6-2012

Removal of electrocardiographic corruption from electromyographic signals using a localized wavelet based approach.

Nihit Bajaj
University of Louisville

REMOVAL OF ELECTROCARDIOGRAPHIC CORRUPTION FROM ELECTROMYOGRAPHIC SIGNALS USING A LOCALIZED WAVELET BASED APPROACH

By:
Nihit Bajaj
B.S. Bioengineering, University of Louisville, May 14th 2011

A Thesis
Submitted to the Faculty of the
University of Louisville
J.B. Speed School of Engineering
As Partial Fulfillment of the Requirements
For the Professional Degree

MASTER OF ENGINEERING

Department of Bioengineering
June 2012

REMOVAL OF ELECTROCARDIOGRAPHIC CORRUPTION FROM ELECTROMYOGRAPHIC SIGNALS USING A LOCALIZED WAVELET BASED APPROACH

Submitted by:____________________________________

Nihit Bajaj

A Thesis Approved On

____

(Date)

by the Following Reading and Examination Committee:

____

Ayman El-Baz, Ph.D. Thesis Director

____

Alexander Ovechkin, M.D., Ph.D.

____

Palaniappan Sethu, Ph.D.

____

Olfa Nasraoui, Ph.D.

ACKNOWLEDGMENTS

I would like to personally thank Matthew Nitzken for mentoring, teaching, and contributing in this massive project. If we didn’t have multiple approaches in solving the hurdles that we encountered, I don’t think we would have this finished and refined product. I want to thank Dr. Ayman S. El-Baz for mentoring me throughout the project, and pushing me to keep investing the time it required to complete the project on time. I would also like to thank Sevda Aslan from Frazier Rehab Institute for collecting numerous patient data, consistently communicating physiological aspects of the project, and for initiating the development of the idea behind the localization of filter. Finally, I want to thank Dr. Alexander Ovechkin for his instrumental support throughout the project. His expertise in physiological aspects of the removal process was critical in pushing the algorithm to become clinical friendly. All of these people had a significant contributing role in the completion of this project, and I cannot thank them enough for all that they have done.

In terms of personal motivation, I want to thank my parents, Umesh Bajaj, and Neelu Bajaj, for providing me the support and time I needed to complete this thesis research. I cannot thank them enough for all that they have provided me in my five years of college. My sister, Mehak Bajaj, has always provided me positive encouragement and feedback needed to complete this project on time. Finally, I want to grudgingly thank my twin brother, Udit Bajaj, for being my competitive motivation to complete the thesis, and graduate on time.

ABSTRACT

Introduction: Approximately 12,000 new cases of spinal cord injury (SCI) are reported each year in the US. Currently, the most widely used method of assessing the recovery of voluntary capability after spinal cord injury is the American Spinal Injury Association Impairment Scale (AIS). However, this test is not objective and is not sensitive enough to detect small activities. Recent studies have been using surface electromyography (EMG) to develop objective and sensitive spinal cord injury (SCI) characterization protocols. EMG recordings from the trunk muscles are contaminated with the electrical activity of the heart (ECG, electrocardiography). Depending on the level, spinal cord injury may disrupt the control of the trunk muscles, and EMG recordings from these muscles will be very weak compared to those in non-injured individuals. The elimination of ECG artifacts play critical role in precise evaluation of the trunk muscles in these individuals. While common global digital filters may generally remove some of the ECG corruption from the signal, these filters also remove or alter valuable EMG signal, which makes the physiological importance of these signals irrelevant.

Methods: Local filtering approach was developed to remove this ECG noise, without significantly altering the EMG signal. The local filtering approach uses externally recorded ECG signals, in a separate lead configuration, as a mask to locate the area of ECG spikes within the noisy EMG signal. The areas of the signal containing the ECG noise are decomposed into 128 sub-wavelets using custom-scaled Morlet Wavelet Transform. Sub-wavelets pertaining to ECG within the signal at the ECG spike location are then removed, and the signal is reconstructed to create a clean EMG signal. This process is analytically tested for robustness and accuracy, using customized validation metrics, on simulated phantom signals. It is compared with a global Morlet Wavelet filter that does not localize its filtering process on the ECG spikes.

