FETAL ECG DETECTION

Submitting:

Yotam Nizri

Omri Berman

Professional Supervisor:

Prof. Yonina Eldar

instructor:

Gal Mazor

Professional Supervisor:
Dr. Shai Tejman

Registration Semester:

Spring 2017

Submission date:
24/12/2017

Table of Contents

Table of Figures

Chapter1 – Project A

1. Introduction

1.1 ECG

ECG(ElectroCardioGraphy) is the process of acquiring the electrical activity of one's heart. The acquired signal demonstrates the heart's functionality in Voltage as a function of time, i.e. V(t).

The acquisition is done by placing electrodes on a patient's body.
These electrodes detect electrical changes on the patient's skin which originate in the depolarization during each heartbeat.

ECG conveys a large amount of information about the structure of the heart and the function of its electrical conduction system.
Among other things, an ECG can be used to measure the rate and rhythm of heartbeats, the size and position of the heart chambers, the presence of any damage to the heart's muscle cells or conduction system, the effects of cardiac drugs, and the function of implanted pacemakers.

1.2 The ECG Signal

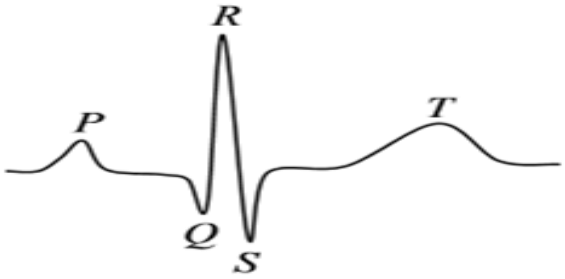


Fig1. ECG signal in Voltage vs Time

The ECG signal consists of several parts: P, Q, R , S , T

P wave- represents depolarization of the atria

QRS – depolarization of the right and left ventricles.

T - repolarization of the right and left ventricles.

1.3 The Problem

Cardiac MRI is crucial in complex congenital heart diseases for understanding the anatomy and physiology of the complex heart and helps the cardiologists and surgeons determine the route the patient will follow in his corrective or palliative surgical future.
Fetal echocardiography has made a tremendous change in the way of treating patients, allowing to prepare ahead of time for complex congenital malformations and even contemplate abortions in devastating conditions.

Fetal echocardiography has its limitations as complex congenital conditions sometimes stay unsolved until the baby is born and an echo or an MRI is performed.
The cardiac MRI needs to be synchronized to the heart beat, since the cardiac cycle is sampled consistently in the same manner.
Therefore, MRI is feasible only after the child is born and the ECG is properly measured, since, currently, there is not an excitant way to separately obtain fetal ECG.

A very popular way to try and obtain the fetal ECG is by recording it through skin electrodes attached to the mother's abdomen.
The process of recording is usually complicated since the **maternal** ECG is of higher amplitude compared to the fetal one.
In addition, the **fetal** ECG is contaminated by many sources of noise such as the electronic equipment, the patient respiration and movement.
However, an effective signal processing study is needed in order to separate the two wanted sources, FECG (Fetal ECG) and MECG (Maternal ECG) components, from the corrupted mixture recordings.
Ever since 1960, many signal-processing techniques have been introduced to improve the quality of the FECG detection with varying average of success.
The most popular techniques include adaptive filters, singular-value decomposition (SVD), wavelet transform, adaptive Neuro-Fuzzy inference systems to treat the nonlinear relationship between the thoracic ECG and the maternal ECG component in the abdominal ECG signals.
Another efficient work was the use of blind source separation (BSS).
The BSS aims to recover unknown source signals from a set of observations which are an unknown mixtures of source signals.

All of the above methods were tested on simulated ECG signals, based on assumptions that were made regarding the behavior of the signal and the noise.
However, non of them were tested on a real superposition of MECG and FECG.

1.4 Solution

In order to test separation methods on real ECG signals of pregnant patients, an application was issued to Helsinki Committee for Human Rights.
After the application was approved, we tried to implement two different algorithms for separating the FECG from the superposition signal:

a. Dictionary Learning and KSVD.

b. Sequential Total Variation Denoising.

Method a. did not provide the desired results due to lack of correlation between the given signals and the theoretical model.
Therefore, we will focus on method b.

2. Theoretical background

Although we eventually didn't get all the way through with the dictionary learning and KSVD algorithm, we spent a lot of time dealing with it. Hence, we will address it at this section as well.

2.1 Compressed Sensing

Compressed Sensing is a method to acquire a signal $x \in R^n$ by taking only m <n linear measurements $y = Ax$ using an m x n compressed sensing matrix A.
y will be referred to as the measurement vector.
The motivation behind the design of A is, therefore, to allow for distinct signals x, x' within a class of signals of interest to be uniquely identifiable from their measurements $y = Ax, y' = Ax'$, even though m<<n.

2.2 Orthogonal Matching Pursuit (OMP)

OMP is an algorithm designed to find a sparse representation of given signal, using a compressed sensing matrix and a measurement vector as inputs:

Let A be the CS matrix, y the measurements vector , $\hat{x}$ the sparse representation, r the residual signal estimation, Λ the support of the residual, and l the sparsity of x.

The algorithm can be described by the following pseudo-code:

a. Initialize: $\hat{x}_0 = 0, r = y, \Lambda = \phi, l = 0$

b. While halting criterion is false, do:

- $l = l + 1$
- $b = Ar$
- Update Λ with the support of the residual
- $\hat{x}_l |_\Lambda = A_\Lambda^\dagger y, \qquad S.T \ \hat{x}_l |_{\Lambda^c} = 0$
- $r = y - A\hat{x}_l$

c. Return $\hat{x} = \hat{x}_l$

2.3 Dictionary learning

Dictionary learning is a method which aims at finding a sparse representation of an input data. The above representation is by a linear combination of bases elements, as well as those bases elements themselves. With a given set of training signals, we can create a dictionary matrix, using an iterative algorithm. This matrix would supposedly be used to separate the desired embryo ECG signal from the abdominal superposition.

The dictionary learning objective can be posed as the following optimization tasks:

$$\min_{\mathbf{D},\{\underline{\alpha}_k\}} \sum_{k=1}^{N} \left\|\underline{\alpha}_k\right\|_0^0 \text{ s.t. } \left\|\underline{x}_k - \mathbf{D}\underline{\alpha}_k\right\|_2 \le \varepsilon$$

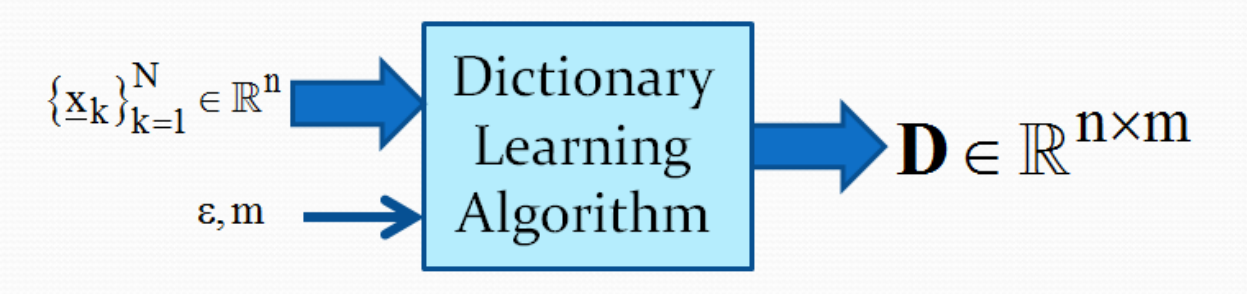


Fig2. Dictionary learning algorithm scheme

Where D is the Dictionary matrix, α_k is a K sparse vector, and x_k is the given signal.

2.4 Total Variation De-noising (TVD)

Total variation de-noising, also known as total variation regularization is a process, most often used in digital image processing, that has applications in noise removal. It is based on the principle that signals with excessive and possibly spurious detail have high total variation, that is, the integral of the absolute gradient of the signal is high. According to this principle, reducing the total variation of the signal subject to it being a close match to the original signal, removes unwanted detail whilst preserving important details such as edges.

This noise removal technique has advantages over simple techniques such as linear smoothing or median filtering which reduce noise but at the same time smooth away edges to a greater or lesser degree. By contrast, total variation de-noising is remarkably effective at simultaneously preserving edges whilst smoothing away noise in flat regions, even at low signal-to-noise ratios.

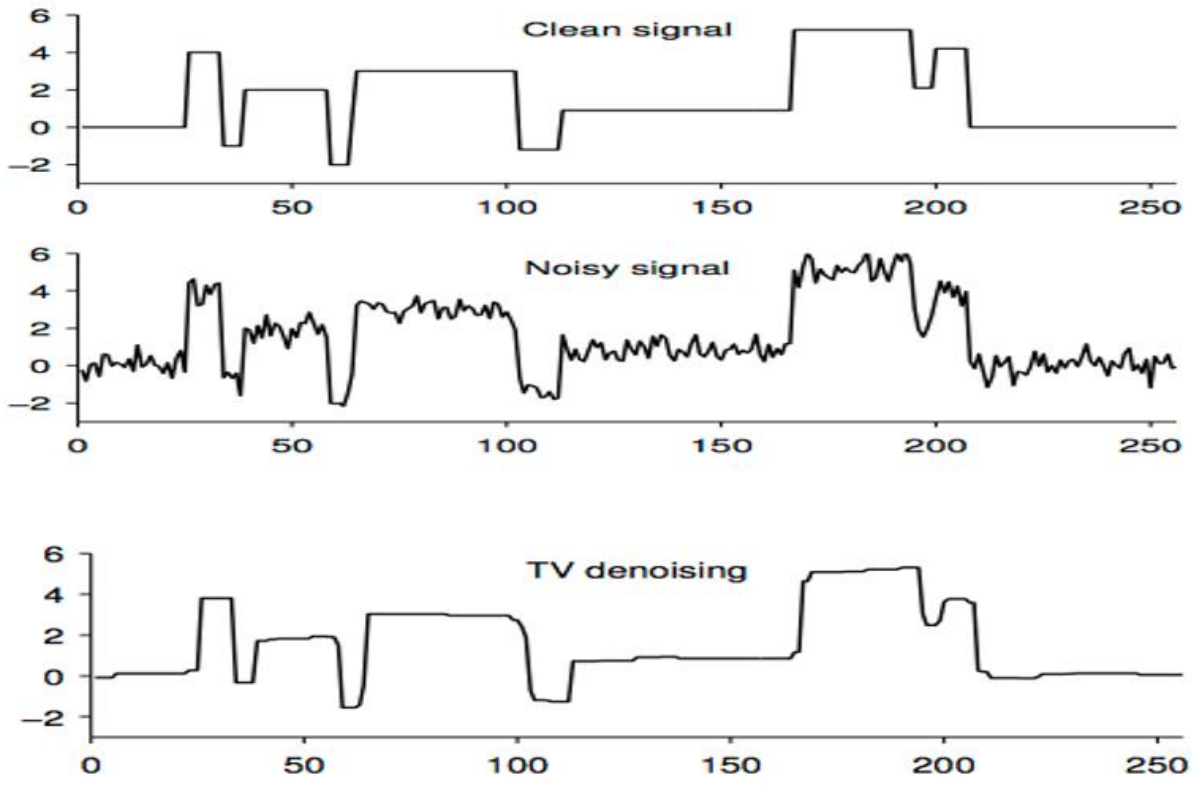


Fig3. TV example

For a digital signal y_n, we can, for example, define the total variation as:

$$V(y) = \sum_n |y_{n+1} - y_n|$$

Given an input signal x_n,the goal of total variation de-noising is to find an approximation, call it y_n, that has smaller total variation than x_n but is "close" to x_n.

One measure of closeness is the sum of square errors:

$$E(x, y) = \frac{1}{2}\sum_n (x_n - y_n)^2$$

So the total variation de-noising problem amounts to solve the following discrete optimization problem over the signal y_n:

$$\hat{y}_n = \arg\min_{y_n} \left(\frac{1}{2}\sum_n (x_n - y_n)^2 + \lambda \sum_n |y_{n+1} - y_n| \right)$$

By differentiating this functional with respect to y_n, we can derive a corresponding Euler Lagrange equation, that can be numerically integrated with the original signal x_n as initial condition.

2.5 Singular Values Decomposition (SVD)

In linear algebra, the singular value decomposition is a factorization of a real or complex matrix.

Formally, the singular value decomposition of an mxn real or complex matrix M is a factorization of the form $U\Sigma V^*$, where U is mxm real or complex unitary matrix, Σ is a mxn rectangular diagonal matrix with non-negative real numbers on the diagonal, and V is a nxn real or complex unitary matrix. The diagonal entries σ_i are the singular values of M. The columns of U and V are called the left-singular vectors and right-singular vectors of the matrix M, respectively.

M = $U\Sigma V^*$

2.5.1 Low-rank matrix approximation

Some practical application solve the problem of approximating a matrix M with another matrix M, which has a specific rank r.
One offered approximation is based on minimizing the Frobenius norm of the difference between M and M under the constraint that $rank(M) = r$.

SVD turned out to be the solution to this problem, simply by replacing Σ matrix with the matrix Σ when Σ is the same as Σ except that it contains only the r largest singular values when the other singular values are replaced by zero.
This is known as the Eckart-Young theorem, as it was proved by those two in 1936.

3. Implementation

3.1 General Description

The Sequential Total Variation Denoising idea was taken from an Article written by Kwang Jin Lee and Boreom Lee, which was published on 1 July 2016.

The algorithm proposed in the article consists of three major steps:

a. TVD_1

b. TS_{pca}

c. TVD_2

The above algorithm was tested on simulated signals, i.e. superposition of Gaussian functions and random additive noise.

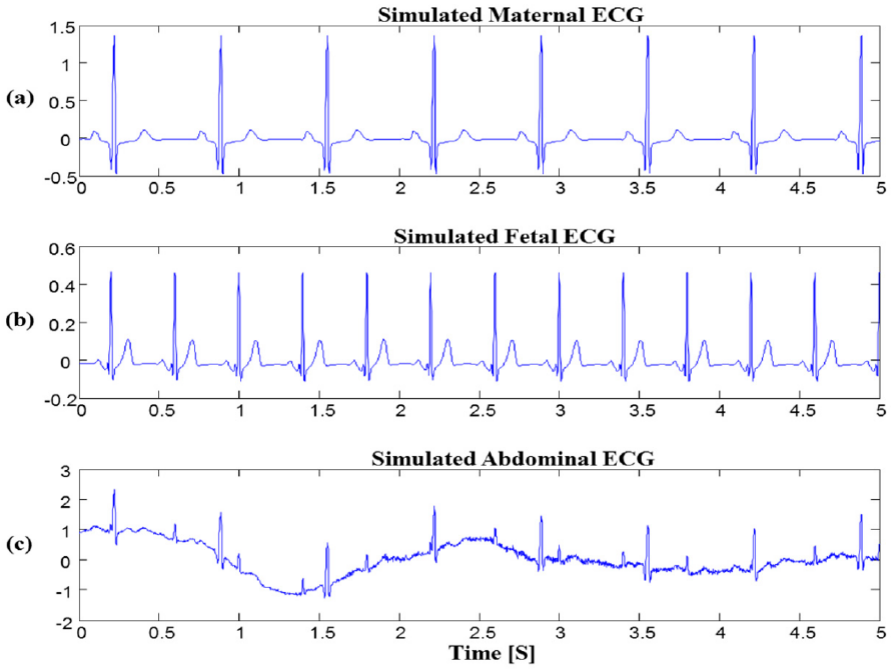


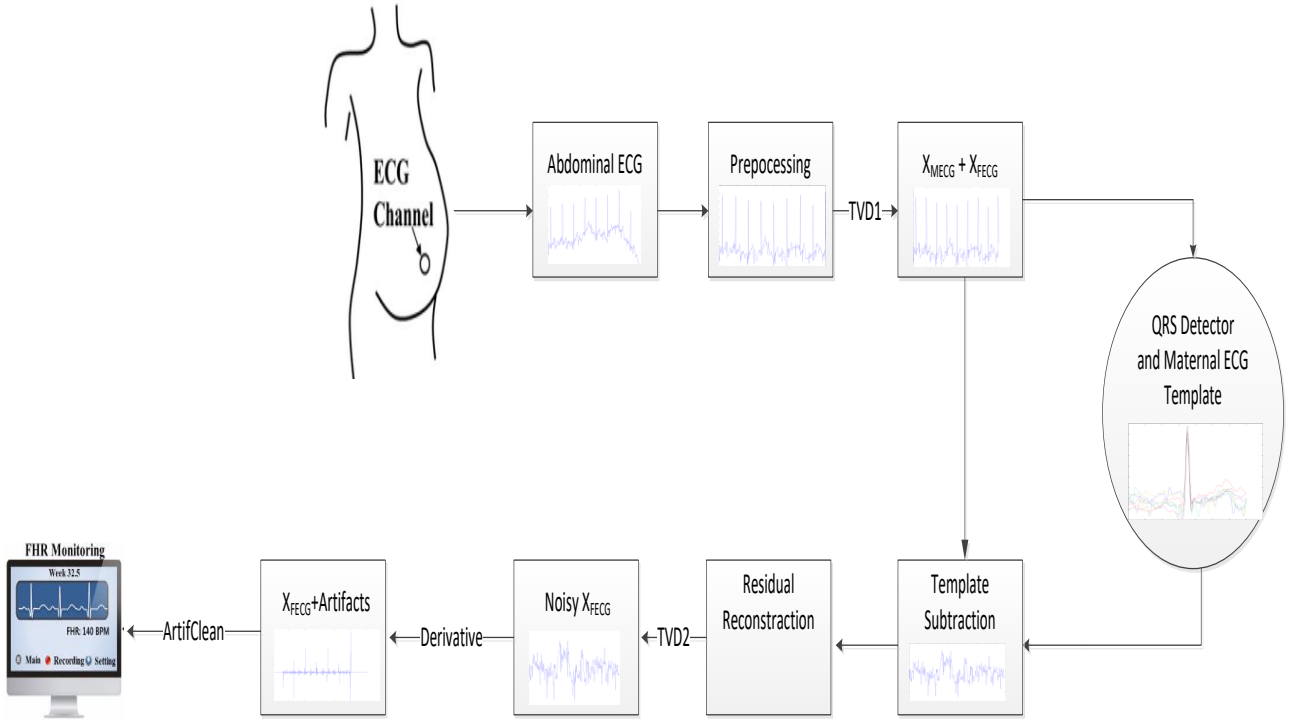
Fig4. Simulated MECG, FECG & abdominal example

The results of the algorithm were compared to other known methods ,which were also tested on simulated signals, and achieved relatively better results.

Our algorithm implementation was based on the above article and includes several changes that brought better results on real superposition of MECG and FECG signals.

3.2 Algorithm Architecture

Our implementation consist of several blocks as described in the following scheme:



- **Preprocessing**

  Based on medical researches, the MECG and FECG components are only likely to be bounded between 0.5 and 45 Hz.
  Therefore, We applied a BPF (Band Pass Filter) on the given abdominal ECG, with 0.5,45 cutoff frequencies, in order to eliminate as many Electromyogram noise components as possible.

TODO: add pictures

- **TVD1**

  Let the MECG signal be represented by y1, then:

  $$Y_1 = X + e_1$$

  Where X is the clean abdominal vector, and e1 is the residual noise, meaning Y1 is a contaminated measurement of X.

  The clean abdominal signal X is given by a superposition of the MECG and FECG:

  $$X = X_{MECG} + X_{FECG}$$

  X can be extracted using TVD (Total variation denoising) as follows:

  $$\hat{X} = \arg\min_x \left(\frac{1}{2} \left\| Y_1 - X \right\|_2 \right) + \lambda_1 \left\| D_2 X \right\|_1$$

  Where $\lambda_1 > 0$, and D_2 is the second order difference matrix, which aims at making the signal sparser:

  $$D_2 = \begin{bmatrix} -1 & 2 & -1 & 0 & \dots & 0 \\ 0 & -1 & 2 & -1 & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -1 & 2 & -1 \end{bmatrix}$$

  In general, D2 is a Topelitz matrix of size $(N-2)xN$.

- **QRS Detection**

  In order to be able to subtract the MECG component from the abdominal, we first have to locate QRS peaks of the MECG.

To do so, we applied a function that finds the peaks in the abdominal signal.
In order to avoid misdetections of noise and fetal components, we only detect peaks that are greater from a predefined parameter used as threshold.

- **Maternal ECG Template**

  After obtaining the location of the MECG QRS peaks, We divided the abdominal signals into pulses, so that each pulse contains a different QRS peak. The division was made so that the QRS peak is located at the middle index of every pulse.

  In order to build a matrix composed of the pulses, we had to cut some of the pulses in order for them to have the same length. For this aim, we identified the pulse of minimal length, and cut the rest of the pulses symmetrically to fit it's dimensions.

  After the pulses are consistent in terms of dimensions, we set each pulse as a row of a matrix, meaning all the QRS peaks appear at the same column.

- **TS_{PCA}**:

  This phase's goal is to obtain a template of the MECG from the abdominal and subtract it from the superposition, leaving us only with the FECG. To do so, an SVD transform is applied on the ECG pulses matrix, and all singular values of the orthogonal matrix are set to zero, excluding the biggest singular values (based on Eckart-Young theorem).
  This can be easily done, since the singular values are sorted from biggest to smallest in the diagonal matrix. This leaves us with the principle components of the abdominal signal, which should approximately be the MECG, since all other abdominal components are inconsistent between different pulses.

  To complete the Template subtraction, the achieved MECG component is subtracted of each pulse.

- **TVD2**

  Although we assume that by this phase the MECG is gone, some of its components still appear in the supposedly clean FECG signal. Moreover, some independent EMG noise components might still appear as well. Hence, a second TVD is applied on the residual signal to enhance the fetal ECG signal as shown below:

  $$Y_2 = X_{FECG} + e_2$$

  $$\hat{X}_{FECG} = \arg\min_{X_{FECG}} \left(\frac{1}{2} \| Y_2 - X_{FECG} \|_2 \right) + \lambda_2 \| D_2 X_{FECG} \|_1$$

  Where e_2 represents the residual artifacts, including remains of MECG and noise.

