2016

An algorithm for extracting the PPG Baseline Drift in real-time

Tuan Ngoc Nguyen

2016

An algorithm for extracting the PPG Baseline Drift in real-time

Tuan Ngoc Nguyen

Edith Cowan University

Copyright Warning

USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.

AN ALGORITHM FOR EXTRACTING THE PPG BASELINE DRIFT IN REAL-TIME

This thesis is presented in the fulfilment of the requirements for the degree Master of Science (Computer Science)

Tuan Ngoc Nguyen

Edith Cowan University

School of Science

2016

Abstract

Photoplethysmography is an optical technique for measuring the perfusion of blood in skin and tissue arterial vessels. Due to its simplicity, accessibility and abundance of information on an individual's cardiovascular system, it has been a pervasive topic of research within recent years. With these benefits however there are many challenges concerning the processing and conditioning of the signal in order to allow information to be extracted. One such challenge is removing the baseline drift of the signal, which is caused by respiratory rate, muscle tremor and physiological changes within the body as a response to various stimuli.

Over the years there have been many methods developed in order to condition the signal such as Wavelet Transform, Cubic Spline Interpolation, Morphological Operators and Fourier-Based filtering techniques. All have their own individual benefits and drawbacks. These drawbacks are that they are unsuitable for real-time usage due to the computation power needed, or have the trade-off of being real-time at the cost of deforming the signal which is unideal for accurate analysis. This thesis aims to explore these techniques in order to develop an algorithm that can be used to condition the signal against the baseline drift in real-time, while being able to achieve good computational efficiency and the preservation of the signal form.

COPYRIGHT AND ACCESS DECLARATION

I certify that this thesis does not, to the best of my knowledge and belief:

(i) incorporate without acknowledgement any material previously submitted for a degree or diploma in any institution of higher education;

(ii) contain any material previously published or written by another person except where due reference is made in the text; or

(iii) contain any defamatory material.

Signed (signature not included in this version of the thesis)

Date…….

Acknowledgements

To Professor Kamal Alameh, I would like to give my thanks for your guidance, and for the time that was committed to helping me with the revision of my thesis drafts. From your guidance, I was able to further understand the intricacies of scientific writing.

To Dr. Hoang Nghia Nguyen, I would like to give my thanks for your guidance, for opening the doors for me, and giving inspiration to this project. Your guidance taught me invaluable lessons which helped me tremendously throughout the project.

To Mr. Paul Roach, I would like to give my thanks for your support and advice in dealing with administrative matters and more.

To my friends, my family, my parents and my sister, thank-you for being there for me during trying times. I wouldn't have been able to travel this far without your love, encouragement and support.

Finally, to my late high-school teacher Ms. Wiese, I dedicate this thesis in your memory. If it were not for your support and guidance in earlier years, my reading and writing skills would not be at the level that allowed me to open doors to be where I am today.

No words can truly express my gratitude for the amount of support you all have given me. If it were not for your constant support and encouragement along this difficult journey, this thesis would have never been possible. It is with these words that I close this chapter, and embark onwards to the many more yet to come in my ever growing saga.

Contents

List of Figures

List of Tables

List of Abbreviations and Terms

AC: Alternating Current

AHR: Average Heart Rate

ADS: Adaptive Segmentation Algorithm

ADT: Adaptive Threshold Algorithm

BP: Blood Pressure

BPM: Beats per Minute

CBSI: Cubic Spline Interpolation

DC: Direct Current

DCBD: Derivative Calculation Based Discrimination

EMD: Empirical Mode Decomposition

FFCF: The proposed algorithm name: FIR-FFT Cascaded Filter

FIR: Finite Impulse Response, a type of cut-off filter

FFT: Fast Fourier Transform

FWT: Fast Wavelet Transform

High-pass: A type of filter that isolates high-range frequencies while eliminating low-range frequencies

HR: Heart Rate

HRV: Heart Rate Variability

ICA: Independent Component Analysis

IHR: Instantaneous Heart Rate

IIR: Infinite Impulse Response, a type of cut-off filter

IMFs: Intrinsic Mode Function

LMS: Least Mean Square criteria, a measure of error between two numeric series

Low-pass: A type of filter that isolates low-range frequency while eliminating high-range frequencies

MAD: Median Absolute Deviation score

OLA: Overlap Add Algorithm

OLS: Overlap-Save Algorithm

PPG: Photoplethysmography/Plethysmogram signal

PTP: Peak-to-Peak

PTV: Peak-to-Valley

PW: Pulse Wave

PWV: Pulse Wave Velocity

R.B.D: Representative Benchmark Dataset

RR: Respiratory Rate

RT- FFCF: Real-time variant of the FFCF

SBD: Spectral Based Discrimination

WCF: Wavelet Cascaded Filter

ZPF: Zero-phase Filtering

Chapter 1

Introduction and Background

1.1. Motivations

Having being directly correlated to an individual's wellbeing due to its relation to other biological systems, the monitoring of cardiovascular health is a quintessential subject of medical practice. A study conducted in 2013 by the World Health Organisation revealed that cardiovascular diseases now account for 17 million deaths a year. Of these deaths approximately 9.4 million are attributed to Hypertension or high blood pressure, and the remaining amount due to heart complications *[1]*. In this modern age where unhealthy lifestyles are becoming more and more prevalent, the importance of monitoring one's cardiovascular system has grown even greater than before.

The heart rate (HR) corresponds to one's cardiovascular fitness or how the heart adapts in certain circumstances. It is often used as an indicator of sympathetic and parasympathetic nervous system activity in response to physiological conditions *[2, 3]*. It is necessary to track its pattern in order to determine and identify abnormalities such as arrhythmias, since these can potentially indicate a heart defect or a serious response by the nervous system that requires immediate intervention to prevent fatality *[2-5]*.

Blood pressure (BP) corresponds to blood perfusion caused by physiological influences. The pressure of blood flow in the arteries is considered a reflection of an individual's arterial health system. The organs within the body rely on a steady

regulation of blood pressure in order to maintain optimal operation, and may become damaged if not enough blood is supplied. Alternatively the arteries may become worn down over time due to constant pressure, rupturing under high blood pressure which can lead to clots, which leads to strokes and heart attacks. Consistent monitoring is deemed crucial for the diagnosis and regulation of blood pressure to ensure optimal health *[1].*

There have been many methods presented over the years to measure and track them. Intermittent measurements usually being the common method, can only provide a periodic overview of the cardiovascular system's state so as a result, these measurements cannot be considered truly accurate. In order to completely understand the cardiovascular system's behaviour, an overview of how it behaves under different scenarios over time must be available for study. Due to this, continuous monitoring has been a prevalent topic of study, and is a highly recommended practice by physicians.

There are existing methods such as tonometry, ECG, cardio phonography for HR monitoring, but all of these have the limitations of being negatively influenced by the environment, from elements such as electrical and sound interference. In addition, these methods cannot be used on a continuous basis without restricting the user's movements. For monitoring BP, the most accurate method is known to be the invasive catheter, but it possesses the potential to introduce the patient to infections and is only recommended under dire circumstances such as surgery.

The ambulatory cuff can only be used every 30 minutes which leads to a sparseness in data. Continuous operation is not possible nor recommended, as prolonged uses of the cuff will expose the patient to the risk of muscle damage *[6, 7]*. Sphygmomanometer cuffs are non-invasive but require a trained physician to operate, but it relies on the skill of the physician and can be difficult to obtain an accurate reading if the hearing of the physician is poor.

Photoplethysmography is an optical-based technique for monitoring blood flow with the arteries of the skin tissue *[8]*. Due to the information contained within the signal with its non-invasive and accessible properties, it has become a popular topic within recent years for research towards the continuous monitoring of the cardiovascular system. It is simpler than the ECG signal to process and understand due to having less characteristics. But with these advantages, it also has associated challenges with processing, since it can be contaminated by multiple sources of physiological and movement noise.

1.2. Background to the Study

Photoplethysmography *(PPG)* as described earlier is a technique for measuring blood perfusion within the arteries underlying the skin tissue *[9]*. The sensor is constructed using an infrared light emitter and a corresponding photodetector that is able to detect the infrared light emitted. These components are placed in a specified layout of either two positions *(illustrated in Fig. 1.1)*. One is known as the *"Trans-illumination"* layout, and the other is known as the *"Reflective"* layout.

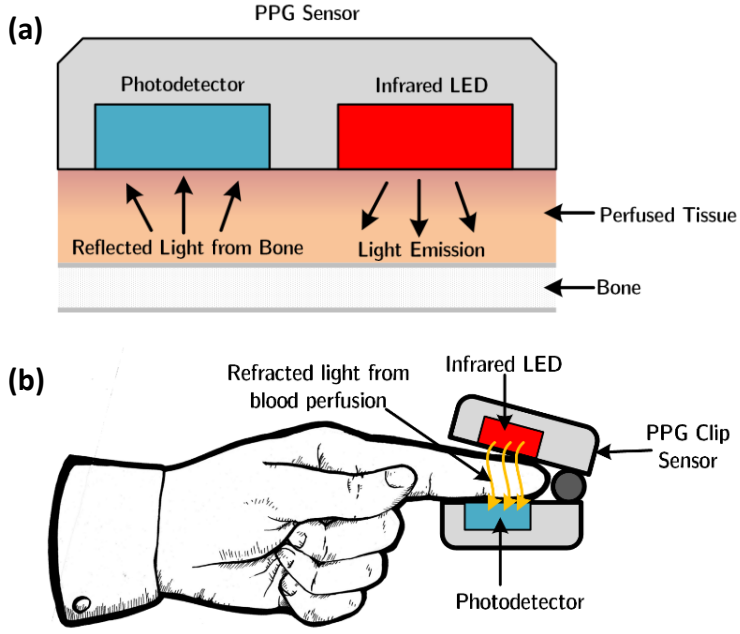


Figure 1.1 a). *Adjacent or "Reflective" PPG layout* ***b).*** *Transmission or the "Trans-illumination" PPG layout*

Light is passed through the skin tissue and undergoes a series of complex optical processes. Depending on how the sensors are positioned, reflected or refracted light is picked up by the photodetector to yield a signal of the varying blood flow within the vessels. The resulting signal *(illustrated in Fig. 1.2)* commonly referred to as a plethysmogram contains two essential elements; an Alternating Current *(AC)* and a Direct Current *(DC)*.

The AC component corresponds to the changes in blood volume levels, caused by cardiac and physiological activities such as vasoconstriction and vasodilation in response to external or internal stimuli. The DC component alternatively corresponds to respiratory rate *(RR)* changes within an individual's system. Both of these currents contain vital information that can be derived to diagnose, and monitor cardiovascular conditions along with the influences of other physiological conditions.

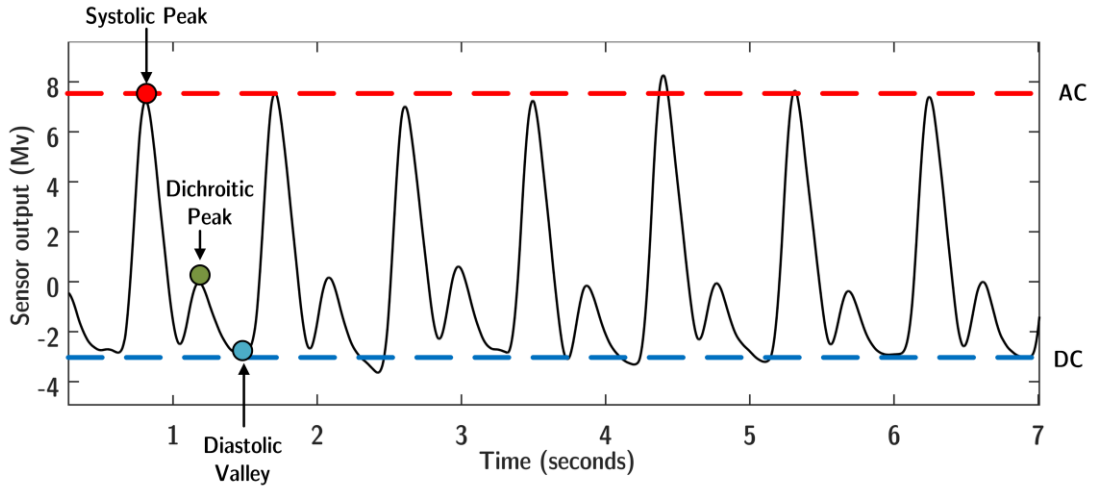


Figure 1.2 *Example of a filtered PPG signal showing its components. Sample taken from Karlen 2010 [10], file sample 0147_8min.*

1.2.1. The AC Component and Associated characteristics

The AC Component containing the higher frequencies of the signal is composed by individual fluctuations known as Pulse Waves *(PW),* whom vary in shape and size depending on the state of the individual's cardiovascular health. The PW represents a single cardiac cycle, and is defined by two predominant points that represents the systolic and diastolic activities of the heart; the "Diastolic valley" and the "Systolic peak". The "Diastolic Valley" correlates to the heart's rest period, where blood is drawn back into the heart as it prepares for the next pump. The "Systolic Peak" correlates to when the blood flow is at maximum volume from the heart pump activity.

In cases of healthy patients there exists the *"Dichroitic peak",* which correlates to blood reflecting off the ends of the distal arteries and returning back to the heart *[11]*. The average heart rate (AHR) can be derived through counting the systolic peaks that occur over a window of time. While this gives an indication of how many beats occurs a minute, it cannot be considered accurate as the average will smooth out any given variations that may have occurred *[5]*. The instantaneous heart rate (IHR) on the other hand is considered a more accurate measure of the HR, since it utilizes the distance between currently an occurring peak, and its previous neighbour to derive the possible beats that can occur in a minute (BPM).

When the IHR is measured over long periods of time, the heart rate variability *(HRV)* pattern is produced. This pattern is utilized in studies in order to determine how the heart reacts to different stimuli from the autonomous nervous system, as well as to determine the whether the heart functions are normal. An abnormal IHR may correlate to arrhythmia caused by the influences of drugs, or fatal neurological disorders such as autonomic neuropathy found in diabetic patients [2]. Some of these conditions require immediate intervention in order to prevent fatality hence, precision and speedy signal analysis is highly sought after.

Many studies have focused extensively on the usage of the valley, in order to calculate what is known as the *"Pulse Transit Time" (PTT)*. The PTT is utilized in the calculation of the pulse wave velocity (PWV), which measures how fast blood travels within arteries from one location to another [12]. The PWV is defined as:

$$PWV = \frac{\Delta x}{PTT}\ , where\ PTT = T_1 - T_2\ and\ \Delta x = x_1 - x_2 \quad (Eq.\ 1.1)$$

Using Fig 1.3 as an illustrative guide, the parameter Δx is defined as the distance between two points on the same arterial branch, which are defined as $x_{1,2}$. The parameter $T_{1,2}$ is described as time it takes for a single PW to travel between the two points. These parameters can be obtained through placing two sensors that have the ability to detect PW, on the same arterial branch, separated by a small distance.

The PTT is reliant on the exact location of the valley in order to allow for accurate PWV calculations hence, distortions are generally undesirable and are removed where possible.

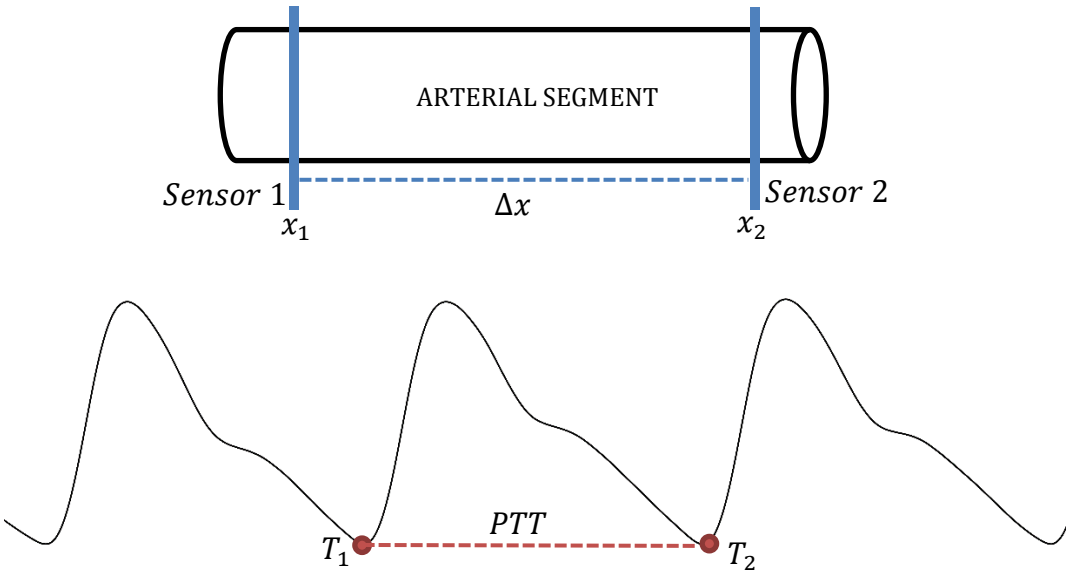


Figure 1.3 *Obtaining parameters to calculate the PWV and PTT.*

The PWV has been agreed upon by many studies to be directly correlated to arterial compliance of an individual *[13-16]*. When blood is pumped through the arteries, it relies the elasticity of the arterial walls help push blood along to the destination. As a result, the heart does not have to work hard to pump or require large amounts of blood to drive the process *[17]*. In arteries that are not elastic, the heart can no longer rely on the elasticity of the walls to push it along.

Consequently, the heart has to pump higher volumes of blood which leads to a faster PTT and in turn a faster PWV. The increased volume of blood exerts more pressure on the arterial walls which causes the individual to suffer from a condition known as *"High Blood Pressure"* or *"Hypertension"*. Older people *(illustrated in Fig 1.4)* tend to have less elastic arteries so as a result, the dichroitic peak caused by the reflection, often becomes smaller due to the force of blood dampening the return wave.

In comparison, healthier and younger patients have the dichroitic peak relatively evident in their PW signal. The state of an individual's arterial compliance and blood pressure is however not reflective of only on age, but other factors such as lifestyle and bodily responses to illness and medication. Due to this relationship, it is deemed crucial to monitor and track the PW for the diagnosis and monitoring of these conditions.

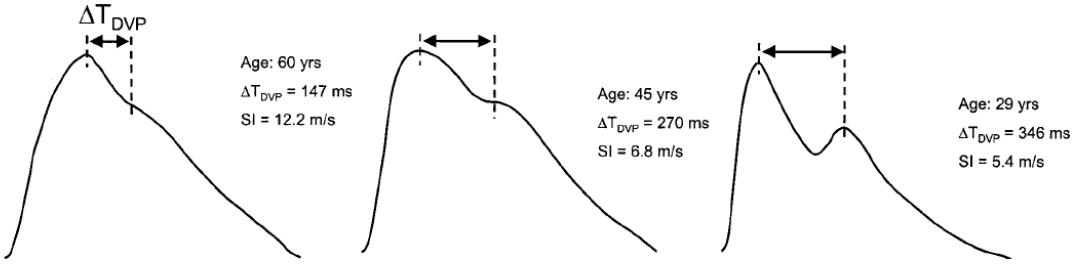


Figure 1.4 *Difference in PW shapes with age. Image taken from Elgendi 2010 [18].*

Given how easily that the PW can be obtained from PPG, the PW calculated from the PPG signal has been extensively researched in BP measurement models for non-invasive continuous BP measurement *[19-24]*. This is as an alternative to existing limited methods that may cause harm to the patient such as invasive catheters, or methods that can only be used intermittently such as the ambulatory cuff *[25, 26]*. In addition to the PWV, the shape of the PW and the PTT has been used in conjunction to evaluate the possible existence of arrhythmias *[18]*.

1.2.2. The DC Component and Associated Characteristics

The DC component normally existing at 0.1-0.5Hz on the frequency spectrum, is superimposed onto the AC component which causes additional fluctuations to the signal PW known as the *"Baseline drift" [17]*. The DC component is related to sympathetic nervous system responses to various stimuli such as temperature and at times respiration. Due its relationship with the respiratory activity, it has been the subject of studies attempting to extract an individual's respiratory rate from the PPG signal *[27-29]*.

Providing that the respiratory rates are controlled and exist within the normal range, it can be extracted through conventional means of extracting the baseline drift. This consists of using a cut-off frequency filter, specified to correspond to the desired respiratory frequency range *(shown in Fig 1.5*

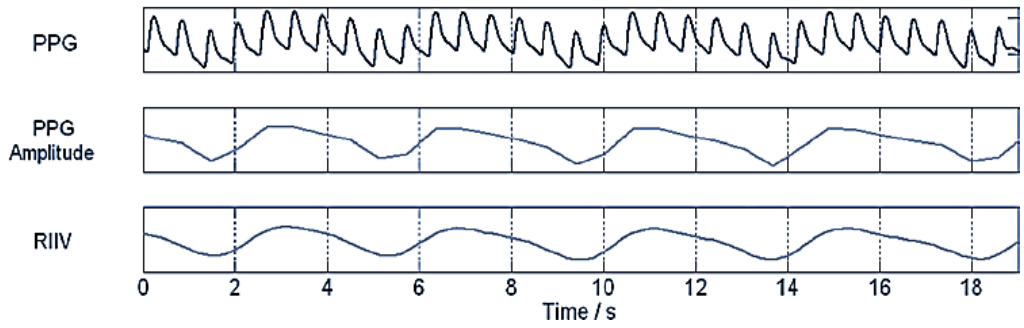


Figure 1.5 *A respiratory signal derived by low-passing the PPG signal to obtain its baseline drift. Image taken from Meredith 2012 [27].*

Another motivation for extracting the baseline drift is to normalise the signal for analysis. The DC component is superimposed onto the AC component so as a result, the features of the PPG signal may be displaced or distorted by the drift amplitudes *(shown in Fig 1.6),* thus making it difficult to identify and locate features accurately.

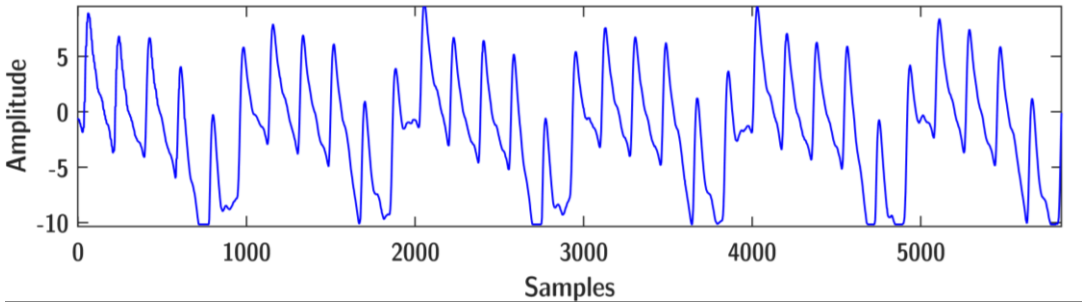


Figure 1.6 *Example of baseline wander displacing the location of the valleys within the PPG signal. Sample taken from Karlen 2010 [10], file sample 0009_8min.*

Small movements and muscle tremors by the patient are also known to add to the severity of the baseline drift *[18]*. In order to properly analyse the PPG signal, the baseline drift must be removed or at the very least, softened to reduce its negative impacts on the core signal features and allow for accurate identification. While it is trivial to apply a cut-off filter to remove the drift frequencies beneath 0.5Hz, the main challenge is that these drift frequencies may not exist at the expected level of 0.5Hz and below.

Depending on physiological and external physical conditions, they may exist above 0.5Hz, overlapping the frequencies of the HR which normally exist from 0.5Hz to 2Hz *[17]*. *Garde et al. 2013* in their study *[28]* reported respiratory rates of up to 0.75Hz, which exceeds the level of respiration for what is considered to be normal breathing. Taking this factor into consideration, the removal of the drift frequencies therefore no-longer becomes trivial.

Through using fixed cut-off frequency filters at higher ranges than the normal drift frequencies, the drift can be removed but the core features of the signal may also be removed due to overlapping of the HR and the drift frequencies *[30]*. In other circumstances, they yield poor approximates caused by time varying frequencies.

This is due to changes within the physiology of the patient *[31, 32]* which causes the RR rate frequency to rise outside of the filter's range. This challenge has led to a wide proposal of techniques to isolate and remove it ranging from the standard cut-off filter, to mathematical and signal decomposition techniques. Many methods have been used to reduce and eliminate the noise, but these methods cannot be used within real-time constraints.

In addition, they have questionable outcomes, such as the trade-off of discarding information in order to preserve computational efficiency for portable devices. In essence, the algorithms that trade off information are only suitable for measuring the heart beats which correlates to HR. These types of algorithms are not suitable for usage towards advanced and detailed analysis which requires the preservation of the PPG morphology.

1.3. Contributions of this study

In this thesis a method is explored and realised for conditioning the PPG signal against the baseline drift in real-time, while providing a means to detect the peaks and valleys on-the-go.

Specifically the algorithm presented within this thesis:

a) Applies a real-time filtering technique to soften the baseline drift complexity.

b) Detects the peaks and valleys of the PPG signal in data chunks or a sample-by-sample basis as they are processed by the filtering technique.

c) Apply an estimation technique utilizing the valleys detected previously in order to calculate and remove the remaining traces of baseline drift

The method is verified step-by-step through visual and numerical analysis against an existing wavelet-based method that has been proven to previously work in non-real time operation.

1.4. Thesis Outline

Chapter 2 reviews the work that has been undertaken in order to condition the signal against baseline drift, and the work that has been undertaken in order to derive meaningful information through the identification of the morphological features.

Chapter 3 presents the problems to be addressed, and proposes a potential path of study to resolve the presented problems and limitations within current literature. Data acquisition and analysis is undertaken, where a benchmark dataset is derived from common PPG characteristics within the acquired database. This is used for within and potentially outside the thesis, for future studies concerning the filtering of the PPG signal, and the identification and analysis of PPG morphological features.

Chapter 4 focuses on exploring the popular wavelet transform as a means for filtering the PPG signal, in comparison to the IIR and FIR filter based methods to investigate how well they are able to remove the drift. The FIR filter is investigated explicitly in order to test its viability of operating in real-time through optimization techniques.

Chapter 5 explores feature detection algorithms in order to identify an algorithm, which is suitable for usage towards the development of the final algorithm. The algorithm with the fastest computation times and the highest detection rates is selected as the optimum solution.

Chapter 6 explores a selected cascaded filter against a proposed cascaded filter design. Results of a selected wavelet cascaded filter and the proposed cascaded filter are compared for similarities, and the computation run-times of each of the algorithms are analysed for efficiency.

Chapter 7 applies the proposed cascaded filter structure to a design for a real-time cascaded filter algorithm. Simulations are carried out to investigate its real-time potential, and results are compared against a wavelet cascaded filter to assess its overall effectiveness.

Finally, Chapter 8 concludes the study efforts, and presents a list of possible future work using the algorithm for further study and refinement.

1.5. Summary

Chapter 1 presented an outline of the current state of cardiovascular diseases, and the associated methods that have been proposed within the past in order to track cardiovascular vital signs. The limitations and problems associated with these techniques are presented, and the concept of how PPG can overcome these limitations were then discussed.

The concept of PPG and the plethysmogram is explored, where a summary is provided of how cardiovascular signs such as HR and BP can be extracted from the signal. Following this, the concepts of how the signal can be used as a diagnostic tool for arterial compliance, and for the investigation of cardiovascular system functions are presented. In proceeding discussions, we explore the challenges of extracting this information from the signal, with emphasis on the preserving the signal form. Finally, an outline of the study is presented along with the contributions achieved through the research process.

Chapter 2

Background and Literature Review

This chapter reviews the concepts and past proposed techniques for conditioning the PPG signal. A review of techniques for identifying and tracking features within the PPG signal, is also presented as background for the development of the proposed cascaded filter.

2.1. Baseline Removal Techniques

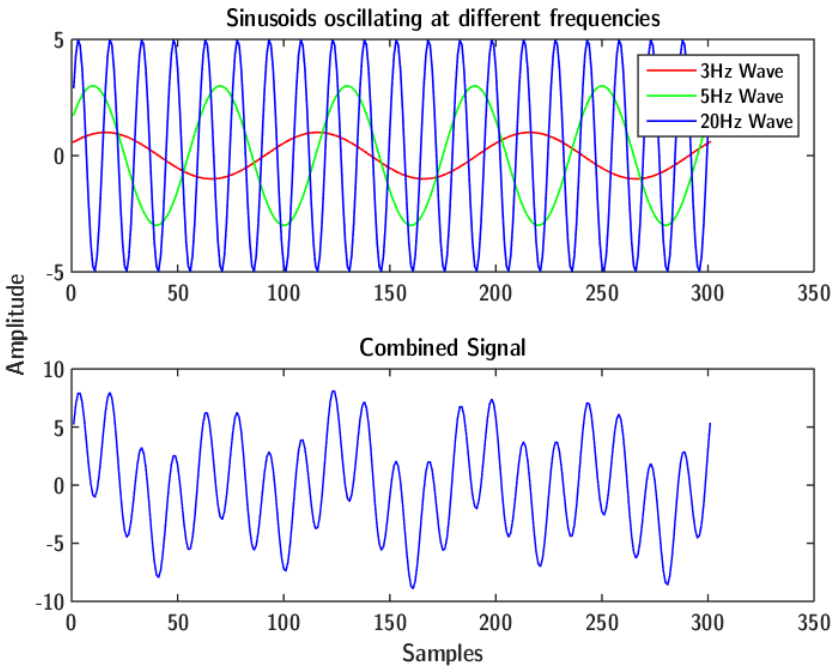
This section discusses previous techniques that have been proposed for removing the baseline drift of the PPG signal.

2.1.1. Fourier's Theorem and Cut-off Frequency Filters

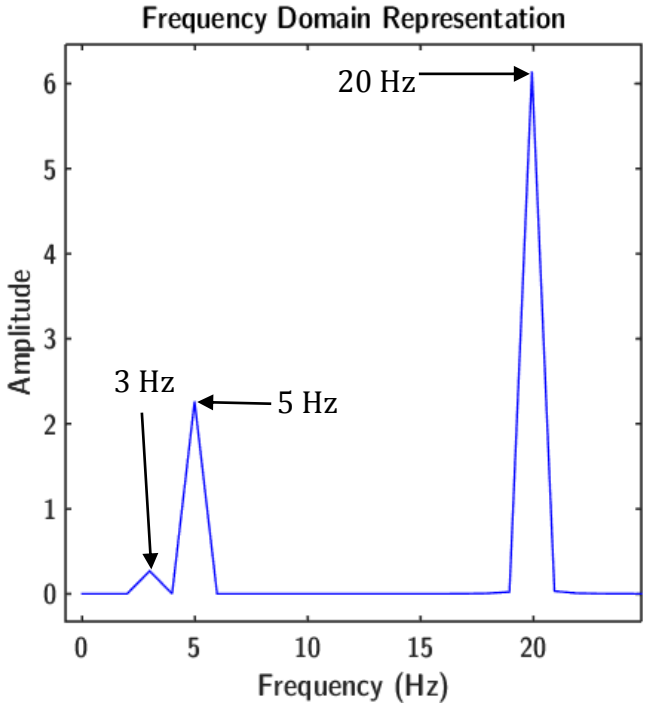
The most direct method to eliminating noise is through extracting its corresponding frequency within the signal's spectral contents. Fourier's theorem *(described in equation 2.1)* states that any given signal or time series denoted by $x(t)$, is composed by potentially the sum of an infinite number of sine and cosine sinusoids. Each sinusoid series is potentially oscillating at different frequencies denoted by f, times denoted by t and at a base amplitude a_0 *[33]*.

$$x(t) = a_0 + \sum_{n=1}^{\infty} \cos(2\pi f t n) - \sum_{n=1}^{\infty} b_n \sin(2\pi f t n) \qquad \textit{(Eq.2.1)}$$

Through using a variant of the Fourier Transform, it is possible to decompose a signal from its original state in the time domain, and into the frequency domain representation *(illustrated in Fig 2.1 a. and b.).*



(a). *A signal composed using 3Hz, 5Hz and 20Hz sinusoids in the time domain.*



(b). *FFT of the composed signal showing its spectral contents in the frequency domain.*

Figure 2.1 *Example of signal composition using two sinusoids.*

For discrete time series, the Discrete Fourier Transform *(DFT)* is used but in modern times, the Fast Fourier Transform *(FFT)* is preferred as it is more computationally efficient. The transformation process simplifies the representation of its sinusoidal characteristics in the time domain, into bins that represent the individual frequencies that correspond to each sinusoid series. The amplitude of the bins correlate to how prevalent these components are over one another within the signal.

By eliminating the frequency bins within the spectral domain, one can eliminate the noise or unwanted components, which corresponds to the eliminated frequency. The application of the convolution operation *(described in Eq. 2.2.)* and its associated properties, allows the modification to have an equal effect within the time domain on reverse transformation.

$$x(n) * y(n) \overset{DFTF}{\longleftrightarrow} H(W) * W(W) \qquad \textit{(Eq. 2.2)}$$

Where x(n) and y(n) are two time series represented in the time domain by the lower case, the frequency domain representation are given by the uppercase X(W) and Y(W). The convolution theorem states that *"any modifications done to the signal in frequency domain, will have equal effect in the time domain representation and vice versa".* Hence, through the convolution theorem, it is possible also to mathematically model a filter, based on a group of coefficients within the time domain that corresponds to the desired frequency response. Two categories of filter design commonly used are the Finite Impulse Response *(FIR)* filter and the Infinite Impulse Response *(IIR)* filter.

2.1.1.1. Finite Impulse Response (FIR) Filters

FIR filters *(illustrated in Fig 2.2)* are the simplest types of filters, utilizing only past and current sample processes in a convolution process with the filter kernel. Due to this structure, their impulse response or filter output is considered finite. Stability is usually guaranteed since their output is predictable, having being only dependent on the signal samples for convolution with the filter kernel and nothing more.

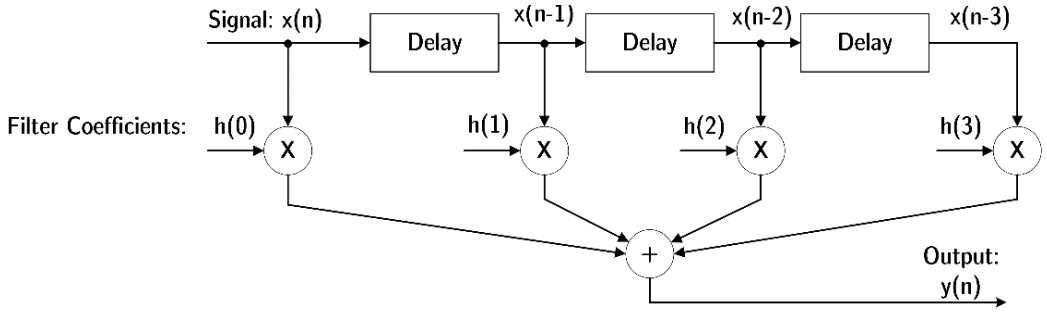


Figure 2.2 *FIR filter structure (referenced from Lyons 1997* ***[34]****). Past samples are multiplied and summed together in the convolution process with the filter kernel.*

FIR filters possesses, in addition to inherent stability the property of *"linear phase"*, where the shape of the signal on filtering is preserved with only a slight time delay of half the filter's length. This property is crucial for accurate analysis and diagnosis of biomedical signals, but the problem with FIR filters is that they may require an extremely large numbers of filter coefficients, in order to achieve the precise frequency response desired.

From studies by *Pilt et al. 2013 [35]* and *Geun et al. 2014 [36]*, a filter order of 500-650 taps has been reported as necessary for a low-pass filter to achieve a 0.5Hz cut-off. This is for the purpose of isolating the baseline drift signal caused by respiration and nervous system activities. In some circumstances the number of coefficients required had reached 4000 taps for isolating specific individual frequencies within the signal.

Given the large amount of filter taps that is required to achieve the desired frequency cut-off response, FIR filters require powerful processors and large amounts of memory to store the calculations *[36]*. Although computationally intensive, the shortcomings of the FIR filter's computation time can be overcome. This is possible through the exploitation of the FFT and the properties of convolution *[37].* As shown earlier, a signal or series within its time domain representation can be infinitely long. But when it is transformed into the frequency domain, the number of samples are limited to only the bins that represent the frequency content of the signal. Hence, by transforming both the filter coefficients and the signal into the frequency domain, the number of calculations needed can be significantly reduced, in comparison to directly filtering via direct convolution within the time domain.

This is only effective in comparison to direct convolution, when the length of the two signals are significantly long. The FFT requires the padding of the signal to the nearest power of two in order to operate. Due to this, the number of calculations may exceed that which is required, if filtering was performed directly on shorter segments. *Smith 2010 [37]* presented a calculation of the number of operations required *(presented in Table 2.1)*, and found that in order for FFT convolution to be effective, the length of both the filter kernel and the signal must be greater than at least 128 samples.

Table 2.1 *Computation costs for "Fast-Convolution" where N is the length of the filter kernel and the signal input, values taken from Smith 2010 [37].*

	Number of operations required				
N	**FFT**	**Direct Convolution**	**N**	**FFT**	**Direct Convolution**
4	176	16	**128**	13,312	16,384
32	2560	1024	**256**	29,696	65,536
64	5888	4096	**2048**	311,296	4,194,304

Note that given the large filter order required for PPG filters, this technique would be applicable for improving computational efficiency. An extension of this concept for filtering long and continuous signals in real-time, is through the application of the Overlap-Add *(OLA)* algorithm or the Overlap-Save algorithm *(OLS)*. The OLA algorithm involves partitioning the signal *(illustrated in Fig 2.3)* and then processing each segment individually. Segments are divided into lengths that are selected based on the minimum requirement length for the filter, the available computational hardware and the time-delay constraints required. The remainder of the segment is padded with zeros at the end to the nearest power of two, based on the segment's current length for efficient FFT processing.

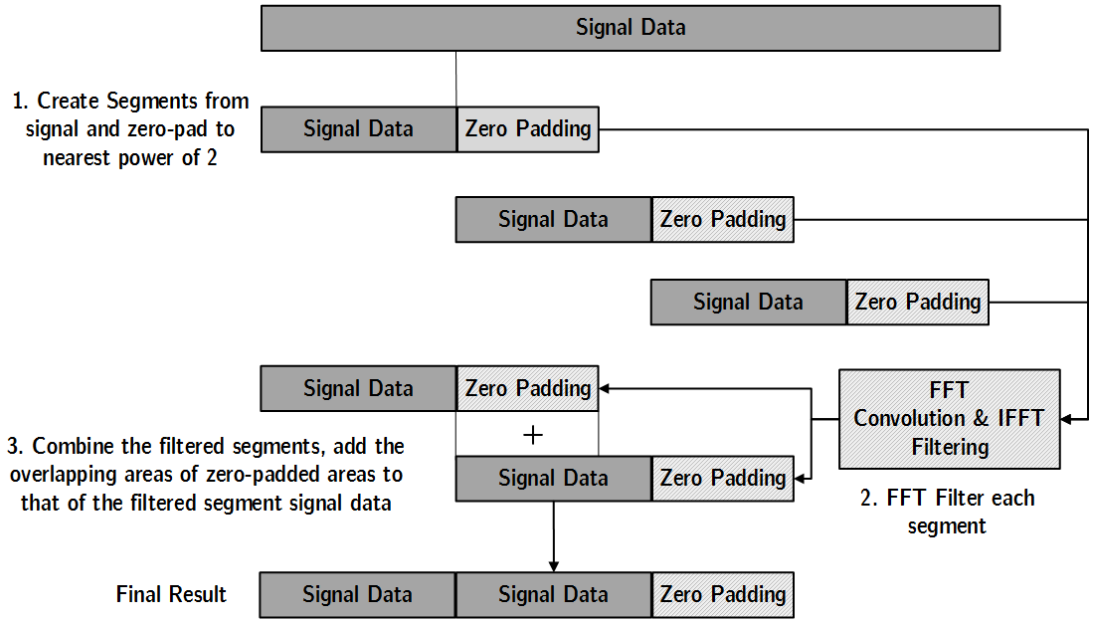


Figure 2.3 *The Overlap-Add Algorithm Process.*

On taking the FFT of each segment, the values are convolved with the FFT transformed filter coefficients. From here, the resulting filtered segment is transformed back into the time domain. The start of the signal to the length of the zero padding is then added to the overlapping zero padding of the previous segment. Due to the extension, the length of the zero padding post-convolution only contains a portion of the final convolution result. However, on the addition

of the overlapping portion of the next segment with the length of padded data, the complete result can be obtained as if the process was carried out continuously. The OLS algorithm *(shown in Fig 2.4)* alternatively does not utilize zero-padding, but utilizes an overlapping portion from the previous segment. Segments are padded to the nearest power of 2 in order to ensure optimal FFT operation similarly to the OLA algorithm. On processing, the starting segment that was taken to be the overlap is instead discarded, since the previous segment contains the convoluted result of the overlap already.

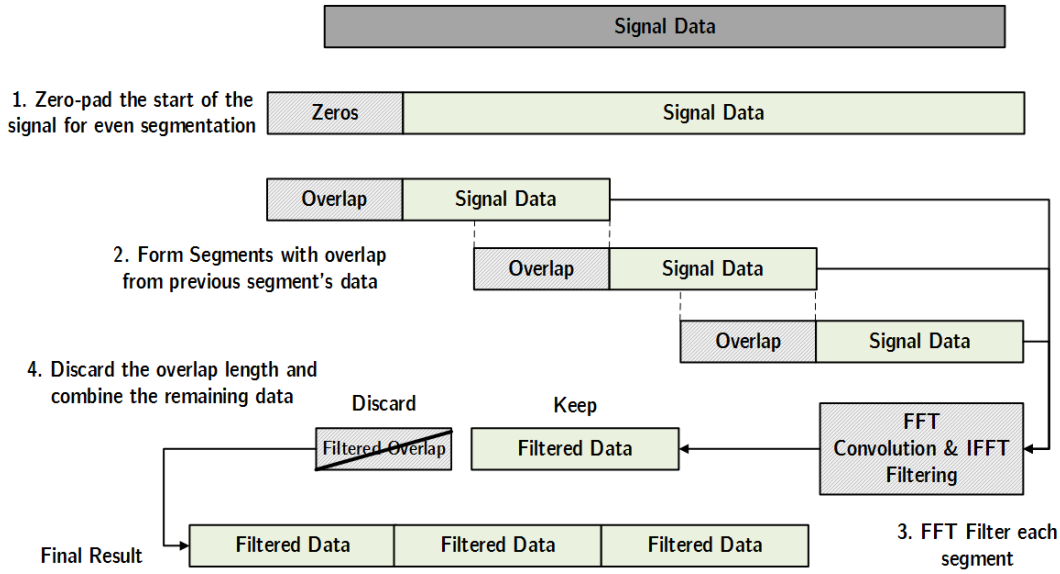


Figure 2.4 *The Overlap-Save Algorithm Process.*

Adding or keeping these overlapping portions would result in errors when the segments are concatenated. Owing to this, the OLS algorithm is considered cheaper than the OLA algorithm since it does not require the addition operations. Theoretically the collection of data samples does not have to be long, but there is a key requirement of having enough data gathered, in order to fulfil the minimum number of samples needed for the FIR filter to properly function. For the usage within real-time applications, incoming data can be collected continuously and stored in a buffer queue. When enough data is available, blocks can be formed

using previously collected data and processed as seen fit. As required with the OLA algorithm, the length of each segment will vary depending on the limitations of the system's hardware and the filter complexity. Smaller lengths of collection will allow for the system to be closer to being able to produce continuous real-time output, whereas larger lengths will result in longer delays for information output.

2.1.1.2. Infinite Impulse Response (IIR) Filters

In contrast to the FIR filter the IIR filter *(illustrated in Fig 2.5)* also known as a *"recursive filter"*, utilizes both past, current inputs and outputs in order to filter the signal. Past and present samples are parsed initially by the filter only to have their outputs reused within the filter to obtain the result desired. For this reason, IIR filters come with two sets of coefficients for forward filtering and backwards filtering.

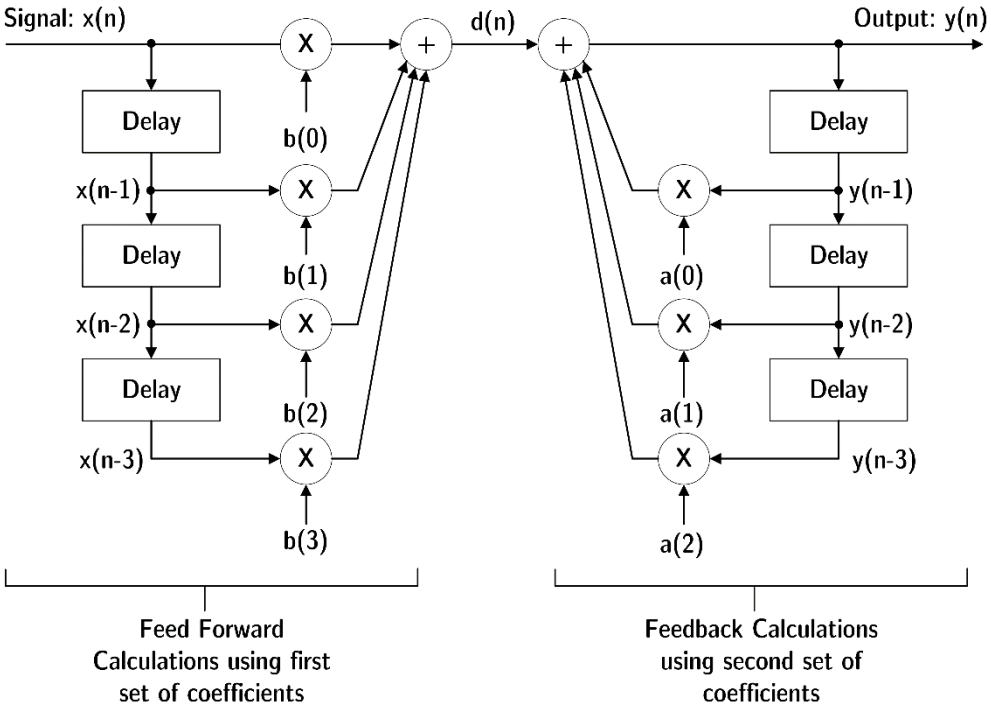


Figure 2.5 *Example of an IIR filter structure (referenced from Lyons 1997* ***[34]****).*

IIR filters output constantly regardless of input due, owing to the recursive operations performed by the filter with its own internal coefficients. Thanks to this structure they are able to obtain a much sharper frequency response desired. The number of filter coefficients required are also significantly less than the FIR filters making them much more computationally efficient. The main drawback is that the output of these filters may be unpredictable in some circumstances due to their own feedback mechanism, making these types of filters potentially unstable.

Due to this, IIR filters have the tendency to distort the phase or shape of the signal which gives it the property of *"non-linear phase",* making them unsuitable for applications that require precise analysis of the signal. A technique called *"Zero-phase"* filtering *(ZPF)* is, however, known to be effective for cancelling the effects of phase distortion. ZPF involves reversing the signal after an initial filtering, and then filtering the signal again in order to cancel phase distortions caused by the filtering process *[38]*.

This has been used widely *[39, 40]* for PPG signals by using a Butterworth design IIR filter with a cut-off frequency of 0.3-0.5Hz, corresponding to the end of respiratory frequencies and the start of HR frequencies. The drawback to ZPF however is that it cannot be performed in real-time, since it requires for all the data to be present for the forward-reverse filtering process. While ZPF can be applied to FIR filters, as well on a chunk to chunk basis for semi real-time processing, it cannot be applied to the IIR filter. With its innate infinite impulse response effects, the application of ZPF to the IIR filter will result significant distortions on combination of the chunks *[41]*.

2.1.2. Wavelet Filtering

In contrast to the Fourier transform, which only breaks down the component into the power-frequency representation, the wavelet transform breaks the signal down into the time-frequency representation *[42]*. Described as the following mathematical notion for the continuous variant:

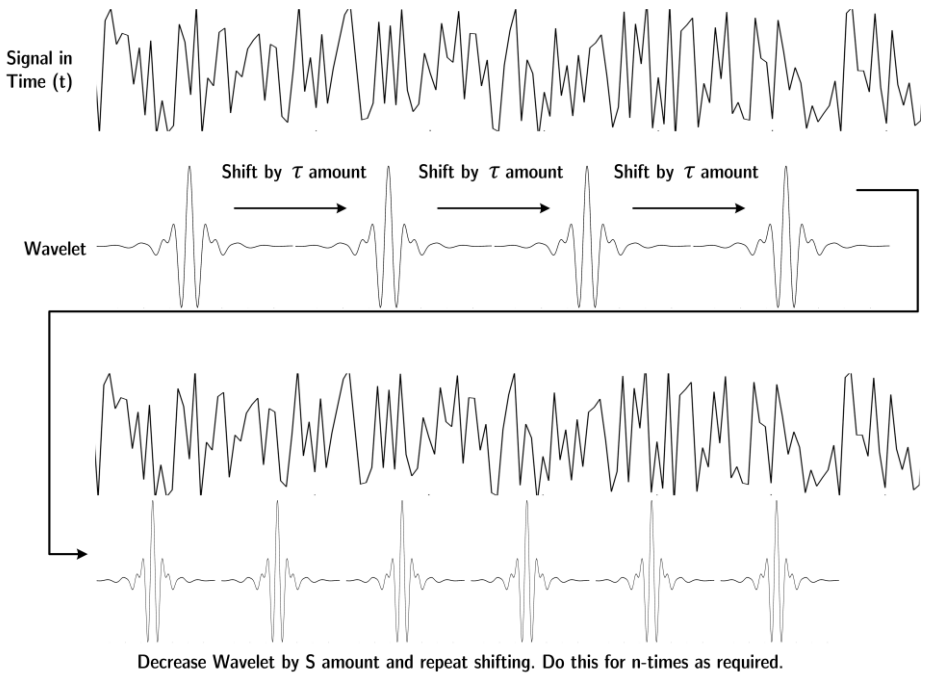
$$CWT(\tau, s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{\infty} f(t) h^* \left(\frac{t-\tau}{s}\right) dt \quad \textit{(Eq. 2.3)}$$

Where:

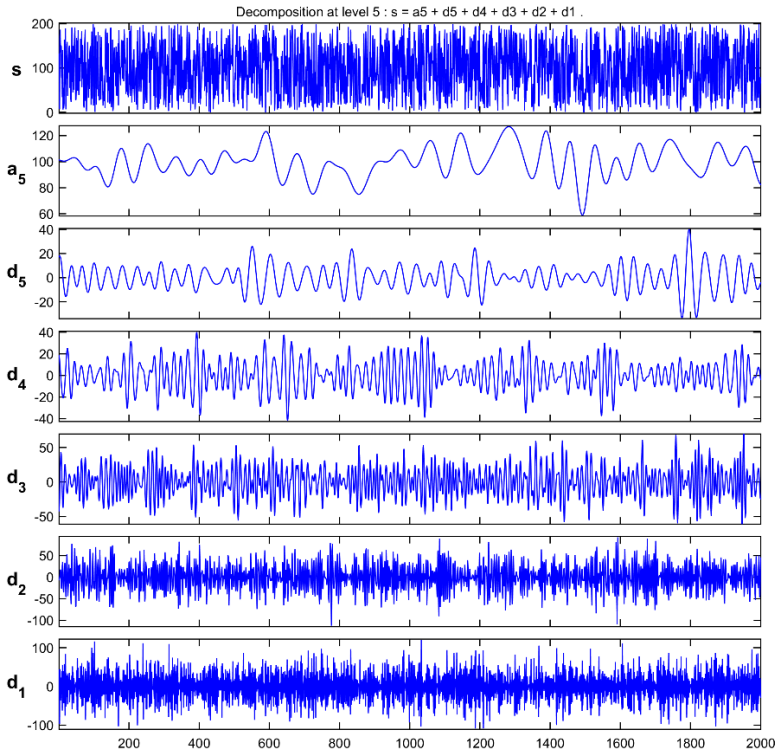
- S = Scaling parameter of the wavelet
- h^* = Smaller Wavelet function derived from Mother Wavelet function
- τ = Translation parameter
- T = Time
- f(t) = The function representing the signal

A wavelet can be described as a wave-like oscillation formed through a mathematical equation, and the term *"Mother Wavelet"* means the main wavelet that all smaller wavelets are derived from. The concept of the wavelet transform is based on selecting a suitable mother wavelet that has similar characteristics to the signal's own form.

In addition to form, linear phase and coefficient lengths are also considered for the preservation of signal morphology and computation time. Smaller wavelets are derived from the mother wavelet *(illustrated in Fig 2.6 a)* which has a finite length or scale of *S*. These smaller wavelets are shifted at a rate of τ through the signal's segment time span T, where signal elements within the window of these wavelets span are convolved by the wavelet's own internal elements. The resulting effect smoothens and amplifies the signal in certain areas.



(a). *Shifting the baby wavelet through the signal.*



(b). *Example of Wavelet Transform Output in MATLAB. From top to bottom shows the layers of frequencies that exist within the signal ranging from low to high resolutions.*

Figure 2.6 *The Wavelet Transform Process.*

This is done several times with the scale *S* being decreased each time. As a direct result, the signal undergoes a series of high-pass and low-pass filtering processes *(shown in Fig 2.6 b)* through the scaling of the derived wavelets. By allowing the signal to undergo sampling at different resolution scales, a frequency filter bank in essence is created. Each corresponding bank contains a range of frequencies correlating to that specific scale.

Lower scales of *S* result in a more general resolution of the signal which correspond to lower frequencies, whereas higher scales of *S* will result in a more detailed resolution of the signal which corresponds to higher frequencies. In theory, by decomposing the signal enough times using the appropriate wavelet with an appropriate scale of *S*, the trend or baseline existing at lower frequencies can also be estimated.

The Wavelet filter is not bounded to a fixed cut-off hence, it is able to approximate drift frequencies that vary with time, where it increases or decreases below the cut-off threshold. The general accepted scale for drift approximation within PPG signals is around 7-8 however, there are circumstances where drift frequencies exist outside of these scales alongside the HR frequencies. Due to this, there have been development and investigations into hybrid filters *[40, 43, 44]* that utilizes other techniques in order to remove drift the wavelet filter cannot target.

Wavelet filters have also been shown to be prone of picking up higher frequency elements when the drift has a smaller presence than the main HR frequencies *[43]*. The difficulty of using wavelet filters lies in selecting an appropriate level of decomposition and a suitable mother wavelet. Both of these aspects are highly dependent on the skill level of the user in order to be effective. But despite this, wavelet filters are still regarded as one of the more effective methods for the analysis of biological signals and usage as conditioning filters.

With its potential greatly outweighing its drawbacks, it has become a subject of popular study within recent years. To lessen the difficult of selecting parameters, *Dianguo et al. 2008 [45]* proposed an adaptive algorithm *(illustrated in Fig 2.7)* to estimate the baseline of a PPG signal using wavelet transforms.