Results: It was found that the localized filtering significantly reduced the Root-Mean-Squared (RMS) of the area of the signal containing ECG spikes. The Localized Filter also significantly reduced the error produced from removal of EMG signal in the areas outside of ECG spikes compared to global filter. The proposed local filter doesn't degrade the signal, even at low ECG amplitudes (~60% improvement), compared to the global filter, which destroys the signal at this low amplitude ECG (-100% improvement). The proposed local filter is also far more efficient at removing larger amplitude ECG (more critical) than the global filter, which has a narrow range of signals that it can efficiently remove ECG. Hence, the proposed local filter is more robust and clinical-ready than the global filter.

Conclusion: Proposed approach is far superior in terms of ECG removal accuracy, and introduction of artifact error from processing, compared to comparable global filter. It provides a mean to improve analysis of EMG signals as a tool to assess recovery from SCI.

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

I. INTRODUCTION

Approximately 12,000 new cases of spinal cord injury (SCI) are reported each year in the United States (DeVivo 2010). SCI is a severe condition in which the neural elements of a spinal cord are injured. This type of injury generally results in a loss of functional processes in the areas of sensation and motor function. Loss of motor function typically results in muscular paralysis below the injury site (Thomas, Zaidner et al. 1997). Muscular paralysis is one of the primary reasons that recovery from a spinal cord injury is difficult. Currently, the most widely used method of assessing the recovery of voluntary capability after spinal cord injury is the American Spinal Injury Association Impairment Scale (AIS) (Bartolo, Roberts et al. 1996). The AIS scale is a subjective grading rubric. The physician determines area of movement at certain limbs, and provides a grade based on the control of the patient. Not only is this test highly subjective (magnitude of movement is not defined, different physicians characterize movement or recovery in different manner, etc.), it is a highly inaccurate tool for assessment. Specifically, this test is not objective and is not sensitive enough to detect small muscular activities. If the patient is recovering some sort of control by slightly twitching muscles, or starting a firing pattern of motor neurons, this form of recovery is neglected by the scale. In order to properly diagnose and assess the degree of muscular paralysis and to help treat the injury it is necessary to analyze the patients' Electromyogram signals. Electromyogram signals (EMG) are electrical signal that are recorded from active muscles. Specifically, EMG detects the electrical potential generated by the muscle cells during their contraction and relaxation phases. In order to evaluate EMG signals, it is crucial to be able to effectively quantify them, such as determining their power spectrum (von

Tscharner 2000; Clancy, Bouchard et al. 2001). Quantifying the EMG signals provides a new analytical way to assess recovery.

EMG signals that are recorded in close proximity to the heart may experience large spikes of signal contamination caused by the heart. Majority of the respiratory muscles, and other muscles close to the spinal injury site, which are essential in assessing recovery, are also part of this group of contaminated muscles. The electrical signals that the heart produces are known as Electrical Cardiogram signals (ECG). ECG interference can significantly affect the recordings of EMG signals and makes analysis and quantification difficult, if not impossible. While numerous studies have attempted to digitally eliminate ECG spikes from EMG signals, the problem has proved a challenging one.

Gating is a common technique used for removing ECG spikes from EMG signals by using a brute force removal of the area within an EMG signal where ECG spikes are found to be visually present (Redfern, Hughes et al. 1993). There are many simple to advanced forms of gating procedures, however all of them generally end up in contaminated and choppy signal. Drake and Callagan (Drake and Callaghan 2006) used this technique, and combined a high-pass filter with a Butterworth filter to digitally subtracting raw ECG from the EMG signal. This technique, however, creates additional noise in the signal by removing part of the signal along with the ECG spike. Additionally it removes the additional parts of the EMG signal that are encompassed within the gate. The removal of critical signal portions is especially problematic in areas where active EMG contraction is occurring with ECG interference due to the fact that high-pass filters generally attempt to remove low-frequency ECG spikes at 1-20 Hz (Schweitzer, Fitzgerald et al. 1979; Zschorlich 1989; van Boxtel 2001; Rodríguez-Carreño, Malanda-Trigueros et al. 2006). The frequency range of ECG spikes is also set-up for debate, as there is a lot of overlap between ECG and EMG frequency spectrum. Hence, simple frequency-based filters are generally not effective in removing these spikes.