- **First order derivation**

  After all the filtering and cleaning methods, when applied on real abdominal signals, the signal at this phase is still contaminated by noise of relatively high amplitude comparing to the FECG.

  Despite the impossibility to identify the FECG by amplitude, it's frequency is much higher than the other noises and artifacts.

  Therefore, we apply a first order derivation. The result is a signal with great amplitudes in the location of the FECG peaks, and much smaller amplitude in the noises and artifacts locations (excluding some abnormally great amplitudes originating in discontinuity in the sampled abdominal signal).

- **Artifacts cleaning**

  At this phase, we encounter misdetections due to Discontinuity. The reasons are:

  a. Reconstruction with the residual parts from the MECG template generation phase.

  b. Sampling errors

  In addition, the remaining noises need to be filtered, which are of lower amplitude, as explained under "First order derivation" phase.

  To achieve these goals, we apply the following procedures:

  1. Double Thresholding on the signal, to get rid of the overshoots and the noises
  2. Masking the edge point between the residual signal to the derived FECG.

3.3 Algorithm Pseudo Code

Let X be the abdominal signal, i.e. $X = X_{FECG} + X_{MECG} + \text{Noise}$.

Fetal_Detection *(X , λ_1, λ_2 , TH_{QRS} , singular_val, TH_h , TH_l):*

X_prep ← BPF(X, 0.5, 45) ;

X_clean ← TVD1(X_prep, λ_1);

QRS_idx ← QRS_Detect (X_clean, TH_{QRS});

[QRS_mat,Residue] ← Built_Mat(X_clean, QRS_idx);

[U S V] ← SVD(QRS_mat);

S_shrink ← zeros; *//set S_shrink to be zeros in the size of S*

For i from 1 to singular_val+1 do

S_shrink[i,i] ← S[i,i];

QRS_tmplt ← U S_shrink*V;*

QRS_sub ← QRS_mat - QRS_tmplt ; *//template subtraction*

QRS_sub_clean ← TVD2 (QRS_sub(:) , λ_2) ;

QRS_clean_rec ←Reconstract(QRS_sub_clean, Residue);

F_ECG ← D_1(QRS_clean_rec) ; *//first order derivative*

F_ECG ← Artifacts_Clean(F_ECG, TH_l ,TH_h , Residue, QRS_pulse_width) ;

F_ECG_idx ← FindPeaks (F_ECG);

Plot(F_ECG_idx , X);

return F_ECG_idx;

4. Results & Conclusions

4.1 Signal Processing

In this section, we will present the results of our algorithm, phase by phase, as described under section 3.2.

- **Raw signal**

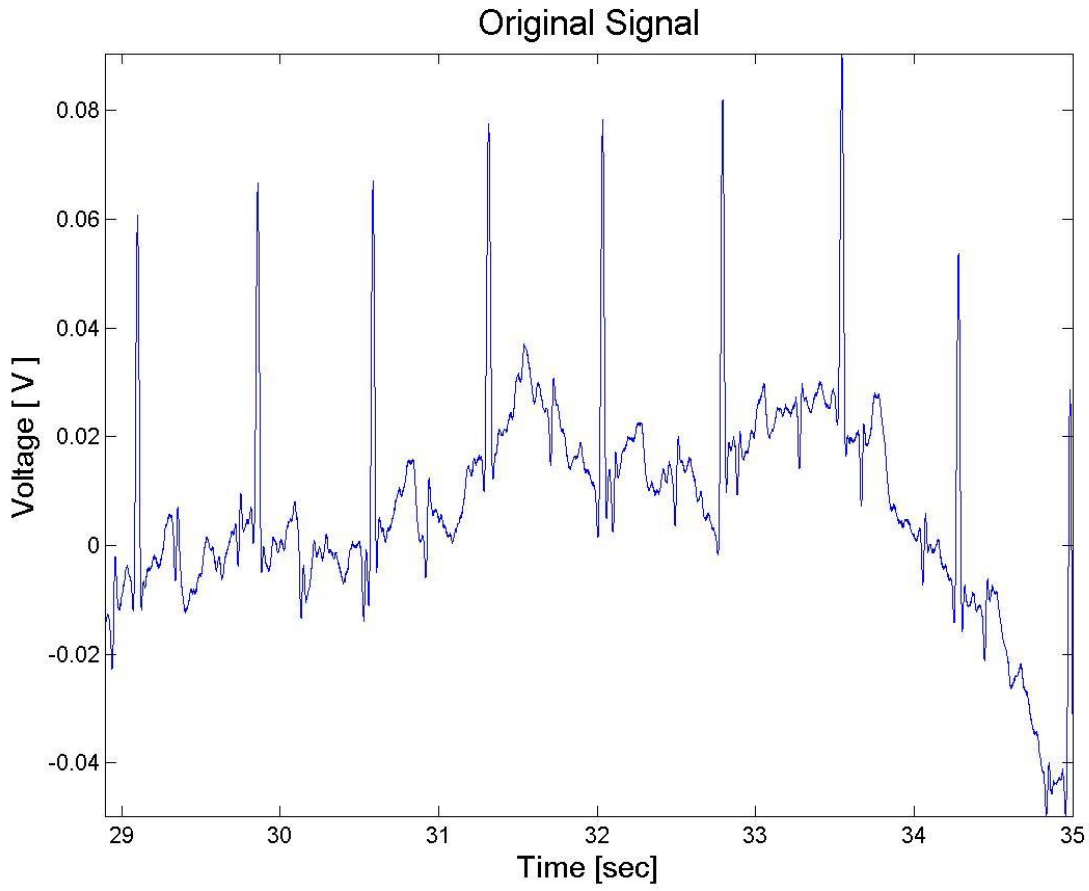


Fig5. Original signal in Voltage verses sample number

- **Preprocessing**

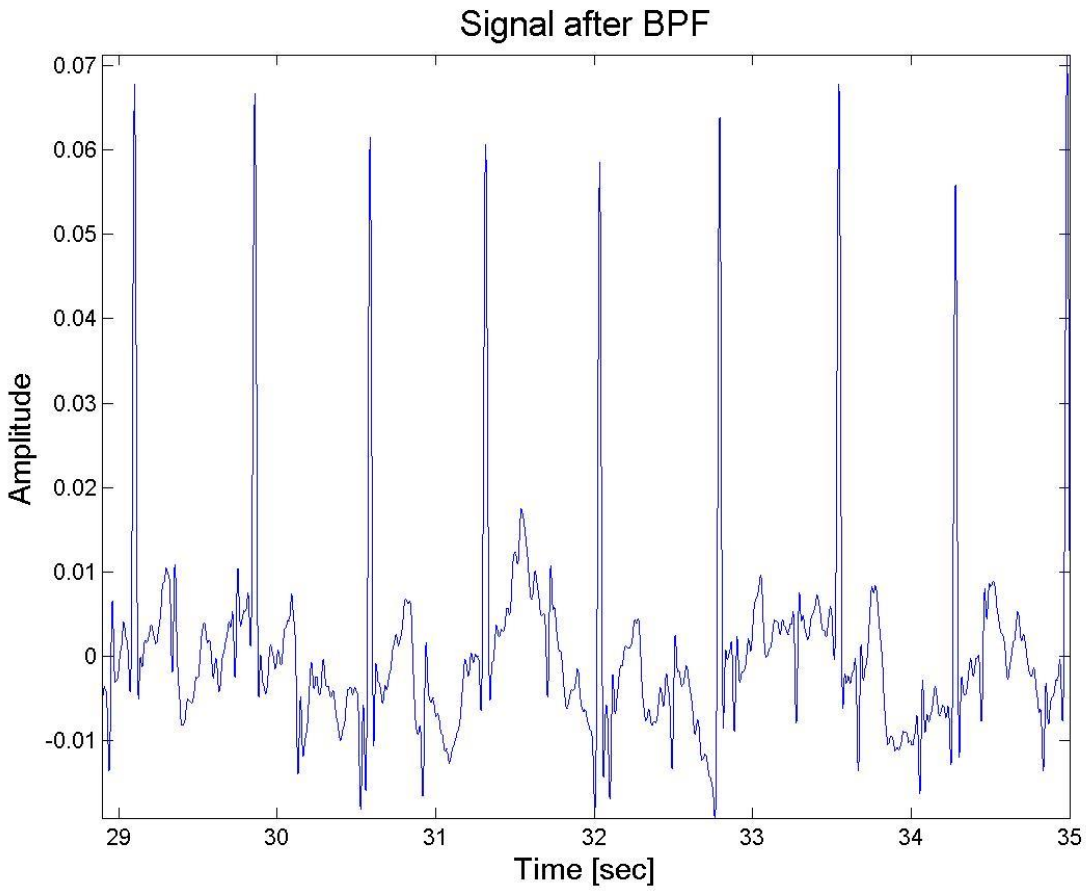


Fig6. Original signal after BPF

- **TVD**

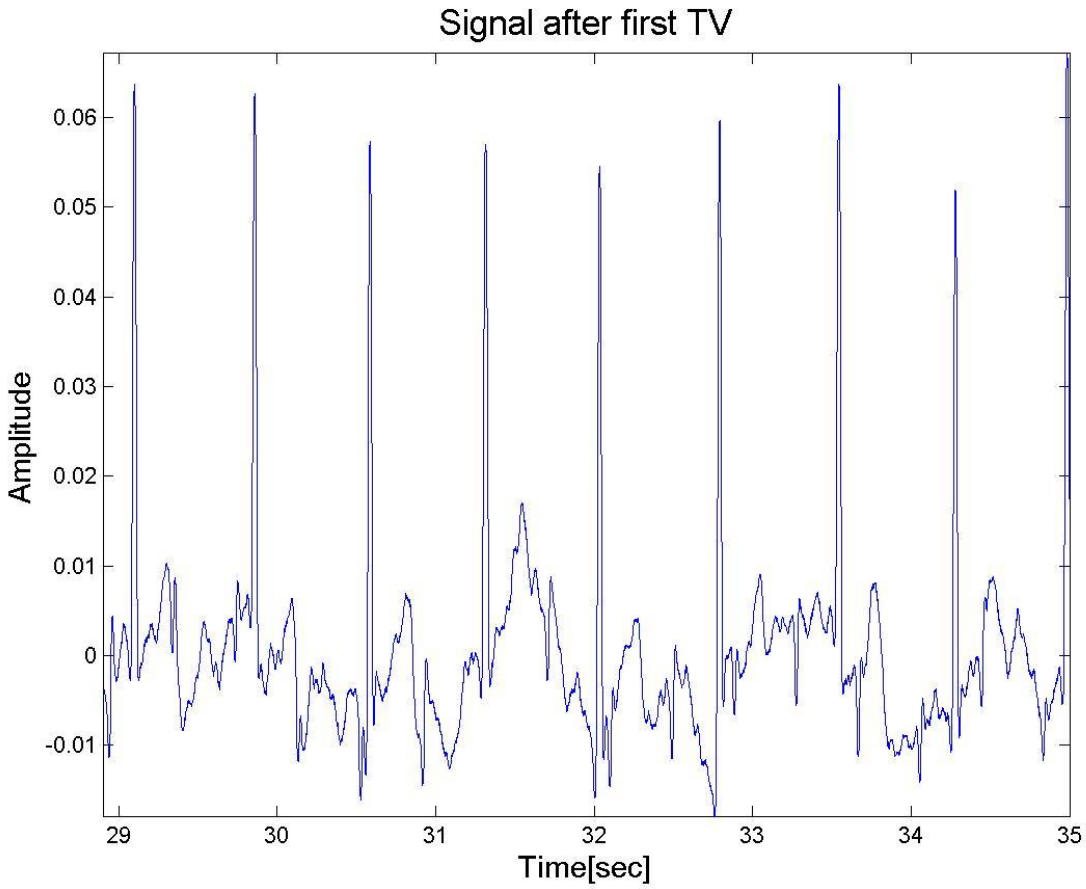


Fig7. signal after BPF & 1st TVD

- **QRS Detection and Maternal ECG Template**

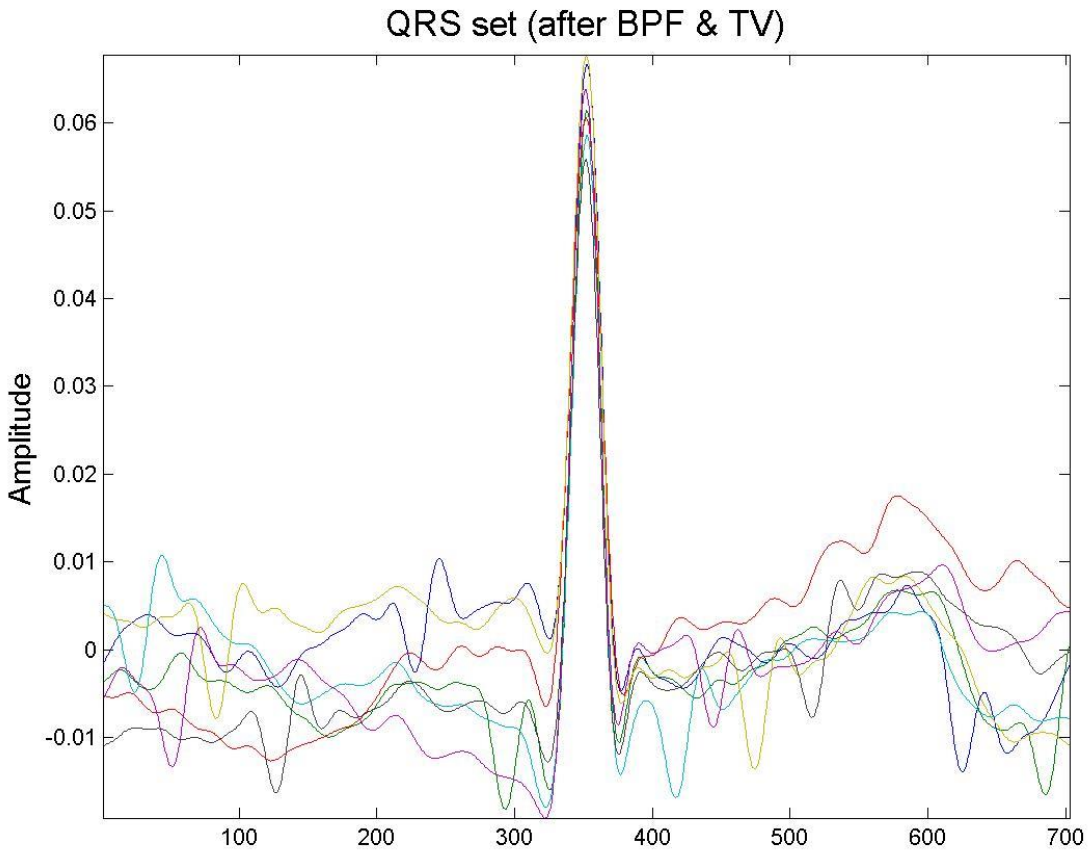


Fig8. QRS pulses set

- **TS_{PCA}:**

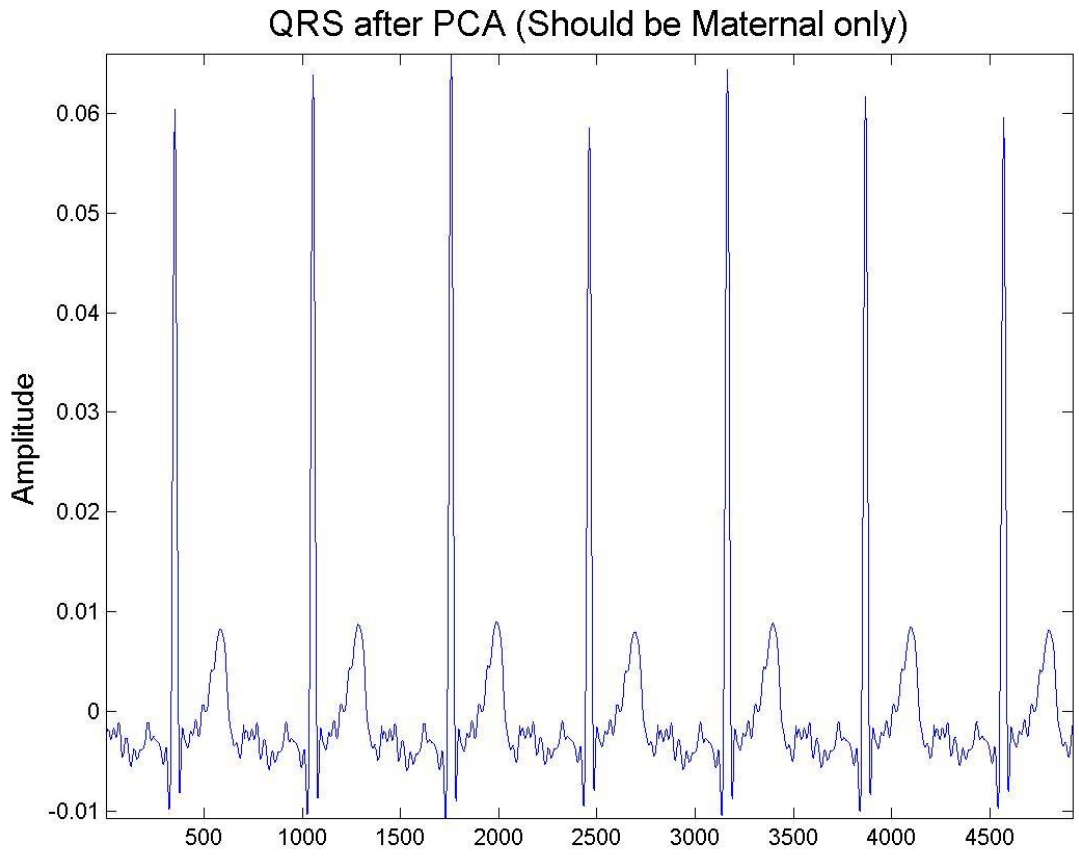


Fig9. QRS pulses set concatenated after PCA

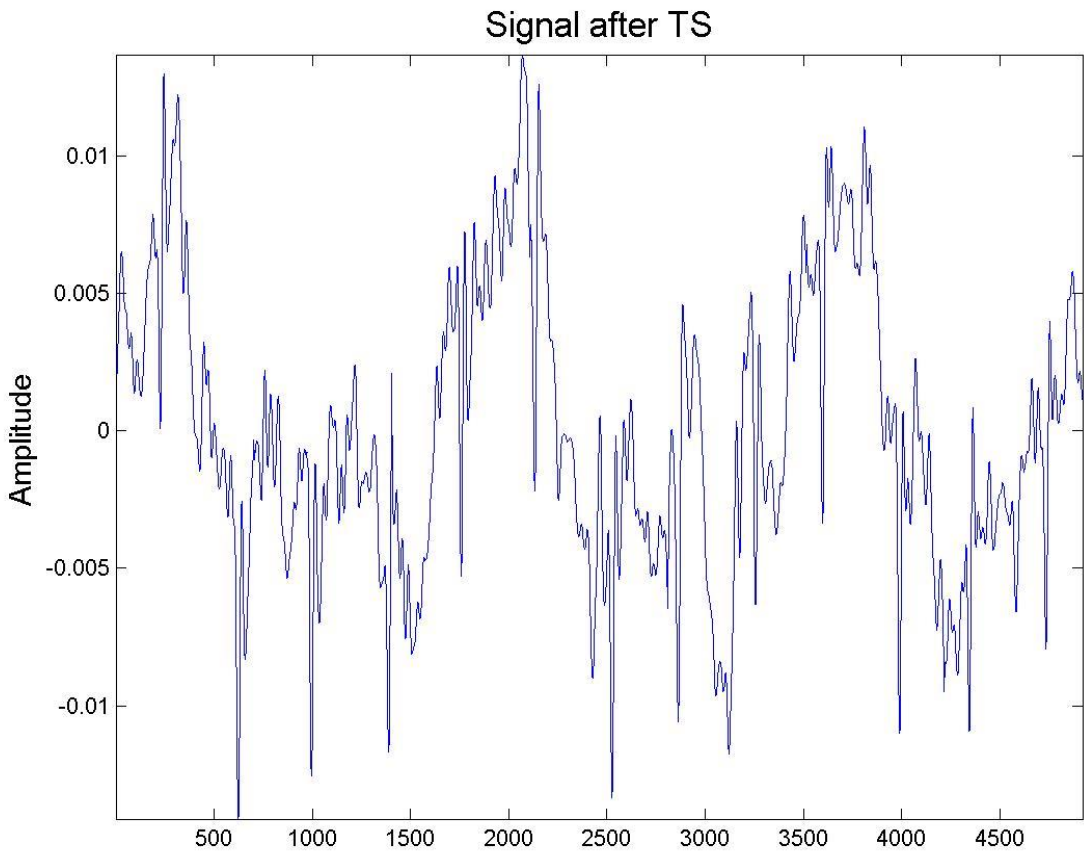


Fig10. QRS pulses after TS (noisy FECG)

