Phonocardiography and analysis using standard deviation profile

A thesis report submitted to the Academic Faculty

By

Tahsin Mahmud Chowdhury (092426)

Munif Nazmus Sakib (092403)

Hasan Md. Rafsanjani (092415)

In partial fulfillment of the requirements for the degree Bachelor of Science (B.Sc.) in Electrical and Electronic Engineering Islamic University of Technology, October, 2013

Under the Supervision of

Dr. Md. Ashraful Hoque

Professor

Department of Electrical & Electronic Engineering, IUT

Phonocardiography and analysis using - standard deviation profile

A Thesis Presented to
The Academic Faculty

By

Tahsin Mahmud Chowdhury (092426)

Munif Nazmus Sakib (092403)

Hasan Md. Rafsanjani (092415)

Approved by
Prof. Dr.Md.Ashraful Hoque

…….
Prof. Dr.Md.Ashraful Hoque
Thesis Supervisor
Professor
Dept. of Electrical & Electronic Engineering, IUT

……..
Prof. Dr. Md. Shahid Ullah
Head of the Department
Dept. of Electrical & Electronic Engineering, IUT

Member:

Tahsin Mahmud Chowdhury	Munif Nazmus Sakib	Hasan Md.Rafsanjani
……..	……	……

Islamic University of Technology (IUT)
The Organization of the Islamic Cooperation (OIC)
Gazipur-1704, Dhaka, Bangladesh,
October-2013

Table of Contents

ACKNOWLEDGEMENTS

- The undergraduate thesis, **"**Phonocardiography and analysis using standard deviation profile**"** has been written for the completion of Bachelor of Science degree at Islamic University of Technology, Bangladesh.

- This thesis work and writing has been done during the year 2013 under the supervision of ***Professor Dr.Md.Ashraful Hoque, Professor of EEE Department*** and under the co-supervision of ***Md. Abu Naser, Assistant Professor of EEE Department.***

- We would like to pay our special thanks and express our deepest gratitude to ***Prof. Dr. Md. Shahid Ullah,*** head of the department, ***Dr. Ashraful Hoque and Mr.Abu Naser*** for their constant guidance, help and encouragement to our work. Without their support it would have been impossible to complete such a task successfully.

- We would like to thank ***Dr. Siddique-e-Rabbani,*** Professor, Department of Biomedical physics and technology, University of Dhaka for collaborating with our thesis and helping us during the research.

- We are also grateful to all of our will-wishers, who provided their perpetual support towards accomplishing this task successfully.

Abstract

This thesis gives a brief idea on heart signal processing by using the method of phonocardiography. A digital stetescope was used to pick up the heart sounds from different patients and the corresponding analysis was performed. The thesis mainly attempts the use of standard deviation profile in heart rate calculation instead of the existing Shannon Energy profile at noisy environments. The thesis also emphasizes on the heart signal analysis using cheap and simple hardware and highly sophisticated use of software.

1. Introduction

In developing nations of the world there is often a shortage of medical specialists and limited access to diagnostic equipment which lead to a poor basic healthcare system. For example in Bangladesh, the number of physicians for every 1000 people is 0.3 while in the United States it is 2.4 (World Bank 2010). This problem is particularly eminent in rural areas and as a result people residing in such communities are forced to travel long distances from their home to seek medical attention in more urban areas. This travel can be financially burdensome, uncomfortable to the patient and includes risk of road accidents. In many cases, it might deplete a family's savings and leave them destitute.

Lack of medical attention for patients in developing nations has fatal consequences. The most alarming statistics are in the case of cardiovascular diseases (CVDs) .CVDs are the number one cause of death globally. According to the World Health Organization (WHO), an estimated 17.3 million people died from CVDs in 2008, representing 30% of all global deaths. Of these deaths, an estimated 7.3 million were due to coronary heart disease and 6.2 million were due to stroke. Low- and middle-income countries are disproportionally affected: over 80% of CVD deaths take place in low- and middle-income countries. The number of people who die from CVDs, mainly from heart disease and stroke, will increase to reach 23.3. million by 2030 and CVDs are projected to remain the single leading cause of death. This emphases the need for improved healthcare access for these people.

An abnormal heart rate can often be an indicator of a CVD. The aim of this thesis is to develop an algorithm that can be used to calculate the heart rate of a patient. The device that used for detection of the heart sound is a modified stethoscope. The earpiece of the stethoscope is replaced with a microphone input jack. The sound recorded by the stethoscope can then be fed into the signal processing software in a computer or a cell phone. For people in rural areas, the use of a cell phone and stethoscope to measure the heart rate would be very convenient. Before moving on to the signal processing part an overview of the heart and heart sounds is in order.

1.1 Physiology of the heart

The heart is made up of four chambers: the right and left ventricles, and the right and left atria.

The right and left sides of the heart are separated by the septum. Circulatory blood flow proceeds through the following sequence: body circulation, vena cava, right atrium, right ventricle, pulmonary artery, lungs, pulmonary vein, left atrium, left ventricle, aorta, body circulation (see Figure below) In order for this flow to occur, the heart muscle contracts and relaxes in a repeated cycle, with the blood being forced to follow the correct pattern with the help of heart valves.

Initially, the atria collect the blood from the circulation or lungs, depending on the side of the heart, in a passive way. This is the filling stage of the cardiac cycle, and is called the diastole.

During this period, the valves separating the atria and the ventricles are open; these are the right (mitral) and left (tricuspid) atrioventricular (AV) valves. The valves between the right ventricle and pulmonary artery (pulmonary valve) and the left ventricle and aorta (aortic valve) remain closed during the diastole.

At the very end of the diastole, the atria contract, forcing more blood into the ventricles. The depolarization of the cardiac cells which caused the contraction of the atria then reaches to the ventricles. When the ventricles contract, the AV valves close due to the pressure difference and the blood is ejected through the aorta or pulmonary artery. Upon completion of ventricular contraction, the ventricles begin to relax and the dropping pressure quickly goes below that of the aorta and pulmonary artery, at which time the aortic and pulmonary valves close.

Heart valves operate passively, acting as check valves to prevent back flow and take no active role in controlling the direction of flow. This design allows the timing of the cardiac cycle to be controlled by a single source which under normal conditions is the depolarization of the sinoatrial (SA) node. The SA node has its own timing sequence, and although the brain may modify its action, it is not under the direct control of the brain. This makes the heart a very independent organ.

The final discussion point regarding the physiology of the heart relates to the nature of pathological dysfunction. The heart valves may become calcified with deposits related to blood contents. Due to the passive nature of their operation, the resulting stiffening is clearly detrimental to the function of the valves and is therefore the factor of most

concern when diagnosing the heart. The cardiac valves may become weakened by other factors, causing them to be unable to function appropriately in the high-pressure environment of the heart.

The first heart sound, S_1 is associated with the closing of the mitral and tricuspid valves. The second heart sound, S_2 is associated with the closing of the aortic and pulmonary valves. This is further subdivided into the aortic valve (A_2) and pulmonary valve (P_2) closures. The sounds are actually slightly separated since the pulmonary valve closes slightly after the aortic valve; this may become more pronounced in pathological situations.

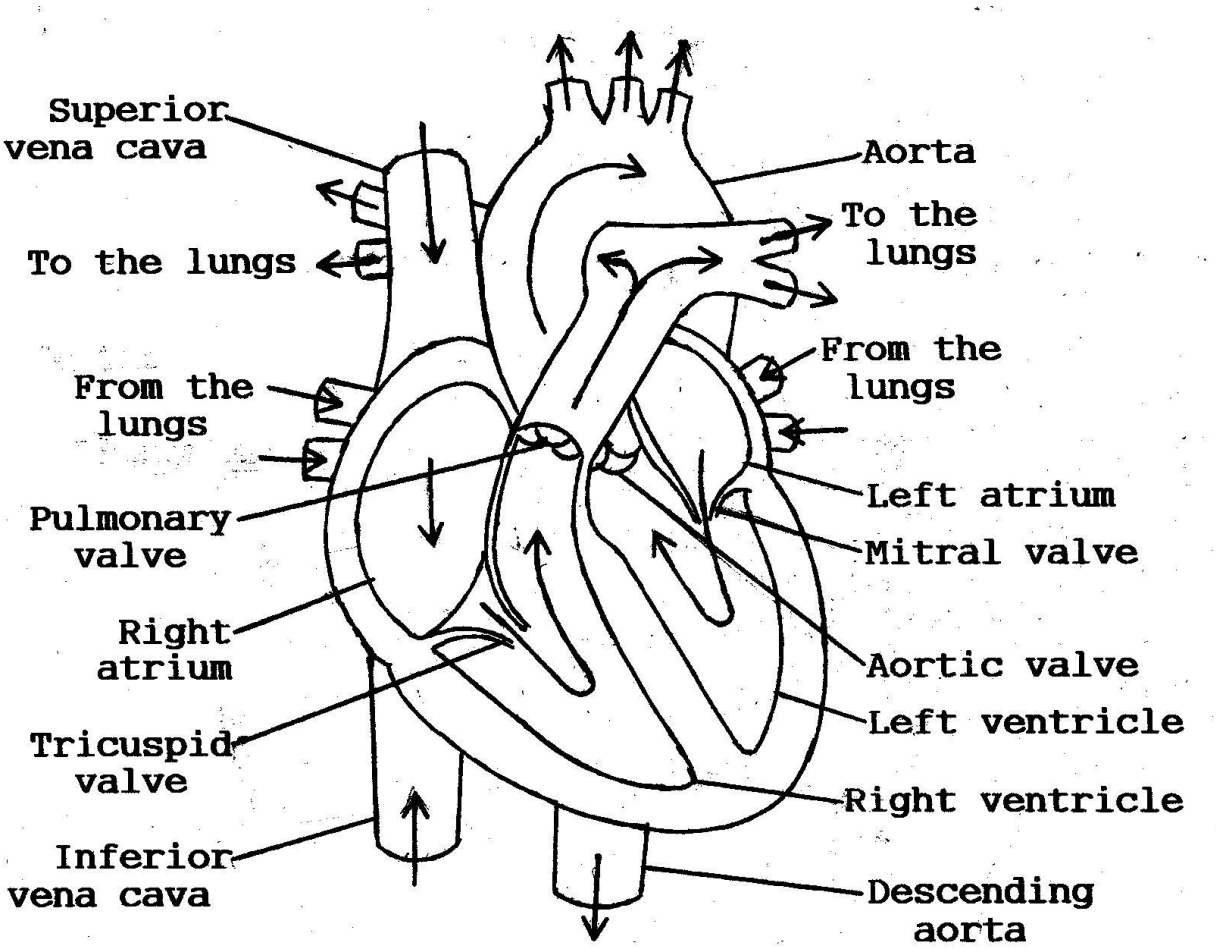


Fig: Structure of the Heart

1.2 Origins of Heart Sounds and Murmurs

There is wide agreement that normal heart sounds (SI and S2) as heard by cardiac auscultation are not caused directly by the closing of the valves" but by vibrations set up in cardiac blood vessels by the sudden pressure change caused by the closing of the heart valves. As such, the amplitude of the heart sounds are expected to correspond with the size of pressure drop which in turn corresponds to the amount of fluid flow before valves closure. Also, the frequency characteristics correlate with the size of the valves, with lower frequency characteristics corresponding to larger valves.

There has been much more debate regarding the origins of heart murmurs, likely as a result of the wide variety of physiological causes for the murmur themselves. However, studies have demonstrated that the origins of heart murmurs are the vibrations associated with turbulent flow. The sound energy density was also found to be proportional to the level of turbulence. Since laminar flow is the normal state of blood flow in human circulation, audible rnurmurs are usually associated with some pathological state. They are normally characterized by a higher frequency content than any other heart sounds4. Often heart murmurs do not indicate a pathological state. Any movement of fluid in a distensible chamber of vesse1 will cause vibrations of some sort, so everyone has some murmur if the recording instrument is sensitive enough.

1.3 Recoding the heart sound

Cardiac sounds occur within a low frequency range. These can be classified into the categories discussed above. It is important to keep in mind that the audible presence of a certain class of heart sound can be normal for some ages but abnormal for others.

For recording the signal, the stethoscope has to be placed on the chest and the output connected to a computer of cell phone. Amplitude is usually the loudest when the stethoscope is placed on the either the tricuspid or mitral valve area.

