CHALMERS

R-wave detection algorithms using adult and fetal ECG signals

Master's Thesis in Biomedical Engineering

IRIS ELFA SIGURDARDOTTIR

Department of Signals & Systems
Chalmers University of Technology
Gothenburg, Sweden 2013
Master Thesis EX044/2013

Abstract

Monitoring of the fetal heart rate during pregnancy and labor gives experienced clinicians information about the physiological condition of the fetus. The heart rate is calculated from the heartbeat interval and is updated for each heartbeat. Therefore, an accurate and reliable algorithm for R-wave detection is crucial. R-wave detection is constantly improving and therefore it is important for Neoventa to compare the performance of new algorithms to the one currently implemented in fetal monitor STAN S31.

The aim of this project is to implement various algorithms and validate their performance using adult and fetal ECG signals.

Within the current project, three different published algorithms were implemented, validated and compared to the current algorithm in STAN S31. The result indicate that the heartbeat detection performance in STAN S31 could be improved by replacing the existing algorithms with a non-linear method previously published by Pan and Tompkins.

Key words: fetal monitoring, electrocardiogram (ECG), fetal heart rate (FHR), R-wave

Preface

This project is a result from a master's thesis for a degree Biomedical Engineering at Chalmers Technical University, Göteborg, Sweden. The project was carried out at Neoventa Medical, Sweden, from February 2013 to September 2013.

Nils Löfgren was the supervisor at Neoventa Medical and he provided guidance and input throughout the project, The examiner at the department of Signals and Systems at Chalmers Technical University was Assistant Professor Sabine Reinfeldt.

Göteborg September 2013,
Iris Sigurdardottir

Contents

1 Introduction **1**
- 1.1 Objective . . . 2
- 1.2 Delimitations . . . 2

2 Method **3**
- 2.1 Literature Studies . . . 3
- 2.2 Validation of Algorithms . . . 4
  - 2.2.1 Relation between frequency contents for adult and fetal ECG signal 5
  - 2.2.2 Adult ECG Signal . . . 7
  - 2.2.3 Fetal ECG Signal . . . 8
- 2.3 Implementation of Algorithm 1 . . . 12
  - 2.3.1 Preprocessing . . . 13
  - 2.3.2 Event and R-wave detection . . . 15
- 2.4 Implementation of Algorithm 2 . . . 15
  - 2.4.1 Preprocessing . . . 15
  - 2.4.2 Event and R-wave detection . . . 18
- 2.5 Implementation of Algorithm 3 . . . 18
  - 2.5.1 Preprocessing . . . 18
  - 2.5.2 Event and R-wave detection . . . 20
- 2.6 Algorithm implemented in STAN S31 . . . 22
  - 2.6.1 Preprocessing . . . 22
  - 2.6.2 Event and R-wave detection . . . 22

3 Result **25**
- 3.1 Adult ECG signal . . . 25
- 3.2 Fetal ECG signal . . . 27

4 Discussion **29**
- 4.1 Obstacles in the project . . . 29
- 4.2 Results . . . 29

5 Conclusion **31**

Bibliography **32**

A Appendix: Algorithm 1 **33**

B Appendix: Algorithm 2 **39**

C Appendix: Algorithm 3 **42**

1

Introduction

Cardiotocography (CTG) refers to the fetal heart rate (FHR) and uterine contraction monitoring during labor. The heart rate monitoring during late pregnancy and labor provides the experienced clinician information about the physiological condition of the fetus that are needed to identify hypoxia which can lead to permanent brain damage or even death [1].

Fetal electrocardiogram (FECG) is used when determining the FHR. Figure 1.1 shows two cycles in the ECG. Each QRS complex refers to one heartbeat and to find the heart rate, the RR-interval is calculated for each heart beat so the HR is updated for each beat. From this information the HR in beats per minutes (bmp) is calculated [1].

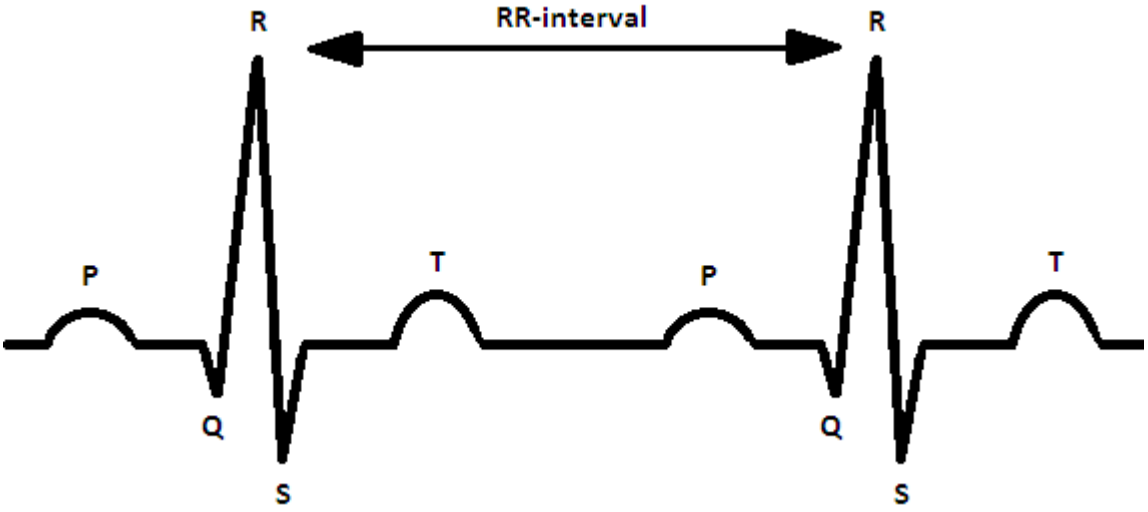

Figure 1.1: Electrocardiogram of one heart beat

Neoventa Medical manufactures STAN S31 that is a system used for fetal monitoring. The system combines CTG and ST-analysis of the FECG. When hypoxia related abnormalities in the ST segment occurs, the system sends an alarm [1].

STAN S31 uses two methods for a heart rate measurement. Ultrasound transducers are used on the mothers belly before the membranes rupture, and after rupture an electrode is placed on the fetus scalp, to record the FECG [1].

The methods for R-peak detection from ECG signals are constantly improving and it is important for Neoventa to compare the performance of new algorithms with the one currently implemented in fetal monitor STAN S31.

1.1 Objective

The aim of this project is to investigate different algorithms for R- peak detection and implement and validate suitable algorithms for FHR measurements. The project is done in five steps which are the following:

- Research different R-peak detection algorithms
- Implement algorithms in Matlab
- Decide the criteria to validate the algorithms
- Validate the algorithms by using Massachusetts Institute of Technology/Beth Israel Deaconess Medical Center (MIT/BIH) database and clinical data from Neoventa
- Implement the most suitable algorithm in C#

1.2 Delimitations

Many different algorithms have been investigated for R-peak detection and out of them six were chosen for possible implementation, and they are all fundamentally different. Three of these algorithms were chosen from a review paper by Köhler [2]. The paper summarizes the performance of different algorithms when using adult ECG signal. The other three algorithms were found when searching databases. All six algorithms seemed promising in a way that all of them had high sensitivity and positive predictive value.

2

Method

2.1 Literature Studies

Literature studies was performed to find suitable methods to implement in Matlab. Two database were used, IEEE Xplore digital library and Springer link. In addition some articles were provided by Neoventa. The search words used were ECG detection, QRS detection, ECG Pan, ECG Afonso, ECG triangle, adaptive filter, R-wave detection, filter-banks.

Table 2.1 shows the articles used as support for each implementation, the theory they are based on and their authors.

Table 2.1: Algorithms that were chosen for a possible implementation

Name of the Paper	Method	Author	Algorithm Number
QRS Detection Using Zero Cross Count	Zero cross count	Kohler	1
A Real Time QRS Detection Algorithm	Filters and window integration	Pan and Tompkins	2
ECG Beat Detection Using Filter Bank	Filter banks	Afonso and Tompkins	3
DSP implementation of wavelet transform for real time ECG wave forms detection and heart rate analysis	Wavelet transform	Bahoura	x
A new approach of QRS complex detection based on matched filtering and triangle character analysis	Triangle characteristics	Li and Yan	x
Superiority Analysis of MLMS over Adaptive Filtering Methods for Hearth Arrhythmias Detection	Adaptive filter	Khan and Billal	x

When choosing the suitable algorithms for implementation there are two things that need to be kept in mind:

1. Fundamentally different methods
2. Simple and easy implementation

When looking theoretically at an ECG signal it should be easy to distinguish the P-wave, the QRS complex and the T-wave but that is not always the case in reality. Therefore an R-wave detection algorithm has to be simple, robust and be able to distinguish the R-wave when using various ECG signals.

2.2 Validation of Algorithms

For the validation of the algorithms the Massachusetts Institute of Technology/Beth Israel Deaconess Medical Center (MIT/BIH) database was used for adults ECG signals and records from Neoventa for a fetal ECG signals. To compare the performance and accuracy of the algorithms, the sensitivity (Se) and positive predictive value (+P) were calculated for all algorithms and for both adult and fetal ECG signal, see equations 2.1 and 2.2.

$$Se = \frac{TP}{TP + FN} \tag{2.1}$$

$$+P = \frac{TP}{TP + FP} \tag{2.2}$$

where TP is the true positive, FN is the false negative and FP is the false positive.

2.2.1 Relation between frequency contents for adult and fetal ECG signal

Since all three algorithms were designed for adult ECG signals, they had to be adjusted to the fetal ECG signals. The frequency components for the QRS complex are different for adult and fetal ECG signal and therefore the relation between the frequency contents has to be explored. This can be done by looking at the QRS duration for both adult and fetal ECG signal and the QRS frequencies are directly proportional to the QRS duration. The QRS duration for the adult signal can not be over 120 ms [3] and for the fetal ECG signal it is maximum 80 ms [4]. This gives the relation of the QRS duration:

$$\frac{QRS_{fetal}}{QRS_{adult}} = \frac{80ms}{120ms} = 0.67 \tag{2.3}$$

The frequency ratio is therefore:

$$\frac{f_{fetal}}{f_{adult}} = \frac{1}{0.67} = 1.5. \tag{2.4}$$

This means that the bandpass filter from algorithm 1 which is designed to have frequency range of 18-35 Hz for the adult ECG signal, should be re-designed to have the frequency range of 27-53 Hz for the fetal ECG signal. When looking at the frequency response of the filter in algorithm 1, the upper and lower cut-off frequencies can be found by looking at -3 dB.