- **TVD2**

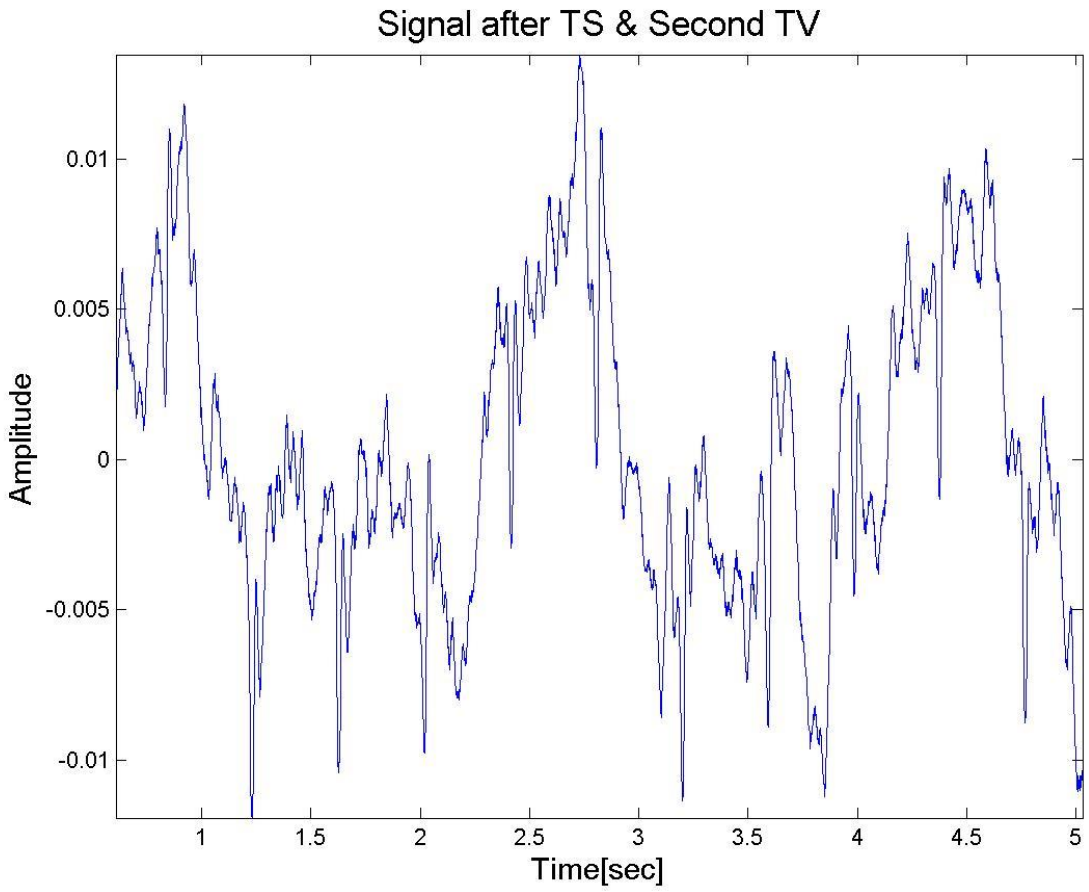


Fig11. noisy FECG after 2nd TVD

- **First order derivation**

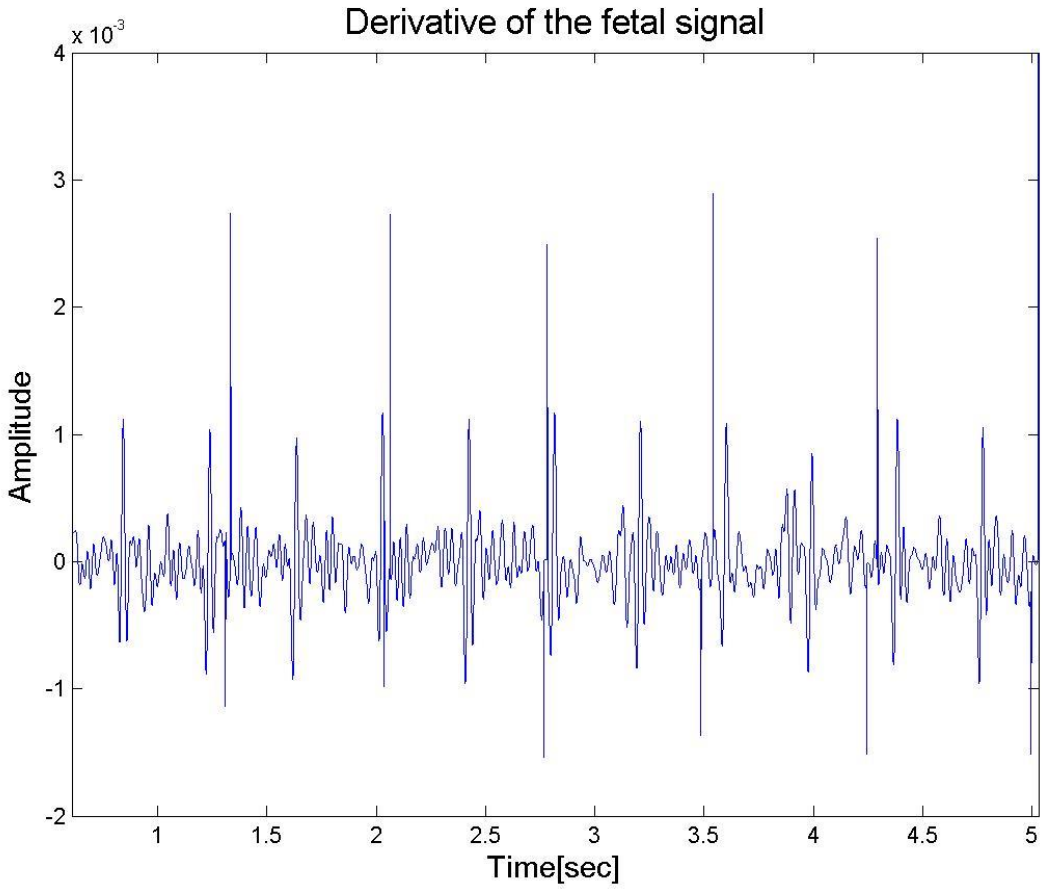


Fig12. FECG after derivation

- **Artifacts cleaning**

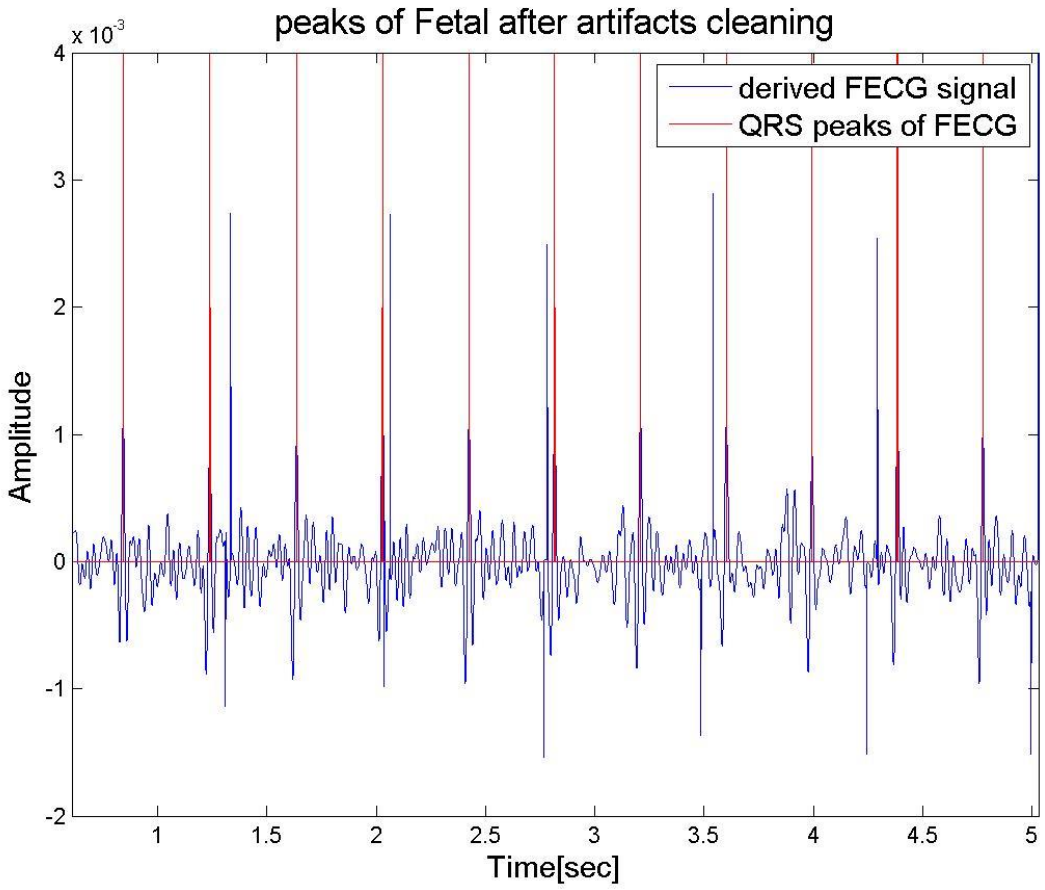


Fig13. QRS Peaks of derived FECG after artifacts cleaning

- **Final Result**

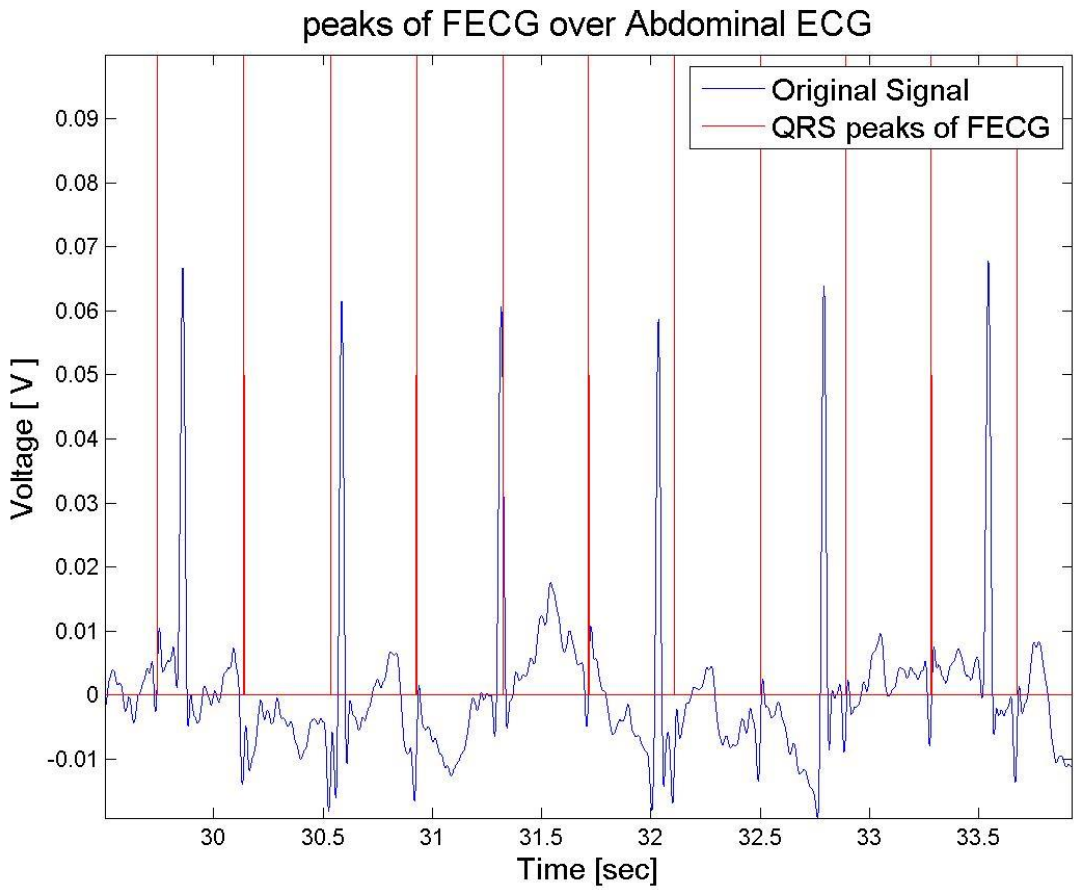


Fig14. QRS Peaks of FECG on top of the original signal

4.2 Additional Results

In favor of gaps-finding in our algorithm, we used different input signals. the following are the final outputs:

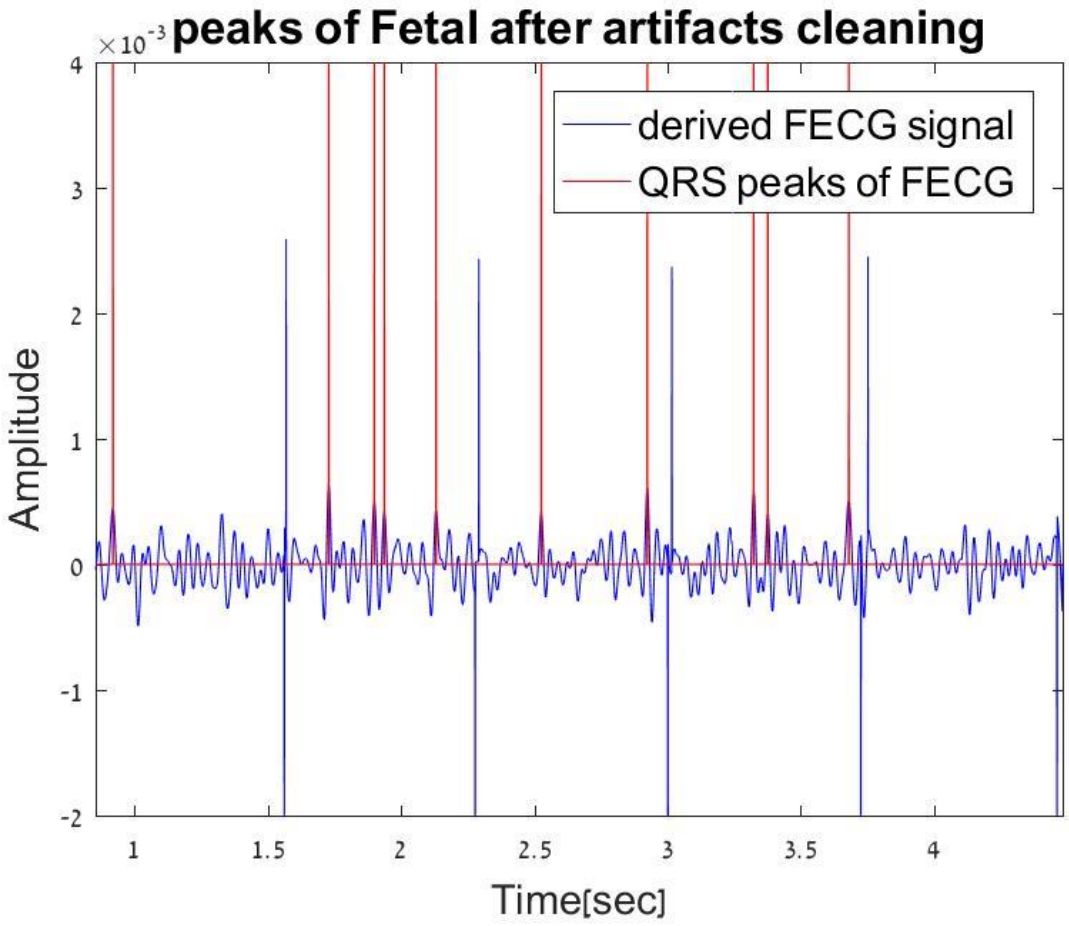


Fig15. Sample 11-lata algorithm result

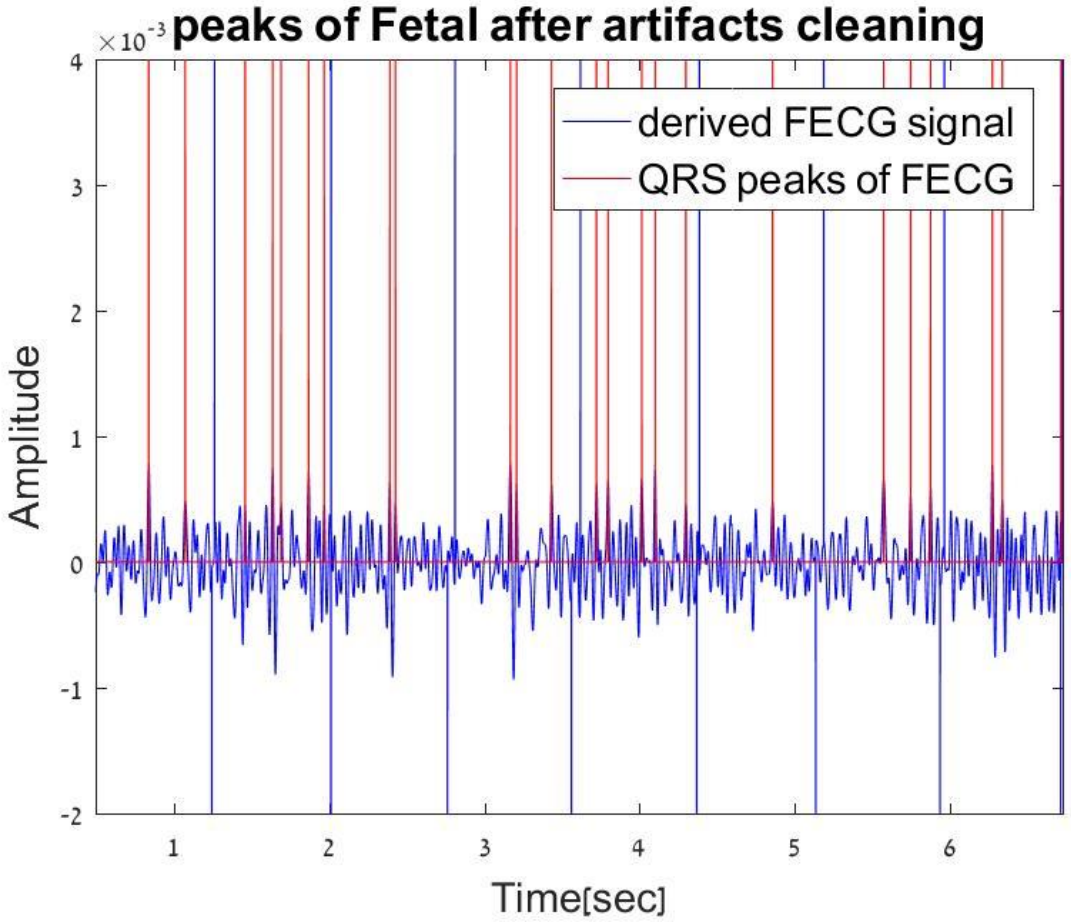


Fig16. Sample 13-lat algorithm result

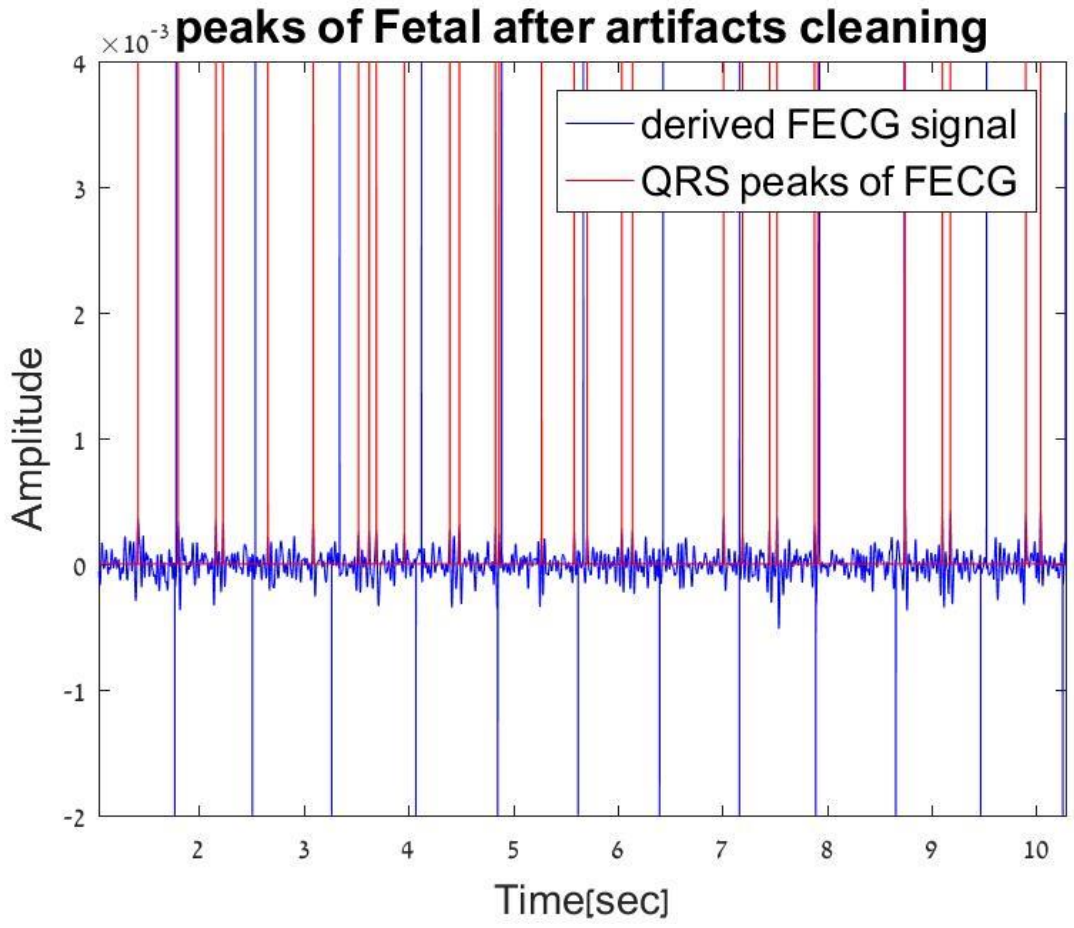


Fig17. Sample 17-lat algorithm result

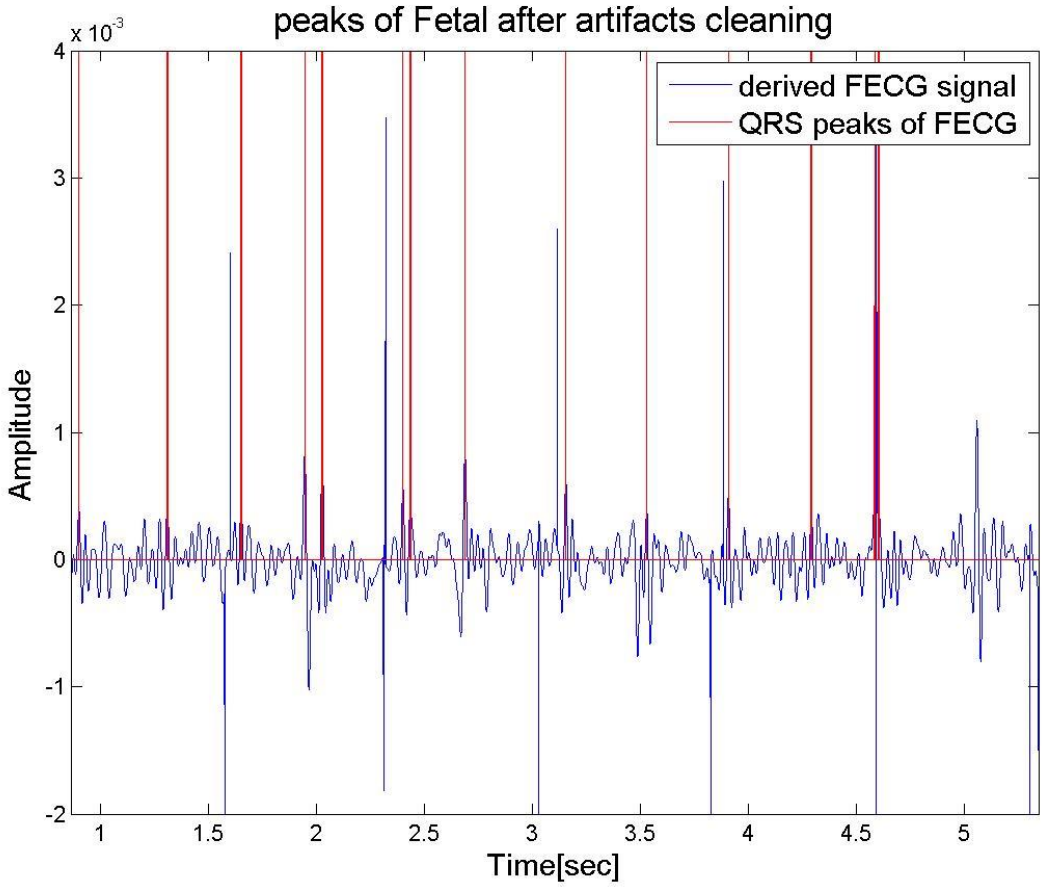


Fig18. Sample 12-lat algorithm result

- The conclusion from the results above is that more effort should be invested artifact cleaning robustness.