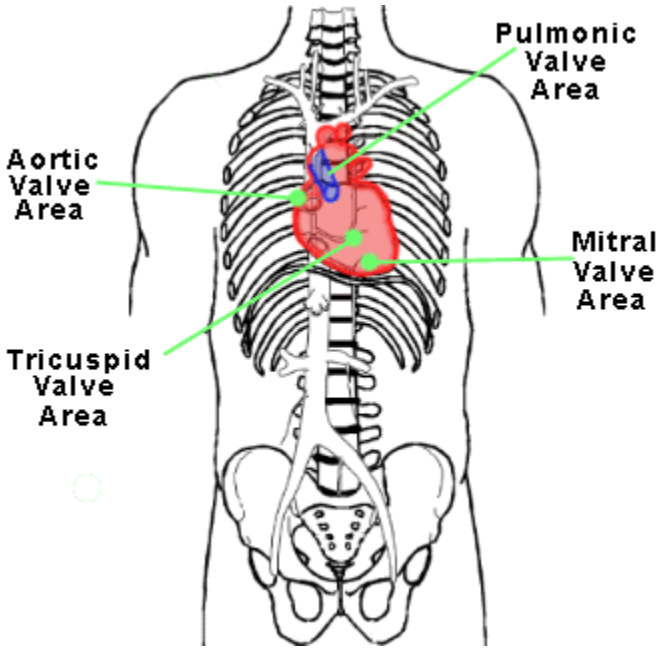


Fig: Various regions on the chest for placing the stethoscope

The heart rate can be defined as the number of heart beats per minute(bpm). The range of acceptable heart rates for an average person at rest is claimed to be: 60 to 100bpm for an adult, 50-100 bpm for a teenager, 74 to 140 bpm for a child and 70 to 170 bpm for an infant. Heart rates vary with age but other factors such as fitness level, stress, exercise or any heart disease can also affect the heart rate.

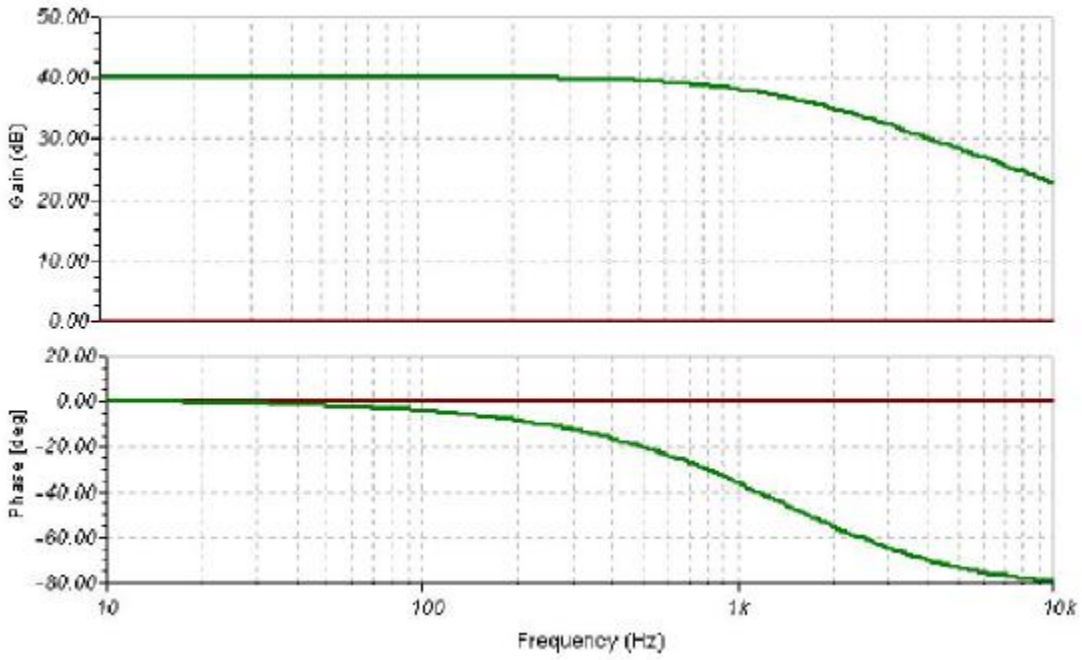
2. Literature Review

2.1 Heart sound signal retrieval and filtering

All heart sound signals were taken using an electronic stethoscope followed by the use of appropriate filters.

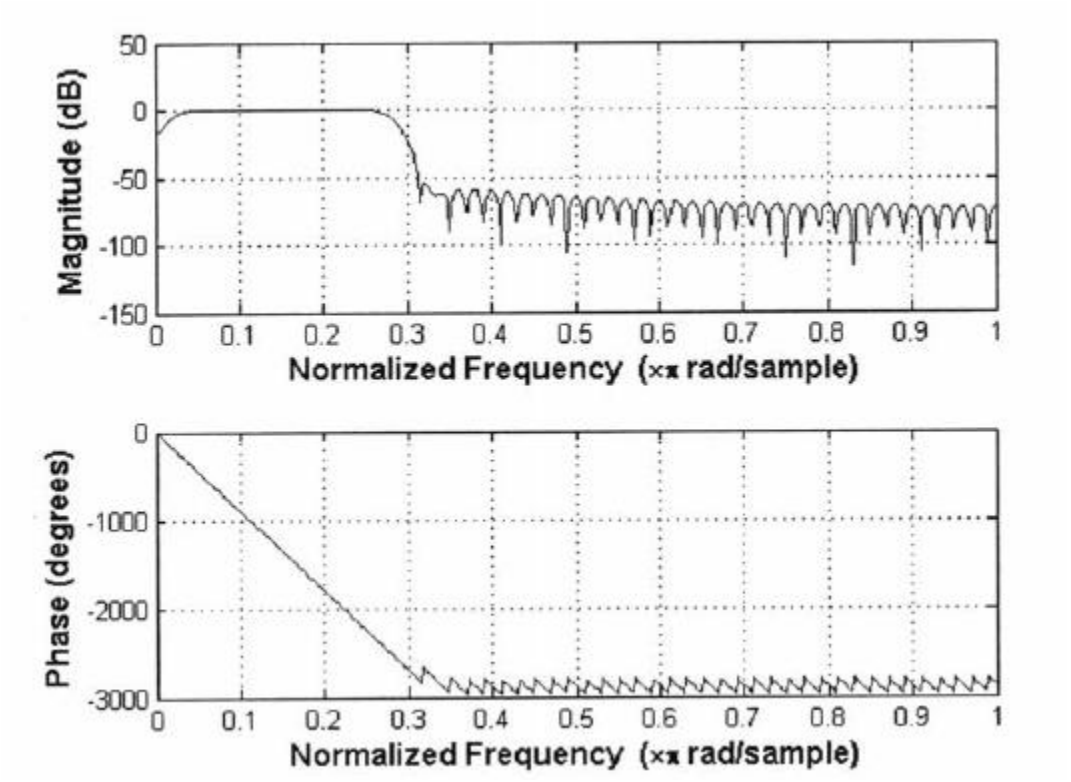
In the paper by Liang et al. (1999), an eighth-order Chebyshev type-I low-pass filter was used with a cutoff frequency at 882Hz. The signal passed through the filter was originally sampled at 11025 Hz with 16-bit accuracy, but was decimated by a factor of 5 by the filter to result in 2205Hz sampling frequency. The filtered signal was reversed and run back through the filter which results in zero phase distortion.

In the paper by Fei Yu et al. (2008), a hardware low-pass filter was used to cut off high frequency components during recording. The frequency and phase response is as below.



D. Kumar et al. (2006), in their paper developed a procedure to first segment the heart sound signal into lobes with defined boundaries. Unlike the previously referred papers, Kumar avoided empirically tuning any filters. He performed the fast wavelet transform (FWT) using the db6 (Daubechies wavelet family) performed upto the 6^{th} level to remove high frequency components. Daubechies wavelet family was used for its effectiveness in capturing transient sounds and the decomposition depth level of 6 was experimentally tuned.

In Kuan's thesis (2010), the algorithm used by Pan and Tompkins in their paper (1985) used to determine the heart rate from ECG signals was carried over to do the same using an acoustic heart sound signal. After recording, the input signal is downsampled from 44.1kHz to 500Hz and normalized. The signal was then passed through a 100-point FIR bandpass filter with cutoff frequencies of 5Hz and 70Hz.



In the paper by Mandal et al. (2010), the retrieved heart sound was firstly pre-amplified before passing it through an LPF-HPF combination to remove unnecessary regions of the signal followed by post-amplification. It then goes on to highlight two different classes of filtering and denoising techniques. The first of these methods employ an adaptive line enhancer (ALE) using the least mean square (LMS) algorithm and also the recursive mean square (RMS) algorithm. The ALE is able to suppress wideband noise of the sound signal by utilisation of an adaptive filter (used as a linear prediction filter) with the reference signal being a delayed version of the input heart sound signal. The purpose of the delay is to decorrelate noise so as the adaptive filter is only able to predict the heart sound signal and not the noise. The output signal is then subtracted from the input signal to obtain the error signal which is used to adaptively control the filter weights and minimise error.

The second method used wavelet-based denoising techniques. The orthogonal wavelet transform is able to compress the energy contained in a signal into a few large components with the disorderly noise characterized by small coefficients scattered throughout the transform. These smaller coefficients are omitted and the wave is reconstructed using the larger wavelet coefficients. Two orthogonal wavelet transform techniques are mentioned; one of them is SureShrink which is a smoothness adaptive algorithm and works at multiple levels of wavelet decomposition. It also estimates risks using Stain's unbiased risk estimator. The SureShrink algorithm's parameters were modified to expedite computation. The other technique, BayesShrink, though outclassed

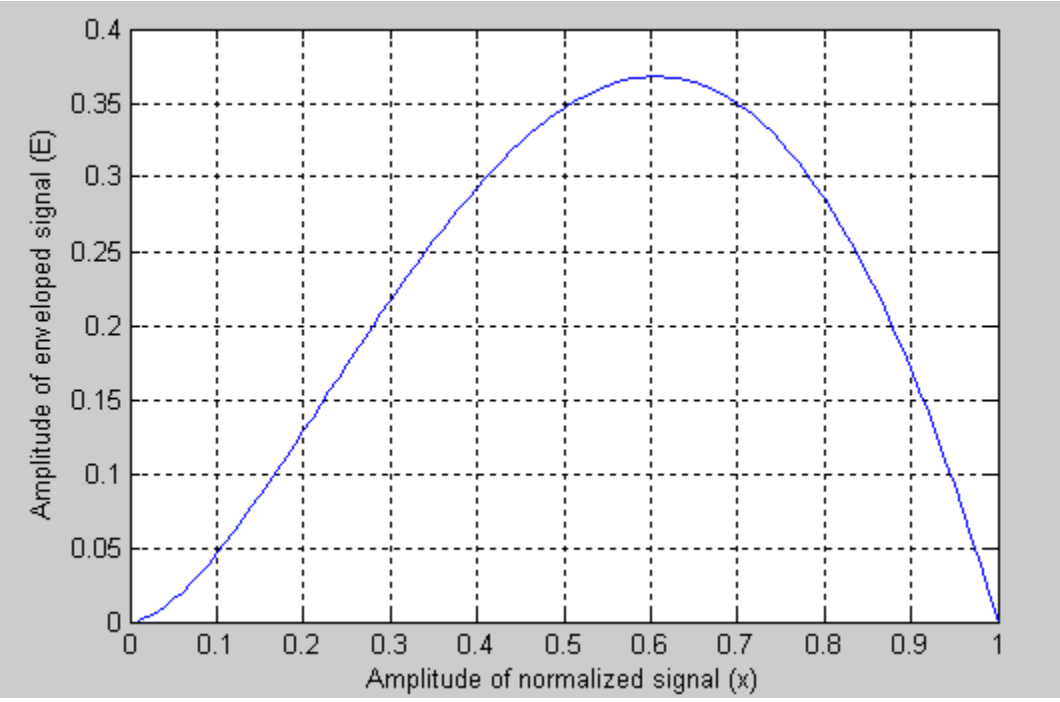
by newer techniques, was chosen for its efficiency in computation time which makes it viable for real-time operations.

2.2 Signal pre-processing

The signal is then processed further to help in the detection of the peaks.

Liang et al. [1] used a technique to help accentuate medium intensity signals over those of low and high intensities. This was achieved by normalizing the signal and determining the Shannon energy envelope of the signal.

$Shannon\ energy:\ \ E = -x^2 . \log x^2$

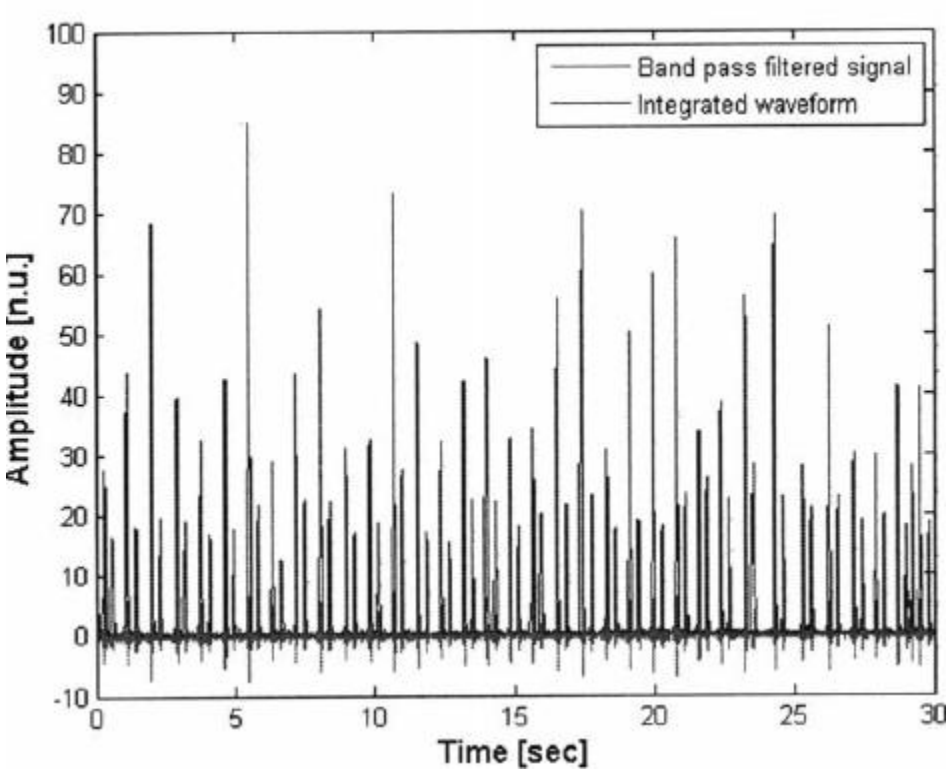


The average Shannon energy was calculated in continuous 0.02 second intervals with 0.01 second overlapping.