An alternative method is to simultaneously record ECG from a separate lead configuration alongside EMG signals. The separate raw ECG recording is then either directly subtracted from the contaminated EMG (Hof 2009), or the raw ECG is indirectly used as reference in advanced filtering and subtraction techniques (Bloch 1983; Yuancheng, Wolf et al. 2000; Clancy, Bouchard et al. 2001; Marque, Bisch et al. 2005; Lu, Brittain et al. 2009). The primary issue with this approach is that in nearly all cases the profile and amplitude of the ECG contamination noise found within EMG signals does not share the same shape as the raw ECG recording. This is due to the fact that the shape of the ECG interference depends on the location of the muscle being recorded and their position with respect to the heart. Hence, subtracting or removing ECG directly from a separate lead configuration generally leads to an introduction of additional noise, and tends to not be very accurate. Additionally the baseline of the ECG recordings introduces external noise.

Advanced mathematical models using Independent Component Analysis (ICA) have also been attempted to separate the ECG components from EMG signal. ICA is a statistical transformation that aims at decomposing an observed multi-dimensional vector into mutually independent source components (Hyvärinen, Karhunen et al. 2001). Costa Junior et al. attempted to use ICA to extract the ECG component from EMG signals at different orientations (Costa Junior, Ferreira et al. 2010). However, due to the overlapping complexity of the surrounding EMG signal, a precise extraction is not feasible. Hence, due to the global filtering approach, the envelope of EMG signal is considerably changed after the filtering process.

Carre and Leman proposed a method using un-decimated wavelet transforms in order to separate the contaminated EMG into sub-wavelets (Carre, Leman et al. 1998). They separated the Electrohystography (EHG), which is similar to EMG, using a Debauchies Discrete Wavelet Transform into approximate and detailed sub-components and performed a filtering on the individual components. The technique assumes that all ECG contamination can be accurately modeled using Gaussian noise, and attempts to filter the signals accordingly. In clinical applications this is typically not valid, especially in locations where the baseline is not flat, and the ECG spikes do not have distinct Gaussian noise characteristics. Similar to other techniques, it generally results in an over filtering of the EMG signal, or a failure to completely remove the ECG contamination. Similarly, techniques based on Multi Resolution Analysis (MRA), such as those used by Mallat and Zhou use Debauchies transforms to separate a signal into its subcomponents (Mallat 1989; Weidong and Gotman 2004). Filtering the sub-wavelets independently, has the effect of de-noising the high frequency EMG and ECG components.

Taelman et al. (Taelman, Van Huffel et al. 2007) and von Tscharner et al. (von Tscharner, Eskofier et al. 2011) proposed approaches based on wavelet-Independent Component Analysis. This technique attempts to find wavelets responsible for the ECG noise by using ICA on the wavelets produced from a standard Discrete Wavelet Transform (DWT). The wavelets responsible for ECG are removed, and the signal is reconstructed. While this approach minimizes the amount of signal degradation due to targeting specific wavelets, it still has an impact on sections of the signal that do not contain ECG noise, similar to many pure global wavelet removal techniques. It is also difficult to remove only noise when a small number of wavelets are used, which occurs when using a standard DWT, as often signal data is mixed with ECG noise corruption.

Overall, the current techniques suffer from the following problems: (i) Many of the techniques remove a significant amount of data in areas of the signal where ECG noise is not present, thus affecting the signal envelope; (ii) many of the techniques involve a form of global filtering that is universally applied to the signal; and (iii) template removal techniques make the general assumption that the ECG interference has the same shape and structure as a separately recorded raw ECG signal, which is inaccurate for many signals.