4.3 Results Analysis

As can be seen in figure 13, the QRS peaks detection fits the format of FECG – it's more current than the MECG and approximately periodic, as expected to be.

The average of the distance between the identified peaks will give us the approximated period, which will directly lead us to the fetal's heartbeat rate.

As an example, the average period of the signal shown in figure 13 is 393.2 samples.

The sampling rate (as indicated by Dr. Tejman) is 1KHz.

Therefore:

$BPM = \left(\frac{samp\_num}{1KHz} \right)^{-1} * 60 = 152.6[\frac{bits}{\min}]$, which fits an embryo's heartbeat rate!

4.4 Our Innovation

As mentioned under section 3.1, The Sequential Total Variation Denoising idea was taken from an Article written by Kwang Jin Lee and Boreom Lee, which was published on 1 July 2016.

However, trying to implement the above algorithm on **real** abdominal ECG signals didn't match the required results regarding the detection of FECG: instead of clean and clear FECG, we got a very noisy signal, full of artifacts, in which identifying the FECG directly is nearly impossible.

Therefore, an innovative initiative was required to enable the desired achievement.

By examining the output of the original Sequential TVD algorithm, we noticed that the amplitude changing rate of the FECG at the QRS peaks environment were significantly greater than the changing rate of the noises' and artifacts' amplitudes.

In order to use this property to our benefit, we decided to implement a first order derivation on the output of the original algorithm. The derivation gave us a much better result to work with, in which the FECG QRS's are identifiable.

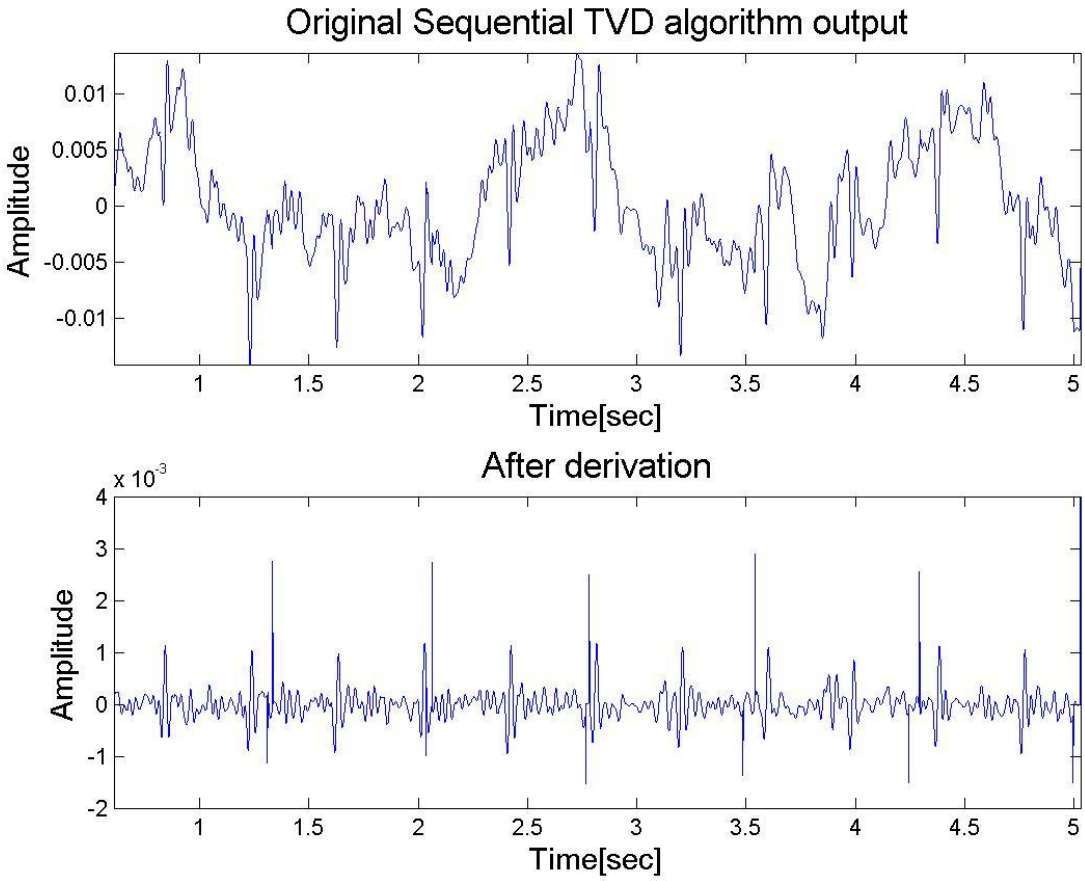


Fig19. QRS Peaks of FECG on top of the original signal

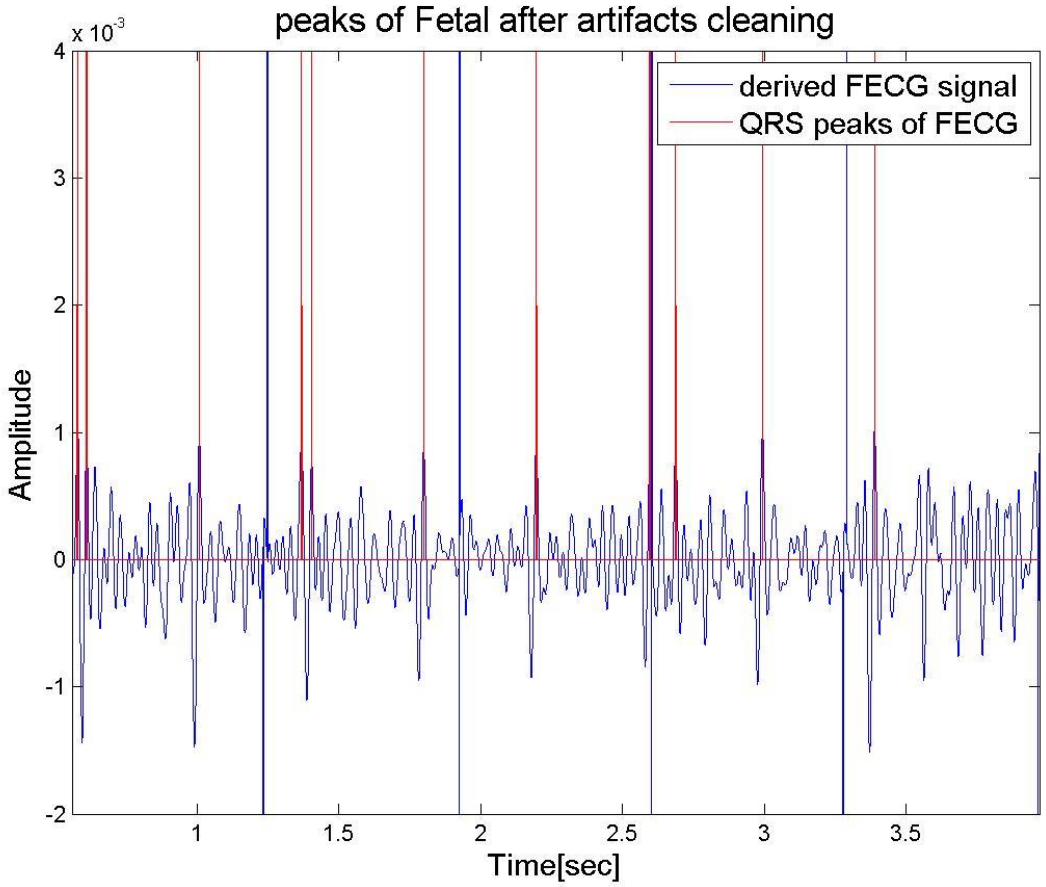


Fig20. QRS Peaks of FECG on top of the original signal

Although the result looks promising, some artifacts and noise are still easily visible, and were needed to be handled in order to successfully achieve our goals.

Accordingly, we decided to apply double thresholding on the signal to remove the influence of the big artifacts and small noises, which we discussed under section 3.2.

Unfortunately, this wasn't enough, since the stitching of the processed QRS after TS and the residual signal from the original signal produced additional artifacts that were undistinguishable from the FECG.

Thence, we created a mask of the indices of the stitching edges, which we used to mask the derived signal.

Locating the peaks of the derived signal after the double thresholding and the mask operation produced the desired result!

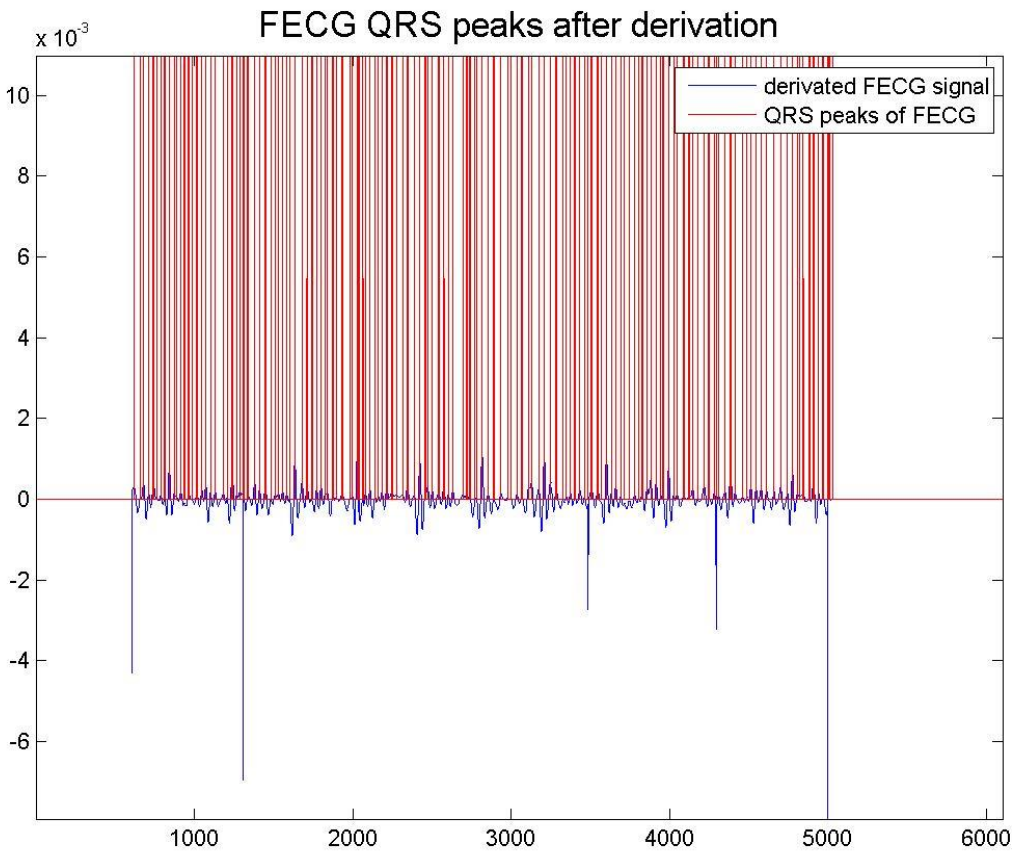


Fig21. QRS Peaks of derived FECG

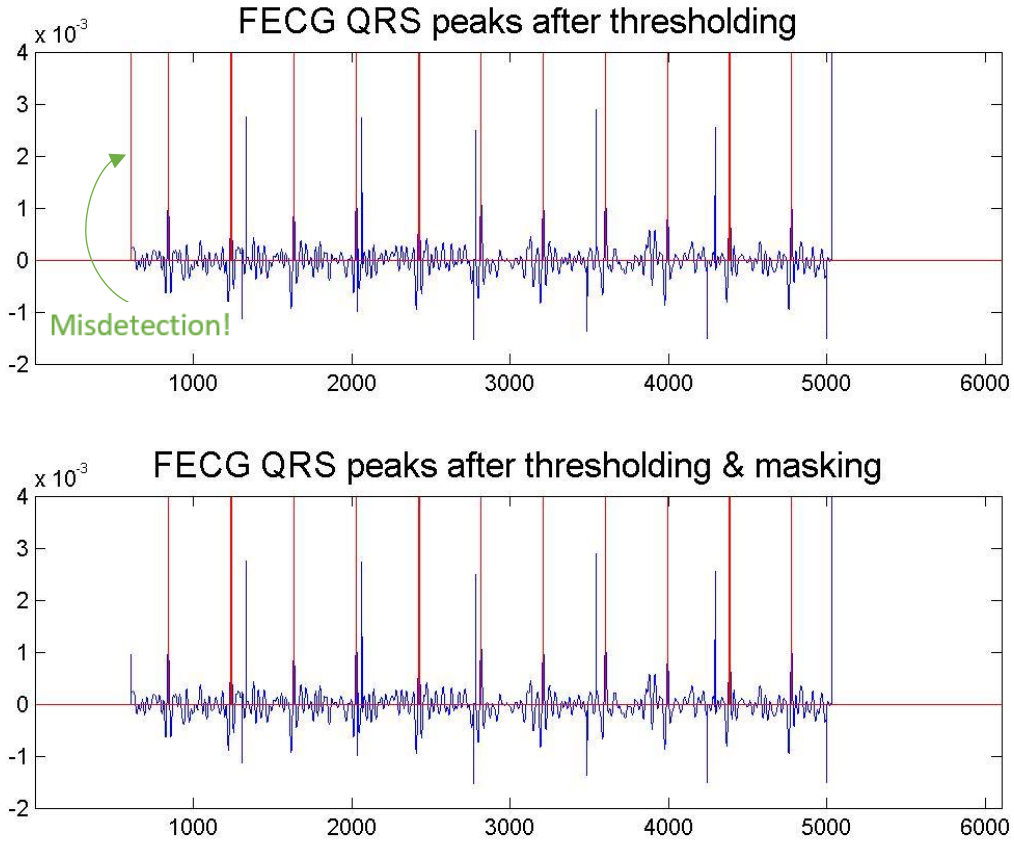


Fig22. QRS Peaks of derived FECG after cleaning

4.5 Gaps

Although the results seem very exciting, there are still several gaps that need to be addressed:

a. Signals quality: the quality of the raw abdominal ECG data base that we worked with was, in most cases, of very bad quality, for the following reasons:

   1. The conditions of the sampling were far from optimal, due to the human factor, e.g. laugh, excitement and movement of the patients, communication between the patient and the sampler, etc. Since the patients are pregnant women, it's extremely difficult to reach accurate and cleaner results, which are fatal to the success of the algorithm.

   2. The data was sampled by a standard ECG sampling machine, and transformed to .mat format. The quality of the machine and the format conversion isn't the state of the art.

   3. Sampling errors and noise.

b. Model mismatch: neither of the set of algorithms that we examined were tested effectively on real ECG signals. Accordingly, there was a big difficulty to adjust the implemented algorithm to fit the characteristics of the real data set.

c. Artifacts: despite the artifacts-cleaning block's implementation, some artifacts are still present in the signal after cleaning. The phenomenon exists since the assumptions made to differ between artifacts and FECG isn't perfect. We assumed that all FECG QRS pulses will have a considerably different amplitude than the artifacts after derivation. In some cases, the artifacts' and the FECG's QRSs' amplitudes are within the same range. The above causes two problems:

   a. Over detection – detection of artifacts as FECG QRS pulses.

   b. Under detection – misdetection of FECG QRS pulses.

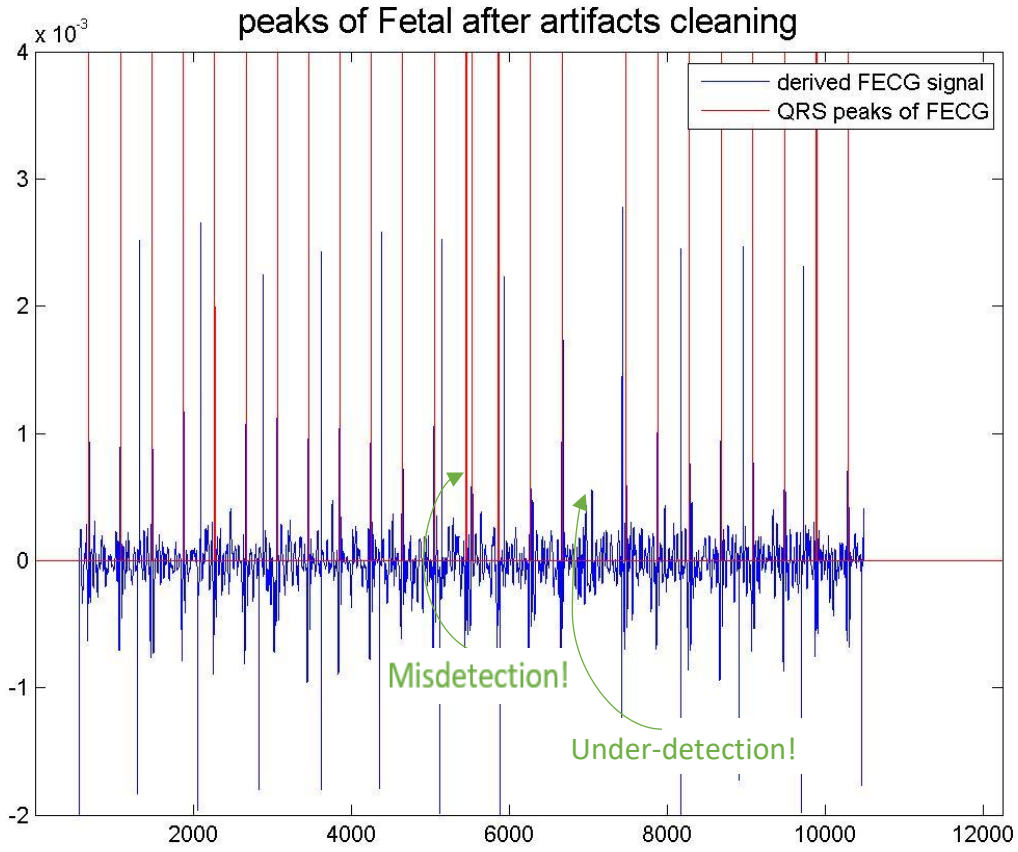


Fig23. Artifacts corrupting QRS Peaks detection

4.6 Conclusions

Considering the analyzed results and the gaps, we got to the following conclusions:

- Identifying the FECG QRS peaks in the abdominal ECG signal is possible! That conclusion is of great importance, since that wasn't a certainty, and it's the key to enable the required synchronization of the cardiac MRI to the FECG.
- The major reason for detection failure of the FECG, in most cases, was the lack of ability to differ it from the noise. More advanced ECG sampling machines and better sampling conditions will lead to better samples, that will unquestionably provide better results.
- Additional effort is needed to improve the outcome of the algorithm. Possible suggestions:
    - A more robust artifacts cleaning, that takes more characteristics of the FECG signal into consideration, e.g. heartrate period consistency.
    - Reconsider the usage of certain blocks used for noise filtering – the total variation doesn't seem to contribute to the process, due to the model mismatch to the real ECG signals. Other filtering methods should be discussed and tested instead.
- An efficient algorithm will possibly provide the ability to analyze the abdominal ECG signal and detect the FECG QRSs in real time. More research is required to check the feasibility of the real-time concept.

Chapter2 – Project B

1. Introduction

Project A's results proved the possibility of identifying FECG QRS pulses. However, the proposed solution suffered from several gaps:

- Lack of robustness - The algorithm worked on a relatively small set of signals
- Thresholding method:
    - Manual calibration was required for each signal.
    - Non-linear processing method.

Hence, a more robust algorithm was needed in order to obtain better results.

1.1 Solution

As could be observed by project A's conclusions, the derivation step enhanced the base algorithm implementation.
Therefore, the proposed solution for project B will be based on the signal after the derivation phase.

In order to overcome the lack of robustness and the non-linearity, an alternative QRS classification algorithm was developed. The algorithm took advantage of the following principles:

- Auto-correlation
- Cross-correlation
- KNN classification
- MSE minimization

2. Theoretical background

2.1 Articles review

In order to achieve better results, several articles concerning FECG separation methods were reviewed. The reviewed articles were chosen by the following criteria:

- State of the art (new articles).
- Single lead.
- Template subtraction.

