A3. These are the Least Mean Squares filter (LMS) [7] and the Recursive Least Squares (RLS) [8] filter.

The basic functionality of an adaptive filter can be explained by the block-diagram found in Figure 4.2. Two signal sources are used. One measures the signal of interests and the other measures noise only. The signal source produces a signal d_k. This signal contains the desired signal and noise at point k, given as s_k and n_k, respectively in Equation 4.1.

$$d_k = s_k + n_k \tag{4.1}$$

The noise source produces a signal u_k that contains information about $\hat{n}_k$. Therefore, u_k is used to estimate $\hat{n}_k$. After, an estimation of the desired signal is done by subtracting $\hat{n}_k$ from d_k, as is seen in Equation 4.2.

$$\hat{s}_k = d_k - \hat{n}_k \tag{4.2}$$

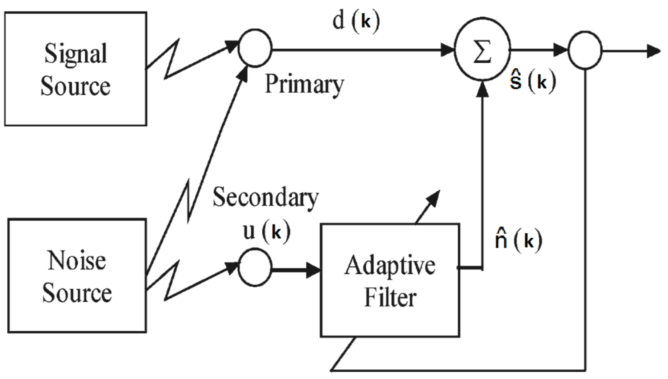


Figure 4.2: ANC Diagram [5]

4.2 TROIKA

TROIKA [1] is an HR monitoring framework that promises robust performance. For this reason TROIKA was the framework our project group focused primarily on. The block diagram of TROIKA can be found in figure 4.3. Here, different functionalities have been divided into different blocks. The most crucial components of this framework are signal decomposiTion, sparse signal RecOnstructIon and spectral peaK trAcking.

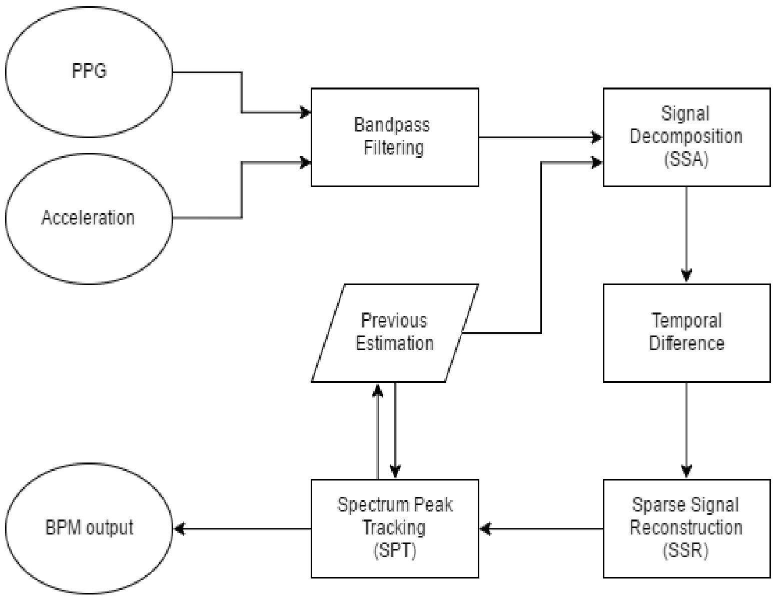

PPG
Acceleration
Bandpass Filtering
Signal Decomposition (SSA)
Previous Estimation
Temporal Difference
BPM output
Spectrum Peak Tracking (SPT)
Sparse Signal Reconstruction (SSR)

Figure 4.3: The framework of TROIKA depicted in a flowchart [1]. SSA, SSR and SPT are the key components of this framework.

Since HR can only have frequencies in a certain frequency region, a band pass filter is used to remove noise outside this region. In the next step a signal decomposition algorithm is used to remove noise inside the frequency region of interest. To this end, algorithms like SSA and EMD can be used. Our project group chose to implement SSA. After SSA has been applied, temporal difference is used to remove the aperiodic components of the PPG signal. After these three steps, SSR is performed in order to create a sparse spectrum. Algorithms like FOCUSS [3] and sparse Bayesian learning [10] can be used to achieve this. Finally, spectral peak tracking (SPT) uses previous HR estimations to estimate the current HR. The work of implementing this algorithm has been divided into two subgroups. Group A2 focused on implementing TROIKA. Our subgroup's contribution was implementing the SSR and the SPT algorithms.

4.3 JOSS

JOSS (JOint Sparse Spectrum reconstruction) is a framework built on top of TROIKA. It makes use of SSR and SPT, just like TROIKA, but it differs in the technique used for MA removal. TROIKA uses signal decomposition techniques to remove MA while JOSS uses spectral subtraction. The block diagram of JOSS can be found in figure 4.4. First an accurate sparse spectrum of the PPG signal and acceleration components are constructed using FOCUSS [3]. Then the spectra are band-pass filtered to remove unrealistic heart rates. The acceleration spectrum is then subtracted from the PPG spectrum. This removes peaks corresponding to MA from the PPG spectrum. The result of this is a cleansed spectrum used by SPT to produce a BPM output. This output is then used as feedback for SPT to search in smaller intervals. JOSS is the framework our subgroup focused primarily on. The sub-components of this framework (SSR, spectral subtraction, SPT) are discussed in more detail in the subsequent chapters.

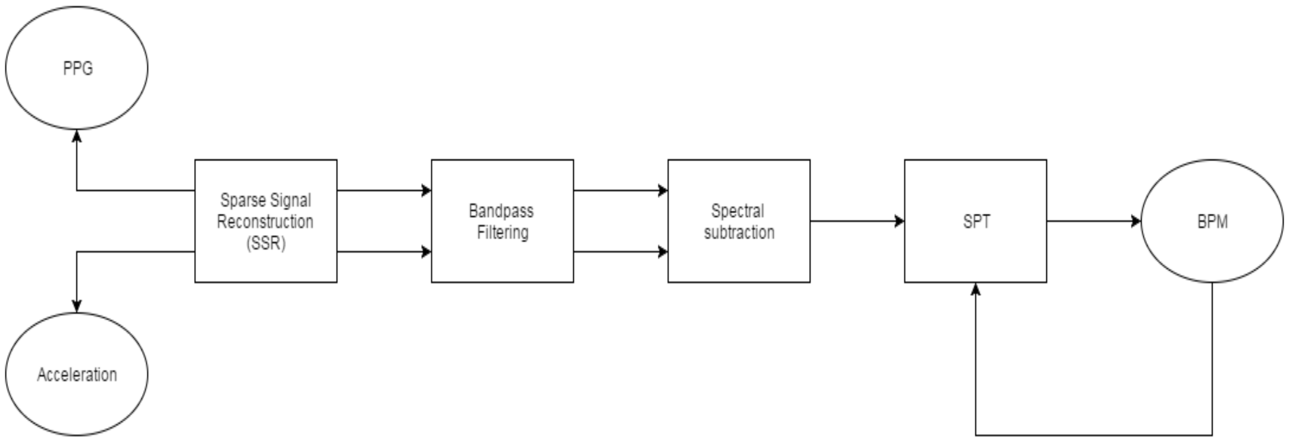


Figure 4.4: Block diagram of JOSS

Chapter 5

Sparse signal reconstruction

Sparse signal reconstruction (SSR) is the signal processing technique used by TROIKA and JOSS to estimate high resolution spectra of sparse signals (signals that contain many zero or near zero-elements). PPG signals are generally sparse. Only at the heart pulse or because of distortions do they spike. SSR then exploits the sparsity of PPG signals in order to construct a sparse spectrum. Using SSR has advantages over conventional spectrum estimation methods since it makes peaks well defined. The basic model for SSR is given by equation (5.1).

$$\mathbf{y} = \mathbf{\Phi x} \tag{5.1}$$

Here $\mathbf{y}$ is the observed PPG time-signal of size M $\times$ 1, $\mathbf{\Phi}$ the unitary Fourier transform matrix of size M $\times$ N (N $>$ M) and $\mathbf{x}$ the sparse frequency vector of size N $\times$ 1. The matrix $\mathbf{\Phi}$ is constructed by setting element $\Phi_{m,n}$ to $e^{j\frac{2\pi}{N}mn}$. $\mathbf{x}$ contains the coefficients of each complex exponential present in the observed spectrum. Matrix multiplication of $\mathbf{\Phi}$ and $\mathbf{x}$ results in a scaled inverse Fourier transform of $\mathbf{x}$.

Since $\mathbf{\Phi}$ is N $>$ M, there are multiple solutions for $\mathbf{x}$ that satisfy 5.1. The goal is to find the sparsest solution that approximates the PPG signal as best as possible. This goal can be expressed by the optimization problem given in Equation (5.2).

$$\hat{\mathbf{x}} \leftarrow \min_{\mathbf{x}} \|\mathbf{y} - \mathbf{\Phi x}\|_2^2 \tag{5.2}$$

An algorithm that solves the optimization problem of Equation (5.2) is FOCUSS [3]. The solution of equation (5.2) leads to a sparse frequency vector $\hat{\mathbf{x}}$. The k-th element of its spectrum $\mathbf{s}$ is given by s_k = $|\hat{x_k}|^2$.

As stated before, SSR is more advantageous over traditional spectrum estimation algorithms. However, SSR assumes that $\mathbf{y}$ contains many zero or near zero-valued elements. Otherwise the performance of the algorithm will be poor [1].

5.1 FOCUSS

FOCUSS is an algorithm that solves (5.2) iteratively. It encourages sparsity of the solution by means of weights. Algorithm 1 describes the basic FOCUSS algorithm [3].

Algorithm 1 Basic FOCUSS algorithm

```
Initialize x
for each iteration do
    $W_{pk} = \text{diag}\ (x_{k-1})$
    $q_k = (\mathbf{\Phi} W_{pk})^{+}\text{y}$
    $x_k = W_{pk} q_k$
end for
```

Here k denotes the k-th iteration, x the sparse solution vector, diag the diagonalization operation, $\mathbf{\Phi}$ the unitary Fourier transform matrix of size M $\times$ N, $(\mathbf{\Phi} W_{pk})^{+}$ the Moore-Penrose inverse (pseudoinverse) of matrix $\mathbf{\Phi} W_{pk}$ and y the PPG time-signal. The Moore-Penrose inverse of a matrix A is defined as $A^{+} = A^{H}(AA^{H})^{-1}$. Here A^H is the Hermitian transpose of A.

There are three parameters that influence the performance of FOCUSS: the number of iterations, the initialization of $\mathbf{x}$ and the resolution of the spectrum N. For the number of iterations we chose 2, as this performs sufficiently for the observed data. More iterations of FOCUSS results into a much larger computational load as this algorithm deals with many-valued matrix operations. For the initialization of $\mathbf{x}$ any initialization can be chosen for FOCUSS to converge to an optimum, as is discussed in [3]. We set $\mathbf{x}$ as a vector of ones. This is done to reduce the computational load of the first iteration. The diagonalization of a vector of ones is simply the identity matrix. The identity matrix multiplied $\mathbf{\Phi}$ is simply $\mathbf{\Phi}$. Thus, the computational load is reduced since this matrix multiplication does not have to be performed. The last parameter, N, was chosen as $60 \cdot \mathrm{Fs}$ to obtain a spectrum that has an accuracy of 1 BPM.

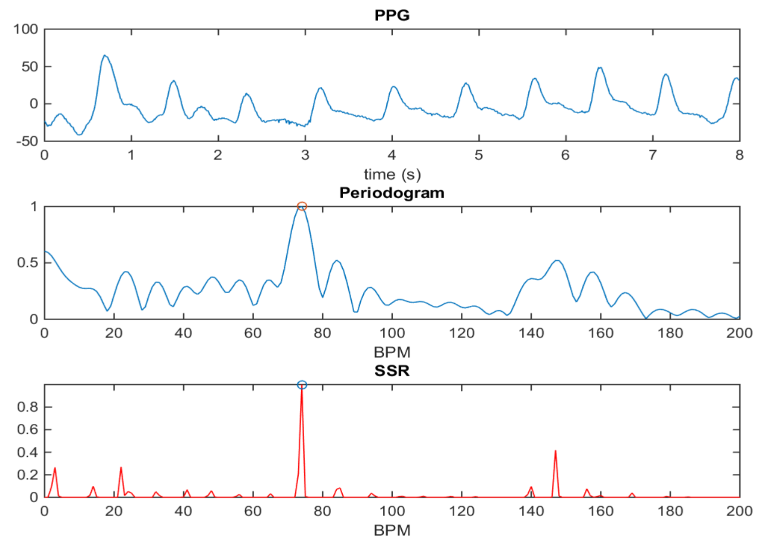
5.2 Decimation

When performing FOCUSS, the PPG signal is read in time windows of T = 8 seconds. Sampled at a sampling frequency Fs = 125 Hz, this corresponds to M = 1000 samples. To maintain an accuracy of 1 BPM in the spectrum, we set N = $60 \cdot$ Fs = 7500. This results in a $\mathbf{\Phi}$ matrix of size 1000×7500. Matrix operations on matrices of these sizes are computationally heavy. To significantly reduce the computational load of FOCUSS, decimating the signal was considered. This reduces the size of $\mathbf{\Phi}$ greatly.