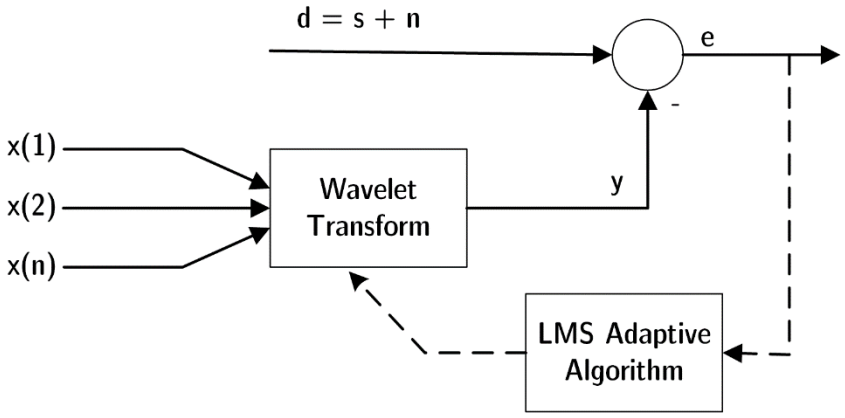


Figure 2.7 *The adaptive filter by Dianguo et al. 2008 [45].*

Where:

- d = the original signal composing of the information portion, s and the noise portion n (baseline wander)
- x(n) = Inputs of the high-frequency components from the Wavelet Transform
- y = Output of the wavelet transform for the nth time
- e = error of the resulting subtraction which is passed into the LMS adaptive algorithm

The original signal is decomposed by the wavelet transform using a *"D. Meyer"* wavelet for n-times to produce a series of high-frequency detail components. The filtered signal was then subtracted from the original signal to produce an error ratio e. This was passed into the Least Mean Square (LMS) adaptive algorithm, in order to adjust the weights for selecting a proper decomposition. On the selection of a correct decomposition level from the series of inputs, the original signal was subtracted by the high frequency component input. This left behind the low-frequency baseline component, which could then be processed as seen fit.

The authors conducted experiments using a generated faux pulse signal in which they contaminated with a generated baseline drift component. In application of their method, they were able to successfully estimate the baseline drift with a 99% correlation and a maximum RMSE of 0.0010 to that of the original. *Zhao et al. 2010 [32]* later utilized a similar approach, but with slight additional improvements to the algorithm proposed by *Dianguo et al. 2008 [45].* Their contribution involved the additional usage of *'Daubechies'* Wavelet filtering to remove high frequency noise *(algorithm outlined in Fig 2.8).*

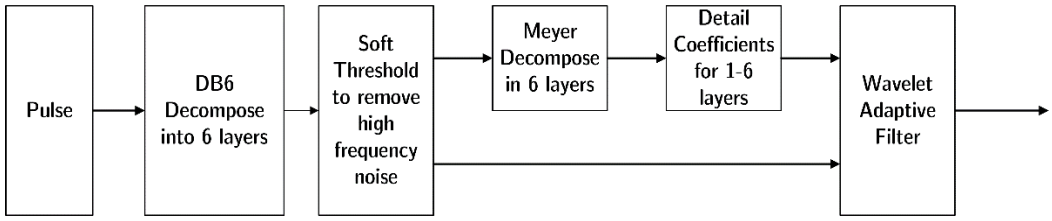


Figure 2.8 *Adaptive Wavelet Algorithm by Zhao et al. 2010 [32]*.

The authors' motivations behind using wavelet filtering, in response to traditional means of smoothing, was that smoothing caused some of the signal's essential features to be lost. Since wavelet transforms could decompose the signal into frequency components at the time of their occurrence, this could be used to remove the high frequency noise more accurately while leaving the remaining features of the signal intact.

Simulations results using the Adaptive Wavelet Algorithm have shown an improvement in signal-to-noise ratio on average in comparison to the direct Wavelet filter method. However, despite the effectiveness of this algorithm, it is not able to completely remove the baseline drift, due to outlier frequencies that exist outside of the scale's range. The Wavelet filtering process is often computationally intensive, owing due to the amount of calculations required for decomposition which limits its real-time applicability. While there exists optimized variants such as the Fast Wavelet Transform *(FWT)* [46], in comparison to directly removing the drift, this method would not be as efficient.

2.1.3. Cubic Spline Interpolation Filtering

The concept of Cubic Spline Interpolation *(CBSI)* is to connect a series of points through a polynomial equation generated line, which passes through all of these points continuously *[47]*. The concept of utilizing CBSI for filtering, is that through approximating the line that passes through each of the valleys within a signal, the drift can be completely approximated through the line generated by solving the equations based on these points *(illustrated in Fig 2.9)*.

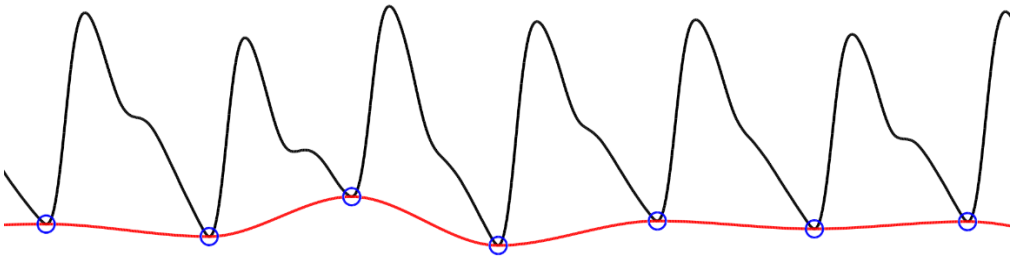


Figure 2.9 *Visual example of Cubic Spline Interpolation on a PPG signal. Valleys are shown in blue, where the spline generated from them that passes through each of them is shown in red.*

Lin et al. 2010 [30] introduced an algorithm for approximating the baseline drift via CBSI. The PPG signal would be filtered for powerline interference, but not for baseline drift, since the authors considered the usage of cut-off filters to have negative influences on the signal through over approximation. The smoothened signal would be passed through valley detection and storage.

Valleys are utilized in CBSI in order to approximate the drift between the two latest points that was detected. By keeping the number of points small enough and relevant to the latest data, real-time removal of the baseline drift can be achieved. Applying CBSI to a pulse signal contaminated with a simulated baseline drift, the authors were able to reconstruct the signal with approximately 1-2.27% error in comparison to the original signal.

In contrast, *Xu et al. 2007 [43]* earlier utilized CBSI as part of their cascaded filter design incorporating the Wavelet filter *(illustrated in Fig 2.10)*. In their studies, they identified the limitations of utilizing both techniques which was traced to the amount of drift complexity that the signal contained.

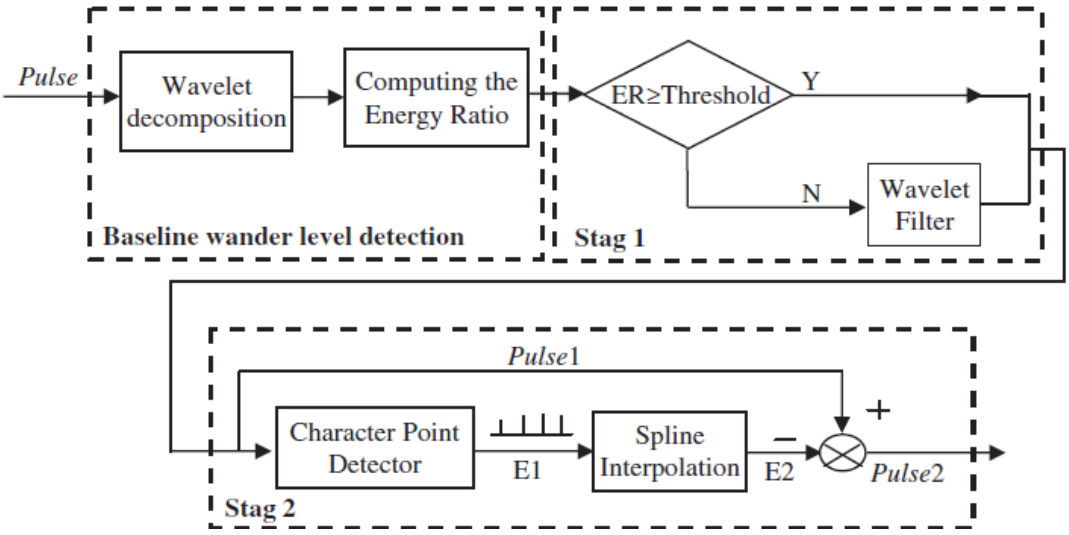


Figure 2.10 *Algorithm outline for Xu et al. 2007 [43].*

The authors utilized an Energy Ratio *(ER) (calculated using Eq. 2.4)* in order to approximate the ratio of drift to HR frequency content. The ER is calculated using the first level and the seventh level of detail from the Wavelet transform. The first level contains the high frequency information, whereas the seventh level contains the lower frequency information which correlates to the baseline drift.

$$ER = 20log_{10}\frac{||A1-A7-mean(A1-A7)||}{||A7-mean(A7)||} \qquad (Eq.\ 2.4)$$

Within equation 2.4, A1 and A7 are the respective details for the first and seventh layers of decomposition details. The drift complexity is correlated to the value of the ER; a signal containing high to large amounts of drift frequency presence would possess a smaller ER, whereas a signal containing little to moderate drift frequency presence would possess a high ER.

Their initial experiments showed that the Wavelet filter had performed relatively well, when exposed to signals with high levels of drift frequency present indicated by low ER levels. However, the filter overall performed rather poorly, when exposed to signals with little to no drift indicated by high ER levels *(indicated in Fig 2.11)*.

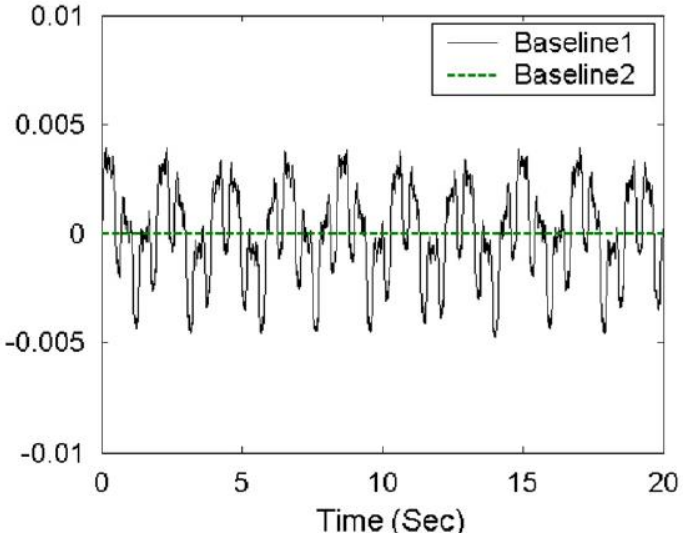


Figure 2.11 *Results of Wavelet approximation when ER is low based on little to non-existent drift. Baseline 1 is the wavelet approximation of the baseline at approximation level 7 and Baseline 2 is the true baseline.*

In their comparison of filtering methods, the authors found that CBSI was suitable for signals with little to moderate levels of drift frequencies and frequency presence. In contrast to the Wavelet filter, CBSI performed poorly when there was too much drift frequency presence due to the complexity *(indicated within Fig 2.12).* Based on these concepts a cascade filter was therefore proposed; wavelet decomposition was carried out where the approximation levels would then be used to calculate the ER. If the ER was found to be above 50dB, then feature detection was applied and CBSI would then be utilized to approximate and remove the remaining drift elements.

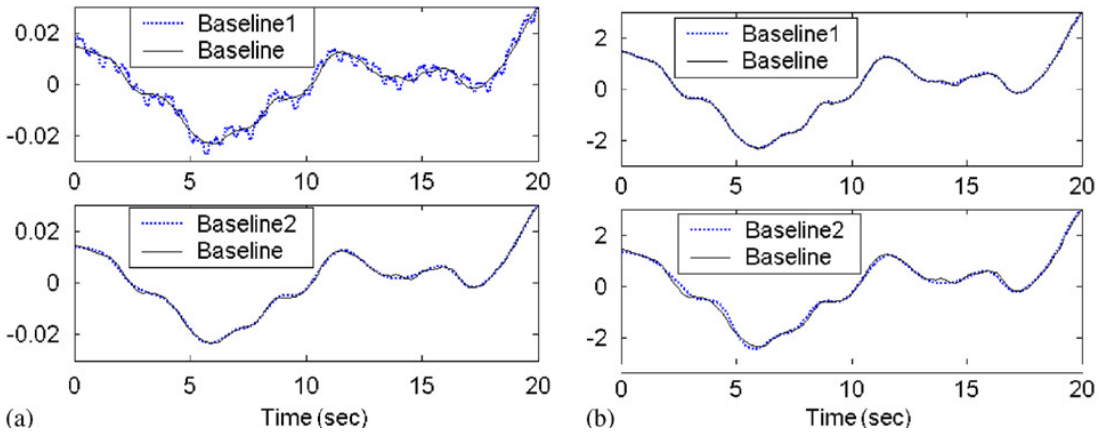
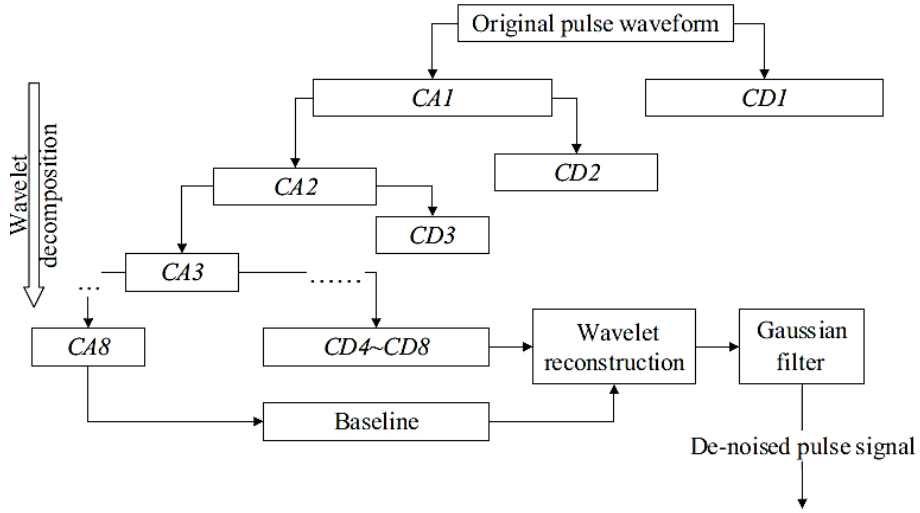


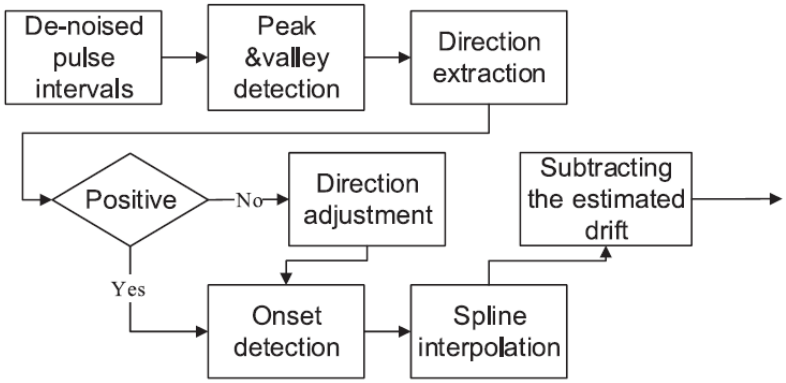
Figure 2.12 *Comparison tests of CBSI vs Wavelet Filter under different levels of drift presence extremities by Xu et al. 2007 [44].* ***(a)*** *Presents results where Wavelet Filtering (Top) and CBSI (Bottom) was applied to approximate a drift with low presence.* ***(b)*** *Represents their approximations on a drift with high presence in comparison to the HR component.*

Alternatively, if the ER was found to be below 50dB, then the seventh approximation would be subtracted from the signal to soften the drift by reducing the complexity. CBSI would then be applied to remove the remaining drift frequencies. The algorithm was noted by the authors to be able to successfully remove the entirety of the baseline drift. On analysis by Traditional Chinese Pulse Diagnosis experts, the visual diagnosis accuracy improved from 67% to 83% for the selected signals.

Another cascaded filter by *Wang et al. 2016 [44]* involving one of the original authors from *[43],* built on from the latter's work previously as a pre-processing method to segment the PW for medical analysis. A Wavelet filter utilizing the "Sym8 Wavelet" combined with a Gaussian filter *(illustrated in Fig 2.13 a)* was utilized to remove the high frequency noise and to soften the baseline drift similarly to *Zhao et al 2010 [35]*. CBSI was then later used to remove the remaining drift *(illustrated in Fig 2.13 b)*, and an interval selection process was utilized to divide the PW within the signal while identifying and ignoring segments with distortions.



(a). *The Wavelet filtering process to remove powerline noise and soften baseline drift.*

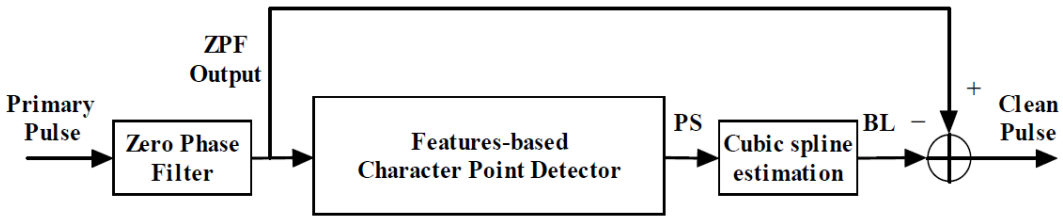


(b). *The detection process and the cubic spline interpolation process in order to remove the remaining drift post-wavelet filtering.*

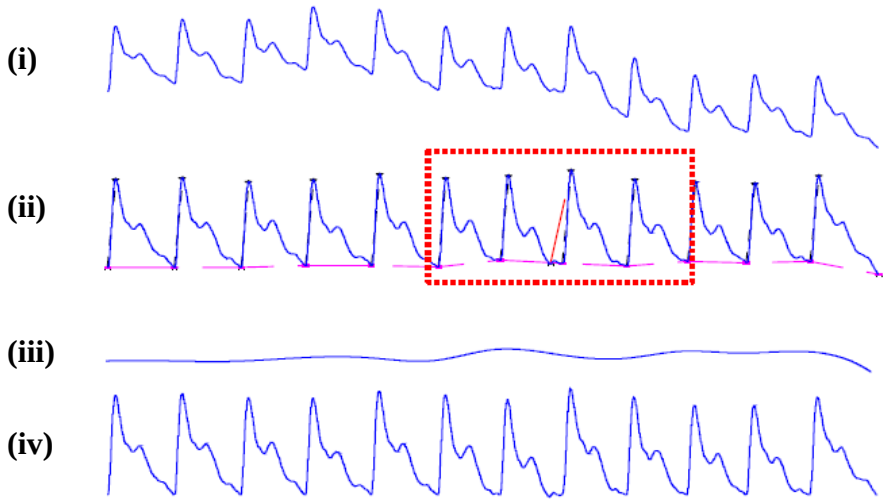
Figure 2.13 *The pulse de-noising process by Wang et al. 2016 [44].*

Their method was noted to have achieved better detail preservation through the usage of Wavelet de-noising for removing powerline noise in comparison to FIR filtering, Moving Average Filtering and Direct Wavelet Filtering. It was able to improve the diagnosis accuracy rate of potentially diabetic PWs in comparison to previously proposed methods, achieving an accuracy rate of around 82.3-85.9% up to 91.6%.

Kan et al. 2012 [40] presented a similar approach *(outlined in Fig 2.14)* based on the concepts presented by *Xu et al. 2007 [43].* Instead of utilizing the Wavelet filter which they deemed to be too computationally intensive, they opted to remove the frequencies of the baseline drift directly through employing IIR-ZPF. In addition, rather than setting the cut-off frequency slightly above 0.5Hz like most applications tend to do, the cut-off frequency was reduced to 0.3Hz instead in order to ensure no distortions from filtering.



(a). *The filtering steps in order*



(b). *The results of various filtering steps. Within the figure:* ***(i)*** *is the raw signal,* ***(ii)*** *is the Zero-phase filtered signal undergoing detection and CBS Interpolation,* ***(iii)*** *is the resulting baseline from the CBS interpolation process and* ***(iv)*** *is the resulting signal from the subtraction of the CBS approximated baseline from the Zero-phase filtered signal.*

Figure 2.14 *Outline of Kan et al. 2012's [40] algorithm.*

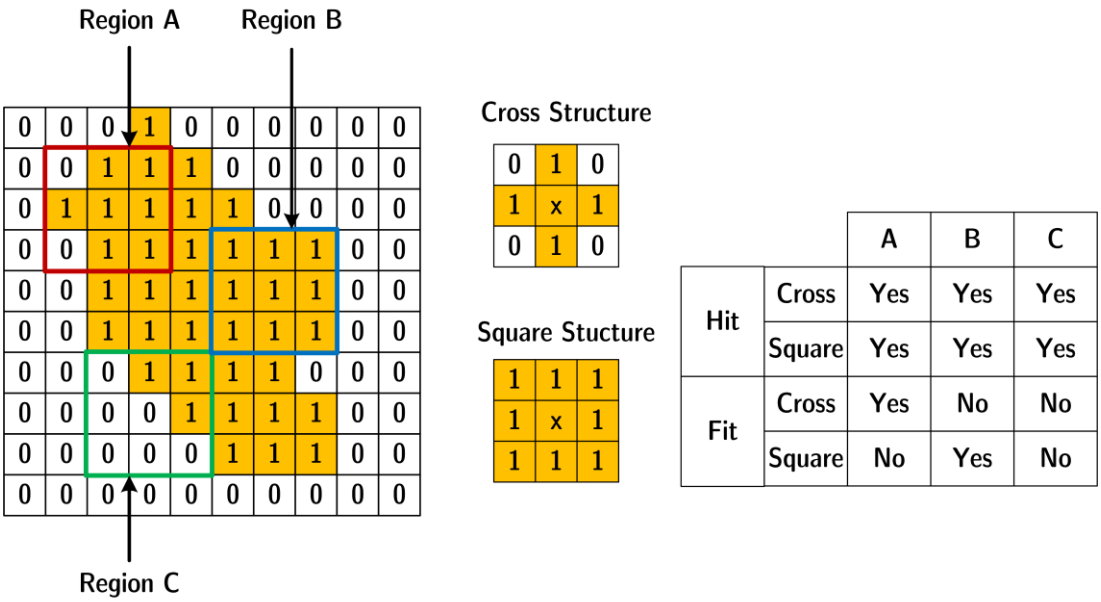
Post ZPF, the authors employed feature detection and CBSI in order to remove the remainder of the baseline drift as *Xu et al. 2007 [48]* had done. Within a test using simulated baseline wander added to synthesized signals, the authors found that their method was able to achieve a correlation of 99% on reconstruction. In terms of timing, it overall achieved significantly better computation run times of 0.0027 seconds, in comparison to 0.0129 seconds, which was achieved by a wavelet filter and the 0.3153 seconds by an EMD based method.

The reduction of computation times can be attributed to directness of ZPF, and its structure requiring significantly less values for its kernel. This meant that less calculations are required for every pass, resulting in faster computation times. In contrast, Wavelet filters require an ever growing filter kernel, and EMD filters require significantly large amounts of calculations per pass to approximate the different levels of decompositions required. The amount of calculations required per pass, as well as the multiple cycles required make Wavelet and EMD filters highly costly in terms of computation times.

2.1.4. Morphological Operators

Morphological operators are commonly utilized in the field of computer graphics and image processing in order to eliminate, isolate and clarify features within images *[49]*. The operators revolve around the principle of binary matching using an image that has been converted into a black and white binary representation, and a structure element selected as a shape such as a circle, square, line etc. The structure itself is defined by 1s and 0s in a pattern to resemble the associated shape with a point of origin *(shown in Fig 2.15)*.

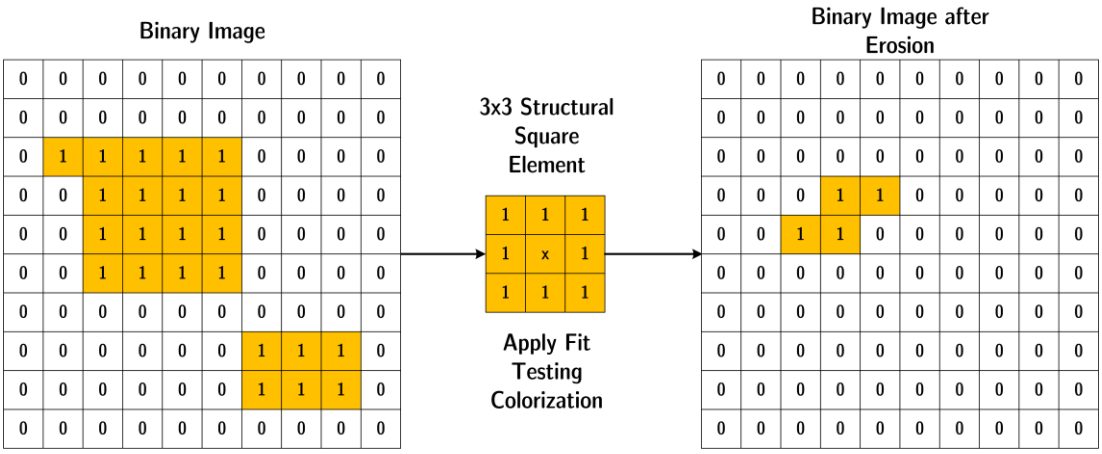
For operation, the structuring element is compared and matched to the image. If the origin of the structure *(marked by x in Fig 2.15 and 2.16)* falls onto a value of 1 then it can be said that a hit has occurred otherwise it is a miss. If all values of the structure element land on a value of 1 within the binary image, then it is said that a fit has occurred.



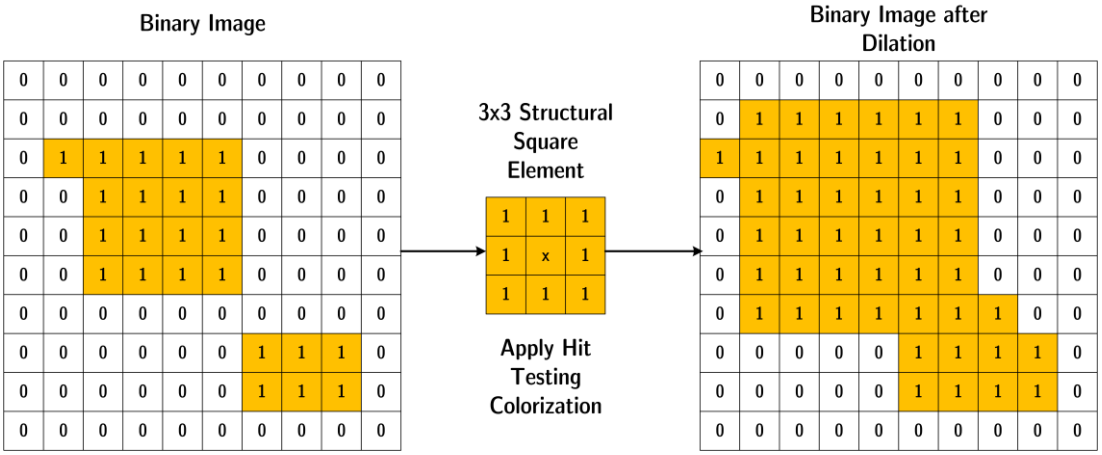
		A	B	C
Hit	Cross	Yes	Yes	Yes
	Square	Yes	Yes	Yes
Fit	Cross	Yes	No	No
	Square	No	Yes	No

Figure 2.15 *Outline of the morphological matching process.*

Two basic operators are used to build compound operators; these are the erosion operator denoted by $\ominus$, and the dilation operator denoted by $\oplus$. The erosion *"shrink"* operator replaces all values on a fit of the structure with 1 and 0 otherwise. This in essence removes areas of the image that falls outside of the fit-test *(illustrated in Fig 2.16 a)*. The dilation *"grow"* operator works in a similar manner *(illustrated in Fig 2.16 b)* but instead of replacing the values with 0, these areas are replaced with 1s in essence expanding the image element.



(a). *The effect of an erosion operator using a 3x3 square structure.*



(b). *The effect of a dilation operator using a 3x3 square structure.*

Figure 2.16 *Examples of erosion and dilation operators*

These two operators form the basis for isolation, removal and expansion of structural elements within images when combined together to form the opening and closing operators *(defined by Equations 2.5 and 2.6).*

$$Opening\ f \circ s = (f \ominus s) \oplus s \qquad (Eq.\ 2.5)$$

$$Closing\ f \bullet s = (f \oplus s) \ominus s \qquad (Eq.\ 2.6)$$

Where *f* is the feature and *s* is the selected structure, opening operators are used to isolate and enhance finer features such as shapes within an image through

removing unnecessary links and then growing the remaining elements *(shown in Fig 2.17)*. Closing operators alternatively are utilized to enhance features, and then remove the unwanted excess after the enhancement of coarser details such as backgrounds and outlines.

(a).

(b). **(c).**

Figure 2.17 *Effects of opening and closing operators.* ***(a).*** *Original Image.* ***(b)*** *Closing operator with 5x5 square structure for isolating outlines and backgrounds.* ***(c).*** *Opening operator with a 5x5 square structure for isolating inner details. Image is a standard computer graphics test image taken from [50].*

Zheng-Gang et al. 2010 [51] utilized modified variants of the morphology operators within their studies in order to remove the baseline drift. An opening operator was utilized to eliminate the peak structures within the signal, whereas a closing operator was utilized on the remaining elements in order to approximate the baseline drift, by connecting the valleys through a generated bridge. While no clear visual results of their filter was given, their method was reported to yield higher correlation values compared to a Wavelet and EMD filter. An 89% similarity was found when applied in simulations to a clean signal.

In their study *Dae-Gun et al. 2014 [36]* utilized morphological operators in a similar fashion in order to filter and identify features within the PPG signal. Their method was noted to be relatively effective in comparison to existing methods, but the visual results *(shown in Fig 2.18)* provided by the authors indicates distortions on subtraction of the baseline drift approximation. This was due to its roughness making it unideal for the purposes of preserving the entirety of the signal details.

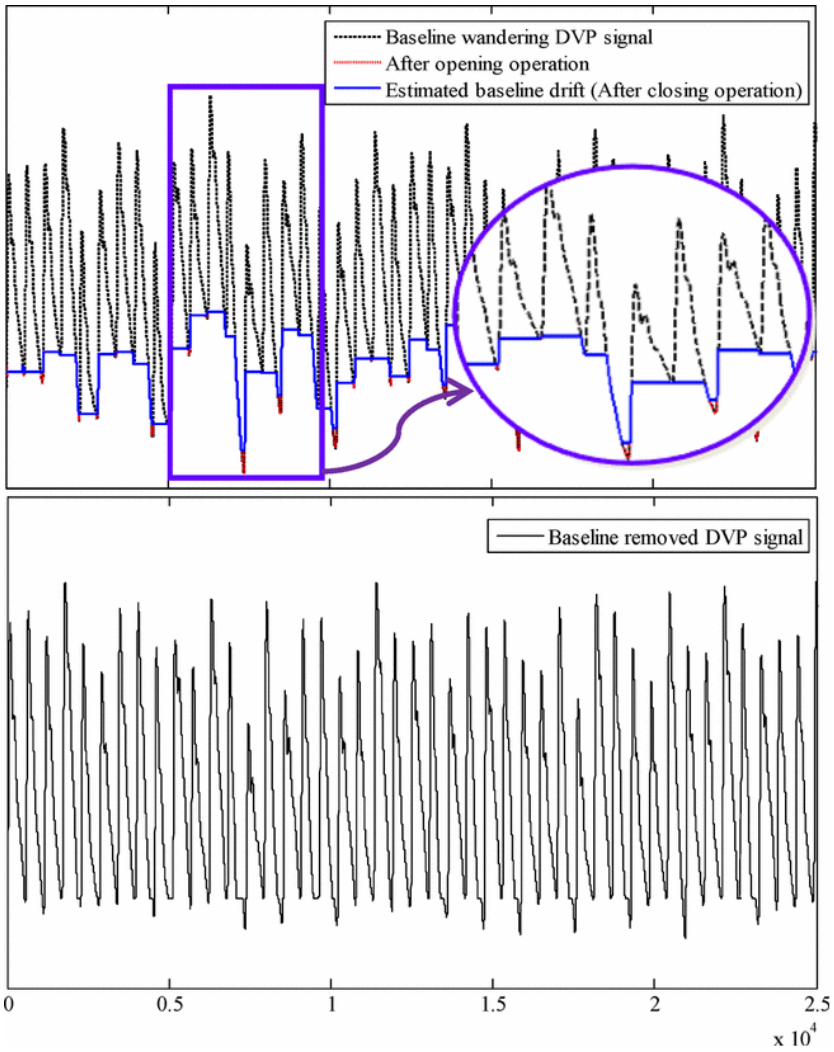


Figure 2.18 *DVP Algorithm Baseline drift approximation by Dae-Gun et al. 2014 [36] using modified morphology operators showing block artefacts on subtraction.*

2.1.5. Empirical Mode Decomposition

Empirical Mode Decomposition *(EMD)* is a subset transform of the Hilbert-Huang Transform *[52],* which is used for the decomposition of signals and time series into detail components. The process is similar to the Wavelet transform but unlike the former, EMD requires no selection of a mother wavelet. It instead relies on the extrema points of the signal to approximate sinusoids that contain frequency components, and a defined stopping condition for the decomposition process.

The basis of EMD revolves around producing oscillating waves known as *"Intrinsic Mode Functions" (IMF)*, which possesses two unique properties:

1. The number of extrema composed by the sum of all minima and maxima within the wave must be equal or differ at the very most by one extrema point.

2. The mean of all values within the IMF must be equal to zero.

IMFs are produced through a process known as *"sifting" (illustrated in Fig 2.19),* where all maxima and minima within a signal are detected based on level thresholds. Through the detection of the extrema points envelopes are created using cubic spline interpolation, resulting in a lower envelope and an upper envelope. These envelopes are combined and averaged in order to produce the IMF which corresponds to a level of frequency range within the signal.

Like the detail approximations of the Wavelet filter, the resulting IMFs of the EMD process corresponds to a specific range of frequency components within the signal. The IMFs are then used as a point for further decomposition until a stopping criteria is met, or until there remains only two or less extrema points left from the resulting decomposition.

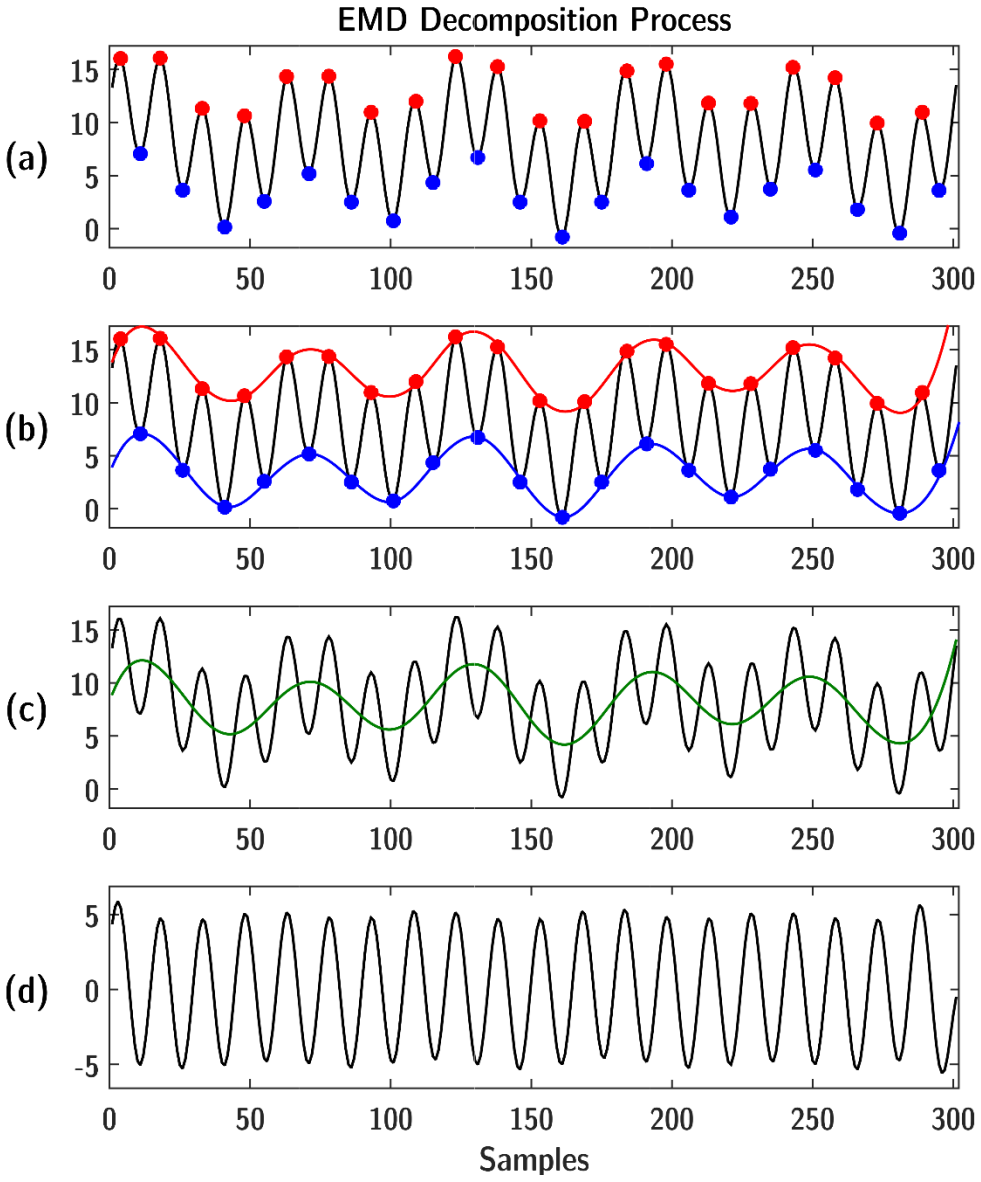


Figure 2.19 *Example of the EMD Decomposition process for baseline removal. In* ***(a)*** *Minima and Maxima points are identified within the signal where these points are used as Cubic Spline Interpolation in* ***(b)****, to produce IMF layers in* ***(c)****. The resulting IMF is the subtracted in order to remove the baseline drift within the signal in* ***(d)****.*

The problem with EMD similarly to wavelet decomposition is determining the optimal IMFs to remove from the signal. The decomposition process is not perfect, and may result in higher frequencies being included in the lower IMF levels. While the lowest IMF can be subtracted to remove the baseline, the results would not be optimal since there would be remaining elements of drifts at higher frequencies or amplitudes.

Vasco et al. 2008 [53] in their study utilized the EMD method in order to remove both the high-frequency powerline noise, as well as the lower-frequency drift within the ECG signal. Statistical tests were utilized to determine how many lower levels of IMFs should be removed, in order to eliminate the high frequency noise. A multiband filtering scheme based on using the last few remaining IMFs was proposed, where the frequency associated to the baseline wander is extracted from each IMF using a series of low-pass filters to reconstruct the baseline wander. By filtering each IMF level at a cut-off frequency related to the baseline wander, the process filtered out the higher frequency information of the ECG signal.

The parameters for the multi-band filtering scheme was recommended to be selected based on the behaviour of the drift through experimental trials. In application to a real ECG signal with comparisons to other methods, including the Butterworth IIR filter and the traditional Wavelet filter, the algorithm was found to be able to outperform the latter, achieving the highest SNR on reconstruction of a simulated noisy signal.

Zheng et al. [54] in their study utilized EMD and independent component analysis *(ICA)* of the IMFs from decomposition, in order to isolate and eliminate high frequency noises as well as the baseline drift. The 7th IMF was found to be correlated to respiration rates whereas IMF levels of 2 to 6 was found to have information in relation to the HR.

Using ICA to mix and combine the IMFs for analysis, they were able to reconstruct the signal with 89% similarity to the original signal's shape in contrast to using plain EMD filtering. EMD similarly to wavelet transform, has been found to be successful, but at the same time also can be computationally intensive, owing due to the number of decompositions required to obtain the baseline wander *[55].*

2.2. Techniques for Feature Identification

This section discusses previous techniques that have been previously proposed for identifying features within the PPG signal.

2.2.1. Benchmarking Standards

The assessment of detection algorithms utilizes the AAMI standards *[56],* for assessing peak and feature detection within cardiovascular measurement instruments. Two performance measures are described within the standards; the Sensitivity rate $(Se, calculated\ as\ Eq.2.7)$ and the Accuracy rate $(+P, calculated\ as\ Eq.2.8)$. The Sensitivity rate describes an algorithm's ability to adapt to various circumstances, and the Accuracy rate describes an algorithm's ability to detect real features over distortions and fakes.

$$Sensitivity\ (Se) = \frac{TP}{TP+FN} \times 100 \qquad (Eq.\ 2.7)$$

$$True\ Positives\ (+P) = \frac{TP}{TP+FP} \times 100 \qquad (Eq.\ 2.8)$$

Where:

- TP = True positives, detected peaks/valleys verified against the visual annotated results. This defines the accuracy rating of the algorithm.
- FN = False negatives, the number of skipped false peaks/valleys by the algorithm verified against the visual annotated results
- TP = True positives, the detected peaks/valleys verified against the visual annotated results.
- FP = False positives, the detection of the false peaks/valleys against the visual annotated results.

2.2.2. Techniques Involving Derivative Calculations

It is known that taking the first derivative of a signal *(as illustrated in Fig 2.20)* results in calculating the change rate between every single point within the signal *[57]*. Providing that the signal is relatively smooth, by identifying the highest peak within the first derivative of the signal, the point of the greatest change between a peak and valley can be identified. This then can be utilized as a pivot to identify peaks and valleys.

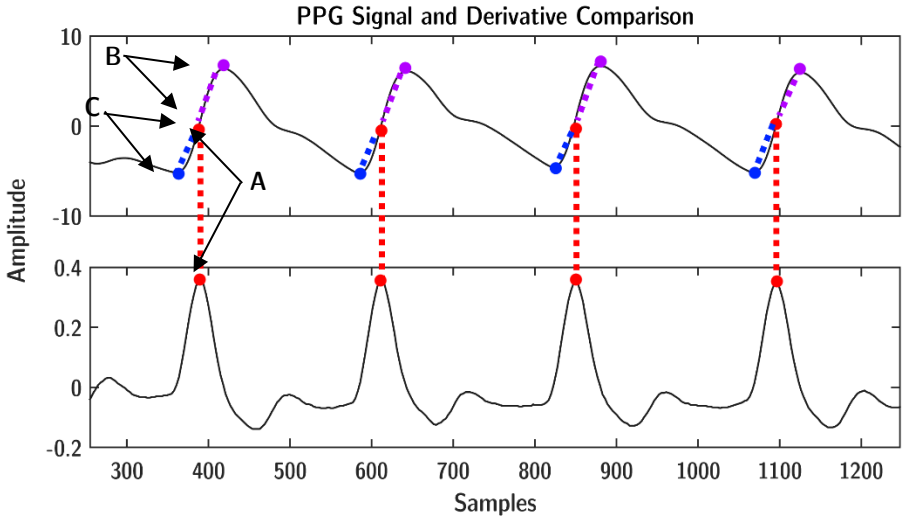


Figure 2.20 *Comparing PPG Signal and Derivative Peak locations (A) as well as search back processes to identify the peaks (B) and valleys (C). Signal sample was taken from file 0028_8min of the Capnobase dataset by Karlen, 2010 [10].*

Normally the identified pivot points would directly correlate to the mid-point, however, in circumstances of powerline noise, the largest derivative peak *(shown in Fig 2.21)* may correlate to mid-points between noisy peaks and valleys. This in turn makes it difficult to properly trace the middle point that exists between the real peak and valley to which, is considered crucial for acting as a pivot in order to identify and verify the latter. Although smoothing is an option to remove the spikes, the zero-crossing method is also not favoured under the circumstances of noisy signals where all detail regardless of noise is required, as it may remove and distort characteristics within the signal which are desired for analysis *[58]*.

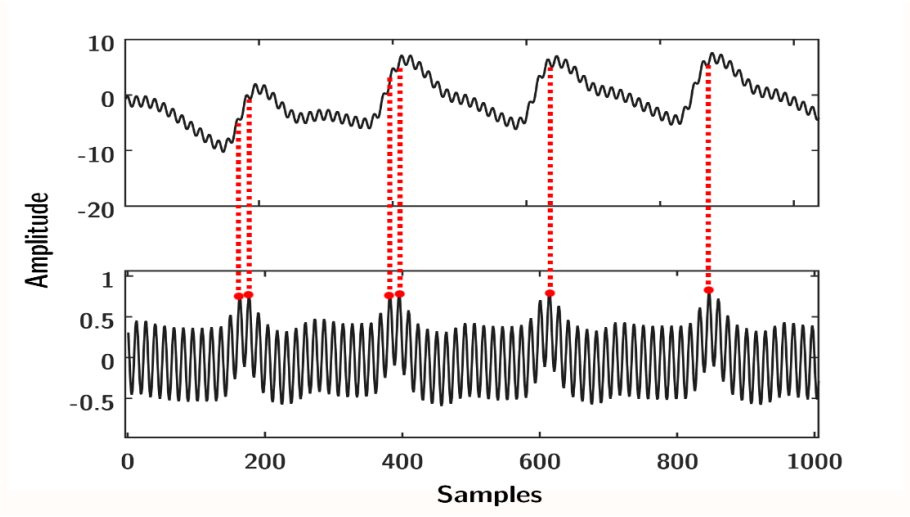


Figure 2.21 *Derivative of a PPG signal contaminated with high frequency noise and the attempt to identify the pivot point for the real peak and valley.*

Fortunately, the content of interest within the PPG signal is mainly the smoothened shape outline for visual and efficient computer analysis. Hence, the signal is typically filtered for high frequency powerline noise through the usage of moving average filters or notch filters. This makes the technique viable since the resulting signal is relatively smooth, but smoothing should also be kept within moderation in order to preserve the finer PPG details.

Xu et al. 2007 [43] utilized a modified variant of the derivative method *(illustrated in Fig 2.22)* where smoothing was kept to a minimum in order to ensure signal detail. The first derivative was taken with any points below zero being eliminated. This was done in order to remove minor distortions and any errors that may have occurred due to motion and jittering within the signal. Taking the first six largest peaks within the derivative an amplitude threshold was established. Any future peak value that fell outside of this threshold was considered noise, and was therefore excluded from the detection process. From the identification of the derivative peaks, a search-back process was performed to identify the starting valley point of the PW.

The same concept could theoretically be extended easily for the detection of peaks as well. No means were provided to discriminate based on distance hence, there may be problems where the algorithm encounters derivative peaks that resemble real features, but exist at a time-distance that is not considered normal.

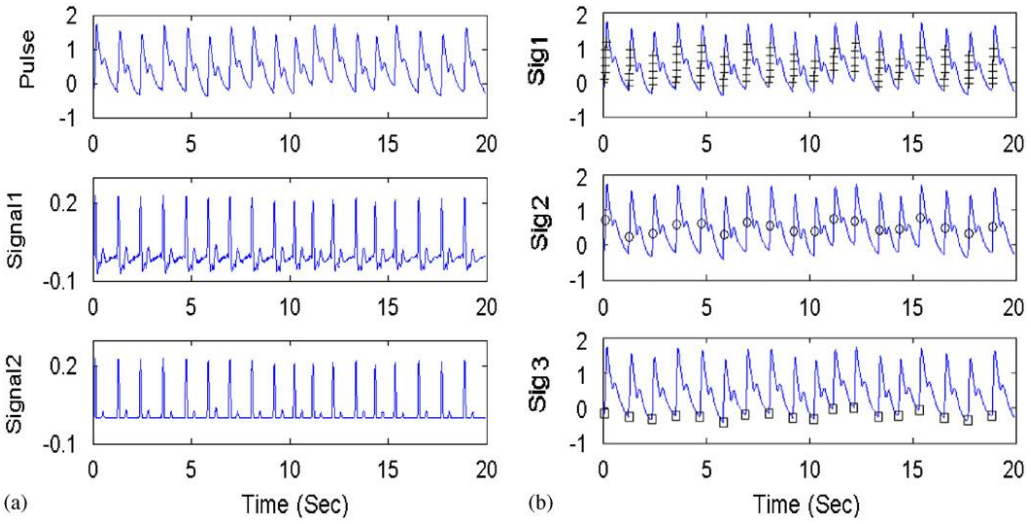


Figure 2.22 *The algorithm steps for Xu et al. 2007's algorithm [43]. Signal1 is the first derivative, Signal2 is the result of values below 0 removed, Sig1 values from the derivative that are larger than the threshold, Sig2 represents the central point calculated from the first derivative peak, and Sig3 represents the valleys found through search-back.*

Chen et al. 2011 [59] later presented a similar algorithm in their study of a cascaded filter that utilized a derivative-based method. Similarly, their algorithm entailed removing all values that were below zero within the derivative. This in turn removed values associated with jitter effects that can cause misdetection. The algorithm was noted to achieve better performance than the traditional zero-cross method, but the authors provided no means to identify and discriminate features caused by distortion, which may exist above zero alongside with the real-features at the same amplitude and size.

Chunming et al. 2008 [39] presented an approach that involved amplitude checking and time verification of systolic peaks. A signal was scanned until a negative derivative change occurred which indicated the existence of a systolic peak. From this point the position of the peak occurrence was verified using a time discrimination threshold. Any peaks that occurred with a higher succeeding peak, but possessed a lower time-distance difference than its neighbours, was subsequently replaced with the higher peak. This overcame the problem with sudden jitters that may exist in the signal *(illustrated in Fig 2.23)* which had the potential to throw off the segmentation process.

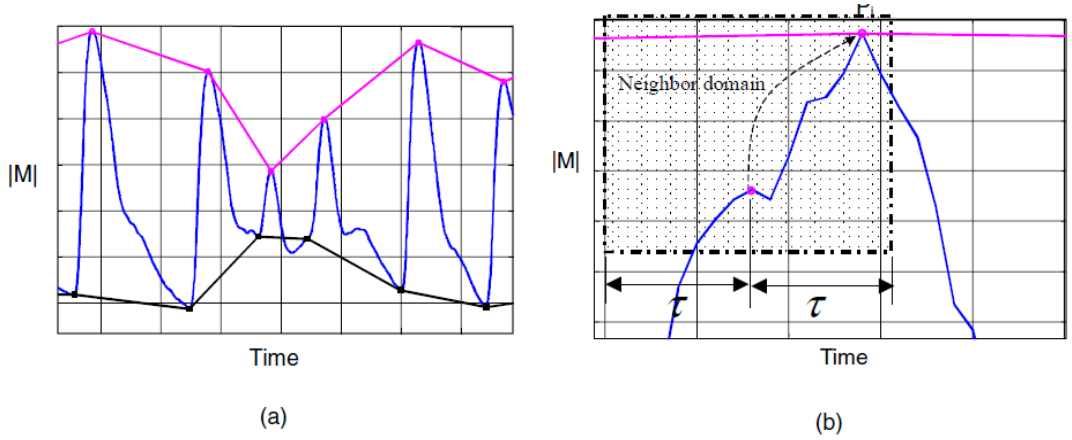


Figure 2.23 *The result of the segmentation algorithm by Chunming et al. 2008's algorithm [39] in (a) and its ability to overcome jitter effects in (b).*

On the identification of the peak, a back-wards search process was initialized in order to identify the corresponding valley to the peak. Once the algorithm finds a valley, an extended search was conducted by searching backwards and calculating the derivative of the points in the search. If a derivative was identified to be at least 3% or less of the maximum derivative that was identified during the climbing process, the valley would be replaced with that point. The amplitude threshold was specified to be calculated from the max and the min values within a window that was defined by two presumably valid points. The authors, however, provided no means on how to identify where the origin of these points.

A potential problem with using the maximum and the minimum values within a window, is that these values may correspond to sudden jolts out of the norm. These values have no relation to actual peak-to-valley pair slopes, and can lead to the wrong discrimination threshold being established. Hence, by utilizing the wrong thresholds, valid features may be accidentally excluded from the detection process.

Karlen et al. 2012 [60] presented a similar segmentation algorithm to that of Chunming et al. 2008 [39], for separating PW that included a basis for identifying features and avoiding distortions. Based on the *"Incremental Search and Merge"* algorithm, lines were formed using the signal points *(shown in Fig 2.24).* Lines with similar ascending slopes were merged together to form a segment. Segments on formation have their amplitude calculated and were compared against a size window, formed by a lower boundary and an upper boundary which were both modified based on the algorithm's detection results.

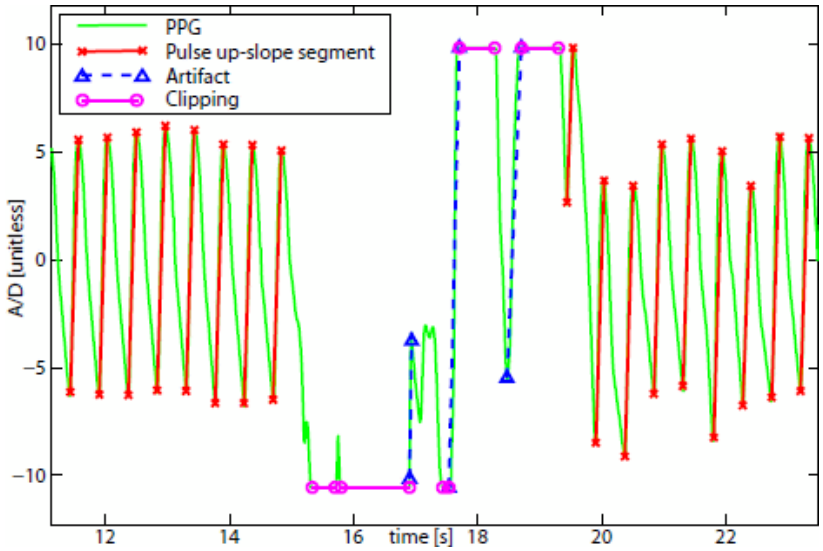


Figure 2.24 *The Adaptive Segmentation process by Karlen et al. 2012 [60]. Shown in red are the segments formed from lines with similar rising slopes whereas artefacts shown in blue and disconnects shown in purple are discarded or ignored*

If the algorithm deemed that the segment was within the boundaries, the boundary parameters were adjusted to be tighter and closer to the detected segment amplitude. Under circumstances where the algorithm considers the segment to be out of bounds, the boundary parameters were adjusted in order to broaden the range as to adapt to the signal's trend through time.

The problem that *Chunming et al. 2008 [39]* encounters can be resolved as the algorithm can adapt itself to normality, since the amplitude thresholds are established and updated over time with newer slope trends. Since the discrimination parameters have a direct relation to actual peak-valley distances, the algorithm achieves not only potentially higher accuracy, but adaptability properties as well.

The final detection rates based on the AAMI standards as reported by the authors managed to achieve a moderately high rating of 93.01% accuracy rate and 99.81% sensitivity rating. The lowered accuracy rating was mostly due to the algorithm encountering indistinguishable artefacts that were comparable to real features in terms of amplitude.

2.2.3. Techniques Involving Spectral Information

As covered earlier, all signals or time series can be represented by a sum of infinite sinusoids each oscillating at different frequencies. The distance of each sinusoid cycle *(shown in Fig 2.25)* is determined by the frequency; higher frequencies will have cycles closer to one another reducing the distance between them, and lower frequencies will have cycles separated further from each other thus increasing the distance between them.