We propose a new approach for the removal of ECG interference spikes from an EMG signal by using a localized wavelet filtering technique. The approach uses a combination of simultaneously recorded raw EMG and raw ECG signals from a separate lead configuration to determine the exact locations of ECG contamination within an EMG signal. Then the contaminated EMG signal is filtered locally using a 128 component Morlet Wavelet decomposition to remove the wavelets that correlate with the ECG noise. This approach allows the filtering procedure to be localized around the ECG contamination and create an effective wavelet filter. Additionally, it avoids removal of important data and the introduction of additional corruption within an EMG signal due to the localization. The localized filter removes the ECG components of a signal at the location of the ECG contamination. Therefore, it is a smart filtering process that only targets specific areas of the signal.

II. METHODS

We propose a localized filtering approach for the removal of ECG noise that is present in recorded EMG signals. The proposed approach consists of four main steps: (i) Automated detection of the presence of ECG noise in EMG signals; (ii) Alignment and synchronization of raw ECG data with the ECG noise present in an EMG signal; (iii) Analysis and removal of the wavelets present in the EMG signal that make up the ECG noise; (iv) Localized filtering and removal of the ECG noise from the ECG signal. This framework is shown in Figure 1.

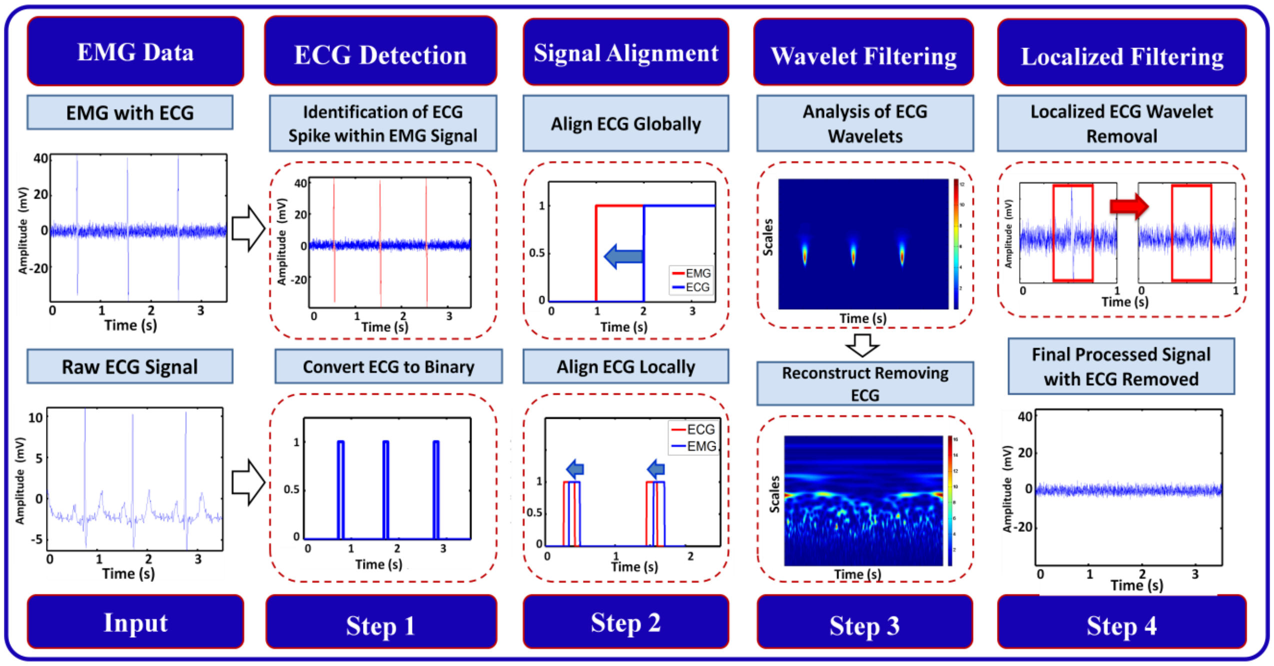


Figure 1. Block-diagram of the proposed localized ECG noise filtering framework.