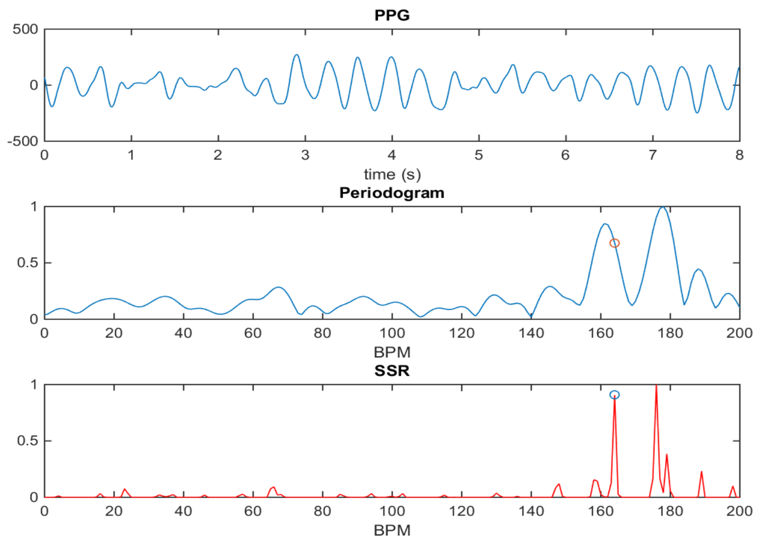
Since 200 BPM (3.33 Hz) is the highest heart rate we read out, a theoretical sampling rate of 6.67 Hz is feasible. To evaluate the algorithm we are comparing our calculated output to a ground-truth BPM trace that has true BPM values for every 2 seconds. This corresponds to $2 \cdot$ Fs samples. To make sure we take every sample, we chose Fs such that the product $2 \cdot$ Fs is an integer. Which means that Fs was chosen as a multiple of 0.5. For these reasons 10 was chosen as the decimation factor. This results in a sampling rate of 12.5 Hz. This reduces the size of $\mathbf{\Phi}$ to 100×750 (100 times smaller!) and has minimal effect on the estimation of the spectrum.

5.3 Performance

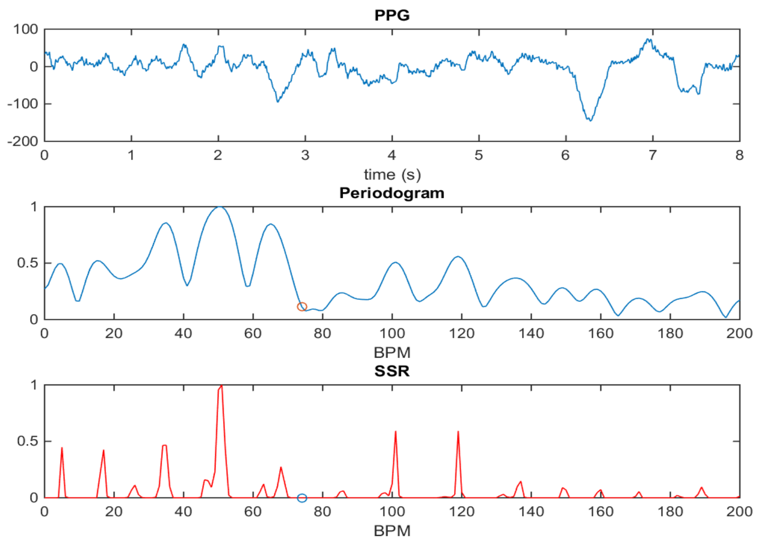
To evaluate the performance of SSR, the periodogram and the sparse spectrum of multiple PPG signals were analyzed. In Figure 5.1a a good PPG signal, its periodogram and its sparse spectrum are shown, respectively. The number of iterations in FOCUSS was set to 5 to better illustrate the principle of SSR. The Figure shows that SSR sparsifies the spectrum, allowing for more dense peaks. The circle indicates the true BPM value. For this PPG signal both spectra show a peak at the HR. In Figure 5.1b a PPG signal is shown that is a lot more noisy. Here periodogram does not show a peak at the true BPM. Only the sparse spectrum is able to locate the true BPM. Thus, for more noisy signals SSR becomes valuable. It can locate the HR peak often when periodogram cannot. However, this is not always the case. When the signal is really poor, SSR fails to create peaks at or near the true BPM, as can be seen from Figure 5.1c. This is where peak tracking comes into play.



(a) Good PPG time signal, its periodogram and its sparse spectrum respectively. The circle indicates the true BPM value.



(b) Noisy PPG time signal, its periodogram and sparse spectrum respectively. The circle indicates the true BPM value. SSR is able to locate the HR peak while periodogram cannot.



(c) Noisy PPG time signal, its periodogram and sparse spectrum respectively. The circle indicates the true BPM value. From the plots it can be seen that neither periodogram nor SSR can locate the HR peak.

Figure 5.1: Performance of SSR

Chapter 6

Spectral subtraction

Spectral subtraction is the signal processing technique that was used by JOSS to remove MAs in the PPG signal. If these MA peaks remain present, spectral peak tracking might identify these peaks as the HR peak, resulting in false readings of HR. Before this technique can be applied, the sparse spectra of the PPG and acceleration data were constructed. These were band-pass filtered to eliminate impossible HR frequencies and then normalized to distribute the energy in the spectra equivalently. Then, spectral subtraction was performed. Algorithm 2 describes this process.

Algorithm 2 Spectral subtraction

```
for each frequency bin $n_i(i = 1, \dots ,N)$ do
    set $C(i) = \max(SSa_x, SSa_y, SSa_z)$
    $SSPPG(i) = SSPPG(i) - C(i)$
end for
```

Here a frequency bin (frequency index) denotes an index in the vector of a spectrum. The conversion of a frequency bin N_f in a spectrum of length N to a physical frequency f is given by Equation (6.1). The physical frequency multiplied by 60 results in the BPM value of bin N_f. SSa_x, SSa_y and SSa_z here denote the sparse spectra of the acceleration data of the x, y and z components, respectively. SSPPG is the sparse spectrum of the PPG signal.

$$f = \mathrm{Fs} \cdot \frac{N_f - 1}{N} \tag{6.1}$$

The algorithm subtracts the maximum of the acceleration spectra from the PPG spectrum for each frequency bin. This results in a PPG spectrum that is cleansed from peaks originating from MAs. This process is visually illustrated in Figure 6.1. The first plot shows the sparse spectrum of a PPG signal. This spectrum has 3 dominant peaks: at 75 BPM, 105 BPM and 150 BPM. The second plot shows the maximum of the sparse spectra of the acceleration data. This spectrum also has peaks at 75 BPM and 150 BPM. This would indicate that the peaks at 75 and 150 BPM in the PPG spectrum are caused by motion artifacts. By subtracting the acceleration spectrum from the PPG spectrum the spectrum in the third plot is created. It contains a clean spectrum with the only dominant peak being the true heart rate. As a final modification to the resulting spectrum, all values of the spectrum lower than $\frac{1}{5}$-th of the maximum are set to 0. The overall result of these signal processing techniques is a clean spectrum that only contains peaks at relevant frequencies.

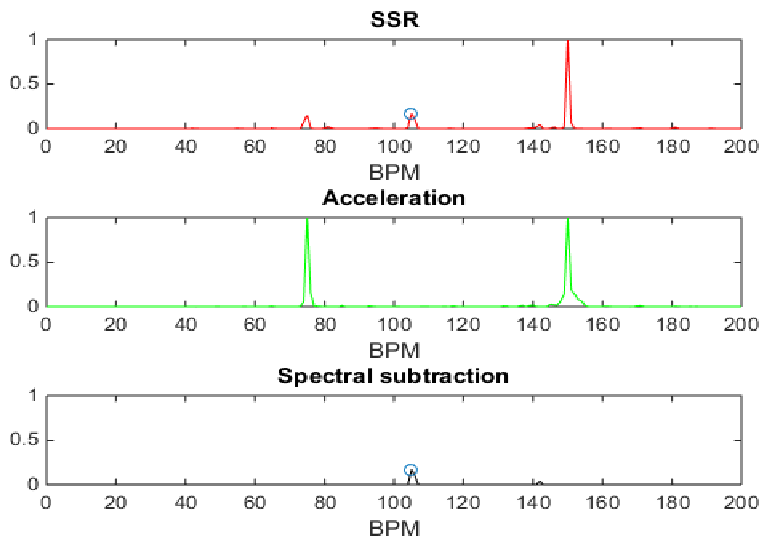


Figure 6.1: Sparse spectrum of a PPG signal, sparse spectrum of acceleration data and the subtraction of the two, respectively. MAs are removed by subtracting the acceleration spectrum from the PPG spectrum. The circle indicates the true BPM value.

Chapter 7

Spectral peak tracking

Once SSR is applied and MAs are removed, the final step in the algorithm is to determine the HR. The algorithm that does this is Spectral Peak Tracking (SPT). SPT has as input a cleansed spectrum and gives the HR as output. It uses the previously estimated HR as feedback to narrow its search window for the current HR. SPT becomes valuable when HR peaks are difficult to identify, as was the case in Figure 5.1c, found in Chapter 5. SPT has two main functions:

- Identify and track the HR peak as effectively as possible.
- Deal with cases when there are no peaks present at or near the previously calculated HR.

SPT can be divided into four stages: initialization, peak detection, verification and discovery. This chapter discusses the implementation of SPT.

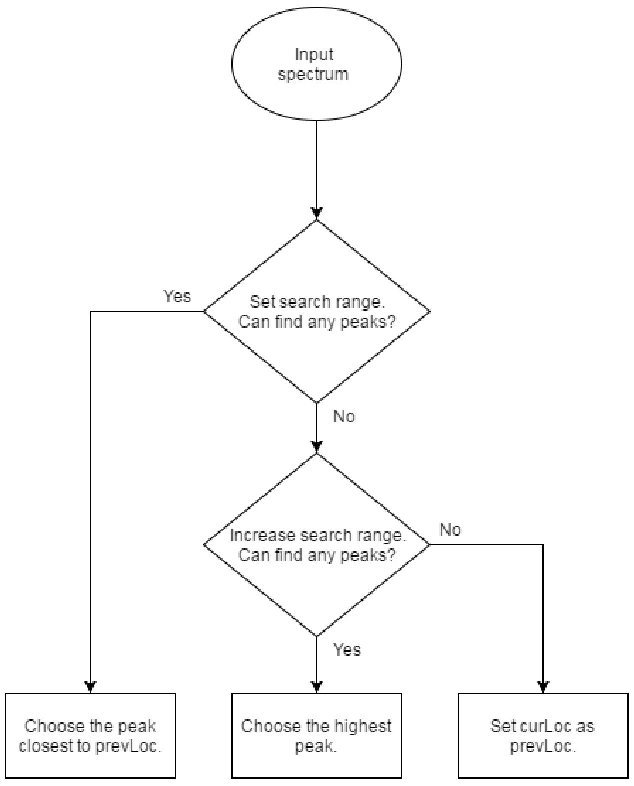
7.1 Implementation

7.1.1 Initialization

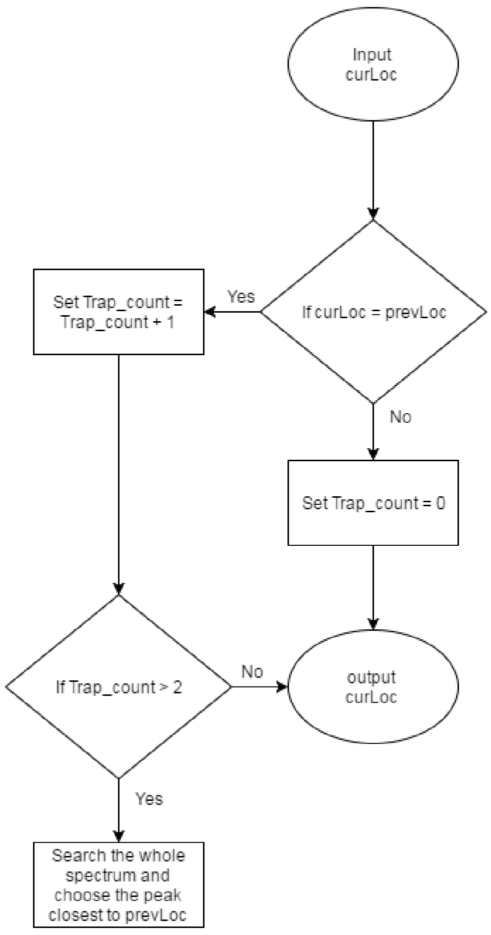
A correct initialization is needed to effectively track the HR. The approach for this is to search for the most dominant peak in the spectrum when the quality of the PPG signal is sufficient. Chapter 8 discusses this criterion in more detail. Once the PPG quality is sufficient, an initialization can be chosen for SPT and the algorithm enters the next stage.