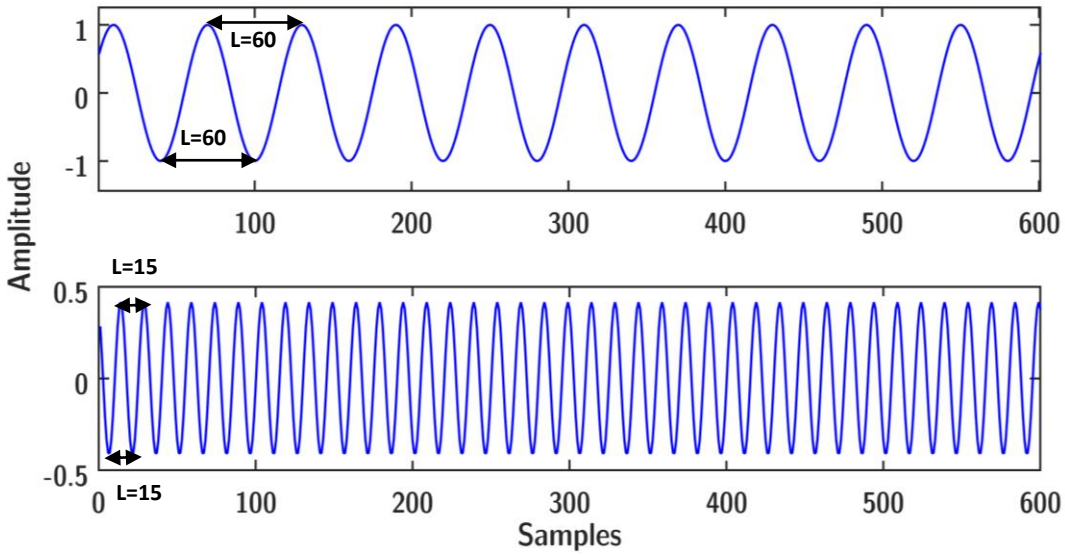


Figure 2.25 *Difference in distances between a 5Hz sinusoid wave (top) and a 20Hz sinusoid wave (bottom), where L is the distance between the inflection points in points.*

The concept behind using spectral information in order to detect features, revolves around obtaining the fundamental frequency associated with the systolic peak oscillations. From this, the periodicity or the time length of a single cycle of a sinusoid can be calculated *(via Eq. 2.9)*.

$$Periodicity = \frac{1}{systolic\ peak\ frequency} \qquad (Eq.\ 2.9)$$

Through calculating the periodicity of a sinusoid, a distance threshold that separates each cycle can be established. Based on this threshold if the algorithm encounters any features that fall over or under this distance, it can be considered false or noise. *Aboy et al. 2005 [61]* presented a systolic peak detection algorithm based on the decomposition of the signal using filter banks *(process is illustrated in Fig 2.26).* Each filter bank contained contents corresponding to a range of frequencies which in turn corresponds to individual characteristics within the PPG signal such as the dichroitic peak, systolic peak and the drift.

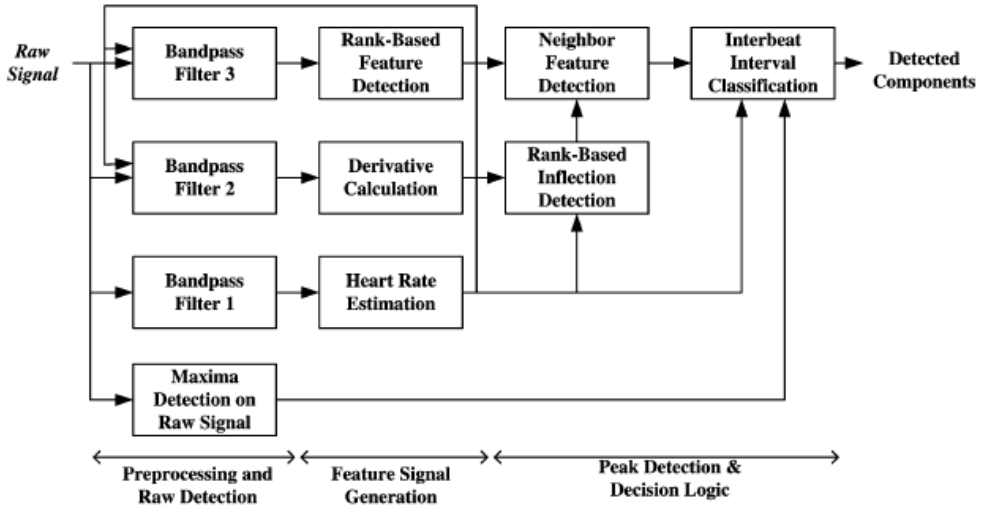


Figure 2.26 *Outline of the algorithm presented by Aboy et al. 2005 [61]*

By decomposing the signal into filter banks, the fundamental frequency of the systolic peak frequency was obtained. This was used to calculate the estimated HR periodicity from which a distance discriminator was generated. Peaks within the signal were then checked against the generated thresholds through using a series of logical comparisons in order to verify their authenticity. Their method is noted to achieve a sensitivity rate of 99.36% and a predictively of 98.43%, however, due to the nature of the decomposition process their algorithm was noted to be fairly computationally intensive, and was not suitable for real-time application.

Liangyou et al. 2009 [62] utilized a smoothing method based on the frequency of the HR to establish a basis in order to identify valid peaks. The basis for using the HR frequency was to obtain an optimal smoothing line which eliminated or soften the majority of the remaining features, while preserving the sinusoidal series corresponding the systolic peaks. From this, a number of peaks that have amplitudes above the smoothened signal were taken based on a certain threshold. The selected peaks were arranged from largest to smallest, where all peaks that exceed at least 2/3rds of the middle value were taken as valid peaks.

The median absolute deviation score (MAD) was then calculated for these peaks' distance, which are then utilized to identify and remove false or missing peaks that may have been included in the previous process. In the final stages a search-back process is utilized in order to identify the starting valley points of the peaks. The algorithm was noted to perform well with clean signals *(indicated within Fig 2.27 a),* but performed poorly in circumstances where there was moderate to severe noise and distortions *(indicated in Fig 2.27 b).*

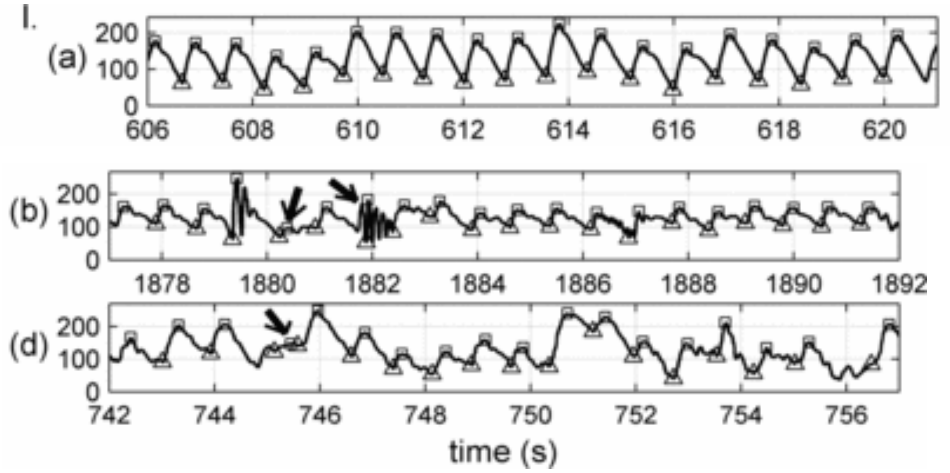


Figure 2.27 *Liangyou et al. [62]'s algorithm results* ***(a).*** *Excerpt from the test results of the algorithm showing detection results on a clean signal.* ***(b). & (d).*** *Excerpt from the test results of the algorithm showing detection results on distortion faults. Plot (b) indicates errors due to disconnect oscillations whereas plot (d) indicates motion artefacts.*

Although the algorithm utilized time-distance based discrimination, it provided no way to verify whether the features of the algorithms was fake or real based on their characteristics. Should a fake feature be detected within the time-span of the real feature, while appearing closer to the designated time of occurrence approximated, it would most likely be considered valid. Algorithms that utilize similar schemes, but do not provide a secondary means to re-affirm the validity of features these would potentially also suffer from the same problems.

Alternatively *Kan et al. 2012 [40]* as part of their cascade filtering solution, presented a similar technique where the signal was decomposed using the FFT post using IIR-ZPF. With the assumption that the majority of the baseline drift frequencies were removed as well as power-line noise frequencies, the remaining largest bin from the FFT transformation was said to correspond to the systolic peaks. The algorithm utilized both the slope of the peak-valley pairs, as well as time-distance discrimination in order to identify and avoid distortions within the signal.

For calibration purposes, a window was established using the periodicity of the systolic peak within the FFT result. The periodicity is multiplied by three for the length of the window to ensure that there was at least a minimum of three PW cycles for calibration. From this, a random segment was selected within the signal based on the defined size where the first peak was detected. Using this location peaks are further searched, based on the distance calculated from the frequency of the systolic peak in order to identify all valid peaks. On detection of the peaks, the algorithm searched back in order to identify the associated valley of the peak.

From the detection within the calibration window, the peak-to-valley slope was calculated for every peak. These were averaged to produce an amplitude discrimination threshold for the detection process, which utilized both the time distance between peaks and the peak-to-valley slope threshold in order to identify and discriminate features. On failure to detect a feature, the algorithm inserted a point at the location of where the feature should have been *(shown in Fig 2.28)* and proceeds to search onwards for the next occurrence.

Although the authors gave no indication of detection rates, the algorithm was noted to perform relatively successfully when applied to a series of simulated signals, but still possessed errors at times due to multiple valleys caused by jittering effects. This was however rectified through a verification search process based on the distance threshold.

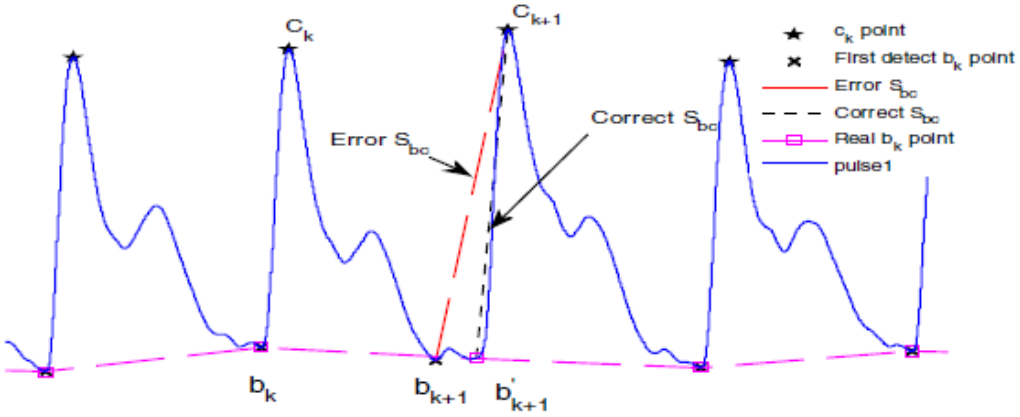


Figure 2.28 *Detection process by Kan et al. [40] showing recovery from error due to wrong slope discrimination.*

2.2.4. Techniques Involving Direct usage of Signal Characteristics

Alternatively instead of utilizing techniques such as spectral analysis or derivatives, the signal characteristics can be used directly in order to establish parameters for or guide the detection process itself. *Shin et al. 2009 [63]* developed an adaptive algorithm for the detection of both peaks and valleys, based on the variation of the signal's trend *(shown in Fig 2.29*) to overcome respiration effects.

By utilizing the trend of the signal movement and a slope that is calculated on only a small amount, the algorithm could potentially avoid sudden distortions and dichroitic notches. In the scenario that it does encounter such a fault, distance discrimination is utilized so that the feature will be included only if it exceeds the distance of the previous features to their respective neighbours. The algorithm however was noted to be negatively influenced by the effects of varying perfusion which causes differences within the peak amplitudes causing misdetection.

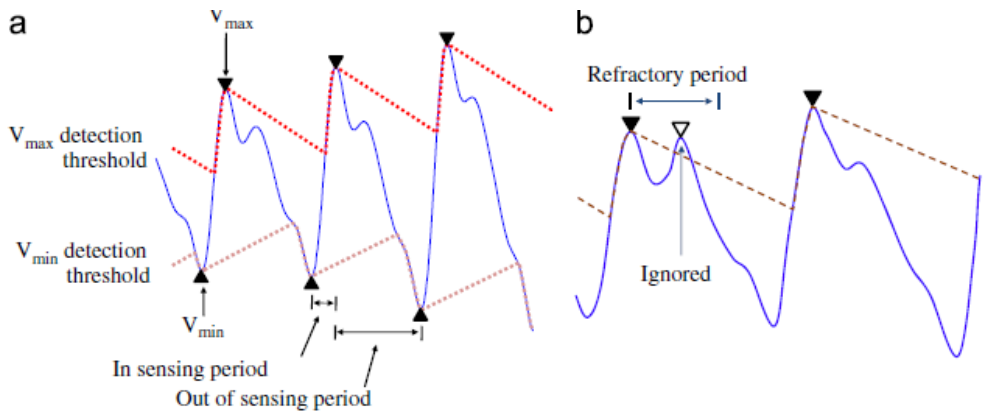


Figure 2.29 *Outline of Shin et al. 2009's [63] algorithm process presented visually. Figure* ***(a)*** *represents the detection process while figure* ***(b)*** *represents the skipping of errors and dichroitic peaks based on a distance threshold calculated from the distance between previous neighbouring peaks.*

2.3. Summary

In this chapter we analysed a variety of techniques for removing the baseline drift for their benefits and drawbacks. These techniques included Frequency Based Filtering, Wavelet Filtering, Cubic Spline Interpolation Filtering, Morphological Operators, Empirical Mode Decomposition and Hybrid techniques.

The review proved that there is no single method able to remove the drift entirely, hence cascaded filters were implemented which lead to the discussion on the necessity of feature detection. From this the main challenge of feature detection was discussed, and a series of techniques were presented and analysed for their various drawbacks and advantages to one another. The review covered techniques including the usage of the first derivative, time and spectral based transformations, and directly using signal characteristics.

Chapter 3

Problem Analysis, Research Methodology and Materials

From Chapter 2 in the literature review an overview of a current limitations of PPG signal filtering was presented. This chapter provides an analytical summary of the problem and the research methodology that was adopted to investigate the problem. Data acquisition and analysis of the procured PPG signals are undertaken to establish a benchmark dataset for investigating the filters and detection algorithms.

3.1. Analysis and Problem Formulation

The literature review presented a variety of techniques each with their own individual draw-backs and benefits. Morphological filters while being computationally efficient, has the potential to distort the signal making it undesirable. Wavelet and EMD have shown to be relatively useful for decomposing the signal to obtain the drift approximation and is the general standard. But the amount of calculations for decomposition to obtain the baseline drift is undesirable for computational efficiency. The final approximation furthermore may not be fully optimal as some of the drift may exist outside the scale.

CBSI has been noted to be computationally efficient, but cannot provide an optimal approximate in circumstances where the drift frequency content exceeds the HR content. It also relies on accurate detection of features in order to be effective. Both FIR and IIR filters have been criticised for not being able to obtain a good approximation, but there have been studies that show otherwise.

IIR filtering is known to distort the signal and while ZPF can be used to remove distortions, it cannot be used in real-time. FIR filtering is able to preserve the signal and can be used in real-time, but it's computational times make it undesirable. Due to its beneficial properties, however, it is generally overlooked, and there exists techniques that can be used to optimise the process so it can also be used in real-time. Above all, it was seen that none of these methods can completely extract the drift on their own, but the combination of wavelet or IIR-ZPF and CBSI has been shown to achieve complete delineation.

Based on the analysis of the literature in the summary above, the goal of the study was:

> *"To investigate and explore the potential of using FIR filtering in a cascaded filter in real-time against methods such as the Wavelet and IIR Zero-phase cascaded filter."*

Breaking this down the sub-goals are:

a) Determine the effectiveness of the optimized variants of the FIR filters, and their capabilities to be applied in a real-time scenario with optimization techniques.

b) Analyse detection algorithms in order to identify the one with the most optimal detection rates and the most optimal computation times. The selected algorithm will be used as part of the cascaded filter design.

c) Apply these techniques in a cascaded filter with CBSI similarly to the ones proposed by *Xu et al. 2007 [43]*, and investigate if it can perform on-par with the method in real-time.

3.2. Research Methodology

For the study that is to be undertaken, the engineering methodology by *Basili et al. 1993 [64]* was used as a basis to formulate the experimental processes *(illustrated in Fig 3.1 and Fig 3.2)*. The experimental study was divided into five individual phases.

- ***Phase One:*** Focuses on the collection of high resolution PPG signals. The data samples are analysed critically and organised into categories based on their characteristics as described in Chapter 2 within the literature review.

  A Representative Benchmark Dataset (R.B.D.) was formulated containing samples that represented each of the states of drift and morphologies based on the organisation. These samples were manually analysed and annotated for their real features as a basis for further study. The visual result of the analysis can be found listed in Appendix A.

- ***Phase Two:*** Involves the investigation of the FIR filter against the Wavelet filter and IIR Zero-phase filter. The focus of this study is to analyse how well the filter performs against established methods, as well as to gain an understanding of how the Wavelet and IIR zero-phase filter behaves on the PPG data.

  Timing and effectiveness of the baseline drift removal is considered. FIR filter optimization techniques from the literature review are applied, in order to investigate whether the computation time can be reduced, as well as if these methods differ from the direct method in any way.

- ***Phase Three:*** Focuses on the analysis of feature detection algorithms that can be used towards the development of the cascaded filter. For this stage we analysed *Xu et al. 2007 [43]'s* Derivative Calculation Based Discrimination algorithm (DCBD), *Kan et al. 2012's [40]'s* Spectral Based Discrimination algorithm (SBD), the Adaptive Threshold Detection (ADT) algorithm by *Shin et al. 2008 [63]* and the Adaptive Segmentation (ADS) algorithm by *Karlen et al. 2012 [60]*.

  We select these algorithms since they have adaptive potential while being simple, provide a solid basis for discriminating features and possess an extensible structure.

- ***Phase Four:*** Focuses on the exploration of the cascaded filters through the application of CBSI in order to remove the remaining drift. The selected detection algorithm from stage three is utilized to construct the cascaded filter with the designed FIR filter, and is compared with the entirety of the method presented by *Xu et al. 2007 [43]*.

- ***Phase Five:*** Applies the explored methods of FIR filtering utilizing the OLS algorithm with the selected feature detection algorithm and CBSI to form a cascade filter. The algorithm is applied and tested in a simulation where it is refined until component integration results are optimal.

  The analysis focuses on the algorithm's feasibility of being able to be used on a real-time basis, and the effectiveness of the methods overall in comparison to the existing techniques.

The duration of each phase went on re-iteratively for as long as necessary, until all necessary analysis had been completed, and optimal results from observations have been obtained.

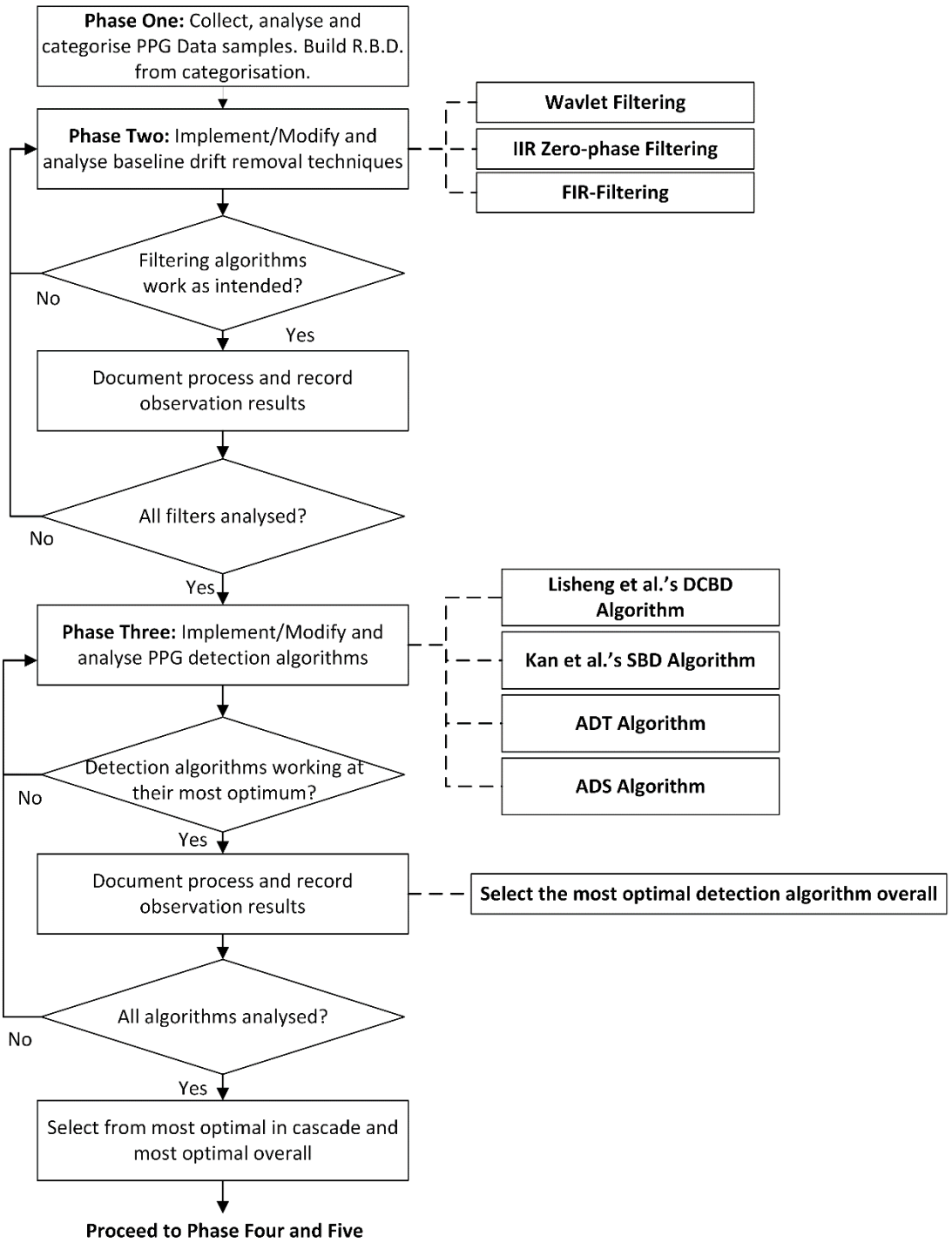


Figure 3.1 *Outline for Phases One, Two and Three.*

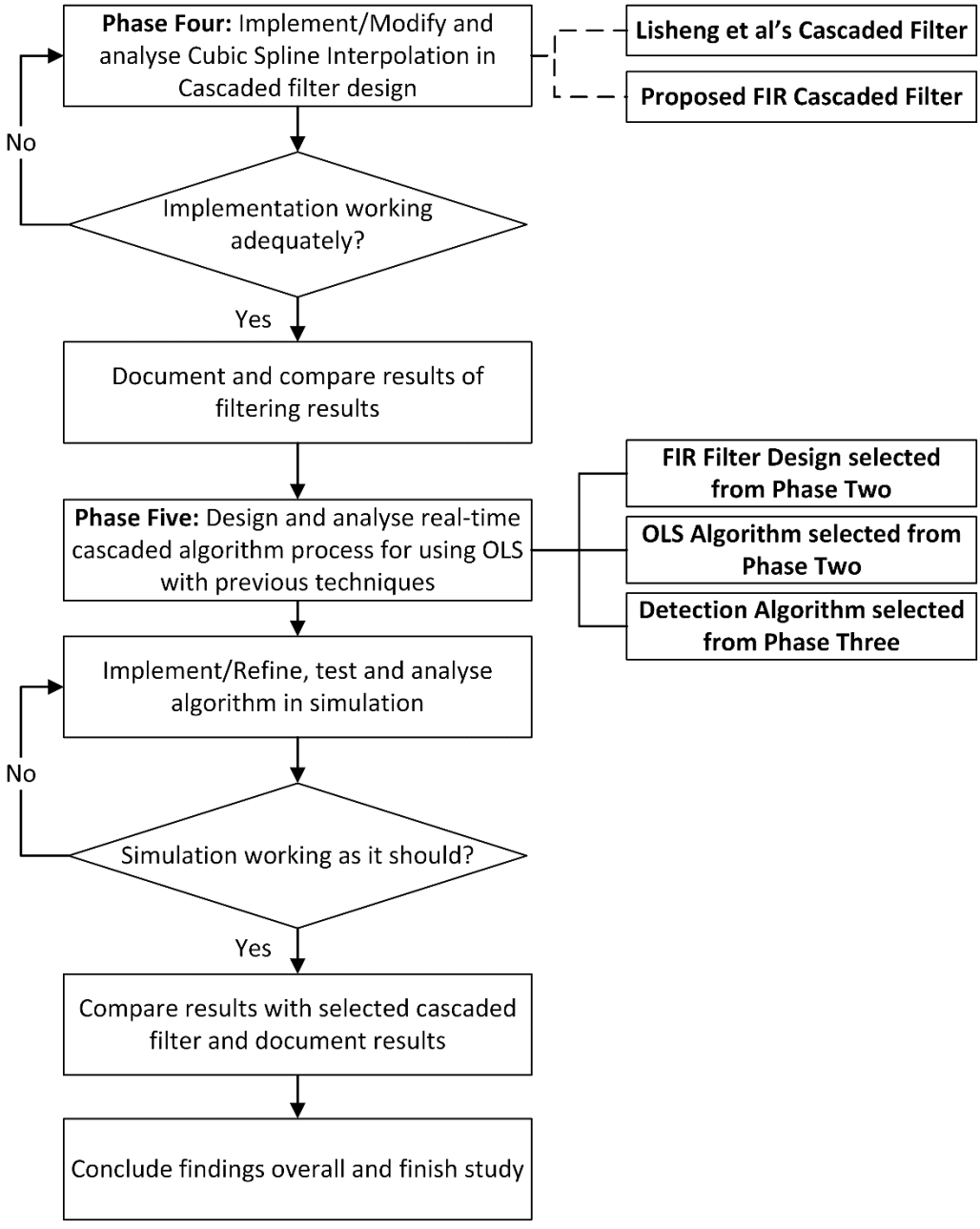


Figure 3.2 *Outline for Phases Four and Five.*

3.3. Materials Used

For the study we utilized MATLAB 2014 in order to implement the filters and algorithms selected for analysis and to analyse the data. A laptop running Windows 7 was used as the test machine. For the data acquisition, we considered the usage of the Capnnobase Data samples by *Karlen et al. 2010 [10].* We considered these samples since they were of high resolution, and was sampled from a diverse background which would allow for a variety of PPG morphologies to be captured. All source code for algorithm testing and development can be found within the online repository under the associated chapter folders[1].

3.4. Data Acquisition and Analysis

This section covers the acquisition and analysis of the PPG samples to form a basis for future studies.

3.4.1. Organisation

Data was plotted within MATLAB and analysed for the morphologies as well as drift conditions. In referencing what was discussed in Chapter 1, the PPG PW morphology can vary as having:

1. A well-defined dichroitic peak, found in younger, healthy patients
2. A moderately well-defined dichroitic peak, found in healthy, adult patients
3. Little to a complete lack of a dichroitic peak altogether, found in unhealthy or elderly patients

[1] https://bitbucket.org/nthegestalt/dissertation-code

In consideration to baseline drift there is knowledge of that there can be either:

1. Minimal drift: The signal form is almost perfect with little to no distortions
2. Moderate drift: The signal form is intact but drift displaces some of the features
3. Large drift: The signal form is heavily distorted with the drift displacing a majority of the features

Based on this knowledge of the conditions that can exist for the PPG signal we proposed an organisation matrix *(shown in Table 3.1)* in order to categorise the samples.

	Well-defined dichroitic peak	**Moderately-Little defined dichroitic peak**	**No dichroitic peak visible**
Minimal Drift			
Moderate Drift			
Large Drift			

Table 3.1 *Proposed organisation matrix structure for data samples.*

Approximately 1 minutes' worth of data or 18000 data samples is selected from each of the samples from the start. The length of 1 minute is selected to fulfil the necessary condition for observing the signal trend behaviour over a small period of time. In addition to this, the first 10-20 seconds worth of data from the start is selected for analysis to identify the morphology contained within the signal for categorization. In the event of severe distortions, another segment within the signal was selected.

From the analysis, we found that the signals *(shown in Fig 3.3)* contained healthy PWs from patients with young and elastic arteries, signals that contained unhealthy PWs from patients with old and rigid arteries, and signals that contain abnormally slow HR.

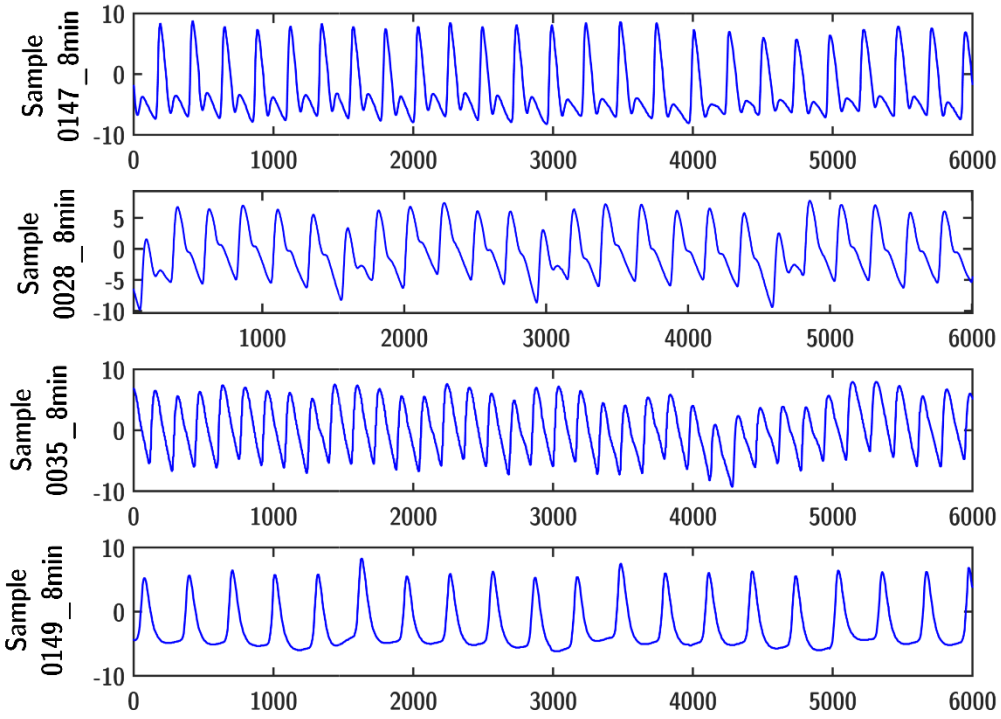


Figure 3.3 *Example of different morphologies found within the Capnobase Data samples.*

Sample 0147_8min *(shown in Fig 3.3)* shows a clear distinction between the systolic and dichroitic peak which indicated fairly elastic arteries typically found in healthy young patients. The dichroitic peak is found to merge in sample 0028_8min, but is still evident indicating healthy arteries commonly found in adults. Sample 0035_8min has the absence of the dichroitic peak indicating an elderly patient, or a patient with unhealthy arteries and high blood pressure. Sample 0149_8min possesses an irregular morphology possibly due to irregular physiological conditions. Its shape shows a complete lack of a dichroitic peak as well as wide pulse intervals, which may be attributed to a slow heart rate as well

as high blood pressure. There were data samples that contained distortions *(shown in Fig 3.4)* where the signal terminated unexpectedly due to sudden disconnect or motion from the patient. Sample 0016_8min contained a continuous signal with disconnect distortions, sample 0105_8min contains sudden motion spikes, sample 0115_8min contained disconnect and motion spikes, and sample 0370_8min contained a mixture of motion spikes and disconnects in a single area.

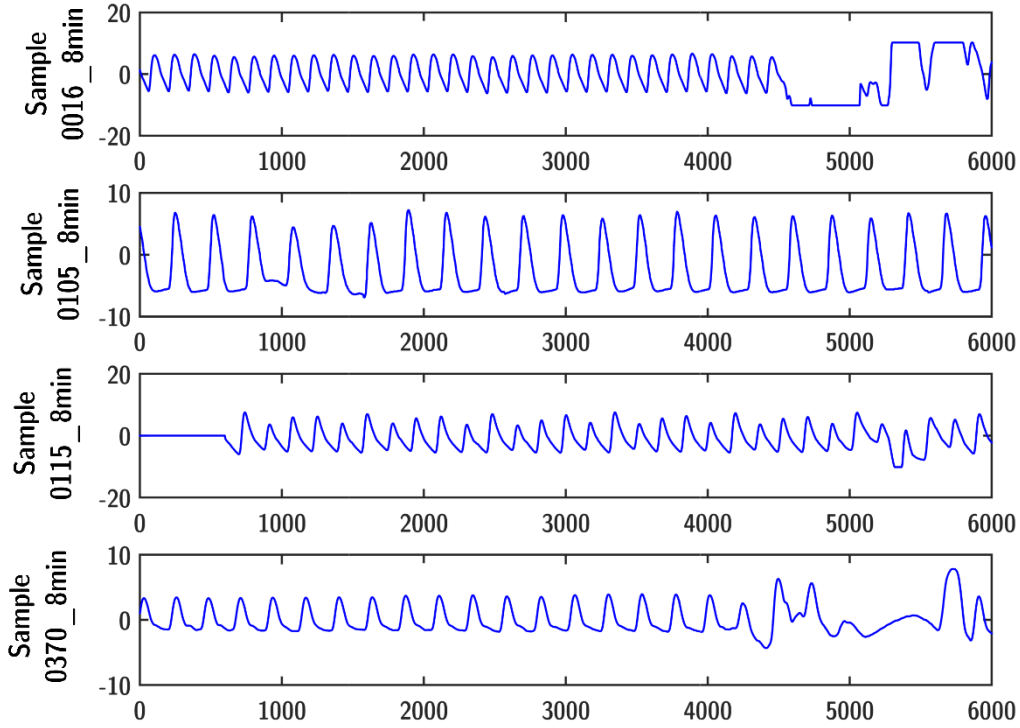


Figure 3.4 *Distortions found within the Capnobase Samples.*

Aside from distortions, samples 0028_8min and 0029_8min were found to contain a moderate drift and a morphology similar to that of the typical healthy PW morphology. Since the features and drift are somewhat in between the spectrums of extremities, either of these samples can used for the initial calibration of the algorithm parameters before the application on other data samples. The analysis allowed us to organise the samples into the categories of the matrix listed in Table 3.2.

Table 3.2 *Organisation of Data Samples from Capnobase PPG database.*

	Well-defined dichroitic peak	**Moderately-Little defined dichroitic peak**	**No dichroitic peak visible**
Minimal Drift	0121_8min 0122_8min 0127_8min 0128_8min 0134_8min 0147_8min	0031_8min 0125_8min 0142_8min 0329_8min 0370_8min *(corr)*	0016_8min (corr) 0018_8min 0104_8min 0105_8min (corr) 0133_8min 0148_8min 0149_8min 0311_8min 0313_8min 0325_8min
Moderate Drift	0029_8min 0309_8min 0333_8min	0023_8min 0032_8min 0150_8min 0322_8min	0015_8min 0331_8min
Large Drift	0028_8min 0103_8min 0123_8min 0328_8min	0030_8min 0038_8min 0312_8min	0009_8min 0035_8min 0115_8min (corr)

(corr) – Indicates that the sample was found to be corrupted from sources other than baseline wander such as disconnect and motion.

3.4.2. Establishing the Representative Benchmark Dataset

From the categorisation a select number of samples were picked in order to form a Representative Benchmark Dataset *(R.B.D)*, which comprises of samples that represents a summary of the varying conditions found in the Capnobase samples. The R.B.D. is used as a benchmark to investigate how each algorithm or filter reacts towards the different conditions that may exist within PPG signals. We selected the following samples for the following characteristics:

- ***0009_8min*** **-** Contains no dichroitic peak and an extreme but stable drift component

- ***0028_8min*** – Contains the standard PPG PW form with a moderate but stable drift

- ***0030_8min*** **–** Contains somewhat of a defined dichroitic peak which is superimposed onto the near top of the systolic peak. Drift varies greatly and has unstable amplitudes through time.

- ***0031_8min*** **-** Contains a moderately defined dichroitic peak which is superimposed onto the main systolic peak possessing low drift but a sudden spike

- ***0032_8min*** – Similar to sample 0031_8min but the dichroitic peaks are more defined and the drift is more moderate

- ***0147_8min*** - Contains well separated and defined dichroitic peaks from the main systolic peaks

- ***0149_8min*** – Contains no dichroitic peak and a low drift component

- ***0309_8min –*** Contains a well-defined dichroitic peak but exists close to the drift noise which varies moderately but remains stable

- ***370_8min*** – Contains both disconnects and motion based distortions found in the other samples with distortions

These samples were analysed accordingly for their peaks and valleys which are totalled for benchmarking purposes, and a preview of these signals are provided within Appendix A.

3.5. Summary

In this chapter, we presented an overview of the techniques with their associated strengths and limitations. From these possible optimal techniques were chosen, and their limitations was used to formulate a goal for the study to overcome these limitations. The problem formulation gave way to the establishment of the research methodology, and the processes that we aimed to take in order to reach the end goal of building a cascaded algorithm, which can be used in real-time to delineate the PPG signal.

From there, high resolution PPG signals were acquired and analysed. These data samples were organised based on their characteristics of drift and morphology within a proposed matrix structure. Using the matrix, we selected individual samples that represented the individual morphologies. These samples formed the basis of the representative benchmark dataset *(R.B.D)* which is to be used within future studies for the development of the algorithm.

Chapter 4

Analysing Techniques for Softening the Baseline Drift

Chapter 4 focuses on the exploration of techniques to soften the baseline drift. Wavelet filtering is explored in order to establish an understanding how much of the drift it is able to eliminate, and to establish a basis for comparing the other techniques. IIR and FIR filtering are explored to see if they can perform on par with the Wavelet filter. In addition, optimization techniques for the FIR filter are applied and explored, in order to test the feasibility for reducing the computation constraints that will allow for real-time processing.

4.1. Experimental Setup and Procedure

The filters were implemented based on the description provided within selected literature, and were tweaked where needed in order to allow adequate operation and performance. Calibration was done through the usage of sample 0028_8min since we considered it to contain the standard PPG signal with features existing in between extremities. Two tests were utilized; a short test and a long test. The short test spanned 20 seconds *(6000 samples)* worth of data taken from the start, with the purpose of investigating to see how the filters performed on a short term basis. The long test spanned 60 seconds *(18000 samples),* and had the purpose of verifying the results of the short test, as well as to investigate how the filters performed in the long run.

Visual and numeric analysis was carried out on the results of the tests to verify their closeness to the wavelet filter's approximation. Similarly to previous studies shown in the literature review, Pearson's Correlation Coefficient *(defined in Eq. 4.1)* was used to measure to the similarities between the methods' approximations.

$$R = \frac{cov(x,y)}{\delta_x \delta_y} \qquad \textit{(Eq. 4.1)}$$

Where:

- x = Wavelet Filtered signal
- y = Other Filtered signal
- $cov(x, y)$ = Covariance of x and y
- $\delta_{x,y}$ = The standard deviation of x and y respectively

Whereas the RMSE *(defined by Eq. 4.2.)* is used as a measure of errors or differences between the results of the method.

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(y_i - x_i)}{N}} \qquad \textit{(Eq. 4.2)}$$

Where:

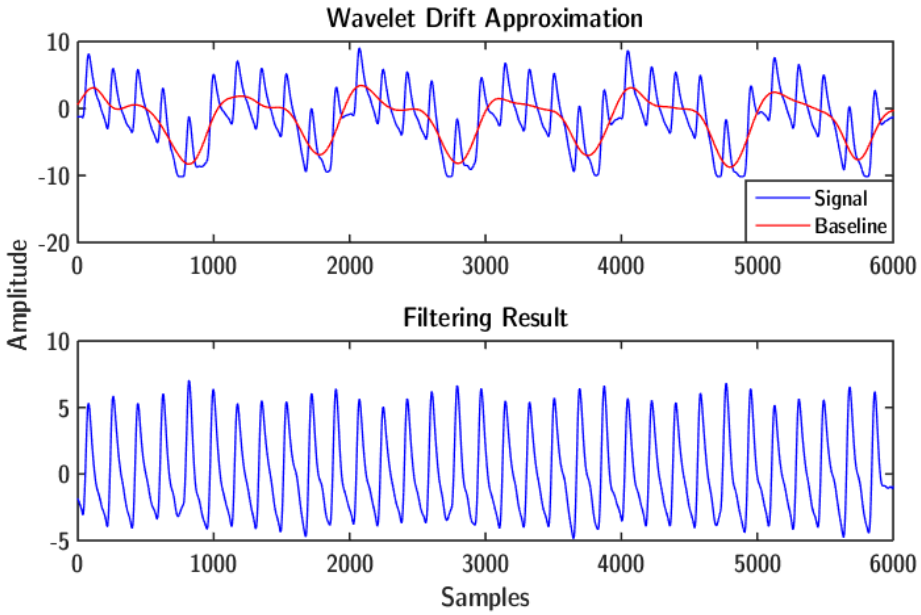
- y = Wavelet filter conditioned signal
- x = Other Filtered signal
- N = Length of signal
- i = The i-th element of x and y respectively

In addition to the correlation and error measurements, the computational time of the methods are taken for a comparison in efficiency. The complete list of numerical correlation test results can be found listed in Appendix B1. The code for our filters can be found within their associated file names on our code repository[2] under the *"chapter_four"* folder.

[2] https://bitbucket.org/nthegestalt/dissertation-code/

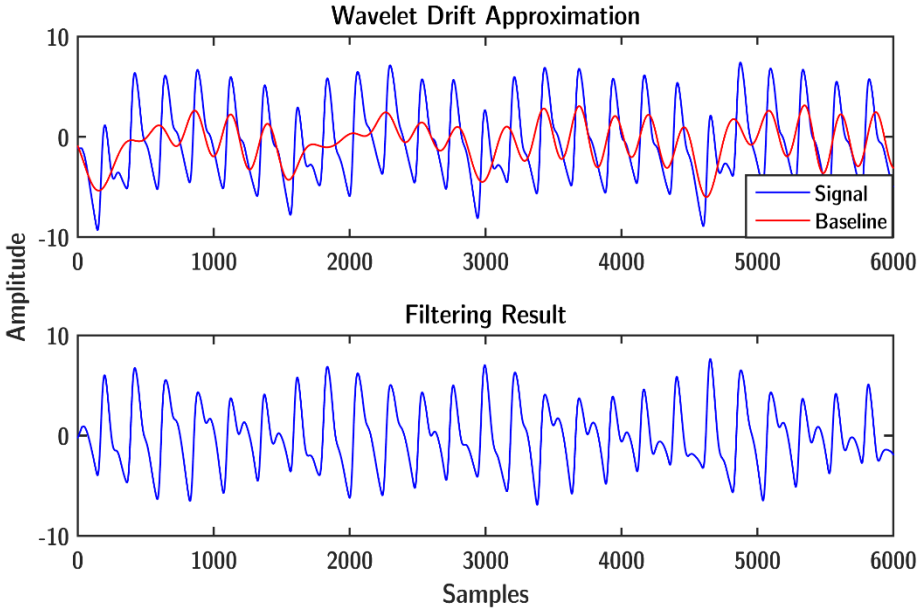
4.2. Wavelet Filtering

The Wavelet filter was implemented in MATLAB using the FWT, with the mother wavelet selected as the *"D.Meyer"* wavelet in accordance *Xu et al. 2007's [43]* report; it demonstrated higher resolution at lower frequencies in contrast to other wavelets making it beneficial for our purposes. An initial scale of 7 selected as a starting point for exploration, and the filter was utilized on sample 0028_8min. The results revealed an insufficient decomposition *(shown in Fig 4.1 a)*, but further tests yielded mixed results such as that seen in sample 0009_8min, revealing that the scale was perfect for approximating the drift *(shown in Fig 4.1 b)*.



(a). *Example of an ideal estimation using sample 0009_8min*

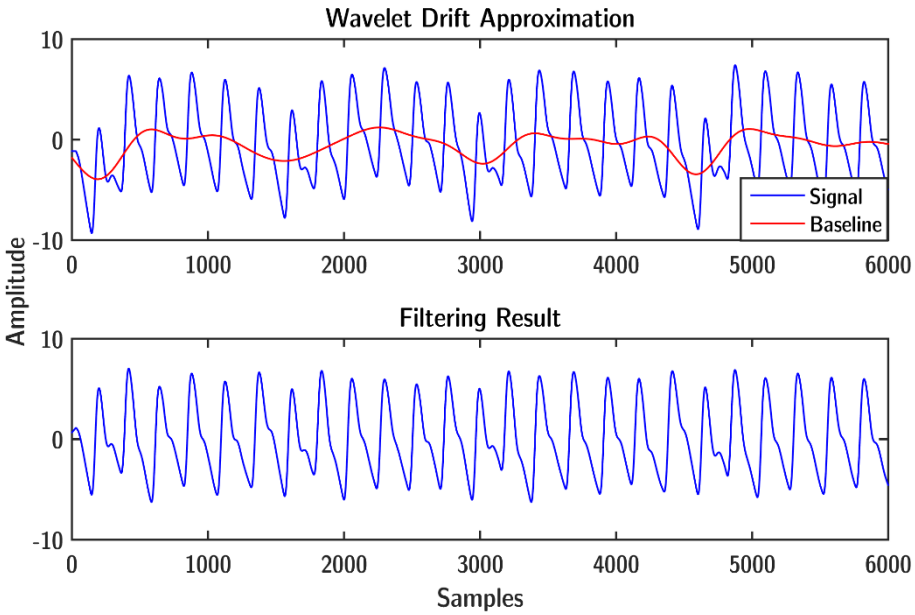
Figure 4.1 *Examples of Wavelet Decomposition at scale = 7.*



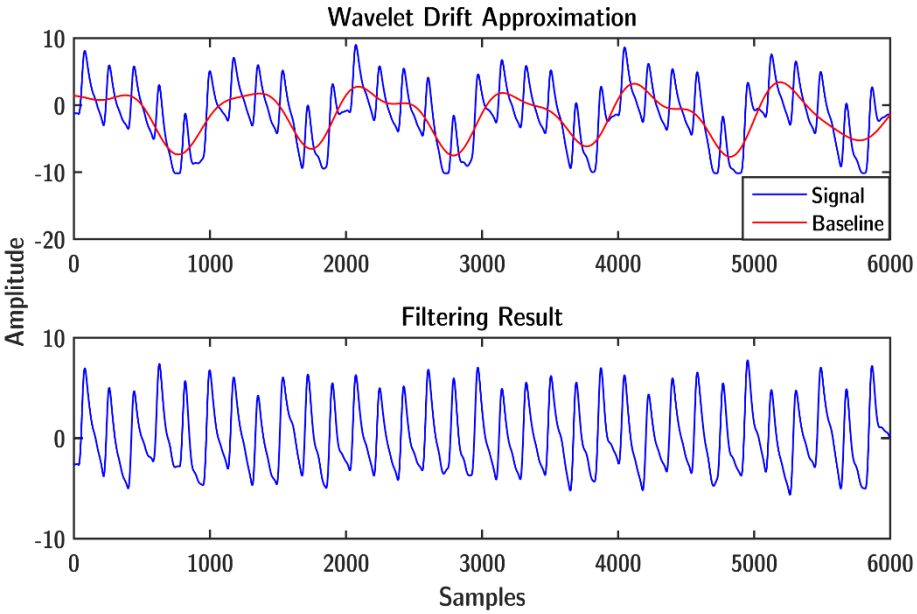
(b). *Example of overestimation causing distortions using sample 0028_min.*

The scale was subsequently increased to 8 which allowed for better approximations to be yielded with some of the samples. However, it was found to achieve less optimal approximations when applied to sample 0009_8min, in comparison to using a scale of 7 *(shown in Figure 4.2 a and b).* Despite the less optimal scale used, we can see that within sample 0009_8min that by softening the drift, we were able to nevertheless make the displaced valleys more identifiable. Based on our initial testing, each of the samples within the R.B.D. was filtered within short and long tests using scales of 7 or 8 in order to obtain the most optimal drift approximation.

The behavior from the filtering process analysis indicates potentially a similar problem as described by previous studies *[43, 53, 59]*. While the baseline drift can be approximated by the lower level details, there are times when the decomposition cannot completely approximate the baseline drift. This was due to some of the drift frequencies existing outside of the range of the decomposition details. In the scenario where a higher level is utilized in the hopes of approximating the drift, there may be contaminants of HR frequency information which can lead to distortions.



(a). *Example of an ideal estimation with sample 0028_8min.*



(b). *Example of a slightly less ideal estimation with sample 0009_8min. In comparison to Fig 4.1 (a), the approximation seems to be slightly less detailed which led to an incomplete removal.*

Figure 4.2 *Examples of Wavelet Decomposition at Scale = 8.*

The results affirms the need for hybrid algorithms in order to completely delineate the signal instead of only relying on a single method. In addition to this, the analysis also indicates the potential difficulty of parameter selections in order to achieve the most optimal decomposition level as described by *Dianguo et al. 2008 [45].* The run-times for tests *(described in Table 4.1)* reported an average of 0.02828 seconds for the short term tests and 0.04832 seconds for the long term tests. The complete timing results of the filtering as well as scales used is listed found in Appendix B1.

Table 4.1. *Average run-times for the Wavelet filter*

	Filtering Run-times
Short Term (20 Seconds)	0.02828
Long Term (60 Seconds)	0.04832

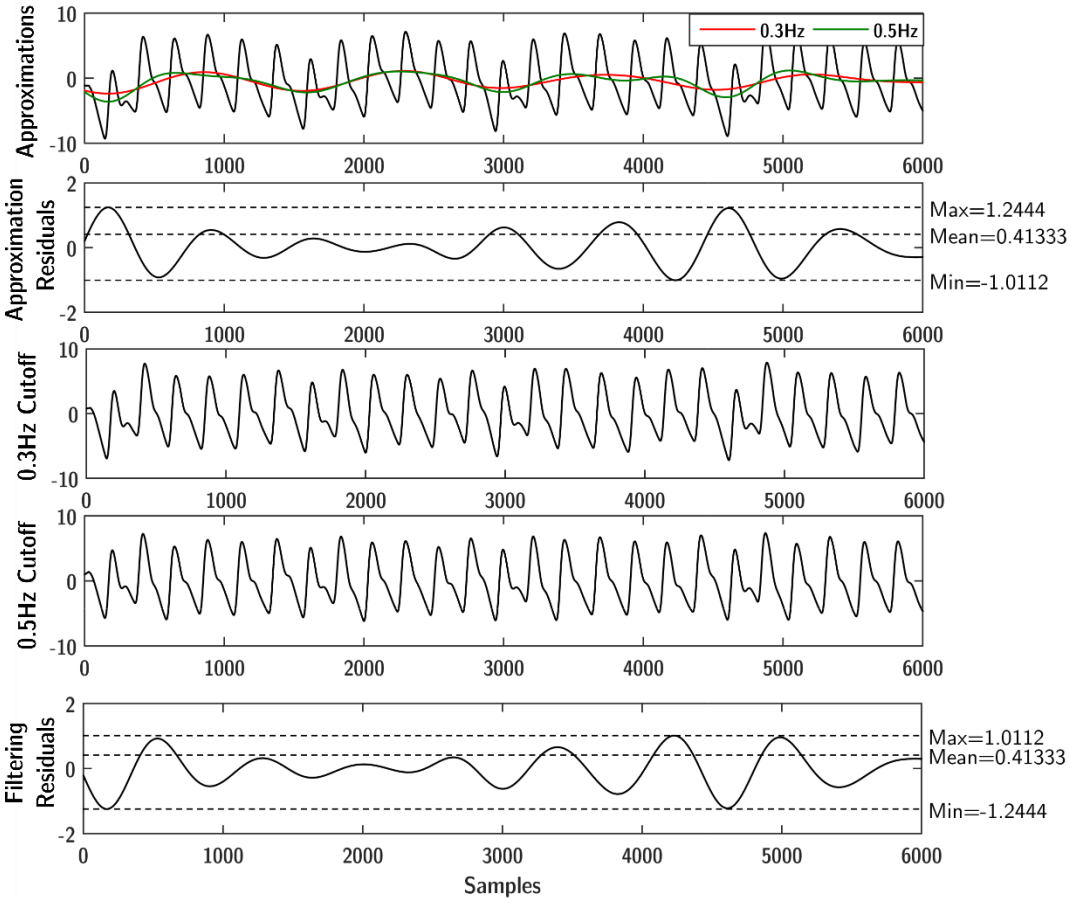
Regardless of the PW characteristics, the visual test results shows that the filter was able to soften the baseline drift and in some cases, eliminate it entirely if the drift was small enough. Scale selection should still be considered in order to prevent the distortion of the signal, but overall Wavelet filtering remains to be an effective means to soften the drift of the signal, but not remove it completely.

4.3. IIR Zero-phase Filtering

For the design of the IIR filter, several Butterworth filter design method have been described in literature with an order ranging anywhere from 2nd to 6th for the removal of the baseline drift *[39, 65, 66]*. We selected an order of 6 since higher filter orders will result in sharper frequency cut-offs. The cut-off frequencies recommended are either 0.3Hz by *Kan et al. 2012[40]* or 0.6Hz by *Chunming et al. 2008 [39],* which exists below or above the frequencies of the respiration and nervous system activities, and within the levels of the HR

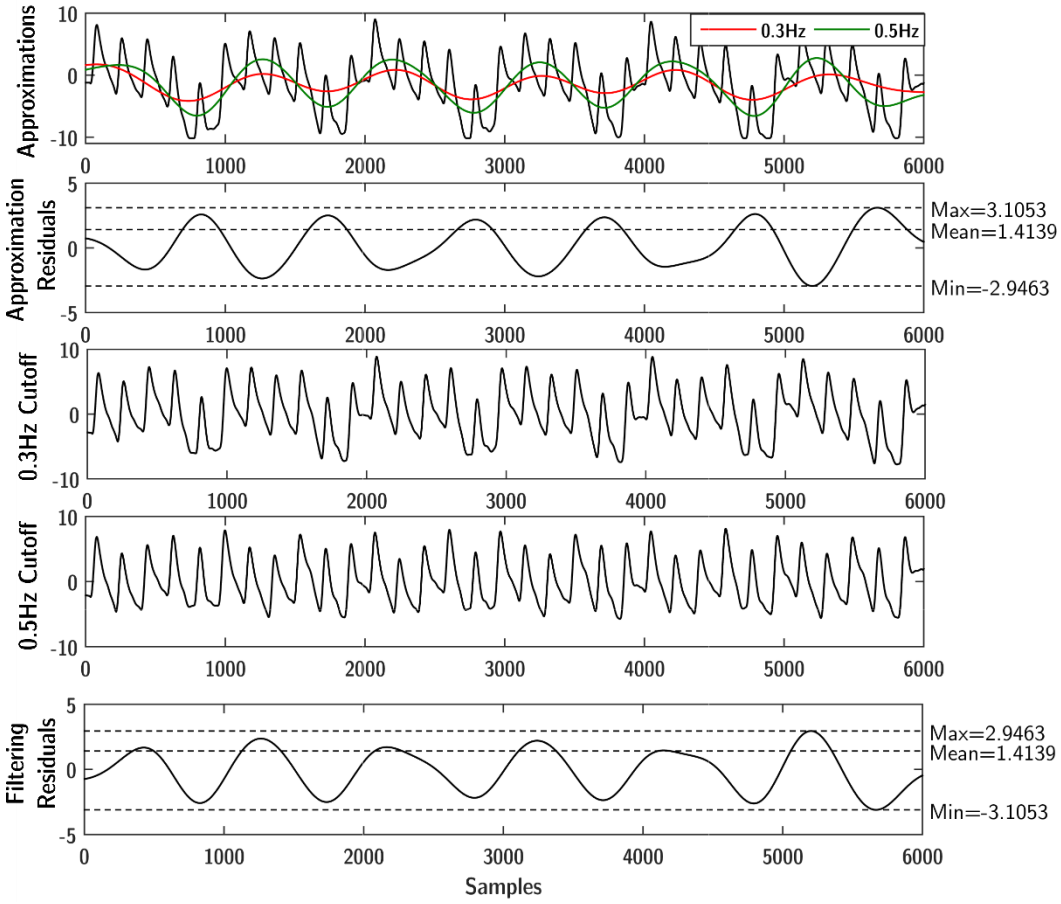
frequencies. As an alternative, values of 0.3Hz and 0.5Hz were used instead of 0.6Hz, since it may cause over approximation of the HR frequencies. A short test on sample 0028_8min, yielded no significant visual differences in the filtered results *(shown in Fig 4.3 a)* except that the 0.5Hz cutoff was more detailed.