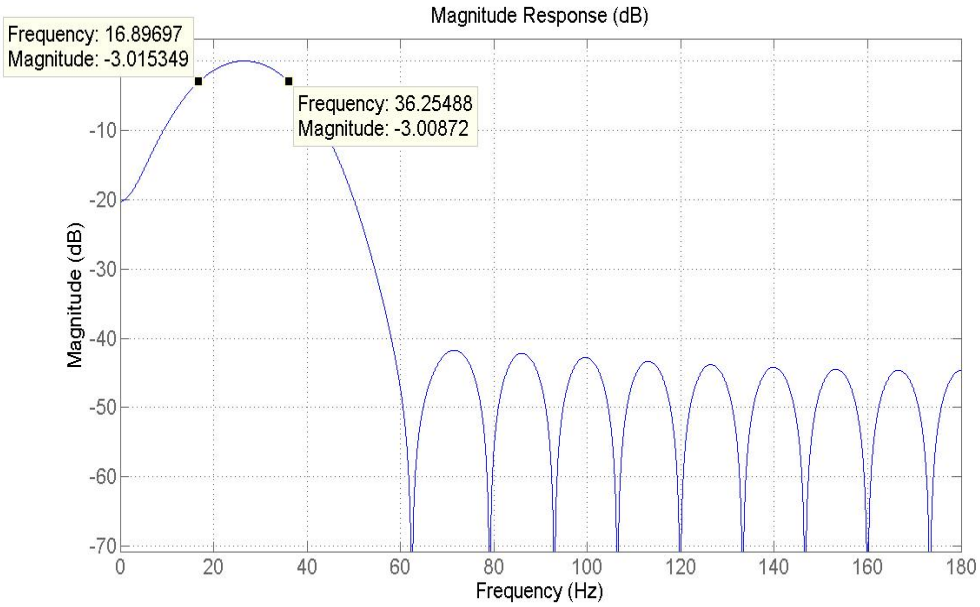

Figure 2.1: Frequency response and the cut-off frequencies for the adult ECG signal

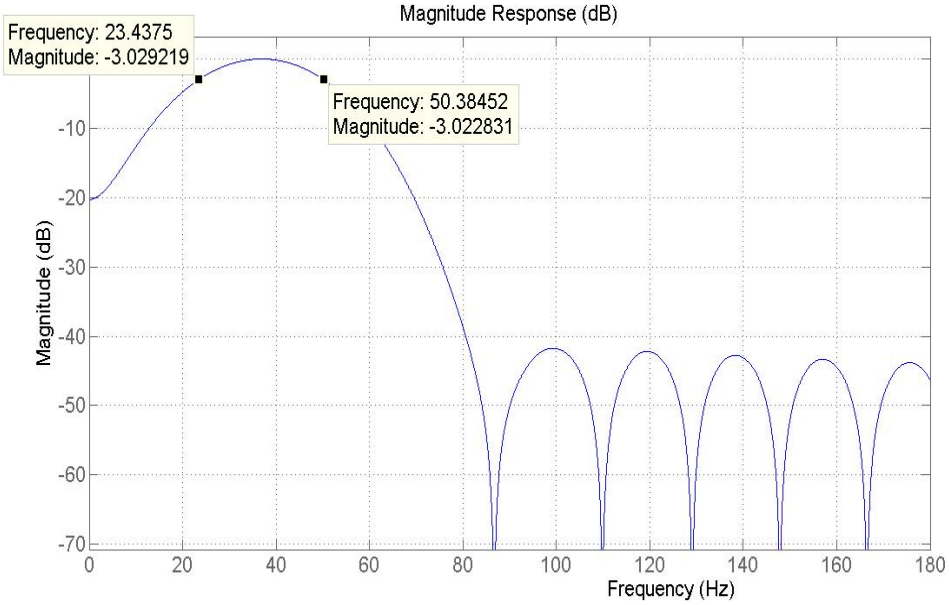

Figure 2.2: Frequency response and the cut-off frequencies for the fetal ECG signal

From figures 2.1 and 2.2 it can be seen that the lower cut-off frequency for the adult ECG signal is 17 Hz and 23 Hz for the fetal ECG signal and the upper cut-off frequency for the adult ECG signal is 36 Hz and 50 Hz for the fetal ECG signal. Calculating the

frequency ratio for both cases results in ratio of approximately 1.4. This means that is it unnecessary to change the filters since changing the sampling rate from 360 Hz to 500 Hz will be sufficient to attain correct cut-off frequencies for the fetal ECG signal.

2.2.2 Adult ECG Signal

The MIT/BIH database contains 30 minutes long records from 48 adult patients which were sampled at 360 Hz. The first 23 recordings contain randomly chosen signals and the other 25 recordings have been chosen from patients with various arrhythmia [5]. Since the dataset contains different variations of ECG signals with known location of the R-wave, the result will be accurate and they will show how robust and stable the algorithms are.

Validation: Each recording contains the ECG signal and the location and amplitude of the R-wave. It is therefore easy to calculate sensitivity and positive predictive values by comparing the detected values from the algorithms to the given values in each dataset, see figure 2.3. The detected values that were within ±20 ms from the given values in the dataset were classified as a true positive (TP), the rest of the detected values were classified as false positive (FP) and the rest of the given values from the dataset were classified as false negative (FN).

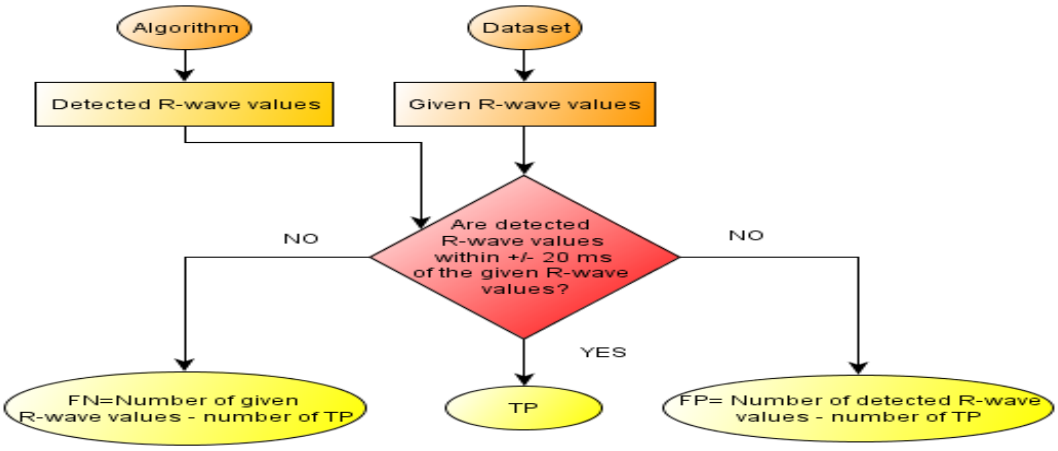

Figure 2.3: Flow chart of the validation for the adult ECG signal

For each algorithm the focus was to look at the overall performance so the sensitivity and positive predictive value were calculated from all 48 records in stead of calculating the values for each record. The threshold value was changed 5 times for each algorithm to see the effect on the sensitivity vs. positive predictive value.

2.2.3 Fetal ECG Signal

The data from Neoventa contains 30 min records obtained during birth of 82 children This data contains the ECG signal and the R-wave detection from STAN S31, but the correct location of the R-wave are unknown and therefore another method has to be applied when estimating number of TP, FN and FP. Figure 2.4 shows a flowchart of the method used for the estimation. The detection from STAN S31 are also evaluated with the detections from the three algorithms.

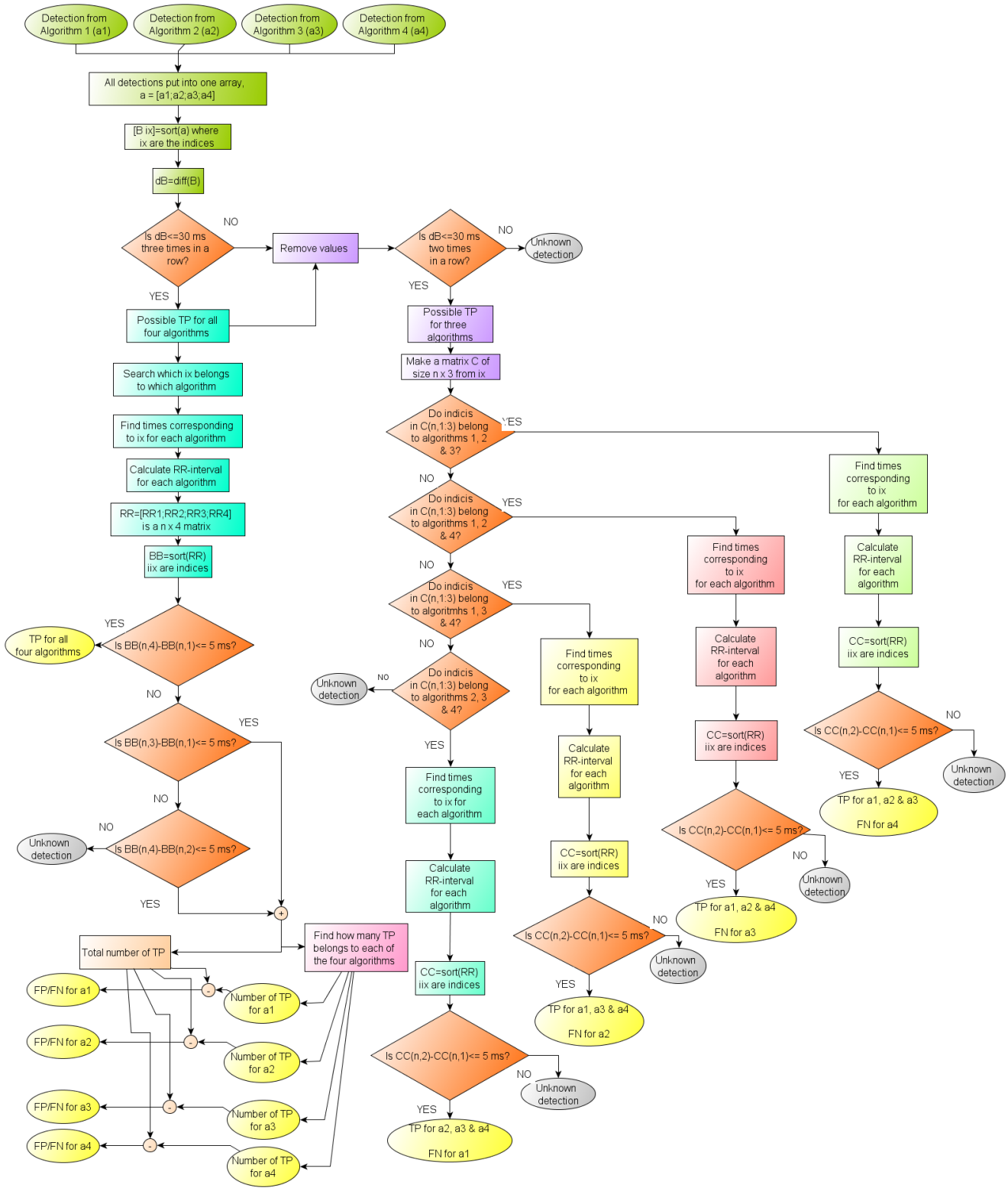

Figure 2.4: Flow chart of the validation for the fetal ECG signal

Validation: First step was to put all detections from all four algorithms into array a. Array B contains array a, which has been sorted from lowest to highest value. From there the differences between each values in B were calculated and the validation process was split into two groups depending if there were four or three detection within 30 ms. See figure 2.5 for an example of the first step in the validation.

```
Detections from the four algorithms:
A1 = [4372 4760 5146 5534];
A2 = [4370 4758 5144 5530];
A3 = [4372 4760 5146 5535];
A4 = [4371 4757 5143 5529];

All detections put into one vector
A=[4372 4760 5146 5534 4370 4758 5144 5530 4372 4760 5146 5535 4371 4757 5143 5529];

Vector A is sorted after size, from lowest to highest
B=[4370 4371 4372 4372 4757 4758 4760 4760 5143 5144 5146 5146 5529 5530 5534 5538];

Difference between values in B calculated
dB= [1 1 0 385  1 2 0 383 1 2 0 383 1 4 4];
```

Figure 2.5: Example of step 1 in the validation process for fetal ECG

Four detections: If four detections were found within 30 ms, a back search was done to find which time belonged to which algorithm and the RR-interval was calculated. At that point there are four RR-interval arrays of same length n, one for each algorithm. A new matrix RR, of size n x 4, was created using one RR-interval from each algorithm. Each row is sorted from lowest to highest value and if the difference between the first and the fourth value was less than 5 ms then the assumption was made that those detection were TP for all four algorithms. See figure 2.6 for an example.