7.1.2 Peak detection

In the peak detection stage, peaks near the previously calculated HR are explored. The previous HR corresponds to a frequency bin (frequency index) B_{prev}. To find the current HR peak, a range R_1, defined as $R_1 = [B_{prev} - \delta_1, B_{prev} + \delta_1]$ is set to search in. Here δ_1 is a positive integer and denotes the maximum deviation from B_{prev} in 1 time window. If there are any peaks present in range R_1, the frequency bin corresponding to the closest peak to B_{prev} is chosen as the current HR bin. If there are no peaks present in range R_1, the search range is increased. R_2, defined as $R_2 = [B_{prev} - \delta_2, B_{prev} + \delta_2]$ with $\delta_2 > \delta_1$ is then used to search for peaks. If any peaks are found, the maximum peak is taken. If there are no peaks found, the previous HR bin is chosen as the current HR bin. Figure 7.1a shows this process.



(a) Flowchart of the peak detection stage


Input
curLoc
If curLoc = prevLoc
Yes
Set Trap_count = Trap_count + 1
No
Set Trap_count = 0
If Trap_count > 2
No
output
curLoc
Yes
Search the whole spectrum and choose the peak closest to prevLoc

(b) Flowchart of the verification stage

Figure 7.1: Spectral peak tracking displayed by flowcharts

7.1.3 Verification and discovery

From the peak detection stage we now enter the verification stage. The verification stage counts the number of times the previous HR was chosen as the current HR. If this occurs 2 times in succession, it is likely that the HR peak is outside the search range. In order to discover the HR peak, the full spectrum is searched and the peak closest to the previously calculated HR is chosen. Figure 7.1a shows the flowchart of this process. In Figure 7.2, a BPM trace is given where the HR peak was lost and then rediscovered using peak discovery.

7.1.4 Choice of parameters

SPT has two parameters that can be chosen: δ_1 and δ_2. δ_1 denotes the maximum deviation of HR in 1 time window. This parameter was set to 15 as a change of HR bigger than 15 BPM in 2 seconds is very unlikely to happen. δ_2 is set to 25 to catch cases when there are no peaks in R_1 due to poor signal quality or the wrong peak was chosen in prior windows. Changing these parameters had very little to no effect on the final results of the algorithm.

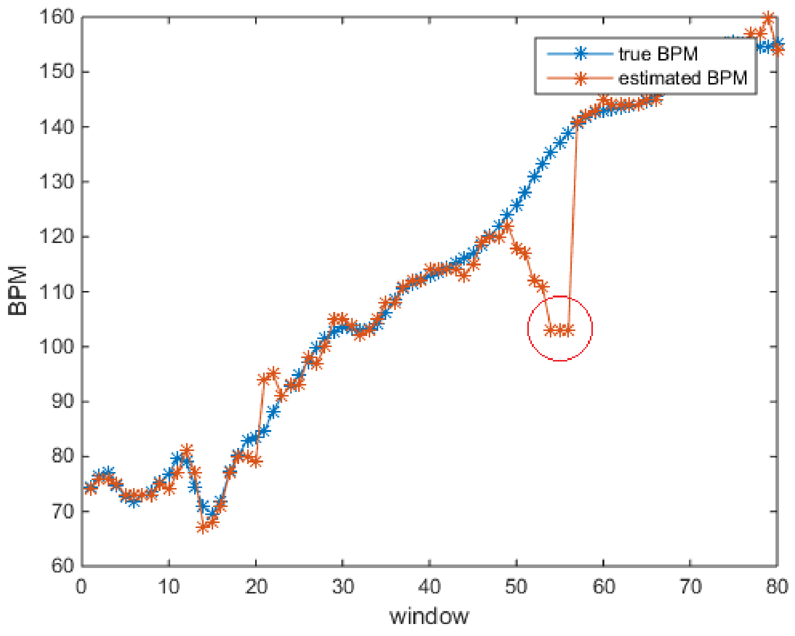


Figure 7.2: No peaks are found for 2 successive windows. Peak discovery rediscovers the correct HR peak.

Chapter 8

Data Analysis

One of our initial goals was to accurately estimate the HR using an ECG signal. This was done in order to gain ideas on how to estimate the HR using PPG signal. Although a correct BPM trace was obtained with an ECG signal, the same method could not be used with a PPG signal. The explanation of this finding is elaborated in Section 8.1. Other research, was to try to identify when the PPG signals are good enough to be analyzed with a periodogram, or with a periodogram using the ANC filter. This research was needed to implement SMART. This research can be found in Section 8.2.

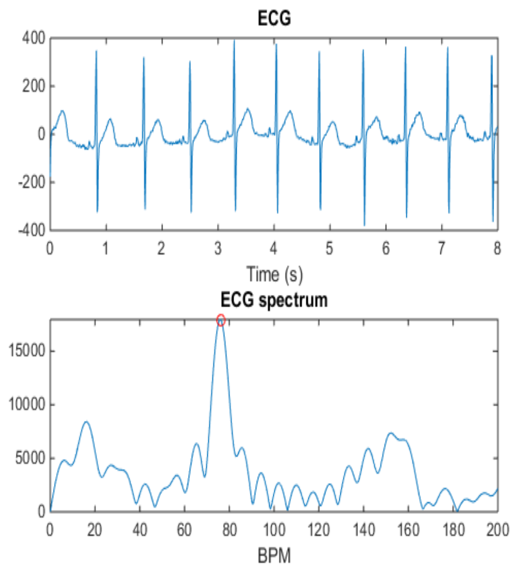
8.1 ECG signal

In general, obtaining the HR form an ECG signal is easier than from a PPG signal. Mainly because it experiences less or no distortion from MAs. An example of a good quality ECG signal is depicted in Figure 8.1a. In this case, determining the HR is easy, both in time and in frequency domain. Nevertheless, it is not always possible to determine the correct HR that easily. An example of this is shown in Figure 8.1b.

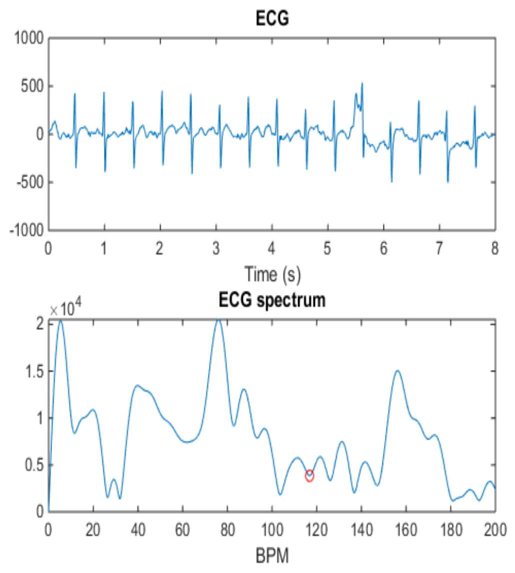
To solve this, the ECG signal is processed in the time domain. The first step of this process is to take the temporal difference of the signal. The process can be seen in Figure 8.2. Since the maximum change of an ECG is in the negative slope of the HR pulse, the result will have maximum deviation in the negative direction. Hence why the result is reversed and everything below a value of 100 is put to zero. The result will be a collection of pulses that repeat with the frequency of the heart beat. It is worth noting that in the last step the DC component is removed. With this processed signal, the correct HR can be read from the frequency spectrum. The improvement can clearly be seen from Figure 8.4a. With this technique the BPM average error was calculated with Equation 10.1. The results for all datasets and the total average are available in Table 8.1. From these results, it is concluded that this method of reading the HR using an ECG is working well. Now that a solution has been found for reading HR values with ECG signals, the question then becomes whether it can also be used for reading HR values with PPG signals. The same idea was applied to PPG signals and as a result the BPM estimations became worse, Figure 8.4b is an obvious example. The average BPM error over the twelve datasets without processing the PPG signal was 19 BPM, but increased to 33 BPM when processed. It is suspected that this technique does not work on PPG signals because the HR pulses are ill-defined, relative to ECG signals. The result of applying this process to a PPG signal can be seen in Figure 8.3, which illustrates the problem.

Table 8.1: Average absolute errors on the 12 datasets using a processed ECG signal. In the final column the final value is available.

Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9	Set 10	Set 11	Set 12	**Average**
0.32	0.67	0.32	0.33	0.29	0.52	0.26	0.26	0.22	0.24	0.36	0.30	**0.34**



(a) An ECG signal of good quality and its periodogram.



(b) An ECG signal of bad quality signal and its periodogram.

Figure 8.1: An ECG signal of good quality and bad quality, with their spectra, are shown. The red circle corresponds to the HR. The HR can be easily determined for the signal in 8.1a and for the signal in 8.1b it cannot.

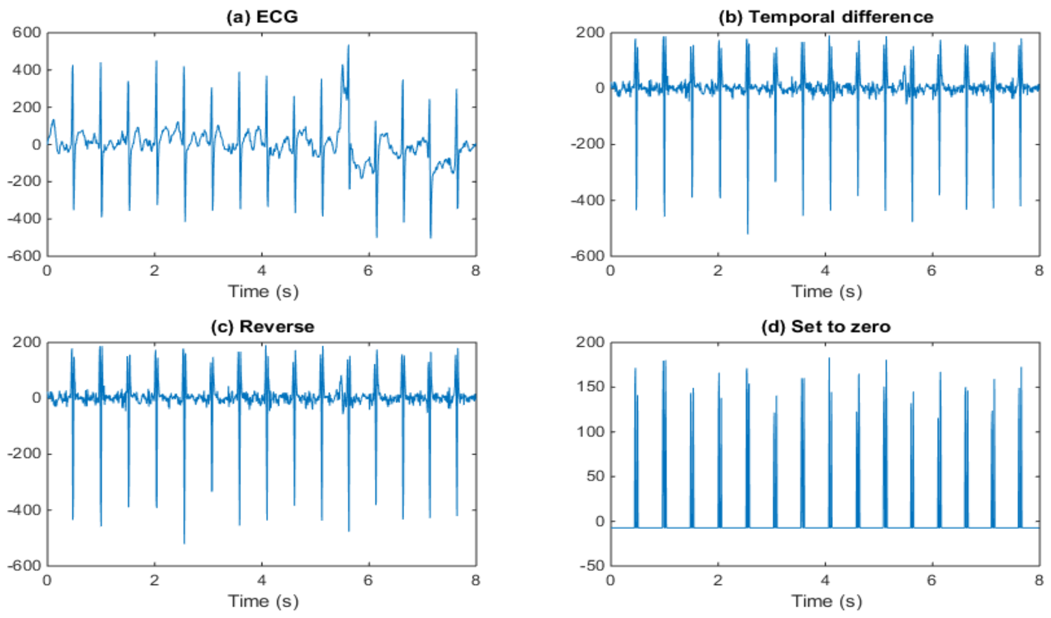
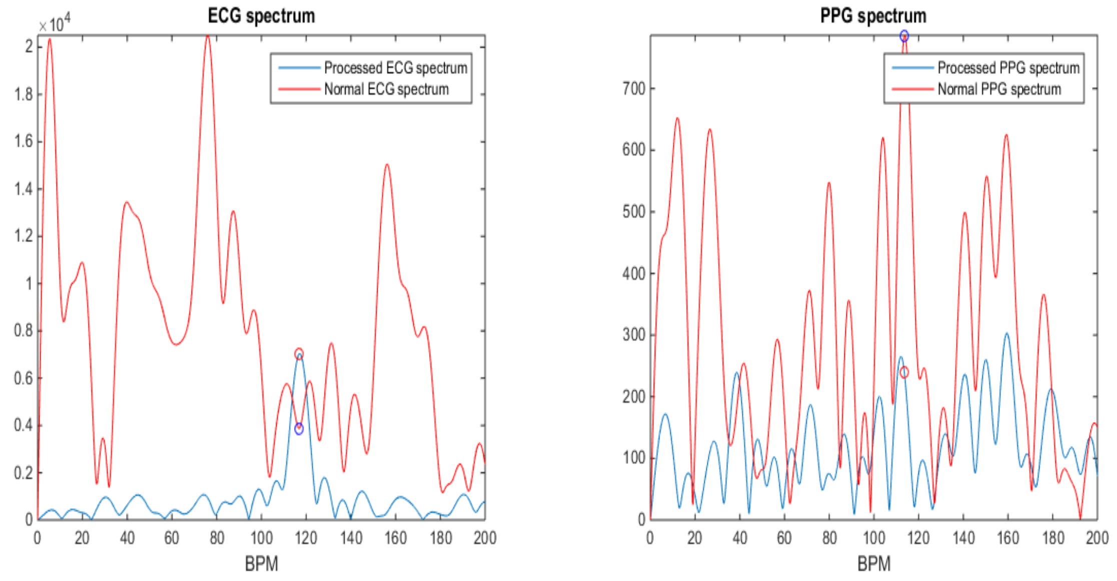


Figure 8.2: The processing of the ECG signal is depicted. Each step is marked subsequently from a to d. At the end, a time signal is obtained from which the HR can easily be calculated in frequency spectrum.