$E_s = -\frac{1}{N}.\sum_{i=1}^{N} x_{norm}^2(i).\log x_{norm}^2(i)$ where, x_{norm} is the normalized and decimated signal.

Fei Yu et al. [2] manually extracted the peaks before subsequent processing. Details are given in the next section.

Kuan [4] used the same technique employed by Pan and Tompkins. The energy of the heart sound signal was quantified by differentiating, squaring and integrating over an empirically determined window length of 58ms (or 29 samples long). The resulting integrated quantity peaked in high energy areas, i.e the S1/S2 sounds.



Kumar et al. [3], using the signal approximation coefficients of the FWT he performed previously and the Shannon energy operator (given below), he computed the signal envelope.

$$E(x[n]) = -\frac{1}{N}\sum_{n=1}^{N}(x[n])^2.\log\,(x[n])^2$$

where, $x \in$ {a1,d1,a2,d2,…,a6,d6} and aj, dj are j[th] level approximation and detail coefficients of the wavelet transformed heart sound signal, respectively, and N is the number of samples in the selected window.

A window of 20ms was selected with 10ms overlaps to compute the Shannon energy. Boundaries of the sound lobes are identified out by analyzing the zerocrossing of the normalized Shannon energy,

$$E_n(a5) = E(a5) - < E(a5) >$$

where, '< >' represents the average operator and E_n the normalized Shannon energy. The 5[th] level approximation coefficients were used.

The lobe boundaries are validated by checking the sound lobe duration dt_i, the interval between two consecutive sound lobes T_i^{int} and their loudness measured as the root mean square of the segment RMS_i.

n_i^{start} and n_i^{stop} is the start and stop sample index of the i^{th} segment, respectively.

$dt_i = T_s(n_i^{stop} - n_i^{start})$ where, T_s is the sampling period

$T_i^{int} = T_s(n_{i+1}^{start} - n_i^{stop})$

$RMS_i = \sqrt{\frac{\sum_{j=n_i^{start}}^{n_i^{stop}} S(j)^2}{n_i^{stop} - n_i^{start}}}$ where, i = 1,2,…segments,

and S is the heart sound signal

The conditions used for validation are stated below.

i. If dt_i < 30ms or dt_i > 250ms, the segment is considered to be noise-ridden and discarded. The durations were chosen on the basis of observed values in normal population

ii. If T_i^{int} is less than 50ms, the two segments are considered to belong to the same sound and the result of a split diastole. As the first segment (the aortic sound lobe) is more important than the second segment (the pulmonary sound lobe), the second is discarded.

iii. Splitting might not be captured by the Shannon energy due to low loudness of low frequency sounds, but pronounced local minima is observed nonetheless. If the local minima is less than 25% of the maximum value of the Shannon energy, the splitting is considered to occur at that position and the part of the segment with the highest loudness is kept.

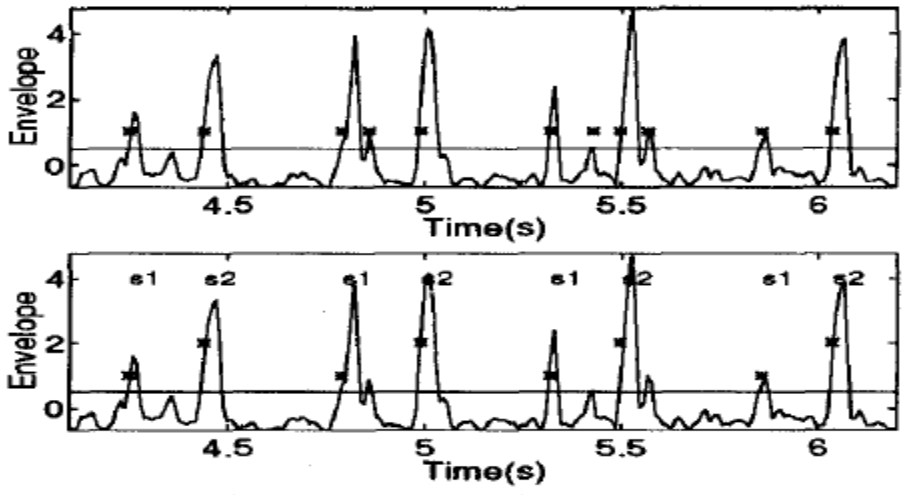
iv. Sustained high frequency sounds may be mixed in with the recording like swallowing sounds, speech and abnormal valve closure among other disturbances. These noises are identified by computing jitter.

Mandal et al. [5] performed decomposition of the signals upto level 5 by using Daubechies 6 (db6) wavelets. Levels 4, 5 and 6 detail coefficients were used to calculate the Shannon entropy after which the subbands were scaled and added up to give a partial reconstruction. The normalized Shannon energy signal envelope was computed and passed through a peak detector.

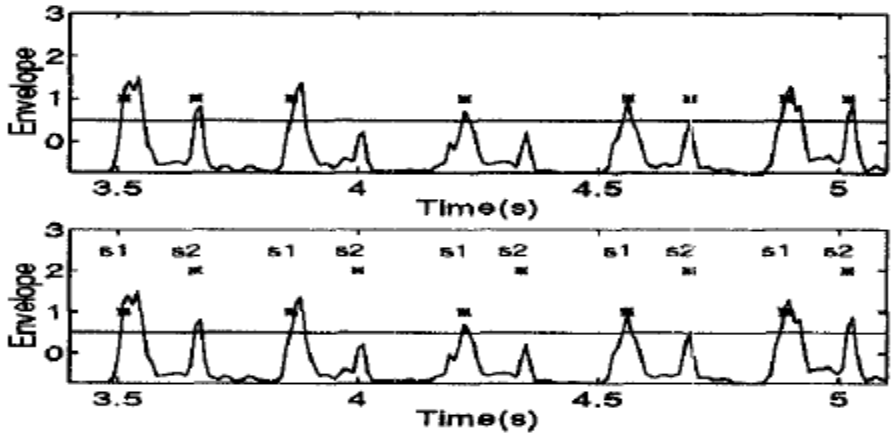
2.3 Peak detection and identification of S1 and S2s

The relevant peaks are then identified by introducing a threshold.

Liang et al. [1] further delved into this process. Extra peaks were rejected by imposing a time-limit computed for each recording based on the mean interval and standard deviation. Any peaks occurring before the computed time was considered to be irrelevant, and disregarded. Similarly, an upper time-limit was used to search for any lost peaks. If none was found within that time, the threshold was lowered until one was picked up.



Rejection of extra peaks.



Recovering the "lost" peaks.

Then, to properly classify S1 and S2 sounds, Liang's algorithm took the largest interval between two peaks as the diastolic period and marked the successive intervals, both forwards and backwards, accordingly if they were within imposed tolerance values. Hence, S1s and S2s are also identified consequently with the start of the systolic period being S1 and the end being S2.

Kumar et al. [3] utilized the fact that S2 sounds contain high frequency components that S1 sounds do not due to aortic valves closing with relatively higher pressure than mitral valves. In rare cases, like for patients using prosthetic single tilted disk valves, the S1 sounds are seen to contain the higher frequency components. Proper assignment of S1 and S2 is seen in the next section with this part discussing the determination of the location of the two sounds.

To extract the high frequency components of the segments identified previously, the Shannon energy operator is applied once more, but this time to the segments and using the FWT detail coefficients. The adaptive threshold given below is used to make out the heart cycle,

$th = E(d6) - \lambda < E(d6) >$ where, λ is a constant initially fixed to 3.0 and later adapted in certain situations

Applying this threshold, high-frequency segments (HFS) and low-frequency segments (LFS) can be detected in the segmented heart sound. A heart cycle can therefore be assumed to comprise of an LFS between two HFS with the duration between the two HFS being used to calculate the instantaneous heart rate. The duration is computed as follows,

$T_{cycle}^{k} = \frac{dt_k}{2} + T_s\left(n_{k+1}^{stop} - n_k^{start}\right) + \frac{dt_k}{2}$ where, T_{cycle}^{k} is the duration between k[th] and (k+1)[th] HFS dt_k is the duration of HFS

Two conditions are imposed for the proper identification of heart cycles.

i. If the heart cycle duration is greater than the average of the previous 3 durations, an HFS peak is assumed to have been missed. This is dealt with by lowering the HFS threshold by increasing λ in the adaptive threshold's algorithm by 0.1 intervals until the peak is found. The 3-cycle average is seen to be sufficient even in arrhythmic cases.
ii. If extra HFS markers are picked up due to noisy segments in the heart cycle, the correct one is identified by checking which peak was present in the previous cycle.

S1 and S2 sounds are classified by first estimating the systolic (S1-S2) interval duration from the cycle duration and comparing with every segment pair duration. The interval

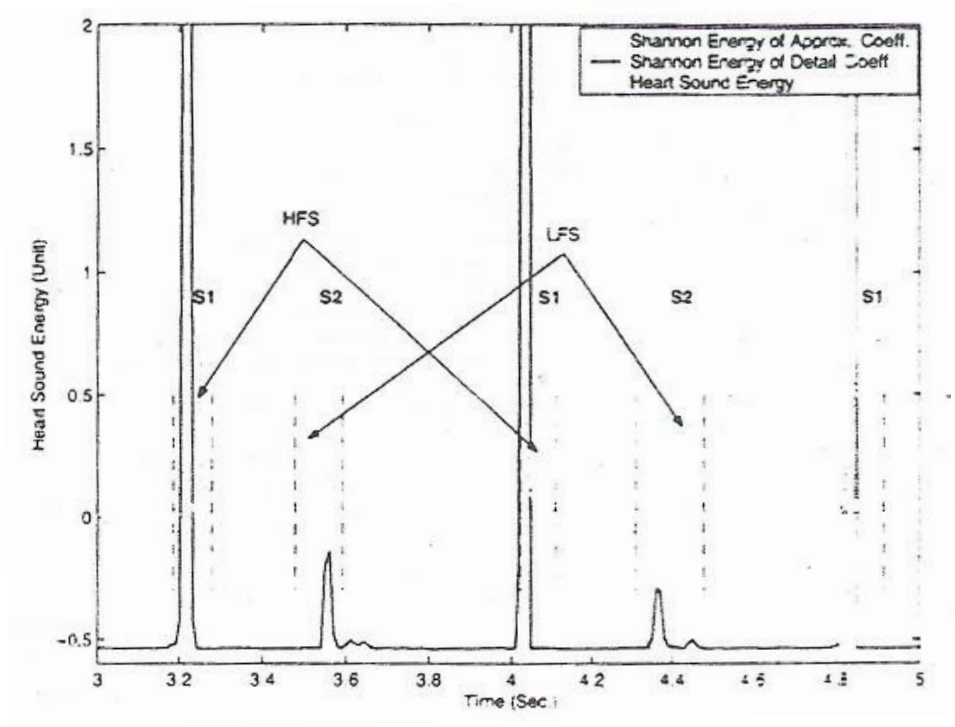
with the least deviation from t he estimate is considered to be the S1-S2 interval with the starting segment designated as the S1 sound and the end segment, the S2 sound. In order to account for extreme arrhythmic cases which might render this classification process inefficient, the following steps are performed.

i. HFS cannot automatically be assigned to be S2 sounds in atypical cases, so proper identification of S1 and S2 sounds requires further checks to be performed. A correctly detected heart cycle is chosen containing 2 HFS and 1 LFS and the systolic interval is estimated from the cycle duration using the formula below,

$T_{sys}^{est,k} = 0.2T_{cycle}^{k} + 160(ms)$ where, $T_{sys}^{est,k}$ is the estimated systolic interval of the k^{th} cycle

This estimated duration is compared to the two intervals (HFS-LFS and LFS-HFS) comprising the cycle and the interval with the smallest deviation from the estimated value is taken as the systolic interval. Hence, the starting segment class (either HFS or LFS) can be assigned as the S1 sound.