Further testing indicates that the cut-off frequency was too low such as in the case of sample 0009_8min and sample 0309_8min *(shown in Fig 4.3 b and c)*. No distortions were found from utilizing a cut-off frequency of 0.5Hz throughout the testing, but significant improvements were seen overall.

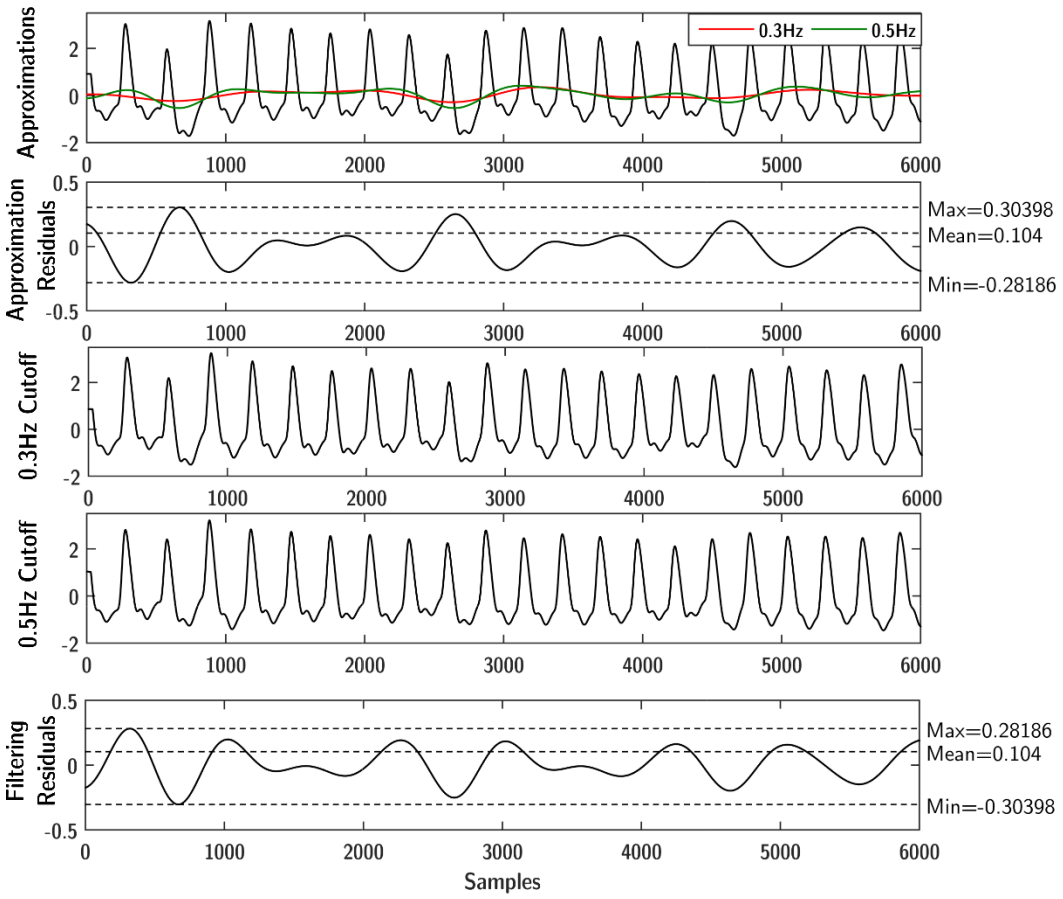


(a). *No significant differences in sample 0028_8min shown by the overlap between 0.3Hz and 0.3Hz cut-off frequencies.*

Figure 4.3 *Testing cut-offs against R.B.D. samples.*



(b). *Differences in cut-off frequencies on tested on sample 0009_8min.*



(c). *Differences in cut-off frequencies on tested on sample 0309_8min.*

To affirm the results seen in the time domain view, we analysed the spectral contents of the filter results for each of the cut-off frequencies selected. Within the spectral contents, we can see clearly that the 0.5Hz was able to extract a more detailed approximation in comparison to the 0.3Hz with the current filter order and design method. The criteria for filter design was to approximate as much of the drift as possible, while preserving any frequency content above 0.5Hz. In consideration to the results (shown in Fig 4.4), we determined that a cut-off of 0.5Hz was therefore a suitable cut-off frequency for the filter design.

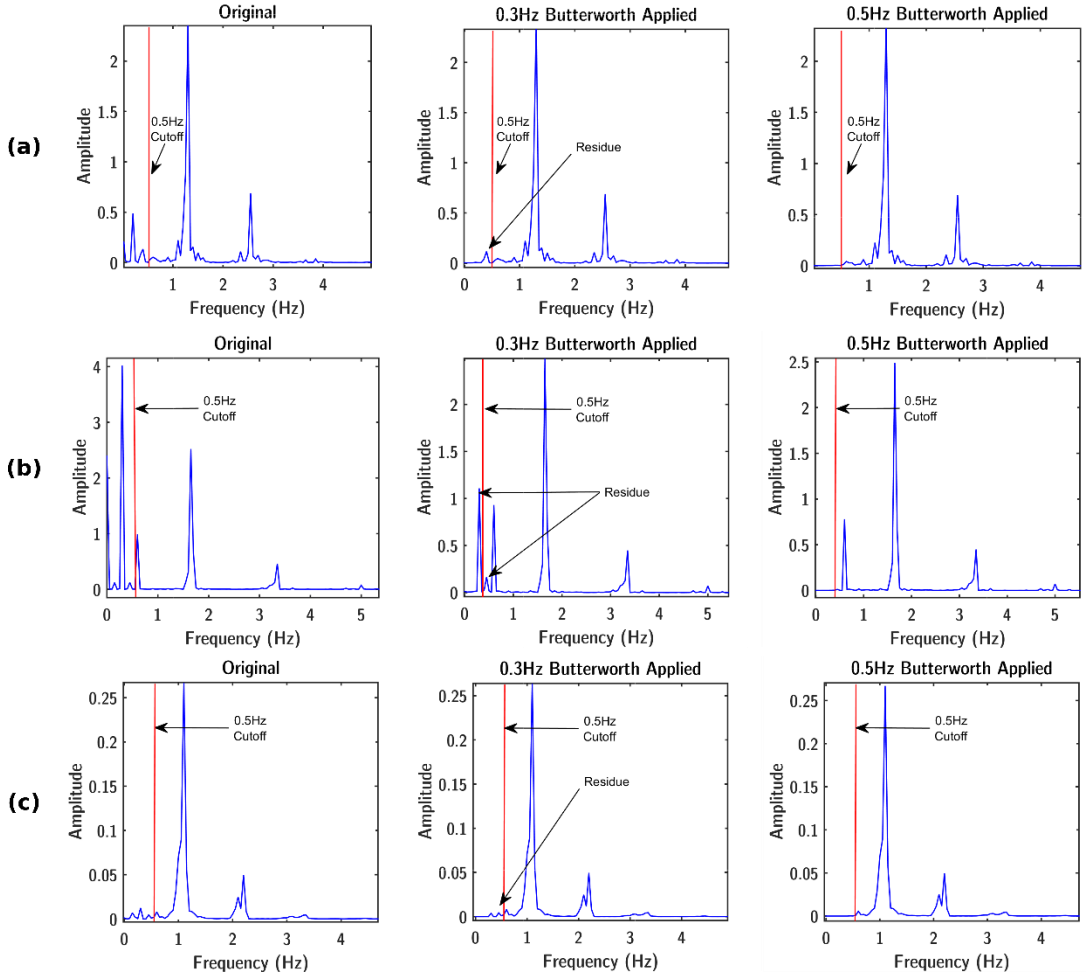


Figure 4.4. *Spectral View of 0.3Hz vs 0.5Hz cut-off in filter design. Rows are as follows:* ***a).*** *Sample 0028_8min* ***b).*** *Sample 0009_8min* ***c).*** *Sample 0309_8min*

We filtered the remaining signals within the R.B.D. samples and performed correlation, and comparative timing tests with the wavelet filter results. The full visual and numerical results of these tests are listed in Appendix B1.2. The average results of the short tests *(displayed in Table 4.2)* indicated that the IIR-ZPF filter outperformed the wavelet filter significantly by around 96-97% of the wavelet filter's run-time. This was possible since it was able to remove the drift directly instead of relying on a series of decompositions.

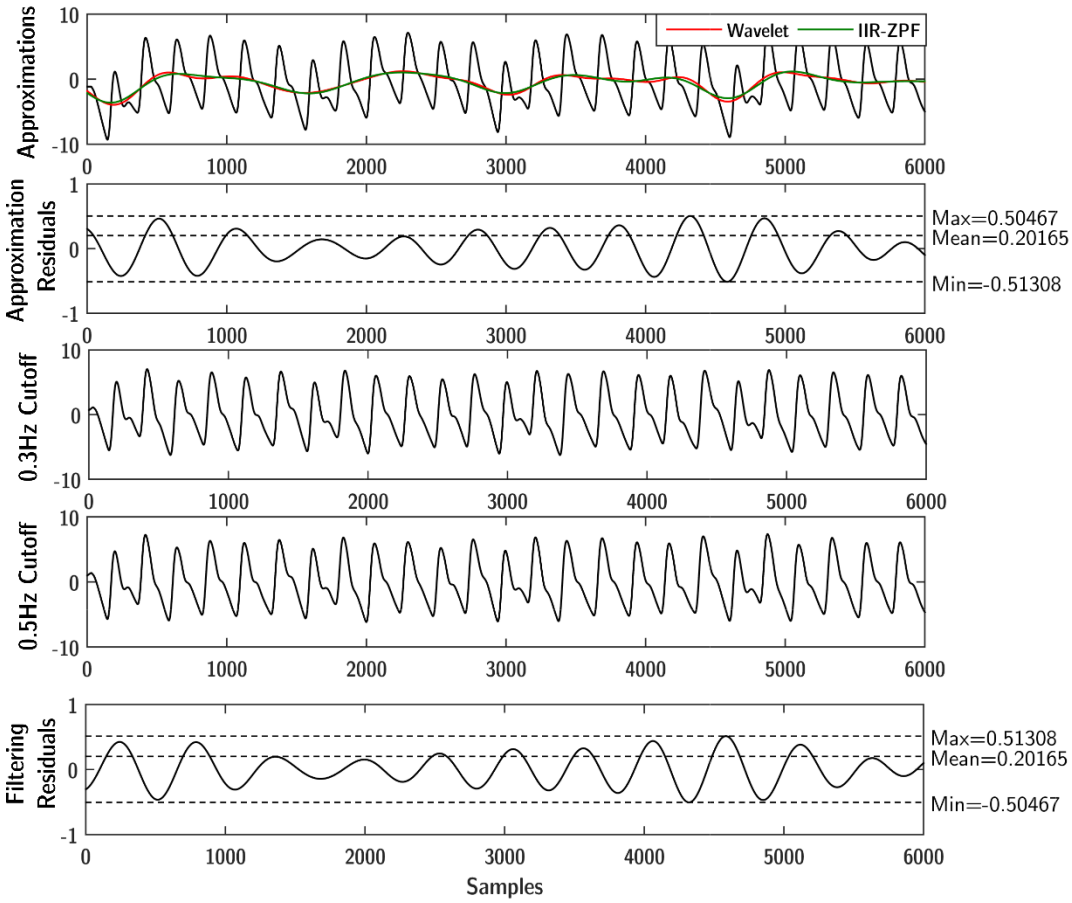
Table 4.2 *Timing Result Comparison for the Wavelet filter and IIR-ZPF*

	IIR ZPF	**Wavelet**
Short Test	0.000578	0.02828
Long Test	0.001584	0.04832

Approximation-wise the wavelet filter provided a better approximation *(shown in Fig 4.5 a and b)* in circumstances such as that of sample 0009_8min. In other circumstances it remained relatively similar such as that of sample 0028_8min. Numerical analysis for the test results *(described in Table 4.3)* confirms that there are similarities, but there also existed a relative amount of error between the approximation methods, and the final filter results at an average of 97% correlation and 0.438-0.465 RMSE.

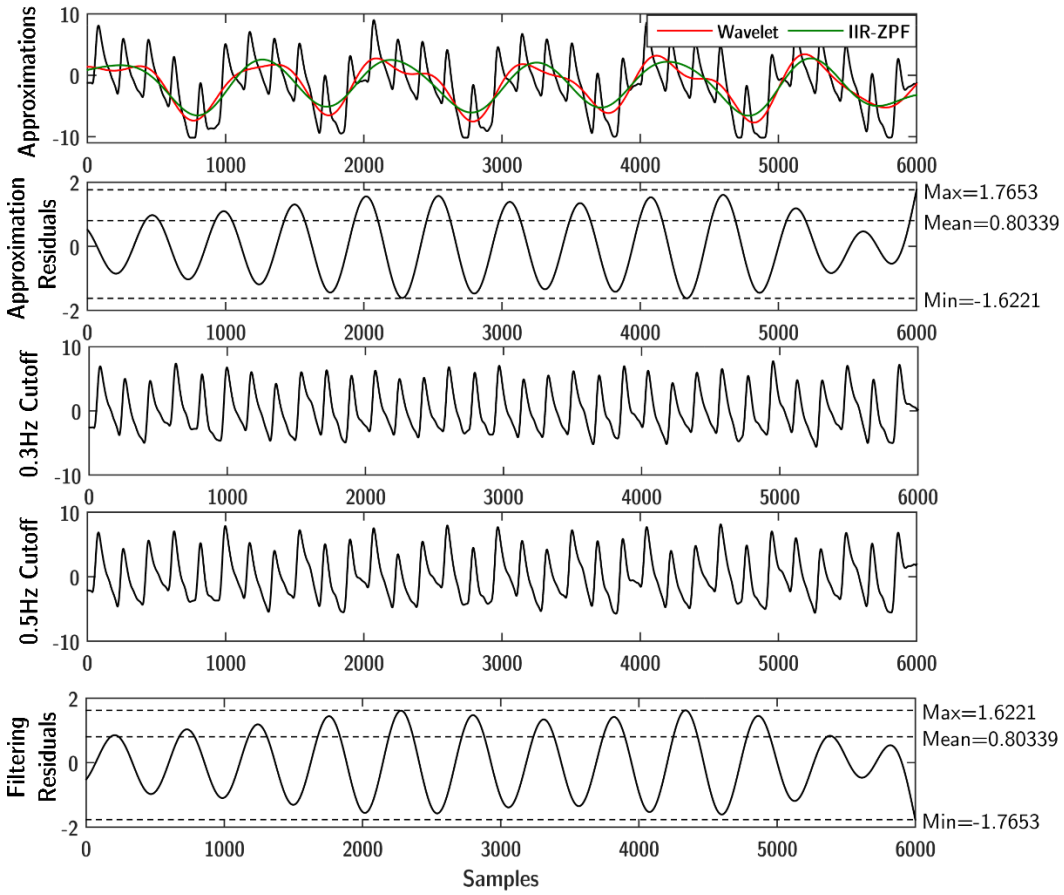
Table 4.3 *Correlation and error tests for the R.B.D. samples overall as an average*

	Approximation Correlation	**Approximation RMSE**	**Result Correlation**	**Result RMSE**
Short Test	0.94816925	0.424868125	0.97951513	0.4389035
Long Test	0.949114125	0.46501025	0.97882725	0.46501025



(a). *Sample 0028_8min showing high similarities*

Figure 4.5 *Comparison of Wavelet filtering and IIR-ZPF results.*



(b). *Sample 0009_8min showing extreme differences.*

Analyzing the spectral contents of the pre-filtering and post-filtering results *(shown in Fig 4.6)* for sample 0028_8min, we see that there were drift frequencies removed at 0.5Hz and below. Sample 0009_8min contained a drift that existed predominantly at 0.3Hz, but also contained some drift that existed at 0.6Hz as well which is considered to be above the normal frequency range for controlled respiration. Both the Wavelet filter and the IIR-ZPF were able to successfully remove the drift within sample 0028_8min, however, the IIR-ZPF was unable to remove the 0.6Hz drift in sample 0009_8min compared to the wavelet filter at scale 7.

On further comparisons of the filter approximations *(shown in Fig 4.7)*, we saw that the scale 8 wavelet filter on sample 0009_8min still had better approximations overall than the IIR-ZPF. The results indicated that the IIR-ZPF designed is relatively precise, but it also tells us that the Wavelet filter would still achieve better results. This is primarily due to its ability to decompose the signal into frequency ranges based on the signal, rather than a specified range based on a cut-off value. But this ability however also indicates potential spectral leakage as described by *Xu et al. 2007* [43] when there is little to no drift within the signal.

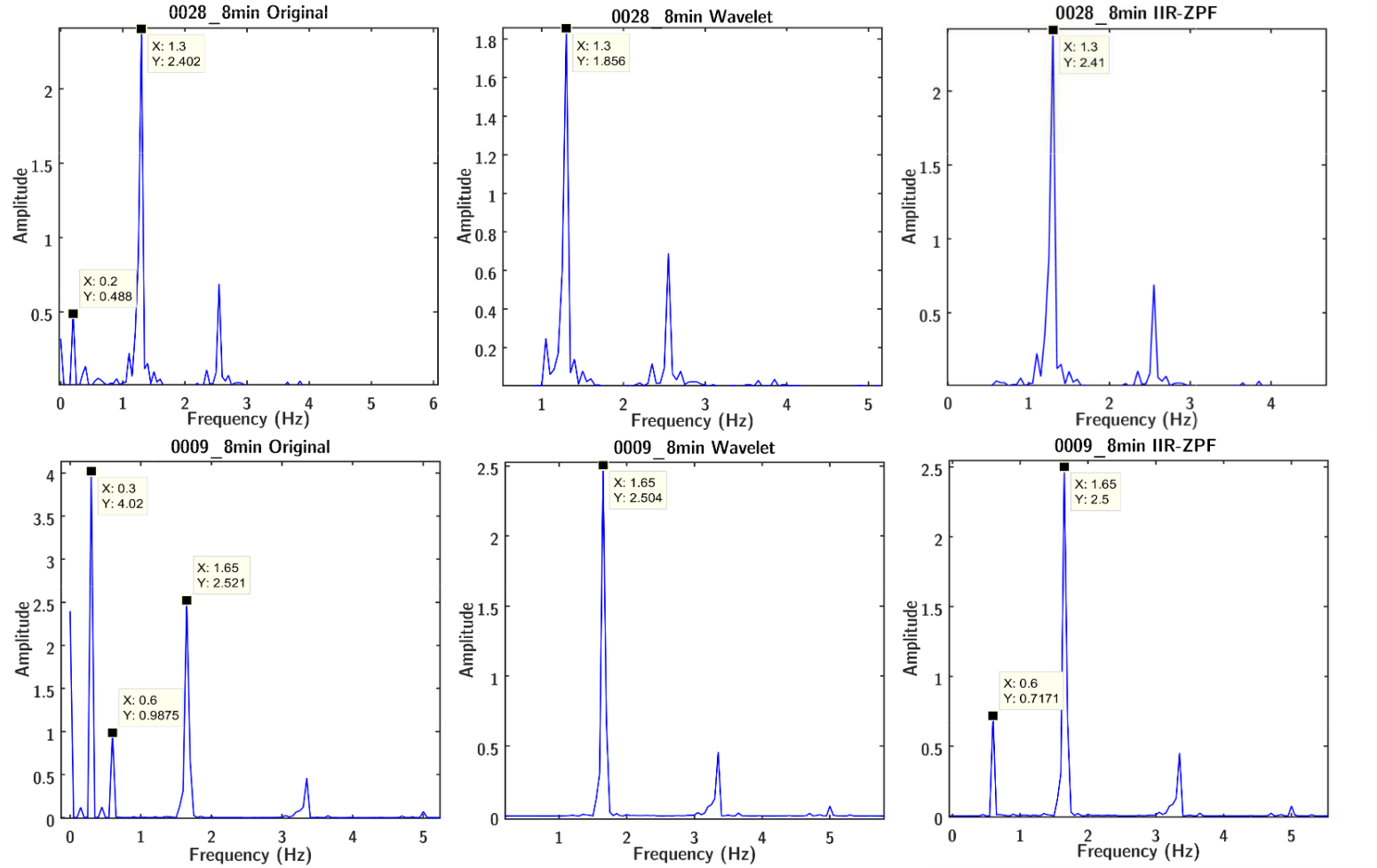
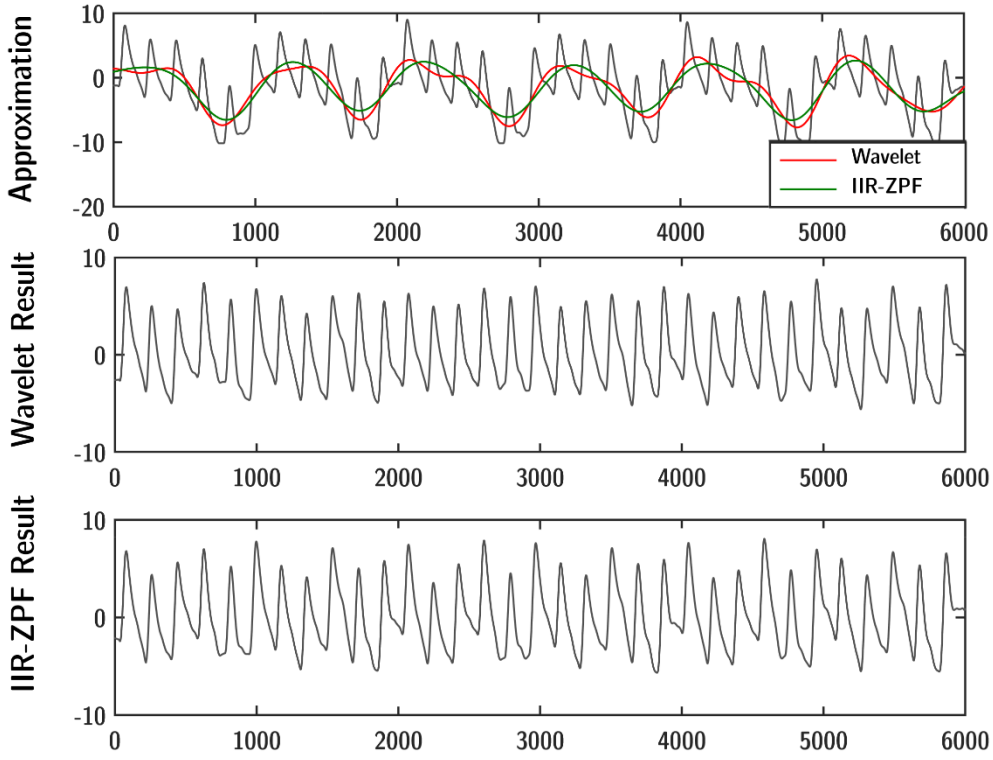
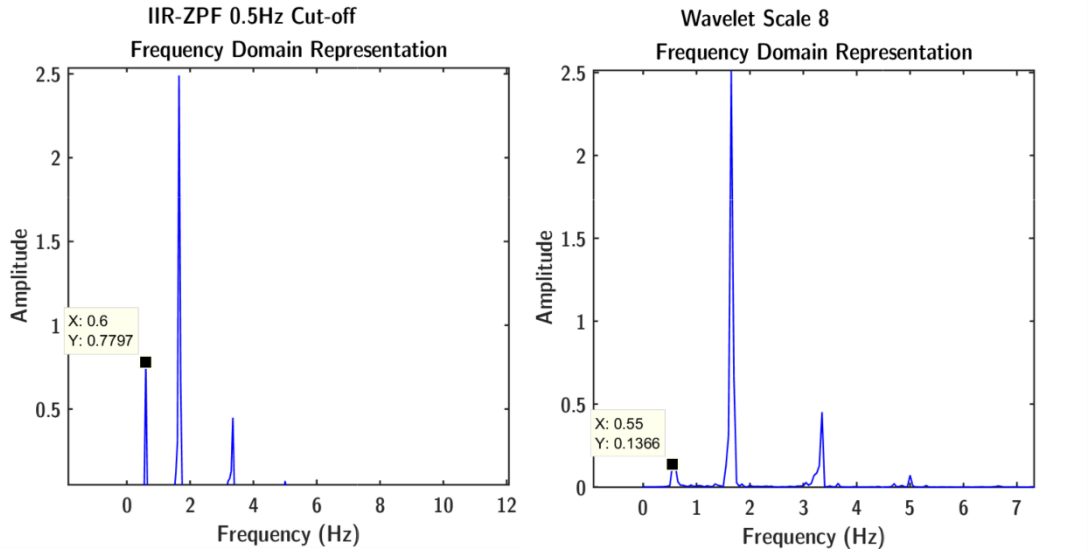


Figure 4.6 *Comparison of spectral contents for samples 0028_8min and 0009_8min pre-filtering and post-filtering with Wavelet filter and IIR-ZPF filter.*



(a). *Time domain comparisons.*



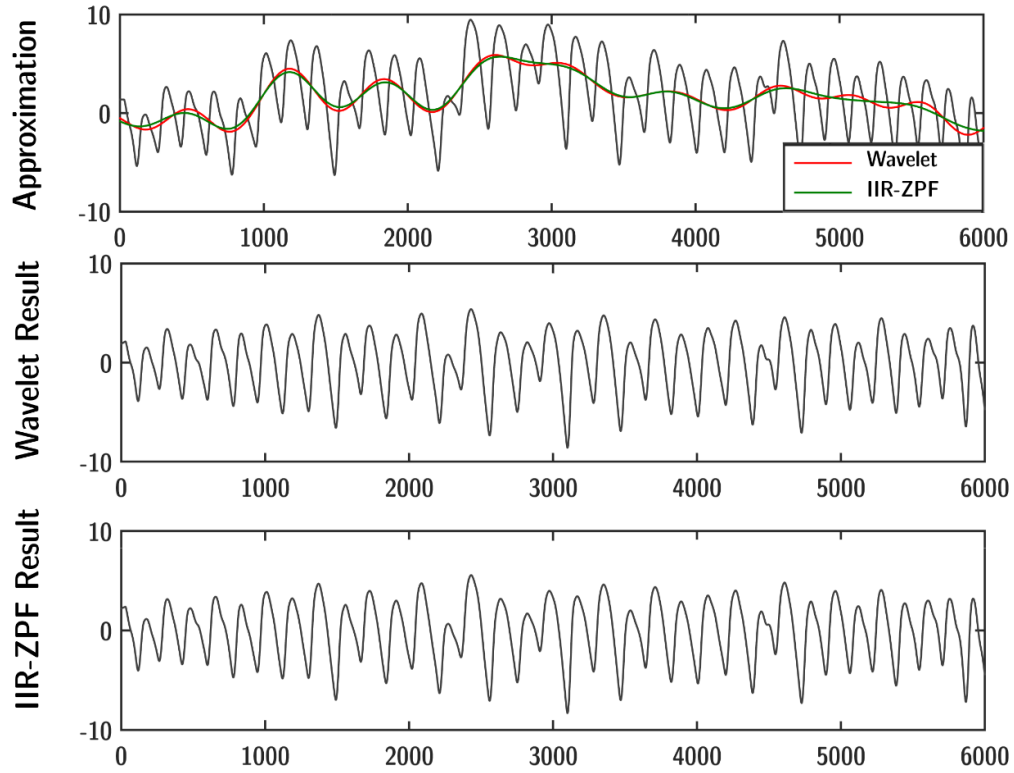
(b). *Spectral contents of results showing the difference in the elimination of the 0.5Hz to 0.6Hz drift component*

Figure 4.7 *Comparison of the IIR-ZPF at 0.5Hz and the Wavelet filter at scale 8.*

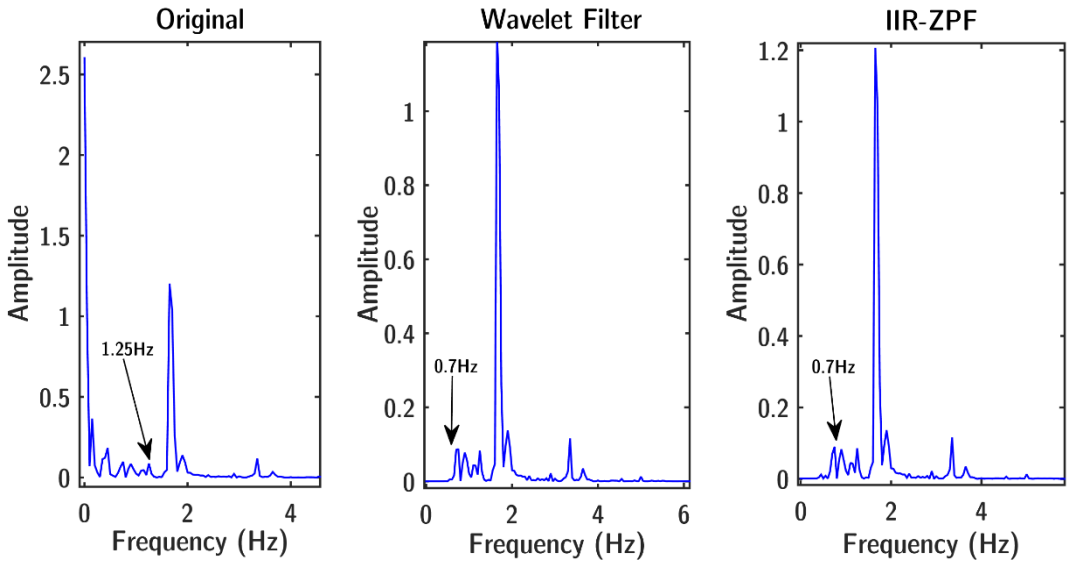
To investigate how the filters performed with time-varying drift, we analysed sample 0030_8min and found indications that both filters were able to achieve roughly the same results *(shown in Fig 4.8 b).* Both filters was able to only eliminate the range of drift frequencies up from 0.1 to 0.5Hz in essence reducing content complexity. In sample 0030_8min we see a variety of drift frequencies from 0.1Hz all the way to 1.25Hz with gaps between 0.45Hz and 0.7Hz, and a systolic frequency component of around 1.65-1.8Hz. In comparison to the Wavelet filter, the IIR-ZPF was able to eliminate the drift frequencies below 0.5Hz. But the results were not as clean as the Wavelet approximation indicated by the small bumps behind the 0.7Hz component.

This effect can be visually seen within the time domain *(shown in Fig 4.8)* through the small discrepancies seen within the IIR-ZPF result in comparison to the Wavelet filter result. The Wavelet approximation appeared to contain slightly more detail than that of the IIR-ZPF approximation. Taking into consideration what we observed when sample 0009_8min was parsed by different wavelet scales, we can say that the Wavelet filter in most circumstances will be limited to approximating frequencies within a specified range. However, it nevertheless possesses the potential to obtain a more detailed approximation.

While it can approximate parts of higher frequencies outside its specified range providing it is close enough, such spectral leakage can be undesirable due to the potential it has of distorting the signal by over-approximating. Due to this, techniques such as polynomial interpolation should be used to target specific areas, where the drift exists above 0.5Hz to remove its addition to the signal without risking signal distortion. Overall, similarly to the wavelet filter, we found no adverse visual and spectral effects on the PW morphologies nor was there any significant change. The results conclude that properly designed cut-off filters can perform on-par with the wavelet filter, and can obtain better computation times.



(a). *Time domain comparison.*



(b). *Spectral Content of original and filtered results.*

Figure 4.8 *Comparison of the IIR-ZPF filter against the wavelet filter at scale 8 on sample 0030_8min.*

4.4. FIR filtering

For the construction of the FIR filter we utilized the window design based method as utilized by previous literature *[35, 36]*. The common windows Hamming, Hanning, Kaiser, Rectangle, Elliptical, Flat Top and Blackman windows were considered for usage. The criteria to search for the most optimal window base window is based on two categories *[67]*:

1. The size of the main lobe and how fast it falls off in amplitude, indicating how sharply the cut-off frequency is achieved. The right sharpness of the cut-off will allow for frequency components within the pass-band to be kept, and other out-of-band frequencies to be eliminated. A slow and gradual cut-off is undesirable, since it will cause the inclusion of other frequencies within the approximation.

2. The size of the side lobes, and how low they are in comparison to the main lobe are considered. Higher lobes indicate the potential for spectral leakage to occur into the approximation of the main lobe hence, lower amplitude level lobes with a fast fall-off from the main lobe are generally preferred.

For the case of isolating the baseline drift from the signal, a slightly sharp cut-off is desired in order to prevent the leakage of higher frequency components from the cardiac information frequencies. From the analysis of the window functions *(shown in Fig 4.9)*, the rectangle window shown in blue possesses an ideally narrow main lobe, but the side lobes remain close and relatively high in amplitude levels making it undesirable due to potential spectral leakage.

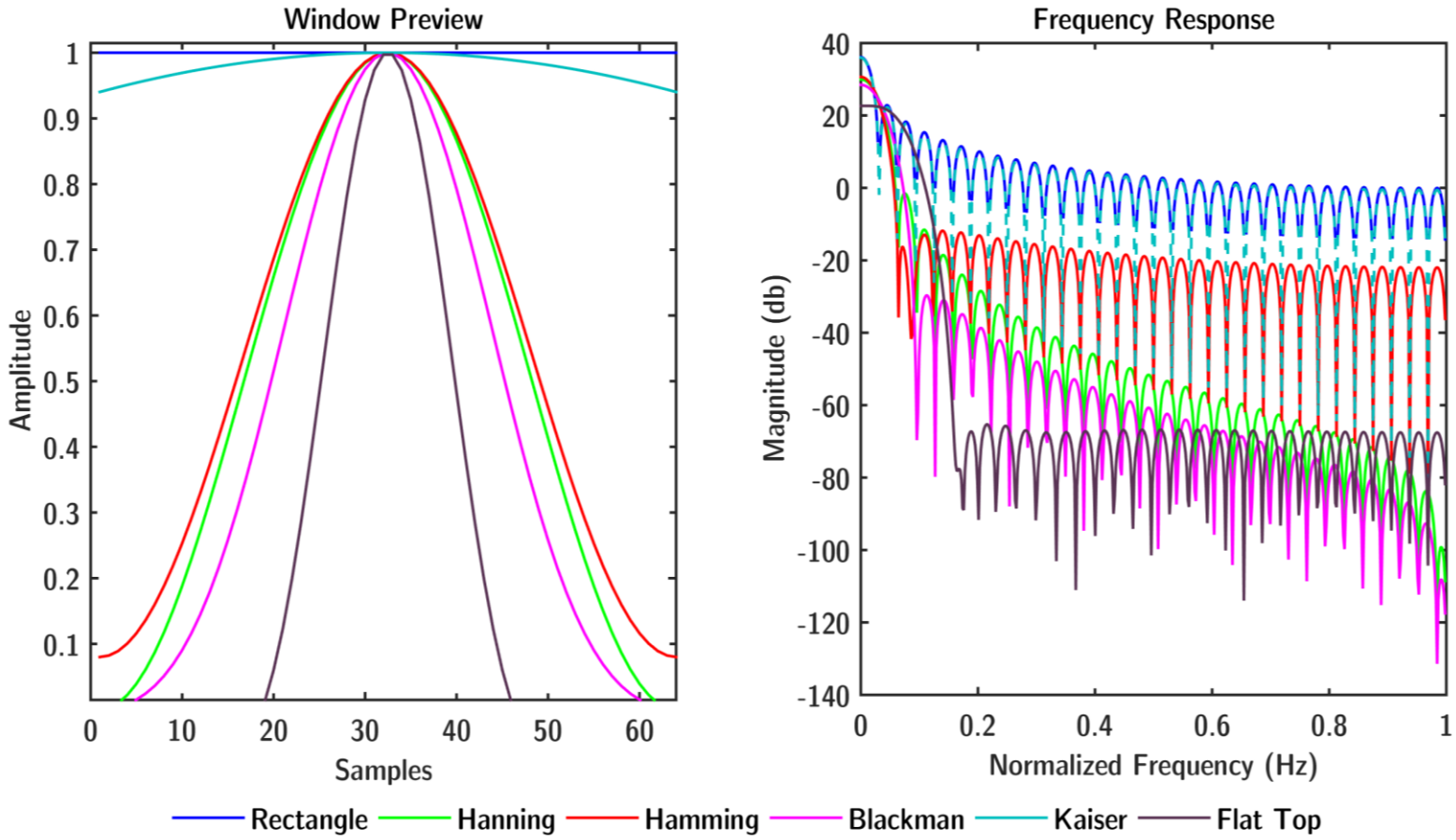
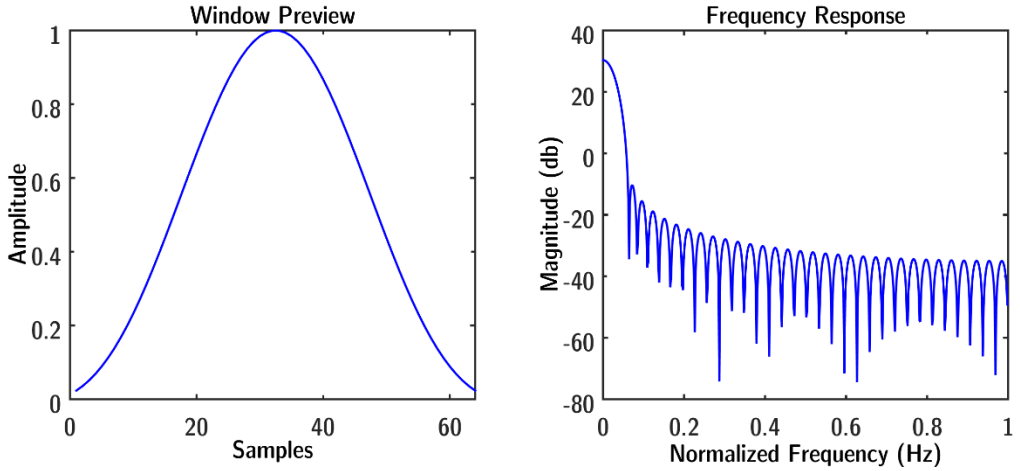
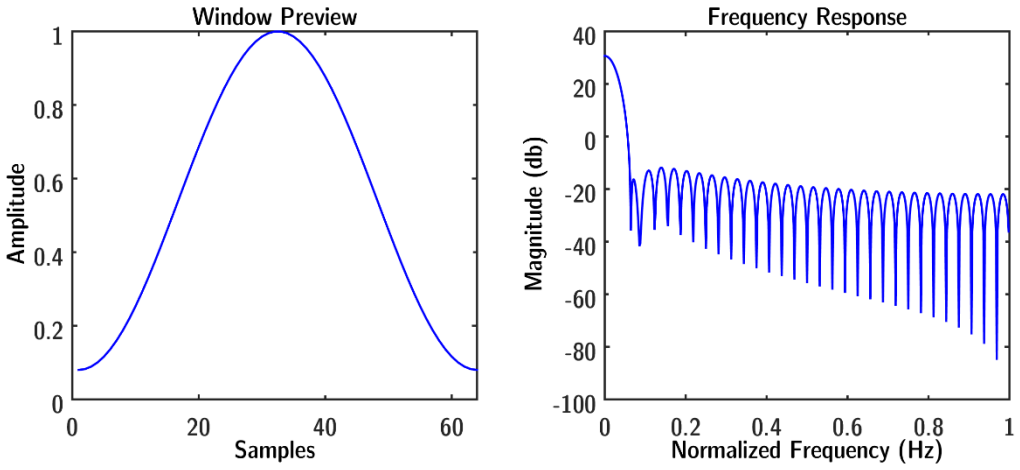


Figure 4.9 *Comparison of Window Functions and Frequency Responses*

The Kaiser window shown in the overlapping cyan to the dark blue was initially found to be similar to the rectangle window with the size for side lobes set at 2.5, but upon adjustment of the parameter to 5.5 it was found to be close to the Hamming window's frequency attenuation *(shown in Fig 4.10)*. The drop-off of the second most side lobe was also found to be faster and more stable in comparison to the Hamming window hence, the Kaiser window with a scale of 5.5 was selected for the design.



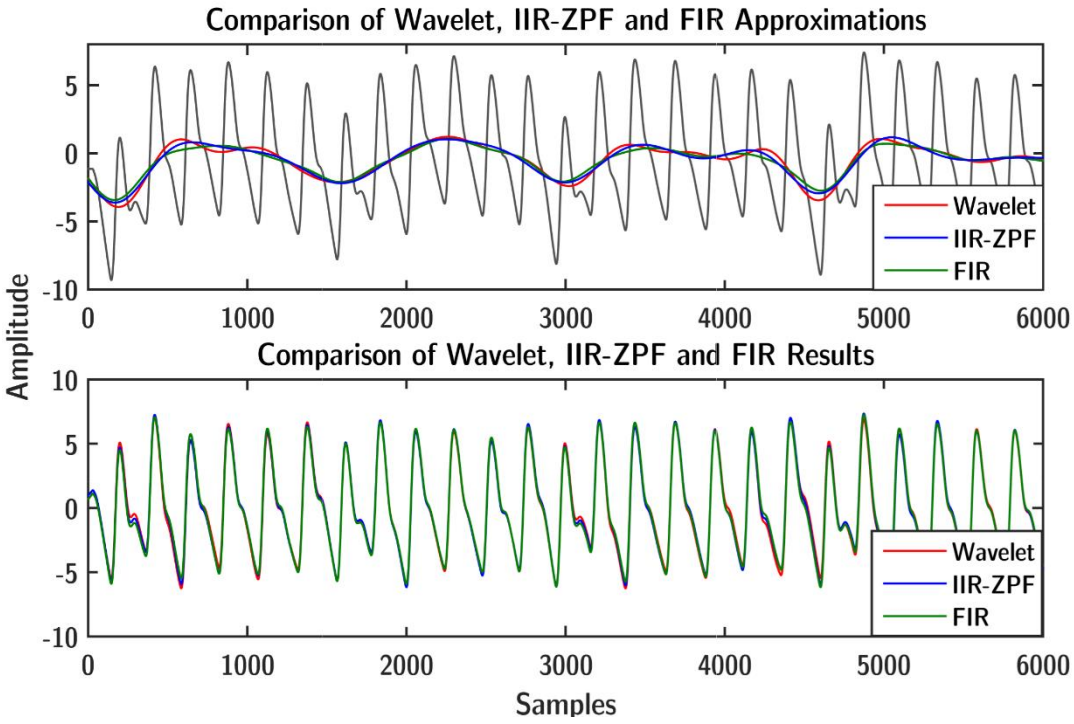
(a). *Kaiser window with a scale of 5.5.*



(b). *Hamming Window.*

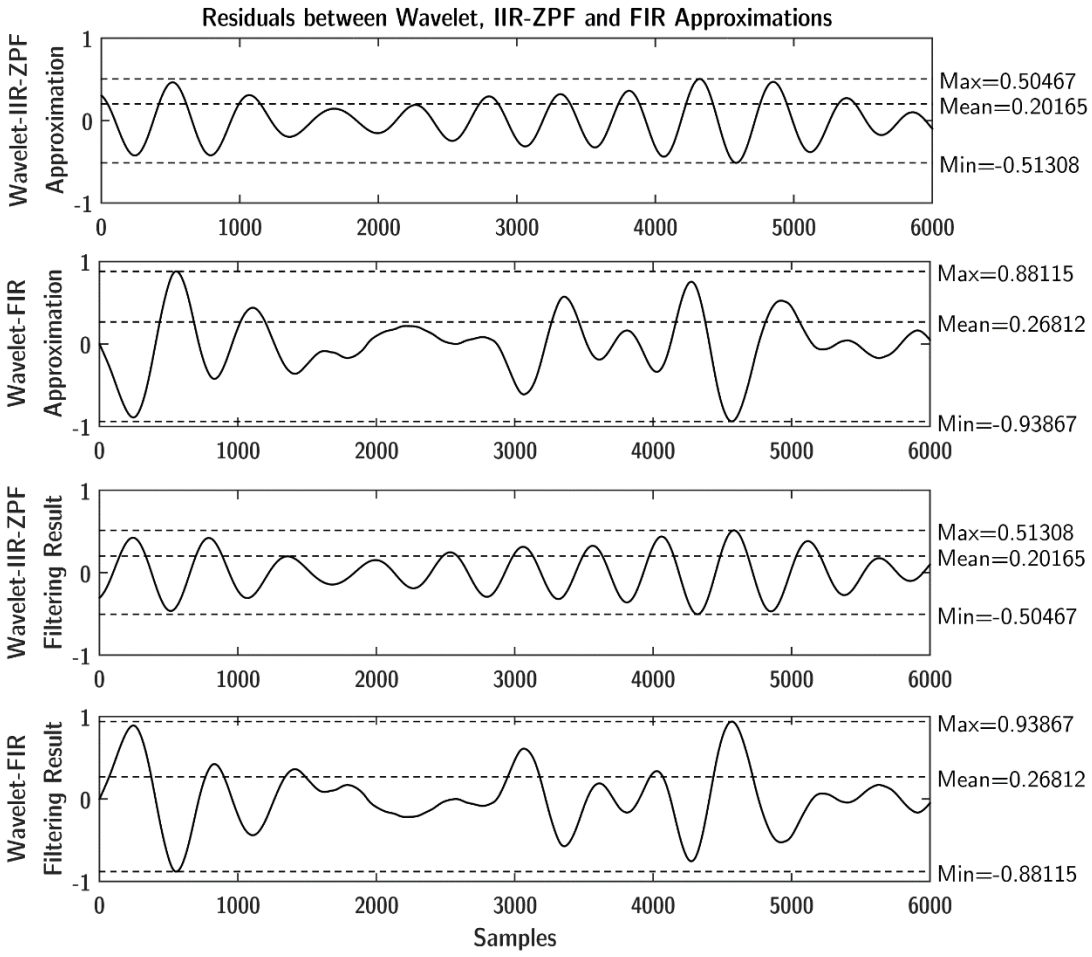
Figure 4.10 *Kaiser Window at 5.5 Side-lobe scale comparison to Hamming Window.*

For the filter design criteria, we ideally want a filter that has the lowest number of taps possible, while being able to achieve the 0.5Hz cut-off frequency required. We were able to obtain the most computationally optimal filter required through a process of trial and error in adjusting the design parameters. Our final filter utilized a cut-off of 0.2Hz with 650 taps to achieve the 0.5Hz cut-off required, and was able to achieve a moderate delay of close to a second. On testing sample 0028_8min with a short test *(shown in Fig 4.11),* the results of the test were compared to the Wavelet filter and the IIR-ZPF.



(a). *Overlap comparison of the filter outputs*

Figure 4.11 *Comparison filter outputs*



(b). *Residual differences between FIR and IIR-ZPF against the Wavelet filter*

The analysis of the results shows that the IIR-ZPF was able to achieve a slightly closer and more precise approximation to that of the Wavelet than the FIR filter. But overall, there wasn't much of a difference *(shown in Fig 4.11b within Residual tests)* between the methods' approximations to make a visibly significant impact on the end result. Parsing the remainder of the R.B.D. samples, the average results *(listed in Table 4.4.)* indicated performance wise that the FIR filter had more than twice the computation time as the IIR Zero-phase filter. Despite this, these computation times were overall still significantly lower than the Wavelet filter for both short and long tests.

The additional computation time from the FIR filter in comparison to the IIR-ZPF filter was because of the amount of calculations required in order to process the lengthy filter kernel.

Table 4.4 *Comparison of the timing results for the IIR-ZPF, Wavelet filter and FIR filter*

	IIR-ZPF	**Wavelet**	**FIR**
Short Test	0.000578	0.02828	0.001354
Long Test	0.001584	0.04832	0.003381

In terms of correlation and error, the FIR filter was able to produce results *(shown in Table 4.5)* that were on-par with the IIR-ZPF filter and in some cases better, however, the filter suffers the same limitations as the IIR-ZPF due to usage of cut-off frequencies *(indicated by Fig 4.12)*. For the majority of the R.B.D. samples, the FIR filter was found to be equally effective to the IIR-ZPF filter, achieving high correlation rates and relatively high correlation of around 97-98%, and low errors of around 0.386-0.484 RMSE similarly to the IIR-ZPF.

Table 4.5 *Average correlation and error tests for the R.B.D. samples overall as an average*

	Test	**Approximation Correlation**	**Approximation RMSE**	**Result Correlation**	**Result RMSE**
IIR-ZPF	**Short Test**	94.82%	0.425	97.95%	0.439
	Long Test	94.91%	0.465	97.88%	0.465
FIR	**Short Test**	95.55%	0.375	98.62%	0.386
	Long Test	94.95%	0.484	97.83%	0.484

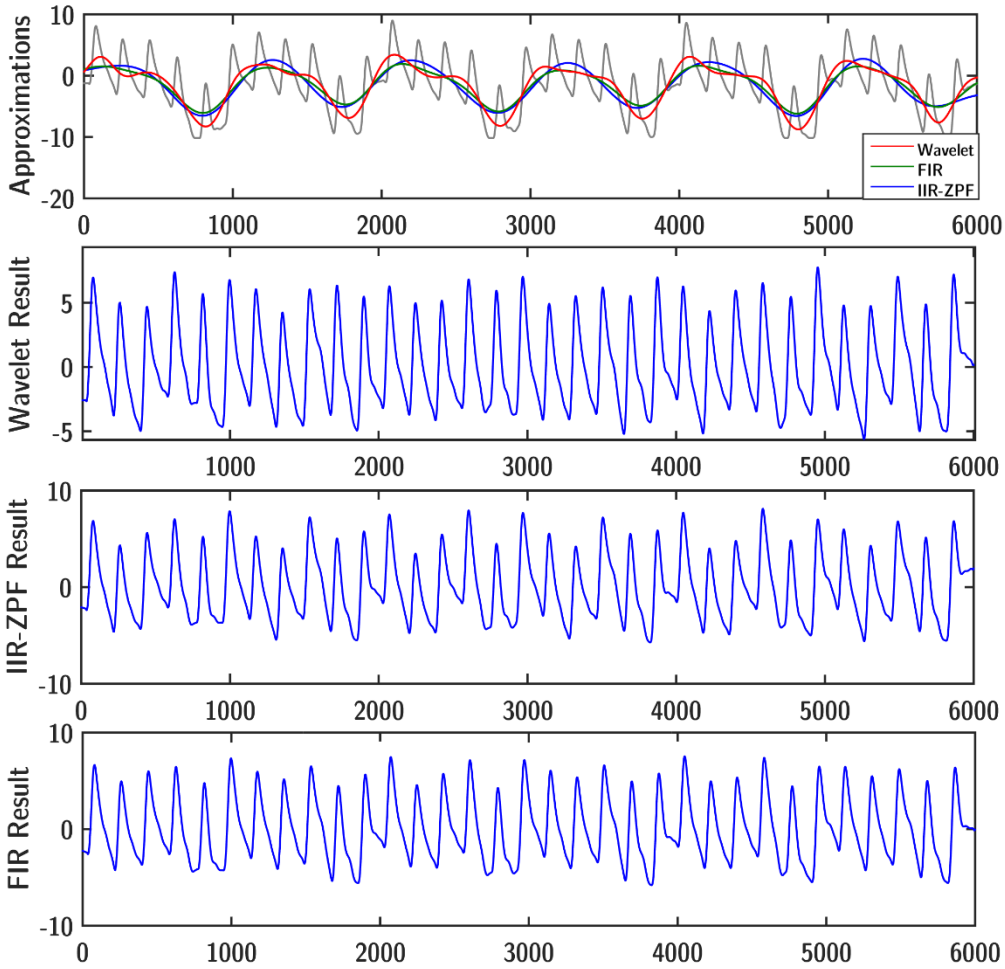


Figure 4.12 *Comparison of the various methods on sample 0009_8min, showing the cut-off problems with the IIR-ZPF and the FIR filter in comparison to the wavelet filter at scale 8.*

The FIR filter from the testing has shown that it can be just as effective as the Wavelet and IIR-ZPF with the right design, but the computation times for the FIR filter make it an undesirable alternative when compared to IIR-ZPF. With that being said, there are optimization techniques available to improve the computation efficiency which warrant further investigation.

4.4.1. Reducing the FIR filter time by FFT Convolution

FIR filtering possesses causality which is important for real-time applications but at the same time, it can also be computationally demanding given long signals and long filter lengths which often counters its real-time plausibility. As discussed earlier, through the application of the FFT and the property of the convolution theorem, it is possible to reduce the amount of computation required by filtering within the frequency domain as opposed to the time domain.

We tested this concept by applying the method onto the R.B.D. samples, where the filter kernel and the samples taken at specified lengths the short and long tests were transformed and filtered via FFT convolution. The results of the FFT convolution filter is visually almost identical to that of the direct convolution filter *(shown in Figure 4.13)*, achieving about the same correlation and difference errors *(described in Table 4.6)* when compared to the Wavelet filter.

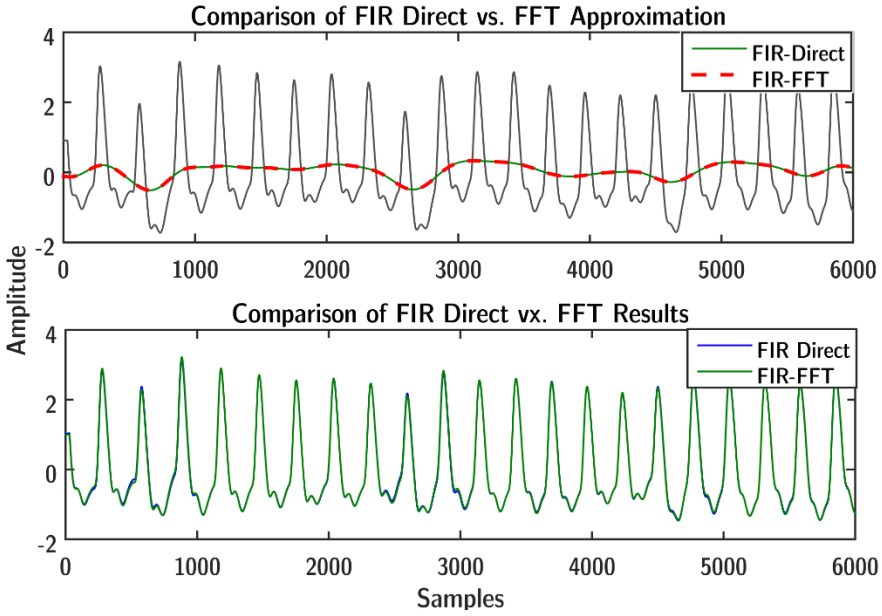


Figure 4.13 *Visual example of the identical results obtained by the FIR-Direct and FIR-FFT based convolution methods in sample 0309_8mins*

Table 4.6 *Average correlation and error tests for the R.B.D. samples of the IIR-ZPF, FIR Direct Convolution filter and FIR-FFT filter to the wavelet filter*

	Test	Approximation Correlation	Approximation RMSE	Result Correlation	Result RMSE
IIR-ZPF	**Short Test**	94.82%	0.425	97.95%	0.439
	Long Test	94.91%	0.465	97.88%	0.465
FIR DIRECT	**Short Test**	95.55%	0.375	98.62%	0.386
	Long Test	94.95%	0.484	97.83%	0.484
FIR-FFT	**Short Test**	94.34%	0.401	98.45%	0.411
	Long Test	94.89%	0.442	98.31%	0.442

In terms of computation time *(shown in Table 4.7),* the amount was drastically reduced by 76% of the average time required for direct convolution. The computation time rose steeply when the long tests were run, and in comparison to the IIR-ZPF, it was slightly more than ~20% of the IIR-ZPF run-time. Despite this, the timing results were still significantly less in comparison to the Wavelet filter, and was overall somewhat close to the IIR-ZPF indicating the potential efficiency of the method.