The following paragraphs will contain a review of 3 major articles that were reviewed:

2.1.1 Blind separation of fetal ECG from single mixture using SVD and ICA

Date: 2003
Writers: Ping Gao, Ee-Chien Chang, Lonce Wyse
Main ideas:

Using a single channel recording
Tested on both simulated and real recordings
SVD solely is insufficient. ICA is also required for good results.

Algorithm:

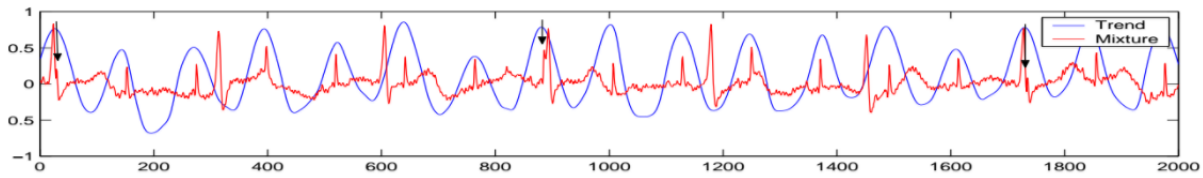
Compute the spectogram of the raw ECG. Using stationarity and independency assumptions, the spectogram S can be represented by $S = u_m v_m{}^t + u_f v_f{}^t + n$ Where u_m and u_f are the heartbeat trends of the mother and the fetal respectively (n stands for noise).
Choosing the right window, u_m and u_f should behave aproximatley sinousoidaly, where each cycle is a heartbeat. v_m and v_f are then the aproximation of the spectrum of the ECG complex.
Minimize the noise and decompose S to two independent components using SVD and ICA.

Prons:

Relatively simple - very few steps comparing to our algorithm.

Cons:

In both the recorded data and the simulated data, the amplitude of the FECG is by far more prominent than in the case of the samples that we got from Shai (the "small" red peaks):



Taking that into consideration - we doubt that this method would provide good results on our signals, since the FECG is expected to be the 2nd most dominant component in the ICA.

The method is very sensitive to Initial conditions and window size selection for the Spectogram (a window should contain precisely one complex of the maternal ECG).
The solution involves an iterative minimization algorithm (e.g fastICA/ gradient decend) which probably results in greater latency and complexity, while our algorithm currently (without TV) relies on linear operation.

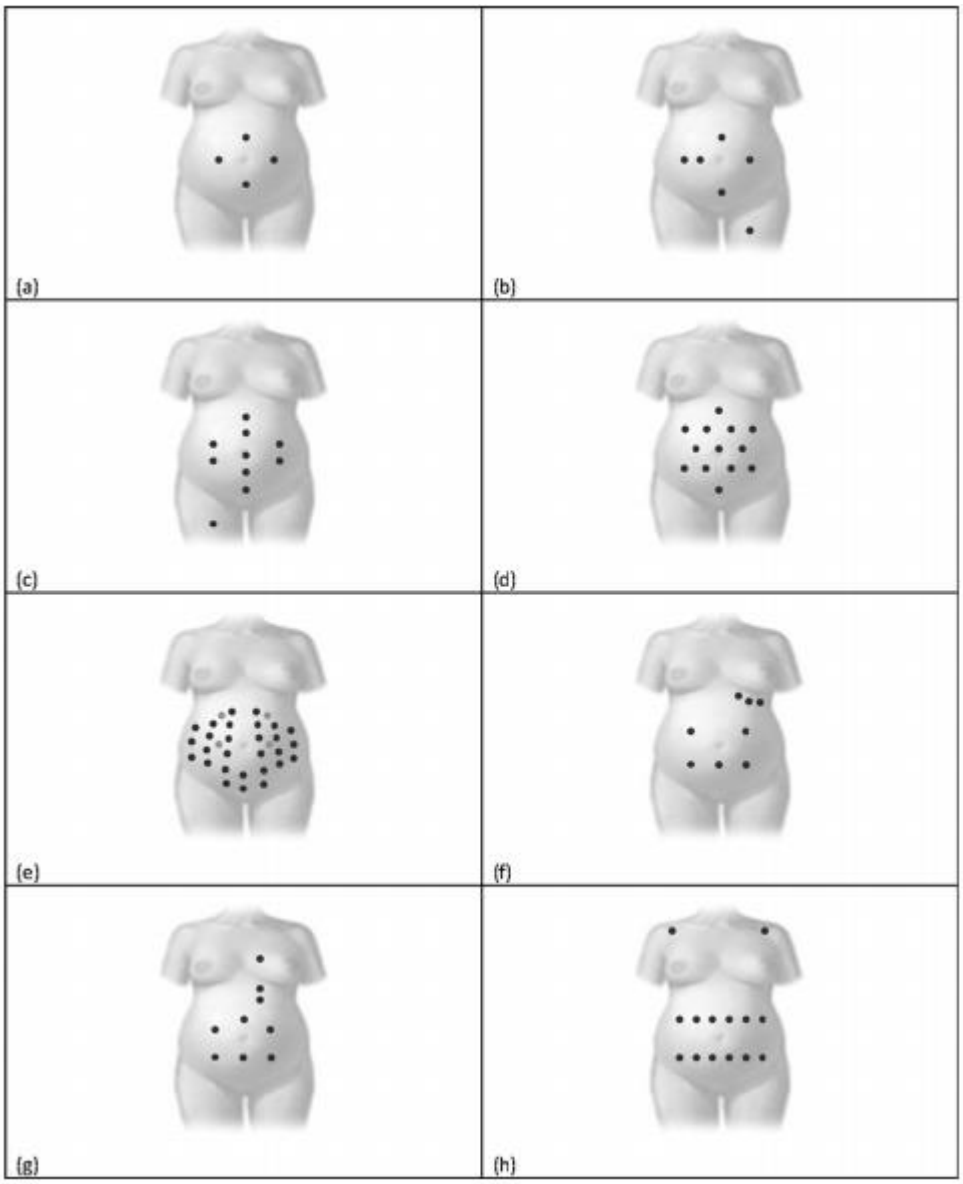
2.1.2 Noninvasive fetal Electrocardiography: An overview of the signal electrophysiological meaning, recording procedures and processing techniques.

Date: 2015

Writers: B.Eng Angela agostinelli, Marla Grillo, M.D Alessandra Biagini, B.Eng Corrado Giuliani, M.D Luca Burattini, B.Eng Sandro Fioretti, B.Eng - Ph.D Francesco Di Nardo, M.D Stefano R.Giannubilo, M.D Andrea Ciavattini, B.Eng-Ph.D Laura Burattini

Main ideas:

- Prolog:
  - Cardiotocography - monitoring the fetal using two separate methods:
    i. Ultrasound, to detect the fetal's heart motion.
    ii. Pressure measurement of the maternal uterine
    - Usually a non-invasive procedure
    - Used to detecting early stages hypoxia
    - Prons: reduces the chances of cerebral palsy
    - Cons: leads to many unnecessary cesarean births
  - Non-invasive fECG monitoring:
    - Prons: non-invasive (more safe)
    - Cons: Low SNR
- FECG Morphology:
  - The fetus cardiac electrical axis (VCG) points to the right ventricle - opposite to an adult ECG , which is to the left
- FECG acquisition applications:
  - Fetal growth parameters - can be inferred from the duration of the P and QRS segments
  - Supraventricular arrythmias - can be identified by widened QRS segment.
  - Detecting negative energy balance due to lack of oxygen, based on ST segment variability
  - Detecting fetal distress by identifying a prolonged variations of the FECG, in particular on the ST segment. Instability for longer than 15 second - might lead to critical states.
- Noninvasive FECG recording techniques:
  - Electrode configurations:
    - No optimal cofiguration was proven
    - #leads>8 - optimizes feasibility of signal acquisition
    - #leads<8 - optimizes simplicity
    - Two main configurations:
      - Pure Abdominal electrode configurations (a-e)
      - Mixed electrode configurations(f-h)



- FECG extraction algorythms:
  - Lead components:
    - fECG:
      - BW 0.5-100Hz (typically 0.5-45 is good enough)
      - Amp Typically around 60uV
    - amEFG:
      - BW similar to FECG
      - Amp tipically 100-150uV (can be up to 10 times greater than the FECG's amplitude)
    - aNoise:
      - Mixture of interferences:
        - Physiological noise (electromyogram, electroencephalogram, respiration)

        - Non-physiological noise (instrumentation noise, poor cable shielding, electrode-skin interface)
- Noise components:
    - lfNoise(0-0.5 HZ) - caused by respiration
    - hfNoise(45 and above) - caused by the line interference, electromyogram and electroencephalogram.
    - ibNoise (0.5-45 Hz) - caused by all the above.
        - Can't be removed using BPF
- Main steps:
    - Abdominal signal prefiltering - applying a BPF to remove lfNoise and hfNoise
    - amECG cancelation - subtraction of an estimated amECG from the filtered signal:
        - Pure abdominal electrode (mixed fECG and amECG only):
            - ICA
            - Template subtraction
        - Mixed electrode (mixed fECG and amECG + pure amECG):
            - Adaptive filtering

$$aLead = fECG + amECG + aNoise$$

$$faLead = fECG + amECG$$

- Cancelation methods:
    - ICA
        - Multilead abdominal recordings
        - Acquired Signal is a linear combination of independently distributed sources
        - The more recordings - the better the extraction is
        - Cons: feasibly difficult + not comfortable to pregnant women.

$$\overline{\mathrm{fECG}} = \mathrm{S} \times \overline{\mathrm{faLead}},$$

    - TS (template subtraction)
        - Relies on a single lead
        - Takes advantage of the maternal repeatability
        - The PQRST segments of the maternal are being located
        - A template is produced (usually using averaging)
        - All beats are reconstructed using the template, and then concatenated before the subtraction.
    - Adaptive filtering
        - Kalman filter
            - using two input measurements:
                - faLead
                - mECG (highly correlated to the amECG)
            - Find a trasnformation from mECG into amECG, using MMSE between mECG and faLead.

        - Neural network
            - Input - mECG
            - Target - faLead

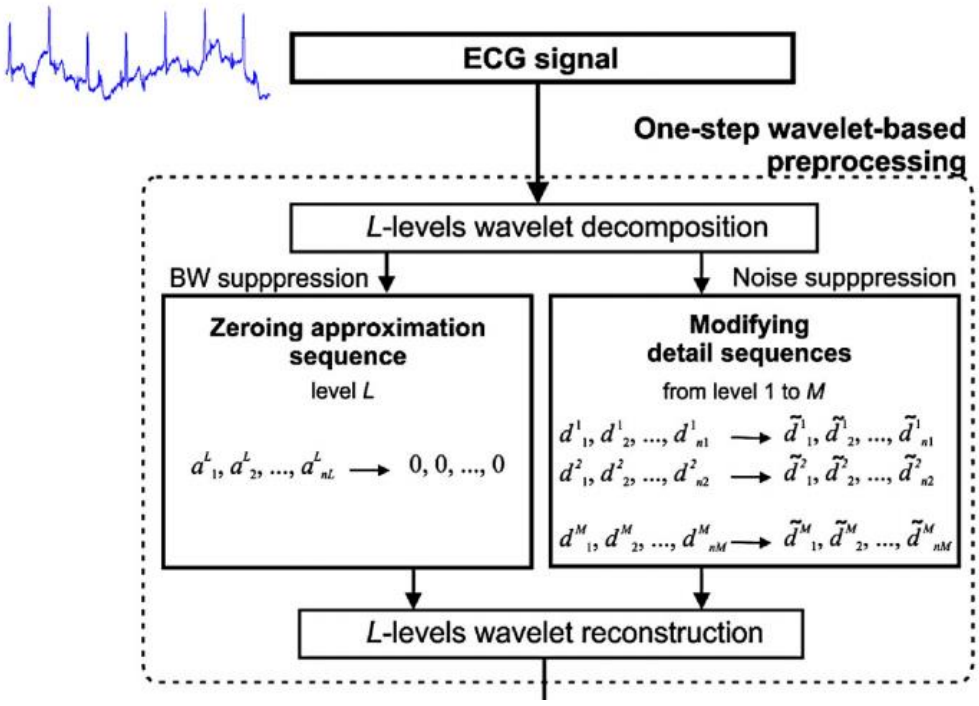
- Former studies achievements:
    - Except for one, all studies developed and proved their results over a very few recordings
    - The largest study in terms of recordings used ICA technique, over 20 pregnant women (at week 38).
        - Achieved 85% success rate

- Sampling Technical information:
    - two physical layers are determining the conductivity:
        - Amniotic fluid - highly conductive
        - Vernix caseosa - poorly conductive. It is formed between 28-32 week of gestation, making the FECG sampling nearly impossible
            - It disolves between 37-38 week of gestation
    - The position of the fetal inside the mother's womb isn't constant and varies during the pregnancy. It might even change during a single acquisition. Hence, no standard protocol for electrodes configuration can be inferred as optimal.

2.1.3 Efficient wavelet-based ECG processing for single-lead FHR extraction.

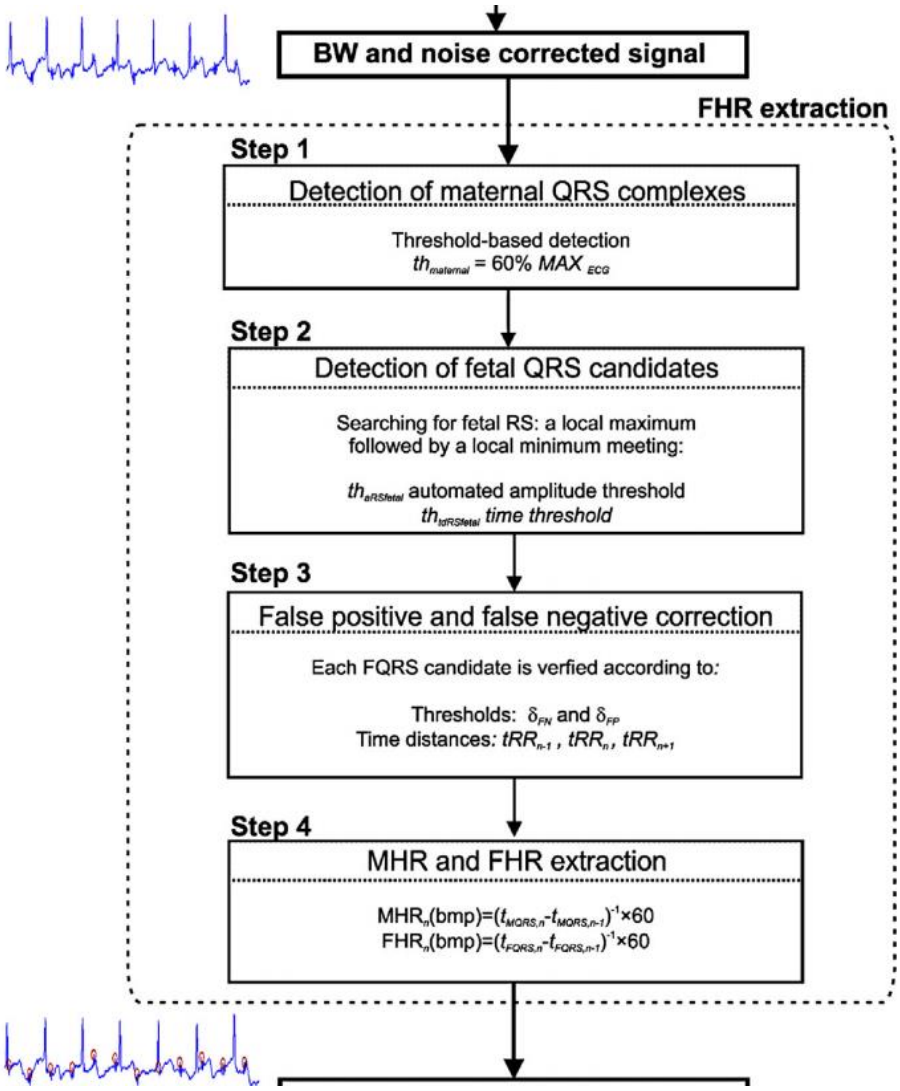
Date: 2013
Writers: E. Castillo , D.P. Morales, G. Botella , A. García, L. Parrilla , A.J. Palma

- Prolog:
  - Main idea: extracting FECG from single lead using wavelet transformation
    - Relatively feasible in hardware comparing to complex software implemented algorithms
    - More efficient in denoising than FIR
    - Separates the different signal components better than fourier transform.
- AECG preprocessing:
  - The FECG suffers from various contaminations:
    - Power line interference - narrow band noise (1HZ) around 50-60HZ
    - Baseline wondering - around 0.15-0.3 HZ
  - Some software can efficiently solve these problems. However, they aren't very feasible in terms of hardware implementation.
  - DWT is suitable for the desired filtering and also makes hardware implementation feasible.
- Algorithm steps:
  - Wavelet based preprocessing:



    - Wavelet decomposition is used to eliminate BW and noise from the signal

  - FHR extraction:



- Step 1:
  - Proportional threshold is implied to find the maternal R peaks
- Step 2:
  - Look for local maxima followed by local minima (RS peaks), in between two consequent maternal R peaks (detected in step 1)
  - Two thresholds are applied on the fetal RS peaks candidates:
    - Amplitude threshold, assuming that the FRS peaks are of greater magnitude than other components
    - Since the first threshold doesn't always apply, a time based threshold is applied to assert the time distances between two consequent FRS peaks.
- Step 3:
  - Validating the correctness of the detections from step 2 by comparing contiguous FR-R peaks, and by so identifying FP and FN detections.
- Step 4:
  - Extracting the Maternal and fetal BPM by the time differences between two consequent R-R detections.

- Fixed point modeling:
  - The algorithm was implemented using 16bit fixed-point based architecture
  - The above representation method was used both for the inputs/outputs.
  - DWT was also tested using 16bit word's length, and resulted in acceptable outcomes (~67dB SNR).
  - In some cases, the fixed point constraint produces liability, e.g when calculating logarithms, square roots, etc.. These require special algorithms for HW implementation.
- Results:
  - Signals resources:
    - DaISY database: 8 channles of skin potential recordings of pregnant woman, sampled @250 Hz, 10 seconds long
      - 3 thoracic
      - 5 abdominal
    - Synthetic AECG: 3 types of synthetic AECG signals, elaborated from ecg.m(Matlab function)
      - 21.6 seconds long
      - Three lengths of signals:
        - 5400 - 250sps
        - 10800 - 500 sps
        - 21600 - 1000sps
    - Phisionet DB:
      - 2 thoracic
      - 4 abdominal
      - 1Ksps
      - 60 seconds long
      - 0.01-100 Hz BW
      - 16 bit ADC
    - Abdominal and Direct Electrocardiogram DB:
      - Each recording is made of:
        - 4 differential signals acquired from the maternal abdomen
        - 1 reference direct fetal electrocardiogram from the fetus head
      - 1Ksps
      - 16 bit resolution
      - 5 minutes long
      - 1-150 Hz
  - Wavelet preprocessing:

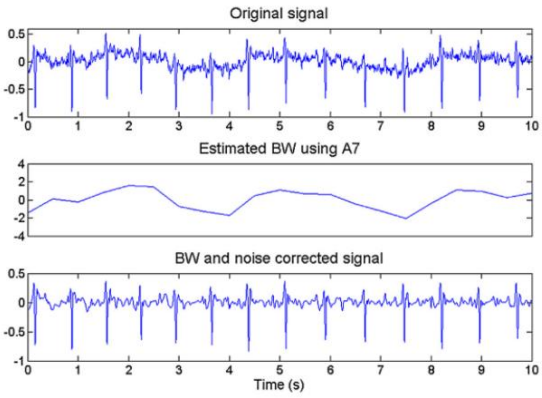


Fig. 2. Lead-4 *DaISy Database*, estimated BW from A_7 and BW and noise corrected signal using the proposed preprocessing model.

- BW suppression
  - The BW suppression efficiency was evaluated on synthetic AECG signals, composed of a sine wave and DC level using frequencies between 0.15-0.31 Hz.
  - Used levels 1-12 of a 6 degree daubechies.
  - Better BW suppression is achieved as the sampling frequency grows.
- Noise suppression
  - Wavelet parameters were tested and evaluated with respect to the optimization of the SNRIMP:

$$SNRIMP[dB] = SNR_{output} - SNR_{input}$$
$$= 10 \log \frac{\sum_n |\hat{f}_n - x_n|^2}{\sum_n |\hat{x}_n - x_n|^2}$$

    - All different parameters configurations produced more or less the same results
  - Soft thresholding was chosen for the preprocessing procedure, since it produced better results in terms of distortion implied to the signal (it doesn't add any distortion, in contrast to hard thresholding).

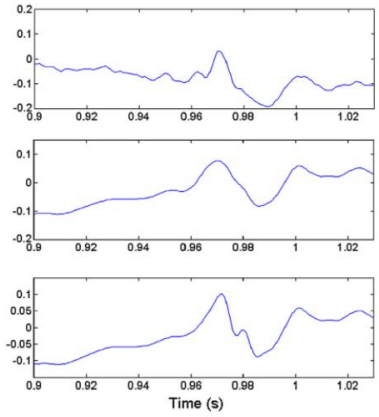


Fig. 3. Original signal (top plot) preprocessed signal using soft thresholding (central plot) and preprocessed signal using hard thresholding (bottom plot).

- Simultaneous BW and noise suppression
  - Several different parameters and wavelet configurations were tested to simultaneously remove BW and noise

- No big difference was observed regarding the SNRIMP of the different configurations.
- The signals (noise and ECG) were taken from MIT's database.
- Due to BW and noise in the supposedly clean ECG signals taken from the MIT DB, comparison between the original and processed signal were mostly qualitative, and less quantitative.