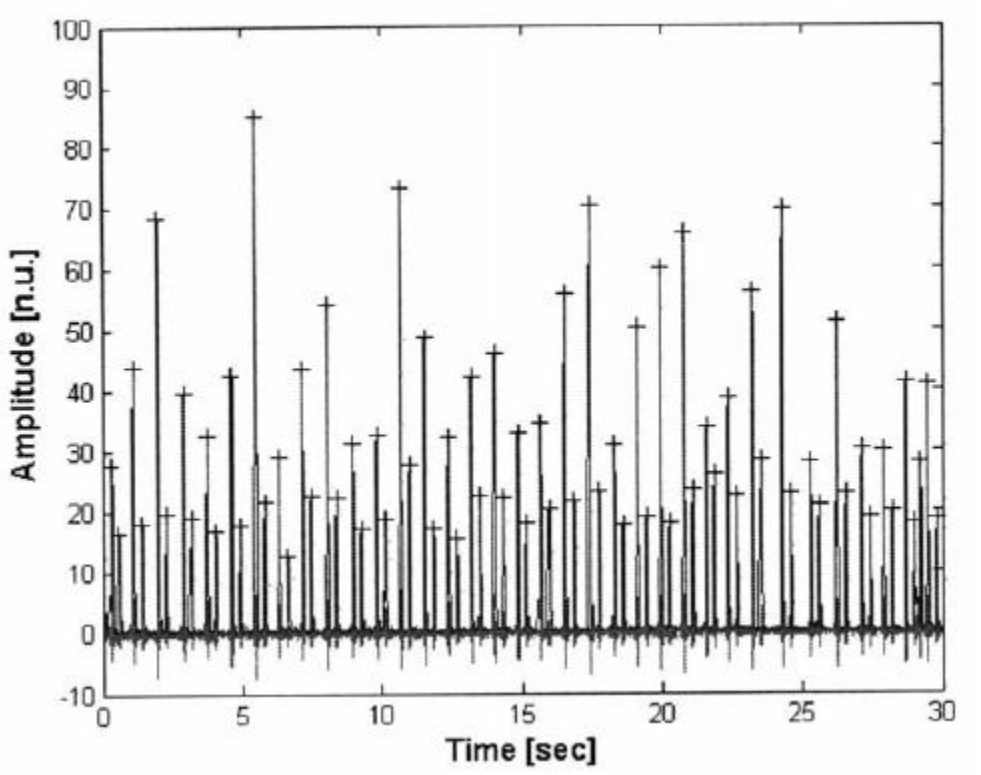
ii. The other segment class is assigned as the S2 sound.



Extraneous LFS segments might exist in the signal creating giving rise to more than one LFS segments between two HFS. The irrelevant LFSs can be removed by comparing the estimated systolic interval with all the possible LFS-HFS (or HFS-LFS, if HFS is assigned as S1) pair durations. The pair with the least deviation from the estimate is considered to be the correct LFS-HFS pair with all other LFSs making up the other pairs removed.

Kuan [4] did not discuss any methods used to properly pick up any missing peaks or eliminate extra peaks. She only used a threshold to pick up the peaks belonging to the S1/S2 sounds. After locating the peaks, intervals between the peaks were tabulated. This resulted in two Gaussian-like clusters. The intervals closer to the cluster with the smaller

time period were considered the systolic periods and the other intervals, the diastolic periods with the relative peaks being assigned as S1 and S2 sounds.



Fei Yu et al. [2] also used a simple threshold. He then manually isolated the two peaks and plotted their frequency spectrums to pick out distinguishing characteristics. It was observed that there are discernible differences between the two spectrums at the frequencies 15Hz and 65Hz. The S1 beat possesses higher valued components at around 15Hz than the S2 beat and similarly, S2 contains components with greater amplitude at 65Hz than S1. Hence, a bandpass filter with cutoff frequencies at 10Hz and 20Hz was used to eliminate all S2 peaks from the signal leaving only the S1s. This signal was then normalized and the energy square calculated using the formula,

$$Y_{norm}(t) = \frac{[Y(t)]^2}{\max([Y(t)]^2)}$$ where, Y is the amplitude of the signal

The S1 peaks were evident when the above normalized signal was plotted against time. The peak locations can then be stored but to correct for the delay created by the filter the time value, taken to be T_{max}, S1 is assumed to start 0.26s earlier than T_{max}. The S1 peaks can be extracted and saved by taking out a 0.3s interval window from the plot starting from the acquired location. The same procedure was applied to find the S2 peak locations and the S2 extracted signal. Murmurs can also be isolated using the same algorithm.

The extracted signals can be convolved with the original signal to detect the quality, intensity and locations of S1s, S2s and murmurs. This can be used to monitor variation of the heart sound signal and perform diagnosis.

In the paper by Mandal et al. [5], after passing the normalized Shannon entropy envelope through the peak detector, the subbands are again scaled and added up to give a partial reconstruction amplifying S1-S2 factors. Any peaks detected are sent to a boundary-calculation algorithm and adaptive thresholding is used to recalculate peak points in case none are detected. No further details of the algorithms were provided nor any mention of correctly distinguishing the S1 and S2 segments.

2.4 Heart rate calculation

Liang et al. [1] did not delve into this section, their paper emphasized on the correct detection of peaks and their locations.

Fei Yu et al. [2] computed energy square of the heart sound signal and passed it through a low-pass filter with a cutoff frequency of 10Hz. The resulting signal was then normalized and a threshold of 0.2 was used to correctly identify the peaks. Taking three of the peak locations, the heart rate was calculated as given below.

$$T = \frac{\Delta n}{f_s} = \frac{n_3 - n_1}{f_s}$$

where, $f_s$is the sample frequency, n is the index, $\Delta n = n_3 - n_1$ is the number of samples in a heart beat cycle, and T is time period.

$$R_{hb} = \frac{60}{T} = \frac{60 f_s}{\Delta n}$$

where, R_{hb} is the heart beat rate

Kuan [4] calculated the instantaneous rate by taking the median of the S1-S1 interval values over 9 beats around specified index peak location.

$$HR_i = \frac{60}{median(SS_{-4}, SS_{-3,\ldots,} SS_3, SS_4)}$$

where, HR_i is the instantaneous heart beat rate at i^{th} beat,

and, SS_i is the S1-S1 interval between beat i and $i+1$

Kumar et al. [3] calculated the instantaneous heart rate previously (as detailed in the previous section) and used it for S1/S2 classification.

Mandal et al. [5] did not include any mention of heart beat rate calculation in their paper.

2.5 Results

The algorithm developed by Liang et al. [1] was able to correctly identify peaks with about 93% accuracy and, tested under different conditions remained fairly robust in its efficiency. Misinterpretations were due to large background noises and serious murmurs.

Fei Yu et al. [2] did not provide such specific numbers as their focus was primarily on the fabrication of an intelligent electronic stethoscope.

Kuan [4] used an ECG signal as a reference to test their system. The algorithm was able to correctly pick up peaks from the heart sound signal with 100% accuracy. It was also able to positively predict S1 peak locations with an accuracy of 88.4% and had a sensitivity of 92.1%.

Kumar et al. [3] were able to achieve an average sensitivity of 97.95% and specificity of 98.20% testing on a range of patients possessing mechanical valves, bio-prosthetic valves and native valves. In the best case heart sound, 100% sensitivity and specificity was achieved and in the worst case sample, 95.67% sensitivity and 96.12% specificity. Wrong

detection was mainly the cause of relevant adjacent segments being affected during removal of noise. The achieved sensitivity of noise detection by jitter approach was 92.20%, though high frequency noisy segments lasting less than 50ms were not able to be detected. A manual inspection of corresponding ECG signals' QRS complexes and T-waves was used as the reference.

Mandal et al. [5] calculated the noise reduction and heart sound recovery percentages using the formulae below and compared the different filters and denoising techniques.

$$HS_{recover} = \frac{1 - E\{x_{HS}^2(n)\} - E\{y^2(n)\}}{E\{x_{HS}^2(n)\}} \times 100\%$$

$$NOISE_{reduction} = \frac{E\left\{x_{hs_{noi}}^2(n)\right\} - E\{y^2(n)\}}{E\{x_{hs_{noi}}^2(n)\}} \times 100\%$$

where, $x_{HS}(n)$ is the original heart sounds signal and $x_{hs_{noi}}(n)$ is the one corrupted with noise

The LMS-ALE algorithm resulted in an HS recovery of 91-93% with 89-92% noise reduction. RLS-ALE turned up 95-97% HS recovery with 90-92% noise reduction. Both BayesShrink and SureShrink algorithm recovered the heart segments with 96-98% accuracy with 88-91% noise reduction with BayesShrink and 91-92% noise reduction with SureShrink. Considering the above results, the SureShrink algorithm is seen to provide the best performance.

2.6 Conclusion

We observed that heart sound segmentation and peak extraction results were generally quite favourable but none of the papers delved into matters concerning environmental noise contamination of the recordings, the conditions under which the signals were taken and the effectiveness of their algorithms in non-ideal circumstances. Incorrect identification of peaks mostly occurred when faced with atypical heart signals indicating possible presence of cardiovascular diseases (CVDs). Carrying out diagnoses when dealing with such signals would prove more robust if compared with a database of heart sounds of patients with various CVDs. Hence, we conclude by highlighting that this field has room for more research and improvements which, if realized, could vastly help mitigating the increasing number of heart related deaths.

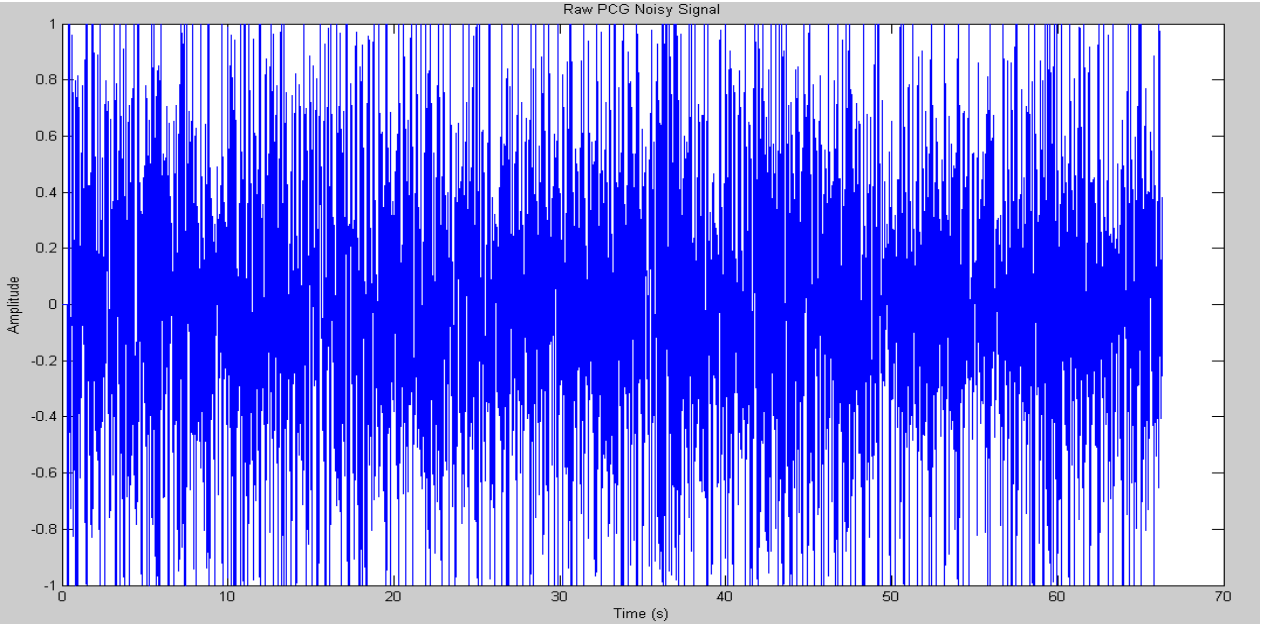
3. Our Approach

We have opted to work in the time domain rather than frequency domain as most of the noise introduced into the recordings are of low frequency and have components in the same region as our signal of interest. This makes it quite difficult to extract and distinguish the signal's frequency characteristics over the noise's spectrum. Complicated frequency domain analysis is required to work with signals contaminated to this level, which in turn necessitates the availability of significant processing power and time. As our aim is to eventually port our software for use in cellphones where processing power is limited, frequency domain analysis is hardly feasible.

Below is an outline of our developed noise tolerant algorithm for calculation of heart rate using PCG signals.