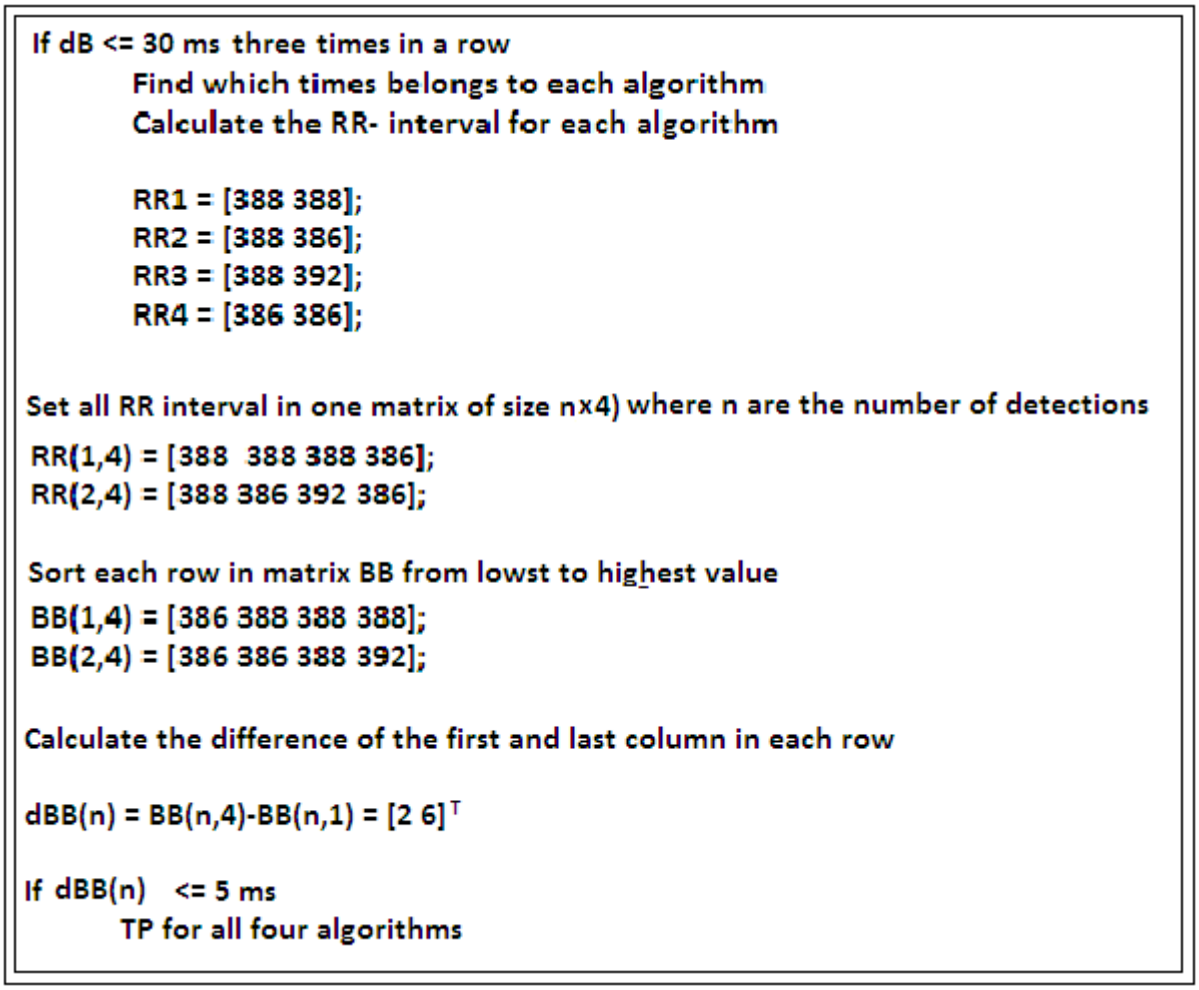

Figure 2.6: Example of step 2 in the validation process for fetal ECG

If the difference was more than 5 ms then it was not a TP for all four algorithms but if the difference between the first and the third or the second and fourth value was less than 5 ms than it was a TP for those three algorithms belonging to those times and a FP and FN for the fourth algorithm. Finally a back search was done to find out how many TP, FN, FP each of the algorithm had. See figure 2.7 for an example.

```
If dBB > 5 ms
        calculate the difference of the first and third column in each row
        If BB(n,3)-BB(n,1) <= 5 ms
                then it is TP for the three algorithms that those
                detections belong to
         If BB(n-4)-BB(n,2) <=5 ms
                then it is TP for the three algorithms that those detections belong to
```

Figure 2.7: Example of step 3 in the validation process for fetal ECG

Three detections: If three detections were found within 30 ms, a back search was done to find out which time belonged to which algorithms. From there four different groups were made depending on which three algorithms had those detections. The same logic was used as in steps 1-2 in figures 2.5 and 2.6 for the RR-interval like when there were four detections but instead the RR matrix was of size n x 3.

The total number of TP, FN, FP and unknown detections were summed up. For all extra detections the classification was unknown detections and they were eliminated when calculating Se and +P.

2.3 Implementation of Algorithm 1

Algorithm 1 uses the zero crossing count method which is based on the signal constantly crossing over the threshold when it changes signs and in this case the R-wave can be located where the signal decreases its crossing. The block diagram of algorithm 1 can be seen in figure 2.8 [6].

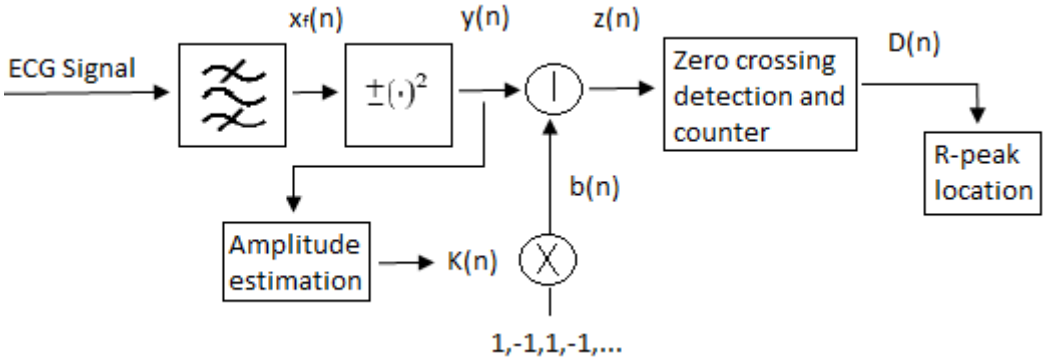

Figure 2.8: Block diagram for algorithm 1

2.3.1 Preprocessing

The preprocessing steps involve using different filters and then the signal is nonlinearly transformed. The amplitude is estimated and a high frequency sequence is added to the signal and finally the zero crossing is detected and counted. Figure 2.9 shows how the ECG signal looks after each preprocessing steps and table 2.2 shows the design parameters for both adult and fetal ECG signal. Different threshold values were chosen for the adult ECG signal to see if and then how it would affect the performance of the algorithm.

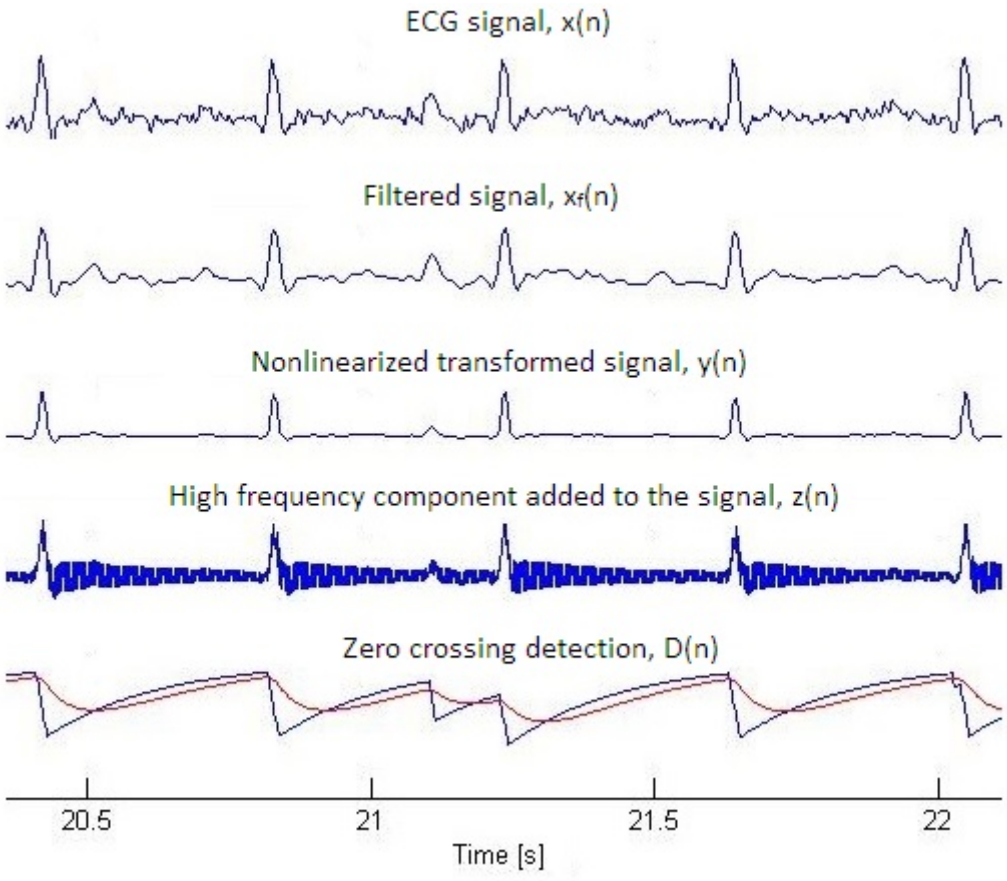

Figure 2.9: Preprocessing steps for algorithm 1

Linear and nonlinear filters: The signal was first filtered with a 27 tap linear phase finite impulse response (FIR) bandpass filter with cut-off frequencies at 18 Hz and 35 Hz for the adult signal and 25 Hz and 49 Hz for the fetal signal. By doing this, the signal to noise ratio was increased, but for even better signal quality the signal was nonlinearly

Table 2.2: Design parameters for algorithm 1

Design Parameters	Adult ECG Signal	Fetal ECG Signal
Filter frequencies	18-35 Hz	35-49 Hz
λ_K	0.99	0.99
Gain c	4	4
λ_D	0.99	0.99
λ_Θ	0.99	0.97
p	0.94; 0.96; 0.98; 1; 1.02	1

transformed. Equation 2.5 shows the nonlinear transformed signal where $x_f(n)$ is the filtrated signal [6].

$$y(n) = sign(x_f(n)) \cdot x_f^2(n) \tag{2.5}$$

High frequency sequence and amplitude estimation: A high frequency sequence was added to the signal since the bandpass filter reduces the high frequency components. Equation 2.6 shows the high frequency sequence and equation 2.7 shows the signal after adding the sequence to the nonlinear transformed signal [6].

$$b(n) = (-1)^n \cdot K(n) \tag{2.6}$$

$$z(n) = y(n) + b(n) \tag{2.7}$$

This was done to increase the number of zeros for the non QRS components. The amplitude estimation is calculated from equation 2.8 where K(n) is the amplitude. The value for the amplitude can not be too small since that will give noisy signal, and the difference between the QRS and non QRS complex is too small for classification. On the other hand, if the amplitude is too large, the number of zero crossing will be the same for both QRS and non QRS complex. Ideally the values of the signal, $D(n)$, should be equal to the number of zero crossing during non QRS complex and less than the number of zero crossing during QRS complex. Equation 2.8 shows $K(n)$ where $\lambda_K \in (0; 1)$ is the forgetting factor and c is the constant gain [6].

$$K(n) = \lambda_K K(n-1) + (1 - \lambda_K)|y(n)| \cdot c \tag{2.8}$$

Detection and counting of zero crossing: For the zero crossing detection, equation 2.9 and 2.10 were used [6].

$$d(n) = \left| \frac{sign[z(n)] - sign[z(n-1)]}{2} \right| \tag{2.9}$$

$$D(n) = \lambda_D D(n-1) + (1-\lambda_D)d(n) \tag{2.10}$$

Where $\lambda_D \in (0;1)$ is the forgetting factor [6].