(a) Spectra of an processed and unprocessed ECG signal.

(b) Spectra of an processed and unprocessed PPG signal.

Figure 8.4: An illustration of the influence of processing the signals. In 8.4a the ECG spectrum is enhanced, but in 8.4b the spectrum of the PPG has worsened.

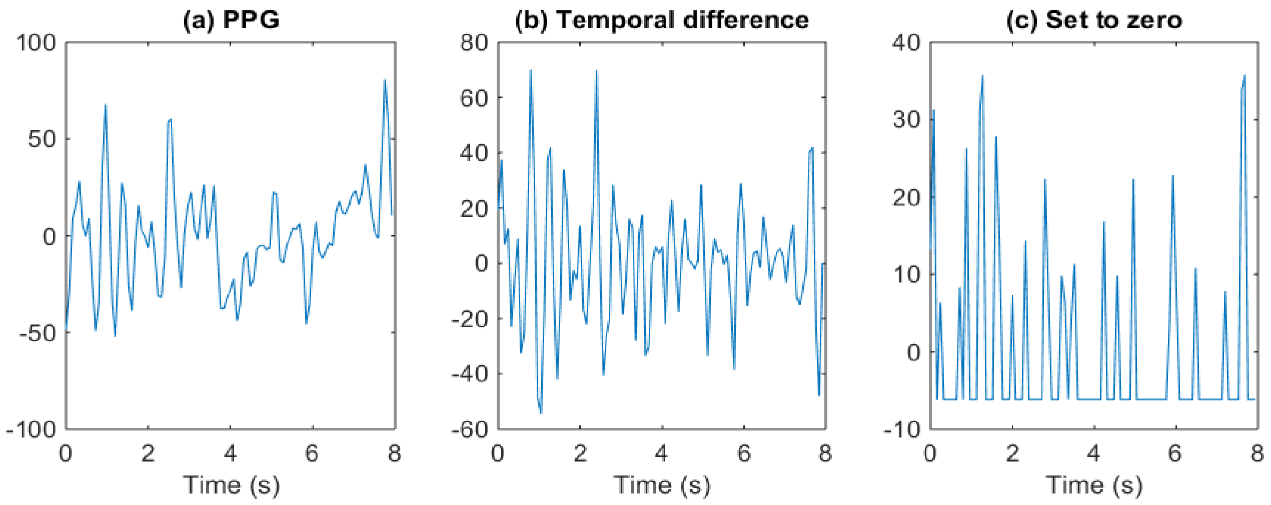


Figure 8.3: The processing of the PPG signal is depicted. Each step is marked subsequently from a to c. At the end a time signal is obtained from which the BPM is more difficult to calculate than the original signal.

8.2 Quantifying quality of a PPG signal

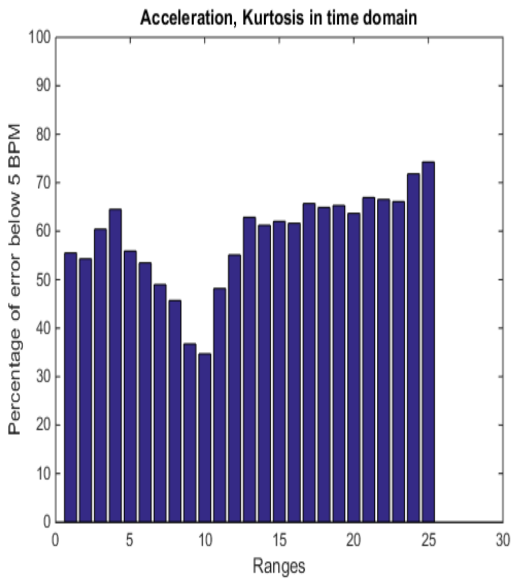
To identify when a simple periodogram can be used to determine the HR for PPG signals, a parameter has to be found to identify whether the PPG signal has good quality. The goal is thus to find a criterion for which the error produced by simply using the periodogram is low. A lot of different parameters have been tried out in order to identify the best possible criterion. Some of the parameters that have been researched are signal power, maximum peak in time domain, maximum peak in frequency domain and kurtosis in both domains. Kurtosis is a measurement for how peaked a signal is, and is calculated with the function **kurtosis** supplied by Matlab [6]. These parameters were researched for both the PPG and acceleration signal.

For each time window the HR error and the corresponding parameter are then saved. Using this information, a relation could be found between the corresponding parameter and the error. The quality is defined as the amount of times the error is below 5 BPM within a certain range of a parameter. To identify this property, the total amount data points have been sorted and divided into 25 units. The percentage of errors that are below 5 BPM is then calculated for each unit, and plotted against the units. In Figure 8.5a the kurtosis of the time domain acceleration signal was used as a the parameter. This figure illustrates an example where no useful relation between a parameter and the error can be found. On the other hand, using the power of the acceleration signal gives a better result, as can be seen from Figure 8.5b. From this figure it becomes evident that as the power of the acceleration increases, that percentage of errors below 5 BPM decreases. This result is intuitive, since having no acceleration would mean that there are little to no distortions due to MAs. From this, a criterion can be set to identify the quality of a PPG signal. Although power is a good parameter to set the criteria, a better parameter is the power of acceleration multiplied by the maximum peak of acceleration in time domain. The relation between this new hybrid parameter is displayed in Figure 8.5c. In this new relation there is more difference in the error percentage between low and high values of the parameter. Hence, power times the maximum peak is chosen as the criteria for identifying if the periodogram can be used. This parameter will be referred to as variable v_1. For using the periodogram two ranges have been identified and used. These ranges have been placed in Table 8.2. P_{range} is the percentage of times a windows has fallen within the given range and P_{error} is the percentage of errors that have been below 5 BPM in the given range. In general, if $v_1 < 10$, the PPG signal is qualified for the periodogram to be used for HR estimation. How these two ranges were used will become clear in the next chapter.

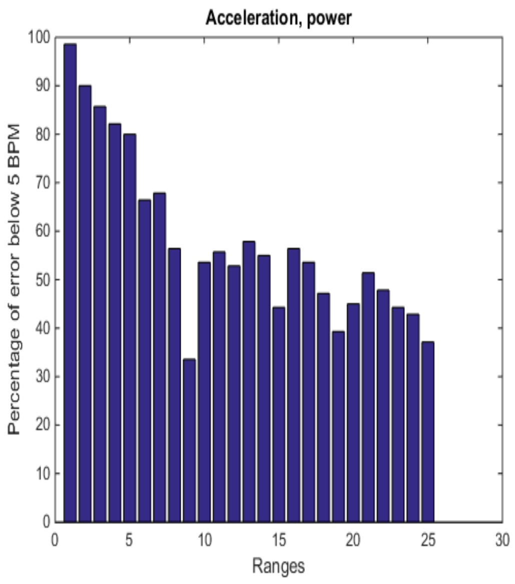
The same was done for ANC. Namely, trying to find a parameter for which the method, in this case ANC, could be used to easily determine the HR. In this case, kurtosis of the PPG signal in frequency domain was chosen to be the optimal parameter. The plot of the percentage of errors against the 25 units of the kurtosis can be observed in Figure 8.5d. Clearly, the quality of the PPG signal increases when the kurtosis of the PPG frequency spectrum is high. The range chosen with its characteristics are also displayed in Table 8.2, the value of the parameter is denoted with v_2.

Table 8.2: Ranges used in SMART for v_1 and v_2. v_1 is the parameter used for a simple periodogram, which is the power of acceleration multiplied by the maximum peak of acceleration in time domain. v_2 is the parameter used for ANC, which is the kurtosis of the spectrum of the PPG signal. P_{range} is the percentage of times a windows has fallen within the given range and P_{error} is the percentage of errors that have been below 5 BPM in the given range.

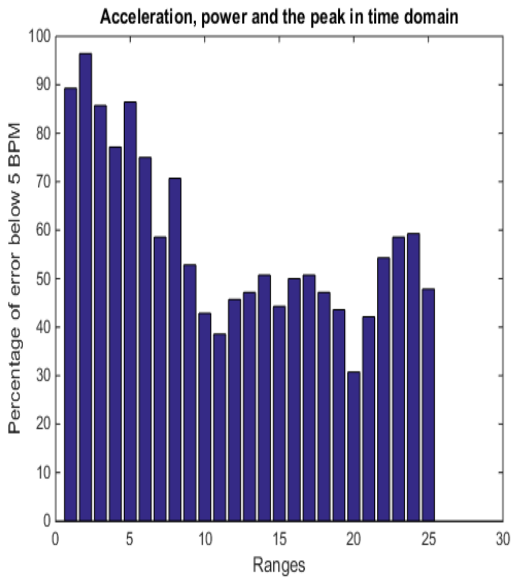
Range	P_{range}	P_{error}
$1 < v_1 < 15$	88 %	70 %
$v_1 < 1$	10 %	95 %
$v_2 > 20$	78 %	74 %



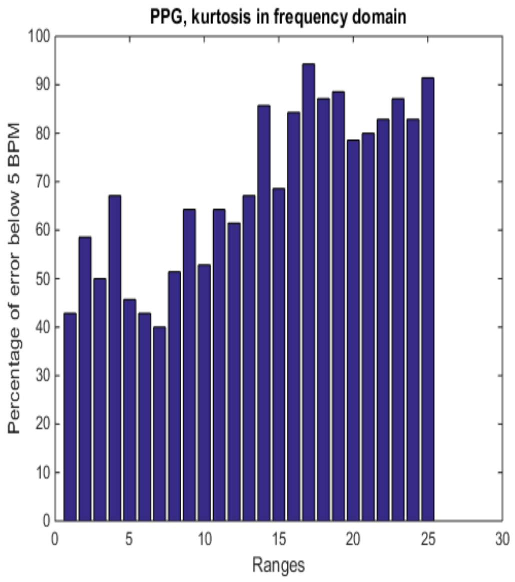
(a) Kurtosis of the acceleration signal is used as the parameter.



(b) Power of the acceleration signal is used as the parameter.



(c) The power of the acceleration is multiplied by the peak in time domain. This is used as the parameter.



(d) Kurtosis of the PPG signal in frequency domain is used as the parameter

Figure 8.5: For these plots different parameters were used. For each parameter the total amount of data points has been sorted and divided into 25 units. The percentage of errors that are below 5 BPM is then calculated for each unit, and plotted against the units.

Chapter 9

SMART

JOSS is a robust solution for estimating the HR with low error, as will be shown in chapter 10. Nevertheless, this solution requires a lot of computing power and could drain the battery of the device. Therefore, for monitoring the HR throughout the day, SMART is proposed as an alternative framework. SMART combines multiple signal processing techniques by means of stages. In the first stage a fast method of determining the HR is present, such as a simple periodogram. In the subsequent stages there are slower, but more accurate, signal processing techniques. If none of the techniques within these stages are accurate enough to calculate the HR, the last stage is triggered. In the last stage, a Robust Peak Tracking System (RPTS) is used for computing the HR. JOSS and TORIKA are examples of these systems. SMART determines which of these techniques to use based on the quality of the PPG signal and the previously determined HR. This framework uses the robustness of a RPTS, while using less computing power whenever possible. The block diagram of this framework can be found in Figure 9.1. The criteria needed to determine the quality of a PPG signal, are documented in Chapter 8.

9.1 Working principle

The working principle of SMART will be explained by means of the block diagram found in Figure 9.1. The inputs for this algorithm are the PPG signals, acceleration signals and the previously calculated BPM value. The BPM value is initialized with the value -1. This is used to check whether SMART is in the initialization stage. If so, then the data is sent to the RPTS to compute the HR. If not, the information is sent to the first test block. This test block determines if the quality of the PPG signal is sufficient to determine the HR by using the first method. For validation, the HR calculated by the first method is checked to see if it is within a certain range of the previous HR. If this is the case, the calculated HR is deemed correct and is given as output. If this is not the case, the next stage in SMART is triggered. It is worth noting that the criteria and ranges for each stage can vary. Finally, if all tests fail, the data is passed to the RPTS to compute the HR. It is important that the faster methods are evaluated first, in order to minimize the computing load of SMART. An additional note is that all methods used in the framework should be applied frequently. If for example method 3 is never applied, computing power is lost since the criterion for this method is constantly checked, but the method is never used. In conclusion, this framework combines the speed of certain signal processing techniques, but has a RPTS to provide robustness when the quality of the PPG signal is low.