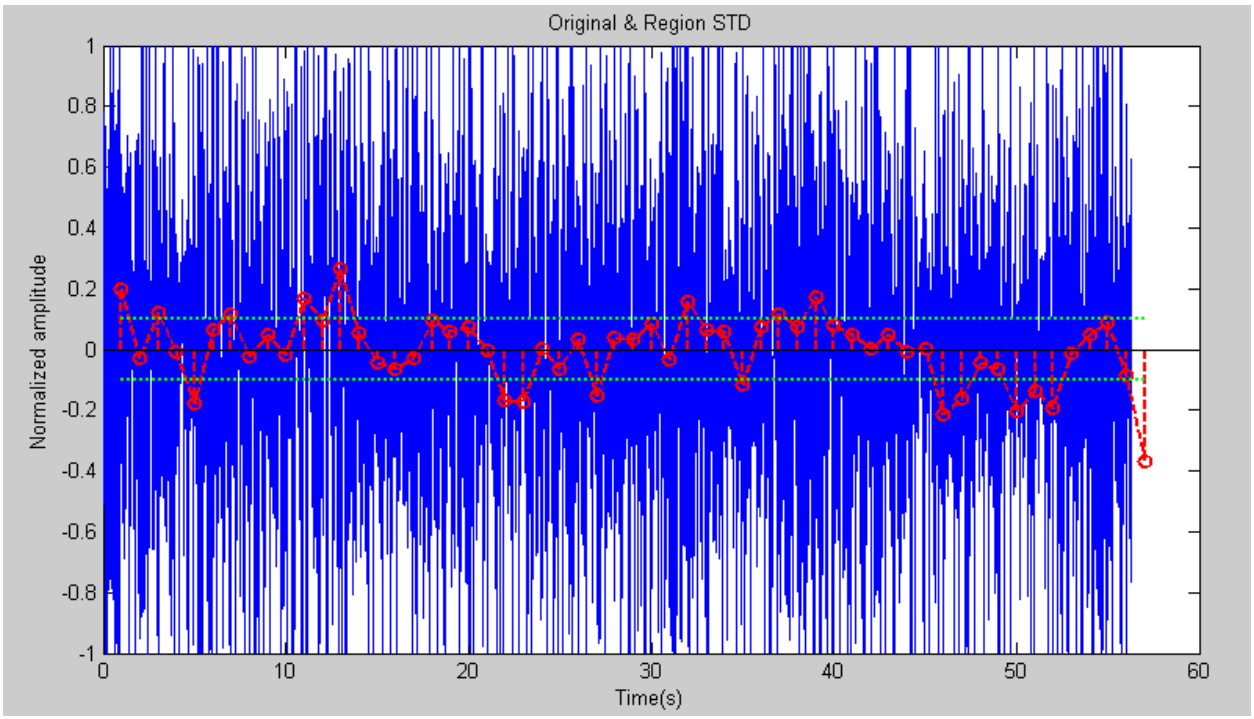
3.1 Signal pre-processing

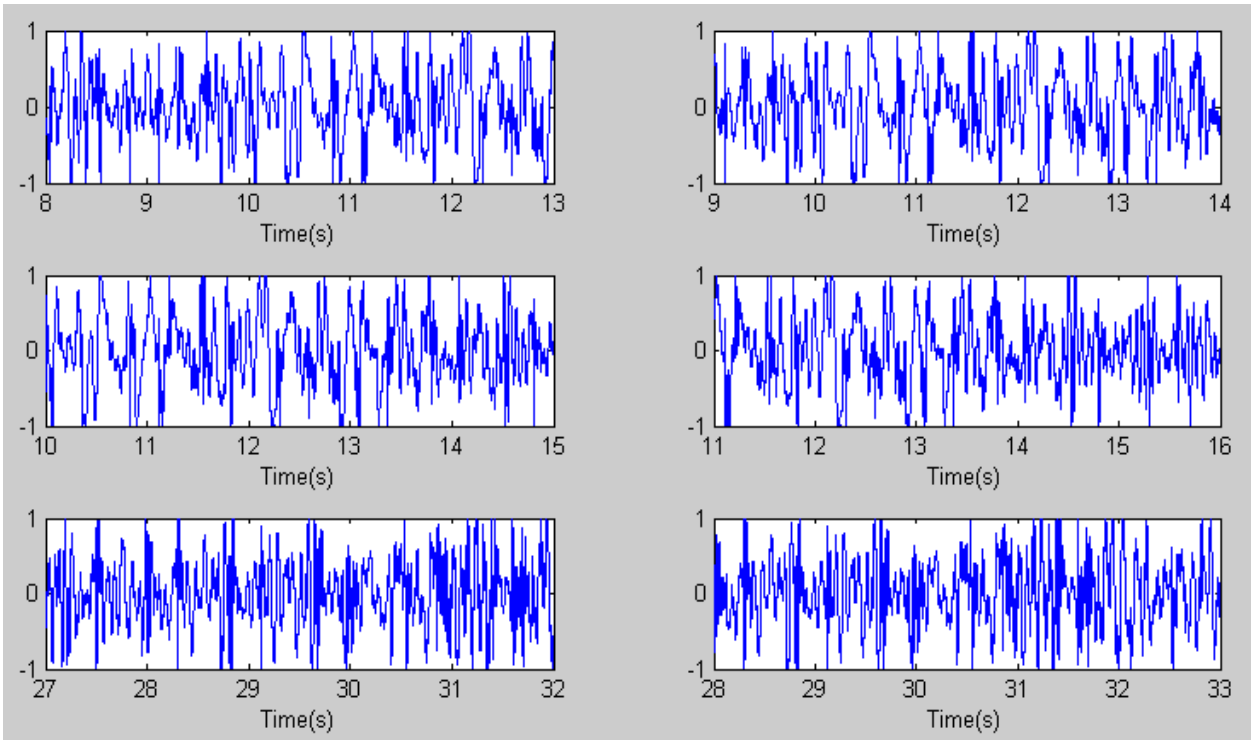
We start by reading the recorded heart signal into MATLAB. The first and last 5 seconds of the signal were discarded as those segments are usually contaminated with placement noises. The remaining signal was then normalized by subtracting the mean and dividing by the maximum value of amplitude.



3.2 Extracting regions of interest

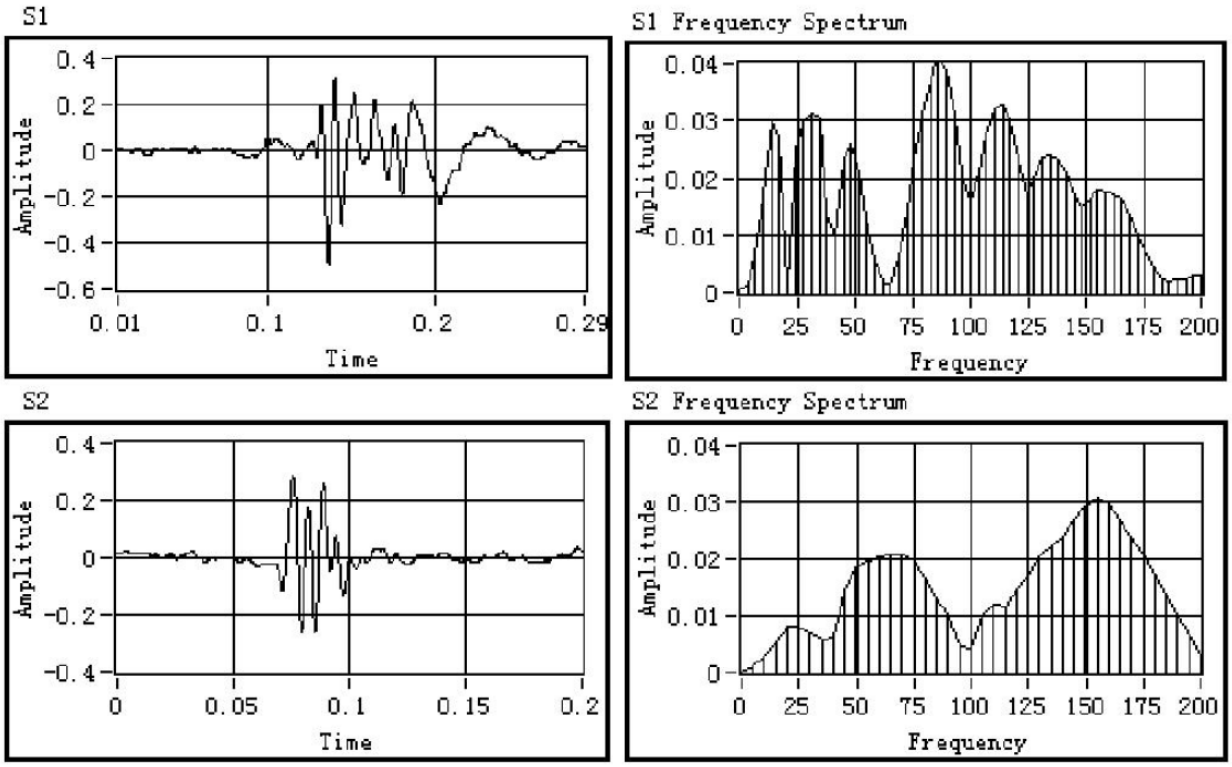
The normalized signal was then divided into windows 1 second in length and the standard deviation value of each window was calculated using all the points enclosed by the window. It was observed that windows with lower standard deviation values tended to contain less noise than those with higher values, so 5 successive windows with standard deviation values within the defined threshold of +0.1 to -0.1 were taken as a region of interest. All 5 second intervals satisfying the criterion are extracted and stored for subsequent signal processing steps.


Original & Region STD
Normalized amplitude
Time(s)

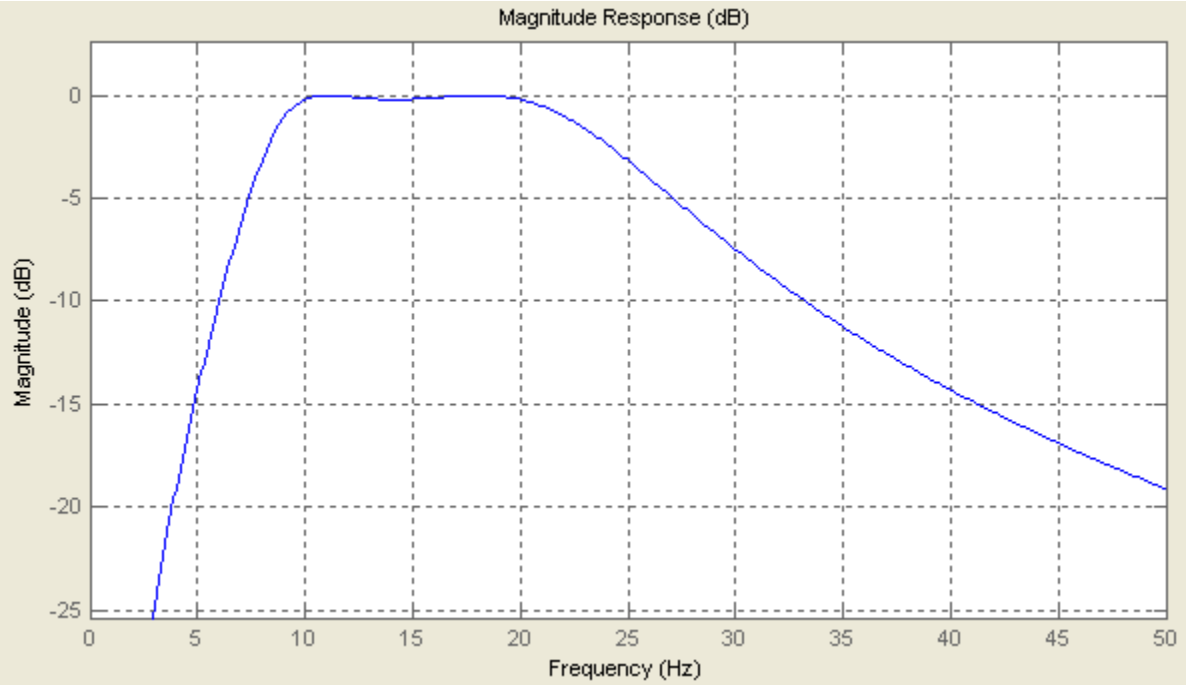

Time(s)
Time(s)
Time(s)
Time(s)
Time(s)
Time(s)

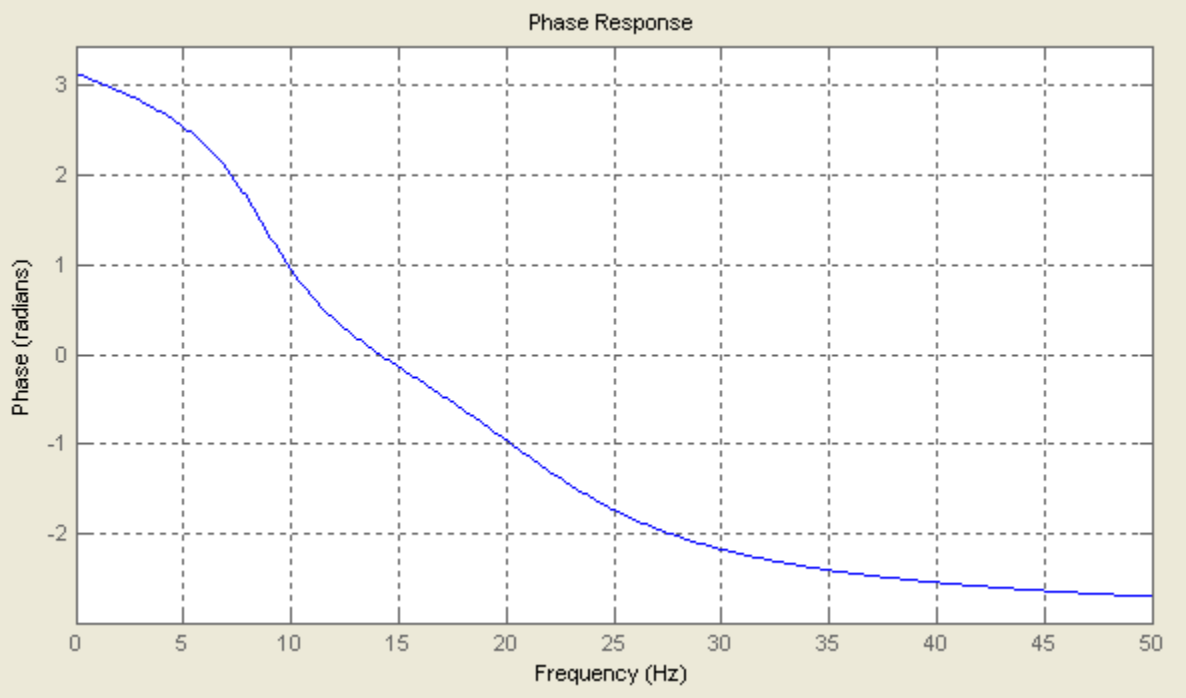
3.3 Filtering

The extracted signals were then run through a Chebyshev Type-I bandpass filter with cutoff frequencies at 10Hz and 20Hz. According to a research by Fei Yu et al. [2], observing the frequency spectrums of isolated S1 and S2 sounds, peaks at 15Hz and 85Hz were found to be present in the S1 spectrum which are absent in S2 and could be used to distinguish between S1 and S2 sounds. As the heart rate can be determined using either all the S1 sounds or the S2s, the louder and more distinct S1 sounds were chosen to be kept suppressing the S2 sounds using the filter proposed. Furthermore, the 15Hz peak was used to keep the S1s rather than 85Hz as another major source of noise contamination which are lung sounds mostly have frequency components in the 60-100Hz region. Hence, lung sounds are also effectively eliminated using the chosen filter.