Table 4.7 *Comparison of the average timing results for the IIR-ZPF, Wavelet filter, FIR Direct Convolution filter and FIR-FFT filter*

	IIR ZPF	Wavelet	FIR-Direct	FIR-FFT
Short Test	0.000578	0.02828	0.001354	0.000573
Long Test	0.001584	0.04832	0.003381	0.001900

4.4.2. Chunk Processing through the Overlap-Save Algorithm

Through FFT convolution, it is possible to use FIR filters for processing moderate to long lengths of data, provided that both the filter kernel and the length of data are lengthy in nature. The OLS algorithm is an extension that allows the application of this technique to process, and combine lengthy chunks of signal data on the go in real-time. For the implementation of the algorithm, we followed the steps described in the literature review *(illustrated in Chapter 2, Fig 2.4).* The minimum data required for the FIR filter designed previously is 650 samples which forms the basis of the segment. The remaining length of the segment to be filled should be selected for how often the filter processes and outputs information.

We selected 150 samples which is equivalent to half a second delay based on the sampling rate of 300 samples for intermittent processing. From these parameters, there is an overlap of 650 samples and an addition of 150 newer samples. The 150 samples are the non-aliased samples that are taken to form the filtered segment. Running the filter on the R.B.D. samples, the total filtering time was found to be slightly longer than regular direct convolution by 1.5-1.6 times. This was owing to the amount of repetitive processing required by the overlap process, but the average run-time for filtering each segment was approximately 0.000073-0.000076 seconds. This timing result included the FFT of the segment, the convolution filtering and the IFFT of the result. From a real-time processing point of view, where there are samples constantly being collected, the algorithm was shown to be able to operate within real-time constraints by the definition[3].

[3] *68. Kuo, S.M., et al., Real-Time Constraints, in Real-time digital signal processing: fundamentals, implementations and applications. 2013, Wiley: Chichester, West Sussex, UK. p. 15-16. states the definition of a real-time DSP system as "A real-time DSP system demands that the signal processing time must be less than the sampling period, in order to complete the processing before the new sample comes in."*.

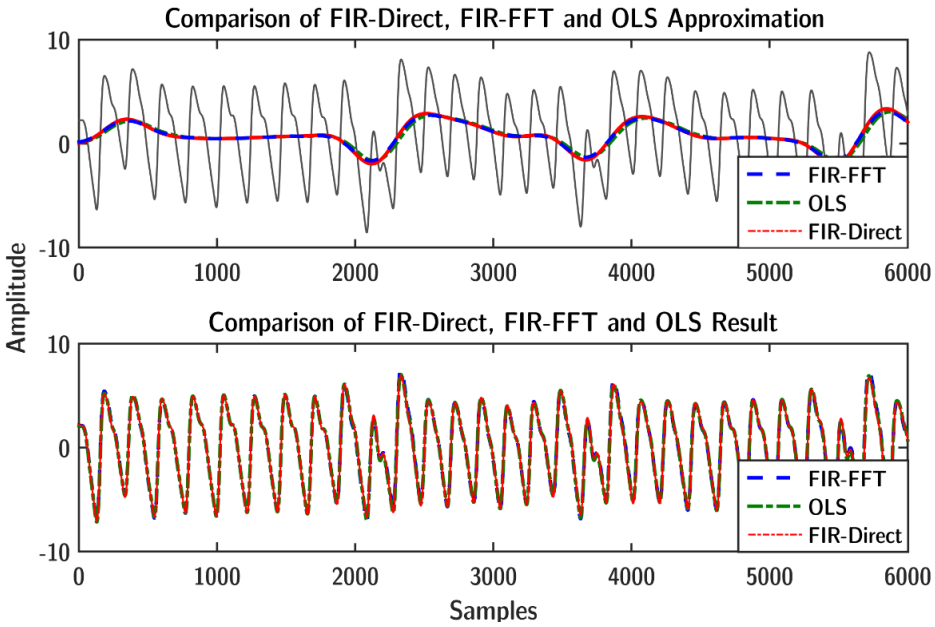
Given the relatively low chunk processing time required for the algorithm, more time-limit flexibility is also gained for other algorithms such as detection and CBSI to be applied in conjunction with the OLS algorithm.

Table 4.8 *Comparison of the total average timing results for the filters*

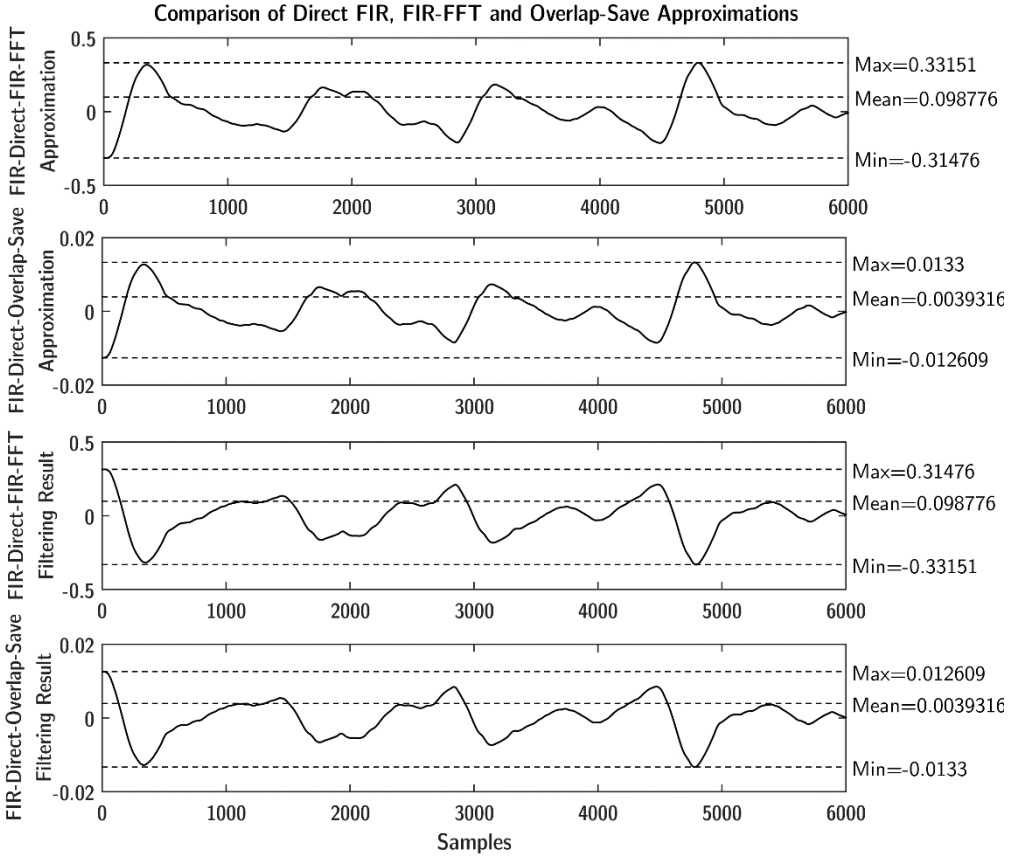
	IIR ZPF	**Wavelet**	**FIR-Direct**	**FIR-FFT**	**OLS (Summed)**	**OLS (Per Segment)**
Short Test	0.000578	0.02828	0.001354	0.000573	0.003427	0.000073
Long Test	0.001584	0.04832	0.003381	0.001529	0.009084	0.000076

In terms of correlation and errors in comparison to the Wavelet filter *(described in Table 4.9),* the OLS algorithm achieved a consistent result akin to that of the FIR-FFT and FIR-Direct filters. There were little to no difference between the three methods visually *(shown in Fig 4.14),* and numerically at around approximately 1-2% difference in correlation. The OLS algorithm having being based on the FIR filter, however, still suffers from the same limitations in regards to the cut-off frequency range. This led to a decrease in correlation when the OLS algorithm was applied to sample 0009_8min.

Through the application of chunk processing and FFT-based convolution through the OLS algorithm, we were able to show that it is possible to use FIR filters efficiently and effectively. Given the relatively low chunk processing time required for the algorithm, more time-limit flexibility is also gained for other algorithms such as detection and CBSI to be applied in conjunction with the OLS algorithm.



(a). *Overlap comparison of filtering methods*



(b). *Residual differences between filtering methods*

Figure 4.14 *Visual comparison of the FIR-Direct, FIR-FFT and Overlap-Save algorithm approximations and results for sample 0032_8min as an example*

	Test	Approximation Correlation	Approximation RMSE	Result Correlation	Result RMSE
IIR-ZPF	**Short**	94.82%	0.425	97.95%	0.439
	Long Test	94.91%	0.465	97.88%	0.465
FIR DIRE	**Short**	95.55%	0.375	98.62%	0.386
	Long	94.95%	0.484	97.83%	0.484
FIR-FFT	**Short**	94.34%	0.401	98.45%	0.411
	Long	94.89%	0.442	98.31%	0.442
Overlap Save	**Short**	95.56%	0.258	99.45%	0.270
	Long	96.24%	0.412	98.50%	0.412

Table 4.9 *Comparison of the average correlation and difference errors of the IIR-ZPF, FIR-Direct, FIR-FFT and Overlap-Save algorithm to the Wavelet filter approximations*

4.5. Summary

In this chapter, we have explored the effects of the FIR filter in comparison to the wavelet filter and the IIR-ZPF. From the analysis it has been shown that the FIR filter could be equally effective as the Wavelet filter and the IIR-ZPF, however, the FIR filter suffers from a lengthy kernel which increased the computational power and time required. The wavelet filter has been found to produce optimal results when a correct scale was selected, but the results aren't significantly different in comparison to the cut-off filters.

Despite the claims in the literature of over-approximation, we have found that the IIR-ZPF and FIR filters were able to perform adequately. The latter at times, was found to perform on par with the wavelet filter under certain circumstances when the drift was found to be contained within cut-off range. Throughout the filtering process, we have found no significant distortions to morphologies with the correct filter parameters but furthermore, we found evidence that softening of the drift will aid the visibility and identifiability of features.

Both types of cut-off filters were able to successfully soften the baseline drift to make features more identifiable. From further investigations of the FIR filter, we have explored and confirmed that the FIR filter can potentially be utilized within real-time constraints, through the application of frequency domain filtering and chunk processing via the OLS algorithm. From this, a foundation for further development of the proposed cascaded filter algorithm is established.

Chapter 5

A Review of the Algorithms for Detecting and Identifying Features within the PPG Signal

Chapter 5 analyses the selected feature detection algorithms in order to evaluate their effectiveness and computational efficiency. From the analysis, the algorithms are compared amongst one another in order to identify a suitable algorithm that can be used in the development of the proposed cascaded filter. For the comparison of algorithms, its structure, a low computational runtime and high detection rates are considered as key deciding factors.

5.1. Experimental Setup and Procedure

Standard testing procedures were employed where a short test was applied first with 20 seconds of data to investigate how the algorithm performs initially, and the results were verified through long tests with 60 seconds of data for the goal of assessing possible degradation in algorithm performance and stability. The R.B.D samples prior to being parsed by the algorithm were filtered using the FIR filter derived in Chapter 5, and a 30 point moving average was applied to remove any small discrepancies that may exist within the signal.

Sample 0028_8min with a short test is used as an initial means of calibrating the algorithms. The algorithms were implemented following their respective authors instructions. With this, they were modified where necessary in order to ensure optimum detection rates, while attempting to preserve the algorithm's original ideas and concept. We based the analysis of the algorithms on computational time and the ability to detect features while adapting to faults.

Visual analysis of the results was utilized in order to affirm the detected features. We used the AAMI standards of detection rate and sensitivity *(described as Eq. 2.8 and 2.8 in Chapter 2)* as a method to numerically benchmark the algorithms' performance. Algorithm efficiency and complexity is analysed using the Big-O notation and the arithmetic operations utilized by the algorithm *(described in Appendix D).* The results of the AAMI tests can be found listed within Appendix C, and the MATLAB code for the implementation can be found within the provided repository online[4] under the *"chapter_five"* folder.

As some of the authors provided no definitions for what a peak and a valley may be, we defined peaks and valleys as follows:

$$\boldsymbol{isPeak} = \begin{cases} x[n] > x[n-1]\ AND\ x[n+1] \leq x[n] & ,true \\ Otherwise & ,false \end{cases} \qquad (Eq.\ 5.1)$$

$$\boldsymbol{isValley} = \begin{cases} x[n] < x[n-1] AND\ x[n+1] \geq x[n] & ,true \\ Otherwise & ,false \end{cases} \qquad (Eq.\ 5.2)$$

With x[n] is the signal samples stored in an array, the first condition in the latter equations' first term determines whether the point is a peak or a valley through the slope of its points. The second condition of the equations' first term resolves the issues of plateaus *(illustrated in Fig. 5.1)* existing within the signal, where the start of the plateau is considered as the valley instead of its end or middle as it correlates the beginning of systolic activity.

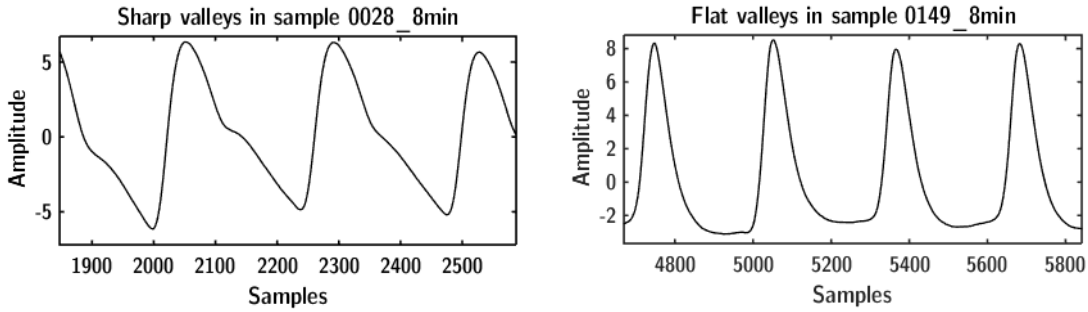


Figure 5.1. *Examples of plateaus found in sample 0149_8min vs. normal sharp valleys found in sample 0028_8min*

[4] https://bitbucket.org/nthegestalt/dissertation-code

5.1.1. Derivative Calculation Based Discrimination Algorithm by Xu et al. 2007's Algorithm

The Derivative Calculation Based Discrimination (DCBD) algorithm by *Xu et al. 2007 [43]* as part of their cascaded filter utilized a derivative based method *(illustrated in Fig 5.2)* in order to discriminate and identify the valleys. Due to its structure it can also be extended to detect and discriminate peaks as well.

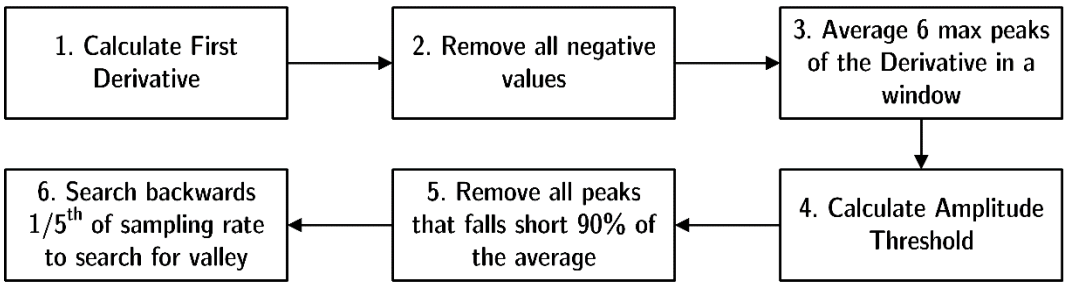


Figure 5.2 *(Xu et al., 2007)'s algorithm process outline*

We utilized the conditions established *(Eq. 5.1 and 5.2)* earlier in order to identify peaks and valleys from ordinary signal points. Implementing this algorithm we encountered a series of problems. The first was that the threshold for discriminating the derivative peaks was too high, being set at 90% of the 6 tallest derivative peaks within the signal. As a result detection within signals with varying perfusion, *(shown in Fig 5.3)* resulted in misdetections since the algorithm removed any derivative peaks that was underneath the threshold.

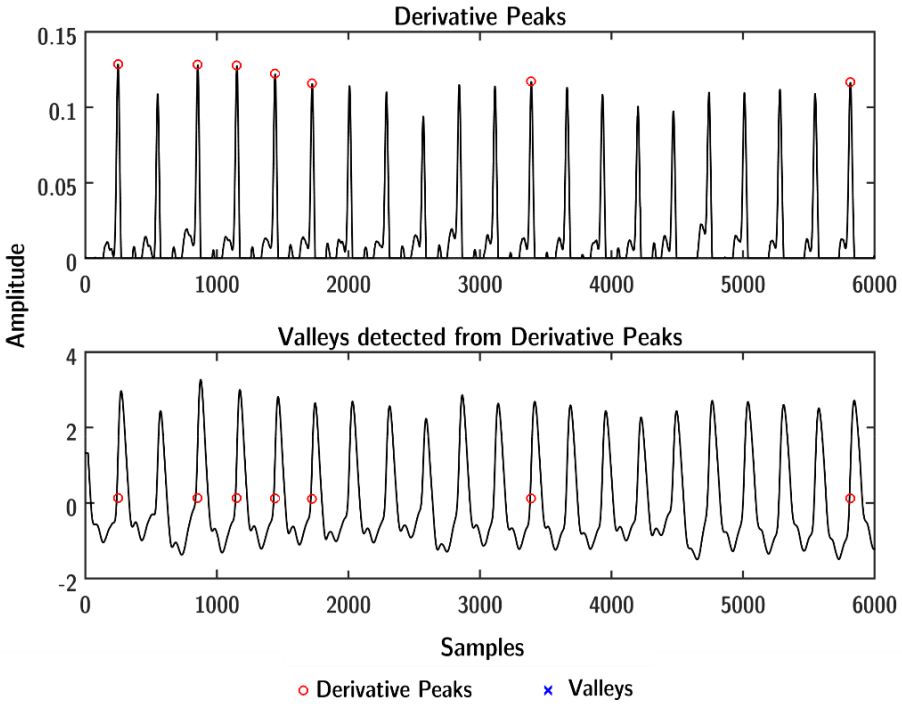


Figure 5.3 *Derivative peak misdetections with size threshold at 90% in sample 0309_8min*

We reduced this value to around 75% to make the algorithm more lenient resolving the issue, but encountered problems with the search back distance; the algorithm would exit before finding the valley due to reaching the limit 20 samples, which was $1/5^{th}$ of the sampling rate as suggested by the authors *(shown in Fig 5.4)*.

Increasing the number of points to 170 promptly resolved the issue however, we encountered problems where there was a notch which threw off the valley search. To overcome this problem, we have the algorithm search back further, based on the remaining search-back length which was not reached within the initial search *(shown in Fig 5.5),* in order to find if there was a valley with a lower value. If such a valley exists, then the detected valley is replaced with the newer one.

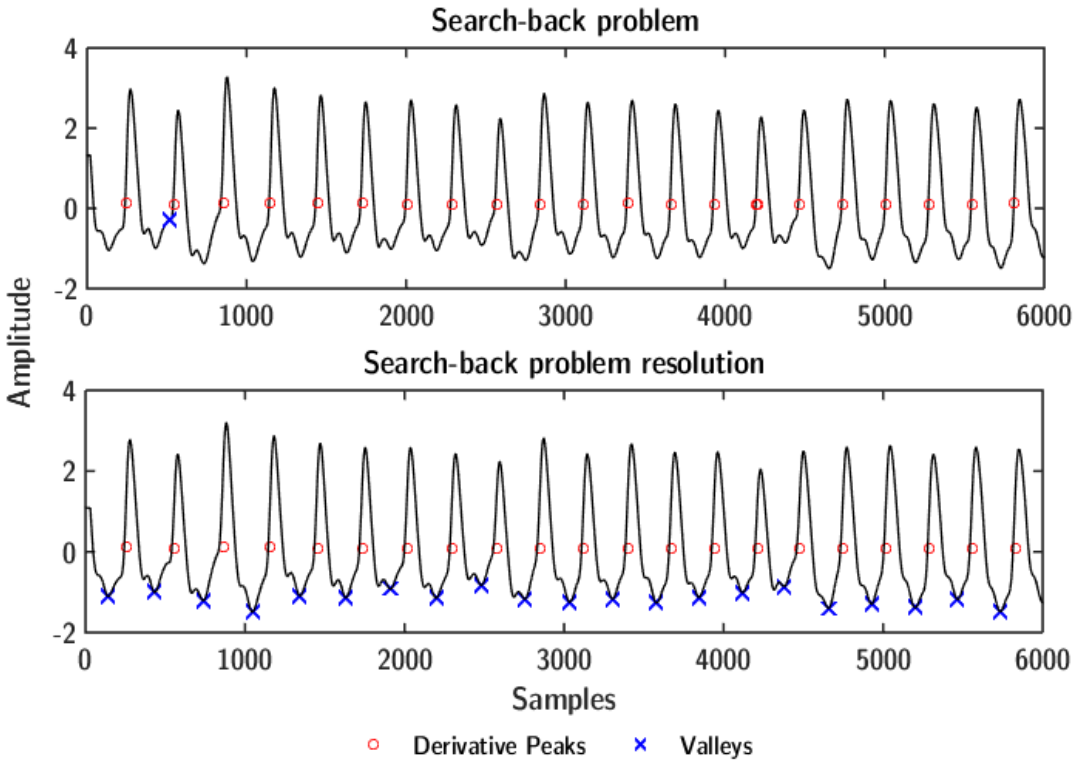


Figure 5.4 *Search-back limit problem before and after resolution*

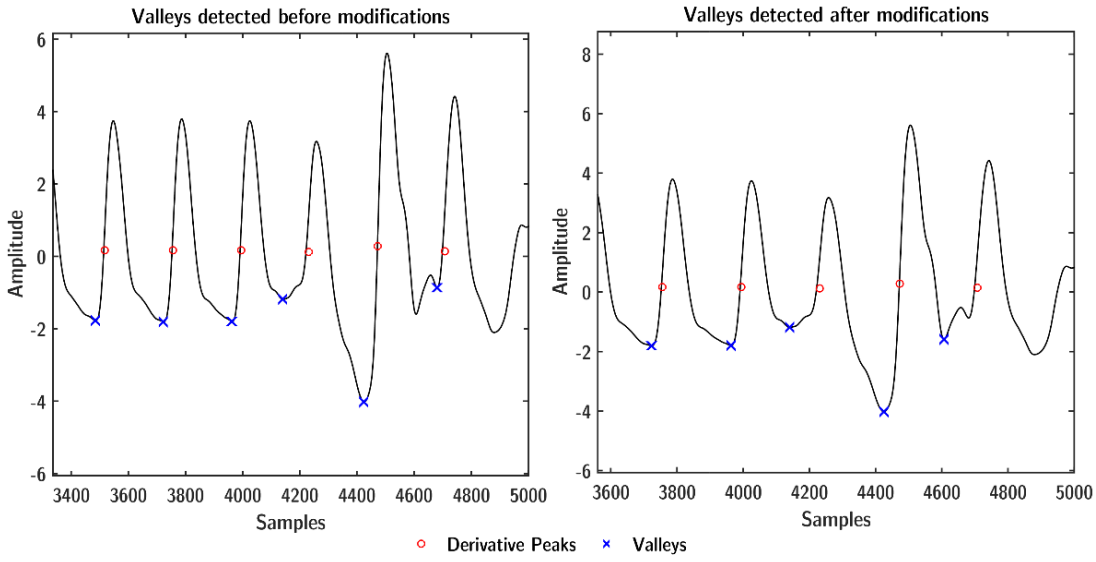


Figure 5.5 *Search-back verification problem before and after resolution*

5.1.2. Spectral Based Discrimination Algorithm by Kan et al. 2012's Algorithm

There are a total of two phases to the Spectral Based Discrimination (SBD) algorithm presented by *Kan et al. 2012 [40] (illustrated in Fig 5.6 a and b)*. The first is the calibration phase where the algorithm parameters are initialised, and the second is the detection phase where the algorithm discriminates using the values obtained from the calibration phase. The calibration phase was built on total of two main parameters which are used to establish the rest of the algorithm's sub-parameters. They are described as being a greater modifier value L_1 and a lesser modifier value L_2 defined as:

$$L_1 = f_b \times f_s \qquad \textit{(Eq. 5.5)}$$

$$L_2 = 0.5 \times L_1 \qquad \textit{(Eq. 5.6)}$$

Where:

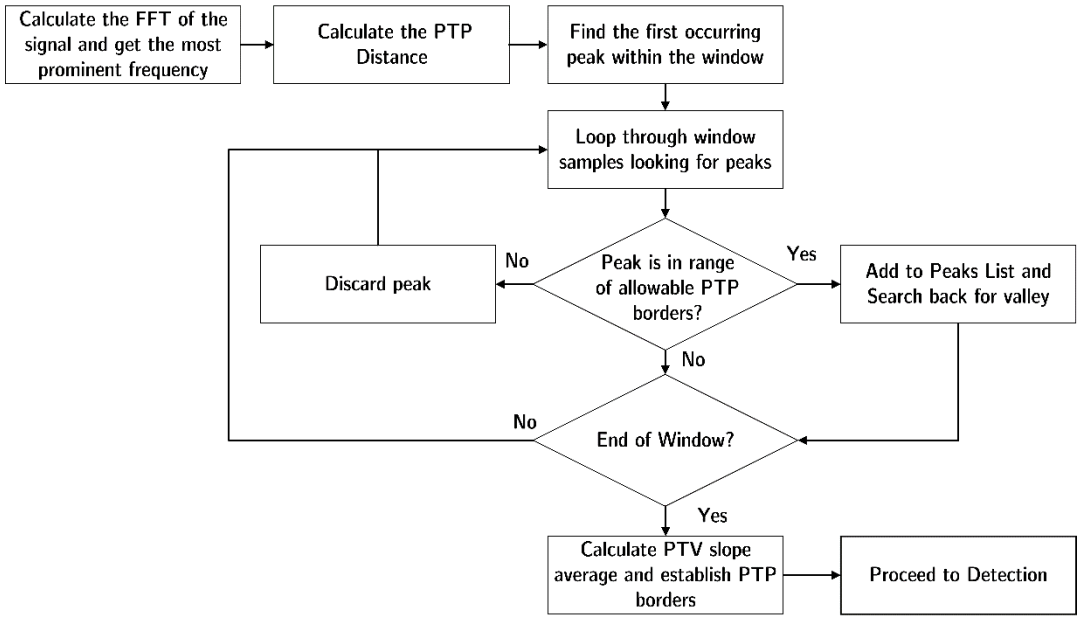
- f_b = The frequency with the highest amplitude from the signal's FFT, corresponding to the systolic peak frequency.
- $f_s =$ The sampling rate at which the signal was recorded.

These two are used to define the "*Peak-to-Peak*" (PTP) search window, the *"Backwards search window* " (BKS) . The borders for the PTP window are responsible for discriminating peaks via distance, are defined by a lower boundary PTP_{Lower} and an upper boundary PTP_{Upper} calculated as:

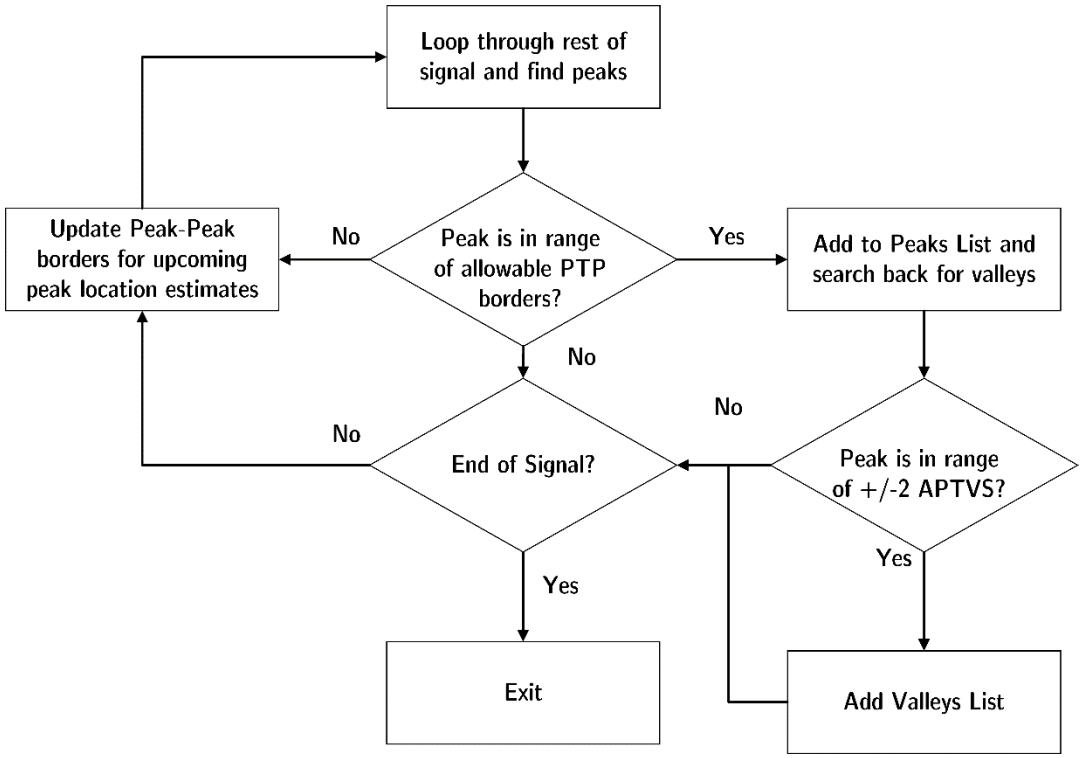
$$PTP_{Lower} = C_k + L_2 \qquad \textit{(Eq. 5.7)}$$

$$PTP_{Upper} = C_k + L_1 \qquad \textit{(Eq. 5.8)}$$

Where C_k is the location of the first peak detected within the window, any peaks that existed outside of the PTP range are considered an error and therefore ignored.



(a). *Calibration Routine.*



(b). *Detection Routine*

Figure 5.6 *Outline for Kan et al. 2012's SBD Algorithm.*

The backwards search *(BKS)* window determines how far from the peak the algorithm should search back in order to identify the associated valley. It is defined with a lower boundary BKS_{Lower} *(calculated by Eq. 5.9)* and an upper boundary BKS_{Upper} *(calculated by Eq. 5.10)*.

$$BKS_{Lower} = C_k - L_2 \qquad \textit{(Eq. 5.9)}$$

$$BKS_{Upper} = C_k \qquad \textit{(Eq. 5.10)}$$

Since there was no specified location for where to apply the window other than *"randomly",* the calibration window was set to be taken from the start of the signal. The result of the calibration phase was a size discriminator threshold described as the average *"Peak-To-Valley"* slope (*APTVS*, *calculated by Eq. 5.11)*. Future PTV pair slopes that do not fall within the author's described allowable distance of ∓ 2 points from the *APTVS* are discarded as false features.

$$APTVS = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{Peak(i)_x - Valley(i)_x}{Peak(i)_y - Valley(i)_y}\right] \qquad \textit{(Eq. 5.11)}$$

On an initial test using sample 0028_8min we found that there were problems *(shown in Fig 5.7)* in regards to using the first peak found within the calibration window. Since there was no way for the algorithm to know which peak was real or fake, it utilized a false peak which led to the skewing of the APTVS, resulting in errors within the later detection process.

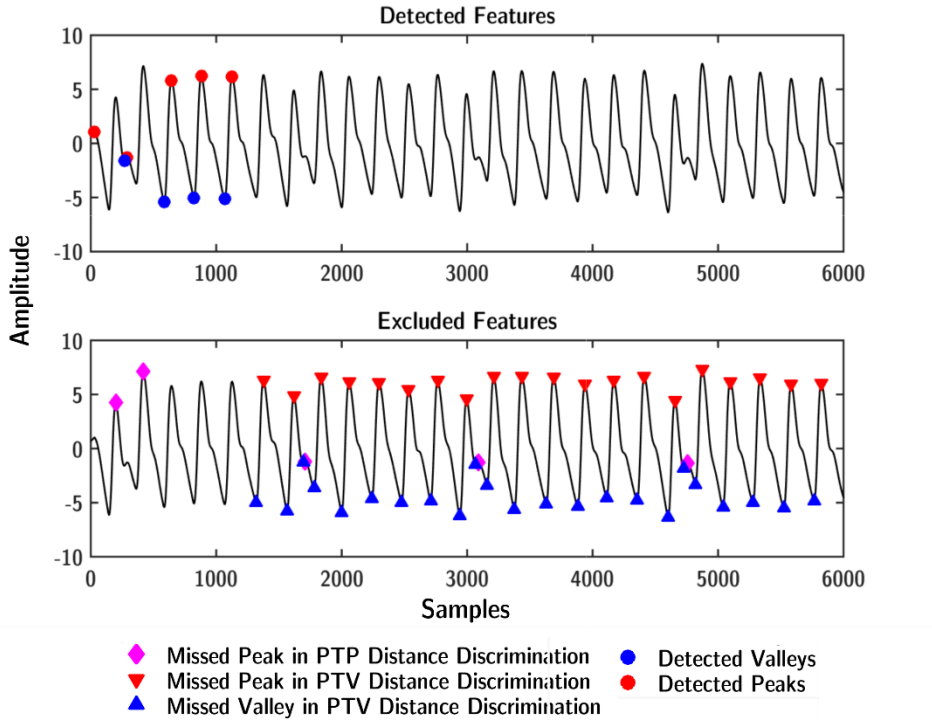


Figure 5.7. *Misdetection caused by APTVS skewing due to algorithm using wrong peak*

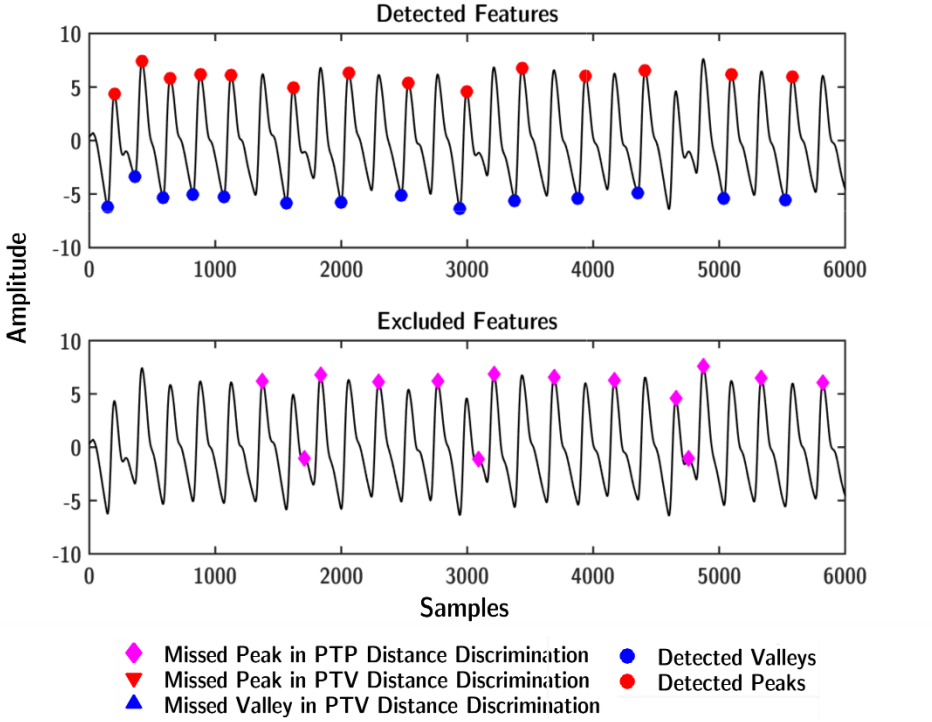


Figure 5.8 *PTP distance errors after calibration attributed to variations*

To rectify this we proposed a simple calibration scheme *(illustrated in Fig 5.8)*; the algorithm would search for all peaks and valleys within the window as specified by the outline, and from this proceeded to calculate the slope for each and every PTV pair. From the slopes, the largest slope was taken to be a discriminator threshold where every other slope calculated thus far is compared to. We kept any PTV pairs with a slope that fell within the range of at least 75% of the maximum PTV slope.

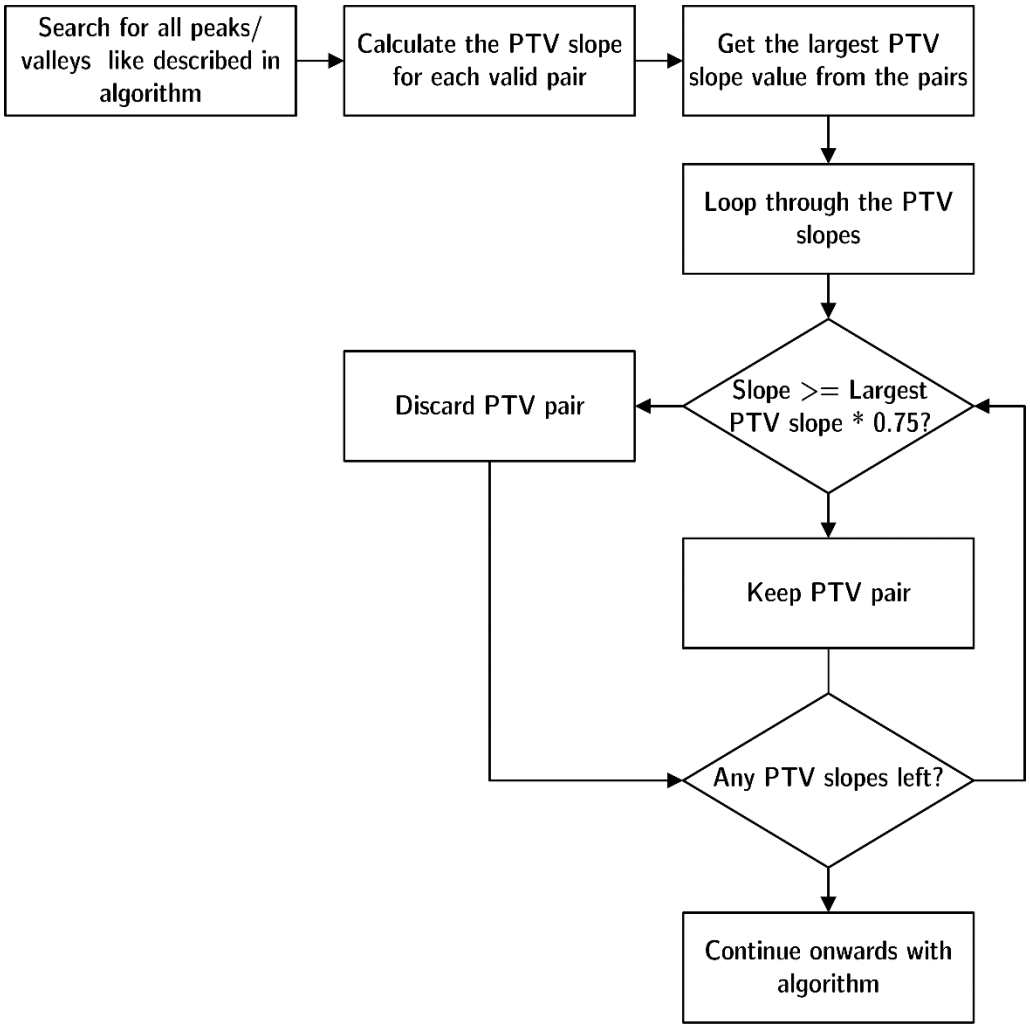
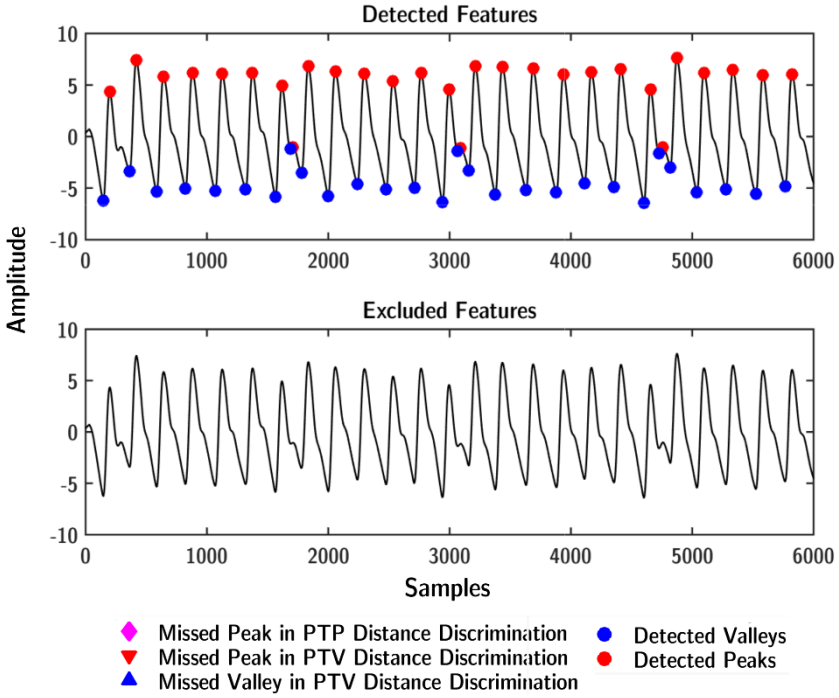


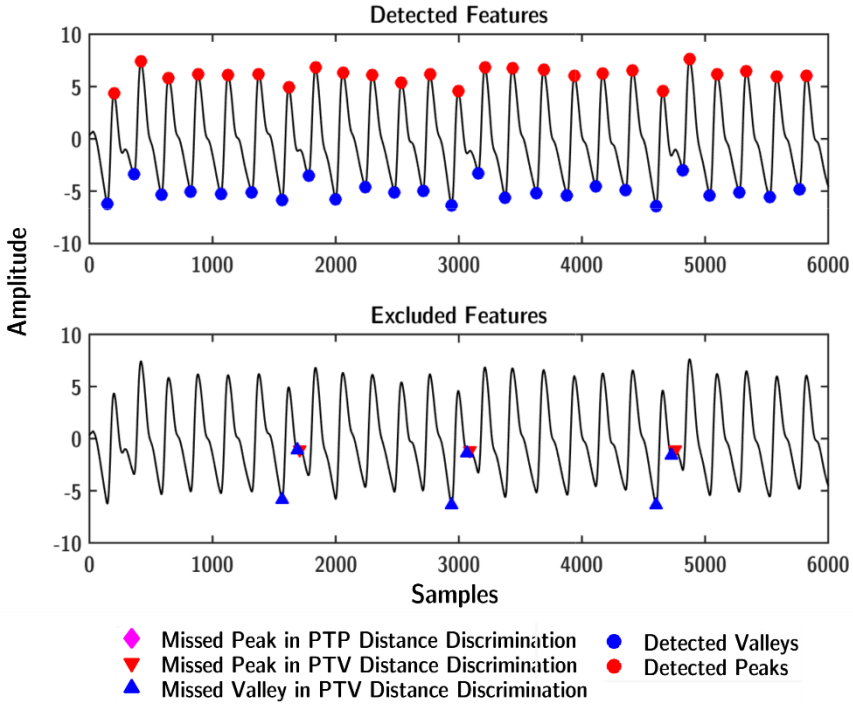
Figure 5.9. *Outline of the proposed calibration scheme for the SBD algorithm by Kan et al. 2012*

Implementing this yielded successful results however, there were still errors in regards to PTP discrimination. The PTP discrimination error may be attributed to the fact that cardiac activity varies over time as well with physiological changes. As a result, the frequency of the systolic peaks may shift and in turn, cause slight amounts of displacement within the time domain. Due to this, the distance calculated from the FFT's approximation of the systolic peak frequencies may not be completely accurate. The value of PTP_{Lower} was lowered to 75% of the original value which resolved the problems for the most part, but issues with the $APTVS$ issue still remained, where all PTV pairs was being detected regardless of slope*(shown in Fig 5.10 a)*.

The analysis of the problem determined that the cause was due to the $APTVS$ being too broad so as a result, the discrimination process was either too lax or too strict. In order to rectify this, we replace the ∓ 2 point threshold with that or a percentage threshold based on the $APTVS$ value. The values we utilized initially was 75% for the lower bound, and 200% for the upper bound which resolved the problems *(shown in Fig 5.10 b)*, but we also found that these ratios required adjustment over the course of testing, in order to produce optimal detection rates for each individual sample.



. ***(a).*** *The problem with using ∓2 points discrimination.*



(b). *The problem resolved using percentage discrimination.*

Figure 5.10. *Misdetection caused by APTVS range being too broad.*

5.1.3. Adaptive Threshold Detection Algorithm

The Adaptive Threshold Detection *(ADT)* algorithm by *Shin et al. 2009 [63]* utilizes the variation of the signal in order to guide the detection process *(outlined in Fig 5.9).* A slope line based on the variance of the signal is used as a guiding reference. This slope line was updated continuously as new signal samples are acquired, where intersection checks was also performed with the main signal line continuously. The search for the signal establishes a period in the algorithm described as the *"in-sensing"* period. The slopes are initialised as equation 5.12 and updated using equation 5.13 respectively.

$$Slope_{init} = \begin{cases} 0.2 \times \max(signal), & V_{\text{Max}\,detection} \\ 0.2 \times \min(signal), & V_{\text{Min}\,detection} \end{cases} \qquad \textit{(Eq. 5.12)}$$

$$Slope_k = Slope_{k-1} + S_r \frac{(V_{n-1} + std_{signal})}{F_s} \qquad \textit{(Eq. 5.13)}$$

Where:

- $Slope_k$ = The amplitude of the current or previous slope at k, the current sample.
- S_r = The slope change rate, selected as -0.6 and 0.6 for max detection and min detection respectively.
- V_{n-1} = The previous peak amplitude if it exists.
- std_{signal} = The current standard deviation of the signal.
- F_s = Sampling Frequency of the signal.

On collision with the signal line the algorithm enters an *"out-sensing period"*, where the slope is set to the values of the signal line so it *"climbs"* it to search for inflection points. Upon the detection of the inflection points the algorithm proceeds to exit the *"out-sensing period",* and enters the *"in-sensing period"* once again until another collision occurs.

As the authors gave no means to identify inflection points, equations 5.3 and 5.4 were utilized. For the intersection of the lines we considered a standard geometric algorithm as described by *Cormen 2001 [69],* which utilizes an orientation test in order to determine the intersection of line segments.

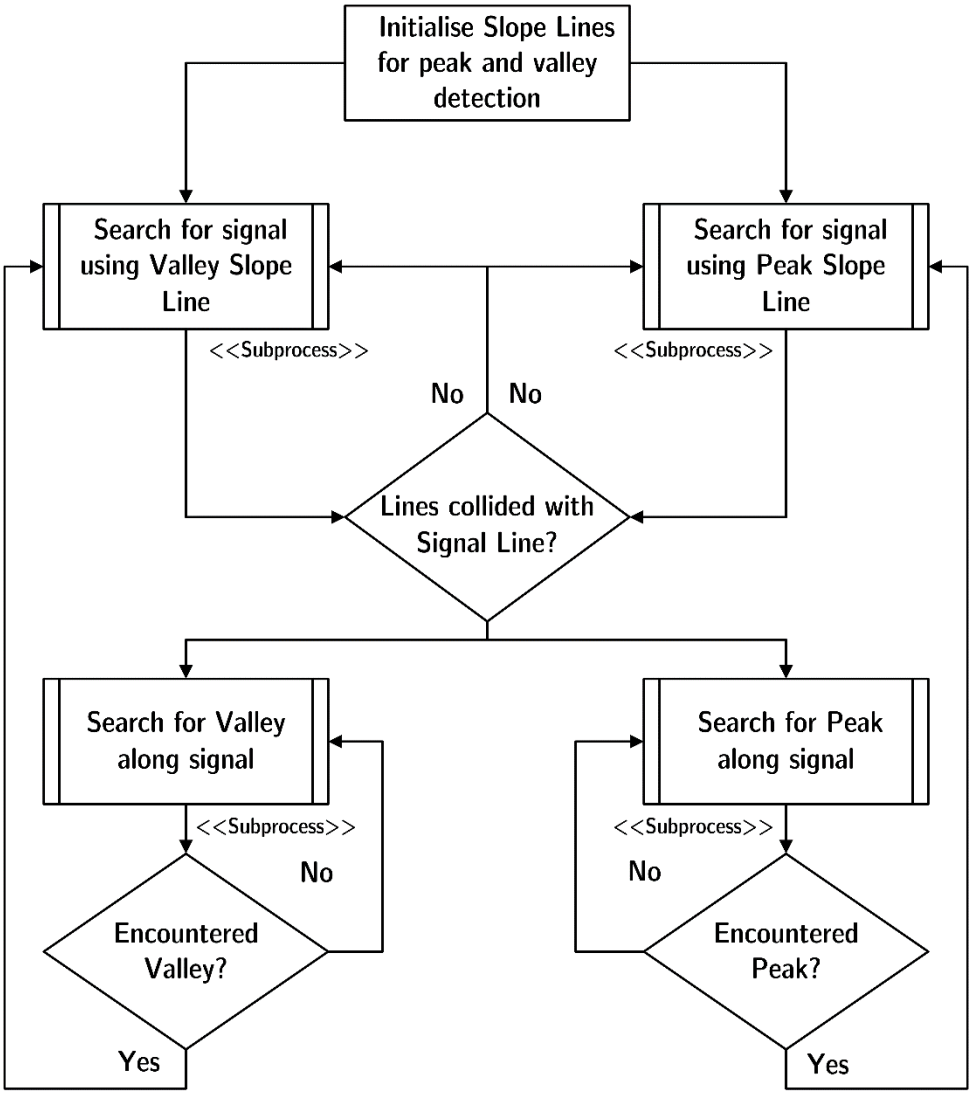


Figure 5.11 *Outline of the ADT algorithm*

An initial run on sample 0028_8min showed that it was able to identify all the peaks but not the valleys *(shown in Fig 5.13)*. On examination, it was found that the slope calculation for the min detection, had become the max slope seeking line due a calculation error with the signal amplitude's negative elements.

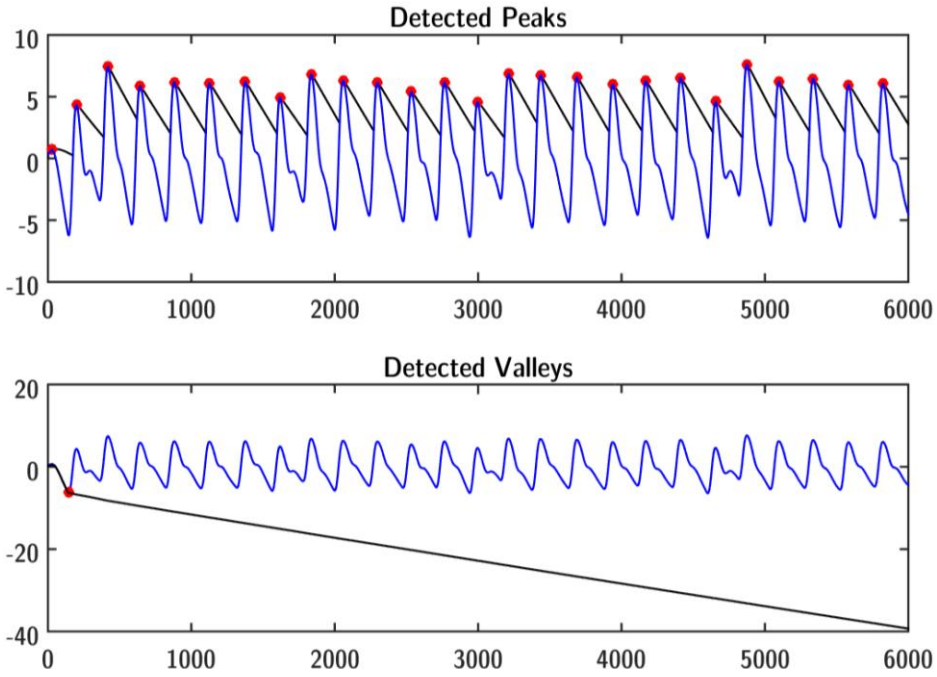


Figure 5.12 *Valley slope line amplitude errors on initial testing*

As a countermeasure, the signal was boosted to be above 0 by adding the value of the lowest signal amplitude to the remainder of the signal to cancel out the negative amplitudes. Re-running the tests yielded results with a good portion of the valleys now detected, but overall some valleys were still missing *(shown in Fig 5.12 a).* We traced the cause to that of the slope was not able to climb fast enough to accommodate the extreme signal amplitude variations.

The change rate S_r was then subsequently increased to 1.5 in order to obtain a better climb rate which proved to be successful *(shown in Fig 5.12 b)*. But even with this change, it was found to be inadequate with the sudden variations in drifts and perfusion levels found in that of sample 0030_8min, a problem that was indicated by the authors in their studies *(indicated by Fig 5.12 c)*.