9.2 Implementation

Trying to find a relation between error and a characteristic of a PPG signal is time consuming. Hence, because of time limitations, searching for criteria was only done for a simple periodogram and ANC. Two versions of SMART were created, both used JOSS as a RPTS since JOSS performs better than TROIKA, as will be shown in Chapter 10. The first version is a version with only one stage before JOSS, this stage uses a simple periodogram to calculate the HR. This version will be referred to as SMART 1. The second version was implemented with two stages, both a simple periodogram and ANC, this version will be referred to as SMART 2. For ANC, the LMS implementation supplied by subgroup A3 was used. For the range tests, the ranges found in Table 8.2 were used. These ranges were used in the decision blocks of the periodogram method as follows. If $v_1 < 1$ then the determined HR has to be within a range of 40 of the previous BPM, or in mathematical notation Delta $= |\text{BPM}_{\text{periodogram}} - \text{BPM}_{\text{previous}}| < 40$. Otherwise, if $1 < v_1 < 15$ then Delta has to satisfy Delta $= |\text{BPM}_{\text{periodogram}} - \text{BPM}_{\text{previous}}| < 10$. If these conditions are met then the determined HR is used as the current BPM value, otherwise the data is sent to the decision blocks of the successive method. In the case of SMART 1 the successive method is just JOSS, in SMART 2 it is ANC. For ANC only one range was used, namely if $v_2 > 30$ then the PPG is deemed good enough for ANC to estimate the correct HR. Although the determined HR (BPM_{ANC}) has to satisfy Delta $= |\text{BPM}_{\text{ANC}} - \text{BPM}_{\text{previous}}| < 7$, in order for it to be used as the current HR. Once again, if this condition is not met for SMART 2 then the data is sent to JOSS to determine the HR.

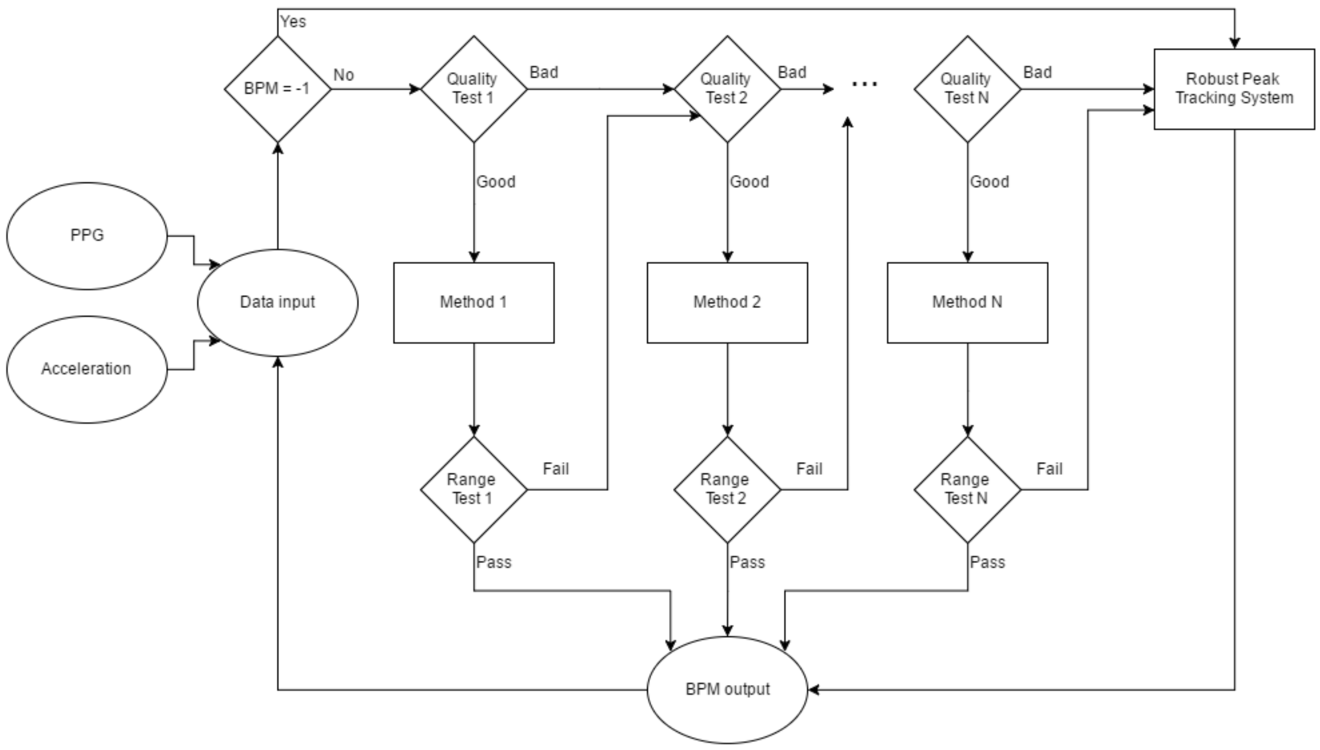


Figure 9.1: The framework of SMART depicted in a flowchart.

Chapter 10

Results

10.1 Evaluation

Twelve datasets were made available for our project group to test our final systems on. The system reads the PPG signals of these datasets and estimates a HR. This HR is then compared to the true BPM value. For evaluation we used the average absolute error given by equation (10.1) [1].

$$\mu = \frac{1}{\mathrm{N}} \sum_{i=1}^{N} |\mathrm{BPM}_{\mathrm{est}}(i) - \mathrm{BPM}_{\mathrm{true}}(i)| \tag{10.1}$$

Here μ is the average absolute error, N is the number of time windows, $\mathrm{BPM}_{\mathrm{true}}(i)$ the true BPM value for the i-th time window and $\mathrm{BPM}_{\mathrm{est}}(i)$ the estimated value of the algorithm for the i-th time window.

10.2 Performance on all datasets

As explained in Chapter 9, two different versions of SMART have been implemented. SMART 1 is a version where only a simple periodogram is added before JOSS, and SMART 2 uses both a simple periodogram and ANC as stages before JOSS. For these three methods of tracking the HR, the error per dataset was calculated with Equation 10.1. These results, and the results of TROIKA and ANC with SPT, are available in Table 10.1. At the end of each row the average of all datasets is displayed. In Table 10.2 the percentage of usage of each of our methods is shown. For easy comparison between methods the average error and the execution time are displayed in Table 10.3.

Table 10.1: Average absolute errors, in BPM, of the 12 datasets and the final value in the last column.

	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9	Set 10	Set 11	Set 12	**Average**
ANC with SPT	1.79	1.09	0.66	1.09	0.84	1.14	0.95	0.62	0.44	3.92	5.87	1.46	**1.66**
TROIKA	3.01	2.50	2.22	1.48	0.68	1.13	0.66	0.73	0.51	6.50	1.13	1.77	**1.86**
JOSS	2.26	2.65	1.03	1.21	0.59	1.18	0.67	0.55	0.57	3.67	0.78	1.48	**1.39**
SMART 1	2.57	2.51	0.72	1.07	0.72	1.14	0.65	0.47	0.46	3.82	0.81	1.65	**1.38**
SMART 2	2.59	2.52	0.74	1.06	0.73	1.19	0.61	0.47	0.46	4.09	0.88	1.68	**1.42**

Table 10.2: The usage of each method in SMART.

	Periodogram	ANC	JOSS
JOSS	0 %	0 %	100 %
SMART 1	64 %	0 %	36 %
SMART 2	64 %	21 %	15 %

Table 10.3: The usage of each method in SMART.

	ANC with SPT	TROIKA	JOSS	SMART 1	SMART 2
Execution Time (s)	42	183	186	67	37
Average error (BPM)	1.66	1.86	1.39	1.38	1.42

10.3 BPM traces

To illustrate the tracking of HR, BPM traces of our best performing algorithm (SMART 2) are made available. The BPM traces for the best performing, average performing and worst performing datasets can be found in figure 10.1, 10.2 and 10.3, respectively. These traces correspond to dataset 9, 10 and 11 respectively.

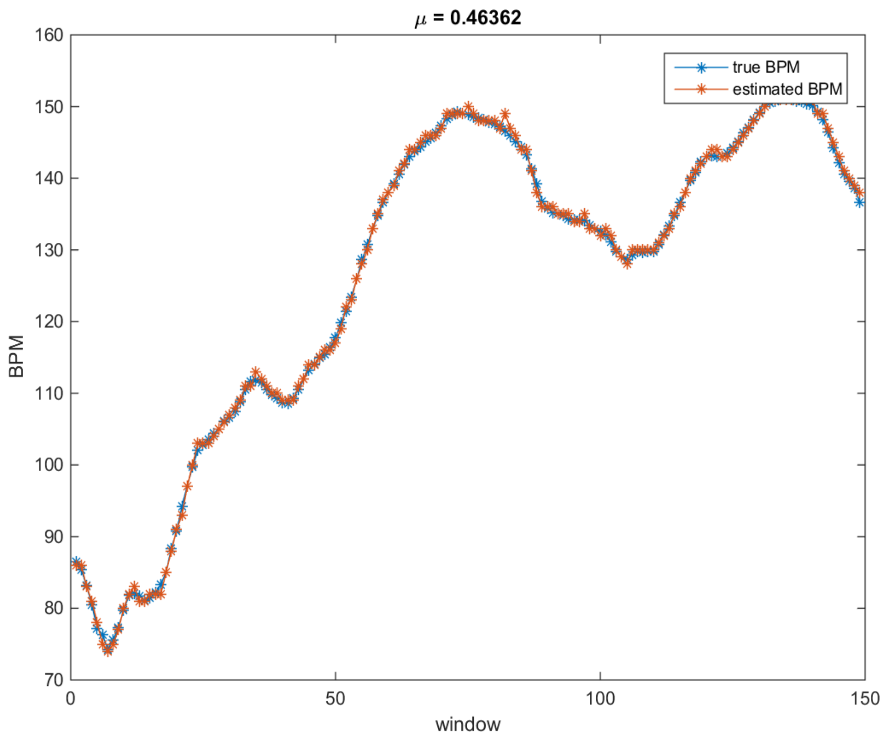


Figure 10.1: Best performing dataset: dataset 9.

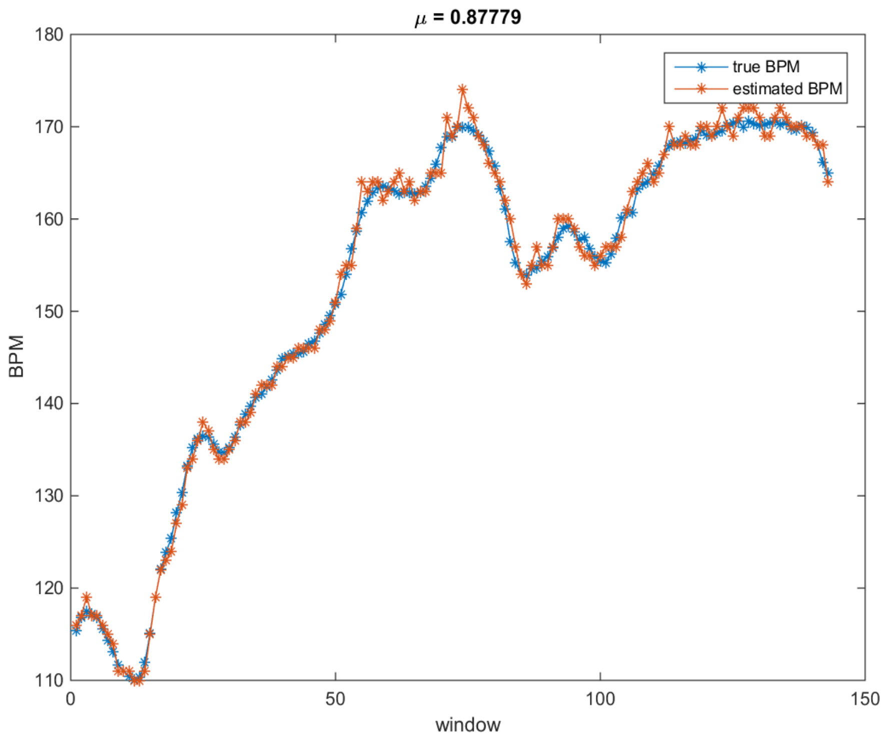


Figure 10.2: Average performing dataset: dataset 10.