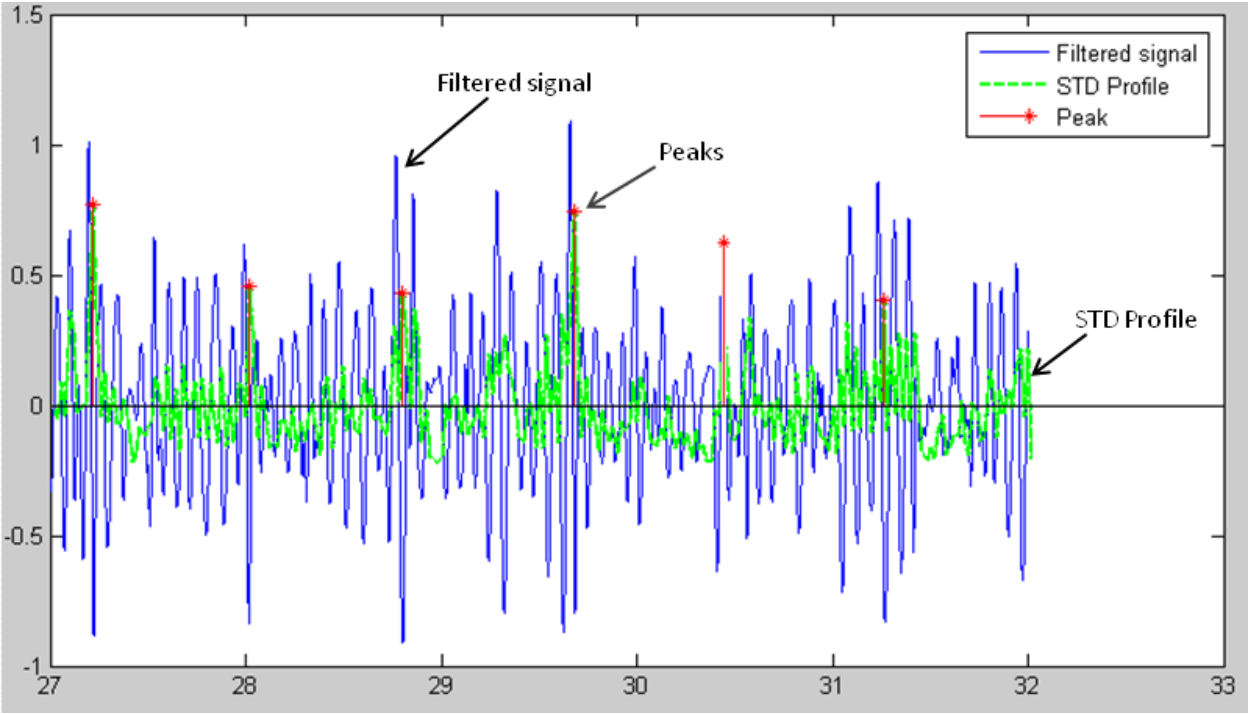
3.4 Magnitude & Phase Response of designed filter



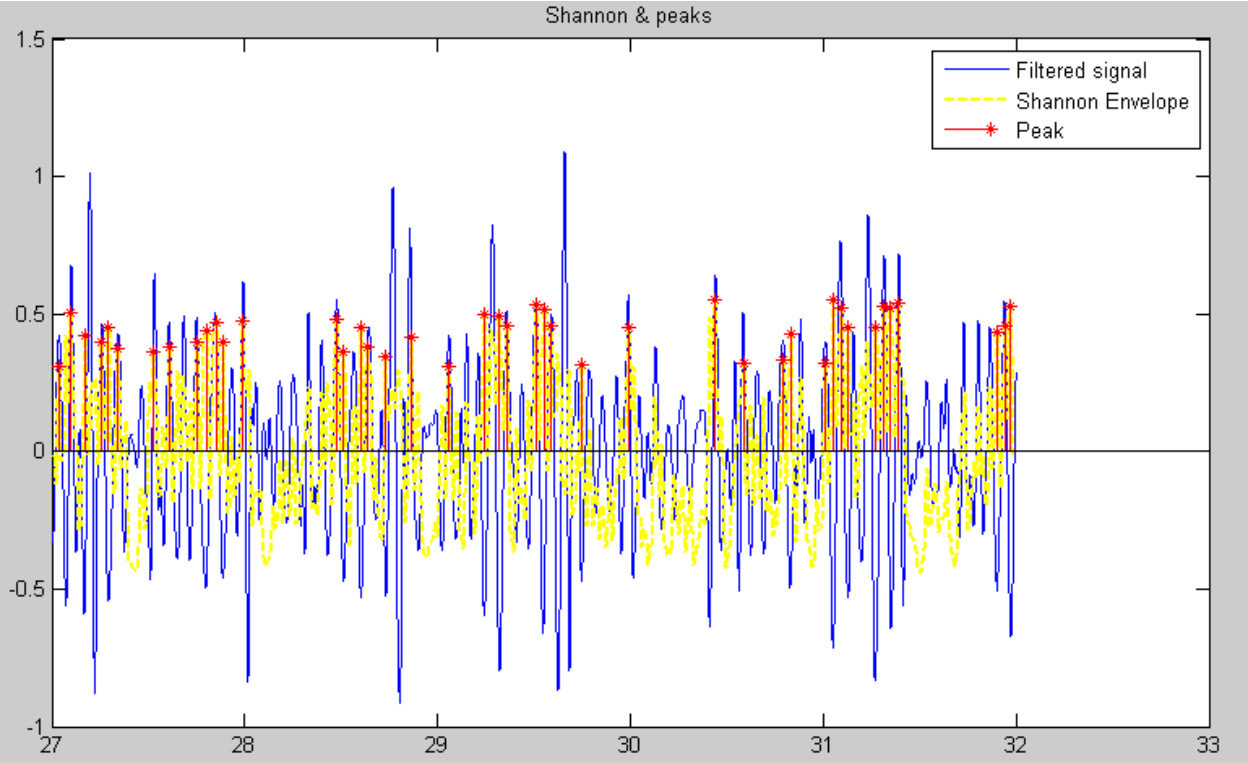


3.5 Peak detection

The filtered signals were then again divided into windows but this time of 0.02s lengths and their standard deviation values computed. The objective this time is to determine the S1 peak locations. Windows containing the peaks were seen to have higher standard deviation values. Adaptive thresholds were used to determine the windows with these high values and designate them to be S1 peak locations. Extra peaks that might have been picked up around these S1 sounds are rejected by using a time limit. If peaks are picked up around 20ms of each other, the peak with the maximum amplitude is kept and the others are discarded.



We have also implemented Liang's [1] algorithm using the Shannon energy profile for peak detection as a basis of comparison. It is clearly evident from the plot below that Shannon energy profile performs poorly when dealing with noisy signals.



Our proposed algorithm is seen to be more tolerant to noise and hence, more robust.

The time intervals between the remaining S1 peaks are computed and the mean interval is used to calculate the heart rate using the formula shown.

$$HR = \frac{60}{\Delta T} \qquad \text{where, } \Delta T = \text{mean interval between peaks}$$

4. Results and Analysis:

4.1 Sample results:

Below are some of the selected results obtained using noisy PCG signals using both the our algorithm and the one proposed by Liang [1].

Subject ID	**Oximeter reading**	**Calculated HR**	**Percentage Error**	**Shannon HR**	**Percentage error**
1	63	64	1.59	66	4.76
2	80	80	0	80	0
5	83	102	22.9	66	20.5
8	73	80	9.58	67	8.22
9	71	63	11.3	94	32.4
11	90	101	12.2	61	32.2
13	95	90	5.26	103	8.42
14	73	71	0	71	0

4.2 Analysis Using GUI in MATLAB

4.2.1 Introduction to GUI:

In computing, **graphical user interface** (**GUI**, sometimes pronounced 'gooey') is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLI), which require commands to be typed on the keyboard.

The actions in GUI are usually performed through direct manipulation of the graphical elements. Besides in computers, GUIs can be found in hand-held devices such as MP3 players, portable media players, gaming devices, household appliances, office, and industry equipment. The term ***GUI*** is usually not applied to other low-resolution types of interfaces with display resolutions, such as video games (where HUD is preferred), or not restricted to flat screens, like volumetric displays because the term is restricted to the scope of two-dimensional display screens able to describe generic information.

4.2.2 GUI in MATLAB:

MATLAB supports developing applications with graphical user interface features. MATLAB includes GUIDE (GUI development environment) for graphically designing GUIs. It also has tightly integrated graph-plotting features. GUIs (also known as

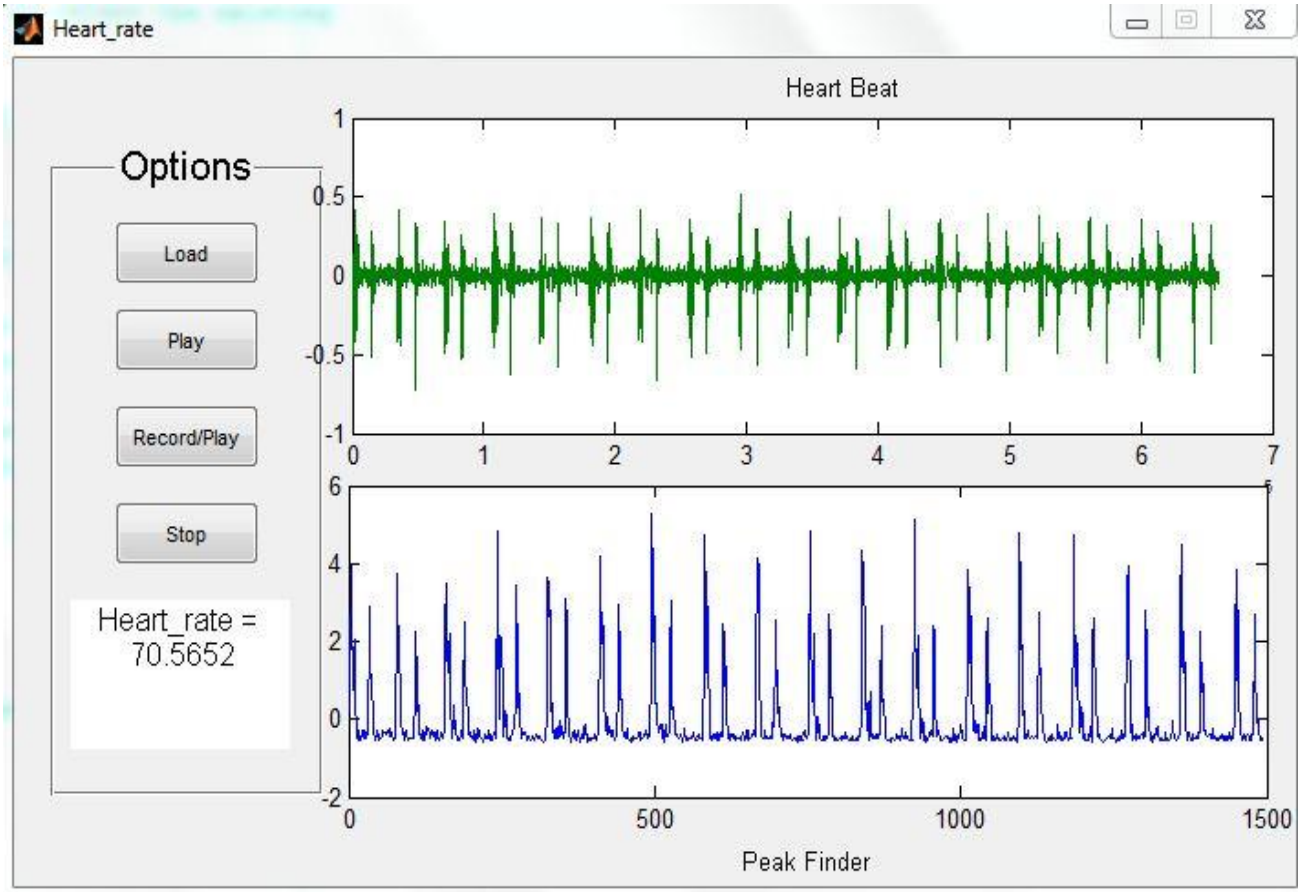
graphical user interfaces or UIs) provide point-and-click control of software applications, eliminating the need to learn a language or type commands in order to run the application.