2.3.2 Event and R-wave detection

To detect events an adaptive filter was implemented, using the featured signal. Equation 2.11 shows the threshold, where $\lambda_\Theta \in (0;1)$ is the forgetting factor. [6]

$$\Theta(n) = \lambda_\Theta(n-1) + (1-\lambda_\Theta)D(n) \tag{2.11}$$

The featured signal, D(n), was compared to the threshold, λ_Θ, to create events. One event takes place during period when $\lambda_\Theta \cdot p > D(n)$, where p is a weighing factor, and therefore each event has a lower and an upper limit. If two event are within 83 ms for adult signal and 60 ms for the fetal signal the two events are combined in one using the lower limit from the first event and the upper limit from the second event [6].

For each event the minimum and the maximum were found from the magnitude of the nonlinear transformed signal. If the minimum is much larger than the maximum then the R-wave is set to location of the minimum. Otherwise the R-wave is at the location of the maximum [6].

2.4 Implementation of Algorithm 2

Algorithm 2 uses different filters, squaring function and a moving window to bring out the features in the ECG signal and then the R-wave is located. Figure 2.10 shows the block diagram of algorithm 2.

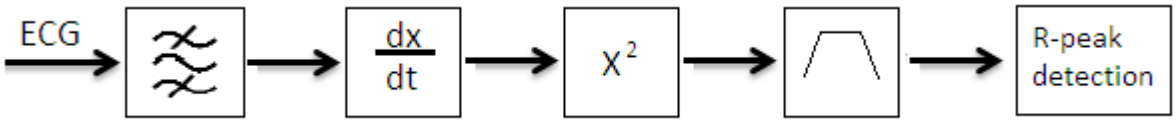

Figure 2.10: Block diagram of algorithm 2

2.4.1 Preprocessing

The preprocessing steps are bandpass filter and differentiation, then the signal is squared and finally a moving window integration is applied. Figure 2.11 shows the ECG signal after each preprocessing steps and table 2.3 shows the design parameters for both the adult and fetal ECG signal. Different threshold values were chosen for the adult ECG signal to see if and then how it would effect the performance of the algorithm.

Table 2.3: Values for algorithm 2

Design Parameter	Adult ECG Signal	Fetal ECG Signal
Filter bandwidth	5-10 Hz	7-14 Hz
Window	54	50
p	0.8, 1, 1.2, 3 and 5	1

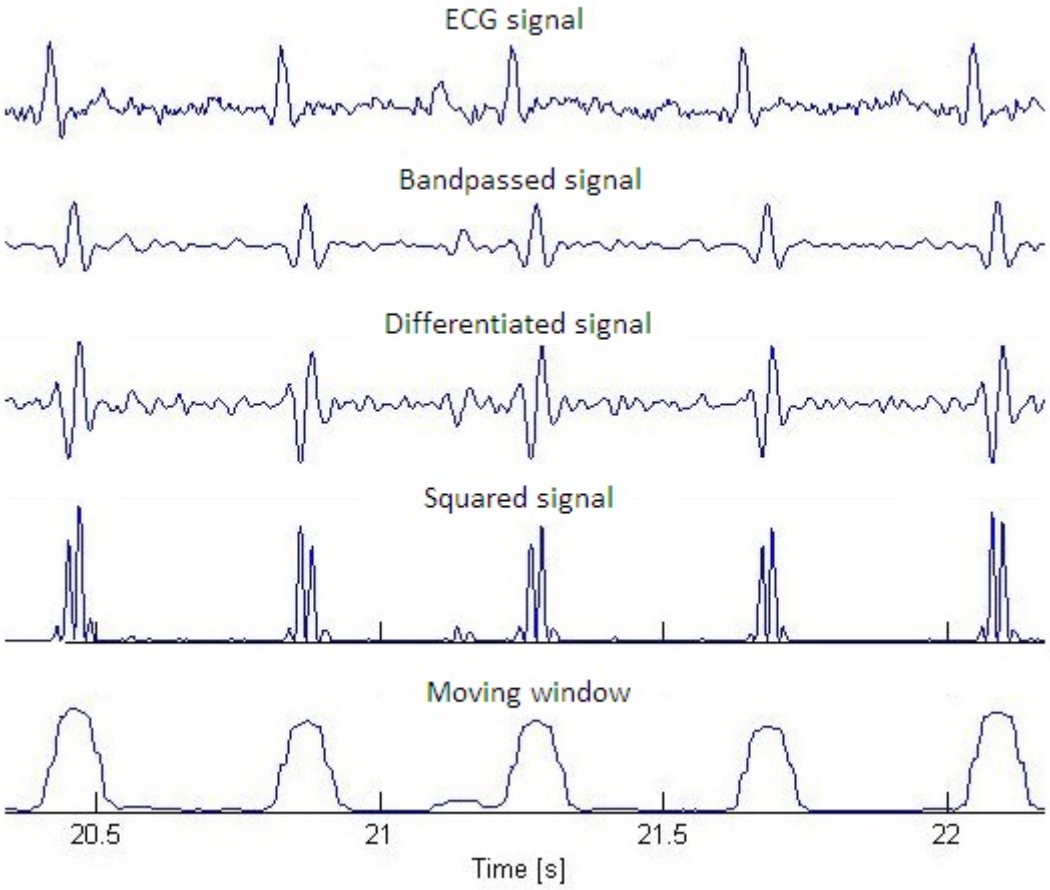

Figure 2.11: Preprocessing steps for algorithm 2

Bandpass filter: The implemented bandpass filter is composed of a low pass and a high pass filter, which are designed to reduce noise from muscles, the power line interference, baseline wander and T-wave interference. Equations 2.12 and 2.13 show the transfer function and the difference equation for the low pass filter. The low pass filter has a gain of 32 dB, a cutoff frequency around 10 Hz for adult ECG signal and 14 Hz for the fetal ECG signal and a total delay of 5 samples [7].

$$H(z) = \frac{(1 - z^{-6})^2}{(1 - z^{-1})^2} \tag{2.12}$$

$$y(n) = 2y(n-1) - y(n-2) + x(n) - 2x(n-6) + x(n-12) \tag{2.13}$$

The high pass filter was obtained by dividing the low pass filter from equation 2.12 with its gain and then subtract it from the all pass filter $H_{all}(z) = z^{-16}$. Equations 2.14 and 2.15 show the transfer function and the difference equation for the high pass filter. The high pass filter has a gain of 32, a cutoff frequency of 5 Hz for the adult ECG signal and 7 Hz for the fetal ECG signal and a total delay of 16 samples [7].

$$H(z) = 32z^{-16} - \frac{1 - z^{-32}}{1 - z^{-1}} = \frac{-1 + 32z^{-16} - 32z^{-17} + z^{-32}}{1 + z^{-1}} \tag{2.14}$$

$$y(n) = -y(x-1) - x(n) + 32x(n-16) - 32x(n-17) + x(n-32) \tag{2.15}$$

Derivative: After the signal had been filtered it was differentiated to get information about the QRS complex slope. A five-point derivative was used where equations 2.16 and 2.17 show the transfer function and the difference equation. This gave a delay of two samples [7].

$$H(z) = \frac{-z^{-2} - 2z^{-1} + 2z^{1} + z^{2}}{8} \tag{2.16}$$

$$y(n) = \frac{-x(n-2) - 2x(n-1) + 2x(n+1) + x(n+2)}{8} \tag{2.17}$$

Squaring function: The signal was squared point by point using equation 2.18

$$y(n) = [x(n)]^2 \tag{2.18}$$

where x(n) is the derivate signal. This makes all points positive and the signal is non-linearly amplified which emphasizes the higher frequencies [7].

Moving window integration: A moving window was implemented.

$$y(n) = \frac{x(n-(N-1)) + x(n-(N-2)) + ... + x(n)}{N} \tag{2.19}$$

Where N is the width of the integration window, and should be approximately the same as the duration of the QRS complex which in this article was chosen to be 150 ms [7].

2.4.2 Event and R-wave detection

For the R-wave detection a new approach was used [8] since the R-wave detection logic in article [7] was too complicated for implementation.

A threshold was calculated from equation 2.20, where p is a weighing factor. Events were located where the output of the moving window was higher than the threshold. The lower and upper limit of each event were located and to find the R-wave the delay of the bandpass filter had to be taken into consideration. For each event the maximum was found and the location of it set as the R-wave [8]

$$threshold = p \cdot max(y(n)) \cdot mean(y(n)) \tag{2.20}$$

where y(n) is the output of the moving window integration.

2.5 Implementation of Algorithm 3

Algorithm 3 uses filter banks which are used to divide the frequency range into sub bands and then the signal is processed for all the sub bands. Figure 2.12 shows the block diagram for algorithm 3.

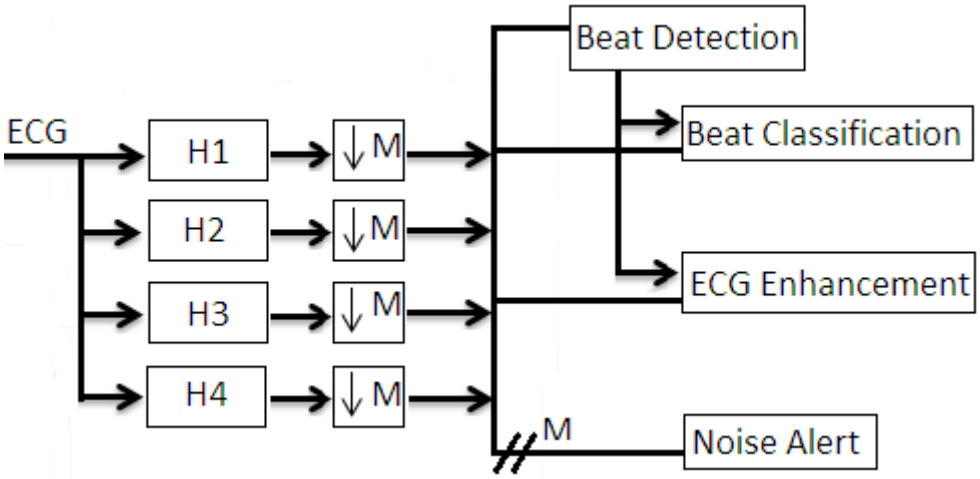

Figure 2.12: Block diagram of algorithm 3

2.5.1 Preprocessing

Four FIR analysis filters with a length of 200 and a bandwidth of 5.6 Hz for the adult ECG signal and 7.9 Hz for the fetal ECG signal. The signal was first filtered and then down sampled by 32 for the adult ECG signal and by 44 for the fetal ECG signal. Table 2.4 shows the values chosen for some design parameters. Equation 2.21 shows the

down sampled signal $W_l(z)$, where $U_l(z)$ is the sub band signal, $H_l(z)$ is the analysis filters, and $X(z)$ is the input signal and M is the down samples rate [9]. Figure 2.13 shows the four filters used in the preprocessing step.