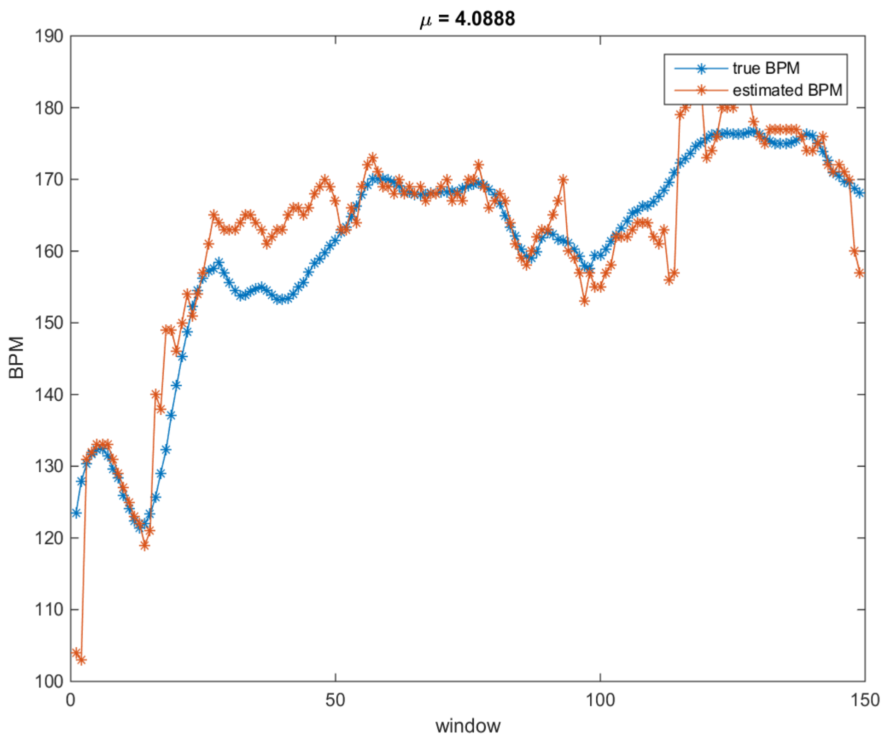


Figure 10.3: Worst performing dataset: dataset 11.

Chapter 11

Discussion

From table 10.1 it is clear that JOSS, SMART 1 and SMART 2 are more or less equally accurate. Although there is not a significant improvement in terms of accuracy, there actually is a significant improvement in execution time. SMART 2 is five times faster than JOSS. Another observation that can be made from these tables is that using SMART 2 instead of SMART 1 increases the error by 0.04 BPM, but decreases the execution time by 30 seconds. This is equal to an accuracy loss of $2.9\,\%$ and a speedup of $45\,\%$. From this observation it is concluded that using ANC within SMART is worth using. SMART 2 is even faster and more accurate than ANC with spectral peak tracking. For this reason we use SMART 2 as our final product. It is worth noting from Table 10.2 that SMART 2 only uses JOSS $15\,\%$ of the time, even though it mostly preserves its robustness. It should be noted that the usage of JOSS would dramatically decrease if SMART is used for HR monitoring throughout the day. Namely, the datasets were of people running on a thread mill. And people are relatively stationary throughout the day, resulting in higher quality PPG signals.

Because TROIKA has the highest average error, it is not a good canditate for being the robust peak tracking system in SMART. What's more, it is far too slow to be a candidate to speed up SMART. ANC on the other hand, speeds up SMART by putting it in a stage before JOSS. And because of the low average error it is a good candidate for being a robust peak tracking system.

Section 11.1 will evaluate the system requirements we set in Chapter 3 and Section 11.2 analyses the performance on dataset 10 and discusses the limitations of JOSS.

11.1 Evaluating the system requirements

Over the twelve datasets there are 1768 windows. For JOSS, this would result in an execution time of $0.1\,\mathrm{s}$ per window. Since JOSS is the slowest, it means that all methods meet the requirement of having an execution time per window below $2\,\mathrm{s}$. However, this is done on a desktop computer [12]. A desktop computer is usually much faster than a wearable device. In order for SMART to be implemented on a wearable device, the computation power of the device cannot be more than 20 times slower than the PC. Otherwise this would lead to the computation time being longer than the time step of $2\,\mathrm{s}$. For current generation smartwatches this is usually the case but there are a few older models that do not meet this requirement. The slow execution time of JOSS can be compensated by faster techniques in the earlier stages of SMART, but during intense physical exercising, JOSS should ideally be faster. Nevertheless, for current generation smartwatches we can confidently conclude that SMART is runnable on these devices. All methods also meet the requirement in accuracy of having an average error below 5 BPM and an individual error below 10 BPM.

11.2 Analyzing dataset 10

Dataset 10 is the worst performer with an error of 4.09 BPM. If we were to analyze the BPM trace of dataset 10, we can see that from window 26 to 47 the error in BPM stays around 10. This contributes significantly to the average error of the dataset. In Figure 11.1 window 28 from dataset 10 is plotted. The PPG signal, periodogram, sparse PPG, sparse acceleration and the specral substraction are shown, respectively. From the acceleration spectrum it can be seen that motion artifacts are present close to the HR. The HR is likely buried in the peak of acceleration. When the energy of the HR is low in the periodogram, SSR struggles to recover the peak and therefore it becomes lost. This situation occurs for many successive windows, which explains the errors for these windows.

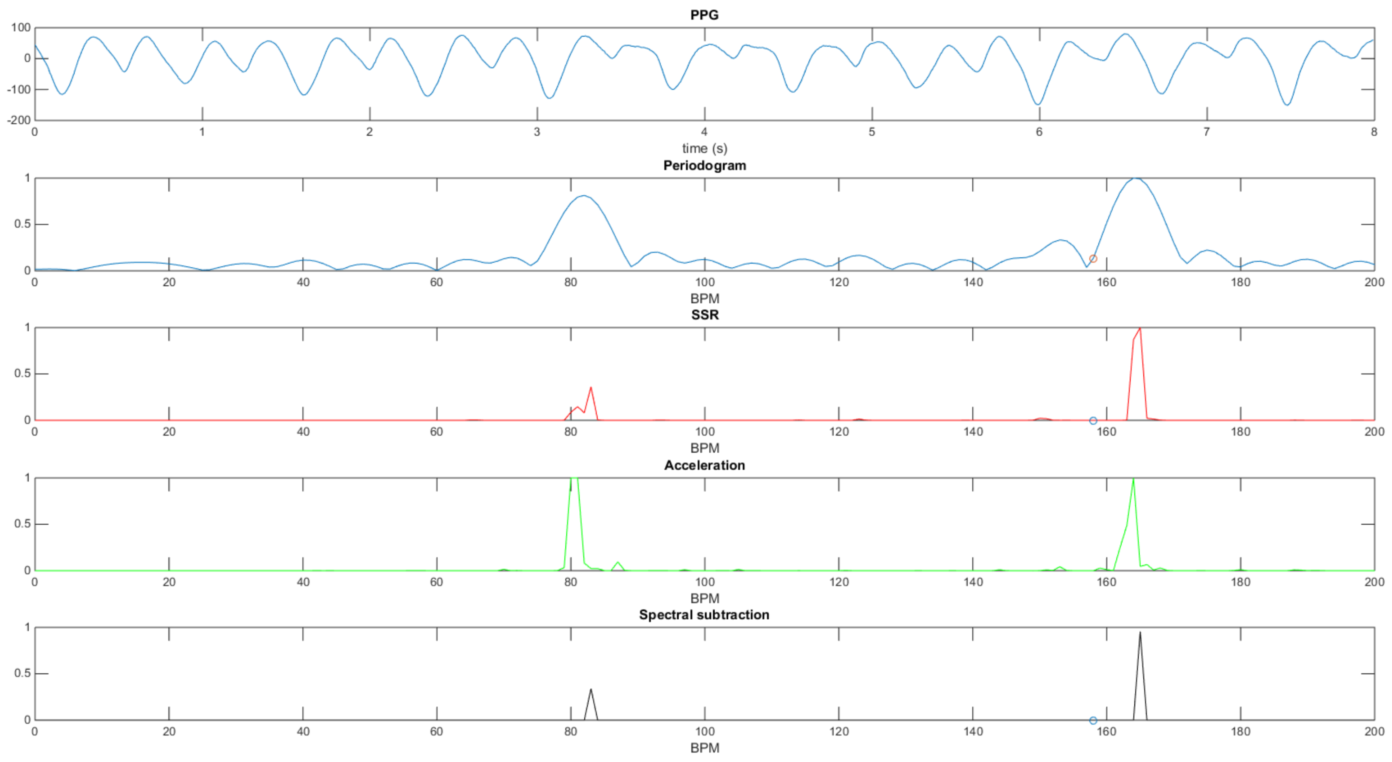


Figure 11.1: PPG signal, periodogram, SSR, acceleration and spectral subtraction of window 28. The dot indicates the true BPM. From the figure it can be seen that the true HR is close to an acceleration peak

Another interesting situation is window 115. The error in BPM from window 115 to 119 is above 10 on average. From Figure 11.2 we can see that the HR is at the exact bin where motion artifacts are present. Since these are subtracted from the PPG signal, the BPM peak is lost. Peak tracking looks for the peak closest to the previous BPM and then chooses 179 BPM. Since this situation occurs consecutively, the wrong peak is tracked multiple windows.

These are a few examples of where the performance of JOSS is limited. Despite these bad cases, JOSS has very robust performance and error handling. Peaks lost by motion artifacts, noise or poor signal quality are dealt well with by spectral peak tracking.

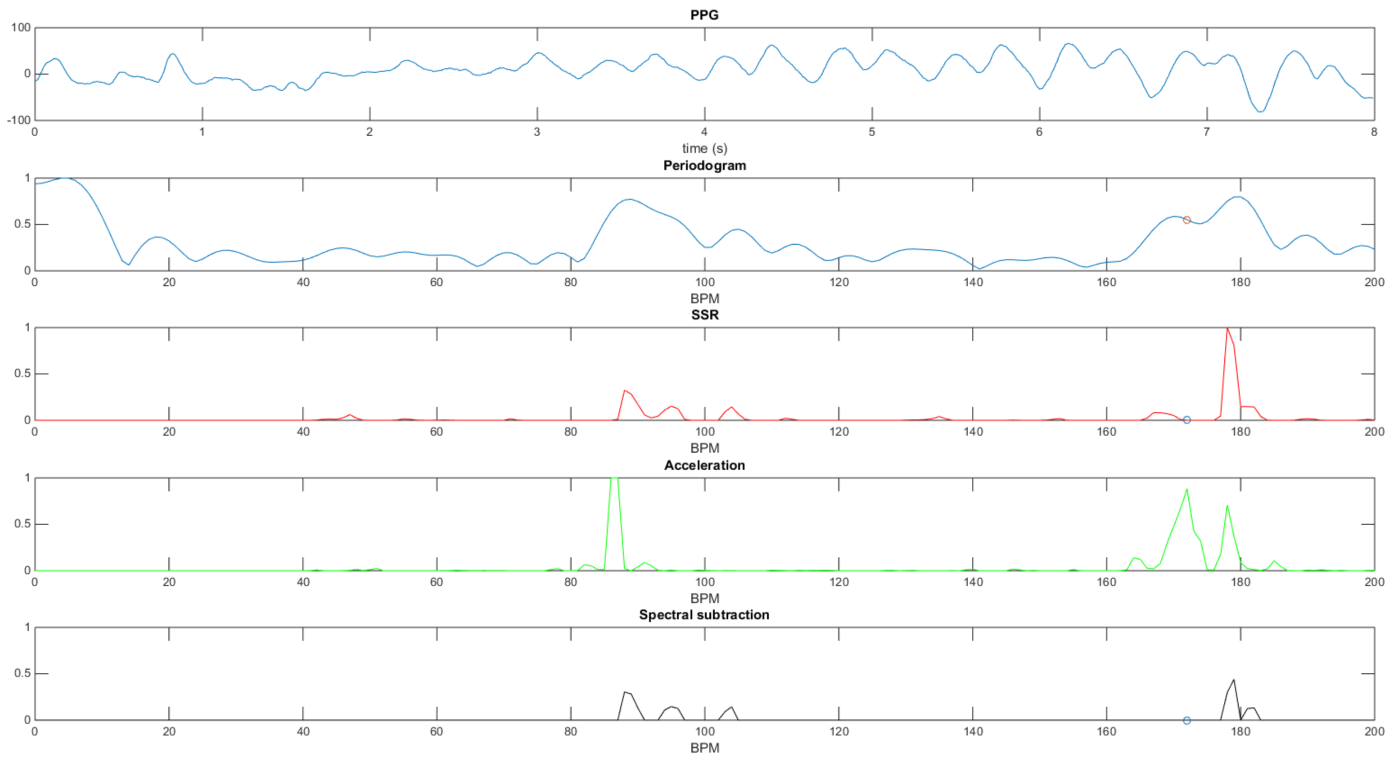


Figure 11.2: PPG signal, periodogram, SSR, acceleration and spectral subtraction of window 115. The dot indicates the true BPM. From the figure it can be seen that the true HR is at the acceleration peak.