MATLAB apps are self-contained MATLAB programs with GUI front ends that automate a task or calculation. The GUI typically contains controls such as menus, toolbars, buttons, and sliders. Many MATLAB products, such as Curve Fitting Toolbox, Signal Processing Toolbox, and Control System Toolbox, include apps with custom user interfaces. You can also create your own custom apps, including their corresponding UIs, for others to use.

GUIDE (GUI development environment) provides tools for designing user interfaces for custom apps. Using the GUIDE Layout Editor, you can graphically design your UI. GUIDE then automatically generates the MATLAB code for constructing the UI, which you can modify to program the behavior of your app.

In our proposed thesis we have used the GUIDE in order to establish a GUI for our signal processing purposes.

4.3 GUI output:



In our GUI, we have the following outputs for the user's benefits:

1. Load: loading a file save in .wav format and then the GUI automatically shows the corresponding Heart Beat and the Peak Finder Output. The GUI will also calculate the heart rate from the proposed algorithm and will display it at the GUI.

2. Play: Once the play button is pushed, the three outputs are shown instantly, and the sound of the heart beat is heard during play.

3. Record/Play: Recording the heart sound live and simultaneously listening to the heart sound with the three outputs shown.

4. Stop: when Stop button is pushed, the playing heart beat sound will stop immediately until played again.

The following GUI program built is mentioned in Appendix E.

5.Conclusion and Discussion:

5.1 Conclusion:

- From our comparative analysis and the results section, we can conclude that, through our proposal the accuracy of heart rate calculation increased using standard deviation profile.
- When the environment is noisy, our proposal of standard deviation profile showed much better results for heart rate calculations than the 'shanon' energy profile.
- From our proposed method, it is possible to calculate the heart beat with significant amount of accuracy without using any external hardware system.
- The Hardware used for the proposed method of ours would cost approximately $8.

5.2 Conclusion(Hardware):

- Cheap/simple: costs around $8 which is quite affordable in developing countries.
- User Friendly: A Digital Stethoscope does not require any training or guidance for usage.
- Reliable: The device itself is highly reliable with no signs of wear.
- Fast Analysis: Using the MATLAB GUI instant results of heart rate and heart beat outputs were found using this device.

- Environmentally friendly: The device itself does not contribute to any harm to the society.

5.3 Future works:

Our thesis has a promising future and can be used for further research in different aspects including:

- Increasing the number of varied heart sound signal samples for improving the accuracy of the project. The more samples are used, the better will be the comparative analysis and a stronger backing for our proposal will exist.
- Port our software for use in cellphones for the benefit of the society and for common use in households.
- Research more on signal processing methods to make a polished algorithm for better results and comparative analysis.

6. References

[1] Liang, H.; Lukkarinen, S.; Hartimo, I., "Heart sound segmentation algorithm based on heart sound envelogram," *Computers in Cardiology 1997*, vol., no., pp.105,108, 7-10 Sep 1997

[2] Fei Yu; Bilberg, A.; Voss, F., "The Development of an Intelligent Electronic Stethoscope," *Mechtronic and Embedded Systems and Applications, 2008. MESA 2008. IEEE/ASME International Conference on* , vol., no., pp.612,617, 12-15 Oct. 2008

[3] Kumar, D.; Carvalho, P.; Antunes, M.; Henriques, J.; Eugenio, L.; Schmidt, R.; Habetha, J., "Detection of S1 and S2 Heart Sounds by High Frequency Signatures," *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE* , vol., no., pp.1410,1416, Aug. 30 2006-Sept. 3 2006

[4] Kuan, Katherine L., "A framework for automated heart and lung sound analysis using a mobile telemedicine platform", Thesis (M. Eng.)--Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 2010

[5] Mandal, S.; Basak, K.; Mandana, K. M.; Ray, A.K.; Chatterjee, J.; Mahadevappa, M., "Development of Cardiac Prescreening Device for Rural Population Using Ultralow-Power Embedded System," *Biomedical Engineering, IEEE Transactions on* , vol.58, no.3, pp.745,749, March 2011

[6] Gretzinger, David T.K., "Analysis of Heart Sounds and Murmurs", Thesis (M.Sc.) —University of Toronto, Institute of Biomedical Engineering, 1996

Appendix A: Computed heart rates using both Shannon Energy profile and Standard Deviation profile on noisy signals

Subject ID	Oxi Rate	Shannon HR	% error	STD HR	% error	% error diff.	Avg. % error Shannon	Avg. % error STD
1	63	66	4.77	64	1.59	3.18	19.37	16.74
2	80	80	0	80	0	0		
3	67	70	4.48	78	16.42	-11.94		
4	63	92	46.04	65	3.18	42.86		
5	83	66	20.49	102	22.9	-2.41		
6	63	66	4.77	90	42.86	-38.09		
7	95	87	8.43	85	10.53	-2.1		
8	73	67	8.22	80	9.59	-1.37		
9	71	94	32.4	63	11.27	21.13		
10	93	88	5.38	94	1.08	4.3		
11	90	61	32.23	101	12.23	20		
12	55	106	92.73	95	72.73	20		
13	95	103	8.43	90	5.27	3.16		
14	73	71	2.74	91	24.66	-21.92		

<u>Appendix B</u>: Computed heart rates using both Shannon Energy profile and Standard Deviation profile on signals recorded under controlled conditions

Subject ID	Oxi Rate	Shannon HR	% error	STD HR	% error	% error diff.	Avg. % error Shannon	Avg. % error STD
15	75	73	2.67	75	0	2.67	4.26	3.4
16	62	62	0	62	0	0		
17	65	65	0	65	0	0		
18	68	70	2.95	66	2.95	0		
19	72	73	1.39	75	4.17	-2.78		
20	64	61	4.69	65	1.57	3.12		
21	67	66	1.5	67	0	1.5		
22	65	66	1.54	61	6.16	-4.62		
23	70	74	5.72	66	5.72	0		
24	80	79	1.25	79	1.25	0		
25	66	54	18.19	60	9.1	9.09		
26	88	92	4.55	83	5.69	-1.14		
27	60	68	13.34	57	5	8.34		
28	67	67	0	63	5.98	-5.98		
29	78	79	1.29	78	0	1.29		
30	67	61	8.96	65	2.99	5.97		

Appendix C: MATLAB program for heart rate computation using Standard Deviation profile

```
clear all
close all

%% Get the list of the audio files
files=dir(fullfile('C:','Educational','DSP_Thesis','*.wav'));
for k=1:1;
close all
name=files(k).name;
[y1, Fs, nbits] = wavread(name);
L1=size(y1,1);
ymid=y1(5*Fs:(L1/Fs-5)*Fs,1);
L=size(ymid,1);
t=(0:L-1)/Fs;
plot(t,ymid);hold on

%% STD Profile for all (S1 and S2) peaks
resolution=.02; % window size 0.02sec
fun= @(x) std (x);
B1 = blkproc(ymid,[Fs*resolution 1],fun);
Bnorm1=(B1-mean(B1))/max(B1);
Tb=[0:size(Bnorm1,1)-1]*resolution; % multiplied by 0.02sec to convert
to time axis
plot(Tb,Bnorm1, '--g','LineWidth',2)


%% Filter Section

 [B,A]=cheby1(2,0.2, [10 20]/(Fs/2), 'bandpass');
h1=dfilt.df2(B,A);
fvtool(h1,'FrequencyScale','log'); % to see filter response
yw=filter(h1,ymid);
figure; spectrogram(yw)

%% STD Profile for ROI extraction
res=1; % window size 2sec
fun= @(x) std (x);
B2 = blkproc(ymid,[Fs*res 1],fun);
Bnorm2=(B2-mean(B2))/max(B2);
Tb=[1:size(Bnorm2,1)]*res; % multiplied by res to convert to time axis
figure;
plot(t,ymid);hold on
plot(Tb,Bnorm2, '--r','LineWidth',2);title ('Original & Region STD');
hold on
stem(Tb,Bnorm2, '--r','LineWidth',2);title ('Original & Region STD');
xlabel('Time(s)');ylabel('Normalized amplitude');

S=std(Bnorm2);
M=mean(Bnorm2);
plot ([Tb(1) Tb(end)], [0.1 0.1], ':g','LineWidth',2); %[S+M S+M]
plot ([Tb(1) Tb(end)], [-0.1 -0.1], ':g','LineWidth',2);%[-S+M -S+M]
plot ([Tb(1) Tb(end)], [S+M S+M], ':g','LineWidth',2); %[S+M S+M]
```

```
plot ([Tb(1) Tb(end)], [-S+M -S+M], ':g','LineWidth',2);%[-S+M -S+M]

Br= find (Bnorm2 >= -0.1 & Bnorm2 <= 0.1);
D=diff (Br); % to find a continuous ROI
OnesRatio= length (find (D==1))/length (Br); %check tif there are
enough continuous regions (i.e. 1s)
if OnesRatio >= 0.4
            fprintf ('Calculating Heart-Rate...WAIT\n')
else
    fprintf ('Please Take AGAIN');name
    continue
end
SetOnes = findstr(D', [1 1 1 1]);
for p=1:length (SetOnes)
    extractPTS1= SetOnes (p);
    tst(p)=extractPTS1;
    extractPTS2= extractPTS1+5;
    ExtMat (p,:)= yw (extractPTS1*Fs:extractPTS2*Fs); % extracting from
YW as the whole signal is filtered
    ExtMat1 (p,:)= ymid (extractPTS1*Fs:extractPTS2*Fs);

subplot(3,2,p);plot((extractPTS1*Fs:extractPTS2*Fs)/Fs,ExtMat1(p,:));xl
abel('Time(s)');

subplot(3,2,p);plot((extractPTS1*Fs:extractPTS2*Fs)/Fs,ExtMat(p,:));xla
bel('Time(s)');
end

%% STD Profile
for h= 1: size (ExtMat,1)
yext= ExtMat (h,:)';
res=0.02; % window size 0.02sec
fun= @(x) std (x);
B = blkproc(yext,[Fs*res 1],fun);
Bnorm=(B-mean(B))/max(B);
Tb=[1:size(Bnorm,1)]*res; % multiplied by 0.02sec to convert to time
axis

figure;
plot(tst(h)+((1:length(yext))/Fs),yext)
hold on;
plot(tst(h)+Tb,Bnorm, '--g','LineWidth',2);title ('Extracted & STD')
hold on;

[r,c]=peakfinder(Bnorm,(max(Bnorm)-min(Bnorm))/4,0.4,1);
plot(Tb+tst(h),Bnorm,'g');
stem((r*res)+tst(h), c,'r','Marker', '*'); title('STD & Peaks')
legend('Filtered signal','STD Profile','Peak')

%% Extra peak removal
II=1;
tlmt=10;
while II<length(r);
    if (r(II+1)-r(II))<=tlmt;
        if c(II)>c(II+1);
            c(II+1)=NaN;
```

```
            c = c(~isnan(c));
            r(II+1)=NaN;
            r = r(~isnan(r));
        else
            r(II)=0;
            c(II)=0;
            II=II+1;
        end
    else
        II=II+1;
    end
end
r(r==0)=NaN;
r = r(~isnan(r));
c(c==0)=NaN;
c = c(~isnan(c));
figure;plot(tst(h)+((1:length(yext))/Fs),yext);hold
on;plot(Tb+tst(h),Bnorm,'g'); hold on; stem(r*res+tst(h),
c,'r','Marker', '*'); title('STD & Peaks')

%% HR calculation
J=diff(r);
Hr(h)=60/(mean(J)*res);
figure;
stem(r, c,'r','Marker', '*'); hold on; plot(Tb+tst,Bnorm);title ('Peak
detection')
end
Hr(find(Hr<=20 | Hr>=120))=NaN; %removing all heartrates lower than 55
and higher than 120
Hr = Hr(~isnan(Hr));
STDHR(k)=mean(Hr);
names{k}=name;

end

k;      %no. of files read
names
HR
STDHR
```