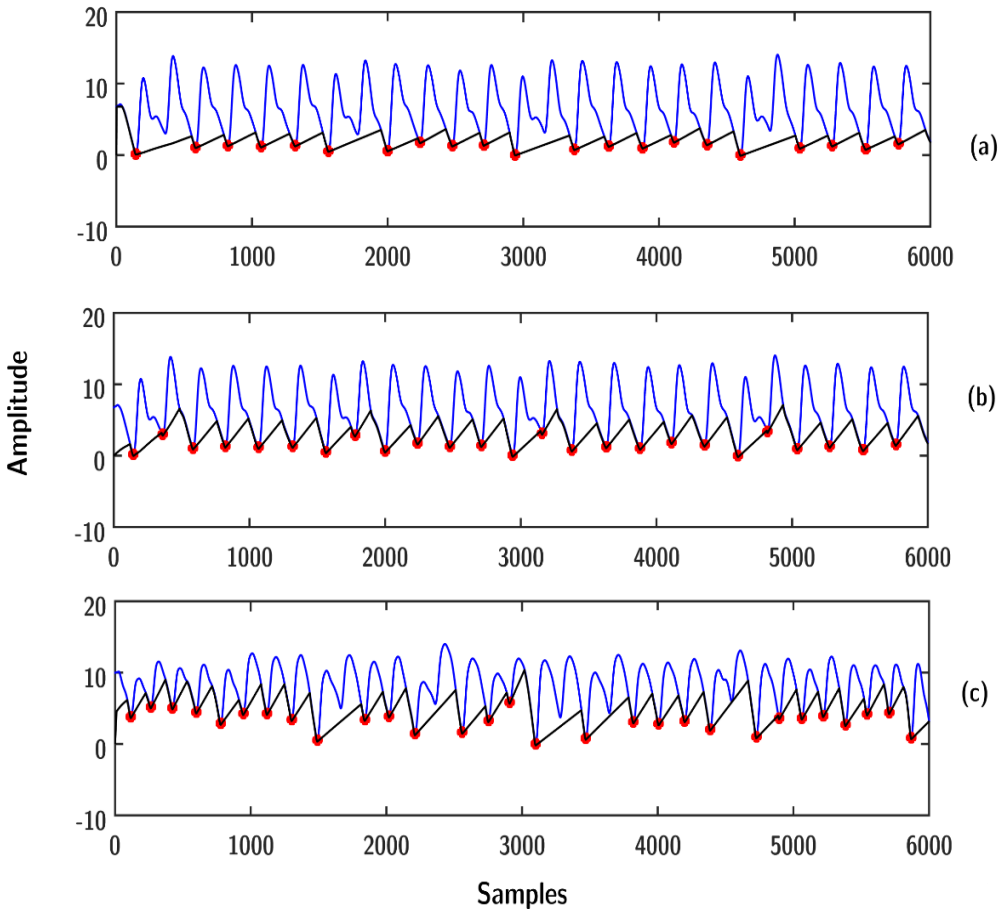


Figure 5.13 *Effects of different S_r values for valley detection.* ***(a).*** *Errors at climb rate at 0.6 in sample 0028_8min* ***(b).*** *Errors rectified with climb rate S_r at 1.5 in sample 0028_8min* ***(c).*** *Errors due to sudden changes in perfusion variation within sample 0030_8min*

Shin et al. 2009 [63] recommended for real peaks to be separated by at least 0.6 of the distances between the previous peaks. In order for this to be possible however, there must be a clear definition for what is real and what is fake first of all through calibration. To achieve this a calibration routine was added similarly to the previous algorithms; on detecting 5 peaks and valleys the algorithm would enter a calibration mode, where the peaks were tested against the largest peak value.

Under the assumption that the majority of the lower frequency drift has been removed by the filter, the peaks can be said to be of equal level hence, if the peaks are at least 75% of the max amplitude of the signal so far, they can be considered valid. On identifying a valid peak, the algorithm was set to search for the valley that was the closest to the peak that exists behind it, where on identification of the valley a verification sub-process was performed similarly to that of introduced to *Xu et al. 2007's [43]* DCBD algorithm and *Kan et al. 2012's [40]* SBD algorithm modification earlier.

This process is carried out until all peaks and valleys collected so far have been determined to be valid. On reaching the end, an average distance threshold was calculated based on their position relative to their neighbours. This threshold is utilized in the discrimination and detection process later. We implemented the calibration procedure and upon testing with sample 0147_8min which contained multiple valleys *(shown in Fig 5.13)*, found it to have a positive effect with 100% detection rate for all valleys with no errors to be found.

A noticeable problem with the algorithm was that if an inflection point proves to be too wide, the slope line would inevitably re-collide with the signal on period exit *(shown in Fig 5.13 and 5.14)*. This caused the algorithm to remain in the *"out-sensing period"* searching along the signal. Although the algorithm was able to discriminate false features based on distance, it had no way of differentiating features based on size, hence, when sample 0370_8min was applied, the algorithm suffered from the misdetection with false features *(shown in Fig 5.14)*. This defeated the algorithm's original goals of utilizing the slope line to overcome minor distortions and notches. The value of the change rate S_r could be reduced, but this came at the expense of the algorithm's sensitivity and adaptability, which led to the problem of the misdetection of real features *(shown in Fig 5.12 a)*.

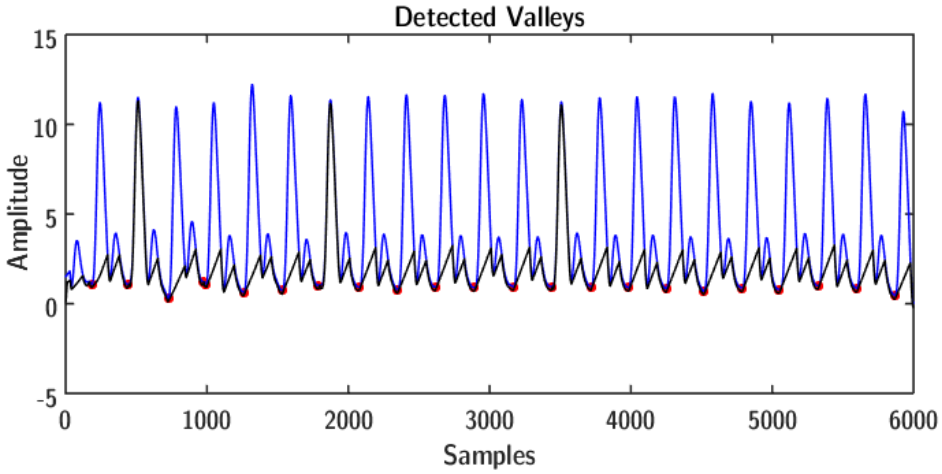


Figure 5.14 *Successful valley detection in sample 0147_8min with early collision problems*

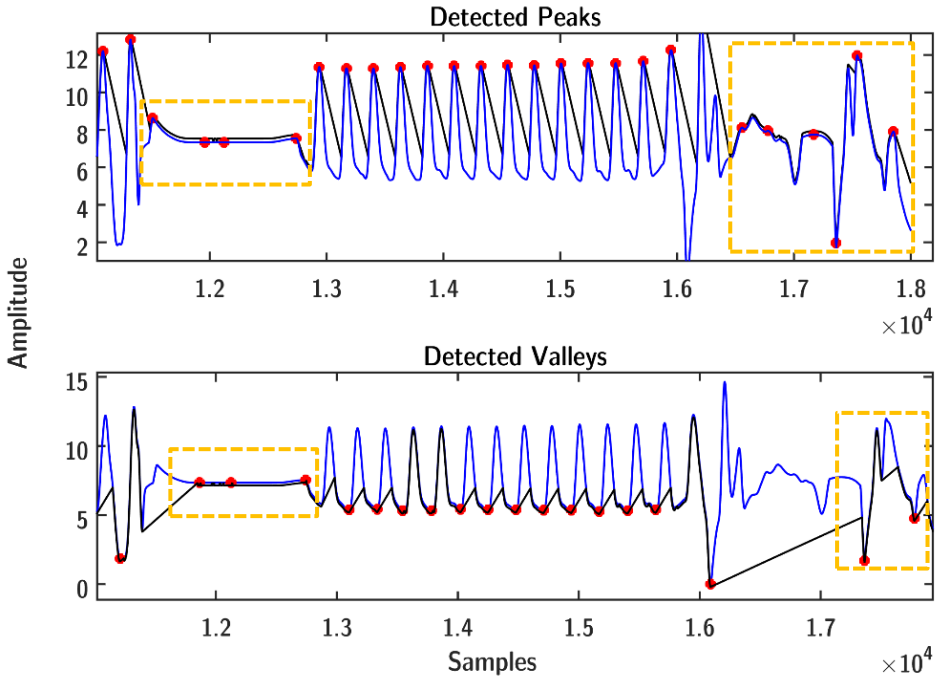


Figure 5.15 *Detection faults within sample 0370_8min due to algorithm's lack of ability to discriminate based on feature size. Errors are encased within orange boxes as shown.*

5.1.4. Adaptive Segmentation Algorithm

The construction of the Adaptive Segmentation *(ADS)* algorithm by *Karlen et al. 2012 [60]* was followed as detailed by the authors algorithm description *(outlined in Fig 5.15)*. The algorithm was originally designed in order to segment the individual PW for analysis, but the algorithm also included the ability to identify inflection points as well.

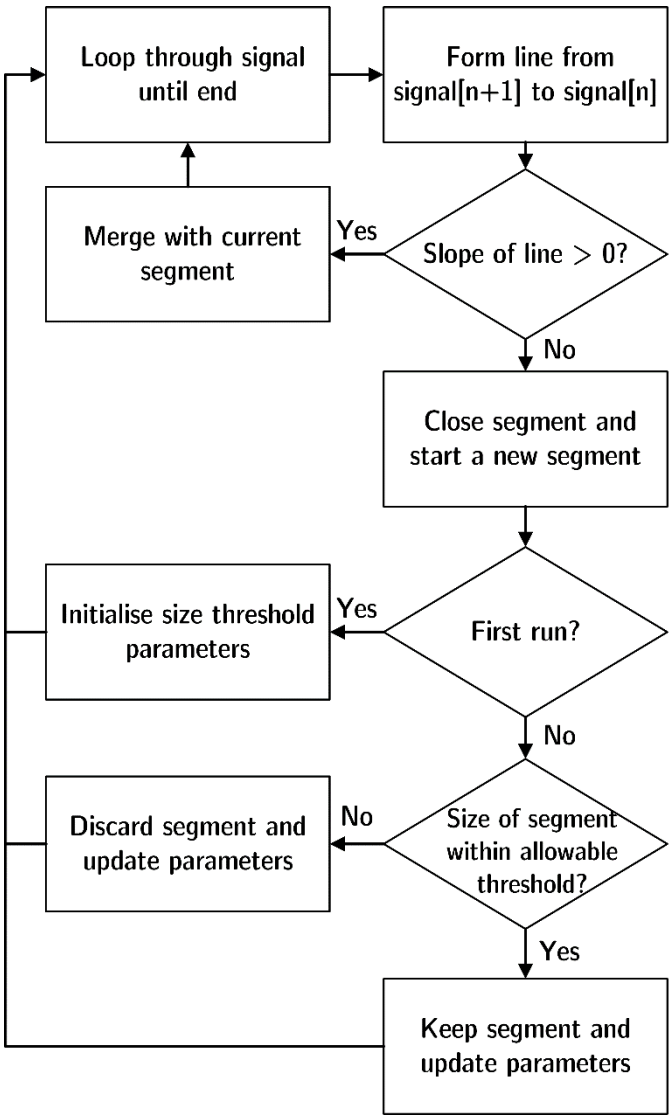


Figure 5.16 *ADS Algorithm outline by Karlen et al. 2012 [60]*

Lines formed from two given points within the signal were checked for their directional slope. On the detection of similar *"upward"* slopes, these lines were combined together to formulate a segment. A segment was closed off and considered to be complete on encountering a line with a downward slope, which correlated to the start or end of a peak *(illustrated in Fig 5.16)*.

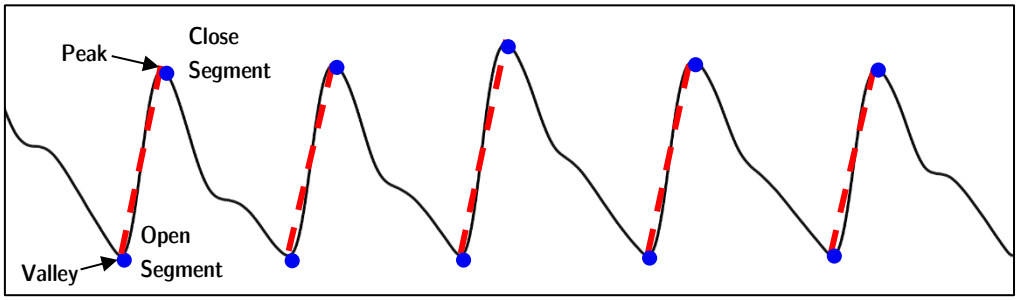


Figure 5.17 *Illustration of the segment formation and the association with inflection points*

On closing the segment the algorithm scanned the signal until another *"upward"* slope was established. At this point a new segment was established in the place of the old, and the process continued as described until no more segments could be formed. Segments were compared to previous neighbours using a threshold that was based on the size of the segments. The threshold range was composed of ThA_{low} for the lower boundary, and ThA_{high} for the upper boundary. These parameters were modified by two additional sets of parameters based on how fast the signal changes *(described as Equations 5.14 and 5.15)*.

$$ThA_{low} = \begin{cases} Seg[1]_{Amplitude} \times 0.6, & ,Initialisation \\ \frac{ThA_{low} + Seg[n]_{Amplitude} \times \alpha_{fast}^{low}}{2} & ,Successful\ Detection \\ \frac{ThA_{low} + Seg[n]_{Amplitude} \times \alpha_{slow}^{low}}{2} & ,Misdetection \end{cases} \quad (Eq.\ 5.14)$$

$$ThA_{high} = \begin{cases} Seg[1]_{Amplitude} \times 1.4 & ,Initialisation \\ Seg[n]_{Amplitude} \times \alpha_{fast}^{high} & ,Successful\ Detection \\ Seg[n]_{Amplitude} \times \alpha_{slow}^{high} & ,Misdetection \end{cases} \quad (Eq.\ 5.15)$$

The parameters described as $\alpha_{fast/slow}^{high/low}$ are parameters to adjust the algorithm's adaptability range for slow changes and fast changes within the signal. For our implementation we utilized the values *(described as eq. 5.16 and 5.17)* that was originally used by the authors in their study.

$$\alpha_{fast}^{high} = 1.4 \text{ and } \alpha_{slow}^{high} = 2 \qquad \textit{(Eq. 5.16)}$$

$$\alpha_{fast}^{low} = 0.6 \text{ and } \alpha_{slow}^{low} = 0.8 \qquad \textit{(Eq.5.17)}$$

Any segment that fell outside this range whether it was greater or smaller than the standard was considered noise and eliminated. In addition to this, any line segments with a zero slope was considered to be a disconnect artefact, and was also subsequently eliminated from the collection.

This threshold was updated every subsequent successful detection and misdetection that occurs; for successful detections the parameters were refined to be closer and closer to the standard size, whereas for misdetection the parameters were adjusted to be broader in the case of adapting to outliers. Through this process the algorithm could adjust for long periods of distortions and disconnects as well as short periods allowing for flexibility and adaptability.

For the calculation of the amplitude *(described as Eq. 6.18),* the starting point of the segment y-value, was subtracted from the corresponding end point y-value in order to establish the absolute amplitude value.

$$Amplitude = y_{end} - y_{start} \qquad \textit{(Eq. 5.18)}$$

There was an additional parameter to be adjusted which is *m*, the length of the lines used to build the segment, whose value is based on the sampling rate of the signal.

To gain an understanding of how *m* affected the algorithm, the parameter was tested against varying values using 20 seconds taken from the start of 0028_8min the testing basis sample *(results shown in Fig 5.17)*.

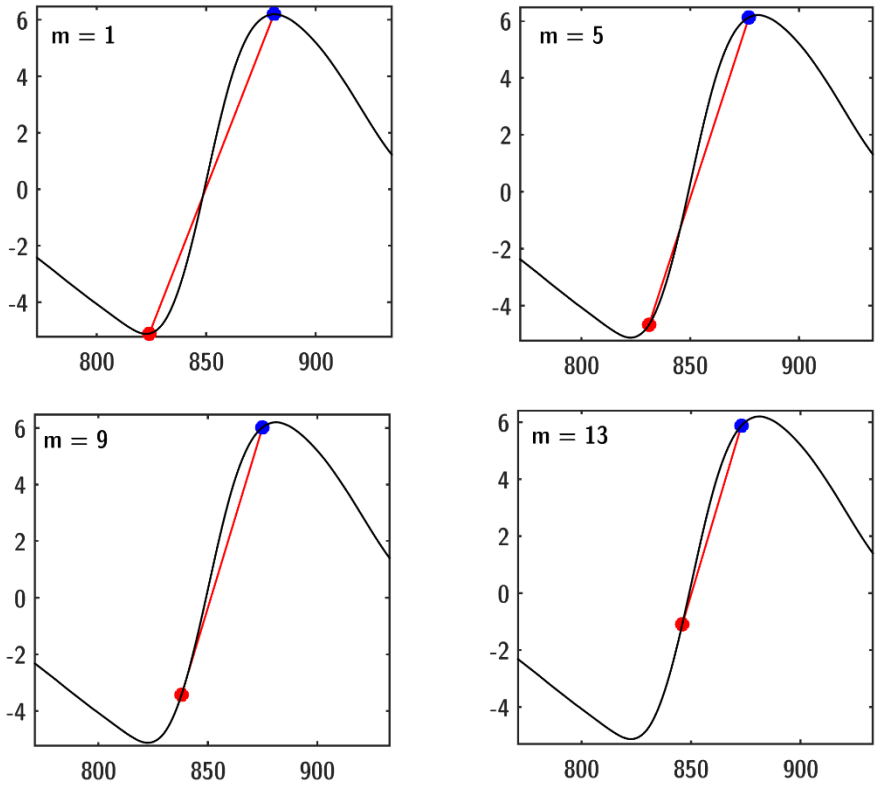


Figure 5.18 *Effects of varying levels of step-size (m) being set*

From the initial testing with the enlargement view of the systolic peak, the results shows that the higher *the* step value *m* is, the narrower the final segment will be. In order to accurately identify the point of inflection, the step size *m* should be set to a size of 1 to ensure that the algorithm searches more thoroughly. For the construction of the segments, a slope comparisons ruleset *(described as Eq. 5.18)* was utilized which compared the values of the points in order to determine the slope direction.

$$Slope = \begin{cases} Upwards, & if(x2 > x1)\ and\ (y2 > y1) \\ Downwards, & if(x1 > x2)\ and (y1 > y2) \end{cases} \qquad (Eq.\ 5.18)$$

The determination of an upward slope is accomplished by comparing the second set of x and y co-ordinates that form the segment. If both are greater than the starting set of x and y co-ordinates, then the slope can be considered *"upwards"* otherwise it is considered downwards.

We used sample 0028_8min in a short test *(shown in Fig 5.18)* as a starting point for calibration, finding that the algorithm was unable to discriminate correctly the segments at the start, but ultimately was able to continue onwards adapting to the PTV amplitudes of future PWs. This was due to utilizing the wrong segment as a basis for constructing the parameters, but the adaptation process indicates a partial success in its implementation *(shown in Fig 5.18)*.

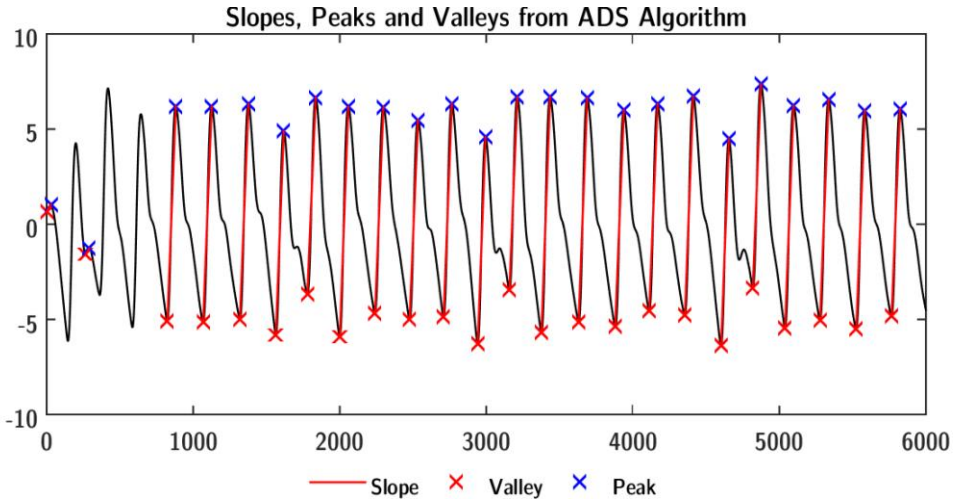


Figure 5.19 *Errors in initial testing of the ADS algorithm using sample 0028_8min*

To rectify this, we introduced a calibration routine similarly to previously analysed algorithms; we collected at least 5 to 6 of the largest segments in a specified window of 15000 samples and obtained the largest PTV amplitude. Using the largest PTV amplitude, we discriminated through the remainder of the segments as the algorithm described. From post calibration, we established the parameters based on the average PTV amplitudes of the discriminated segments, and continued with the detection process as described by the authors.

Additional problems was discovered post-calibration within samples such as 0030_8min; the algorithm was not able to adapt itself to sudden extreme variation shifts. We investigated the matter, and found that it was attributed to the fact that the algorithm only utilizes its most recent neighbour *(shown in Fig 5.19)* hence, it was less sensitive to sudden shifts within amplitude levels. To make the algorithm more sensitive, it was adjusted so that should the detection fail initially, it would go on to compare a threshold calculated based the average PTV amplitude collected so far.

If it was still considered to be outside the range, then it was discarded as a false segment. The modification resolved the issue for the most part, but in circumstances, where the perfusion still changed too suddenly in comparison to the rest of the signal *(shown in Fig 5.20)*, the algorithm would still yield misdetections.

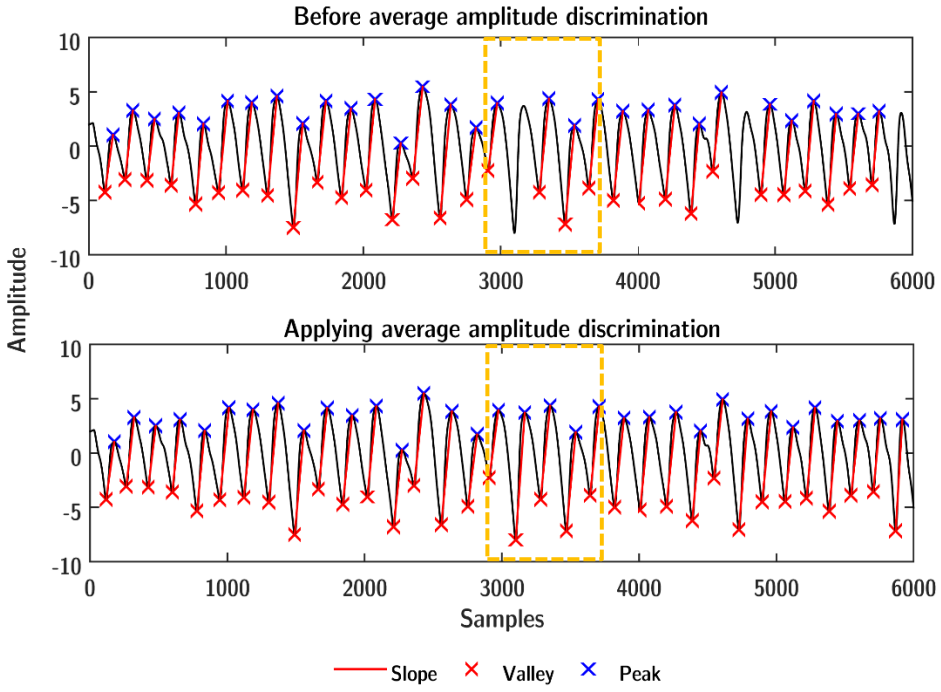


Figure 5.20 *Errors due to sudden changes in signal amplitude (top) and resolution application of post-average discrimination (bottom). Error and rectified error is encased in the boxes.*

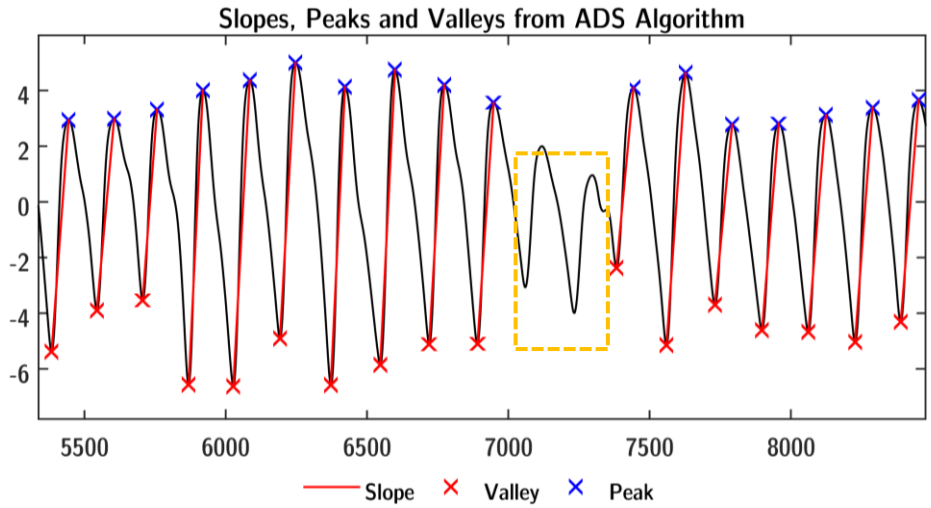


Figure 5.21 *Errors due to sudden and extreme amplitude changes in sample 0030_8min. Point of error is encased in the box.*

We also found with further tests that the algorithm needed valley verification search-back in instances of PWs with multiple notches. This was implemented using a similar process as described previously for past methods. We found that a search-back value of 90 samples was sufficient in order to identify the correct valley *(shown in Fig 5.21)*.

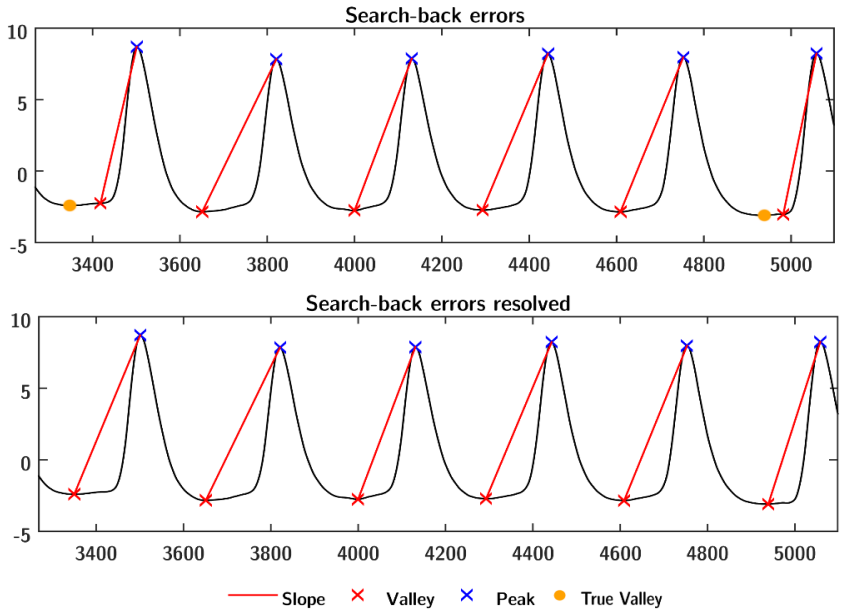


Figure 5.22 *Search-back errors (top) and search-back errors resolved (bottom)*

5.2. Results and Discussion

The analysis of the test results *(shown in Table 5.1 and 5.2)* based on our implementation overall indicated that the algorithms with the highest sensitivity rates was the DCBD algorithm presented by *Xu et al. 2007 [43]* and the SBD algorithm presented by *Kan et al. 2012 [40].* The algorithm with the highest accuracy rating was the ADS algorithm by *Karlen et al. 2012 [60],* which also had the lowest computation run-times. The algorithm that had the lowest accuracy and ratings was the ADT algorithm by *Shin et al. 2008 [63]*, which also happened to have yielded the highest computation times.

Table 5.1. Comparison of the detection rates by the algorithms analysed

	Peak Sensitivity	**Peak Accuracy**	**Valley Sensitivity**	**Valley Accuracy**
DCBD (Xu et al., 2007)	100%	98.99%	100%	98.99%
SBD (Kan et al. 2012)	99.12%	98.70%	99.12%	98.57%
ADT	99.86%	98.15%	96.78%	99.34%
ADS	99.71%	99.86%	99.71%	99.71%

Table 5.2. Comparison of average run times for algorithms

	Short Tests	**Long Tests**
DCBD (Xu et al., 2007)	0.1470	0.3913
SBD (Kan et al. 2012)	0.2079	0.6933
ADT	4.47	15.14
ADS	0.0788	0.2567

The DCBD algorithm by *Xu et al. 2007* utilized the first derivative in order to obtain the linking point between a peak and valley. Using this the algorithm discriminated the features based on the amplitude of the derivative. This allowed for the identification of false features based on their relative size, but led to a decrease in accuracy since it was not able to discern fake features that existed at the same size, but was separated by a distance smaller than the average. The algorithm nevertheless indicated potential if the parameters was more adaptive, and distance discrimination was implemented.

Kan et al. 2012's [40] SBD algorithm utilised size and distance discrimination based on the periodicity of the peaks. In the cases where the HR varies the distance will also change which may require an intermittent FFT on the signal over time in order to identify the correct frequency information. Unlike *Xu et al. 2007's [43]* algorithm, it was able to discriminate based on a size factor between the peak and valley as well as time, but the basis of its parameters are too rigid to be used in adaptive situations. It nevertheless still presents potential for being modified into an adaptive algorithm similarly to *Xu et al. 2007's [43]* algorithm.

The ADT algorithm by *Shin et al. 2008 [63]* while was able to discern fake features by skipping them, overall lacked the logic to differentiate real features from fakes. The algorithm also had problems in adjusting to moderate to large perfusion variations while maintaining its *"in-sensing period".* While the parameters could be adjusted to overcome this by increasing the change rate, it would ultimately defeat the purpose of using the sensing period to avoid distortions and fake features.

The algorithm collision would occur too often leaving the algorithm in the *"out-sensing period"* which is essentially a linear comparison search using the values of the signal. Since the algorithm itself lacked the means to discern fake features from real features characteristics, the detection and sensitivity rates would

significantly decrease. The requirement of calculating the slope line as well as the collision check calculations for the line had also made the algorithm relatively computationally intensive.

The ADS algorithm by *Karlen et al. 2012 [60]* achieved the highest detection rates, owing due its adaptive ruleset that was able to adjust based on varying perfusion amplitudes. Its sensitivity ratings were relatively low due to not being able to adapt quickly enough in circumstances, where there was a sudden dip in perfusion amplitudes. Despite this, overall it was able to recover effectively and carry on when it was able to re-establish a sense of normality, which is what is required for real-life scenarios.

While the algorithm provided a basis for identifying valid features, based on the amplitude relationship between the peak and its associated valley, it provided no means to verify the features based on distance or time. This made the algorithm potentially vulnerable to sudden spikes that contains similar characteristics, but the algorithm could easily be extended to counter this if desired due to its simple structure.

On the analysis of computational complexity and efficiency, the most computationally intensive algorithm was found to be the ADT algorithm. The ADT algorithm requires a single process for the search and a nested process for the peak and valley verification. Hence, the complexity can be described as $O(n^2)$ in terms of Big-O notation when the algorithm was able to identify a peak or valley. Under general circumstances, the algorithm remains at *O(n)* for linear search.

In comparison to the other algorithms despite its simple general case, the ADT algorithm requires several multiplication and division operations in order to calculate the slope, as well as to calculate line intersections. These operations would be carried out for every single new point added to the signal line, unless a

collision of the slope line occurs with the signal line. On collision, comparison operators are mainly used, but these are sparse in comparison to multiplication/division operations, since the algorithm spends the majority of the time in the *"in-sensing"* period.

Xu et al. 2007's DCBD algorithm was noticeably a lot faster than the ADT algorithm. With the two calibration procedures for producing the derivative and parameters, the search process and nested verification procedures required in the algorithm, yielded a total computational complexity of approximately $O(2n) + O(n^3)$. Despite its complexity, the operations used for the algorithm however, were mainly integer based operations such as addition and subtraction.

Simple comparison operators were used for the majority of the algorithm comparisons in determining the location of inflection points. While multiplication and division operations were used, these were used sparsely in the beginning to establish the algorithm parameters. Due to using less expensive arithmetic operations, the algorithm was able to achieve a significantly less computation times than the ADT.

Similarly, Kan et al. 2012's SBD algorithm possesses a similar approach as Xu et al. 2007's algorithm. From valley identification there was also the search process for verification, which gives the algorithm a similar complexity of $O(2n) + O(n^3)$ on the detection of a peak or valley due to their nested structure but otherwise, will remain $O(n)$ in linear search. But the factor behind its increased computation times, is owed to the FFT required which was applied to the entirety of the signal within the simulation.

Although the additional time of the FFT can reduced by taking shorter segments, this may also reduce the accuracy of the overall parameters established, since the length of the FFT is crucial in obtaining a good approximation of the signal. Due to this, intermittent FFTs must occur as the algorithm runs in real-time. This adds a considerable amount of overhead complexity, due to the additional multiplication/division operations required by the FFT.

The ADS algorithm utilizes the derivative similarly to *Xu et al. 2007's* DCBD approach, but alternatively only requires a directional search to identify inflection points, rather than calculating the slop to determine their location. This was achieved by exploiting the directional change of the slopes to match peaks to valleys, giving it a slight edge in terms of computational efficiency over the former.

The algorithm contains within the search process a nested calibration and verification process, which gives it a complexity of $O(n^3)$, however, since calibration only occurs once at the start, the algorithm remained a constant $O(n^2)$ complexity afterwards on identification of a valley. Furthermore, within its verification sub-routines, the operations carried out are mainly comparison operators to compare the value of points to re-affirm the correctness of the valley.

In the general case where it doesn't find anything that falls into range, it possesses a $O(n)$ time complexity. While multiplication/division operations play a prominent role, in the algorithm's structure in the cases of checking and parameter updates, they are performed only intermittently and in comparison, are lesser than the ADT and Kan et al. 2012's SBD algorithm.

The majority of the algorithm consists of comparison operators in order to identify inflection points rather than directly calculating the derivative as was done by Xu et al. 2007's DCBD algorithm. Owing due to not relying on methods such as derivative analysis, and complex floating point operations, the algorithm was able to obtain the fastest operation times of the latter algorithms. From the analysis of the rulesets and algorithm run-times, we selected the ADS algorithm for usage since it was shown to possess the lowest computation complexity, and overall the most optimal detection rates thanks to its adaptable ruleset.

5.3. Summary

In this chapter, an analysis of various feature detection algorithms was carried out with the purpose of understanding their workings, and from the analysis a selection of a suitable algorithm for usage towards the development of the proposed cascade filter. Four algorithms including the two methods reported by the authors who developed the adaptive filter algorithms, namely *Xu et al. 2007[40]* and *Kan et al. 2012[43]*, and two stand-alone adaptive algorithms by *Shin et al. 2009[63]* and *Karlen et al. 2012[60]* were analysed for their accuracy, computational run-times and accuracy against one another.

From the observation and analysis of the computation times as well as the detection rates based on AAMI standards, the ADS algorithm was selected as the candidate to be utilized towards the development of the final algorithm. In comparison to the others it possesses a high accuracy rating, low computational times and the ability to overall adapt as needed towards changes within signal trend.

Chapter 6

The Evaluation of CBSI within Cascaded Filter Design

Chapter 6 explores CBSI in application to the Wavelet Cascaded Filter *(WCF)* by *Xu et al. 2007 [43]*, alongside with a proposed cascaded filter design using FIR-FFT filtering, ADS detection and CBSI. Specifically, the usage of CBSI is evaluated to observe how well it is able to approximate and remove the remaining drift elements within the signal post using softening filters. The WCF algorithm by *Xu et al. 2007 [43]* is selected as the comparison basis since wavelet filtering is considered as the standard optimal filtering method. The plausibility of obtaining similar and or optimal results using the proposed cascaded filter is explored.

6.1. Experimental Setup and Procedure

Using the FIR filter generated from the study in Chapter 5, and the application of the modified ADS algorithm in Chapter 6, experiments were carried out with the aim of observing the feasibility of using these two methods to formulate a cascade filter *(as described in Fig 6.1),* similar to that proposed by *Xu et al. 2007 [43]* and *Kan et al. 2012 [40].* The proposed algorithm is described as the FIR-FFT Cascaded Filter Algorithm *(FFCF)*.

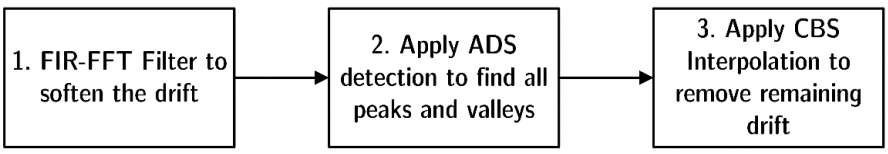


Figure 6.1*Proposed flow of the FFCF Algorithm*

CBSI was applied through the usage of MATLAB's *"pchip"* command with the valleys detected as inputs, and the length parameter for the number of samples to generate being set to the length of the overall signal. Visual analysis was carried out on the individual CBSI estimation, as well as the combined result of the FIR-FFT filter and the CBSI estimation, in order to see how well the baseline approximation was fitted to the signal.

For numerical analysis Pearson's correlation and the RMSE *(listed in Chapter 5 in Equation 5.1 and 5.2 respectively)* again, is used as a basis for measuring similarities and errors between the results. Computation times were recorded as necessary for both short and long tests, using 20 and 60 seconds worth of data respectively to identify any irregularities and similarities. All of the code used can be found within folder *"chapter_six"* within the online repository[5].

6.2. Results and Discussion

6.2.1. The Wavelet Cascaded Filter

Using the full WCF Algorithm with sample 0028_8min in a short test, it was revealed that CBSI Interpolation was able to overall remove the traces elements of drift that the softening process was not able to. There were some distortions at the ends of the signal where the CBSI approximated the drift incorrectly due to the lack of points, but no visual distortions was evident within the main regions of the signal.

[5] https://bitbucket.org/nthegestalt/dissertation-code

To verify that the baseline drift was perfectly fitted underneath the signal, the remaining R.B.D. samples were parsed by the algorithm. The wavelet approximation and the CBSI Interpolation results were combined into one, and then was subtracted from the unfiltered signal.

On visual inspection, we found no distortions applied to the final result in the majority of signals *(example shown in Fig 6.2 and 6.3),* with the exception of sample 0370_8min *(example shown in Fig 6.4)*, where the distortions themselves was further affected and modified by the softening process. Here a false valley was identified and as a result, the CBSI process took the point into consideration as well.

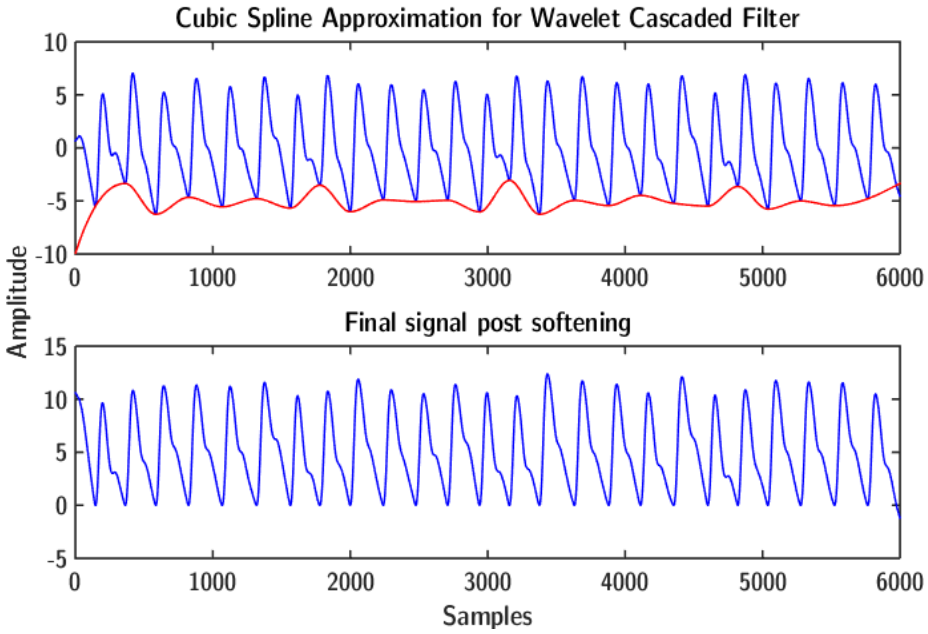


Figure 6.2 *Approximation of the CBSI Interpolation via the Wavelet algorithm on sample 0028_8min. No points were available at the end which caused the approximation to tangent upwards at the end.*

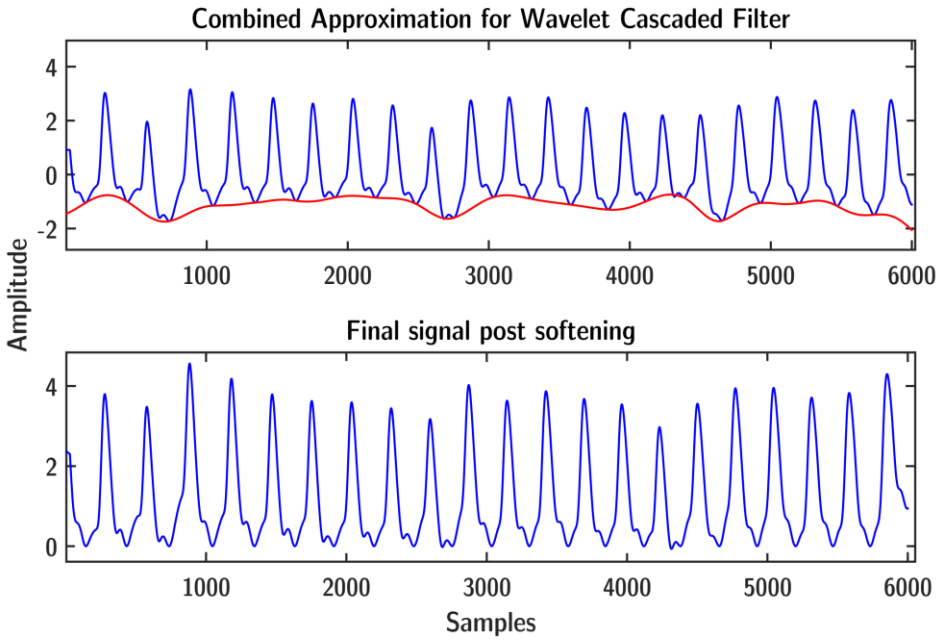


Figure 6.3 *Wavelet Cascaded Filter algorithm applied to sample 0309_8min. The small dichroitic notches were preserved successfully with the baseline fitting just underneath them.*

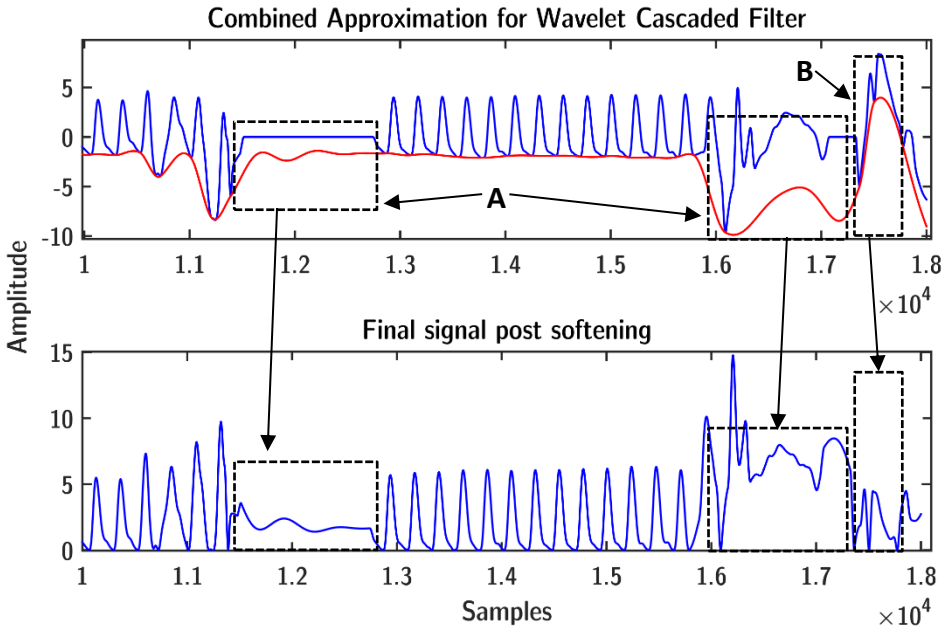


Figure 6.4 *WCF algorithm applied to sample 0370_8min. The encased distortions are due to the Wavelet approximation of the errored area (Point A) and the detection process (Point B).*

The recorded computation times *(shown in Table 6.1)* for both tests indicates that the CBSI process was relatively fast overall in comparison to the rest of the algorithm. The times recorded for the respective length of data shows plausibility for CBSI operation on a continual basis within real-time, but this is providing that segments are kept short enough, and the computation times of the other algorithm components are taken into account of as well.

Table 6.1 *Computation run-times results for the WCF algorithm processes and altogether*

	Wavelet Filtering	**Feature Detection**	**CBSI Interpolation**	**Combined Time**
Short Test	0.02828	0.146951	0.001714	0.176945
Long Test	0.04832	0.391338	0.002991	0.442649

6.2.2. The Proposed FIR-FFT Cascaded Filter Structure

In application of the following processes that make up the FFCF algorithm using sample 0028_8min, we were able to obtain fairly similar results to that of the WCF algorithm. Numerical tests reported a correlation of 95% for the combined final approximation with an RMSE of 0.341, and a 99% correlation for the final result on subtraction with an RMSE of 0.341. Visual inspection of the result showed no distortions to the final result signal *(shown in Fig 6.5)*.

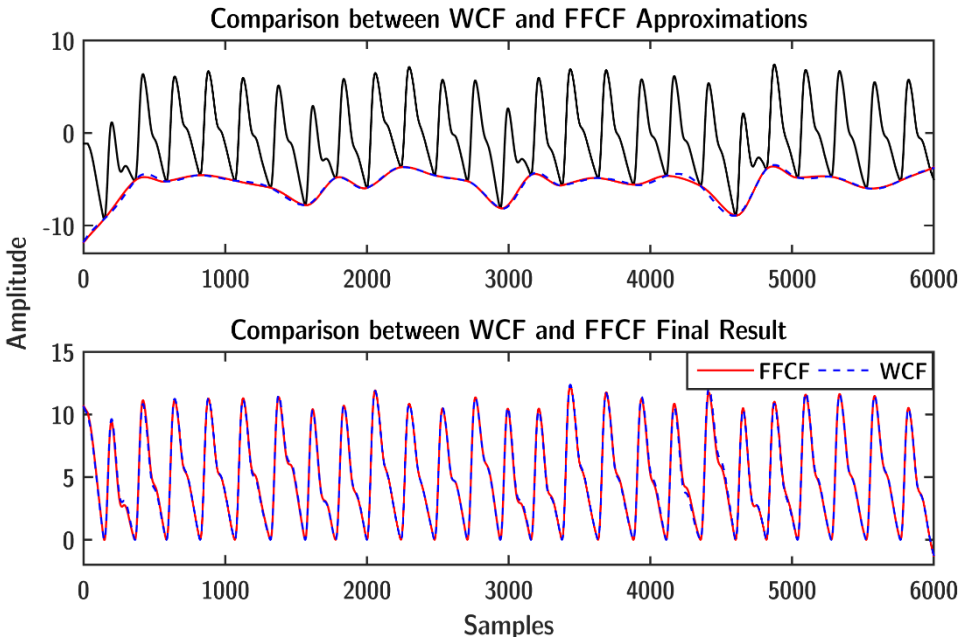


Figure 6.5 *Overlap comparison of the WCF and FFCF drift approximation and final filtering results on sample 0028_8min*

To determine if the same results can be achieved on a signal that differs extremely in the approximation, sample 0009_8min was applied to the FFCF algorithm. The results of the comparison showed a 98% correlation with the approximation and 97% for the final result with an RMSE of 0.634 for both. Given that the maximum amplitude range was approximately 20, it can be said that the error was relatively small. The approximation and the final result are almost identical to one another, proving that the same results can be achieved so long that the signal has been softened enough, to allow for CBSI Interpolation to be carried out properly.

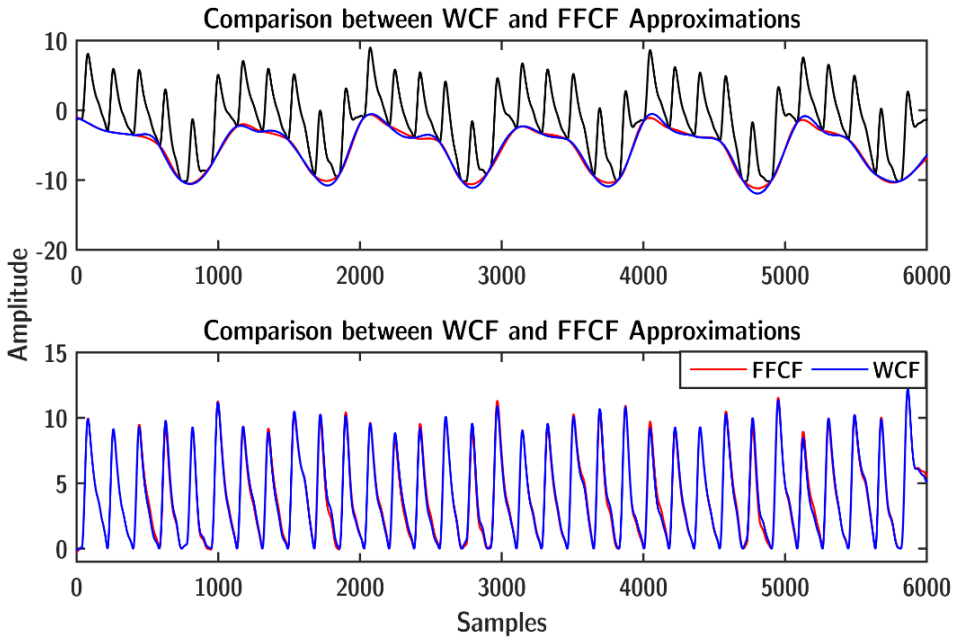
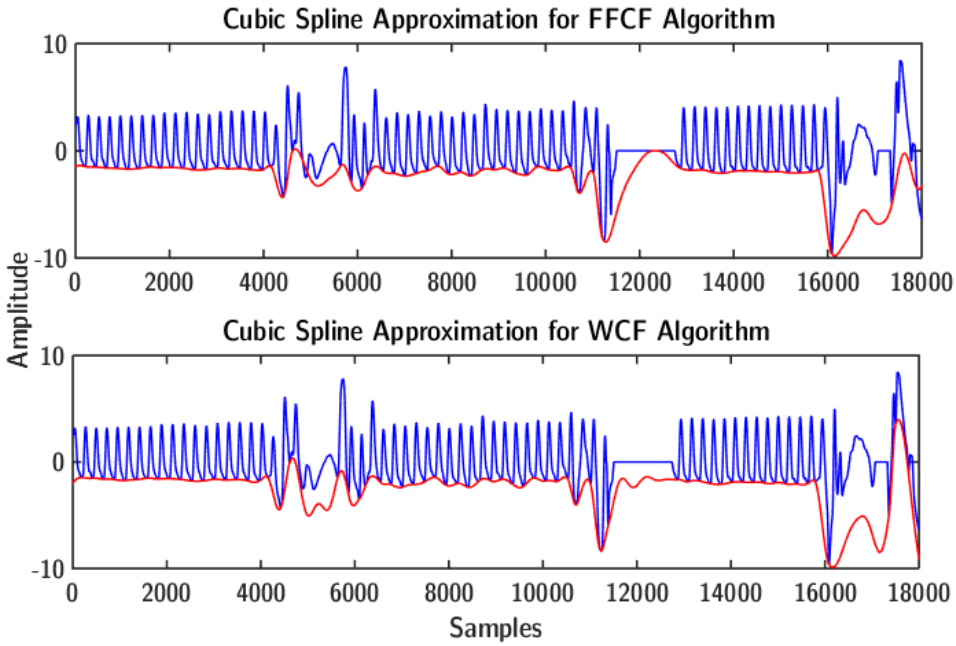


Figure 6.6 *Overlap comparison of the WCF and FFCF drift approximation and final filtering results on sample 0009_8min.*

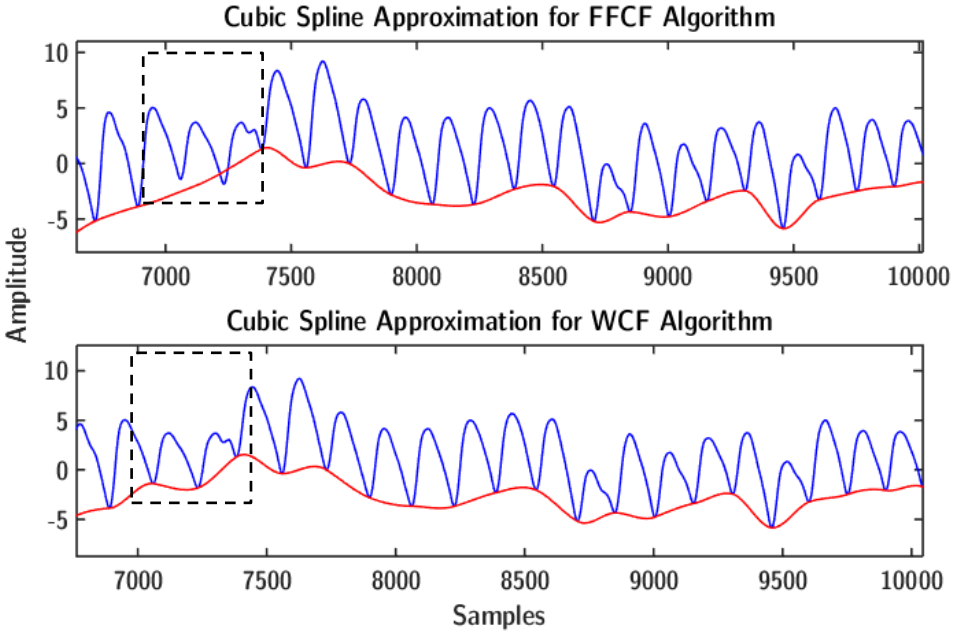
In applying the FFCF algorithm to the remainder of the R.B.D. samples in the short and long tests *(results shown in Table 6.2),* it is revealed that the algorithm was able to achieve the same results as the WCF. There were minor differences due to the softening methods utilized, but the end result overall achieved 98-99% correlation with the exception of sample 0370_8min and sample 0030_8min, where the CBSI Interpolation produced different results owing to detection results from different algorithms *(shown in Fig 6.7 and 6.8)*.

Table 6.2 *Correlation and error results between WCF and FFCF.*

	Approximation Correlation	**Approximation RMSE**	**Result Correlation**	**Result RMSE**
Short Test (20 Seconds)	95.84%	0.221	99.60%	0.204
Long Tests (60 Seconds)	95.60%	0.396	98.38%	0.396



(a). *Different false detection errors leading to different approximations.*



(b). *ADS misdetection errors leading to different approximations. Areas of difference are encased in boxes.*

Figure 6.7 *Differences in end result due to different detection results.*

The computational run-times for the CBSI Interpolation of the FFCF algorithm *(described in Table 6.3)* was approximately the same as the WCF algorithm however, with the combined computational times of the other processes within the FFCF algorithm, we found that the total times were significantly lower than the WCF's by 41-54%. The reduction indicates that our algorithm structure is potentially a more efficient and effective alternative to the WCF, which can be used offline as well if complete decomposition is not warranted for analysis.

Table 6.3 *Computation run-times results for the WCF and FFCF algorithm processes.*

	Drift Softening	**Feature Detection**	**CBSI Interpolation**	**Combined Time**
WCF Short	0.02828	0.146951	0.001714	0.176945
WCF Long	0.04832	0.391338	0.002991	0.442649
FFCF Short	0.000573	0.078793	0.001712	0.081078
FFCF Long	0.001529	0.256726	0.0030511	0.2613061

From these results we demonstrated that our proposed FFCF algorithm can perform on-par with the WCF algorithm, but maintained computational efficiency due to the reduced complexity of the selected algorithm components. The FFCF algorithm described here can potentially have the computation time decreased even further through optimization and application on shorter segments of data. Through the application of the OLS algorithm, it may be possible to process and delineate the signal within real-time constraints, given the results that have been seen through the timing and correlation tests.