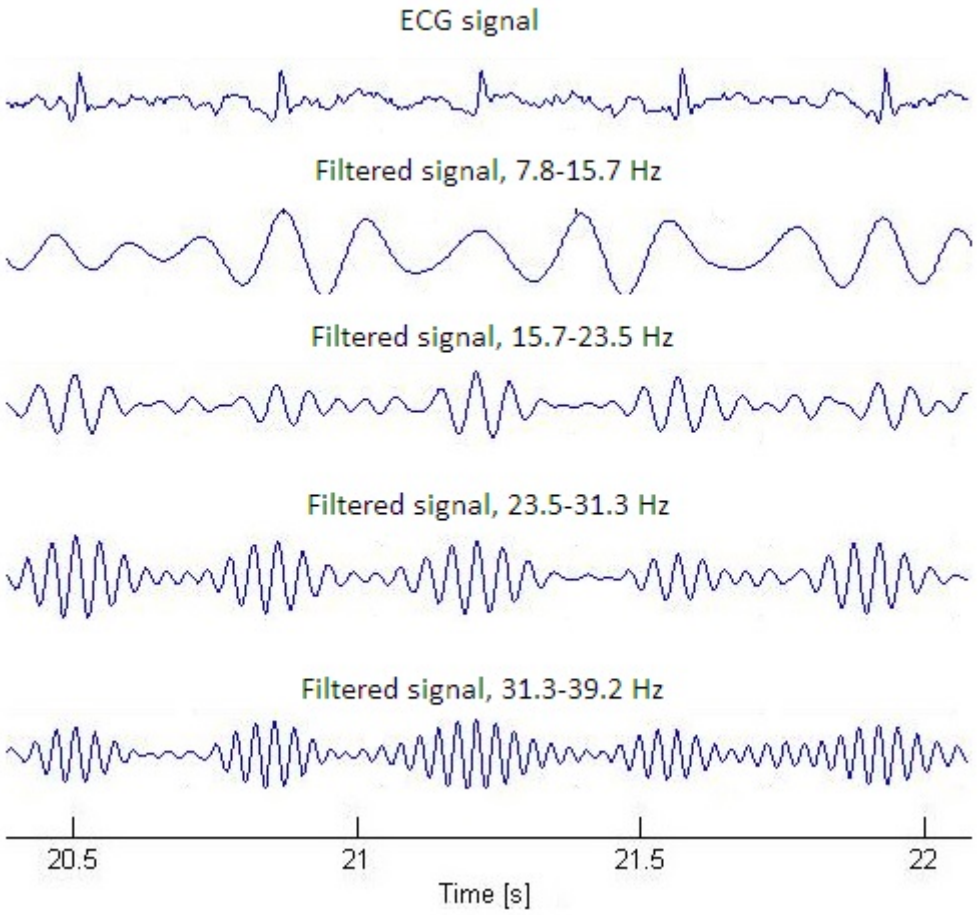

Figure 2.13: Preprocessing steps for algorithm 3

Table 2.4: Design Parameters for algorithm 3

Design Parameters	Adult ECG Signal	Fetal ECG Signal
Filter length	400	400
Down sample rate	32	44
Filter frequency range	5.6-28 Hz	7.9-39 Hz
threshold 1 / threshold 2	0.2/0.1; 0.7/0.3; 0.9/0.5; 1.2/0.9; 1.7/1.5	0.7/0.3

$$W_l(z) = \frac{1}{M}\sum_{k=0}^{M-1} U_l(z^{1/M}W^k) = \frac{1}{M}\sum_{k=0}^{M-1} H_l(z^{1/M}W^k)X(z^{1/M}W^k), \quad l = 0,1,...,M-1 \tag{2.21}$$

Sub bands were combined to create features, P_x, with a certain energy relating to the QRS complex. Equations 2.22, 2.23 and 2.24 show how the three different features were calculated. P_1 has a frequency band of 5.6-22.4 Hz for the adult ECG signal and 7.9- 31 Hz for the fetal ECG signal, P_2 a frequency band of 5.6-28 Hz for the adult ECG signal and 7.9- 39 Hz for the fetal ECG signal and finally P_3 has a frequency band of 11.2-28 Hz for the adult ECG signal and 15.7-39.2 Hz for the fetal ECG signal [9].

$$P_1 = \sum_{l=1}^{3} |W_l(z)| \tag{2.22}$$

$$P_2 = \sum_{l=1}^{4} |W_l(z)| \tag{2.23}$$

$$P_3 = \sum_{l=2}^{4} |W_l(z)| \tag{2.24}$$

These features were the input to a moving window integration (MWI) where two samples were averaged at the sample rate [9].

The detection strength, D_s was calculated to determine if a peak was an R-wave or just noise. Equation 2.25 shows how the detection strength was calculated, where P_x is the incoming feature, S_L is the signal level and N_L is the noise level [9] [10].

$$D_s = \frac{P_x - N_L}{S_L - N_L} \tag{2.25}$$

D_s is set to zero if the features value is less than N_L and to one if the value is higher than S_L. When the detection strength is higher than a certain threshold it is classified as a peak and the history of the signal is updated for the feature value. If the detection strength is less than the threshold, it is classified as noise and the noise history is updated [9] [10].

2.5.2 Event and R-wave detection

The event detection was divided into six levels to maximize the true positives (TP) and minimize the false negatives (FN) and false positives (FP) [9].

Level 1: This level detects all peaks in the output of the MWI for feature P_1. This level has no threshold and therefore it detects most of the true beats but it has high number of FP. Level 1 is the event detection and triggers further logic to reduce FP [9].

Level 2: Level 2 is triggered when there is a peak in level 1 and it uses two channels ($Chan_1$ and $Chan_2$) that operate simultaneously. The output of the MWI for P_2 is used in both channels but $Chan_1$ has a threshold $T_1 = 0.08$ and $Chan_2$ has a threshold $T_2 = 0.70$. When level 2 is triggered the channels calculate the detection strength and compare it to the thresholds. When D_s is higher than the threshold it is classified as an R-wave and the history of the R-wave is updated. If D_s is lower than the threshold it is classified as noise and the noise history is updated. Since $Chan_1$ has a low threshold it will classify some noise as R-wave but the R-wave will be classified correctly. $Chan_2$ has a higher threshold and therefore some R-wave will be classified as noise but the noise will be classified correctly. In other words, $Chan_1$ will have many FP but few FN and $Chan_2$ will have few FP but many FN [9] [10].

Level 3: This level uses the information in level 2 to classify what is a beat and what is noise. Level 3 uses if-then-else rules for the classification. These rules give four possible outcomes, if $Chan_1$ and $Chan_2$ classify an event as an R-wave then level 3 classifies it as a beat. If neither $Chan_1$ or $Chan_2$ classify an event as a beat then level 3 classifies it as noise. Since $Chan_2$ has higher threshold and few FP, it's detection is accurate, and if there is a R-wave detection in $Chan_2$ and not in $Chan_1$ it is classified as a beat. If $Chan_1$ classifies a peak as a beat and not $Chan_2$ the normalized detection strength $\Delta_i i = 1{,}2$ indicate which detection is more likely to be a beat. The logic for level 3 is the following [9] [10]

$Chan_1$	$\surd$	x	x	$\surd$
$Chan_2$	$\surd$	x	$\surd$	x
Outcome	$\surd$	x	$\surd$	$\Delta_1?\Delta_2$

where

$\Delta_1?\Delta_2$: if $\Delta_1 > \Delta_2$ then $\surd$, else x

$\Delta_1 = (D_{S1} - T_1)/(1 - T_1)$

$\Delta_2 = (T_2 - D_{S2})/T_2$

$\surd$ is a beat and x is not a beat

Level 4: Level 4 uses feature P_3 as a MWI input. This level updates the history of beats detected in level 3 and re-evaluates the noise from level 3. All noise peaks from level 3 are compared to threshold $T_4 = 0.30$ and if their detection strength is greater than the threshold their classification is changed to a beat and the noise history is updated. This level reduces the FN and is more accurate than levels 1-3 [9] [10].

Level 5: Level 5 looks at the time between beats. If the time is longer than $1.5 \cdot$ mean of the beat distance, the algorithm does a search back to find any missed beats. If a new

beat is found it's detection strength is compared to a threshold, $T_5 = 0.2$, if it is higher then the beat and noise history are updated [9] [10].

Level 6: This level eliminates beats that are too close together. If their distance is less than 250 ms the one with lower amplitude is eliminated and the beat history is updated.

2.6 Algorithm implemented in STAN S31

The algorithm implemented in STAN S31 uses two input signals; a scalp electrode to scalp reference lead (FHR channel) and a scalp electrode to skin electrode lead (ECG channel). The preprocessing step for both these signals include filters and the R-wave detection uses template matching [11].

2.6.1 Preprocessing

In the preprocessing step a filter is applied to both inputs signal to remove unwanted frequencies not belonging to the QRS complex [11].

2.6.2 Event and R-wave detection

For the R-wave detection a template selection is used on both the FHR and ECG channel and the events from those are compared to each other to find which selection are true R-waves [11].

Template selection: Two templates are used, one for each channel. The templates are 50 ms wide and they reflect the shape of the QRS complex.The FHR channel is more reliable then the ECG channel and therefore the template search is initiated at that channel. From the FHR channel a template is selected and only when a FHR template is selected the ECG template search begins [11].

FHR template selection: When selecting a template, a 2000 ms interval is searched to make sure that the correct template is detected. The algorithm uses three criteria when searching for the highest peak:

1. The amplitude has to be higher than 50 μV
2. The amplitude has to be lower than 2000 μV
3. No another peak within 250 ms that has half the amplitude or higher. This is done to reduce the number of templates which are picked up by noise [11]

If a peak is found it is classified as a beat. Since the template search interval is 2000 ms there might be more than one peak inside the interval but the algorithm is only interested in the latest peak. The template search is repeated in the interval, 200 ms after the first peak detection to the end of the 2000 ms interval. If there is a peak within

the later interval that has an amplitude of at least half the initially detected peak, the new peak is selected as a beat [11].

ECG template selection: When at least four continuous beats are found when using the FHR template and they all match in shape and amplitude, a mean value of those template wide section is calculated and defined as a ECG template. The ECG templates are picked up at the location of the FHR beat positions with no regards to where the highest amplitude is located in the ECG channel [11].

Comparison to the ECG signal: When a template have been chosen, it is constantly compared to the signal and yields in a three different DIFF signals [11]

1. FHR template compared to the FHR channel (DIFF FHR)
2. ECG template compared to the ECG channel (DIFF ECG)
3. Combination of DIFF FHR and DIFF ECG (DIFF COMBINED) [11].

Both DIFF FHR and DIFF ECG are calculated in the same way:

1. If a template is not selected the DIFF value will be set to INT MAX so it won't generate any heart beats.
2. The comparison between template and signal is made sensitive to difference in power and offset, this is done to increase the precision. To normalize against difference in offset, the average of the signal and template is calculated and then subtracted.
3. The power is calculated for both the template and the signal, as the sum of the absolute values. If the difference between the power of the signal and the power of the template is less than 50% or higher than 200% then the DIFF is set to INT MAX to reject it. If the difference is within the range, the signal is made insensitive to difference in power by re-scaling it by template power/signal power.
4. From there the diff value is calculated as the sum of squared differences between template and signal, divided by the square of power of the template. This summing is done over the template width. By dividing the sum with the power of the template the DIFF value is made indifferent to the power of the signal and the same threshold can be used no matter what the signal strength is [11].

The DIFF COMBINED is calculated as:

$$DIFF\,COMBINED = \frac{DIFF\,FHR \cdot DIFF\,ECG}{maximum\,threshold} \tag{2.26}$$

The DIFF FHR and DIFF ECG signals represent how well the FHR and ECG templates correlate to its respective signal. The DIFF COMBINED represents how well both theses template correlate to their respective signals. Since a beat is usually represented in both channels at the same time, there is an advantage to have a combined comparison [11].

Beat detection: All of the DIFF signals are compared to a threshold value. If any of the DIFF signals are lower then the threshold, that time is set as a possible beat [11].