2.1 ECG DETECTION

The first step of our approach is to determine if ECG is present in a signal. This is necessary to automatically determine if there is substantial ECG spikes present in the EMG signal that require removal. The ECG detection is done in the following 4 steps: (i) Smoothing and rectification of the initial signal; (ii) Analysis and calculation of the baseline threshold; (iii) Conversion of signal to binary form using the threshold determined in (ii); (iv) Determination of ECG presence by comparing rate of spikes to physiological viability. The individual steps of the ECG detection process are visually illustrated in Figure 2 (a-d) and are described below.

It is important to note that the best ECG representation occurs in an event free area of the signal, where there is relatively flat baseline. Therefore, the first step in ECG detection is to extract the event-free areas from the signal. To automate this extraction, event markers from the data collection process was provided. In the generic data set collected, there are three fairly evenly spaced event-based contractions. Hence, that would mean there are four possible event-free areas. The first event-free area is the initial signal prior to the first contraction. Second and third event free areas are the signals between the second and third contractions. The final (fourth) event-free area is located at the end of the signal, after the third spike. It was noted that most signals baseline had substantial electrical noise at the beginning and the end of the signal, probably caused by the data acquisition systems. Hence, there was too much baseline shifts to be able to create an algorithm for accurate ECG detection. Therefore, the second and third event-free areas of the EMG signal were targeted as the source of ECG spikes for the detection algorithm.

Initially, the signal is rectified in order to make it easier to analytically calculate the magnitude of the signal, instead of obtaining it through the relative difference within the signal.

The signal was rectified by obtaining an absolute value at every point. A moving window of 10,000 points was then extracted from the specified event-free area. This window represented five seconds of data. In, order to minimize the effects of low magnitude noise spikes that occur in the baselines of recordings, the signal in the window was smoothened. In this step, a moving-average filter, with a window width of 10, is used to smooth the data and eliminate the stray baseline noise spikes. The moving smoothing average filter works by averaging a point in the signal with its ten neighbors (five on each side). The center point of this averaging filter slides across the signal, effectively averaging all of the points based on its neighbor. The first and the last five points of the signal are not averaged, because they don't have ten neighboring points. It is also important to note that as the pivot point of the smoothing filter slides across the signal, it does not use the neighbors that have been smoothened, but the neighboring points from the initial signal in order to accurately and consistently smooth the signal. This is illustrated in Figure 2. Figure 2 (a) shows an initial signal in the window, and (b) shows the signal after the rectification and smoothing has been applied to the signal.

Following this initial signal transformation, the target window is split in half. The absolute maximums on both halves of the window are calculated. These absolute maximums serve as the anchor to generate the smart threshold, which determines the location of the ECG spikes. Therefore, validating that the anchors represent an ECG spike, instead of a huge erroneous noise spike, is important. An analysis of the baseline is also performed, in order to create a relationship of the height of the ECG based on the baseline. The EMG signal is divided into windows, each having a width of 100 samples. Therefore, there are 100 different baseline windows in the target ECG detection window. Within each window the absolute maximum in calculated. Then, from the set of all absolute maxima, the median value is determined. This value appropriately shows the best representation of the height of the baseband in the signal. The analysis of the baseband is important in order to validate the two anchored maximums within the window. One of the key factors in low-to-medium scaled ECG spikes is that the amplitude of the spikes relatively stays the same. There are no huge changes in neighboring amplitudes for ECG spikes. Therefore, in order to ensure that the anchors represent ECG spikes, the difference between the amplitudes of the two anchored spikes must be within 4 times the height of the baseband. This ensures that high amplitude baselines, with higher gain, have a higher tolerance to be considered an ECG spike compared to a low/flat amplitude signals. This key factor discussed above does not hold for signals with large scaled ECG spikes. Due to the abrupt corruptive nature of these huge ECG spikes, they don't fall within the standard that the ECG spikes have similar amplitudes. In order for the ECG spikes to be considered huge, and avoid validation, the anchors on both halves of the window must be greater than 25 times the baseband.