Chapter 12

Conclusion

12.1 Conclusions

PPG signals recorded from the wrist are often subjected to motion artifacts and noise. In order to monitor the HR using PPG signals, existing signal processing techniques and frameworks that deal with similar issues were examined. One of these frameworks is JOSS. JOSS consists of sparse signal reconstruction to estimate a sparse spectrum, spectral subtraction to remove motion artifacts and spectral peak tracking to identify HR peaks. JOSS was then implemented and tested on the 12 given datasets. The performance of JOSS was well within expectations. It resulted in an absolute average error of 1.39 over the 12 datasets. Nevertheless, JOSS has a high computational load caused by large matrix operations. To solve this, an analysis of PPG signals was done in order to identify characteristics of a good quality PPG signal. When good PPG signals are recognized, alternative signal processing techniques can be used that are much faster than JOSS in computing the HR. Integrating this idea with JOSS led to the creation of SMART. SMART uses multiple noise cancelling techniques, but uses JOSS as the key constituent for tracking the HR. SMART has the robustness of JOSS, but the speed of faster techniques when the quality of the PPG signal is sufficient. It has also the flexibility of incorporating other methods, for improvement in speed or accuracy. Our implementation of SMART uses a periodogram, ANC and JOSS to determine the HR. For ANC, the LMS filter implementation of subgroup A3 was used. This implementation of SMART performs almost five times as fast as JOSS, and the average absolute error slightly changed from 1.39 to 1.42. The overall performance of SMART is well within the program of requirements, and shows that the algorithm has a high potential of being used for fitness or medical purposes.

12.2 Recommendation and future work

12.2.1 Combining multiple PPG signals

For this project we were given datasets that include simultaneously recorded ECG, 2 PPG and 3 acceleration signals (the x, y and z components). An investigation on fusing both PPG signals was done by subgroup A3. In theory, this research can be very valuable to increase the quality of the PPG signal. However, for these datasets, 1 PPG signal always performed worse than the other, therefore this research did not add much value to the end result. This work can however be very useful for combining 2 or even more PPG signals where the quality of the signals are similar.

12.2.2 Optimizing SSR

JOSS has a relatively heavy computational load since it uses SSR four times to compute the HR for 1 time window. If SSR can be optimized, JOSS can be significantly faster. When constructing the Fourier matrix for SSR, not all frequency bins are necessary. Since we are band-passing the spectrum, a great portion of the spectrum is thrown away. In theory this means that the matrix can be constructed for selected frequencies only. This can reduce the size of the matrix, which greatly reduces the computational load, increasing the speed of JOSS.

12.2.3 Improving MA removal

Spectral subtraction now simply subtracts the acceleration spectrum of the PPG spectrum per frequency bin. However, this does not always fully remove the peak originating from MA. This is only the case when the

the acceleration peak is bigger than the peak in the PPG spectrum. One may add a factor that can control the amount of subtraction in the PPG spectrum. This factor can be chosen as a function of the acceleration peak value: subtract more when close to a peak, subtract less when this is not the case.

12.2.4 Improving SPT

Our implementation of SPT is relatively simple. It searches for peaks in a selective range and increases this if no peaks are found. If the increased range contains no peaks, the previous value is taken. If this happens multiple times in succession, the full spectrum is searched. A more intelligent version of this approach can be implemented. An approach that predicts the next HR peak based on previous estimations. This can be used to search more selectively and can also be used as an improved validation for the located peak.

12.2.5 Improving SMART

Since SMART is a general framework of noise cancelling methods and a robust peak tracking system, there are a lot of different possible implementations of SMART. From the HR monitoring methods we analyzed we concluded that JOSS, in terms of accuracy, is the best algorithm to be used as the peak tracking system. Nevertheless, SMART could also be used with other methods and with possibly more stages.

Appendix A

MATLAB code

A.1 SMART

Listing A.1: SMART

```
function [ curLoc, curBPM, Trap_count, Lssr ] = SMART( sigWindow, ...
                    prevLoc, prevBPM, Trap_count, N, Fs, Lssr )

%========================================================================%
%                     FUNCTION DESCRIPTION                               %
%========================================================================%
%SMART is a function that performs the algorithm of SMART. It uses a    %
%periodogram or ANC whenever possible, and uses JOSS as the base for HR %
%tracking.                                                              %
%========================================================================%



%============================================================%
%                  INPUT/OUTPUT PARAMETERS                   %
%============================================================%
%  - Sigwindow is a matrix with the signal in its rows:     %
%    SigWindow[PPG1; PPG2; Ac1; Ac2; Ac3].                   %
%  - prevLoc and prevBPM are the bin location and value of %
%    the previous HR repectively.                            %
%  - Trap_count is the amount of times JOSS did not find    %
%    anything.                                               %
%  - N is the resolution of the spectra.                     %
%  - Fs is the sampling rate.                                %
%  - Lssr is the length of the spectrum calculated by JOSS %
%============================================================%

%=========================%
% INPUT INITIALIZATION   %
%=========================%
% prevLoc = -1;          %
% prevBPM = -1;          %
% testLoc = -1;          %
% Trap_count = 0;        %
% Lssr = 0;              %
%=========================%


    %================================================================
    %See if the periodogram is good enough to determine the HR.
    %If the values of testLoc and testBPM are -1, the next stage is
    %triggered. Otherwise the values are used as the current HR values.
    %================================================================
    if prevLoc > 0
        [ testLoc, testBPM ] = Periodogram( sigWindow, prevLoc, ...
                                    Fs, Lssr );
    end
    %==================================================================



    %================================================================
    %See if ANC is good enough to determine the HR.
    %If the values of testLoc and testBPM are -1, the next stage
    %is triggered. Otherwise the values are used as the current BPM
    %values. To disable it (create SMART 1) just comment it.
    %================================================================
    if prevLoc > 0 && testLoc == -1
        [ testLoc, testBPM ] = ANC( sigWindow, prevLoc, Fs, Lssr);
    end
    %================================================================


    %If none of the previous methods could find the HR try JOSS.
    if prevLoc == -1 || testLoc == -1
        %======================================================%
        %                      JOSS
        %======================================================%
        % SSR
        [~, SSRsig] = SSR_JOSS(sigWindow, Fs, N);
        Lssr = length(SSRsig);

        % SPT
        [curLoc, curBPM, Trap_count] = SPT_JOSS(SSRsig, ...
        Fs, prevLoc, prevBPM, Trap_count);
       %=======================================================%
    else
        curLoc = testLoc;
        curBPM = testBPM;
    end
end
```

Listing A.2: Periodogram

```
function [ testLoc, testBPM ] = Periodogram( sigWindow, prevLoc, Fs, Lssr )

%======================================================================%
%                     FUNCTION DESCRIPTION                             %
%======================================================================%
%Periodogram is a function that determines the heart rate using a      %
%periodogram, if the tests of quality and range are not met it returns  %
%the values -1.                                                         %
%======================================================================%



%==========================================================%
%                  INPUT/OUTPUT PARAMETERS                 %
%==========================================================%
%  - Sigwindow is a matrix with the signal in its rows:    %
%    SigWindow[PPG1; PPG2; Ac1; Ac2; Ac3].                 %
%  - prevLoc is the bin location of the previous BPM value %
%  - Fs is the sampling rate.                              %
%  - Lssr is the length of the spectrum calculated by JOSS %
%  - testLoc and testBPM are the bin and value of the HR   %
%    respectively.                                         %
%==========================================================%



    %Calulate the absolute value of the acceleration
    acc = sqrt( sigWindow(3,:).^2 + sigWindow(4,:).^2 + sigWindow(5,:).^2);
    %Calculate the quality of the PPG
    quality = (rms(acc)^2)*max(acc);



    %===================================================================%
    %If the value of quality is below 15 it is categorized as being good %
    %enough for calculating the HR with a periodogram.                   %
    %===================================================================%
    if (quality < 15)

        %Initialize parameters
        %=========================
        threshold = [10 40];
        N = 2^10;
        BPM = 60*Fs/N*((1:N) - 1);
        BPM_max = BPM(end);
        %=========================


        %Get the PPG signal and its spectrum
        %=============================================
        ppg = sigWindow(1,:);
        ppg = ppg - mean(ppg);

        spec = abs(fft(ppg, N));
        Sspec = size(spec,1);
        [ H ] = H_filter( 200, BPM_max, N, -1, Sspec );
        spec = spec.*H;
        %=============================================


        %Determine the previous and current BPM
        %=============================================
        [~, ind] = max(spec);
        testBPM = BPM(ind);

        testLoc = round(testBPM*Lssr/(60*Fs) + 1);

        BPMssr = 60*Fs/Lssr*((1:Lssr) - 1);
        testBPM = BPMssr(testLoc);
        prevBPM = BPMssr(prevLoc);
        %=============================================


        %Range test
        %=============================================
        difference = abs(testBPM - prevBPM);

        if quality < 1
            TH = threshold(2);
        else
            TH = threshold(1);
        end

        if difference > TH
            testLoc = -1;
            testBPM = -1;
        end
        %=============================================

    else
        testLoc = -1;
        testBPM = -1;
    end
end
```

Listing A.3: ANC

```
function [ testLoc, testBPM ] = ANC( sigWindow, prevLoc, Fs, Lssr )
%=========================================================================%
%                      FUNCTION DESCRIPTION                              %
%=========================================================================%
%ANC is a function that determines the heart rate using adaptive noise   %
%cancellation, if the tests of quality and range are not met it returns %
%the values -1.                                                          %
%=========================================================================%



%============================================================%
%                  INPUT/OUTPUT PARAMETERS                    %
%============================================================%
%  - Sigwindow is a matrix with the signal in its rows:      %
%    SigWindow[PPG1; PPG2; Ac1; Ac2; Ac3].                    %
%  - prevLoc is the bin location of the previous BPM value %
%  - Fs is the sampling rate.                                 %
%  - Lssr is the length of the spectrum calculated by JOSS %
%  - testLoc and testBPM are the bin and value of the HR    %
%    respectively.                                            %
%============================================================%


    %Determine parameters
    %=========================
    threshold = 7;
    filterorder = 74;
    step_size = 0.045;
    N = Lssr;
    BPM = 60*Fs/N*((1:N) - 1);
    BPM_max = BPM(end);
    %=========================


    %Get the PPG signal and its spectrum
    %============================================
    ppg = sigWindow(1,:);
    ppg = ppg - mean(ppg);

    spec = abs(fft(ppg, N));
    Sspec = size(spec,1);
    [ H ] = H_filter( 200, BPM_max, N, -1, Sspec );
    spec = spec.*H;
    %============================================


    %Calculate the quality
    quality = kurtosis(spec);


    %=====================================================================%
    %If the value of quality is over 20 it is categorized as being good  %
    %enough for calculating the HR with ANC.                             %
    %=====================================================================%
    if  quality > 20

        %Perform ANC and get its spectrum
        %===============================================================
        acc_data = sigWindow(3:5,:);
        [ clean_signal ] = LMS_sig(ppg,acc_data,filterorder,step_size);

        specLMS = abs(fft(clean_signal,N));
        StLi = size(specLMS, 1);
        H = H_filter( 14, BPM_max, N, prevLoc, StLi );
        specLMS = specLMS.*H;
        %================================================================


        %Determine the previous and current BPM
        %=============================================
        [~,testLoc] = max(specLMS);
        testBPM = BPM(testLoc);
        prevBPM = BPM(prevLoc);
        %=============================================


        %Range test
        %=============================================
        difference = abs(testBPM - prevBPM);
        if difference > threshold
            testLoc = -1;
            testBPM = -1;
        end
        %=============================================

    else
        testLoc = -1;
        testBPM = -1;
    end
    %======================================================================%
end
```

Listing A.4: LMS_sig

```
%=============================================%
%This Function was provided by subgroup A3. %
%=============================================%

function [ clean_signal ] = LMS_sig(signal,acc_data,k,step_size)
    temp = zeros(1,length(signal))';
    mu = step_size; % step size

    lms2 = dsp.LMSFilter('Length',k, ...
   'Method','Normalized LMS',...
   'StepSizeSource','Input port', ...
   'WeightsOutputPort',false);

    for i = 1:3
        x = acc_data(i,:)';
        d = signal';
        [~, err] = step(lms2,x,d,mu);
        temp = temp+err;
    end
    clean_signal = temp;
end
```