Threshold values: The threshold contains four values; Block, low, medium and high. Each DIFF value is compare with on of these four thresholds:

1. If a possible beat is too close (HR> 300 bpm) to a previous beat the block threshold is used to prevent FP detection.
2. If a possible beat is too close (240 bpm<HR<300 bpm) and the threshold/DIFF (FHR, ECG or COMBINED) that is being evaluated is not of the type that generated the lase beat, the block threshold is used.
3. The medium threshold is always used after three consecutive RR-intervals.
4. If the possible beat yields in a HR that is ± 10 bpm from the last one, the high threshold is used.
5. As a means to avoid situations presenting half or, if third HR the candidate HR corresponding to the current sample is one half, or one third the last HR detected, ±5 bpm, the medium threshold is used. This only applies if the threshold/DIFF being evaluated is the same as for the last beat being detected.
6. I all other cases the low threshold is used [11].

Rejection of a beat: There are some cases where a beat is rejected:

1. If the HR is higher than 240 bpm
2. If the HR is lower than 30 bpm
3. if the HR between the last HR and the new HR has increased more than 28 bpm
4. if the HR between the last HR and the new HR has decreased more than 1/3 times the previous HR [11].

3

Result

3.1 Adult ECG signal

The first validation of the algorithms was performed using adult ECG signals from the MIT/BIH database which contains 48 recordings, each 30 minutes long. The focus was to evaluate the overall performance of each algorithm by summing up numbers of TP, FN and FP to calculate the sensitivity and the positive predictive value. The results can be divided into two parts, first how the threshold effects the performance of the algorithms and the second is to determine which algorithms performs best for the adult ECG signal. These result are very accurate since the actual location of the R-waves are known.

For each algorithm the threshold value was changed 5 times to see if and then how it affected the performance of the algorithms. Figure 3.1 shows a sensitivity vs. positive predictive value graph with results from all four algorithms.

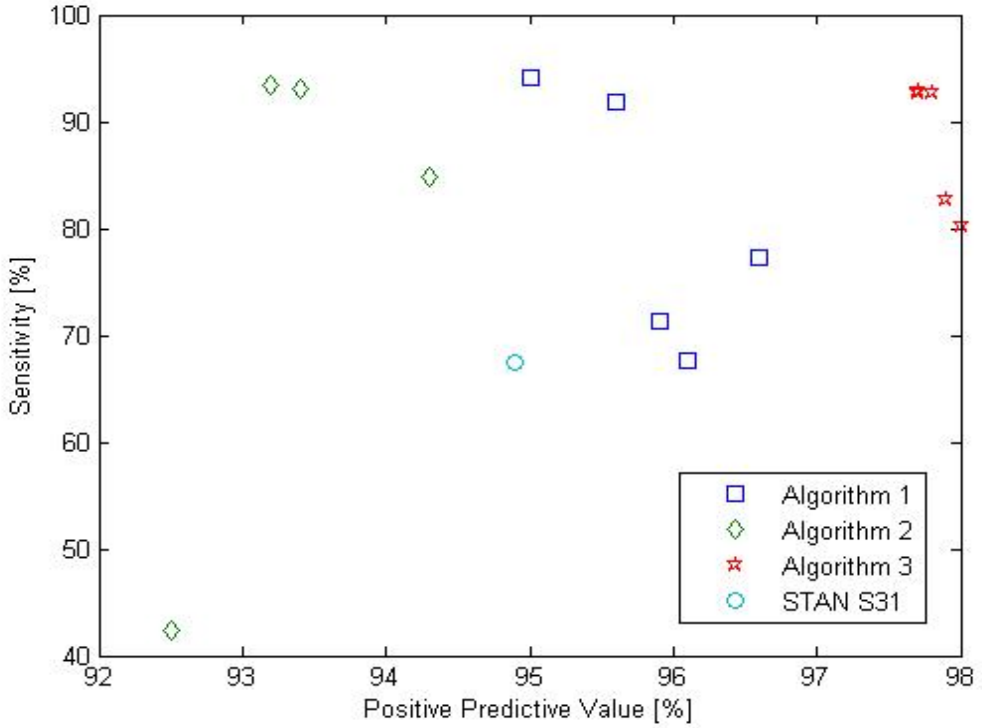

Figure 3.1: Result for the adult ECG signal

From figure 3.1 it can be seen how different values for the threshold affect the sensitivity and positive predictive value. Increasing the thresholds results in a lower sensitivity of the algorithms while the positive predictive value is around the same for all thresholds.

Table 3.1 shows the best obtained result for TP, FN and FP in each algorithm. From those values SE, +P and the variance are calculated. It is very clear from both figure 3.1 and table 3.1 that algorithm 3 performs the best. It has the lowest SE but the highest +P but when looking at figure 3.1 it is obvious the most accurate one when locating R-waves for adult ECG signal while the algorithm implemented in STAN S31 is not suited for adult R-wave detection. Table 3.1 shows the variance which gives indications of how close the detected values from each algorithm are to the actual R-wave location. Algorithm 3 has the lowest variance which means that its detected R-waves are closest to the actual R-waves.

Table 3.1: Best result for the adult ECG signal

Algorithm	TP	FN	FP	Se [%]	+P [%]	Variance
Algorithm 1	103073	6405	5465	94.15	94.96	$3.04 \cdot 10^{-5}$
Algorithm 2	102339	7139	7472	93.48	93.20	$1.21 \cdot 10^{-5}$
Algorithm 3	101677	7801	2416	92.87	97.68	$6.65 \cdot 10^{-6}$
STAN S31	73826	35652	3986	67.4	94.9	$2.4 \cdot 10^{-5}$

3.2 Fetal ECG signal

The second validation of the algorithms was performed by using fetal ECG signals from Neoventa, which contains 82 records, each approximately 30 minutes long.The focus was to evaluate the overall performance of each algorithm by summing up numbers of TP, FN and FP to calculate the sensitivity and the positive predictive value. Since the location of the R-waves are not known this validation will not be accurate. Only detections that three or all four algorithms detect were evaluated, the rest was disregarded.

The results from the validation of the three algorithms plus STAN S31 using fetal ECG signal are shown both in figure 3.2 and table 3.2. Figure 3.2 shows the sensitivity vs. positive predictive values while table 3.2 also show the number of TP, FN and FP.

Table 3.2: Result for the fetal ECG signal

Algorihms	TP	FN	FP	Unknown detections	Se [%]]	P [%]
Algorithm 1	283,074	3162	2138	21,051	98.90	98.25
Algorithm 2	284,178	2059	323	22,067	99.28	99.89
Algorithm 3	262,666	23,568	6809	26,965	91.77	97.47
STAN S31	267,902	18,310	1476	30,897	93.60	99.45

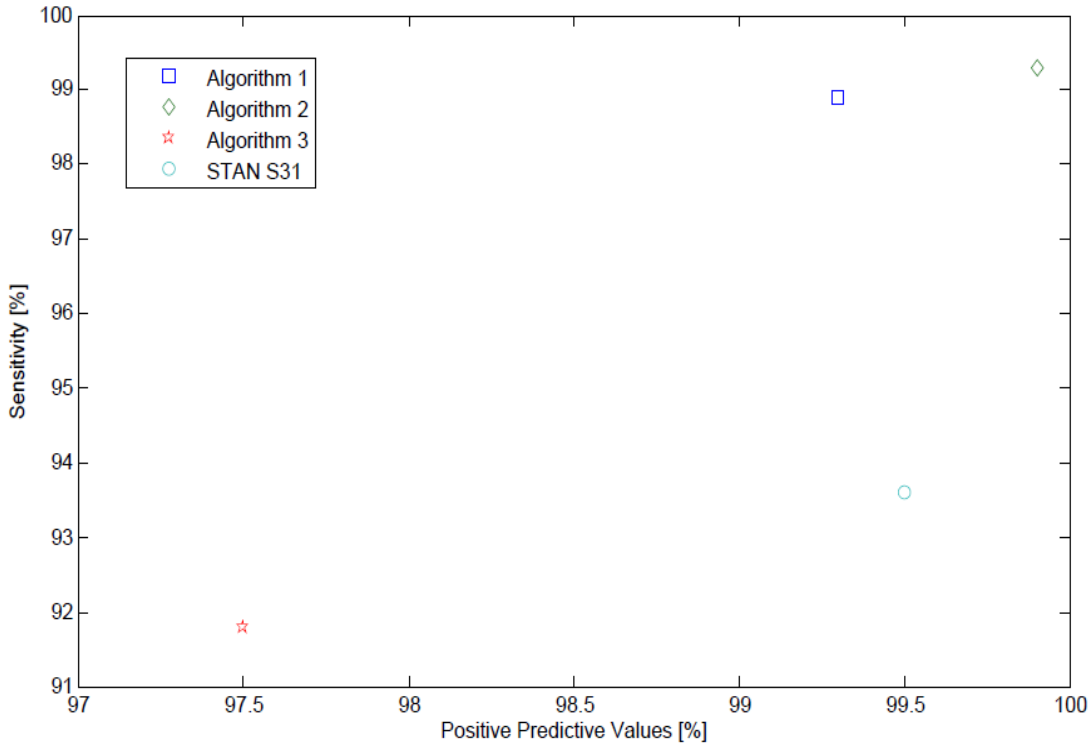

Figure 3.2: Result for the fetal ECG signal

From both table 3.2 and figure 3.2 it can be seen that the algorithm 2 has the highest sensitivity and positive predictive value. This gives an indication that algorithm 2 is the most accurate of those four though the values of sensitivity and positive predictive values are higher than in reality.

4

Discussion

4.1 Obstacles in the project

This project was divided into three steps; Investigate different algorithms, implement three of them and finally perform validation on two different types of ECG signals. When working on this project there were some obstacles that had to be dealt with. The first problem was when implementing the algorithms, the signal processing steps were easy to follow from the articles but the R-wave detections were more complicated and often it was unclear how the authors designed the R-wave detections. Therefore the algorithms may not be exactly the same as in the articles. The second obstacle was when deciding on how to perform the validation using the fetal ECG signal. Due to lack of time algorithm 2 was not implemented in C# like it was proposed in the aim.

4.2 Results

The validation process for the algorithms using two different type of signals are chosen differently depending on what information exists. When using the adult ECG signal the locations of the R-waves are known so the validation process is easily performed. The second and more important validation using the fetal ECG signal is more complex since the R-waves locations are not known. The most accurate way to determine TP, FN and FP is to look at figures of the signal with the detections but that is unrealistic since it is too time consuming. The validation process has to be easily performed and automatic. That is why this validation process is chosen for the fetal ECG signal, even though it will not give accurate information it will still give important information of the performance of the algorithms. The values for the sensitivity and positive predictive value are therefore higher than in reality, the number of unknown detection for each algorithms are 10-14 % of total number of detection.

When looking at the performance of all four algorithms for the adult signal it is interesting to see how poorly the algorithm in STAN S31 performs while the other three algorithms have high sensitivity and positive predictive value. By changing the threshold value it can be seen that the algorithms are stable and robust since they all have high performance rate for most threshold values. Changing the threshold gives lower sensitivity but similar positive predictive value. For the fetal signal all four algorithms have relatively good performance, all values for sensitivity and positive predictive value are over 90 %.

When comparing the results from the two validations it is clear that the algorithms perform differently with different type of signal. When looking at the adult ECG signal algorithm 3 is the most accurate, but for the fetal ECG signal algorithm 2 has the highest performance rate. There can be many reasons why the algorithms perform differently with different types of signal. First thing to keep in mind is that the adult and fetal ECG signal don't look the same. The fetal ECG signal has lower amplitude then the adult ECG signal and the amplitudes varies a great deal between different fetal ECG signals. Secondly the threshold has a great impact on number of detection, algorithms 1 and 2 have adaptive threshold that adjusts to different types of signals while algorithm 3 has fixed thresholds for both the noise and the peaks. This means that is is harder to find one fixed value that is going to work with different signals. Maybe if the threshold in algorithm 3 was to be changed to adaptive threshold the algorithm would have higher performance rate. The third reason is the filter. Each algorithm has different filter bandwidth and for some reason the filter in algorithm 2 seems to be performing better on the fetal ECG signal then the adult signal. Only algorithm 1 seems to be working similar for both the signals.