Once it is validated that the two anchoring spikes in the window have an ECG spike profile, a secondary validation is required. Signals with small noise spikes tend to have similar profile as an ECG, and end up being incorrectly validated. In order to ensure that the anchored ECG spikes don't fall into the small noise spike characteristics, this algorithm required the mean of the amplitude of the two anchored spikes to be at least greater than 1.75 times the baseband. If all of the validation checks passed, and the anchors in the window were legitimately based off ECG profile, a threshold of 60% was created. Specifically, the threshold was 60% of the mean of the anchors in the signal. Values within the window falling below this threshold are determined to belong to the baseline or to erroneous noise spikes in the data. Values above this threshold are determined to be ECG spikes. Figure 2 (c) shows the result of this process. In this figure, locations in the signal that fall below this threshold are classified as belonging to the baseline (shown in red), while locations in the signal that are above this threshold are classified as ECG noise (shown in blue). Because of the adaptive ability of the threshold to an individual signal, it shows good results in signals with both low and high magnitude ECG noise. Using this threshold, the EMG signal is then converted to binary signal. The data representing the ECG noise represented as 1 and the background noise represented as 0.

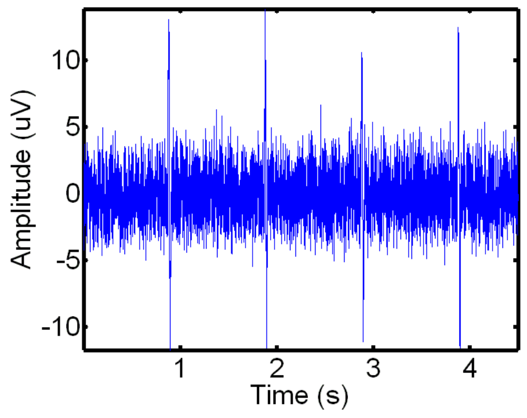
Due to the width, and the profile of an ECG spike, multiple data values in the same ECG spike are larger than the threshold. In order to ensure that the same spike was not being counted multiple times, a binary mask was created. The binary mask consisted of a 200 point neighbor. A standard width of an ECG spike is 80-160 data points. That meant if neighboring spike locations were closer than 200 points, they were considered to be part of the same spike. This was done by flood-filling in all the values between the two spike locations as 1. Hence, a rectangular 2-D representation of the ECG spike was created. The final binary signal is depicted in Figure 2 (d).

One of the main problems that occurred at this point is that certain EMG signals, which are far away from the heart, had spikes characteristics that are similar to ECG noise. However, they cannot be filtered, because they are small, and don’t share the same wavelets as an ECG profile. One of the main goals of this paper was to create an automated and robust system that could handle any type of signal, but only filters ECG spikes. Therefore, another characteristic of the ECG was used to further distinguish the noise spikes compared to true ECG spikes. ECG spikes have a rhythmic pattern, meaning that they don’t occur in clusters. Many times, noisy spikes occur in clusters or at certain area of the signal. Therefore, this next validation step checked for one of two things. The first check was to make sure that there are relative even amounts of spikes (+/- 1) on both halves of the target window. However, this check doesn’t always pass in signals with ECG, because some ECG spikes are so small that they don’t get recognized by the threshold algorithm. Therefore, the algorithm thinks that there are unbalanced spikes in the window, even if there are balanced ECG spikes in the window. In order to compensate, the algorithm has a lighter test to check for rhythm. It ensures that the there is at least 1 spike on both halves of the window to be considered an accurate representation of ECG.