▪ FHR extraction

- The evaluated abdominal leads were taken from the Diasy database
- Only 2.1% of FNs were observed (out of 48 FQRSs)

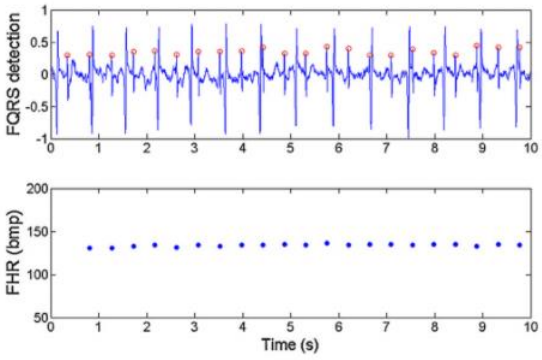


Fig. 4. Lead 1 *DaISy Database* and FQRS detected, and FHR monitoring.

- An input buffer and signal streaming were also used to simulate realtime FECG exraction.

Table 5
Evaluation results for real-time FHR extraction using *PhysioNet Database*.

	T FQRS	Method results			Parameters		
		TD FQRS	FN	FP	Se (%)	PDV (%)	Acc (%)
Non-Invasive Fetal ECG Database recordings (1 min)							
771 Ab-1	150	145	5	1	96.67	99.32	96.02
746 Ab-1	150	140	10	3	93.33	97.90	91.56
840 Ab-1	161	153	8	4	95.06	97.45	92.73
840 Ab-3	161	153	8	6	95.03	96.22	91.62
906-Ab-1	145	142	3	1	97.93	99.30	97.26
Total	767	733	34	15	95.57	97.99	93.73
Abdominal and Direct Fetal ECG Database recordings (5 min)							
r01 Ab-1	646	636	10	6	98.45	99.07	97.54
r04 Ab-3	634	619	15	21	97.63	96.72	94.50
r07 Ab-2	628	616	12	7	98.09	98.88	97.01
r07 Ab-4	628	623	5	9	99.20	98.58	97.80
r10 Ab-1	662	649	13	15	98.03	97.74	95.86
Total	3198	3143	55	58	98.28	98.19	96.53
Non-Invasive Fetal ECG Database recordings (15 min)							
906 Ab-1	2012	1974	38	88	98.11	95.73	94.00
906 Ab-2	2012	2004	47	31	97.71	98.47	96.25
Total	4024	3978	85	119	97.91	97.10	95.12

Prons:

- Good looking success rate (~93.7%)
- Hardware implementation feasability
- Relatively simple model

Cons:

- Many parameters to adjust (not enough robust)
- Wavelet for denoising purposes seems a little excessive (simple BPF seems to produce good-enough results)
- Results were not presented over a big-enough number of data sets (given the amount of data base indicated by the authors).

Mathematical background

Auto-correlation

Autocorrelation is the correlation of the signal with a delayed copy of itself as a function of the delay portion. Informally, it is the similarity between observations as a function of the time lag between them.

Given a signal $f(t)$ the continuous autocorrelation $R_{ff}(\tau)$ is most often defined as the continuous cross-correlation integral of $f(t)$ with itself, at lag τ :

$$R_{ff}(\tau) = \int_{-\infty}^{\infty} f(u)\, f^{*}(u-\tau)\, du$$

Cross-correlation

Cross-correlation is a measure of similarity of two signals as a function of the displacement of one relative to the other.

For continuous functions f,g, the cross-correlation is defined as:

$$(f \star g)(\tau) = \int_{-\infty}^{\infty} f^{*}(t)\, g(t+\tau)\, dt$$

MSE

The MSE (Min Squared Error) assesses the quality of an estimator or a predictor.

If $\hat{Y}$ is a vector of n predictions and Y is the vector of observed values of the variable being predicted, then the within-sample MSE of the predictor is defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

KNN

The K-nearest-neighbors algorithm is a non-parametric method, used for classification.
The input consists of the k-closest trainings examples in the feature space.
The output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its K-nearest neighbors.

3. Implementation

The concept of correlation as a method for BPM estimation and QRS detection was taken from the speech processing domain. We came across this concept during a lab in speech processing, at which we participated as part of our bachelor's-degree studies.

3.1 Algorithm Architecture

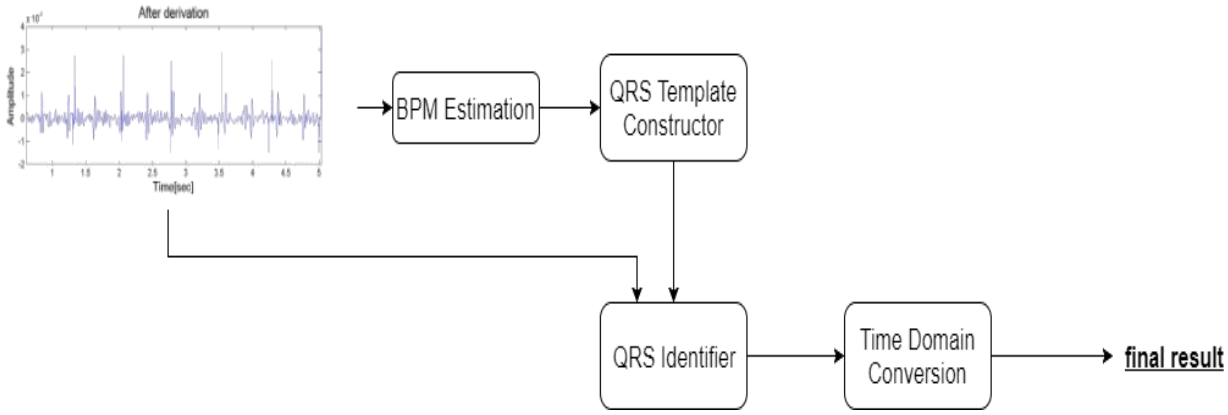


Fig24. Project B's Architecture

3.1.1 BPM (Beats Per Minute) Estimation

Auto-correlation was performed over the noisy FECG signal. The fetal BPM was estimated as the first positive maxima of the auto-correlation that fits fetals' heart rate.

According to Dr. Shai and the reviewed articles, a fetal BPM varies from 120 to 180 [bits/minute].

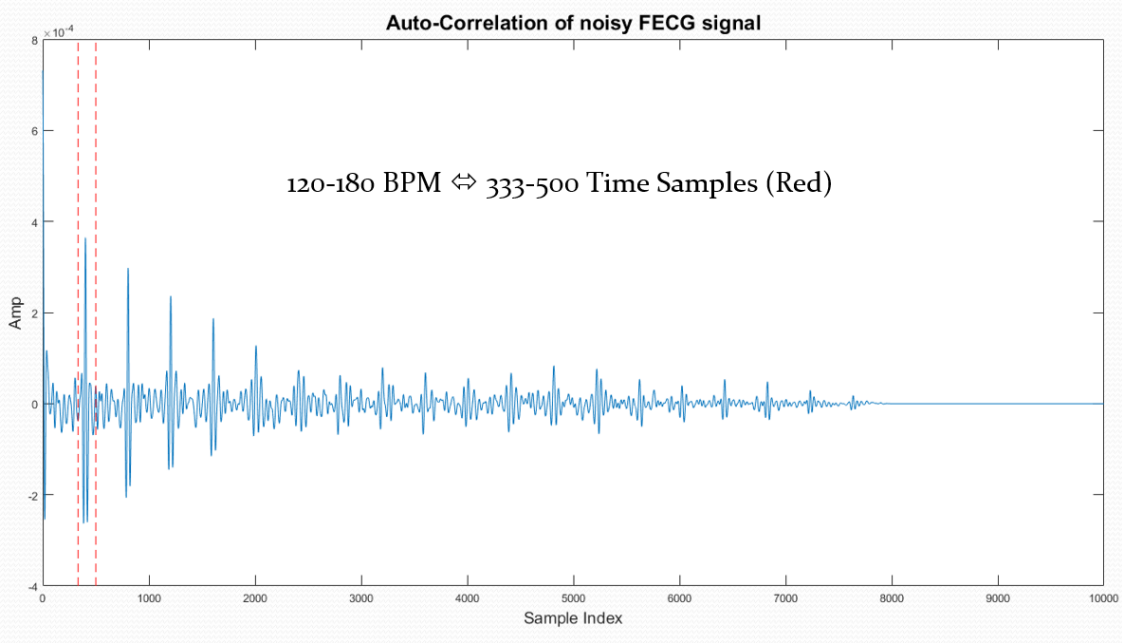


Fig25. Auto-correlation of noisy FECG signal

3.1.2 QRS Template Constructor

1. Multiple fetal QRS pulses were manually chosen from good signals.
2. A typical QRS pulse was generated by averaging over the above pulses.
3. A fetal-QRS-pulses train was generated using the estimated fetal BPM and the typical pulse.

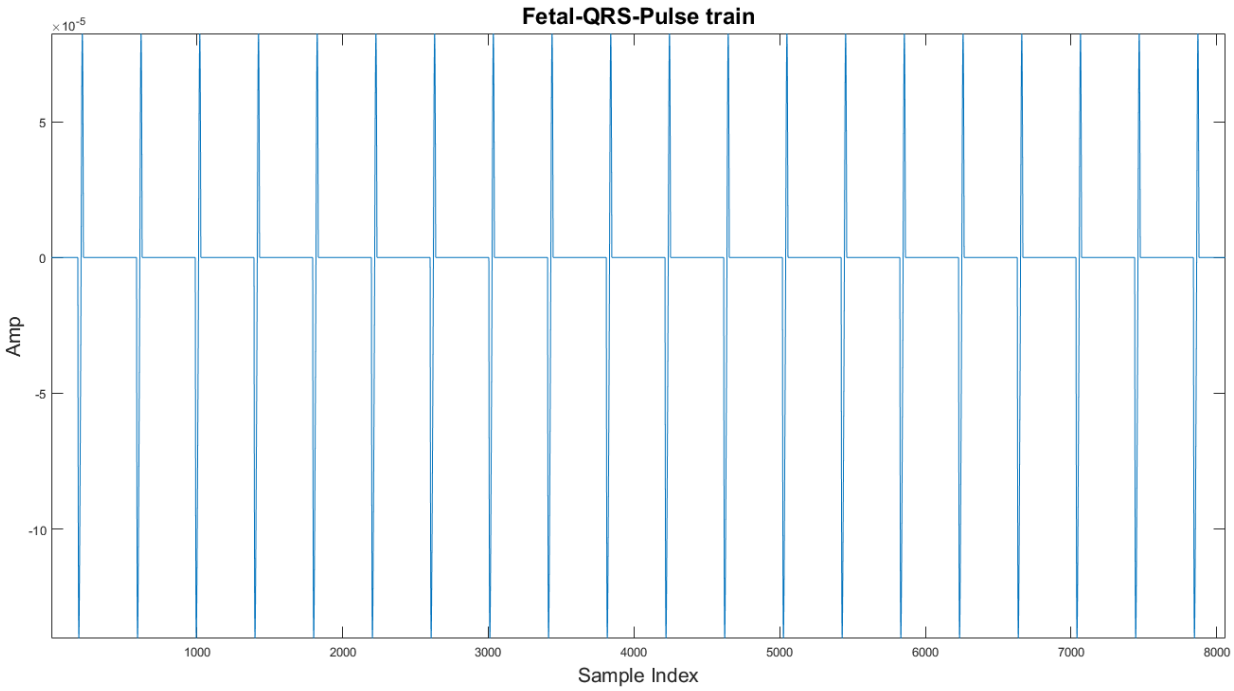


Fig26. Fetal QRS pulse train

3.1.3 QRS Identifier

The QRS identifier consists of 2 major sections:

1. Template matching between the noisy FECG and the template FECG pulse train.
    - Objective: track the best candidates for fetal QRS pulses in the noisy signal.
    - Realization:
        a. The index, nearest to zero, of the maxima in the cross-correlation, was asserted to be the required ***shift size***.

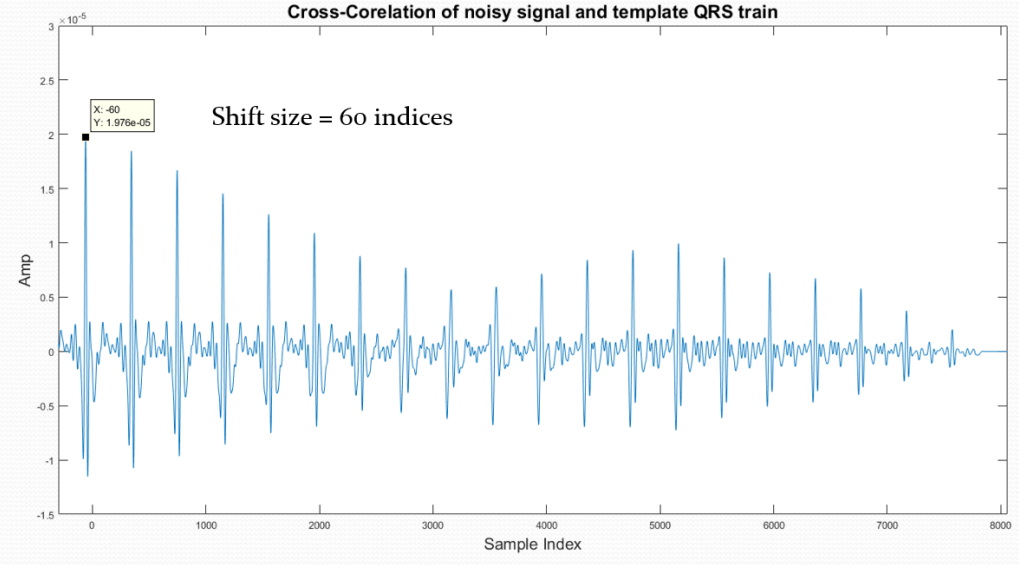


Fig27. Cross-correlation of noisy FECG and FQRS pulse train

b. The template QRS pulses train was shifted by the obtained ***shift size***.

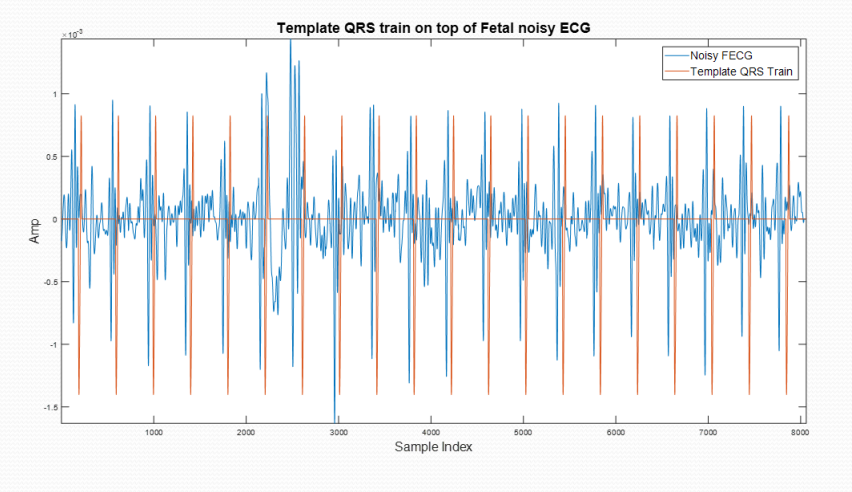


Fig28. Template QRS train over noisy FECG

Shift

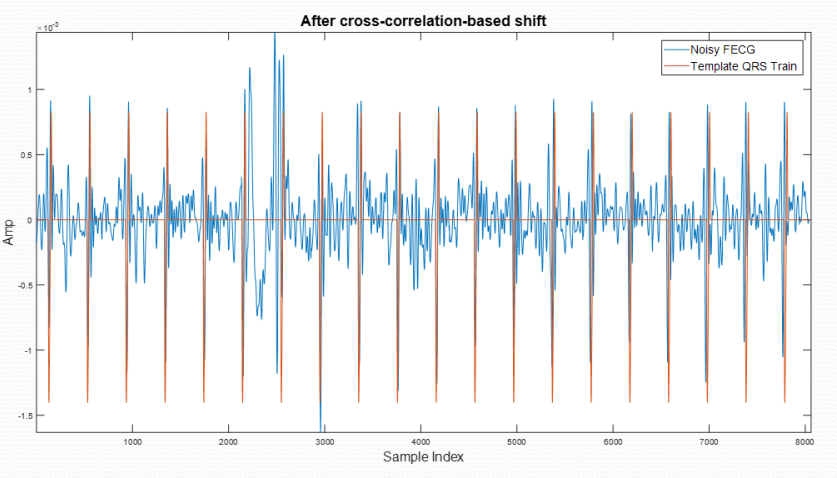


Fig29. Shifted template QRS train over noisy FECG

c. The maxima indices of the shifted template train were used as a reference for classifying fetal QRS pulses candidates.

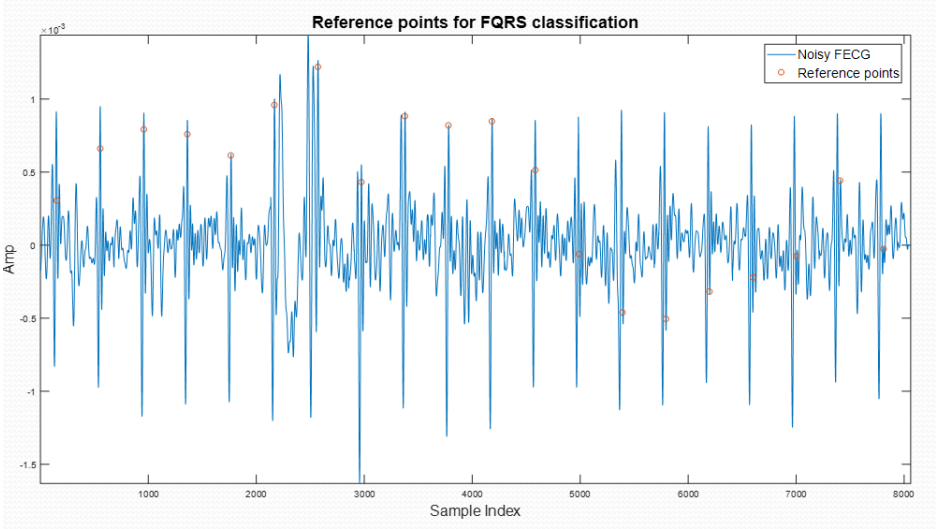


Fig30. Reference points for FQRS classification

2. Classification of possible QRS candidates.

    - Objective: identify the true fetal QRS pulses.
    - Realization:

        a. Find possible sets of fetal QRS pulses candidates using KNN classifier.

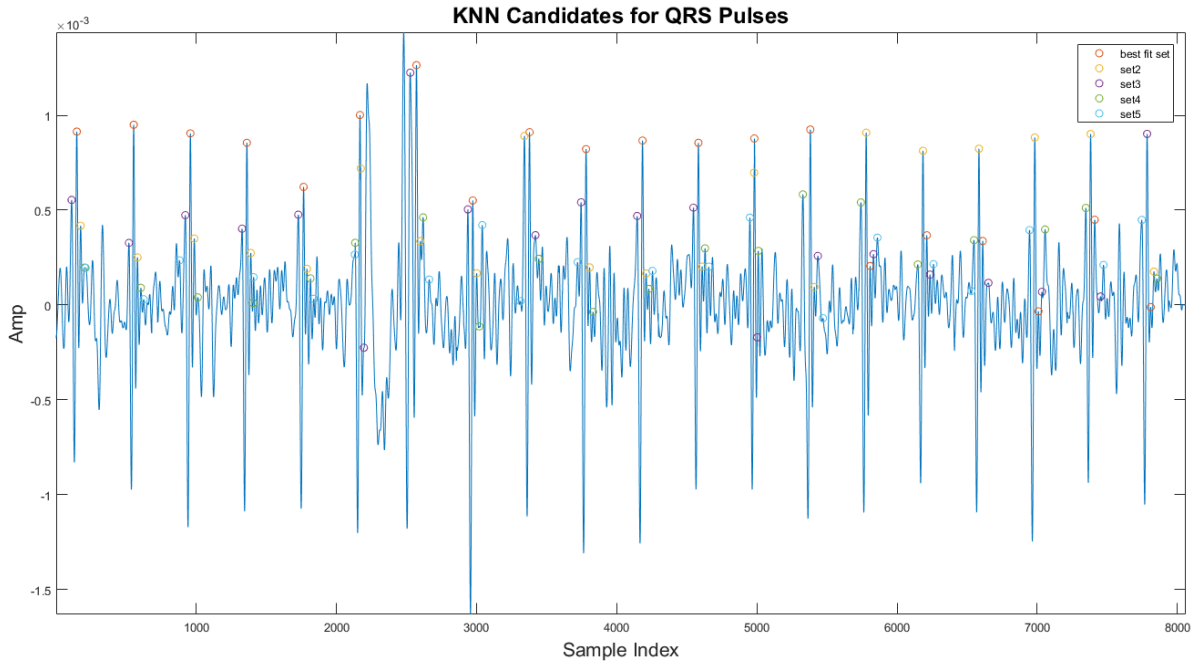


Fig31. KNN candidates for QRS pulses

b. Use a subset of the ***best fit set*** from the KNN classifier as a "**ground-truth**" reference, based on the estimated fetal BPM.