Appendix D: MATLAB program for heart rate computation using Shannon energy profile

```
clear all
close all

%% Get the list of the audio files
files=dir(fullfile('C:','Educational','DSP_Thesis','*.wav'));
for k=1:1;

name=files(k).name;
[y1, Fs, nbits] = wavread(name);
L1=size(y1,1);
ymid=y1(5*Fs:(L1/Fs-5)*Fs,1); % deleting first 5 and last 5 second
L=size(ymid,1);
t=(0:L-1)/Fs;
plot(t,ymid);hold on


%% STD Profile for all (S1 and S2) peaks
resolution=.02; % window size 0.02sec
fun= @(x) std (x);
B1 = blkproc(ymid,[Fs*resolution 1],fun);
Bnorm1=(B1-mean(B1))/max(B1);
Tb=[0:size(Bnorm1,1)-1]*resolution; % multiplied by 0.02sec to convert
to time axis
plot(Tb,Bnorm1, '--g','LineWidth',2)

%% Filter Section

[B,A]=cheby1(2,0.2, [3 300]/(Fs/2), 'bandpass');
h1=dfilt.df2(B,A);
fvtool(h1,'FrequencyScale','log'); % to see filter response
yw=filter(h1,ymid);
figure; spectrogram(yw)

%% STD Profile for ROI extraction
res=0.5; % window size 2sec
fun= @(x) std (x);
B2 = blkproc(ymid,[Fs*res 1],fun);
Bnorm2=(B2-mean(B2))/max(B2);
Tb=[1:size(Bnorm2,1)]*res; % multiplied by res to convert to time axis
figure
plot(t,ymid);hold on
plot(Tb,Bnorm2, '--r','LineWidth',2);title ('Original & Region STD');
hold on
stem(Tb,Bnorm2, '--r','LineWidth',2);title ('Original & Region STD');
hold on

S=std(Bnorm2);
M=mean(Bnorm2);
```

```
plot ([Tb(1) Tb(end)], [0.1 0.1], ':k','LineWidth',2); %[0.1= arbitary
STD value]
plot ([Tb(1) Tb(end)], [-0.1 -0.1], ':k','LineWidth',2);%[-0.1=
arbitary STD value]
plot ([Tb(1) Tb(end)], [S+M S+M], ':g','LineWidth',2); %[S+M S+M]
plot ([Tb(1) Tb(end)], [-S-M -S-M], ':g','LineWidth',2);%[-S+M -S+M]

%% envelope detection_Shannon METHOD

yext=yw; % DO NOT TAKE ymid for energy calculation, it gives NaN
N=length(yext);
windowRes=0.02;
Ns=(Fs*windowRes); % number of samples per window
Nw= floor((N/Ns)*2); % number of windows

i=1;
for c= 1:Nw
    x=yext(i:i+Ns/2-1,1);
    Es(c)= -sum((x.^2) .* log (x.^2))/(Ns/2); % Energy calculation
    i=i+Ns/2;
end

n=1;
for j=1:length(Es)
    if j==length(Es), break, end
    Es1(j)= sum(Es(n:n+1))/2;
    n=n+1;
end

P= (Es1-mean(Es1,2))/std(Es1);
Tp=[1:size(Es1,2)]*0.01;
figure
plot(t,ymid, 'b');hold on
plot(Tp,P, '--r','LineWidth',0.5)
close all
%% Peak finder
[r,c]=peakfinder(P,(max(Bnorm1)-min(Bnorm1))/.4, 2,1);
figure
stem(r, c,'r','Marker', '*'); hold on; plot(P);title ('Peak detection')

Dp=diff(r); % to get S1-S2 interval in terms of P points
funsum= @(x) sum (x);
Dnew= blkproc (Dp, [1 2],funsum); % adding smaller and bigger gap= gap
between S1 and S2
Fr=find (Dnew>=55 & Dnew<=120); % maxHR=120 to minHR=40 % finding
locations of bigger gaps
DelN=Dnew (Fr).* windowRes * Fs ./2; % to get S1-S2 interval in terms
of signal points
heartRate (k)= mean(60*Fs./DelN) % Average HR calculated from all
intervals= hr of that segment

names {k}= name;
HRinfo = {heartRate , names};
load OxiHR
error=heartRate -OxiHR(1:size(heartRate ,1));
```

```
close all
end
```

Appendix E: MATLAB program for GRAPHICAL USER INTERFACE (GUI)

```
function varargout = Heart_rate(varargin)
% HEART_RATE M-file for Heart_rate.fig
%      HEART_RATE, by itself, creates a new HEART_RATE or raises the
existing
%      singleton*.
%
%      H = HEART_RATE returns the handle to a new HEART_RATE or the
handle to
%      the existing singleton*.
%
%      HEART_RATE('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in HEART_RATE.M with the given input
arguments.
%
%      HEART_RATE('Property','Value',...) creates a new HEART_RATE or
raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Heart_rate_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Heart_rate_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 Heart_rate

% Last Modified by GUIDE v2.5 09-Nov-2012 01:54:30

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Heart_rate_OpeningFcn, ...
                   'gui_OutputFcn',  @Heart_rate_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 Heart_rate is made visible.
function Heart_rate_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 Heart_rate (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = Heart_rate_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 pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
[filename1, pathname1] = uigetfile('*.wav','select a wave file to
load');

global fs
global nbits
global y
global z

nbits=8;
fs=11025;
[y,fs,nbits]=wavread([pathname1 filename1]);
```

```
%sound(y);
z= audioplayer(y,fs)



a=wavread([pathname1 filename1]);
plot(handles.Heart_rate,a);

% low pass filter to discard high-freq components
[B,A]=cheby1(2,0.2, 200/(fs/2), 'low');
h1=dfilt.df2(B,A);
% fvtool(h1,'FrequencyScale','log'); % to see filter response
yw=filter(h1,y);
% Nrm=(yw.^2)/max(yw.^2); % normalization
Nrm=yw/max(abs(yw)); % normalization



%% envelope detection_Shannon METHOD
N=length(Nrm);
window=0.02;
Ns=(fs*window); % number of samples per window
Nw= floor((N/Ns)*2);

% fun1=@(x) -sum ((x.^2) .* log (x.^2))/(Ns/2); % Energy calculation
% B1 = blkproc(Nrm,[Ns/2 1],fun1);
% fun2= @(x) mean (x);
% B2 = blkproc(B1,[2 1],fun2);

i=1;
for c= 1:Nw
    x=Nrm(i:i+Ns/2-1,1);
    Es(c)= -sum((x.^2) .* log (x.^2))/(Ns/2); % Energy calculation
    i=i+Ns/2;
end



n=1;
for j=1:length(Es)
    if j==length(Es), break, end
    Es1(j)= sum(Es(n:n+1))/2;
    n=n+1;
end

P= (Es1-mean(Es1,2))/std(Es1);
 [r,c]=peakfinder(P,(max(P)-min(P))/4, 0.02,1);
% figure
% stem(r, c,'r','Marker', '*');
% hold on;
%
% title ('Peak detection')

%%heartrate calculation
%intervals
```

```
I=1;
K=1;
L=1;
f=0;
g=0;
%J=1;
for I=1:(numel(r))-1;
J(I)=r(I+1)-r(I);   %calculating intervals between successive peaks and
storing in matrix 'J'
end

%systolics using mean partitioning
for I=1:numel(J);
if J(I)< mean(J)    %calculating mean of interval matrix J and all
intervals that lie below the mean are considered systolic periods,
these periods are stored in matrix 'f'
f(K)=J(I);
K=K+1;
end
end

%diastolics using mean partitioning
for I=1:numel(J);
if J(I)> mean(J)    %same is done to obtain the diastolic periods
g(L)=J(I);
L=L+1;
end
end

%filtering abnormal values
meanf=mean(f);
stdf=std(f);

for I=1:numel(f);
while f(I) == NaN;
I=I+1;
end
if f(I) < meanf - stdf || f(I) > meanf + stdf;
f(I) = NaN; % all values outside 1 std of mean of 'f' is converted to
NaNs
end
end

f = f(~isnan(f));   % all NaN's omitted from matrix 'f'

meang=mean(g);
stdg=std(g);

for I=1:numel(g);
while g(I) == NaN;
I=I+1;
end
if g(I) < meang - stdg || g(I) > meang + stdg;
g(I) = NaN; % all values outside 1 std of mean of 'g' is converted to
NaNs
```

```
end
end

g = g(~isnan(g));   % all NaN's omitted from matrix 'g'

t=(mean(f)+mean(g))*window;
hr=(60*2)/t

plot(handles.Peak_finder,P);
%plot(P);


%left=y(:,1);            % Left channel
%right=y(:,2);           % Right channel
%subplot(2,1,1), plot((1:length(left))/fs, left,'linewidth',2),grid on;
%title('LEFT Channel')
%subplot(2,1,2), plot((1:length(right))/fs, right,'linewidth',2),grid
on;
%title('RIGHT Channel')% handles    structure with handles and user
data (see GUIDATA)


% Create string for output that we can send to the text control.
text5 = sprintf('Heart_rate = %.4f', hr);
% Now, send the string to the control called txtInfo
set(handles.text5, 'String', text5);


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global y1
global x
global fs
global z

s2=get(handles.pushbutton2,'Value');
if s2==1
play(z)
else
stop(z)
end
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Create string for output that we can send to the text control.
txtInfo = sprintf('Heart_rate = %.4f', hr);
% Now, send the string to the control called txtInfo
```

```
set(handles.txtInfo, 'String', txtInfo);




% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global fs
global nbits
global y
global z
fs = 11025;
t = wavrecord(15*fs,fs,'int16');
wavwrite(y,fs,'new')
wavplay(t)


% nbits=8;
% fs=11025;
% [y,fs,nbits]=wavread(t);
% %sound(y);
% z= audioplayer(y,fs)



a=wavread('new');
plot(handles.Heart_rate,a);

% low pass filter to discard high-freq components
[B,A]=cheby1(2,0.2, 200/(fs/2), 'low');
h1=dfilt.df2(B,A);
% fvtool(h1,'FrequencyScale','log'); % to see filter response
yw=filter(h1,y);
% Nrm=(yw.^2)/max(yw.^2); % normalization
Nrm=yw/max(abs(yw)); % normalization



%% envelope detection_Shannon METHOD
N=length(Nrm);
window=0.02;
Ns=(fs*window); % number of samples per window
Nw= floor((N/Ns)*2);

% fun1=@(x) -sum ((x.^2) .* log (x.^2))/(Ns/2); % Energy calculation
% B1 = blkproc(Nrm,[Ns/2 1],fun1);
% fun2= @(x) mean (x);
% B2 = blkproc(B1,[2 1],fun2);
```

```
i=1;
for c= 1:Nw
    x=Nrm(i:i+Ns/2-1,1);
    Es(c)= -sum((x.^2) .* log (x.^2))/(Ns/2); % Energy calculation
    i=i+Ns/2;
end


n=1;
for j=1:length(Es)
    if j==length(Es), break, end
    Es1(j)= sum(Es(n:n+1))/2;
    n=n+1;
end

P= (Es1-mean(Es1,2))/std(Es1);
[r,c]=peakfinder(P,(max(P)-min(P))/4, 0.02,1);
% figure
% stem(r, c,'r','Marker', '*');
% hold on;
%
% title ('Peak detection')

%%heartrate calculation
%intervals
I=1;
K=1;
L=1;
f=0;
g=0;
%J=1;
for I=1:(numel(r))-1;
J(I)=r(I+1)-r(I);   %calculating intervals between successive peaks and
storing in matrix 'J'
end

%systolics using mean partitioning
for I=1:numel(J);
if J(I)< mean(J)    %calculating mean of interval matrix J and all
intervals that lie below the mean are considered systolic periods,
these periods are stored in matrix 'f'
f(K)=J(I);
K=K+1;
end
end

%diastolics using mean partitioning
for I=1:numel(J);
if J(I)> mean(J)    %same is done to obtain the diastolic periods
g(L)=J(I);
L=L+1;
end
end
```

```
%filtering abnormal values
meanf=mean(f);
stdf=std(f);

for I=1:numel(f);
while f(I) == NaN;
I=I+1;
end
if f(I) < meanf - stdf || f(I) > meanf + stdf;
f(I) = NaN; % all values outside 1 std of mean of 'f' is converted to
NaNs
end
end

f = f(~isnan(f));    % all NaN's omitted from matrix 'f'

meang=mean(g);
stdg=std(g);

for I=1:numel(g);
while g(I) == NaN;
I=I+1;
end
if g(I) < meang - stdg || g(I) > meang + stdg;
g(I) = NaN; % all values outside 1 std of mean of 'g' is converted to
NaNs
end
end

g = g(~isnan(g));    % all NaN's omitted from matrix 'g'

t=(mean(f)+mean(g))*window;
hr=(60*2)/t

plot(handles.Peak_finder,P);

% Create string for output that we can send to the text control.
text5 = sprintf('Heart_rate = %.4f', hr);
% Now, send the string to the control called txtInfo
set(handles.text5, 'String', text5);
```

```
% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
global y
global fs
global z
stop (z)
```

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (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 edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as
a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (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
```