Listing A.5: H_filter

```
function [ H ] = H_filter( delta, BPM_max, N, oldLoc, StLi )


%========================================================================%
%                      FUNCTION DESCRIPTION                              %
%========================================================================%
%H_filter is a function that creates the response spectrum H of a band  %
%pass filter.                                                            %
%========================================================================%
```

```
%============================================================%
%                    INPUT/OUTPUT PARAMETERS                 %
%============================================================%
%  - delta is the halve of the total band pass width.        %
%  - BPM_max is the maximum value of the spectrum to be      %
%    band passed.                                            %
%  - N is the resolution of the spectra.                     %
%  - oldLoc is the bin position of the previous HR.          %
%  - StLi is the size(spectrum) in order to know if H is     %
%    vertical or horizontal.                                 %
%============================================================%


    %================================
    %Check if the oldLoc is a logical
    %value. If so, zoom in with delta.
    %================================
    if oldLoc > 0
        Ndelta = (delta/BPM_max)*N;
        Ndelta = ceil(Ndelta);
        Nmin = oldLoc - Ndelta;
        Nmax = oldLoc + Ndelta;

        if Nmin <= 0 && Nmax > N
            range1 = 1:N;
        elseif Nmin <= 0
            range1 = 1:Nmax;
        elseif Nmax > N
            range1 = Nmin:N;
        else
            range1 = Nmin:Nmax;
        end

        if StLi == 1
            H = zeros(1,N);
        else
            H = zeros(N,1);
        end

        H(range1) = 1;
    else %Just band pass from 40 to 190
        Lmin = floor((40/BPM_max)*N + 1);
        Lmax = ceil((190/BPM_max)*N + 1);
        range1 = Lmin:Lmax;

        if StLi == 1
            H = zeros(1,N);
        else
            H = zeros(N,1);
        end

        H(range1) = 1;
    end
end
```

A.2 JOSS

Listing A.6: SSR_JOSS

```
function [BPM, SSRsig, S, C] = SSR_JOSS(y, Fs, N)
%   Removes MA from y by means of spectral subtraction

%   y       :   time-signal
%   Fs      :   sampling rate
%   N       :   resolution
%   BPM     :   frequency-axis in BPM
%   SSRsig  :   cleansed sparse spectrum
%   S       :   Sparse spectra of PPG and acceleration signals - used
%               for debugging purposes
%   C       :   max of sparse spectra of acceleration signals - used
%               for debugging purposes


    % if no resolution is given
    if (nargin == 2)
        % resolution is set to 1 BPM
        N = 60*Fs;
    end
    % create object
    sigY = signal(y);
    % construct Y matrix
    Y = [sigY.ppg1; sigY.x; sigY.y; sigY.z];
    % Y must be M x L
    Y = transpose(Y);
    L = size(Y,2);
    S = zeros(N,L);
    for i = 1:L
        [BPM, SS] = SSR(Y(:,i), Fs, N);
        % normalize spectra
        SS = SS / max(SS);
        S(:,i) = SS;
    end

    aggression = 0.99;
    C = zeros(1,N);
    SSRsig = S(:,1);

    for i = 1:length(BPM)
        % max of acceleration at f_i
        C(i) = max([S(i,2),S(i,3),S(i,4)]);
        % spectral subtraction
        SSRsig(i) = SSRsig(i) - aggression*C(i);
    end

    % set values smaller than 1/5-th of the max to 0
    p_max = max(SSRsig);
    SSRsig(SSRsig < p_max / 5) = 0;
end
```

Listing A.7: signal

```
classdef signal
    % Creates a signal object based on the given datasets
    % This version deletes ECG
    properties
        ppg1;
        ppg2;
        x;
        y;
        z;
    end
    methods
        function signalObj = signal(sig)
            signalObj.ppg1 = sig(1,:);
            signalObj.ppg2 = sig(2,:);
            signalObj.x = sig(3,:);
            signalObj.y = sig(4,:);
            signalObj.z = sig(5,:);
        end
    end
end
```

Listing A.8: SPT_JOSS

```
function [curLoc, curBPM, Trap_count] = SPT_JOSS(SSRsig, Fs, ...
    prevLoc, prevBPM, Trap_count)
%   Searches for the HR peak by using previously estimated HR values.

%   SSRsig      :   cleaned spectrum
%   Fs          :   sampling rate
%   prevLoc     :   frequency bin of previously located HR,
%                   initialized as -1
%   prevBPM     :   previous HR,
%                   initialized as -1
%   Trap_count  :   counter that keeps track of the number of times
%                   no peaks were found, initialized as 0
%   curLoc      :   frequency bin of currently located HR
%   curBPM      :   current HR

    % parameters SPT
    delta1 = 15;
    delta2 = 25;

    N = length(SSRsig);

    % initialization state
    if ((prevLoc == -1 && prevBPM == -1))
        curLoc = find(SSRsig == max(SSRsig));
        curBPM = 60 * (curLoc - 1) / N * Fs;
    else
        % set search range
        R1 = (prevLoc - delta1) : (prevLoc + delta1);
        [~,locs] = findPksInRange(SSRsig, R1);

        numOfPks = length(locs);
        if (numOfPks >= 1)
            % find closest peak
            curLoc = findClosestPeak(prevLoc, locs);
            curBPM = 60 * (curLoc - 1) / N * Fs;
        else
            % increase search range
            R2 = (prevLoc - delta2) : (prevLoc + delta2);
            % find peaks in range2
            [pks,locs] = findPksInRange(SSRsig, R2);

            numOfPks = length(locs);
            if (numOfPks >= 1)
                % find maximum peak
                [~, maxIndex] = max(pks);
                curLoc = locs(maxIndex);
                curBPM = 60 * (curLoc - 1) / N * Fs;
            else
                % choose prev BPM
                curLoc = prevLoc;
                curBPM = prevBPM;
            end
        end
    end

    % validation
    if (curLoc == prevLoc)
        Trap_count = Trap_count + 1;
        if (Trap_count > 2)
            % discover
            curLoc = discoverPeak(SSRsig, prevLoc);
            curBPM = 60 * (curLoc - 1) / N * Fs;
        end
    else
        Trap_count = 0;
    end
end
```

Listing A.9: findPksInRange

```
function [pks,locs] = findPksInRange(SSRsig, R)
%   Finds peaks of SSRsig in range R

%   SSRsig  :   spectrum
%   R       :   search range
%   pks     :   peak values
%   locs    :   peak frequency bins

    N = length(SSRsig);
    H = zeros(N,1);
    % filter for search range, handles cases when search range is negative
    H(R(R > 0)) = 1;
    % Band-pass filters spectrum to match search range
    SSRsig = SSRsig.*H;
    % returns peak values and locations
    [pks,locs] = findpeaks(SSRsig);
end
```

Listing A.10: findClosestPeak

```
function curLoc = findClosestPeak(prevLoc, locs)
%   Finds the closest peak to prevLoc

%   prevLoc     :   frequency bin of previously located HR
%   locs        :   frequency bins of peaks in search window

    % calculate distance to each peak
    dif = abs(prevLoc - locs);
    % find the closest peak
    [~, index] = min(dif);
    % return bin of closest peak
    curLoc = locs(index);
end
```

Listing A.11: discoverPeak

```
function [curLoc] = discoverPeak(SSRsig, prevLoc)
%   Searches full spectrum to find a peak

%   SSRsig      :   spectrum
%   prevLoc     :   frequency bin of previously located HR
%   curLoc      :   bin of current HR

    % full search range
    R3 = 40:200;
    [~,locs] = findPksInRange(SSRsig, R3);
    curLoc = findClosestPeak(prevLoc, locs);
end
```

A.3 SSR

Listing A.12: SSR

```
function [BPM, s] = SSR(y, Fs, N)
%   Returns the sparse spectrum and BPM-axis of time-signal y

%   y       :   time-signal
%   Fs      :   sampling rate
%   N       :   resolution
%   BPM     :   frequency-axis in BPM
%   s       :   band-passed sparse spectrum

    % if no resolution is given
    if (nargin == 2)
        % resolution is set to 1 BPM
        N = 60*Fs;
    end
    M = length(y);
    % construct fourier matrix
    Phi = zeros(M,N);
    complex_factor = 1i*2*pi/N;
    for m = 1:M
        for n = 1:N
            Phi(m,n) = exp(complex_factor*(m-1)*(n-1));
        end
    end
    % calculate BPM-axis
    BPM = 60*Fs/N*((1:N)-1);
    % construct sparse spectrum
    s = FOCUSS(y,Phi);

    % Band-pass frequencies in BPM
    BPM_bp = [40 200];
    % Band-pass spectrum
    s = BP(s, Fs, BPM_bp);
end
```

Listing A.13: FOCUSS

```
function s = FOCUSS(y,Phi)
%   Constructs a sparse spectrum of y using the FOCUSS algorithm

%   y       :   time-signal
%   Phi     :   Fourier matrix
%   s       :   sparse spectrum

    % number of iterations
    iterations = 2;
    % resolution of spectrum
    N = size(Phi,2);
    % initialization of x
    x = ones(N,1);
    for it = 1:iterations
        W_pk = diag(x);
        q_k = MPinverse(Phi*W_pk)*y;
        x = W_pk*q_k;
    end
    s = abs(x).^2;
end
```

Listing A.14: MPinverse

```
function A_plus = MPinverse(A)
%   MPinverse calculates the Moore-Penrose inverse of matrix A

%   A        : matrix
%   A_plus   : Moore-Penrose inverse of matrix A

    matrix = ctranspose(A);
    A_plus = matrix / (A * matrix);
end
```

Listing A.15: BP

```
function [ SSRsig ] = BP( SSRsig, Fs, BPM_bp )
%   Band-pass filters SSRsig

%   SSRsig  :   spectrum
%   Fs      :   sampling rate
%   BPM_bp  :   Band-pass frequencies
%   SSRsig  :   Band-passed spectrum

    N = length(SSRsig);
    f_lo = BPM_bp(1) / 60;
    f_hi = BPM_bp(2) / 60;
    R = floor(f_lo/Fs*N+1) : ceil(f_hi/Fs*N+1);
    H = zeros(N,1);
    H(R) = 1;
    SSRsig = SSRsig .* H;
end
```

Bibliography

[1] Zhilin Zhang, Zhouyue Pi, Benyuan Liu, ”TROIKA: A General Framework for Heart Rate Monitoring Using Wrist-Type Photoplethysmographic Signals During Intensive Physical Exercise”, IEEE Trans. on Biomedical Engineering, vol. 62, no. 2, pp. 522-531, February 2015

[2] Zhilin Zhang, ”Photoplethysmography-Based Heart Rate Monitoring in Physical Activities via Joint Sparse Spectrum Reconstruction”, IEEE Trans. on Biomedical Enigeering, vol. 62, no. 8, pp. 1902-1910, August 2015.

[3] I. F. Gorodnitsky and B. D. Rao, ”Sparse signal reconstruction from limited data using FOCUSS: a re-weighted minimum norm algorithm”, IEEE Trans. on Signal Processing, vol 45, no. 3, pp. 600-616, 1997.

[4] R. Yousefi, M. Nourani, S. Ostadabbas, I. Panahi, ”A Motion-Tolerant Adaptive Algorithm for Wearable Photoplethysmographic Biosensors”, IEEE Journal of Biomedical and Health Informatics, vol. 18, no. 2, pp. 670-681, 2014

[5] M. Raghu Ram, K. Venu Madhav, E. Hari Krishna, Nagarjuna Reddy Komalla, and K. Ashoka Reddy, ”A Novel Approach for Motion Artifact Reduction in PPG Signals Based on AS-LMS Adaptive Filter”, IEEE Trans on Instrumentation and Measurement vol. 61 , no. 5, pp. 1445-1457, 22 December 2011.

[6] Mathworks: kurtosis function, *http://nl.mathworks.com/help/stats/kurtosis.html.*

[7] Mathworks: dsp.LMSFilter System object, *http://nl.mathworks.com/help/dsp/ref/dsp.lmsfilter-class.html.*

[8] Lecture Slides: Recursive Least Squares Estimation, *http://www.cs.tut.fi/ tabus/course/ASP/Lecture-New10.pdf.*

[9] B.S. Kim, S.K. Yoo, ”Motion Artifact Reduction in Photoplethysmography Using Independent Component Analysis”, IEEE Trans. Biomedical Engineering, vol. 53, no. 3, pp. 566-568, 2006.

[10] Z. Zhang and B. D. Rao, ”Extension of SBL algorithms for the recovery of block sparse signals with intra-block correlation”, IEEE Trans. on Signal Processing, vol. 61, no. 8, pp. 2009-2015, 2013.

[11] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N. C. Yen, C. C. Tung, and H. H. Liu, The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis, Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, vol. 454, no. 1971, pp. 903995, 1998.

[12] PC specifications: Dell OptiPlex 9020, http://www.emc.com.br/PDF_Specs/DELL_OptiPlex_9020.pdf. CPU = i5-4690, Memory = 8 GB, Windows 7 Enterprise 64-bit.