6.3. Summary

In this chapter, the concept of CBSI was explored within the WCF algorithm proposed by *Xu et al. 2007 [43]*. Through the analysis, it was seen that CBSI was a computationally efficient and effective means of approximating the remaining baseline drift within the signal post-softening of the baseline drift. The proposed cascaded filter using FIR-FFT, ADS detection and CBSI Interpolation was able to perform on-par with the WCF proposed by *Xu et al. 2007 [43]*.

A correlation of 98-99% was achieved for the final result with some discrepancies in the approximation due to the differences in detection and filtering methods. The algorithm was able to perform noticeably faster than the WCF, achieving a reduction in run time of by 41-54%, due to not requiring as many calculations as the WCF. It was also somewhat less sensitive in terms on detection mainly due to the algorithm selected, but this can be rectified through further optimisation. From the effectiveness of the demonstrated algorithm, a foundation has been established for further study in Chapter 7 for a real-time algorithm.

Chapter 7

Real-time Algorithm Design and Development

In this chapter a real-time algorithm concept is proposed and implemented through a simulation, based on the concepts explored in previous chapters of the FFCF algorithm. FIR-FFT filtering is substituted with the OLS algorithm in order to emulate real-time segment processing of the PPG signal. From this the computational run-times for each segment and the correlation of the end results, are compared and contrasted, in order to investigate the plausibility of the proposed algorithm.

7.1. Algorithm Simulation Design and Development

Previously in Chapter 6 the FFCF algorithm structure was proposed and was shown through experiments, to achieve not only optimally clean results similarly to the WCF algorithm by *Xu et al. 2007 [43]* but also as superior run-times. The OLS algorithm allows for FIR-FFT filtering to take place on a continual real-time basis hence, it is a suitable substitution to determine if the algorithm potentially real-time viable. The process of developing the algorithm is divided into two sections for discussion; the planning and outline of the basic algorithm logic, and the technical implementation aspects within MATLAB.

7.1.1. Algorithm Process Design

The signal was segmented and processed *(illustrated in Fig 7.1)* on a chunk-to-chunk basis via OLS. From the softening, chunks were combined into a single result generated by the OLS algorithm forming the softened signal line *(SSL)*, which was used as the basis for future conditioning operations. Since the length of the filter was 650, there is around a 1 second delay to the result, but with enough data being accumulated to ensure there was enough PTV pairs, the ADS algorithm was applied on the SSL formed so far to calibrate itself. Note that, typically, the length of required data should be selected based on the sampling rate, so that there is at least 5-6 PW cycles within the segment for proper calibration. We considered 5 seconds to be sufficient which equals to approximately 1500 samples, based on the R.B.D. sampling rate of 300 Hz.

Following the initial run of the ADS algorithm, CBSI was carried out on the SSL using the valleys detected. The resulting CBSI approximation was subtracted from the softened result, and then truncated at the end to the last valley's index to form the basis for the next segment to be joined at the same location. The CBSI process typically results in the inflection points being reduced or boosted to zero hence, these areas can be joined together contiguously.

Any values that existed after the last valley within the segment were discarded since they are incomplete, and would cause errors if kept when the next segment is joined similarly to the OLS algorithm. For any following segments, we did not use the directly formed segments from the filtering result, but instead we formed new segments using the SSL starting at the last valley from the previous segment, to the last valley of the current segment. In this manner all discarded data could be included and processed as parts of a complete PW so no data would be lost.

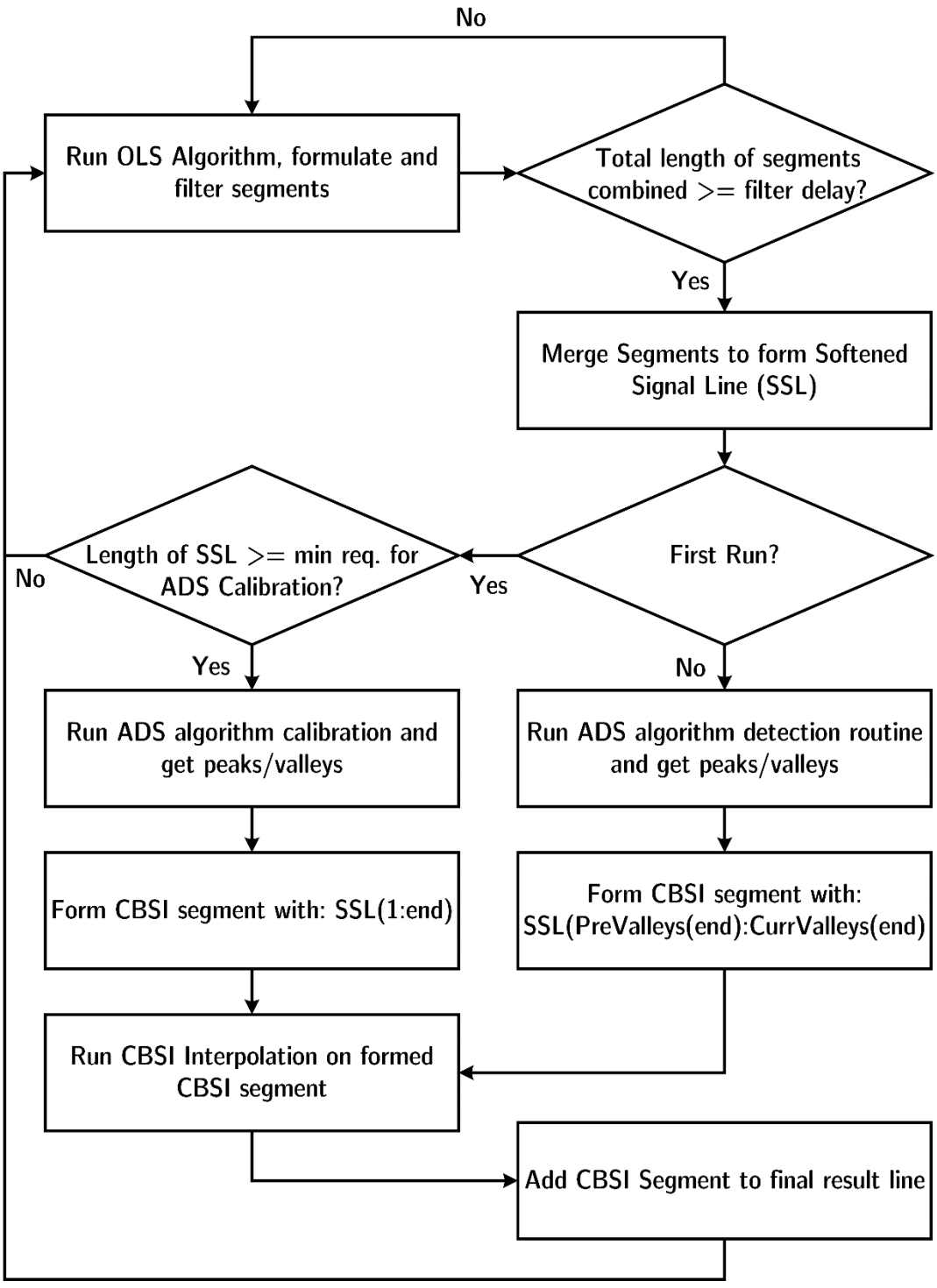


Figure 7.1 *Proposed outline for Real-time FFCF algorithm structure.*

7.1.2. Algorithm Simulation

To make implementation simple and clear, we modularized each of the algorithm's processes based on their characteristics and relationships within functions *(as illustrated in Fig 7.2)*. The OLS algorithm function was set as the main instigator, but in consideration along with the CBS Interpolation function, they were relatively simple and did not require repetitive logic of their own[6]. These were decided as suitable to being implemented as simple self-contained functions.

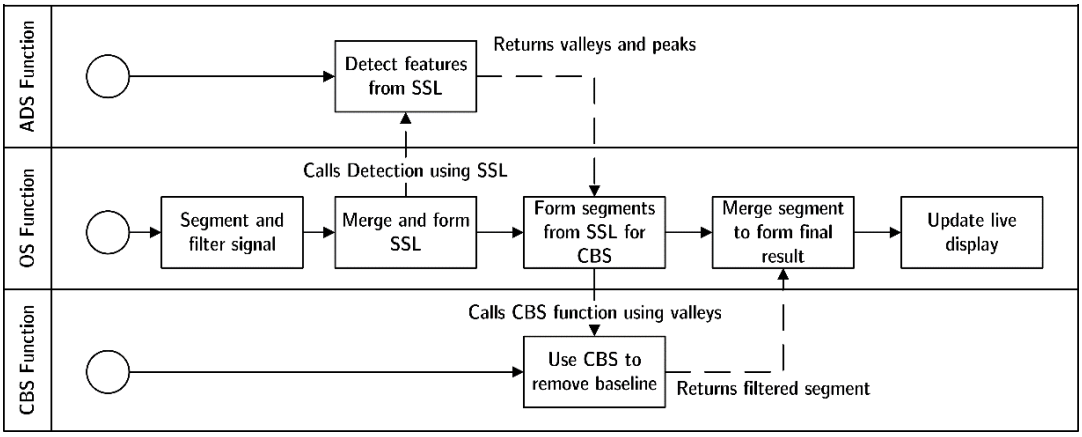


Figure 7.2 *Mapping of the relationship between algorithm processes as functions.*

The ADS algorithm requires persistent variables as well as repetitive calls to its own functions for peak and valley verification, and in addition to these the OLS algorithm function required access to the variables contained within the ADS algorithm function, such as collected peaks and valleys to call the CBSI function. To satisfy these requirements the ADS algorithm was implemented in a class, thus making its required parameters class parameters, for persistence, and allowing access as required. The code for the implementation of the function and class is listed within the folder *"chapter_seven"* in the online repository[7].

[6] *Such examples are verification checks that runs through most of the code while require moderate to large blocks of code to describe their logic.*
[7] https://bitbucket.org/nthegestalt/dissertation-code/

Specifying the parameters of the OLS algorithm, we selected 150 samples to be combined with the 650 samples length required by the filter, forming blocks of 800 samples for the OLS algorithm. This yielded intermittent output of filtered data every half a second. In addition to the implementation of the algorithm logic, a live GUI display was also implemented *(shown in Fig 7.3)* showing the result of each stage within the algorithm, for the analysis and review of the filtering process in case debugging a particular area was needed.

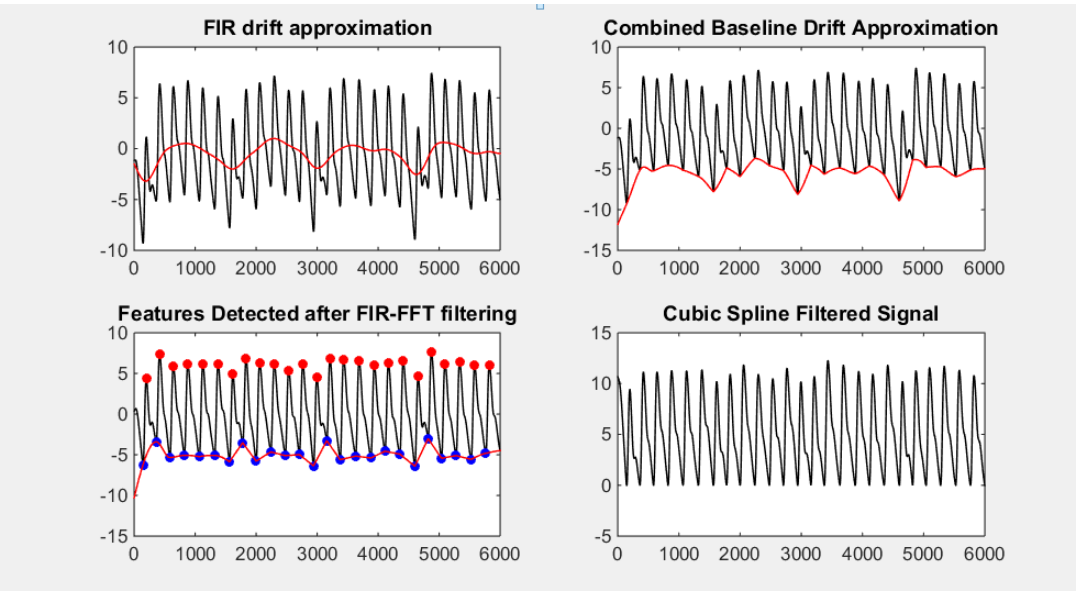


Figure 7.3 *Screenshot of live GUI display for filtering process.*

We selected sample 0028_8min to be used in a short test in order to investigate how the algorithm performed. The results of the testing revealed that the algorithm modification was implemented successfully with a correlation of 99.77% and an RMSE of 0.238. There was almost no difference between the approximations of the normal FFCF algorithm and the RT-FFCF modified algorithm *(shown in Fig 7.4).* With this confirmed the remainder of the R.B.D. samples were processed for analysis and comparison.

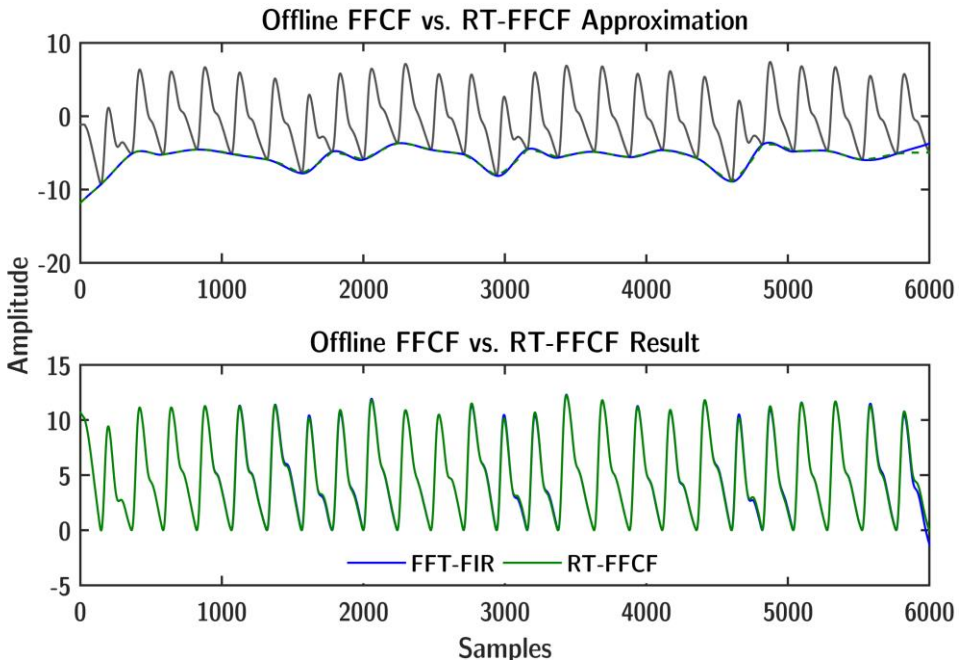


Figure 7.4 *Comparison of the FFT-FIR filter with CBSI and the RT-FFCF algorithm.*

7.2. Results and Discussion

Running the RT-FFCF algorithm on the R.B.D. samples it was shown that the algorithm had worked as intended, achieving results similar *(described in Table 7.1)* to its offline counterpart. The baseline drift correlation was about 95%, and the slight drop within final result correlation to 98% was found in the long test. We traced this again to the different results given by the detection algorithms. Aside from the two minor errors, the majority of the signals yielded a 99% correlation to that of the WCF algorithm output. From this we can conclude that despite segmentation processing, we were able to reconstruct the final result so that it is able to be as effective and similar as the WCF. The visual results of the filtering process for bother algorithms can be found within Appendix B3.

Table 7.1 *Correlation and RMSE tests for the RT-FFCF Algorithm modifications.*

	Approximation Correlation	**Approximation RMSE**	**Final Result Correlation**	**Final Result RMSE**
Short Test	95.06%	0.327	99.16%	0.315
Long Test	95.79%	0.351	98.65%	0.352

The average run time for processing each segment *(described in Table 7.2)* was approximately 0.0103 and 0.0107 seconds for the long and short tests respectively. The majority of the overhead was attributed by the detection algorithm indicating further optimization potential, but overall the timing results of the algorithm indicates that it can still operate in real-time like the OLS algorithm, since the time it takes to process each segment is still below the time it takes in order to collect 150 samples worth of data being under 0.5 seconds at ~0.01 seconds on average.

Table 7.2 *Average run-times for each of the algorithm process per segment.*

	Filtering Process Time	**Detection Process Time**	**CBS Interpolation Time**	**Total**
Short Test	0.000134	0.00953	0.00109	0.01075
Long Test	0.000131	0.00906	0.00111	0.01030

The RT-FFCF algorithm in addition to detecting and extracting the baseline drift in real-time, enabled accurate segmentation and preservation of the PW to occur in real-time as well. Through this real-time analysis can be conducted by specialists or advanced algorithms on the PW as they are procured, to analyse and diagnose arterial compliance and cardiovascular conditions such as arrhythmia which have a relation to the shape of the PW *[13, 70]*. However, while the majority of the algorithm processes has shown plausibility of running in real-time, further investigations can be undertaken to further reduce the associated computation times to optimize the algorithm. This is a particularly important factor for lower-end systems such as embedded hardware.

In terms of current hardware limitations and practicality, due to the nature of the algorithm utilizing heavy amounts of precise floating point calculations, and requiring the processing large amounts of data within such a restricted time frame, a fairly powerful processor would still be required. Taking into consideration of power consumption utilized by most powerful processors today, the practicality of having such a device as a standalone wearable wristwatch device may not yet be feasible due to size and bulkiness.

But with the increasing power of modern smart phones, as well as miniature computers such as the Raspberry Pi, a pseudo-standalone device may be achieved. A dedicated wrist-watch device containing sensors could perhaps be implemented to collect and perform initial processing events that do not require computationally intensive operations. Data from the said device would be passed onto the smartphone or miniature computer through low energy Bluetooth, where more intensive processing could be carried out effectively.

Alternatively, data could be sent via GSM or the internet from the device to a server, where processing could be carried out and the results sent back to the user. This approach could be taken for lesser powerful phones, taking into consideration the variety of the phones in ownership. Overall, the practicality of the algorithm in application to real life would still be highly feasible. But as processors and battery technology grow more powerful and sophisticated, the possibility of a purely standalone, portable wrist watch device being created would increase as well.

7.3. Summary

A real-time method based on the proposed FFCF algorithm structure within Chapter 7 was presented and implemented. Using the OLS algorithm, the signal was processed in chunks to investigate whether it was possible to apply the algorithm on a real-time basis. The concept presented formed the RT-FFCF algorithm where simulation test results concluded that the presented modifications had in no way reduced the effectiveness of the original method, and that the time required to process each segment was far under the time required to form each segment for processing.

These results confirms the real-time plausibility of the algorithm modifications however, while the algorithm presents a fairly low computation time a majority of the overhead resulted from the ADS detection algorithm. It is speculated that the algorithm implementation could further be modified and refined in order to reduce the complexity, and in doing so reduce even further the computational times required for the algorithm.

Chapter 8

Conclusion and Future Work

8.1. Conclusions of the Study

An approach to extracting the baseline drift from the PPG signal in real-time has been developed based on existing concepts and optimization techniques. We have demonstrated that under the right designs and conditions, cut-off frequency filters can perform on-par to that of wavelet filters at a 97-98% final result correlation, when applied in the extraction of a certain frequency range. Through our studies, we have also shown that there were discrepancies due to the differences in the two methods, but these are minor and barely noticeable most of the time. Although the methods are not suited for time-varying noises due to the cut-off limit, this circumstance is only when the drift frequencies exceed the recommended cut-off and merges with the HR frequencies.

Through the usage of CBSI, it has been shown that remaining outlier drift frequencies can be targeted and removed regardless of the softening stage methods analysed. Optimization techniques have been explored and implemented which allowed the reduction of the FIR filter computation times. In an offline test with the proposed algorithm structure, we were able to achieve a reduction in computation times of up to 41-54% to that of the WCF algorithm. This is owed due to the FIR filter's ability to target the drift directly instead of relying on level decomposition, making it an equally effective means to the WCF. On comparison of the final results, we found that the algorithm was able to achieve an average 98-99% correlation to that of the WCF algorithm's final results.

We combined the proposed algorithm structure with the real-time filtering OLS algorithm, to form a concept algorithm that can potentially be used in a real-time basis. Through simulations where we selected 0.5 seconds to be our real-time constraints, the algorithm was able to process segments and output the result in ~0.01 seconds on average. These results indicated that the algorithm was able to successfully operate within these constraints. In regards to the final output result, we found that it was able to match the WCF algorithm's output with a 98-99% correlation once again, showing little errors and distortions within the majority of the signal samples.

8.2. Potential Future Work

Although through this study we have shown through simulations that the concept algorithm is viable, further studies can potentially be undertaken in order to improve and extend its functionality. Some possible directions are:

1. Due to time constraints we were only able carry out a limited testing of the algorithm's effects on a subset of PPG signals, which we have seen to be relatively normal in variation. In order to consider real-life applications, there should be investigation on how the algorithm reacts to stranger morphologies such as arrhythmias. From this, the algorithm can be extended and improved to cater for such conditions. The analysis of the results by an expert would also be welcomed in order to validate and verify the workings of the algorithm.

2. The algorithm currently is only implemented within a simulation via MATLAB. The concepts utilized within the algorithm however, have all seen usage within real-time and efficient applications on lower end systems and embedded devices in one form or another. CBSI although can be

computationally intensive, has been optimized extensively due to its beneficial uses outweighing its drawbacks. It has seen multiple applications within the field of robotics and computer graphics for real-time trajectory planning, smoothing and pixel anti-aliasing applications *[71, 72]*.

Within recent years, it's usage has also been seen within applications for lesser powerful platforms such as smartphone applications *[73, 74]*. In recent years, optimized libraries for microcontrollers such as the Arduino *[75]* have also been realized. Prior to this, it has also been implemented successfully for years on graphics calculators with limited processing power in order to visualize polynomials *[76]*.

The FFT on the other hand have been implemented successfully on microcontrollers *[77]* through the limiting the amount of floating points at the cost of accuracy. Recently, a library by *Andrew Holmes [78],* has been developed for exploiting the discrete GPU of the Raspberry Pi in order to efficiently calculate the FFT. In benchmark tests [79] the library was able to achieve 136ms for an FFT with the size of 1024 samples, with a noticeable amount of accuracy in comparison to the non-limited variant.

These timing benchmark values were found to be relatively close to the values of 131-134ms that was obtained by our algorithm tests. With this, we have indications that there is potential for porting the algorithm to lesser powerful hardware utilizing correct tools and techniques. Through the investigation of optimized libraries and careful design, we believe that the algorithm could possibly be ported to a lower-end, embedded platform. These platforms may include microcontroller boards such as the Raspberry Pi 2 or a modern smartphone-based device, which would ultimately give way for the development of a portable health monitoring system.

3. Given that the algorithm can accurately identify the inflection points, segmentation, the analysis of the PW can be conducted within real-time. This allows for the development of real-time diagnosis and analysis algorithms of the PW. An AI learning system perhaps could be developed to facilitate the adjustment of filter coefficients. This would allow deeper customization and adaptability for specific scenarios, providing for even better filtering effects and computational efficiency. Such a system could be used in conjunction to adaptive motion cancellation techniques [81] to significantly improve the quality of the PPG signal, and lessen the restriction on patient mobility.

Future work is not limited to the described list above, but these directions would allow for the realization of a platform, that can continuously track and analyse an individual's cardiovascular state within real-time, and without limiting the patient's quality life through the known beneficial properties of PPG sensors.

Bibliography

1. World Health Organization's, W.S. *A global brief on hypertension Silent killer, global public health crisis*. 2013.
2. van Ravenswaaij-Arts, C.M.A., et al., *Heart Rate Variability.* Annals of Internal Medicine, 1993. **118**(6): p. 436-447.
3. Rajendra Acharya, U., et al., *Heart rate variability: a review.* Medical and Biological Engineering and Computing, 2006. **44**(12): p. 1031-51.
4. Reunanen, A., et al., *Heart rate and mortality.* Journal of Internal Medicine, 2000. **247**(2): p. 231-239.
5. Neuman, M.R., *Vital Signs: Heart Rate.* Pulse, IEEE, 2010. **1**(3): p. 51-55.
6. Ram, C.V.S., *Introduction to Hypertension*, in *Hypertension: A Clinical Guide*. 2014, CRC Press.
7. Luckson, M., *Measuring blood pressure.* Practice Nurse, 2008. **35**(8): p. 19-22.
8. Bakris, G. and R.R. Baliga, *Hypertension*. 2012, Oxford: Oxford University Press, USA.
9. Tamura, T., et al., *Wearable Photoplethysmographic Sensors—Past and Present.* Electronics, 2014. **3**(2): p. 282-302.
10. Karlen, W., et al., *CapnoBase: Signal database and tools to collect, share and annotate respiratory signals*, in *Annual Meeting of the Society for Technology in Anesthesia (STA)*. 2010: West Palm Beach.
11. Allen, J., *Photoplethysmography and its application in clinical physiological measurement.* Physiological Measurement, 2007. **28**(3): p. R1.
12. Haghighat, A.R., *Wave Travel and Velocity*, in *Snapshots of Hemodynamics: An Aid for Clinical Research and Graduate Education*. 2005. p. 880.
13. Clara, F.M., et al., *Evaluation of arterial propagation velocity based on the automated analysis of the Pulse Wave Shape.* Journal of Physics: Conference Series, 2011. **332**(1): p. 012014.
14. Kim, E.J., et al., *Relationship between blood pressure parameters and pulse wave velocity in normotensive and hypertensive subjects: invasive study.* J Hum Hypertens, 2006. **21**(2): p. 141-148.
15. C., I., *Blood Pressure Measurment*, in *Encyclopedia of Medical Devices and Instrumentation*, J.G. Webster, Editor. 1988, John Wiley \& Sons, Inc.
16. Nichols, W.W., et al., *Effects of Arterial Stiffness, Pulse Wave Velocity, and Wave Reflections on the Central Aortic Pressure Waveform.* The Journal of Clinical Hypertension, 2008. **10**(4): p. 295-303.

17. Kamal, A.A.R., et al., *Skin photoplethysmography — a review.* Computer Methods and Programs in Biomedicine, 1989. **28**(4): p. 257-269.
18. Elgendi, M., *On the Analysis of Fingertip Photoplethysmogram Signals.* Current Cardiology Reviews, 2012. **8**(1): p. 14-25.
19. Chen, W., et al., *Continuous estimation of systolic blood pressure using the pulse arrival time and intermittent calibration.* Medical and Biological Engineering and Computing, 2000. **38**(5): p. 569-574.
20. Shriram, R., et al. *Continuous cuffless blood pressure monitoring based on PTT*. in *Bioinformatics and Biomedical Technology (ICBBT), 2010 International Conference on*. 2010.
21. Chen, Y., et al., *Continuous and Noninvasive Measurement of Systolic and Diastolic Blood Pressure by One Mathematical Model with the Same Model Parameters and Two Separate Pulse Wave Velocities.* Annals of Biomedical Engineering, 2012. **40**(4): p. 871-882.
22. McCombie, D.B., A.T. Reisner, and H.H. Asada. *Adaptive blood pressure estimation from wearable PPG sensors using peripheral artery pulse wave velocity measurements and multi-channel blind identification of local arterial dynamics*. in *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*. 2006. IEEE.
23. McCombie, D.B., et al. *Adaptive hydrostatic blood pressure calibration: Development of a wearable, autonomous pulse wave velocity blood pressure monitor*. in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*. 2007.
24. Puke, S., et al. *Blood pressure estimation from pulse wave velocity measured on the chest*. in *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*. 2013. IEEE.
25. Head, G.A., et al., *Ambulatory blood pressure monitoring in Australia: 2002 consensus position statement.* Journal of Hypertension, 2002. **30**(2): p. 253-266 10.1097/HJH.0b013e32834de621.
26. Pickering, T.G.M.D.D., D.M.D. Shimbo, and D.M.D.M.P.H. Haas, *CURRENT CONCEPTS: Ambulatory Blood-Pressure Monitoring.* The New England Journal of Medicine, 2006. **354**(22): p. 2368-74.
27. Meredith, D., et al., *Photoplethysmographic derivation of respiratory rate: a review of relevant physiology.* Journal of medical engineering & technology, 2012. **36**(1): p. 1-7.
28. Garde, A., et al. *Empirical mode decomposition for respiratory and heart rate estimation from the photoplethysmogram*. in *Computing in Cardiology Conference (CinC), 2013*. 2013. IEEE.

29. Fleming, S.G. and L. Tarassenko, *A comparison of signal processing techniques for the extraction of breathing rate from the photoplethysmogram.* Int J Biol Med Sci, 2007. **2**(4): p. 232-6.
30. Lin, Y., et al. *Removal of Pulse Waveform Baseline Drift Using Cubic Spline Interpolation*. in *Bioinformatics and Biomedical Engineering (iCBBE), 2010 4th International Conference on*. 2010.
31. Lisheng, X., et al. *Implementation of cuff-less continuous blood pressure measurement system based on Android*. in *Information and Automation (ICIA), 2012 International Conference on*. 2012.
32. Zhao, S., H. Chao, and M.Q.H. Meng. *A pulse wave filter method based on wavelet transform soft-threshold and adaptive algorithm*. in *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*. 2010.
33. Steven, W.S., *Chapter 13: Continuous Signal Processing*, in *The scientist and engineer's guide to digital signal processing*. 1997, California Technical Pub. p. 243-260.
34. Lyons, R.G., *Understanding digital signal processing*. 1997, Reading, Mass: Addison Wesley Pub. Co.
35. Pilt, K., et al., *Second derivative analysis of forehead photoplethysmographic signal in healthy volunteers and diabetes patients*, in *World Congress on Medical Physics and Biomedical Engineering May 26-31, 2012, Beijing, China*, M. Long, Editor. 2013, Springer Berlin Heidelberg. p. 410-413.
36. Dae-Geun, J., et al., *A Robust Method for Pulse Peak Determination in a Digital Volume Pulse Waveform With a Wandering Baseline.* Biomedical Circuits and Systems, IEEE Transactions on, 2014. **8**(5): p. 729-737.
37. Smith, J.O. *FFT versus Direct Convolution*. 2010 [cited 2015 25th November 2015]; Available from: http://www.dsprelated.com/freebooks/sasp/FFT_versus_Direct_Convolution.html.
38. Lyons, R. *DSP Tricks: Doing Zero-phase Filtering*. 2009 [cited 2015 23rd November 2015]; Available from: http://www.embedded.com/design/configurable-systems/4008847/DSP-Tricks-Doing-Zero-phase-filtering.
39. Chunming, X., et al. *A Practical Approach to Wrist Pulse Segmentation and Single-period Average Waveform Estimation*. in *BioMedical Engineering and Informatics, 2008. BMEI 2008. International Conference on*. 2008.
40. Kan, L., et al. *A cascade filter for pulse wave baseline drift elimination*. in *Image and Signal Processing (CISP), 2012 5th International Congress on*. 2012.

41. Steven, W.S., *Chapter 19: Phase Response*, in *The scientist and engineer's guide to digital signal processing*. 1997, California Technical Pub. p. 319-332.
42. Polikar, R. *FUNDAMENTAL CONCEPTS & AN OVERVIEW OF THE WAVELET THEORY* The Wavelet Tutorial 2001 12th January 2001 [cited 2015 9th August 2015]; Available from: http://users.rowan.edu/~polikar/WAVELETS/WTpart1.html.
43. Xu, L., et al., *Baseline wander correction in pulse waveforms using wavelet-based cascaded adaptive filter.* Computers in Biology and Medicine, 2007. **37**(5): p. 716-731.
44. Wang, D., D. Zhang, and G. Lu, *A robust signal preprocessing framework for wrist pulse analysis.* Biomedical Signal Processing and Control, 2016. **23**: p. 62-75.
45. Dianguo, C., L. Liu, and W. Peng. *Removing Baseline Drift in Pulse Waveforms by a Wavelet Adaptive Filter*. in *Bioinformatics and Biomedical Engineering, 2008. ICBBE 2008. The 2nd International Conference on*. 2008.
46. Valens, C., *The fast lifting wavelet transform.* NEC researcher index, 1999.
47. Gilat, A. and V. Subramaniam, *Numerical methods for engineers and scientists: an introduction with applications using MATLAB*. Vol. 2nd. 2011, Hoboken, N.J: Wiley.
48. Xu, L., et al. *Adaptive baseline wander removal in the pulse waveform*. in *Computer-Based Medical Systems, 2002. (CBMS 2002). Proceedings of the 15th IEEE Symposium on*. 2002.
49. Auckland, T.U.o. *Morphological Operators*. 2015 [cited 2016 21st January 2016]; Available from: https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm.
50. Playboy, *Lena*. 1972, Playboy Magazine.
51. Zheng-Gang, G., et al. *Pulse wave signal baseline wander elimination using morphology*. in *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*. 2010.
52. Rilling, G., P. Flandrin, and P. Goncalves. *On empirical mode decomposition and its algorithms*. in *IEEE-EURASIP workshop on nonlinear signal and image processing*. 2003. NSIP-03, Grado (I).
53. Blanco-Velasco, M., B. Weng, and K.E. Barner, *ECG signal denoising and baseline wander correction based on the empirical mode decomposition.* Computers in Biology and Medicine, 2008. **38**(1): p. 1-13.
54. Zheng, Y., Y. Zhang, and Y. Liu. *The Research of Pulse Wave Signal Denosing Based on EMD and ICA*.

55. Jia-Ju, L., et al. *An effective photoplethysmography signal processing system based on EEMD method*. in *VLSI Design, Automation and Test (VLSI-DAT), 2015 International Symposium on*. 2015.
56. EC57, A.-A., *Testing and reporting performance results of cardiac rhythm and ST segment measurement algorithms.* Association for the Advancement of Medical Instrumentation, Arlington, VA, 1998.
57. O'Haver, T. *Peak Finding and Measurement*. A Pragmatic Introduction to Signal Processing 2015 December 2015 [cited 2016 4th Janurary 2016]; Available from: http://terpconnect.umd.edu/~toh/spectrum/PeakFindingandMeasurement.htm.
58. Billauer, E., *peakdet: Peak detection using MATLAB*. 2012.
59. Chen, W., et al. *Study on conditioning and feature extraction algorithm of photoplethysmography signal for physiological parameters detection*. in *Image and Signal Processing (CISP), 2011 4th International Congress on*. 2011.
60. Karlen, W., J.M. Ansermino, and G. Dumont. *Adaptive pulse segmentation and artifact detection in photoplethysmography for mobile applications*. in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*. 2012.
61. Aboy, M., et al., *An automatic beat detection algorithm for pressure signals.* Biomedical Engineering, IEEE Transactions on, 2005. **52**(10): p. 1662-1670.
62. Liangyou, C., A.T. Reisner, and J. Reifman. *Automated beat onset and peak detection algorithm for field-collected photoplethysmograms*. in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*. 2009.
63. Shin, H.S., C. Lee, and M. Lee, *Adaptive threshold method for the peak detection of photoplethysmographic waveform.* Comput. Biol. Med., 2009. **39**(12): p. 1145-1152.
64. Basili, V.R., *The Experimental Paradigm in Software Engineering*, in *Proceedings of the International Workshop on Experimental Software Engineering Issues: Critical Assessment and Future Directions*. 1993, Springer-Verlag. p. 3-12.
65. Shin, W., Y.D. Cha, and G. Yoon, *ECG/PPG integer signal processing for a ubiquitous health monitoring system.* Journal of Medical Systems, 2010. **34**(5): p. 891-898.
66. Elgendi, M., M. Jonkman, and F. De Boer. *Heart Rate Variability Measurement using the Second Derivative Photoplethysmogram*. in *BIOSIGNALS*. 2010.
67. Lyons, R.G., *Window Design Method*, in *Understanding digital signal processing*. 1997, Addison Wesley Pub. Co: Reading, Mass. p. 175-191.

68. Kuo, S.M., et al., *Real-Time Constraints*, in *Real-time digital signal processing: fundamentals, implementations and applications*. 2013, Wiley: Chichester, West Sussex, UK. p. 15-16.
69. Cormen, T.H., *33.1 Line-segment properties*, in *Introduction to algorithms*. 2001, MIT Press: Cambridge, Mass;London;.
70. Allen, J. and A. Murray, *Comparison of three arterial pulse waveform classification techniques.* Journal of Medical Engineering & Technology, 1996. **20**(3): p. 109-114.
71. Wagner, P., et al. *Path planning and tracking for robots based on cubic hermite splines in real-time*. in *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*. 2010.
72. Davis, T., et al., *Evaluators and NURBS*, in *OpenGL programming guide: the official guide to learning OpenGL, release 1*. 1993, Addison-Wesley: Reading, Mass.
73. N.Jesse. *Spline Interpolation*. 2013 [cited 2016; Available from: https://play.google.com/store/apps/details?id=de.njesse.android.interpolation&hl=en.
74. SunnyDay. *GraphIt*. 2014 [cited 2016; Available from: https://play.google.com/store/apps/details?id=com.sunnyday.development.graphit.
75. Michael, R. *Arduino Splines*. 2015 [cited 2016; Available from: https://github.com/kerinin/arduino-splines.
76. Slomer, D. *Cubic Splines*. 2008 2nd February 2016 [cited 2016; Available from: https://education.ti.com/en/us/activity/detail?id=6107320EBBDA4D92B474CC3102314ED3.
77. Holden, P. *Develop FFT apps on low-power MCUs*. 2005 19th October 2005 [cited 2016 2nd February 2016]; Available from: http://www.embedded.com/design/prototyping-and-development/4025598/Develop-FFT-apps-on-low-power-MCUs.
78. Holme, A. *GPU FFT*. 2014 [cited 2016 2nd February 2016]; Available from: http://www.aholme.co.uk/GPU_FFT/Main.htm.
79. Upton, E. *Accelerating Fourier Transforms Using the GPU*. 2014 30th January 2014 [cited 2016 2nd February 2016]; Available from: https://www.raspberrypi.org/blog/accelerating-fourier-transforms-using-the-gpu/.
80. Skiena, S., *The Algorithm Design Manual*. 2008, Springer-Verlag.
81. Hyde, R., *13.8. The Relative Cost of Arithmetic Operations*, in *Write Great Code, Volume 2 : Thinking Low-Level, Writing High-Level*. 2006, No Starch Press: San Francisco, US. p. 436-437.

Appendix A Visual Characteristics of R.B.D. Samples

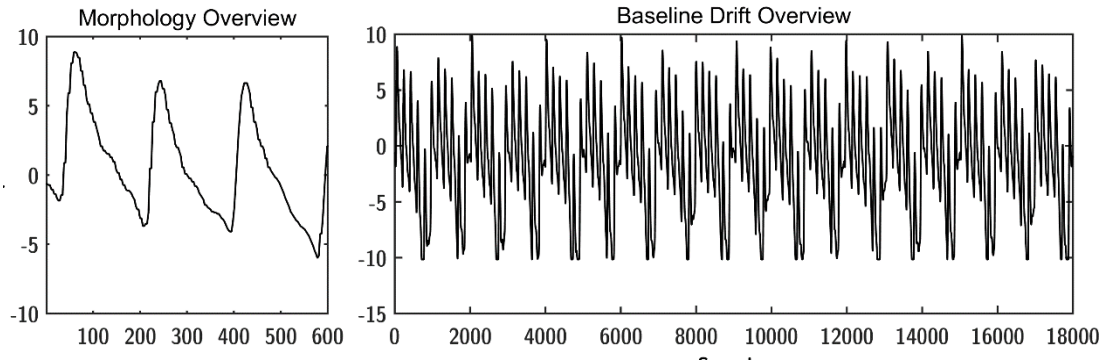


Figure A.1. *Sample 0009_8min Preview*

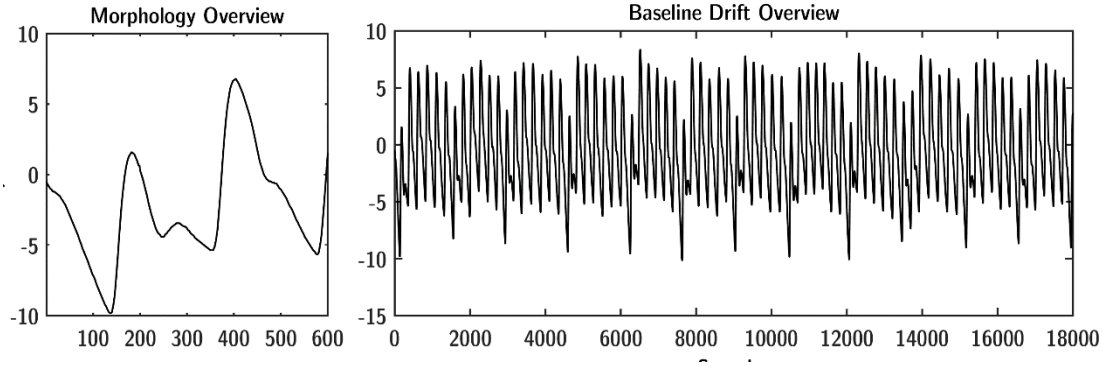


Figure A.2. *Sample 0028_8min Preview*

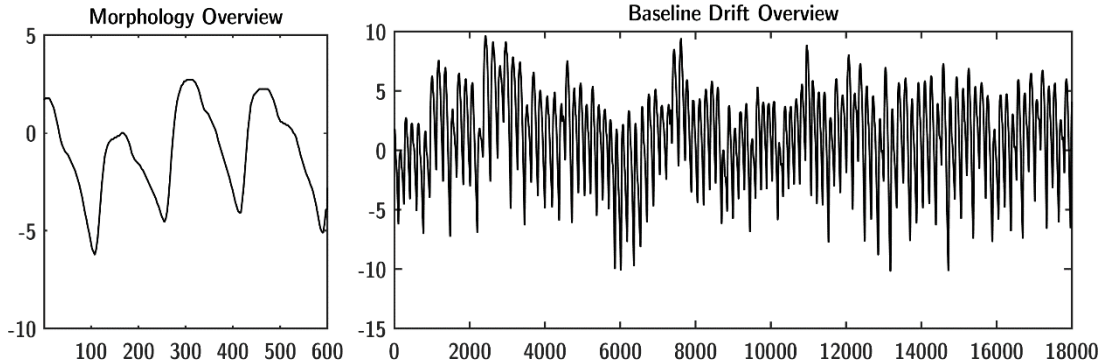


Figure A.3. *Sample 0030_8min Preview*

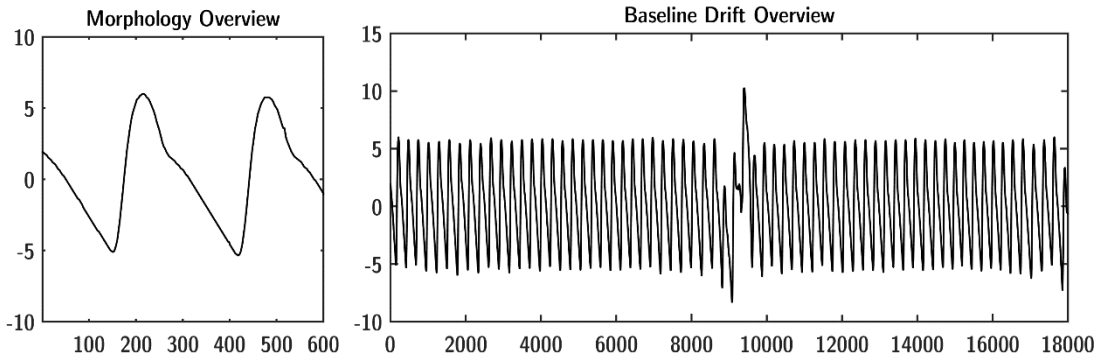


Figure A.4. *Sample 0031_8min*

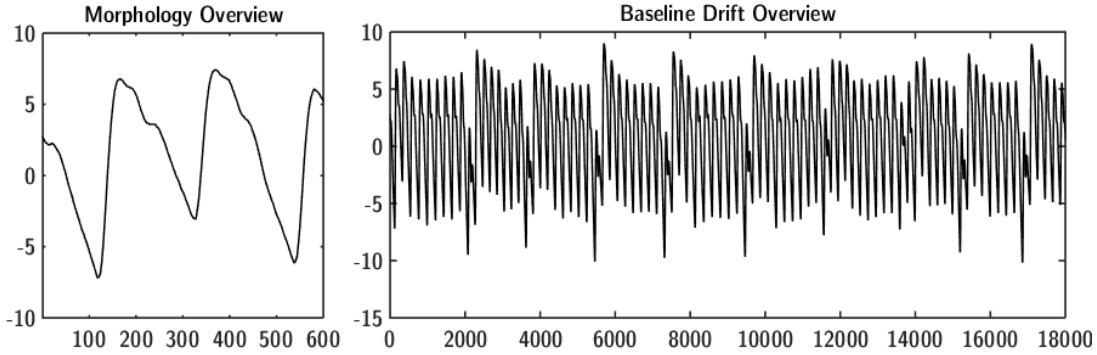


Figure A.5. *Sample 0032_8min*

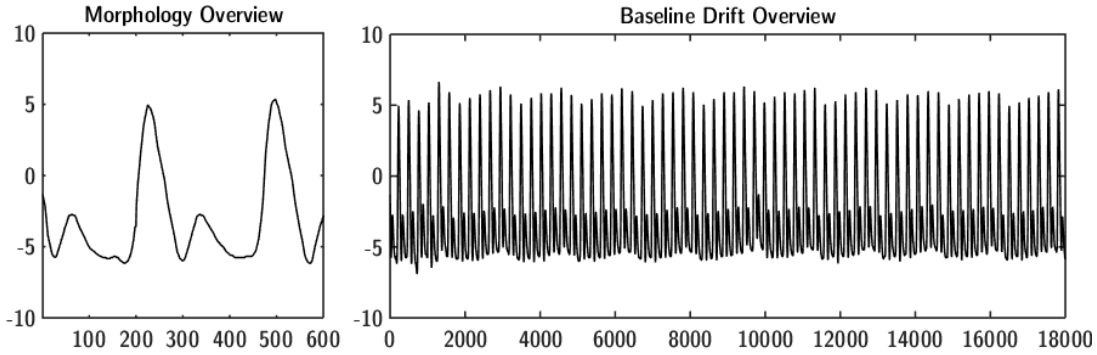


Figure A.6. *Sample 0147_8min*

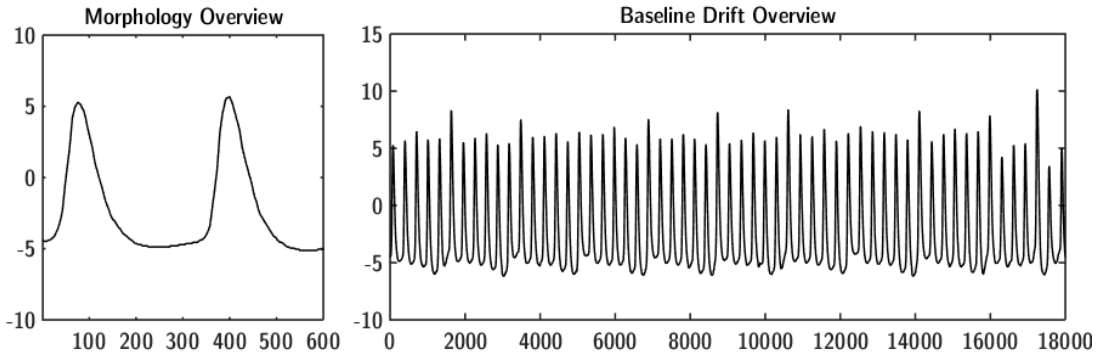


Figure A.7. *Sample 0149_8min*

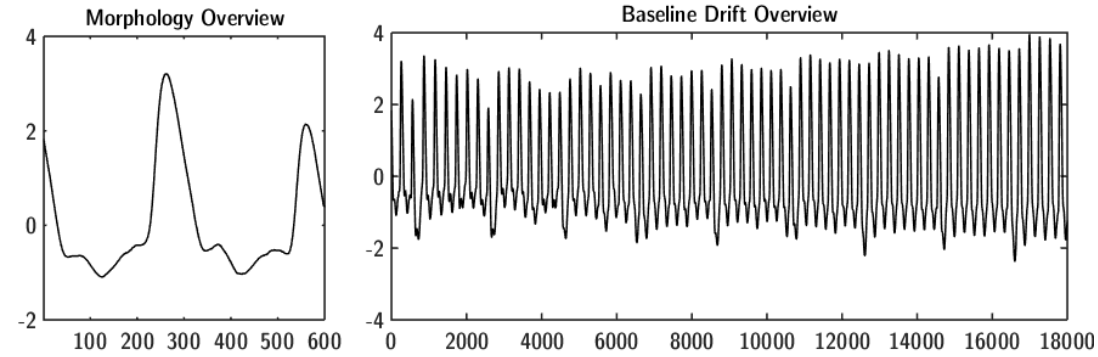


Figure A.8. *Sample 0309_8min*

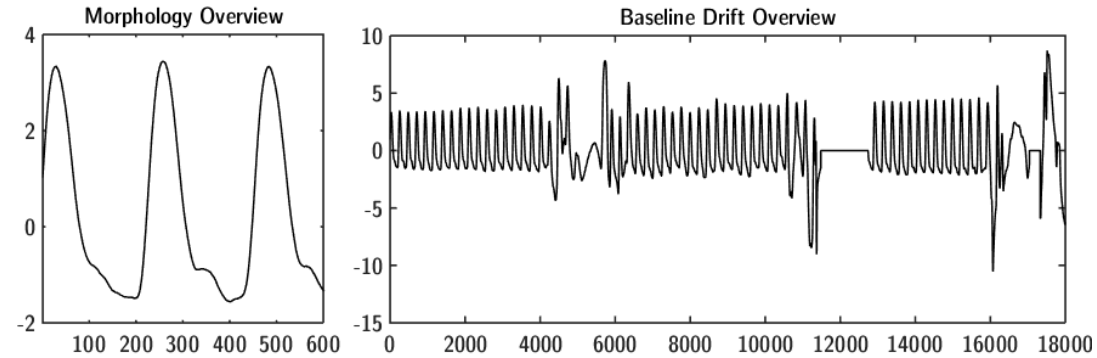


Figure A.9. *Sample 0370_8min*

Appendix B Softening Filter Results

B1. Softening Correlation Tests and Run-times

B1.1Wavelet Filter Results

Table B.1. *Wavelet timing results and scales used*

	Short Test (20 Seconds)	Long Tests(60 Seconds)	Wavelet Scale Used
Sample 0009	0.024984	0.048158	7
Sample 0028	0.036618	0.073718	8
Sample 0030	0.027214	0.041344	8
Sample 0031	0.032972	0.043678	8
Sample 0032	0.025455	0.042983	8
Sample 0147	0.025578	0.044105	8
Sample 0149	0.026882	0.047707	8
Sample 0309	0.02958	0.048215	8
Sample 0370	0.02655	0.04484	8
Average	0.028425889	0.048305333	

B1.2. IIR-ZPF Results

Table B.2. *Results for IIR-ZPF short-period tests*

	Timing	Approximation Correlation	Approximation RMSE	Result Correlation	Result RMSE
Sample 0009	0.000818	0.92131	1.278929	0.923421	1.278929
Sample 0028	0.000567	0.981695	0.239047	0.997667	0.239047
Sample 0030	0.000577	0.990967	0.282768	0.995196	0.282768
Sample 0031	0.000543	0.934758	0.055587	0.999386	0.16787
Sample 0032	0.000586	0.984234	0.27977	0.996874	0.27977
Sample 0147	0.0005	0.957619	0.101723	0.999514	0.101723
Sample 0149	0.000529	0.953689	0.153591	0.999046	0.153591
Sample 0309	0.000533	0.975446	0.055834	0.998925	0.055834
Sample 0370	0.000505	0.972899	0.199029	0.994547	0.199029
Average	0.000573111	0.963624111	0.294030889	0.989397333	0.306506778

Table B.3. *Results for IIR-ZPF long-period tests*

	Timing	Approximation Correlation	Approximation RMSE	Result Correlation	Result RMSE
Sample 0009	0.003477	0.914504	1.335143	0.918314	1.335143
Sample 0028	0.001071	0.971555	0.326048	0.995759	0.326048
Sample 0030	0.001761	0.988636	0.261589	0.996166	0.261589
Sample 0031	0.001264	0.98999	0.117168	0.999384	0.117168
Sample 0032	0.001319	0.963292	0.393515	0.993639	0.393515
Sample 0147	0.001229	0.96149	0.084273	0.999663	0.084273
Sample 0149	0.001302	0.970791	0.137808	0.999247	0.137808
Sample 0309	0.001087	0.962366	0.056181	0.999224	0.056181
Sample 0370	0.001252	0.97601	0.243853	0.992494	0.243853
Average	0.001529111	0.966514889	0.328397556	0.98821	0.328397556

B1.3. FIR-Direct Convolution Results

Table B.4. *Results for FIR Direct Convolution short-period tests*

	Timing	Approximation Correlation	Approximation RMSE	Result Correlation	Result RMSE
Sample 0009	0.001199	0.955193	1.314627	0.91972	1.314627
Sample 0028	0.001196	0.974174	0.357963	0.994775	0.357963
Sample 0030	0.001192	0.975916	0.47869	0.98629	0.47869
Sample 0031	0.001211	0.918784	0.06472	0.999504	0.160568
Sample 0032	0.001747	0.970705	0.485674	0.990582	0.485674
Sample 0147	0.001189	0.936456	0.126977	0.999264	0.126977
Sample 0149	0.001187	0.960231	0.08459	0.997535	0.08459
Sample 0309	0.001928	0.937257	0.190368	0.998556	0.190368
Sample 0370	0.001334	0.970893	0.274067	0.989624	0.274067
Average	0.001353667	0.955512111	0.375297333	0.986205556	0.385947111

Table B.5. *Results for FIR Direct Convolution long-period tests*

	Timing	Approximation Correlation	Approximation RMSE	Result Correlation	Result RMSE
Sample 0009	0.003114	0.953422	1.357852	0.915931	1.357852
Sample 0028	0.003038	0.852377	1.07874	0.932686	1.07874
Sample 0030	0.003956	0.975214	0.409759	0.9906	0.409759
Sample 0031	0.003029	0.974419	0.246243	0.9973	0.246243
Sample 0032	0.003053	0.957999	0.526258	0.98861	0.526258
Sample 0147	0.003036	0.953791	0.097211	0.999556	0.097211
Sample 0149	0.003198	0.959037	0.072459	0.998718	0.072459
Sample 0309	0.003941	0.955439	0.201235	0.998404	0.201235
Sample 0370	0.004064	0.963557	0.370113	0.982789	0.370113
Average	0.003381	0.949472778	0.48443	0.978288222	0.48443

B1.4. FIR-FFT Convolution Results

Table B.6. *Results for FIR FFT Convolution short-period tests*

	Timing	Approximation Correlation	Approximation RMSE	Result Correlation	Result RMSE
Sample 0009	0.00047	0.939628	1.395088	0.910851	1.395088
Sample 0028	0.000466	0.965558	0.384116	0.993992	0.384116
Sample 0030	0.000463	0.971465	0.511828	0.984376	0.511828
Sample 0031	0.000432	0.896062	0.070721	0.999521	0.157195
Sample 0032	0.000479	0.962283	0.515819	0.989423	0.515819
Sample 0147	0.000464	0.935104	0.131296	0.999207	0.131296
Sample 0149	0.000496	0.919093	0.206979	0.99829	0.206979
Sample 0309	0.0005	0.947045	0.091544	0.997116	0.091544
Sample 0370	0.000482	0.95472	0.303794	0.987291	0.303794
Average	0.000472444	0.943439778	0.401242778	0.984451889	0.410851