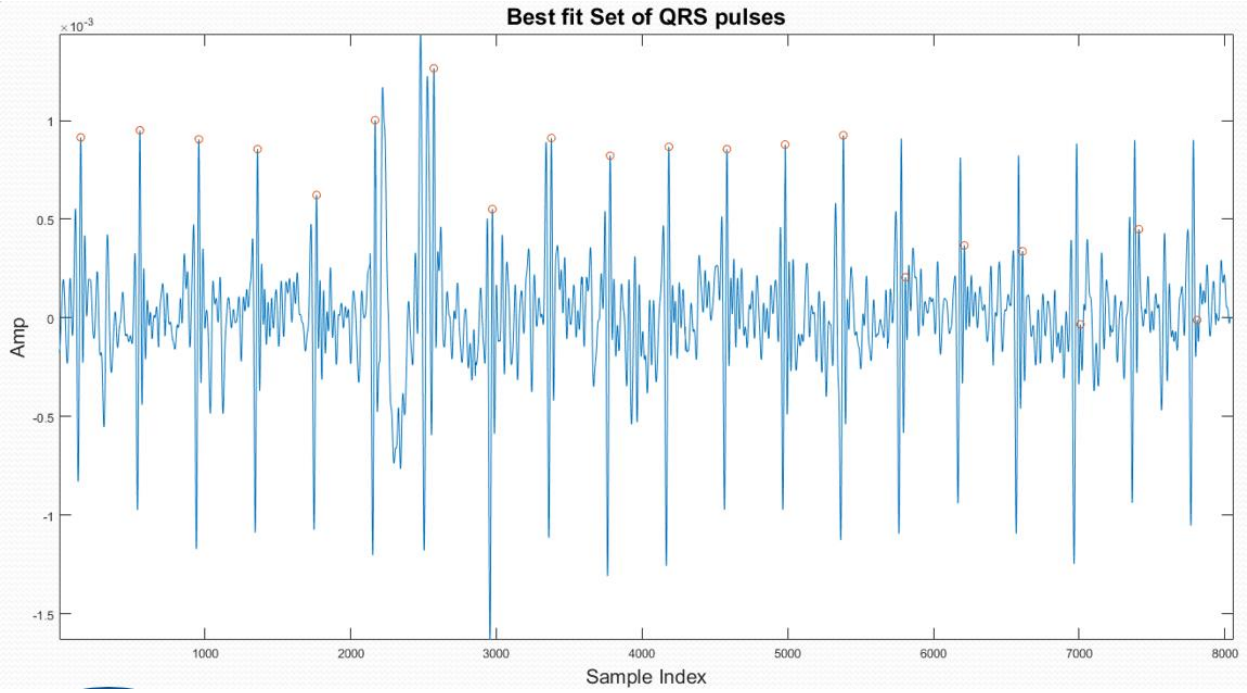


Fig32. Best fit set of QRS pulses

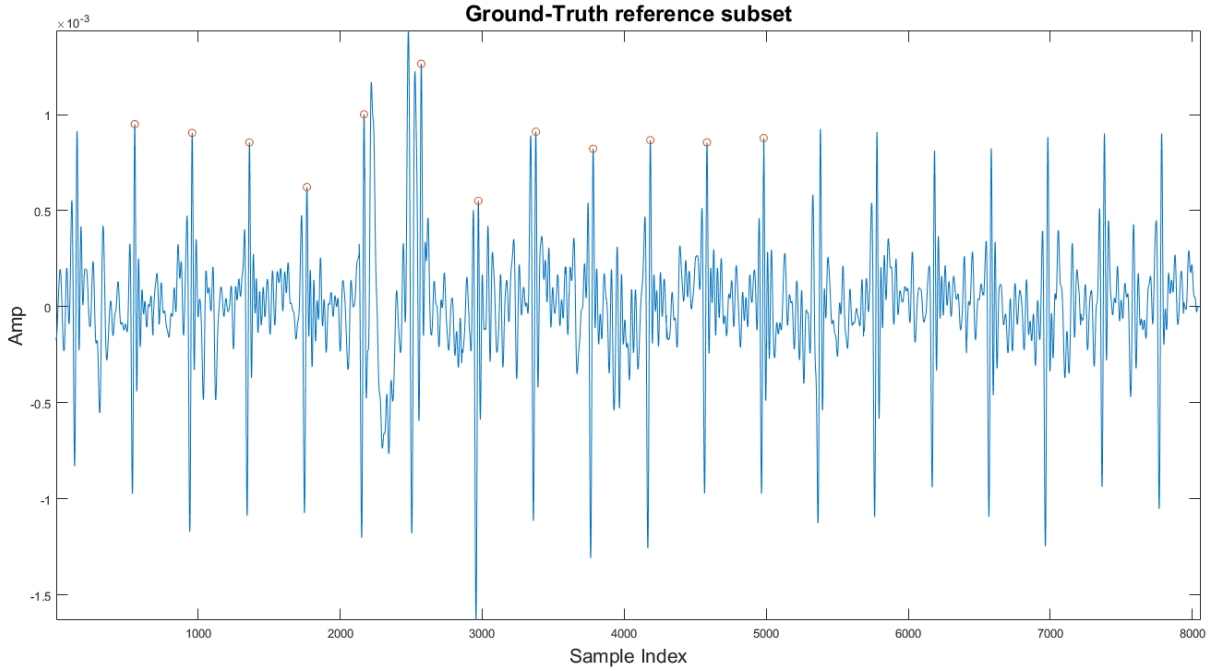


Fig33. Ground truth reference subset

c. Classify the remaining subset of the QRS pulses using the ground-truth reference.

d. Define the entire classified set as the new ***best fit set***.

e. Repeat steps #2-#4 until ***best fit set*** converges.

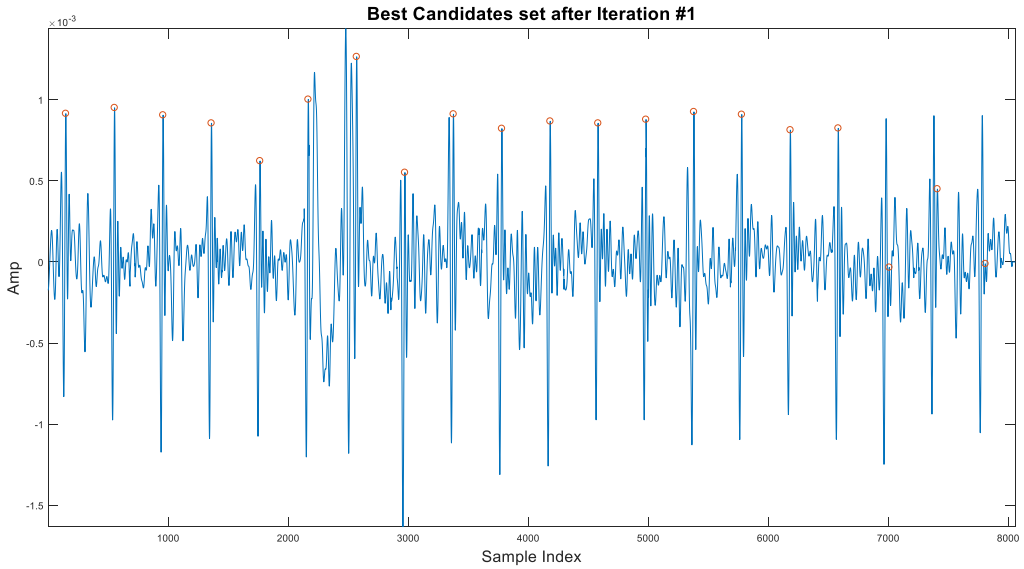


Fig34. Best candidates set after 1st iteration

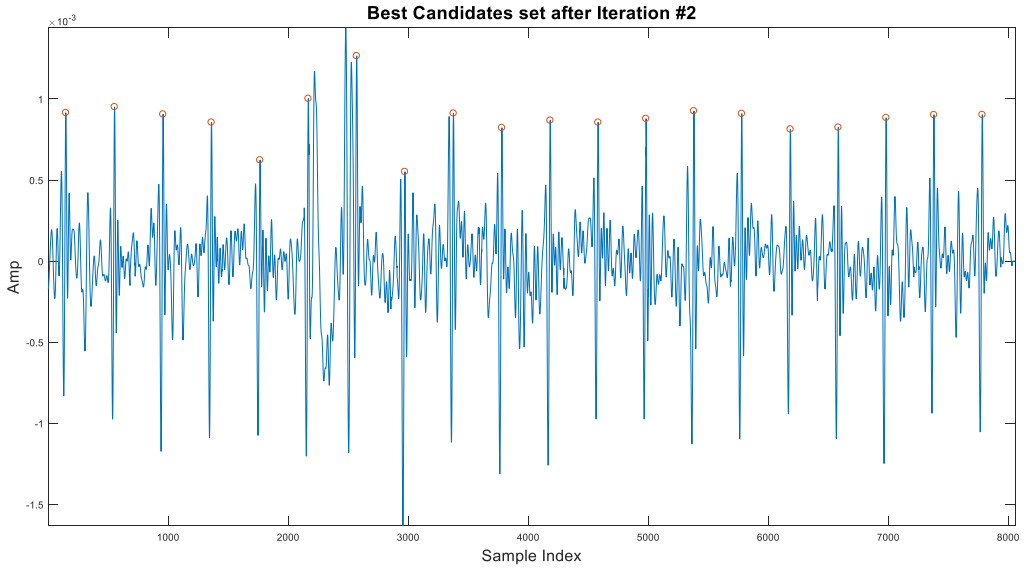


Fig35. Best candidates set after 2nd iteration

3.1.4 Time Domain Conversion

Taking into account that the previous processing steps were made over the derivative of the fetal noisy signal, a conversion back to the time domain is required. The fetal QRS pulses in the time domain were obtained by finding the zero-crossings indices that are adjacent to the classified pulses in the derivative domain.

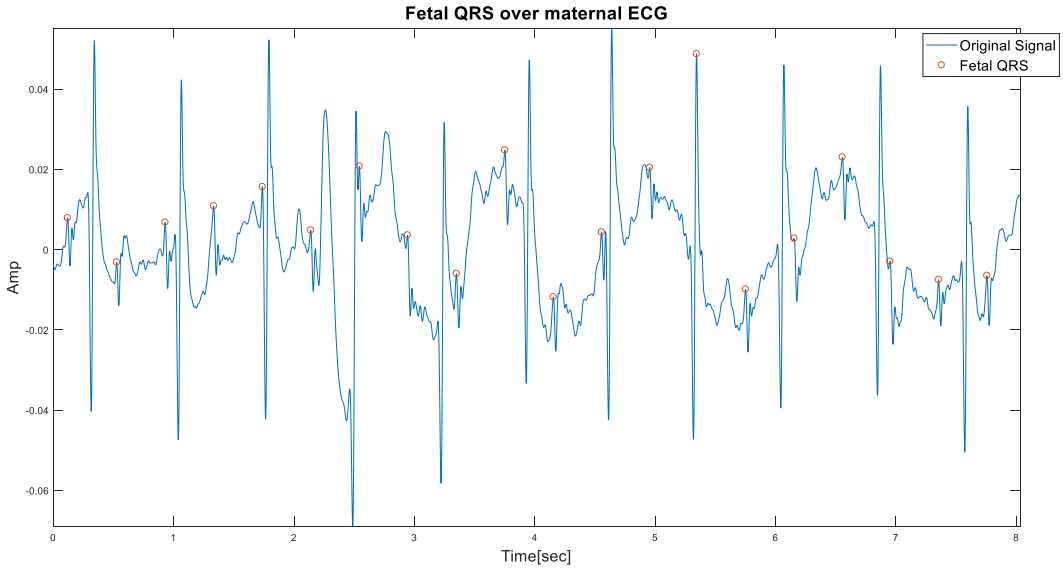


Fig36. Fetal QRS over maternal ECG

4. Results & Conclusions

4.1 Results analysis

Regression was made over a raw-ECG data base of pregnant women, provided by Dr. Shai Tejman from Shiba Hospital.

Our algorithm managed to identify the fetal QRS pulses for **7 Out of 15** different pregnant women.

Shai's feedback*:

- Most of the identified fetal QRS pulses are true!
- In some cases, the results are inconclusive and hard to assess.

4.2 Conclusions

Identifying the FECG QRS peaks from the abdominal ECG signal is possible!

Low SNR is a major cause for miss-detection of the FECG QRS pulses, specifically due to wrong BPM estimation.

For a reasonable SNR, auto-correlation-based classification is a good enhancement to the algorithm.

4.3 Suggestions

Looking forward, several steps can be taken into account, in order to achieve better results.

The suggested measures are as follows:

1. Results analysis by the medical team experts:

- Optimize the sampling method (AP/lat/upside-down).
- Assess the fetus age in which the best results are obtained.

2. Algorithm improvements:

- Optimize the time window for the BPM estimation, and use it as a sliding window.
- Improve the typical FQRS pulse, used for the template pulse train, by averaging over more obtained pulses.
- Efficiency improvement and complexity minimization for real-time processing purposes.

** Dr. Shai's original feedback in Hebrew is located at the appendix section.*

5. GUI

In order to allow the medical experts to explore more sampled ECG signals by themselves, a user-friendly interface was created.

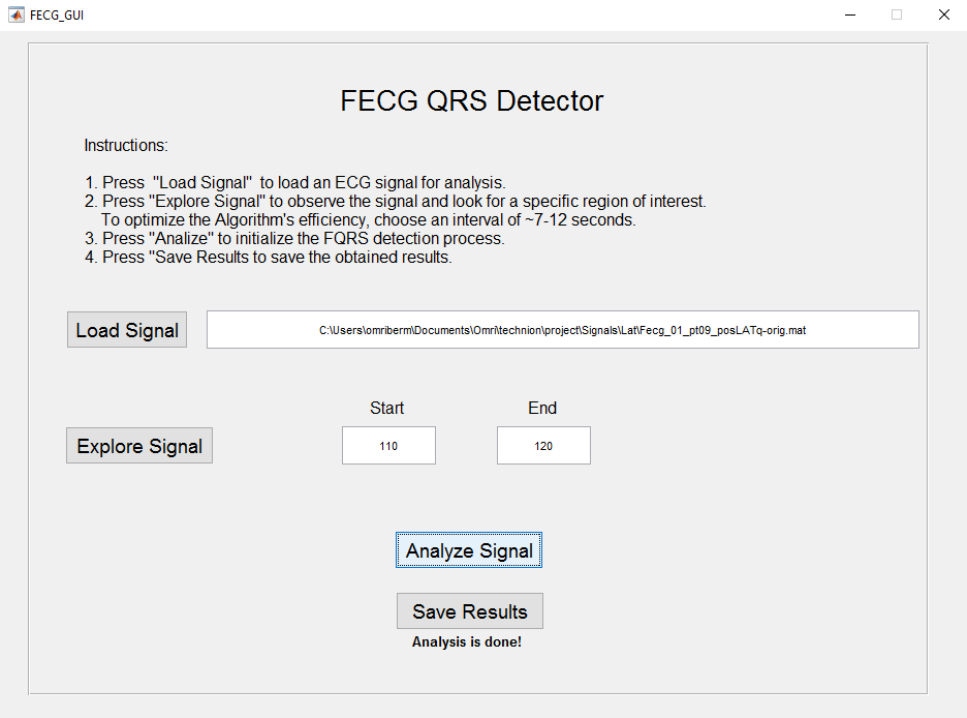


Fig37. GUI home page

An instruction guide regarding the usage of the GUI can be located in the appendix section.

6. References

1. Dr. Eldar Yonina, Dr. Kutyniok Gitta. Compressed Sensing Theory and applications. Chapter 11.
2. Kwang Jin Lee and Boreom Lee. "Sequential Total Variation Denoising for Extraction of Fetal ECG from Single-Channel Maternal Abdominal ECG"
3. Dr. Michael Elad, Sparse and Redundant Representations and Their Applications in Signal and Image Processing course, 236862, lectures 3 and 10
4. Deam A. The use of fetal electrocardiogram. J Obstet Gynecol. 1994;101:9–17.
5. Kurbel S. A vector-free ecg interpretation with p, qrs & t waves as unbalanced transitions between stable configurations of the heart electric field during p-r, s-t & t-p segments. Theor Biol Med Model. 2014;11:10.
6. Ferrara ER, Widrow B. Fetal electrocardiogram enhancement by time-sequenced adaptive filtering. IEEE Trans Biomed Eng. 1982;BME-29:458–60.
7. Kanjilal PP, Saha G. Single-channel maternal ecg using singular value decomposition. IEEE Trans Biomed Eng. 1997;44(2):51–59.
8. Assaleh K. Extraction of fetal electrocardiogram using adaptive neuro-fuzzy inference systems. IEEE Trans Biomed Eng. 2007;54(1):59–68.
9. Zarzoso V, Millet-Roig J, Nandi AK. Fetal ECG extraction from maternal skin electrodes using blind source separation and adaptive noise cancellation techniques. In: Computers in Cardiology 2000. IEEE: 2000. p. 431–434.
10. Callaerts D, Moor BD, Vandewalle J, Sansen W. Comparison of svd methods to extract the fetal electrocardiogram from coetaneous electrode signals. Med Biol Eng Comput. 1990;28:217–24.
11. Mochimaru F, Fujimoto Y, Ishikawa Y. Detecting the fetal electrocardiogram by wavelet theory-based methods. Prog Biomed Res. 2002;7:185–93.
12. Zarzoso V, Nandi AK, Bacharakis E. Maternal and fetal ecg separation using blind source separation methods. MA J Math App Med Biol. 1997;14(3):207–225.
13. Comon P. Independent component analysis, a new concept? Signal Process. 1994;36(3):287–314.
14. Ghazdali A, Hakim A, Laghrib A, Mamouni N, Raghay S. A new method for the extraction of fetal ECG from the dependent abdominal signals using blind source separation and adaptive noise cancellation techniques. Theor Biol Med Model. 2015 Nov 14;12:25.
15. Keziou A, Fenniri H, Ghazdali A, Moreau E. New blind source separation method of independent/dependent sources. Signal Process. 2014; 104:319–324.
16. Ping Gao, Ee-Chien Chang, Lonce Wyse. Blind separation of fetal ECG from single mixture using SVD and ICA. 2003.
17. B.Eng Angela agostinelli, Marla Grillo, M.D Alessandra Biagini, B.Eng Corrado Giuliani, M.D Luca Burattini, B.Eng Sandro Fioretti, B.Eng - Ph.D Francesco Di Nardo, M.D Stefano R.Giannubilo, M.D Andrea Ciavattini, B.Eng-Ph.D Laura Burattini. Noninvasive fetal Electrocardiography: An overview of the signal electrophysiological meaning, recording procedures and processing techniques. 2015.
18. E. Castillo , D.P. Morales, G. Botella , A. García, L. Parrilla , A.J. Palma. Efficient wavelet-based ECG processing for single-lead FHR extraction. 2013.
19. Wikipedia.

7. Appendixes

7.1 code

7.1.1 GUI

```
function varargout = FECG_GUI(varargin)
% FECG_GUI MATLAB code for FECG_GUI.fig
%      FECG_GUI, by itself, creates a new FECG_GUI or raises the existing
%      singleton*.
%
%      H = FECG_GUI returns the handle to a new FECG_GUI or the handle to
%      the existing singleton*.
%
%      FECG_GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in FECG_GUI.M with the given input arguments.
%
%      FECG_GUI('Property','Value',...) creates a new FECG_GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before FECG_GUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to FECG_GUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help FECG_GUI

% Last Modified by GUIDE v2.5 19-Dec-2017 14:13:42

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @FECG_GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @FECG_GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end


% End initialization code - DO NOT EDIT

% --- Executes just before FECG_GUI is made visible.
function FECG_GUI_OpeningFcn(hObject, eventdata, handles, varargin)
```

```
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to FECG_GUI (see VARARGIN)

gui_path_split = strsplit(mfilename('fullpath'),'\');
guiData.gui_path = fullfile(gui_path_split{1:end-1});
cd(guiData.gui_path);
addpath(genpath(guiData.gui_path));

params_path = [guiData.gui_path '\gui_params.mat'];
if exist(params_path)
    load(params_path);
     handles.t_SigPath.String = sig_path;
     if exist(sig_path)
         handles.t_SigPath.UserData = true;
     end
     handles.t_Start.String = t_start;
     handles.t_End.String = t_end;
end

handles.figure1.UserData = guiData;

% Choose default command line output for FECG_GUI
handles.output = hObject;



% Update handles structure
guidata(hObject, handles);

% UIWAIT makes FECG_GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = FECG_GUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in t_LoadSig.
function t_LoadSig_Callback(hObject, eventdata, handles)
% hObject    handle to t_LoadSig (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dialog_txt = 'Choose the .mat file with the signal you wish to analyze';
ext = '\*.mat';
```

```
cur_dir = handles.figure1.UserData.gui_path;
[FileName,PathName,~] = uigetfile([cur_dir,ext],dialog_txt);
if FileName % a valid .mat file was chosen
    handles.t_SigPath.String = [PathName,FileName];
    handles.t_SigPath.UserData = true;
    handles.t_message.String = [];
    handles.t_message.ForegroundColor = [0 0 0]; % Black

else
    handles.t_SigPath.String = [];
    handles.t_SigPath.UserData = false;
    handles.t_message.String = 'Error: File wasn''t chosen properly. Please
retry!';
    handles.t_message.ForegroundColor = [1 0 0]; % Red
end



function t_SigPath_Callback(hObject, eventdata, handles)
% hObject    handle to t_SigPath (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% Hints: get(hObject,'String') returns contents of t_SigPath as text
%        str2double(get(hObject,'String')) returns contents of t_SigPath as a
double


% --- Executes during object creation, after setting all properties.
function t_SigPath_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t_SigPath (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in t_ExplrSig.
function t_ExplrSig_Callback(hObject, eventdata, handles)
% hObject    handle to t_ExplrSig (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if handles.t_SigPath.UserData
    handles.t_message.String = [];
    handles.t_message.ForegroundColor = [0 0 0]; % Black
    preFiltSig(handles.t_SigPath.String,true);
else
```

```
    handles.t_message.String = 'Error: No signal was chosen by the user.
Please choose a signal to process';
    handles.t_message.ForegroundColor = [1 0 0]; % Red
end



function t_Start_Callback(hObject, eventdata, handles)
% hObject    handle to t_Start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of t_Start as text
%        str2double(get(hObject,'String')) returns contents of t_Start as a
double

function t_End_Callback(hObject, eventdata, handles)
% hObject    handle to t_Start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of t_Start as text
%        str2double(get(hObject,'String')) returns contents of t_Start as a
double


% --- Executes during object creation, after setting all properties.
function t_Start_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t_Start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in t_Analyze.
function t_Analyze_Callback(hObject, eventdata, handles)
% hObject    handle to t_Analyze (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if ~handles.t_SigPath.UserData
    handles.t_message.String = 'Error: No signal was chosen by the user.
Please choose a signal to process';
    handles.t_message.ForegroundColor = [1 0 0]; % Red
elseif isempty(handles.t_Start.String) || isempty(handles.s_end.String)
    handles.t_message.String = 'Error: Start/End time aren''t indicated.
Please indicate a valid start/end time';
    handles.t_message.ForegroundColor = [1 0 0]; % Red
else
    sig_path = handles.t_SigPath.String;
```

```
    [t, filt_sig ] = preFiltSig( sig_path ,false);
    t_start = str2double(handles.t_Start.String);
    t_end = str2double(handles.t_End.String);
    if  t_start < t(1) || t_end > t(end)
        handles.t_message.String = ['Error: Start time should be greater than
' num2str(t(1)) ' /End time should be smaller than ' num2str(t(end))];
        handles.t_message.ForegroundColor = [1 0 0]; % Red
    else
        handles.t_message.String = 'Signal Analysis is in process...';
        handles.t_message.ForegroundColor = [0 0 0]; % Balck
        analyzed_fig = analyzeSig( filt_sig,t_start,t_end );
        handles.t_message.String = 'Analysis is done!';

        % save figure temporarily :
        cur_dir = handles.figure1.UserData.gui_path;
        saveas(analyzed_fig,[cur_dir '\tmpFig'],'jpg');
    end
end


% --- Executes during object creation, after setting all properties.
function t_End_CreateFcn(hObject, eventdata, handles)
% hObject    handle to s_end (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function s_end_CreateFcn(hObject, eventdata, handles)
% hObject    handle to s_end (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in t_Save.
function t_Save_Callback(hObject, eventdata, handles)
% hObject    handle to t_Save (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cur_dir = handles.figure1.UserData.gui_path;
if ~exist([cur_dir '\tmpFig.jpg'])
```

```
    handles.t_message.String = 'Error: Analyze wasn''t performed -> no data
is available to be saved' ;
    handles.t_message.ForegroundColor = [1 0 0]; % Red
else
    dialog_txt = 'Choose a folder to save the results';
    PathName = uigetdir(cur_dir,dialog_txt);
    sig_name = strsplit(handles.t_SigPath.String,{'.','\'});
    t_start = str2double(handles.t_Start.String);
    t_end = str2double(handles.t_End.String);
    copyfile([cur_dir '\tmpFig.jpg'],[PathName ,'\' sig_name{end-1} '_s'
num2str(t_start) '_e' num2str(t_end) '.jpg']);
    handles.t_message.String = 'Results have been saved' ;
    handles.t_message.ForegroundColor = [0 0 0]; % Black
end


% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cur_dir = handles.figure1.UserData.gui_path;
sig_path = handles.t_SigPath.String;
t_start = handles.t_Start.String;
t_end = handles.t_End.String;
if exist([cur_dir '\tmpFig.jpg'])
    delete([cur_dir '\tmpFig.jpg']);
end

save([cur_dir '\gui_params.mat'],'sig_path','t_start','t_end');

% Hint: delete(hObject) closes the figure
delete(hObject);
```