The result of the validation using the adult signal can not be compared to the articles since different criteria was chosen for this project. As an example in the article about the zero cross count the authors used 75 ms interval when finding number of TP, FP and FN while in this project the interval was 20 ms. [6] This will result in higher sensitivity and positive predictive values in the article than in the project.

5

Conclusion

In this project, three R-wave detection algorithms were implemented in Matlab and validated for both adult and fetal ECG signal. Algorithm 3 has the best performance when using adult ECG signal with Se of 92.87% and +P of 97.68%, but algorithm 2 has the best performance when using fetal ECG signal with Se of 99.28% and +P of 99.89%.

When designing a R-wave detection algorithm, there are many factors that need to be taken into a consideration but the first thing it to decide what theory the algorithm should be based on. From there the preprocessing steps are decided but all methods have some kind of bandpass filter implementation to eliminate noise and disturbance from the body and surroundings. The frequency range of the QRS complex needs to be explored to find which frequencies the signal should contain after the filtering. The algorithm might contain more filtering steps but it would be convenient to square the signal at some point to emphasize the R-wave and reduce unwanted parts of the signal. For the event detection some kind of threshold is necessary. This threshold can be fixed but it is better if the threshold adapts to the signal being processed since the signal don't look the same between people. When the events have been detected the R-wave can be located by finding the maximum value in each event.
When comparing the performance of the algorithm in STAN S31 to the performance of the other three algorithms it is obvious that the algorithm in STAN S31 could be improved to get higher sensitivity. This could be done by replacing the algorithm with a non-linear method previously published by Pan and Tompkins (algorithm 2).

Bibliography

[1] Neoventa Medical AB. Fetal monitoring and st analysis. 2008.

[2] B-U Kohler, C Henning, and R Orglmeister. The principles of software qrs detection. *IEEE Enginerring in Medicine and Biology*, 21(1):42–57, 2002.

[3] P.J Schwartz, A Garson, and T Paul. Guidlines for interpretation of the neonatal electrocardiogram. *European Heart Journal*, 23:1329–1344, 2002.

[4] Ecg library. http://www.ecglibrary.com/norm.html. Accessed:2013-09-03.

[5] Mit-bih arrhthmia databeas directory. http://www.physionet.org/physiobank/database/html/mitdbdi
Accessed:2013-08-08.

[6] B-U Kohler, C Henning, and R Orglmeister. Qrs detection using zero crossing counts. *Biomedical Research*, 8(3):138–145, 2003.

[7] J Pan and W Tompkins. A real- time qrs detection algorithm. *IEEE*, 32(3):230–236, 1985.

[8] Matlab. http://matlabz.blogspot.se/2011/04/contents-cancellation-dc-drift-and.html. Accessed:2012-03-18.

[9] V.X Afonso, W Tompkins, and T.Q Nguyen. Ecg beat detection using filter banks. *IEEE*, 46(2):192–202, 1999.

[10] Connexions. http://cnx.org/content/m18957/latest/nqrsdetect.m. Accessed:2012-04-14.

[11] Neoventa Medical AB. Fhr detection on combined fhr/ecg channels. 2006.

A

Appendix: Algorithm 1

```
function [RR_time] = function1_fetal(ecg)

% The input is the ecg signal

fs=500; % sampling frequency
A= 26; % filter ordar

%% Bandpass Filter, 27;53 Hz
bandpass = fir1(A,[5/(0.5*fs) 30/(0.5*fs)]); % bandpass filter
filter_signal_delay=conv(bandpass,ecg);
filter_signal=filter_signal_delay(A/2:length(filter_signal_delay)-A/2);


%% Nonlinear Transform of Signal
nonlinear = sign(filter_signal).*filter_signal.^2;

%% High Frequency Sequence
K = zeros; % timevarying amplitude
lambda = 0.99; % forgetting factor: lamda [0:1]
gain = 4; % constant gain

for j=1:length(nonlinear)-1
K(j+1)= lambda*K(j)+(1-lambda)*gain*abs(nonlinear(j+1));
end

b=zeros;
for j=1:length(nonlinear)
```

```
b(j)=(-1)^j*K(j);
end

%% Adding high frequency sequence to the noise_ecg
new_signal = nonlinear+b';

%% Detection and counting of zero crossings
d=zeros;
for j = 1:length(new_signal)-1
    d(j+1) = 0.5*abs(sign(new_signal(j+1))-sign(new_signal(j)));
end

lambda2=0.99; % smooths the signal, higher value -> smoother signal
D = zeros;
for j=1:length(new_signal)-1
    D(j+1)= lambda2*D(j)+(1-lambda2)*d(j+1);
end

%% Event detection
lambda3 = 0.97; % controlls the threshold, lower -> closer to signal
threshold = zeros;
for j=1:length(new_signal)-1
    threshold(j+1)= lambda3*threshold(j)+(1-lambda3)*D(j+1);
end


%% Find all places where D is greater than threshold
% all places where D is smaller then threshold -> negative
% all places where D is larger then threshold ->positive
 positvie= zeros;
 negative = zeros;
for j = 1:length(D)
    if D(j) < threshold(j)
        positive(j) = 1;
    else
            negative(j)=1;
    end
end

% location of every place where D>threshold and D<threshold, this markst
% the points of lower and upper thresholds -> every point where the
% threshold and D cross over each other.
[pk1 locs_pos] = find(positive);
```

```
[pk2 locs_neg]=find(negative);

% positive and negative difference of the locations
difference_pos=diff(locs_pos);
difference_neg=diff(locs_neg);

% Remove all values where the difference is 1.
ind_pos=find(difference_pos>1);
ind_neg=find(difference_neg>1);

% Find the upper and lower limit for all possible QRS segment interval.
upper_limit=locs_pos(ind_pos);
lower_limit=locs_neg(ind_neg);

% Making all possible events
for j=1:length(upper_limit)
    event(j,:)=[lower_limit(j) upper_limit(j)];
end

% Calculate the distance between two events and if the distance is too
% close then the events are grouped together where the lower limit belongs
% to the first one and the upper limit belongs to the last one
n=1;
for j=1:length(event)-1
        dist_event(n)=event(j+1,1)-event(j,2);
        n=n+1;
        if dist_event(j)<=30
            new_event(j,:)=[lower_limit(j) upper_limit(j+1)];

        else
            new_event(j,:)=event(j,:);
            new_event(length(event),:)=event(length(event),:);
        end
end

n=1;
for j=1:length(new_event)-1
    new_dist_event(n)=new_event(j+1,1)-new_event(j,2);
    n=n+1;
    if new_dist_event(j)<20
        new_new_event(j,:)=[new_event(j,1) new_event(j+1,2)];
    else
        new_new_event(j,:)=new_event(j,:);
```

```
        new_new_event(length(new_event),:)=new_event(length(new_event),:);
    end
end

% finding events that have been repeated
for j=1:length(new_new_event)-1
     if new_new_event(j,2)==new_new_event(j+1,2)
        new_new_event(j+1,:)=0;
     end
end

% Remove all the zeros
zeroRows = any(new_new_event==0, 2);
new_new_event(zeroRows,:)=[];

% Repeat the step to make sure that the events are not repeated
for j=1:length(new_new_event)-1
     if new_new_event(j,2)==new_new_event(j+1,2)
        new_new_event(j+1,:)=0;
     end
end

zeroRows = any(new_new_event==0, 2);
new_new_event(zeroRows,:)=[];


for j=1:length(new_new_event)
    event_size(j)=new_new_event(j,2)-new_new_event(j,1);
    final_event(j,:)=new_new_event(j,:);
    if event_size(j)<30
        final_event(j,:)=0;
    end
end

zeroRow = any(final_event==0,2);
final_event(zeroRow,:)=[];



%% Find the min and max value for nonliear signal at each interval
abs_max=zeros;
abs_min=zeros;
for j=1:length(final_event)
```

```
        abs_max(j)=abs(max(nonlinear(final_event(j,1):final_event(j,2))));
        abs_min(j)=abs(min(nonlinear(final_event(j,1):final_event(j,2))));
end

% If abs_min is much larger than abs_max then the R peak is located at
% abs_min but otherwise the R peak is located at abs_max
RR_amp=zeros;
for j=1:length(final_event)
     if abs_min(j) - abs_max(j) > 0.4
          RR_amp(j)=abs_min(j);
        else
          RR_amp(j)=abs_max(j);
     end
end

% Find the location of the peaks
abs_nonlinear=abs(nonlinear);
RR_location=zeros;
for j=1:length(RR_amp)
    RR_loc(j)=find(abs_nonlinear(final_event(j,1):final_event(j,2))==RR_amp(j
end

for j=1:length(RR_amp)
    RR_location(j)=final_event(j,1)+RR_loc(j)';
end

RR_location=RR_location';
RR_loc_time=(RR_location)/fs*10^3;

RR_amp=abs_nonlinear(RR_location);

% erase values that are too close
RR_time=RR_location;
n=1;
for j=1:length(RR_location)-1
    if RR_location(j+1)-RR_location(j)<250*10^-3*500
        if RR_amp(j)<RR_amp(j+1)
            RR_time(j)=0;
        else
            RR_time(j+1)=0;
        end

     end
```

```
end

% R-wave location
RR_time=RR_time(RR_time~=0);

% R-wave location in ms
RR_time=RR_time*10^3/fs;
```

B

Appendix: Algorithm 2

```
function [RR_time] = function2_fetal(ecg)

% Input signal is the ecg signal

fs=500; % Samplinga frequency
N = length (ecg);        % Signal length

%% Low Pass Filter
% delay = 5 samples
% Differnece equation of low pass filter / coefficients a1 and b1
% cutoff frequency is 14 Hz, gain is 36 and delay is 10 ms
% Transfer function: H(z)= (1-z^-6)^2/(1-z^-1)^2
a1 = [1 -2 1];
b1 = [1 0 0 0 0 0 -2 0 0 0 0 0 1];

lp_signal = filter(b1,a1,ecg);
lp_signal = lp_signal/max(abs(lp_signal)); % Normalize

%% High Pass Filter
% delay = 16 samples
% Differnece equation of high pass filter / coefficients a2 and b2
% cutoff frequency is 7 Hz, gain is 32 and delay is 80 ms
%
a2 = [1 -1];
b2 = [-1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 32 -32 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1];
```

```
hp_signal = filter(b2,a2,lp_signal);
hp_signal = hp_signal/max(abs(hp_signal)); % Normalize

%% Derivate
% delay = 2 samples
h=[-1 -2 0 2 1]/8;

derivative=conv(hp_signal,h);
derivative = derivative(2+[1:N]); % Correcting delay of two samples
derivative = derivative/max(abs(derivative)); % Normalize

%% Squaring
squaring = derivative.^2;
squaring = squaring/max(abs(squaring)); % Normalize

 %% Moving-Window Integration
width = 50; % Number of samples in the width (150 ms window)
h2 = ones (1 ,width+1)/(width+1); % Impulse reponse

moving_window = conv (squaring ,h2); % Apply filter
moving_window = moving_window(width/2+[1:N]); % Correcting delay of 15 samples
moving_window = moving_window/max(abs(moving_window));

% QRS different than pan tompkins
max_window = max(moving_window);
mean_window = mean(moving_window);
threshold = mean_window*max_window;
y = (moving_window>threshold)';

left = find(diff([0 y])==1);
right = find(diff([y 0])==-1);

left=left-(5+16); % cancel delay of the low and high pass filter
right=right-(5+16); % cancel delay of the low and high pass filter

R_loc=zeros;
R_value=zeros;

for j=1:length(left)
    if left(j)<1;
        left(j)=1;
    end
    if right(j)>length(ecg)
```

```
        right(j) = length(ecg);
    end
     [R_value(j) R_loc(j)] = max(ecg(left(j):right(j)));
     R_loc(j)=R_loc(j)-1+left(j); % adding offset
end

RR_loc=R_loc(find(R_loc~=0));

RR_amp=ecg(RR_loc);
RR_time=RR_loc;
n=1;
for j=1:length(RR_loc)-1
    if RR_loc(j+1)-RR_loc(j)<250*10^-3*500
        if RR_amp(j)<RR_amp(j+1)
            RR_time(j)=0;
        else
            RR_time(j+1)=0;
        end

     end
end

% R-wave location
RR_time=RR_time(RR_time~=0);

% R-wave location in ms
RR_time=RR_time*10^3/500;
```