Table B.7. *Results for FIR FFT Convolution long-period tests*

	Timing	Approximation Correlation	Approximation RMSE	Result Correlation	Result RMSE
Sample 0009	0.00175	0.937786	1.437717	0.907099	1.437717
Sample 0028	0.001725	0.958896	0.455127	0.991752	0.455127
Sample 0030	0.001946	0.970183	0.439846	0.989189	0.439846
Sample 0031	0.00228	0.962053	0.268773	0.996783	0.268773
Sample 0032	0.002407	0.944714	0.561895	0.987062	0.561895
Sample 0147	0.001754	0.933381	0.113342	0.999395	0.113342
Sample 0149	0.00179	0.939179	0.217127	0.998134	0.217127
Sample 0309	0.001622	0.945108	0.078055	0.99851	0.078055
Sample 0370	0.001833	0.949038	0.404432	0.979546	0.404432
Average	0.001900778	0.948926444	0.441812667	0.983052222	0.441812667

B1.5. Overlap-Save Algorithm Results

Table B.8. *Results for FIR OLS short-period tests*

	Timing (Per Segment)	Timing (Total)	Approximation Correlation	Approximation RMSE	Result Correlation	Result RMSE
Sample 0009	0.0000728	0.002194	0.95401	1.328514	0.917947	1.328514
Sample 0028	0.0000646	0.004258	0.974191	0.357812	0.994779	0.357812
Sample 0030	0.0000760	0.003099	0.975932	0.478638	0.986294	0.478638
Sample 0031	0.0000641	0.00283	0.918557	0.064808	0.999505	0.160435
Sample 0032	0.0000718	0.00282	0.970852	0.485252	0.990599	0.485252
Sample 0147	0.0000736	0.00316	0.937028	0.126584	0.999268	0.126584
Sample 0149	0.0000914	0.003239	0.937117	0.190415	0.998555	0.190415
Sample 0309	0.0000907	0.00402	0.960184	0.084614	0.997533	0.084614
Sample 0370	0.0000756	0.003992	0.970741	0.274356	0.989602	0.274356
Average	0.0000728	0.00342725	0.955401333	0.376777	0.986009111	0.387402222

Table B.9. *Results for FIR OLS long-period tests*

	Timing (Per Segment)	Timing (Total)	Approximation Correlation	Approximation RMSE	Result Correlation	Result RMSE
Sample 0009	0.0000692	0.008579	0.953406	1.35789	0.915927	1.35789
Sample 0028	0.0000739	0.009163	0.968266	0.424459	0.992809	0.424459
Sample 0030	0.0000745	0.009236	0.975219	0.409714	0.990602	0.409714
Sample 0031	0.0000750	0.009301	0.974392	0.246301	0.997298	0.246301
Sample 0032	0.0000670	0.008303	0.957992	0.52627	0.98861	0.52627
Sample 0147	0.0000789	0.009788	0.953414	0.097561	0.999553	0.097561
Sample 0149	0.0000747	0.00926	0.955403	0.201264	0.998403	0.201264
Sample 0309	0.0000764	0.009471	0.958978	0.072485	0.998717	0.072485
Sample 0370	0.0000698	0.008652	0.963475	0.370353	0.982766	0.370353
Average	0.0000733	0.009083667	0.962282778	0.411810778	0.984965	0.411810778

B2. Cascaded Filter Correlation Tests

B2.1. Non Real-time FFCF vs. WCF

Table B.10. *Results for Non Real-time FFCF short-period tests*

	WCF Timing	FCCF Timing	Approximation Correlation	Approximation RMSE	Result Correlation	Result RMSE
Sample 0009	0.184558	0.079698	0.996455	0.286064	0.995925	0.286064
Sample 0028	0.218996	0.085418	0.994501	0.149713	0.999099	0.149713
Sample 0030	0.184914	0.087105	0.998176	0.132685	0.998966	0.132685
Sample 0031	0.217564	0.079721	0.923862	0.190458	0.999931	0.040748
Sample 0032	0.162285	0.076361	0.982469	0.312644	0.996275	0.312644
Sample 0147	0.161393	0.085301	0.973155	0.07541	0.999741	0.07541
Sample 0149	0.148577	0.079678	0.881099	0.326314	0.995725	0.326314
Sample 0309	0.154325	0.094781	0.958901	0.081866	0.997786	0.081866
Sample 0370	0.161203	0.086097	0.916637	0.434374	0.98022	0.434374
Average	0.177090556	0.083932583	0.958361667	0.221058667	0.995963111	0.204424222

Table B.11. *Results for Non Real-time FFCF long-period tests*

	WCF Timing	**FCCF Timing**	**Approximation Correlation**	**Approximation RMSE**	**Result Correlation**	**Result RMSE**
Sample 0009	0.480327	0.248557	0.983033	0.634259	0.979739	0.634259
Sample 0028	0.481181	0.266192	0.97378	0.341006	0.995369	0.341006
Sample 0030	0.531439	0.289945	0.900378	1.005114	0.946085	1.005114
Sample 0031	0.430706	0.292927	0.990035	0.115774	0.999411	0.115774
Sample 0032	0.447926	0.246573	0.989793	0.217118	0.998057	0.217118
Sample 0147	0.408202	0.268463	0.963565	0.078153	0.999717	0.078153
Sample 0149	0.384416	0.275967	0.934615	0.2289	0.997996	0.2289
Sample 0309	0.412766	0.275302	0.970959	0.079889	0.998478	0.079889
Sample 0370	0.406747	0.255824	0.898351	0.861986	0.93966	0.861986
Average	0.442634444	0.268861111	0.956056556	0.395799889	0.983834667	0.395799889

B2.2. Real-time FFCF vs. WCF

Table B.12. *Results for Real-time FFCF short-period tests*

Samples	Baseline Correlation	Baseline RMSE	Final Correlation	Final RMSE	Filtering Time Mean	Detection Time Mean	CBSI Time Mean
0009_8min	0.969995	0.846504	0.964249	0.846979	0.000127	0.009313	0.001108
0028_8min	0.987268	0.234406	0.997727	0.238025	0.000151	0.008935	0.001085
0030_8min	0.987768	0.375031	0.99161	0.378068	0.000125	0.009706	0.001016
0031_8min	0.908678	0.195983	0.999761	0.073981	0.000134	0.009321	0.001091
0032_8min	0.967327	0.423284	0.992998	0.425606	0.00013	0.00878	0.001004
0147_8min	0.974779	0.07222	0.999721	0.080539	0.000129	0.010436	0.001085
0149_8min	0.881654	0.328529	0.995621	0.330425	0.000129	0.009664	0.001129
0309_8min	0.922774	0.110222	0.995946	0.110868	0.000132	0.009416	0.001216
0370_8min	0.955405	0.353292	0.986628	0.35365	0.000143	0.009982	0.001059
Average	0.950627556	0.326607889	0.991585	0.315349	0.000134125	0.00953	0.0010856
Average Time Per Segment		0.01074975					

Table B.13. *Results for Real-time FFCF long-period tests*

Samples	Baseline Correlation	Baseline RMSE	Final Correlation	Final RMSE	Filtering Time Mean	Detection Time Mean	CBSI Time Mean
0009_8min	0.980526	0.689565	0.976215	0.689765	0.000126	0.009189	0.00138
0028_8min	0.981328	0.297115	0.996448	0.298037	0.000133	0.008762	0.001136
0030_8min	0.986911	0.353981	0.992908	0.355007	0.000123	0.00863	0.001102
0031_8min	0.990294	0.124047	0.999309	0.125223	0.000134	0.008487	0.00117
0032_8min	0.98862	0.23476	0.997705	0.23593	0.000129	0.008824	0.00108
0147_8min	0.959487	0.080775	0.999675	0.083886	0.000137	0.008573	0.001111
0149_8min	0.923713	0.287505	0.996803	0.288098	0.000129	0.009872	0.001066
0309_8min	0.968259	0.083713	0.998317	0.084018	0.000133	0.009756	0.001117
0370_8min	0.841949	1.005062	0.920938	1.005104	0.000132	0.009539	0.001115
Average	0.957898556	0.350724778	0.9864797	0.351674	0.00013125	0.00905537	0.00111212
Average Time Per Segment		0.01029875					

B3. Visual Results for the Cascaded Filters

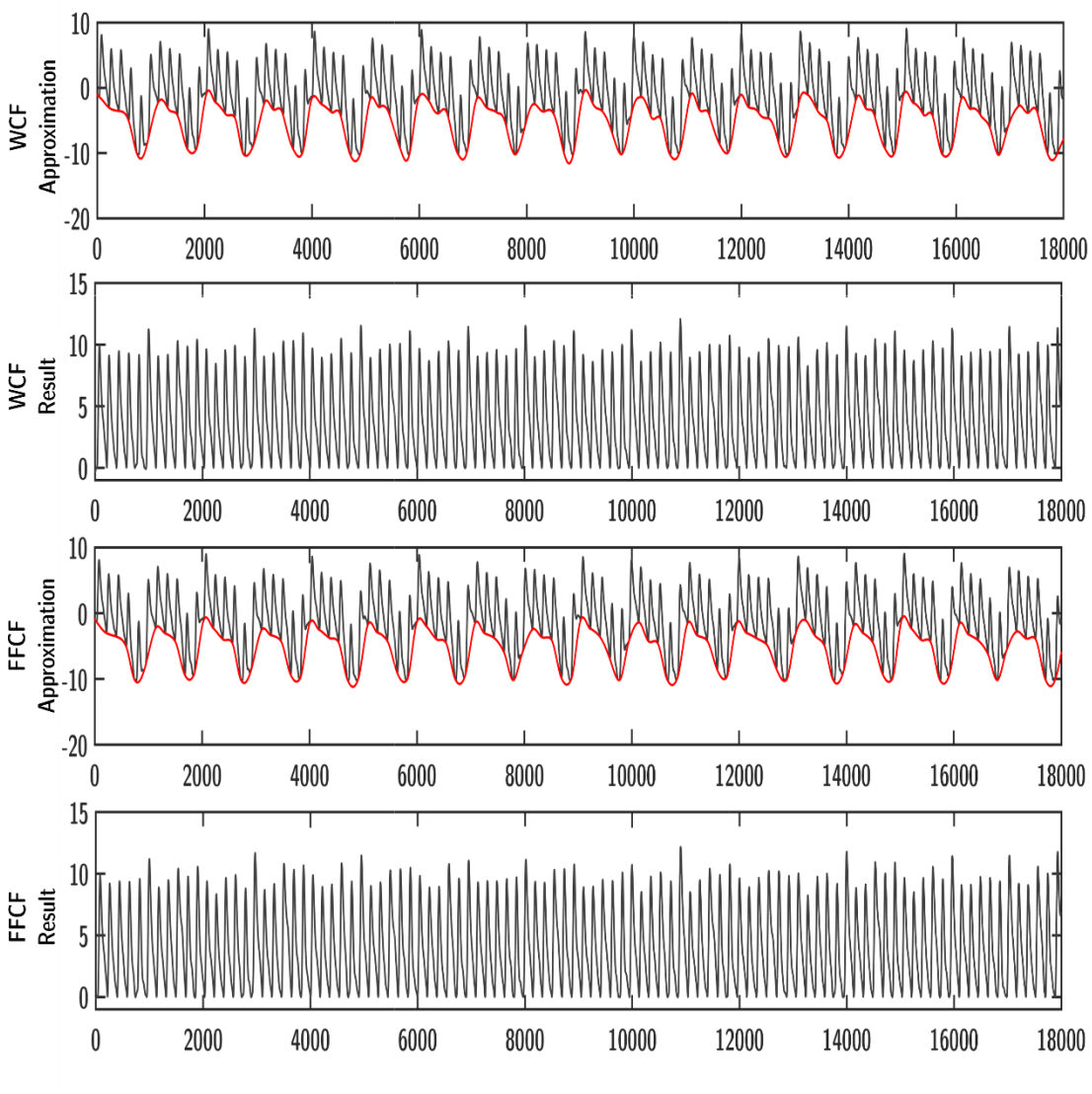


Figure B.1. *Sample 0009_8min WCF and FFCF filtering results*

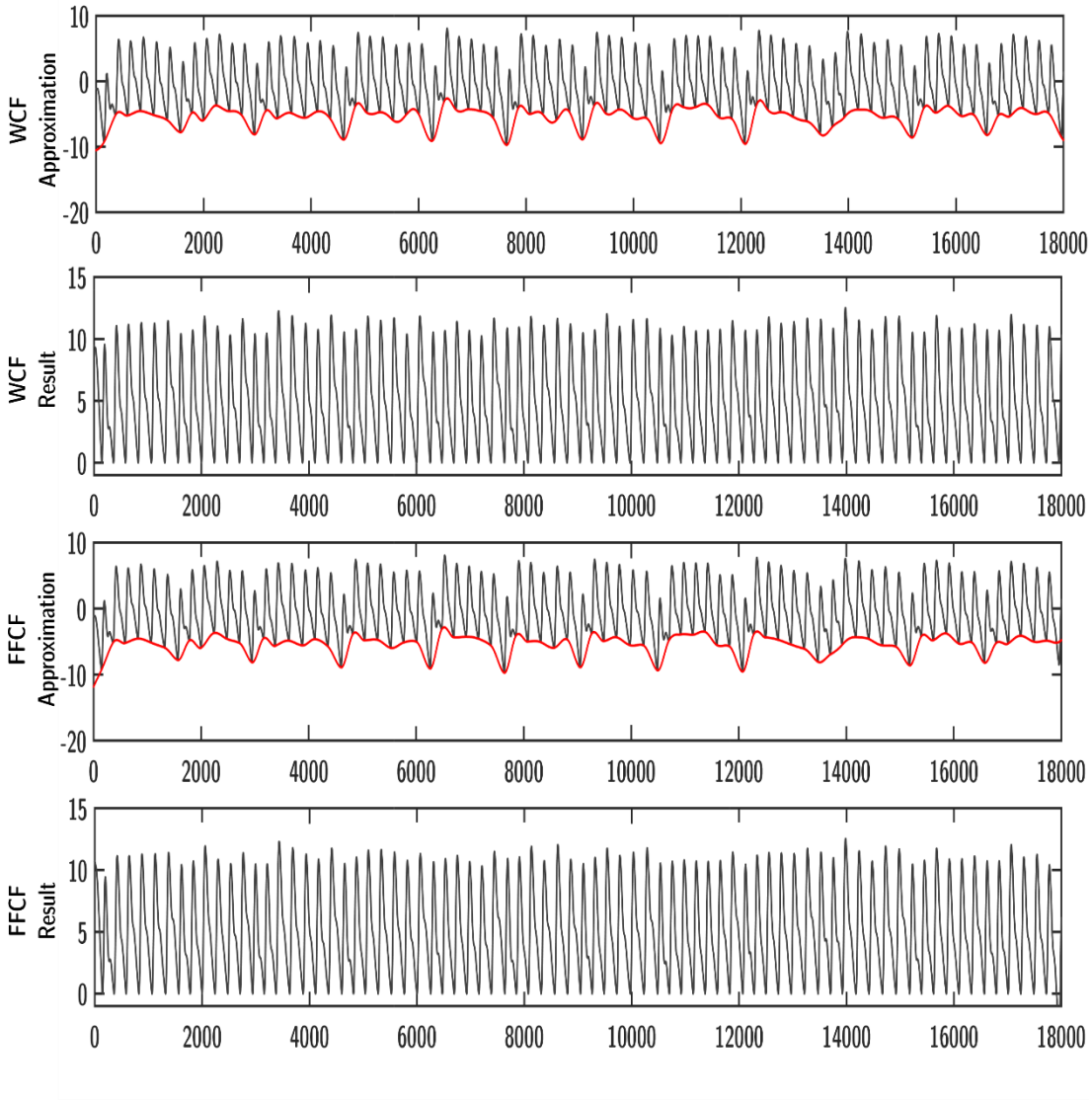


Figure B.2. *Sample 0028_8min WCF and FFCF filtering results*

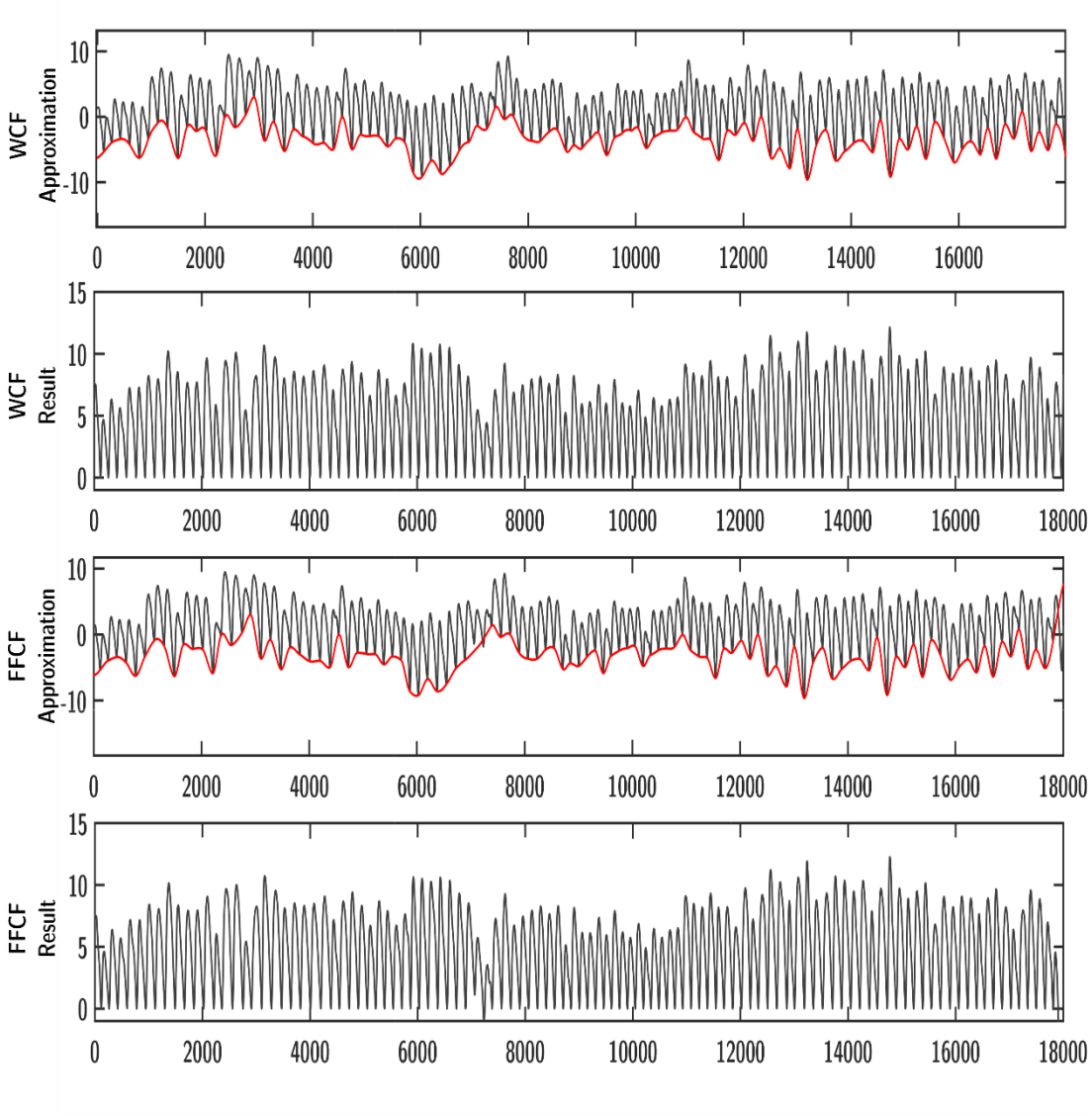


Figure B.3. *Sample 0030_8min WCF and FFCF filtering results*

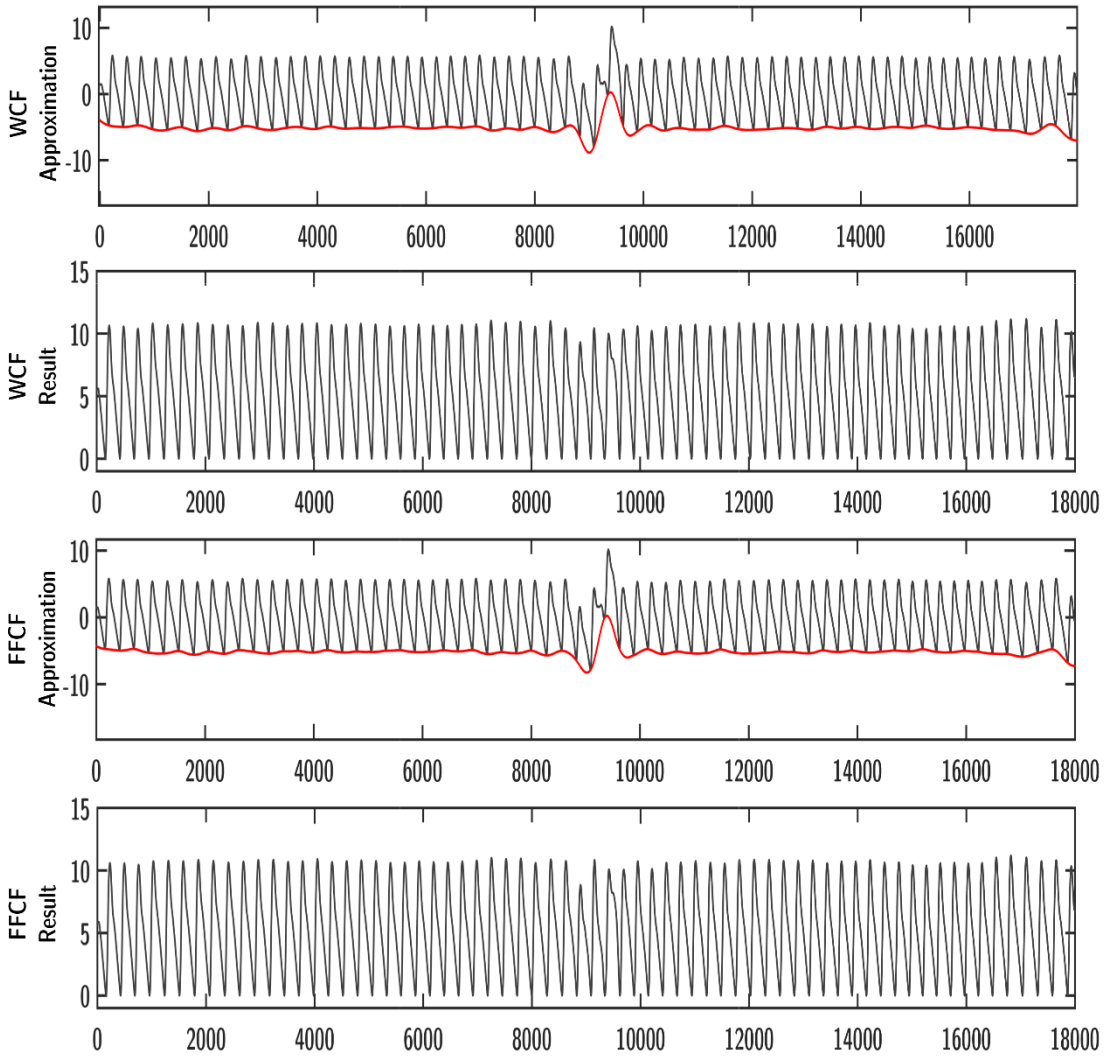


Figure B.4. *Sample 0031_8min WCF and FFCF filtering results*

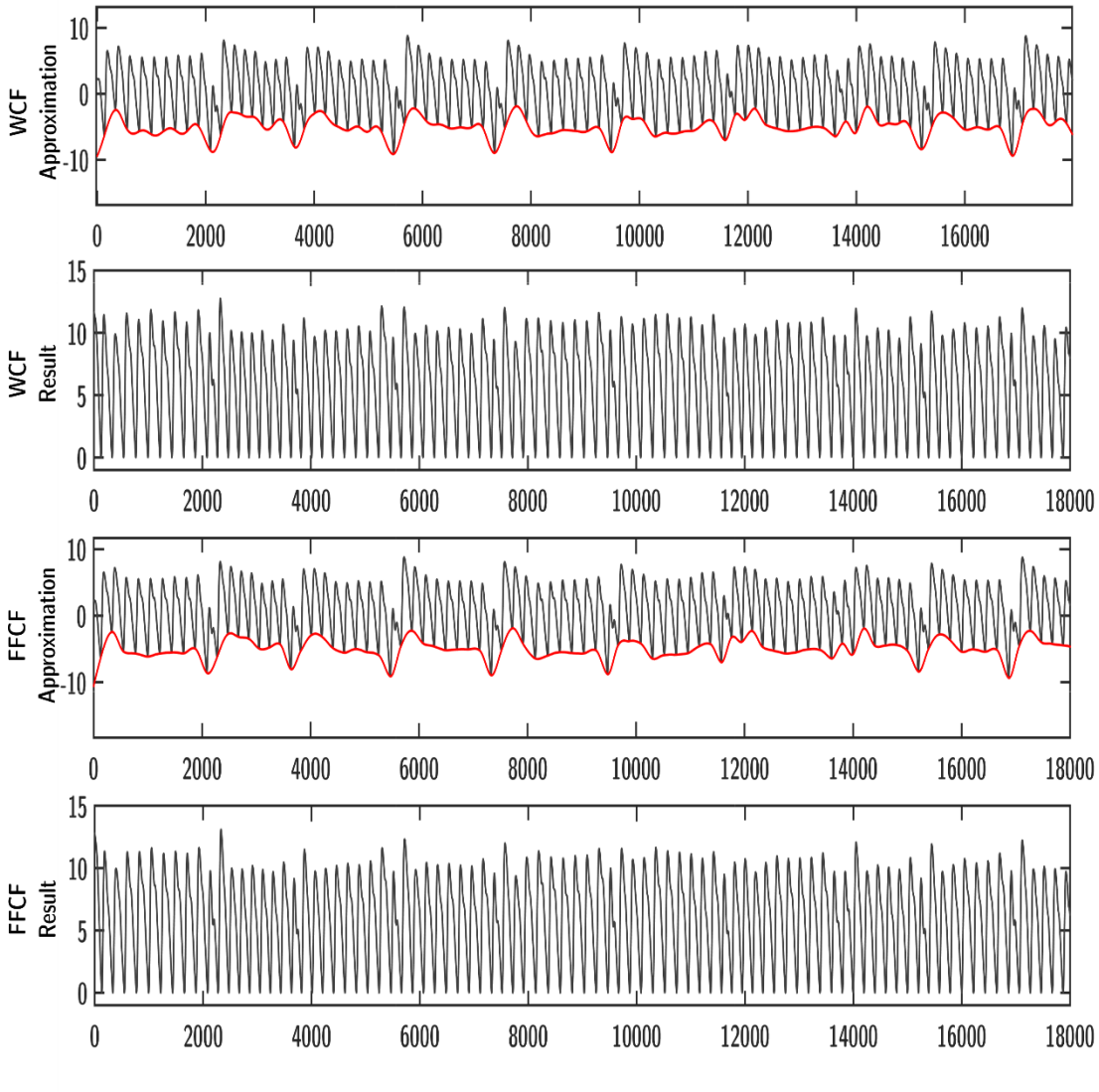


Figure B.5. *Sample 0032_8min WCF and FFCF filtering results*

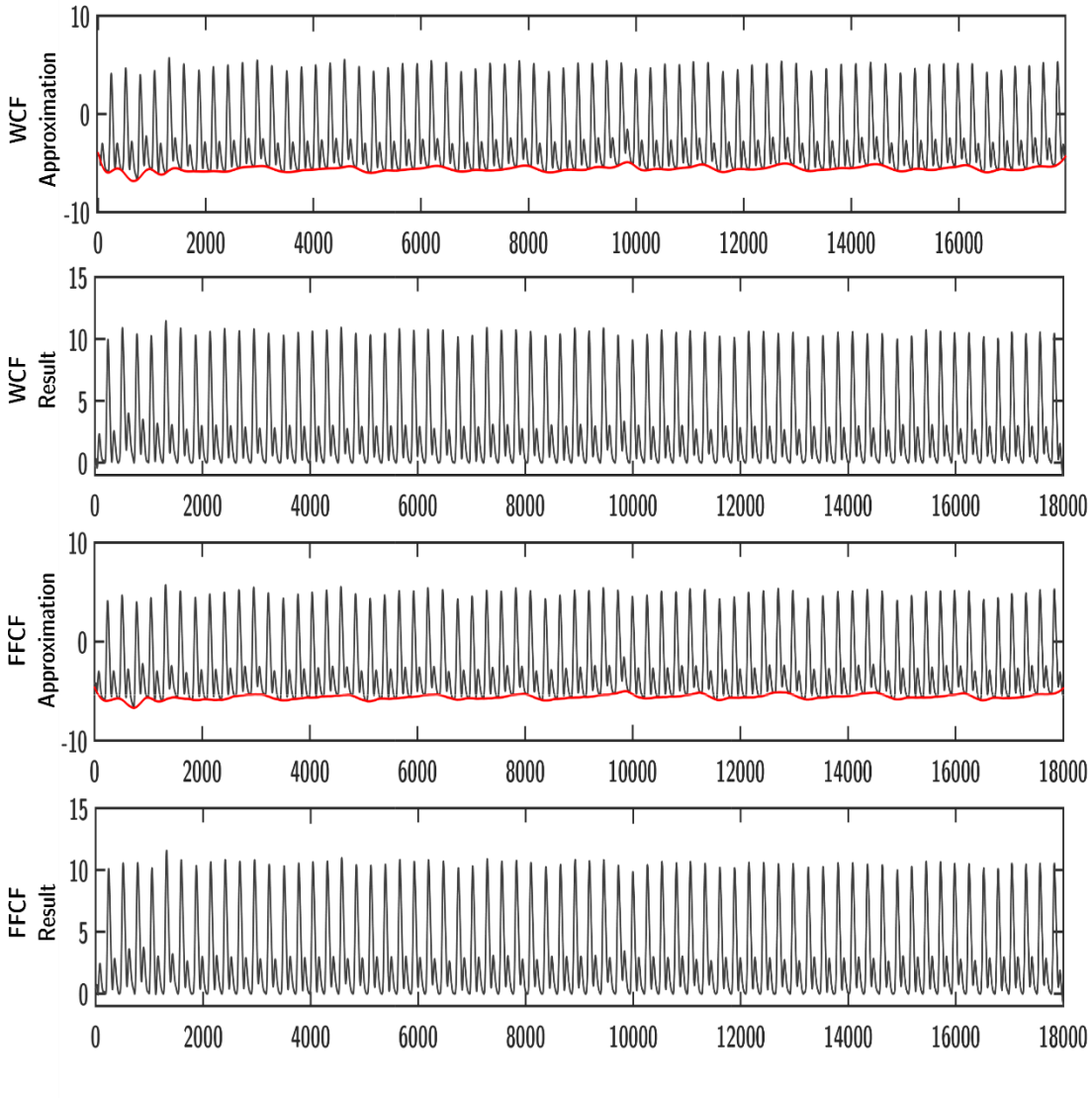


Figure B.6. *Sample 0147_8min WCF and FFCF filtering results*

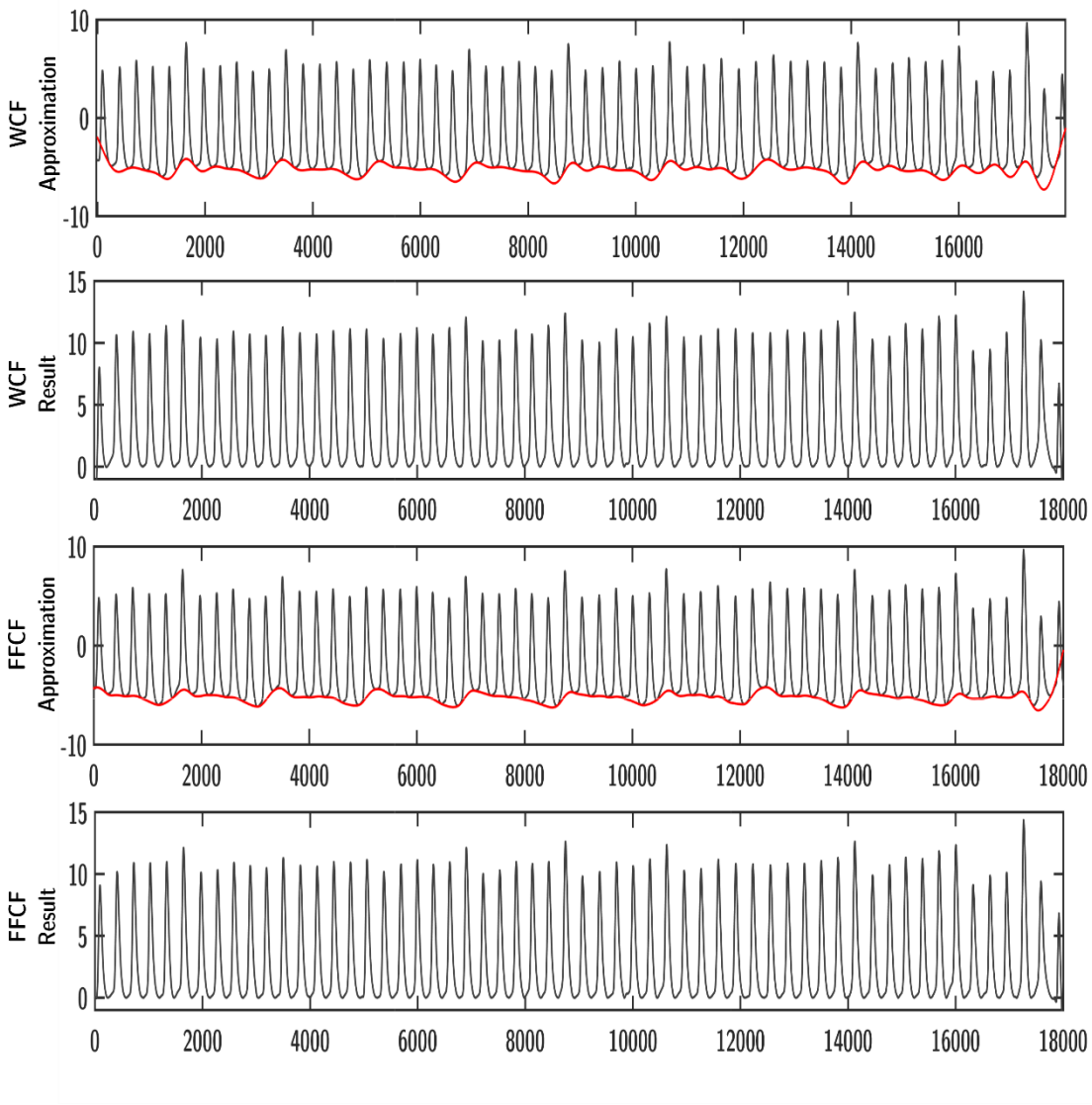


Figure B.7. *Sample 0149_8min WCF and FFCF filtering results*

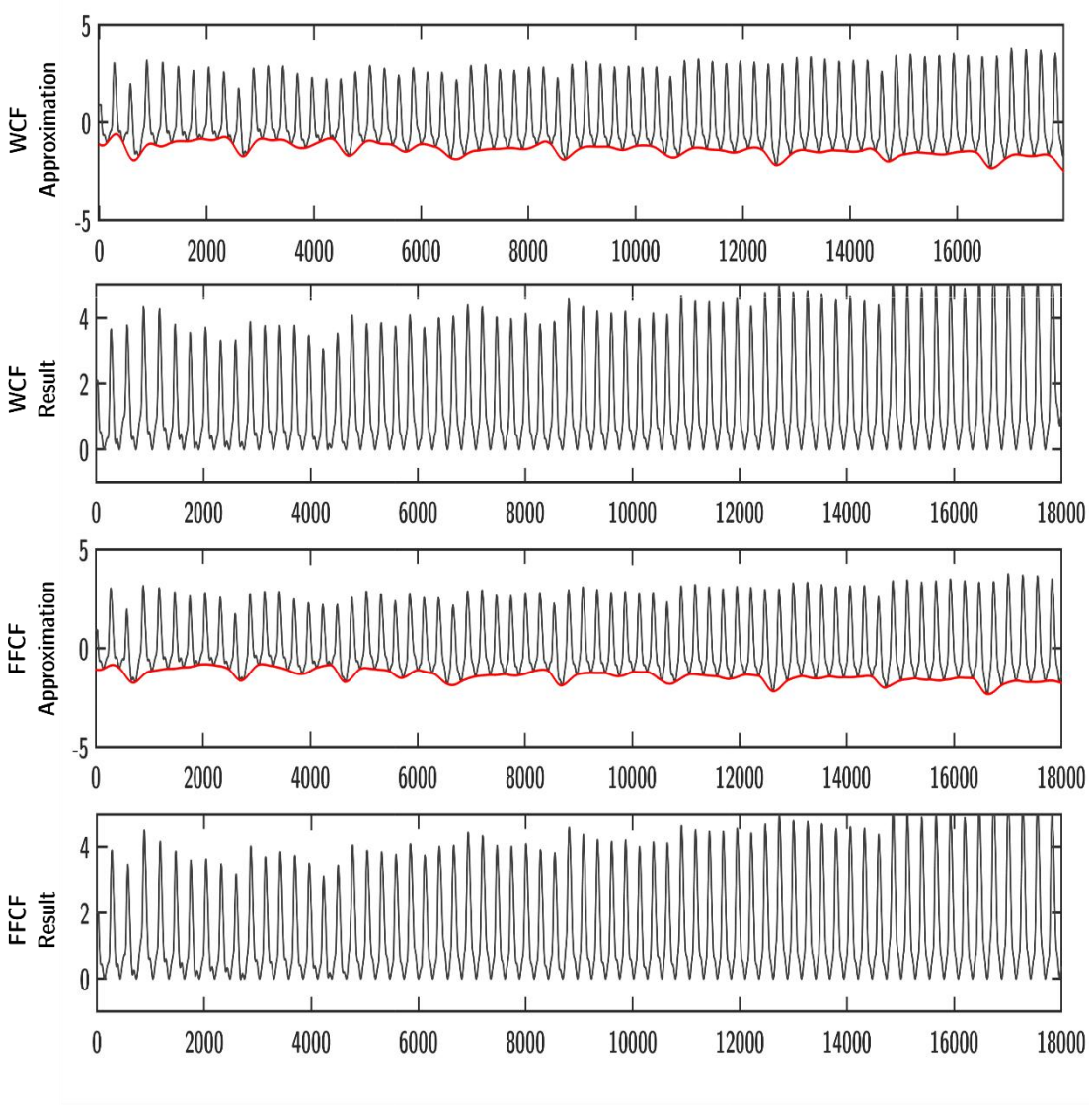


Figure B.8. *Sample 0309_8min WCF and FFCF filtering results*

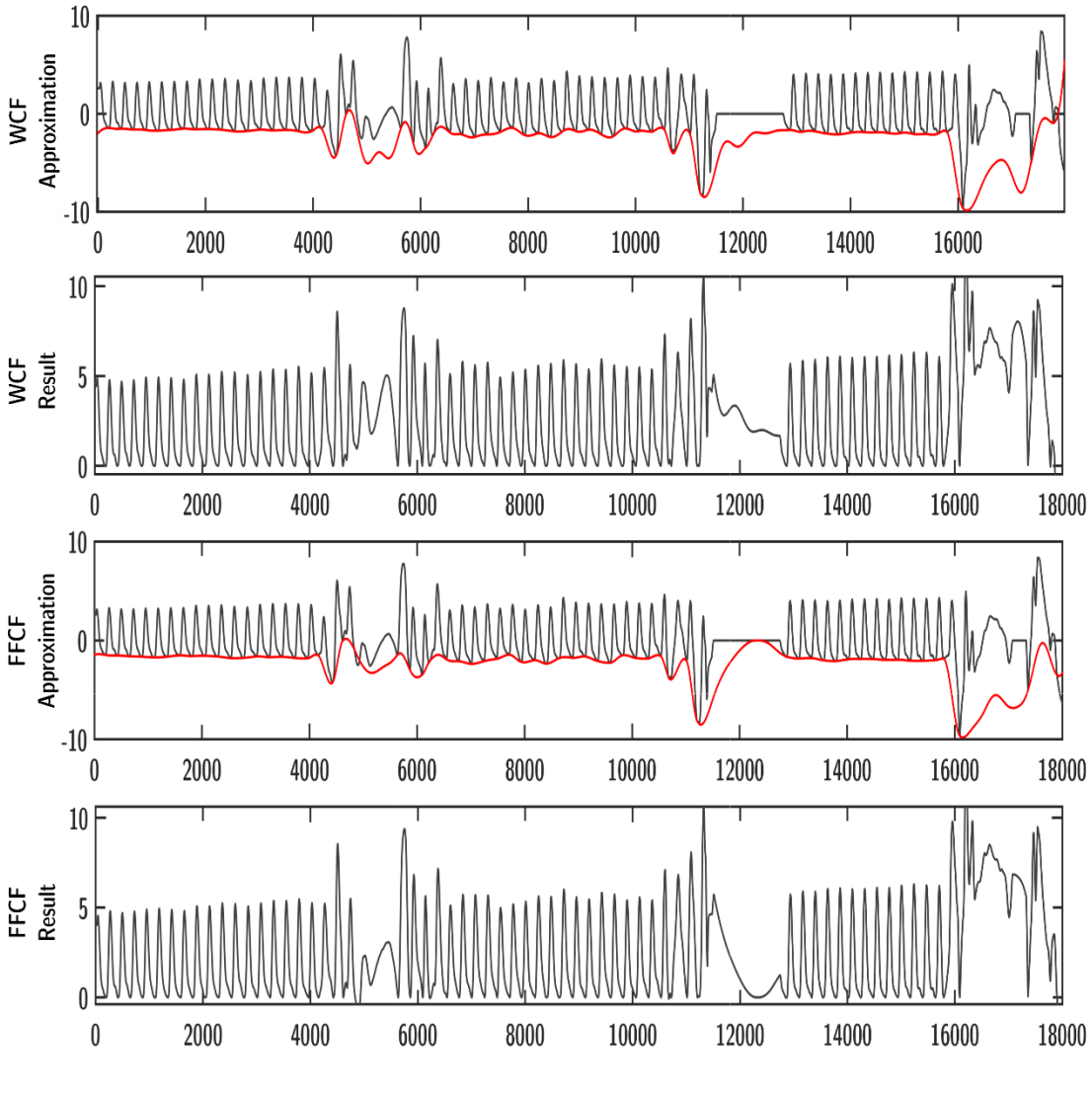


Figure B.9. *Sample 0370_8min WCF and FFCF filtering results*

Appendix C Detection Results

C1. Xu et al. 2007's DCBD Algorithm Results

Table C.1. *Short–period tests for Xu et al. 2007's DCBD algorithm*

Samples	Actual Peaks	Detected Peaks	Peak-TP	Peak-FP	Peak-FN	Actual Valleys	Detected Valleys	Valley-TP	Valley-FP	Valley-FN	Time
0009_8min	33	33	33	0	0	33	33	33	0	0	0.15801
0028_8min	25	25	25	0	0	25	25	25	0	0	0.18068
0030_8min	34	34	34	0	0	34	34	34	0	0	0.15598
0031_8min	22	22	22	0	0	22	22	22	0	0	0.18274
0032_8min	29	29	29	0	0	29	29	29	0	0	0.13512
0147_8min	22	22	22	0	0	22	22	22	0	0	0.13371
0149_8min	20	20	20	0	0	19	20	19	1	0	0.12012
0309_8min	21	21	21	0	0	21	21	21	0	0	0.12317
0370_8min	22	22	22	1	0	21	22	21	1	0	0.13305
Total	228	228	228	1	0	226	228	226	2	0	0.14695
Peak Sensitivity		100%	**Valley Sensitivity**			100%					
Peak Accuracy		99.56%	**Valley Accuracy**			99.12%					

Table C.2. *Long–period tests for Xu et al. 2007's DCBD algorithm*

Samples	Actual Peaks	Detected Peaks	Peak-TP	Peak-FP	Peak-FN	Actual Valleys	Detected Valleys	Valley-TP	Valley-FP	Valley-FN	Time
0009_8min	100	100	100	0	0	99	100	100	0	0	0.42918
0028_8min	76	76	76	0	0	76	76	76	0	0	0.40405
0030_8min	108	108	108	0	0	109	109	109	0	0	0.48697
0031_8min	67	67	67	0	0	67	67	67	0	0	0.38417
0032_8min	87	87	87	0	0	87	87	87	0	0	0.40223
0147_8min	66	66	66	0	0	66	66	66	0	0	0.36120
0149_8min	58	58	58	0	0	57	51	51	0	0	0.33379
0309_8min	66	66	66	0	0	66	66	66	0	0	0.36157
0370_8min	60	67	60	7	0	60	67	60	7	0	0.35888
Total	688	695	688	7	0	687	689	682	7	0	0.39134
Peak Sensitivity		100%	**Valley Sensitivity**			100%					
Peak Accuracy		98.99%	**Valley Accuracy**			98.98%					

C2. Kan et al. 2012's SBD Algorithm Results

Table C.3. *Short–period tests for Kan et al. 2012's SBD algorithm*

Samples	Actual Peaks	Detected Peaks	Peak-TP	Peak-FP	Peak-FN	Actual Valleys	Detected Valleys	Valley-TP	Valley-FP	Valley-FN	Time
0009_8min	33	33	33	0	0	33	33	33	0	0	0.220771
0028_8min	25	25	25	0	0	25	25	25	0	0	0.18512
0030_8min	34	34	34	0	0	34	34	34	0	0	0.227445
0031_8min	22	22	22	0	0	22	22	22	0	0	0.175985
0032_8min	29	29	29	0	0	29	29	29	0	0	0.184218
0147_8min	22	22	22	0	0	22	22	22	0	0	0.254628
0149_8min	20	19	19	0	1	19	19	19	0	1	0.190085
0309_8min	21	21	21	0	0	21	21	21	0	0	0.256078
0370_8min	22	21	21	1	1	21	21	21	1	1	0.18985
Total	228	226	226	1	2	226	226	226	1	2	0.207926125
Peak Sensitivity		99.12%	**Valley Sensitivity**			99.56%					
Peak Accuracy		99.12%	**Valley Accuracy**			99.56%					

Table C.4. *Long–period tests for Kan et al. 2012's SBD algorithm*

Samples	Actual Peaks	Detected Peaks	Peak -TP	Peak- FP	Peak -FN	Actual Valleys	Detected Valleys	Valley- TP	Valley -FP	Valley -FN	Time
0009_8min	100	100	100	0	0	99	100	100	1	0	0.619128
0028_8min	76	76	76	0	0	76	76	76	0	0	0.521351
0030_8min	108	106	106	0	2	109	106	106	0	3	0.600053
0031_8min	67	67	67	0	0	67	67	67	0	0	0.455803
0032_8min	87	87	87	0	0	87	87	87	0	0	0.611986
0147_8min	66	66	66	0	0	66	66	66	0	0	0.772928
0149_8min	58	57	57	0	1	57	57	57	0	0	0.523546
0309_8min	66	66	66	0	0	66	66	66	0	0	0.582658
0370_8min	60	66	60	9	0	60	69	66	9	0	1.478641
Total	688	691	685	9	3	687	694	691	10	3	0.69337075
Peak Sensitivity		99.56%	**Valley Sensitivity**			99.57%					
Peak Accuracy		98.70%	**Valley Accuracy**			98.57%					

C3. The Adaptive Threshold Detection Algorithm Results

Table C.5. *Short–period tests for the ADT algorithm*

Samples	Actual Peaks	Detected Peaks	Peak-TP	Peak-FP	Peak-FN	Actual Valleys	Detected Valleys	Valley-TP	Valley-FP	Valley-FN	Time
0009_8min	33	33	33	0	0	33	30	30	0	3	5.16974
0028_8min	25	25	25	0	0	25	25	25	0	0	4.62515
0030_8min	34	34	34	0	0	34	29	29	0	5	4.51339
0031_8min	22	22	22	0	0	22	22	22	0	0	4.62893
0032_8min	29	29	29	0	0	29	29	29	0	0	4.52157
0147_8min	22	22	22	0	0	22	22	22	0	0	4.71292
0149_8min	20	20	20	0	0	19	19	19	0	0	3.85950
0309_8min	21	21	21	0	0	21	21	21	0	0	5.00125
0370_8min	22	25	22	3	0	21	23	20	1	2	3.89576
Total	228	231	228	3	0	226	220	217	1	10	4.46981
Peak Sensitivity		100 %	**Valley Sensitivity**			95.59%					
Peak Accuracy		98.70%	**Valley Accuracy**			99.54%					

Table C.6. *Long–period tests for the ADT algorithm*

Samples	Actual Peaks	Detected Peaks	Peak-TP	Peak-FP	Peak-FN	Actual Valleys	Detected Valleys	Valley-TP	Valley-FP	Valley-FN	Time
0009_8min	100	100	100	0	0	100	92	92	0	8	16.5696
0028_8min	76	76	76	0	0	76	74	74	0	2	16.9889
0030_8min	108	108	108	1	0	109	99	99	0	10	14.9848
0031_8min	67	68	67	1	0	67	67	0	0	0	15.9502
0032_8min	87	87	87	0	0	87	87	87	0	0	16.8804
0147_8min	66	66	66	0	0	66	66	66	0	0	16.4824
0149_8min	58	58	58	0	0	57	57	57	0	0	13.4630
0309_8min	66	66	66	0	0	66	66	66	0	0	14.3853
0370_8min	60	72	59	11	1	60	64	60	4	0	10.5094
Total	688	701	687	13	1	688	672	601	4	20	15.1349
Peak Sensitivity		99.85%	**Valley Sensitivity**			96.78%					
Peak Accuracy		99.14%	**Valley Accuracy**			99.34%					

C4. The ADS Algorithm Results

Table C.7. *Short–period tests for the ADS algorithm*

Samples	Actual Peaks	Detected Peaks	Peak-TP	Peak-FP	Peak-FN	Actual Valleys	Detected Valleys	Valley-TP	Valley-FP	Valley-FN	Time
0009_8min	33	33	33	0	0	33	33	33	0	0	0.07582
0028_8min	25	25	25	0	0	25	25	25	0	0	0.07962
0030_8min	34	34	34	0	0	34	34	34	0	0	0.08199
0031_8min	22	22	22	0	0	22	22	22	0	0	0.07529
0032_8min	29	29	29	0	0	29	29	29	0	0	0.07187
0147_8min	22	22	22	0	0	22	22	22	0	0	0.08034
0149_8min	20	20	20	0	0	19	20	19	1	0	0.07474
0309_8min	21	21	21	0	0	21	21	21	0	0	0.08906
0370_8min	22	22	22	0	0	21	22	22	1	0	0.08041
Total	228	228	228	0	0	226	228	227	2	0	0.07879
Peak Sensitivity		100%	**Valley Sensitivity**			100%					
Peak Accuracy		100%	**Valley Accuracy**			99.12%					

Table C.8. *Long–period tests for the ADS algorithm*

Samples	Actual Peaks	Detected Peaks	Peak-TP	Peak-FP	Peak-FN	Actual Valleys	Detected Valleys	Valley-TP	Valley-FP	Valley-FN	Time
0009_8min	100	100	100	0	0	100	100	100	0	0	0.23717
0028_8min	76	76	76	0	0	76	76	76	0	0	0.25419
0030_8min	108	106	106	0	2	109	106	106	0	2	0.27796
0031_8min	67	67	67	0	0	67	67	67	0	0	0.28069
0032_8min	87	87	87	0	0	87	87	87	0	0	0.23521
0147_8min	66	66	66	0	0	66	66	66	0	0	0.25454
0149_8min	58	58	58	0	0	57	58	57	1	0	0.26371
0309_8min	66	66	66	0	0	66	66	66	0	0	0.26252
0370_8min	60	61	60	1	0	60	61	60	1	0	0.24456
Total	688	687	686	1	2	688	687	685	2	2	0.25673
Peak Sensitivity		99.71%	**Valley Sensitivity**			99.71%					
Peak Accuracy		99.85%	**Valley Accuracy**			99.71%					

Appendix D Big-O and Algorithm Complexity

The Big-O notation is utilized as a method of determining an algorithm's overall complexity and efficiency, while abstracting the overall specific details of operations required *[80]*. The complexity of an algorithm correlates to its efficiency within time consumption along with a series of other factors such as mathematical operations and input constraints. The time complexity of an algorithm is determined through a series of scales *(described in Table D1 from most to least optimal top to bottom)* representing the computation growth time.

Table D.1. *Common Big-O notation scales*

Growth Rate	***Description***
O(1)	Instantaneous
O(n)	Linear time growth
O(log n)	Logarithmic time growth
O($\mathbf{n^c}$)	Exponential time growth

For functions that have only a single operation such as addition the time complexity would be *O(1)* representing instantaneous time, whereas for a function that has a loop process the complexity would be *O(n)* representing linear time growth. Nested loop processes would produce a time complexity of $O(n^c)$ since the number of operations grow exponentially fast, and in algorithms such as divide and conquer algorithms where the time complexity grows slowly, as the number of operations required grow larger, *O(log n)* is utilized.

Using an arbitrary algorithm function as an example for analysis, here we can see that it has one assignment operation which is constant, one outer loop and one nested loop comprised of two loops.

Table D.2. *The example algorithm function*

```
Function FoobarAlgorithm(x)
    run-condition = x * 2

    if run-condition == 10 then
        for i = 1:10 do
            print(i*2)
        end-for
    else-if run-condition == 20 then
        for i = 1:10 do
            for j = 1:10 do
                print(j*i*2)
            end-for
        end-for
    else
        return
    end-if-else
end-function
```

The computation time complexity based on this structure would be *O(1)* for the assignment, *O(n)* since we have one loop process, and $O(n^2)$ for the nested loop process. The maximum time consumption for this algorithm is $O(n^2)$, but it can also be *O(n)* if the conditions aren't met to execute the nested loop. In the best case scenario where neither conditions are met the algorithm concludes with a time complexity of *O(1)*.

The structural flow of the algorithm however, is not the only determining factor to an algorithm's efficiency. Not all arithmetic operations used by the algorithms are equal. Integer operations such as addition and subtraction for example will yield faster computation times, whereas floating point operations such as divide and multiple will yield slower computation times due to the complexity of evaluating the results. On the analysis of computation times and efficiency in regards to

arithmetic operations, we can trace an algorithm's efficiency to the cycles required and the arithmetic operations that is required. While speed and operation efficiency may vary from processor to processor and compiler to compiler optimization, the general consensus of arithmetic operation costs on a CPU are described as the following in Table D.3.

Table D.3. *Order of arithmetic costs (Table referenced from Hyde, 2009 [81])*

Fastest	Integer Addition, Subtraction, Negation, logical operators (OR, NOR, NOT, AND)
	Logical Shifts
	Logical Rotates
	Multiplication
	Division
	Floating-point comparisons
	Floating-point addition/subtraction
	Floating-point multiplication
Slowest	Floating-point division

With consideration to this aspect, depending on the algorithm's sub-routines and their activation conditions, there may be at times that an algorithm with similar or greater time complexities, will be faster than the one with the assumed faster time complexity.