6.1.2 Auxiliary Functions

```
function analyzed_figure = analyzeSig( filt_sig,t_start,t_end )

    SIG_IND = (round(t_start*1e3):round(t_end*1e3))';
    X1_a_cut = filt_sig(SIG_IND);


    %% Find QRS peaks

    [QRS_set_mat,residue, stich_idx] = create_qrs_mat(X1_a_cut);
    QRS_pulse_width = size(QRS_set_mat,2);
    QRS_set_mat_T = QRS_set_mat.';
    QRS_set_vec = QRS_set_mat_T(:);

    %% PCA

    [QRS_U, QRS_S , QRS_V] = svd(QRS_set_mat);

    % if CALIB_ON
```

```
    %    SING_VAL_NUM = input('enter the desired number of the biggest
singular values to keep nonzero:\n');
    % end
    SING_VAL_NUM = 1;
    QRS_S(SING_VAL_NUM+1:end,:) = 0;
    maternal_T = QRS_U*QRS_S*QRS_V.';

    maternal_T_plot = maternal_T.'; % TODO: change to a more informative name


    %% Template Subtraction
    fetal_noisy = QRS_set_vec - maternal_T_plot(:);


    %% reconstruction of signal - to fit original size

    % We reconstruct the original abdominal ECG (after BPF) from which we cut
the QRS pulses.
    % We also pad the FECG that was processed from the QRS pulses with the
same
    % residue.

    rec_ABDOM_ECG = residue;
    rec_FECG = residue;
    for i=1:length(stich_idx)-1
        rec_FECG(stich_idx(i):stich_idx(i)+QRS_pulse_width-1) =
fetal_noisy((i-1)*QRS_pulse_width+1:i*QRS_pulse_width);
        rec_ABDOM_ECG(stich_idx(i):stich_idx(i)+QRS_pulse_width-1) =
QRS_set_vec((i-1)*QRS_pulse_width+1:i*QRS_pulse_width);
    end


    SIG_IND_PLOT = (stich_idx(1):stich_idx(end))';


    %% Derivation

    stich_idx_diff = sort([stich_idx-1 stich_idx+QRS_pulse_width-1]);
    FECG_diff = diff([rec_FECG' rec_FECG(length(rec_FECG))]');

    FECG_diff(stich_idx_diff) = (FECG_diff(stich_idx_diff-
1)+FECG_diff(stich_idx_diff-2))/2;

    fecg_diff = FECG_diff(SIG_IND_PLOT);

    %% BPM Estimation:
    bpm = getBPM(FECG_diff,1e3,180,120);

    %% QRS Classifier
    [~,fecg_noisy_peaks_idx] = findpeaks(X1_a_cut(SIG_IND_PLOT));
    Fetal_QRS_idx  = FECGClassifier( bpm,fecg_diff,1e3,fecg_noisy_peaks_idx);
```

```
    %% Plot final result:
    orig_maternal = X1_a_cut(SIG_IND_PLOT);
    analyzed_figure = figure;
    plot((SIG_IND_PLOT-SIG_IND_PLOT(1))/1e3,orig_maternal);
    hold on;
    scatter(Fetal_QRS_idx/1e3,orig_maternal(Fetal_QRS_idx));
    xlabel('Time[sec]','FontSize',16);
    ylabel('Amp','FontSize',16);
    title('Fetal QRS over maternal ECG','FontSize',18);
    axis tight;
    legend({'Original Signal','Fetal QRS'},'FontSize',14);

end



function [pulse_mat, residue, stich_idx] = create_qrs_mat
(signal)%,threshold)
    % this creates a matrix of MECG QRS pulses by the following steps:
    % 1. applies Thresholding on the signal to avoid misdetections
    % 2. locates the peaks of the signal, which correspond to QRS MECG peaks
    % 3. cuts the signal to similarly sized pulses. all the pulses are
centered at the indices
    %    of the QRS peaks, and cut simetrically, based on the minimum pulse's
width
    % 4. the QRS pulses are set in a matrix as rows, so that all peaks are
centered at
    %    the same column.
    % 5. the residue of the signal and the indices from which the pulses were
    %    cut are also returned for reconstruction purposes later in the code.

    x1 = signal;

    % Find the Maternal BPM based on the diferential of MECG:
    bpm = getBPM(diff(x1),1e3,120,60);

    % Trace the maternal QRS peaks:
    [~,x1_pks_idx] = findpeaks(x1);
    Metal_QRS_idx  = MECGClassifier( bpm,diff(x1),1e3,x1_pks_idx);


    % x1(x1<threshold) = 0; % thresholding

    x1_pks_idx = Metal_QRS_idx;

    length_x1_pks_idx = length(x1_pks_idx);
    delta_vec = abs(x1_pks_idx(1:length_x1_pks_idx-1) -
x1_pks_idx(2:length_x1_pks_idx)); % a vector of the distances between
neighbour QRS peaks
    delta_min = min(delta_vec);

    half_peak = floor(delta_min/2)-1; % the interval used to simetrically cut
pulse arround the QRS peak
    pulse_mat = zeros(length_x1_pks_idx-2,(2*half_peak)+1);
```

```
    residue = signal; % will contain the residual parts of the signal after
cutting the QRS peaks out
    stich_idx = zeros(1,length_x1_pks_idx-2);
    for i = 2:length_x1_pks_idx-1

        QRS_iter_peak = x1_pks_idx(i)- half_peak : x1_pks_idx(i)+ half_peak;
        pulse_mat(i-1,:) = signal(QRS_iter_peak);
        residue(QRS_iter_peak) = 0;
        stich_idx(i-1) = x1_pks_idx(i)- half_peak;
    end
    residue(1:stich_idx(1)-1) = 0;
    residue(stich_idx(end)+2*half_peak+1:end) = 0;

end


function  Fetal_QRS_idx_true  = FECGClassifier(
bpm,fecg_noisy,fs,fecg_noisy_peaks_idx)
    %FECGClassifier Summary of this function goes here
    %   Detailed explanation goes here

    %% Build Synthectic fetal mask
    load single_FECG_pulse.mat;

    fetal_freq = bpm/60;
    FQRS_delta = fs/fetal_freq;
    zero_pad_len = round((FQRS_delta-length(single_pulse))/2);
    zero_pad = zeros(zero_pad_len,1);
    single_pulse_padded = [zero_pad;single_pulse;zero_pad];
    num_of_pulses = ceil(length(fecg_noisy)*fetal_freq/fs);
    mask = repmat(single_pulse_padded,num_of_pulses,1);

    delta = length(mask)- length(fecg_noisy);
    if delta > 0
        fecg_noisy = [fecg_noisy;zeros(delta,1)];
    elseif delta < 0
        mask = [mask;zeros(abs(delta),1)];
    end

    [corr_res,corr_lags] = xcorr(fecg_noisy,mask);
    [~,max_corr_idx] = max(corr_res);
    if corr_lags(max_corr_idx)>0
        off_ratio = ceil(corr_lags(max_corr_idx)/FQRS_delta);
    elseif corr_lags(max_corr_idx)< -FQRS_delta
        off_ratio = -floor(-corr_lags(max_corr_idx)/FQRS_delta);
    else
        off_ratio = 0;
    end
    mask_offset = corr_lags(max_corr_idx)-FQRS_delta*off_ratio;

    [mask_peaks,mask_peaks_idx] = findpeaks(mask);

    % a single pulse might contain more than one maxima
    if length(mask_peaks)> num_of_pulses
        [~,mask_peaks_sorted_idx] = sort(mask_peaks,'descend');
```

```
        mask_peaks_sorted_idx(num_of_pulses+1:end) = [];
        mask_peaks_idx = mask_peaks_idx(mask_peaks_sorted_idx);
    end

    Fetal_QRS_idx = mask_peaks_idx+mask_offset;
    Fetal_QRS_idx(Fetal_QRS_idx>length(fecg_noisy)) = [];
    Fetal_QRS_idx(Fetal_QRS_idx<1) =
length(fecg_noisy)+Fetal_QRS_idx(Fetal_QRS_idx<1);
    Fetal_QRS_idx_sorted = sort(Fetal_QRS_idx);

    %% classify

    [~,noisy_peaks_idx] = findpeaks(fecg_noisy);

    class_num = 5;
    [indices,~] =
knnsearch(noisy_peaks_idx,Fetal_QRS_idx_sorted,'K',class_num);

    fecg_best_candidates = noisy_peaks_idx(indices(:,1));
    true_fecg_peaks_idx = zeros(1,size(indices,1));
%     fecg_peaks_med=  median(fecg_noisy(noisy_peaks_idx(indices(:,1))));
    fecg_dist_med =  median(diff(fecg_best_candidates));

    did_converge = 0;
    while ~did_converge
        true_fecg_peaks_idx_prev = true_fecg_peaks_idx;

        [ QRS_true_candidates_idx,QRS_true_dist_avg ] = FiltFalsePositives(
fecg_best_candidates ,fecg_dist_med);

        for i=1:size(indices,1)

            if ismember(i,QRS_true_candidates_idx)
                if ~true_fecg_peaks_idx(i)
                    true_fecg_peaks_idx(i) = indices(i,1);
                end
            else
                cur_candidates_idx = noisy_peaks_idx(indices(i,:));
                cur_candidates_idx_mat =
repmat(cur_candidates_idx,1,length(QRS_true_candidates_idx));
                cur_candidates_diffs = cur_candidates_idx_mat-
fecg_best_candidates(QRS_true_candidates_idx)';
                cur_candidates_diffs_w =
cur_candidates_diffs./(abs(QRS_true_candidates_idx-i))';
                dev_from_avg_dist = abs(cur_candidates_diffs_w)-
QRS_true_dist_avg;
                ssd = sqrt(sum(dev_from_avg_dist.^2,2));
                [~,fecg_best_fit_peak_idx] = min(ssd);
                true_fecg_peaks_idx(i) = indices(i,fecg_best_fit_peak_idx);
            end
        end
        fecg_best_candidates = noisy_peaks_idx(true_fecg_peaks_idx);

        fecg_dist_med = QRS_true_dist_avg;
```

```
        did_converge = isequal(true_fecg_peaks_idx,true_fecg_peaks_idx_prev);

    end

    Fetal_QRS_diff_idx = noisy_peaks_idx(true_fecg_peaks_idx);

    %% find original Fetal QRSs:
%     [~,Fetal_QRS_idx_true] =
quantiz(Fetal_QRS_diff_idx,fecg_noisy_peaks_idx,[fecg_noisy_peaks_idx; Inf]);
    [~,Fetal_QRS_idx_true] =
quantiz(Fetal_QRS_diff_idx,fecg_noisy_peaks_idx,[1;fecg_noisy_peaks_idx]);
    Fetal_QRS_idx_true(Fetal_QRS_idx_true==Inf) = [];
end


function [ QRS_true_candidates_idx,QRS_true_dist_avg ] = FiltFalsePositives(
QRS_candidates_idx ,QRS_dist_med)
    %FiltFalsePositives gets the indices of the best candidates for the
    %QRS's indices and filters out the false positives using an assertion
    %over the distances between them. it then returns only the true
    %candidates indices, the corresponding original index of each
    %candidate, and the newly calculated average of the distances between
    %the true positives.

    max_acc_dist_allowed = 20; % the ammount of accumulating distance allowed
for a QRS peak to be from its neighbours

    diff_from_med = diff(QRS_candidates_idx)-QRS_dist_med;
    acc_diff = abs(diff_from_med(1:end-1)) + abs(diff_from_med(2:end)); % the
accumulating diff from the prior and next QRS

    QRS_best_candidates_idx = find(acc_diff<max_acc_dist_allowed)+1;
    QRS_adj_idx_dist = diff(QRS_best_candidates_idx);
    QRS_adj_idx_dist(QRS_adj_idx_dist~=1) = 0;
    QRS_straits = bwconncomp(QRS_adj_idx_dist);
    [~,QRS_longest_strait_idx] =
max(cellfun(@length,QRS_straits.PixelIdxList));
    QRS_longest_strait = QRS_straits.PixelIdxList{QRS_longest_strait_idx};
    QRS_true_candidates_idx =
[QRS_best_candidates_idx(QRS_longest_strait(1));QRS_best_candidates_idx(QRS_l
ongest_strait+1)];

    norm_dists =
diff(QRS_candidates_idx(QRS_true_candidates_idx))./diff(QRS_true_candidates_i
dx);
    QRS_true_dist_avg = mean(norm_dists);


end


function  heart_bpm  = getBPM( sig,fs,max_bpm,min_bpm )
    %getBPM Calculates the inputs signal's heart rate using
```

```
    %autocorrelation.

%     Inputs:
%       - sig: the input signal
%       - fs: the sampling rate
%       - max_bpm: the upper bound for the BPM
%       - min_bpm: the lower bound for the BPM
%     Outputs:
%       - heart_bpm: the identified BPM

    [frame_corr,lags] = xcorr(sig);
    frame_corr = frame_corr(lags>=0);
    [max_vals,max_idx] = findpeaks(frame_corr);
    % sort the autocorrelation's peaks from biggest to smallest:
    [~,Idx] = sort(max_vals,'descend');

    % go over the indices of the autocorrelation's peaks from the biggest to
%     the smallest and see if the inferred frequency applies as a possible
%     BPM regarding the input limitations:
    for i=1:length(max_idx)
        cur_freq = fs/max_idx(Idx(i));
        cur_BPM = cur_freq*60;
        if (cur_BPM < max_bpm) && (cur_BPM > min_bpm)
            heart_bpm = cur_BPM;
            break;
        end
    end

end


function  m_qrs_idx  = MECGClassifier( bpm,Mecg,fs,Mecg_peaks_idx)
    %MECGClassifier: Summary of this function goes here
    %   Detailed explanation goes here

    % Consts:
    std_ratio = 3;
    num_of_suspects = 2;

    % Calculate the number of expected QRS pulses based on BPM and signal
    % duration:
    maternal_freq = bpm/60;
    num_of_pulses = ceil(length(Mecg)*maternal_freq/fs);

    % find and Sort the MECG peaks, descending from high amplitude to low
    [mater_peaks,mater_peaks_idx] = findpeaks(Mecg);
    [~,mater_peaks_idx_sorted] = sort(mater_peaks,'descend');

    % Truncate the sorted peaks, keeping only the number of
    % expected+suspected QRS pulses:
    mater_peaks_idx_sorted(num_of_pulses+num_of_suspects+1:end) = [];
    m_qrs_idx_diff = mater_peaks_idx(mater_peaks_idx_sorted);

    % Calculate the Standard deviation based on the top 2/3rds of the
    % pulses:
```

```
    m_qrs_std = std(Mecg(m_qrs_idx_diff(1:end-
floor(length(m_qrs_idx_diff)/3))));

    % Calculate the average of the expected QRS pulses:
    m_qrs_mean = mean(Mecg(m_qrs_idx_diff(1:end-num_of_suspects)));


   % Identify the wrong suspects, based on deviation from the average and
   % the standard deviation:
    bad_suspect_idx = [];
    for i=length(m_qrs_idx_diff):-1:1
        smallest_qrs_std = abs(m_qrs_mean-Mecg(m_qrs_idx_diff(i)));
        if smallest_qrs_std/m_qrs_std>std_ratio
            bad_suspect_idx = [bad_suspect_idx i];
        else
            break;
        end
    end

    % Remove the bad suspects from the candidates vector:
    m_qrs_idx_diff(bad_suspect_idx) = [];

    % find original MECG peaks corresponding to the candidates:
    [~,m_qrs_idx] = quantiz(m_qrs_idx_diff,Mecg_peaks_idx,[Mecg_peaks_idx;
Inf]);
    m_qrs_idx(m_qrs_idx==Inf) = [];
    m_qrs_idx = sort(m_qrs_idx);
end
% figure,plot(Mecg)
% hold on;
% scatter(m_qrs_idx_diff,Mecg(m_qrs_idx_diff))

function [t, filt_sig ] = preFiltSig( sig_path ,do_plot)
    %UNTITLED Summary of this function goes here
    %   Detailed explanation goes here
    load(sig_path);
    X1_a= data;
    load BPF_butter40.mat;
    X1_a_filtered = filter(BPF,X1_a);
    SIG_DIR_SWITCH =  (abs(min(X1_a_filtered))>abs(max(X1_a_filtered)));
    filt_sig= -(2*SIG_DIR_SWITCH-1)*X1_a_filtered ; % turns the signal upside
down if it's the opposite direction


    t = (1:length(filt_sig))/1e3;
    if do_plot
        figure,
        plot(t,filt_sig);
        title('Explore the signal and look for a time interval to
analize','FontSize',18);
        xlabel('Time[sec]','FontSize',16);
        ylabel('Amplitude','FontSize',16);
        axis tight;
    end
end
```

7.2 Dr. Shai Tejman's comment on the results

אהלן חברים

להלן התגובה לגבי כל אחד מהניתוחים:
AP

1. pt11_120k_130k_145 - התקשתי מאוד בעין לזהות את האקג העוברי ולכן קשה לי מאוד לדעת האם זה באמת תואם - יהיה צורך להדגים לי איך זיהיתם אותו כאן.
2. pt12_40k_50k_155 - בעין אחרי שסימנתם - רואים את התבנית - שוב מעניין איך הגעתם לזה כאן - יפה מאוד !!!

LAT

1. p06poslat2_60k_70k_135 - ניתוח לא טוב ולא מדוייק - אני לא מסכים עם הזיהוי
2. p06poslat3_50k_60k_145 - גם כאן - ניתן לראות באופן ברור את האקג העוברי - ספייקים חדים קטנים בין האקג - האלגוריתם מזהה דברים אחרים ולא קשורים בכלל. - תסתכלו על תמונה של p09poslata_70k_80k_144 - עם ספייקים דומים וזיהוי בול.
3. p07poslat_70k_80k_139 - אני לא רואה כאן את האקג העוברי - וגם הקצב של האלגוריתם הוא לא הגיוני בכלל
4. p09poslata_70k_80k_144 - זיהוי מעולה
5. p09poslatq_70k_80k_154 - זיהוי מעולה
6. p14poslat_40k_50k_142 - זיהוי מעולה
7. p14poslat_60k_70k_138 - קשה לזהות אקג עוברי ואני לא בטוח בכלל לגבי הזיהוי

UP/DOWN

1. pt5_1_60_70_136 - זיהוי מעולה

אני אסתכל מחר על גיל ההריון של החולות שיש בהן זיהוי טוב - לדעתי צריך לדגום עוד - ואני כמעט בטוח שהם צעירים מאוד או לקראת לידה.

לסיכום:
1. בבקשה תריצו שוב את p06poslat3_50k_60k_145 - אני בטוח שתוכלו לקבל שם תוצאה יותר טובה ממה שקבלתם בהסתמך על התוצאות האחרות
2. כשמתאמים עם יונינה בבקשה גם להודיע לי כדי שאבוא לשמוע (ולתת הערות מצחיקות)
3. כשתסיימו את האלגוריתם אני זקוק לעזרתכם להתקינו אצלנו בתל השומר.

תודה! שי

7.3 GUI manual

Instructions:

1. Press "Load Signal" to load an ECG signal for analysis.
2. Press "Explore Signal" to observe the signal and look for a specific region of interest.

   To optimize the Algorithm's efficiency, choose an interval of ~7-12 seconds.
3. Press "Analyze" to initialize the FQRS detection process.
4. Press "Save Results to save the obtained results.