C

Appendix: Algorithm 3

Parts of the algorithm are obtained by using the free software in [10].

```
function [RR_time] = function3_fetal(ecg)

% Input signal is the ecg signal
fs=500;
data=ECG;
N=400;

%% Filter banks
Bw=5.625;      %filter bandwidth
Bwn=1/(fs/2)*Bw;
M=32; %downsampling rate

k1=.002;
h1=fir1(N,[Bwn-k1 2*Bwn+k1]);
h2=fir1(N,[2*Bwn-k1 3*Bwn+k1]);
h3=fir1(N,[3*Bwn-k1 4*Bwn+k1]);
h4=fir1(N,[4*Bwn-k1 5*Bwn+k1]);

w1=conv(data,h1);
w2=conv(data,h2);
w3=conv(data,h3);
w4=conv(data,h4);

y2=downsample(w1,32);
```

```
y3=downsample(w2,32);
y4=downsample(w3,32);
y5=downsample(w4,32);

%% Features
P1=sum([abs(y2) abs(y3) abs(y4)],2);
P2=sum([abs(y2) abs(y3) abs(y4) abs(y5)],2);
P4=sum([abs(y3) abs(y4) abs(y5)],2);

a1=[0; P1];
a2=[0; P2];
a3=[0; P4];

b1=[P1; 0];
b2=[P2; 0];
b3=[P4; 0];

c1=[a1,b1];
c2=[a2,b2];
c3=[a3,b3];

FL1=sum(c1,2)/2;
FL2=sum(c2,2)/2;
FL4=sum(c3,2)/2;

%% Level 1 [1]
[EventsL1_amp EventsL1]=findpeaks(FL1);

%% Level 2 [1]
meanL1=sum(FL2(EventsL1),1)/length(EventsL1);
NL=meanL1-meanL1*0.1;   %Start Noise Level
SL=meanL1+meanL1*0.1;   %Start Signal Level
threshold1=0.08;    %Threshold detection block 1
threshold2=0.7;     %Threshold detection block 2
[SignalL21,Noise1,DS1,Class1]=detectionblock(FL2,EventsL1,NL,SL,threshold1);
[SignalL22,Noise2,DS2,Class2]=detectionblock(FL2,EventsL1,NL,SL,threshold2);

%% Level 3 [1]
ClassL3=[];
for i=1:length(EventsL1)
    C1=Class1(i);
    C2=Class2(i);
    if C1==1
```

```
        if C2==1
            ClassL3=[ClassL3 1];   %Classification as Signal
        else
            delta1=(DS1(i)-threshold1)/(1-threshold1);
            delta2=(threshold2-DS2(i))/threshold2;
            if delta1>delta2
                ClassL3=[ClassL3 1]; %Classification as Signal
            else
                ClassL3=[ClassL3 0];  %Classification as Noise
            end
        end
    else
        if C2==1;
            ClassL3=[ClassL3 1]; %Classification as Signal
        else
            ClassL3=[ClassL3 0];  %Classification as Noise
        end
    end
end
SignalL3=EventsL1(find(ClassL3));   %Signal Level 3
NoiseL3=EventsL1(find(ClassL3==0)); %Noise Level 3
%% Level 4 [1]
threshold=0.3;
VSL=(sum(FL4(SignalL3),1))/length(SignalL3);
VNL=(sum(FL4(NoiseL3),1))/length(NoiseL3);
SL=(sum(FL4(SignalL3),1))/length(SignalL3);   %Initial Signal Level
NL=(sum(FL4(NoiseL3),1))/length(NoiseL3);      %Initial Noise Level
SignalL4=[];
NoiseL4=[];
DsL4=[];   %Detection strength Level 4
for i=1:length(EventsL1)
    Pkt=EventsL1(i);
    if ClassL3(i)==1;   %Classification after Level 3 as Signal
       SignalL4=[SignalL4,EventsL1(i)];
       SL=history(SL,FL4(Pkt));
       Ds=(FL4(Pkt)-NL)/(SL-NL);        %Detection strength
       if Ds<0
           Ds=0;
       elseif Ds>1
           Ds=1;
       end
       DsL4=[DsL4 Ds];
    else        %Classification after Level 3 as Noise
```

```
        Ds=(FL4(Pkt)-NL)/(SL-NL);
        if Ds<0
            Ds=0;
        elseif Ds>1
            Ds=1;
        end
        DsL4=[DsL4 Ds];
        if Ds>threshold          %new classification as Signal
            SignalL4=[SignalL4,EventsL1(i)];
            SL=history(SL,FL4(Pkt));
        else                      %new classification as Noise
            NoiseL4=[NoiseL4,EventsL1(i)];
            NL=history(NL,FL4(Pkt));
        end
    end
end

%% Level 5
%if the time between two RR complexes is too long => go back and check the
%events again with lower threshold
SignalL5=SignalL4;
NoiseL5=NoiseL4;
periods=diff(SignalL4);
M1=100;
a=1;
b=1/(M1)*ones(M1,1);
meanperiod=filter(b,a,periods); %mean of the RR intervals
SL=sum(FL4(SignalL4))/length(SignalL4);
NL=sum(FL4(NoiseL4))/length(NoiseL4);
threshold=0.2;
for i=1:length(periods)
    if periods(i)>meanperiod*1.5     %if RR-interval is to long
        intervall=SignalL4(i):SignalL4(i+1);
        critical=intersect(intervall,NoiseL4);
        for j=1:length(critical)
            Ds=(FL4(critical(j))-NL)/(SL-NL);
            if Ds>threshold          %Classification as Signal
                SignalL5=union(SignalL5,critical(j));
                NoiseL5=setxor(NoiseL5,critical(j));
            end
        end
    end
end
```

```
%% Upsample
Signaln=conversion(data,FL2,SignalL5,M,N,fs);
%%
RR_amp=data(Signaln);
RR_amp=abs(RR_amp);
RR_time=Signaln;

for j=1:length(RR_time)-1
    if RR_time(j+1)-RR_time(j)<250*10^-3*fs
        if RR_amp(j)<RR_amp(j+1)
            RR_time(j)=0;
        else
            RR_time(j+1)=0;
        end

     end
end

% R-wave location
RR_time=RR_time(RR_time~=0);


QRS=RR_time*10^3/fs;

%% subfunctions

function [Signal,Noise,VDs,Class]=detectionblock(mwi,Events,NL,SL,threshold)

% detectionblock - computation of one detection block
%
%   [Signal,Noise,VDs,Class]=detectionblock(mwi,Events,NL,SL,threshold)
%
% INPUT
%   mwi         Output of the MWI
%   Events      Events of Level 1 (see [1])
%   NL          Initial Noise Level
%   SL          Initial Signal Level
%   threshold   Detection threshold (between [0,1])
%
% OUTPUT
%   Signal      Events which are computed as Signal
%   Noise       Events which are computed as Noise
%   VDs         Detection strength of the Events
```

```
%   Class          Classification: 0=noise, 1=signal

Signal=[];
Noise=[];
VDs=[];
Class=[];
sumsignal=SL;
sumnoise=NL;
for i=1:length(Events)
    P=Events(i);
    Ds=(mwi(P)-NL)/(SL-NL); %Detection strength
    if Ds<0
        Ds=0;
    elseif Ds>1
        Ds=1;
    end
    VDs=[VDs Ds];
    if Ds>threshold     %Classification as Signal
        Signal=[Signal P];
        Class=[Class;1];
        sumsignal=sumsignal+mwi(P);
        SL=sumsignal/(length(Signal)+1);    %Updating the Signal Level
    else        %Classification as Noise
        Noise=[Noise P];
        Class=[Class;0];
        sumnoise=sumnoise+mwi(P);
        NL=sumnoise/(length(Noise)+1); %Updating the Noise Level
    end
end
%-------------------------------------------------------------------------
function [pnew]=conversion(data,FL2,pold,M,N,fs)

% conversion - sets the fiducial points of the downsampled Signal on the
% samplepoints of the original Signal
%
%   [pnew]=conversion(data,FL2,pold,M,N,fs)
%
% INPUT
%   data            Original ECG Signal
%   FL2             Feature of Level 2 [1]
%   pold            old fiducial points
%   M               M downsampling rate
%   N               filter order
```

```
%   fs          sample rate
%
% OUTPUT
%   pnew        new fiducial points
%

Signaln=pold;
P=M;
Q=1;
FL2res=resample(FL2,P,Q);        %Resampling
nans1=isnan(data);
nans=find(nans1==1);
data(nans)=mean(data);     %Replaces NaNs in Signal
for i=1:length(Signaln)
    Signaln1(i)=Signaln(i)+(M-1)*(Signaln(i)-1);
end
%-------------------- Sets the fiducial points on the maximum of FL2
Signaln2=Signaln1;
Signaln2=Signaln2';
int=2*M;     %Window length for the new fiducial point
range=1:length(FL2res);
for i=1:length(Signaln2)
    start=Signaln2(i)-int/2;
    if start<1
        start=1;
    end
    stop=Signaln2(i)+int/2;
    if stop>length(FL2res)
        stop=length(FL2res);
    end
    intervall=start:stop;        %interval
    FL2int=FL2res(intervall);
    pkt=find(FL2int==max(FL2int));  %Setting point on maximum of FL2
    if length(pkt)==0   % if pkt=[];
        pkt=Signaln2(i)-start;
    else
        pkt=pkt(1);
    end
    delay=N/2+M;
    Signaln3(i)=pkt+Signaln2(i)-int/2-delay;     %fiducial points according to
end
%Sets the fiducial points on the maximum or minimum
%of the signal
```

```
Bw=5.625;
Bwn=1/(fs/2)*Bw;
Wn=[Bwn 5*Bwn];
N1=32;
b=fir1(N1,Wn,'bandpass');
Sf=filtfilt(b,1,data);      %Filtered Signal with bandwidth 5.6-28 Hz
beg=round(1.5*M);
fin=1*M;
for i=1:length(Signaln3)
    start=Signaln3(i)-beg;
    if start<1
        start=1;
    end
    stop=Signaln3(i)+fin;
    if stop>length(Sf)
        stop=length(Sf);
    end
    intervall=start:stop;   %Window for the new fiducial point
    Sfint=abs(detrend(Sf(intervall),0));
    pkt=find(Sfint==max(Sfint));    %Setting point on maximum of Sfint
    if length(pkt)==0   %if pkt=[];
        pkt=Signaln3(i)-start;
    else
        pkt=pkt(1);
    end
    pkt=pkt(1);
    Signaln4(i)=pkt+Signaln3(i)-beg-1;
end
Signal=Signaln4';   %New fiducial points according to the original signal

cutbeginning=find(Signal<N);    %Cutting out the first points because of init
fpointsb=Signal(cutbeginning);
cutend=find(Signal>length(data)-N); %Cutting out the last points
fpointse=Signal(cutend);
pnew=setxor(Signal,[fpointsb;fpointse]);
%-----------------------------------------
function yn=history(ynm1,xn)

% history - computes y[n]=(1-lambda)*x[n]+lambda*y[n-1]
%
%   yn=history(ynm1,xn)

lambda=0.8; %forgetting factor
```

```
yn=(1-lambda)*xn+lambda*ynm1;
```