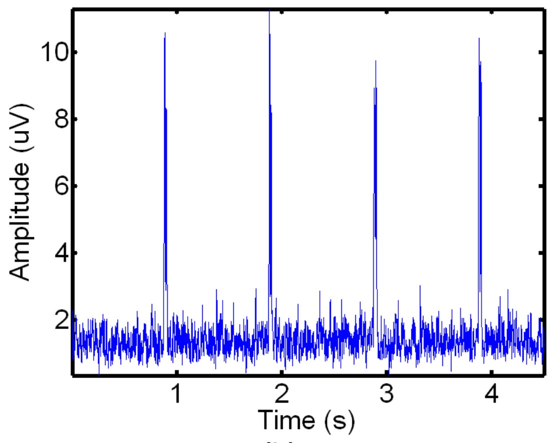
Once the ECG-profiled spikes have been validated to be genuine spikes (and not noisy spikes), the spikes are actually counted to make sure that they are ECG noise. These spikes have to lie in a physiologically relevant range to be considered part of ECG. Remember, that the target window consisted of 5 seconds worth of data. It is assumed that the spikes within the window should lie in a 2-10 spike range. Representing a range of 30 beats per minute to 120 beats per minute, which is typically present in most physiological based data sets. If the amount of spikes lies within this range, then the window is considered to have ECG. The front edges of the binary bins are then extracted as the location of the spikes, which is a crucial step for alignment process. In order to further ensure that these spike edges represented true ECG spikes, and would not mess up the entire alignment process, a last validation check was performed. This spike check had no influence in determining if ECG is present in the signal, as it is used to try to extract the best possible ECG spikes in the signal. This check made sure that there was at least a half second delay between neighboring front edges of ECG spikes. If the window containing ECG spikes did not have that spacing between the spikes, the spike location information for alignment was not collected.

It is important to note that all of these steps have been performed on a small window of 5 seconds in an event-free area of the signal. In order for the signal to be determined to have ECG, at least 40% of the event-free area of the signal must contain ECG spike characteristics. Hence, this whole process was repeated as the window slid through the event-free area of the signal.

40% requirement was essential, since the standards of ECG detection were strict. Hence, some windows with ECG spikes were being failed by the algorithm's validation checks. Probability wise, at least 60% of the signal contained algorithmically relevant ECG, which made the overall detection algorithm accurate and robust. The second output of this algorithm was the location of the ECG spikes, in order to help localize alignment (discussed in alignment section). The algorithm output spike locations at each window that contained ECG. A secondary algorithm sorted through these windows, and only saved the spike locations that had the highest probability to contain the cleanest ECG spike. This sort was done on a basis that the middle of an event-free area of the signal contained the most pure/flat EMG baseline; hence ECG representation in that area was the cleanest. These spikes were saved. The spikes in edges of the event-free areas were discarded, since the EMG baseline had bleed-offs from the contraction event, producing harder to differentiate ECG. The overall algorithm is summarized in Table 1.



(a)



(b)

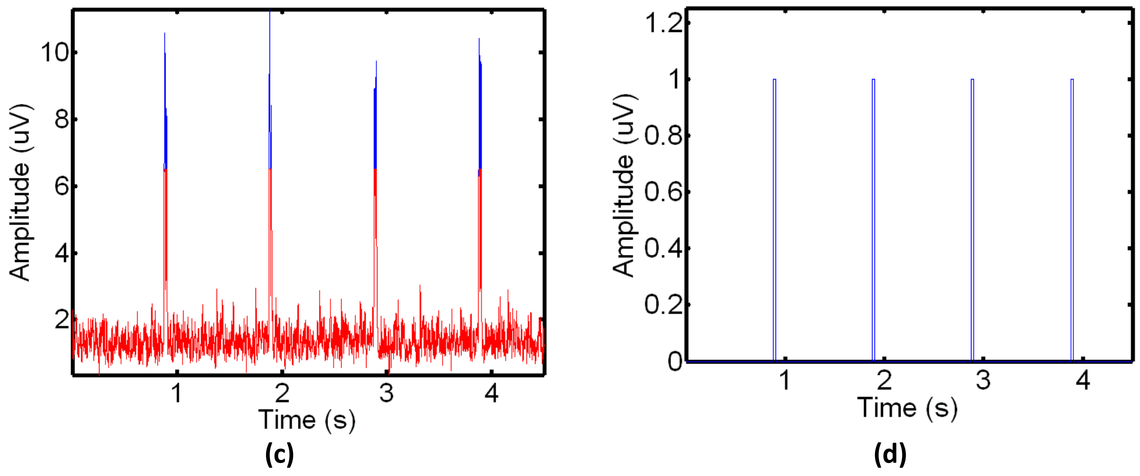
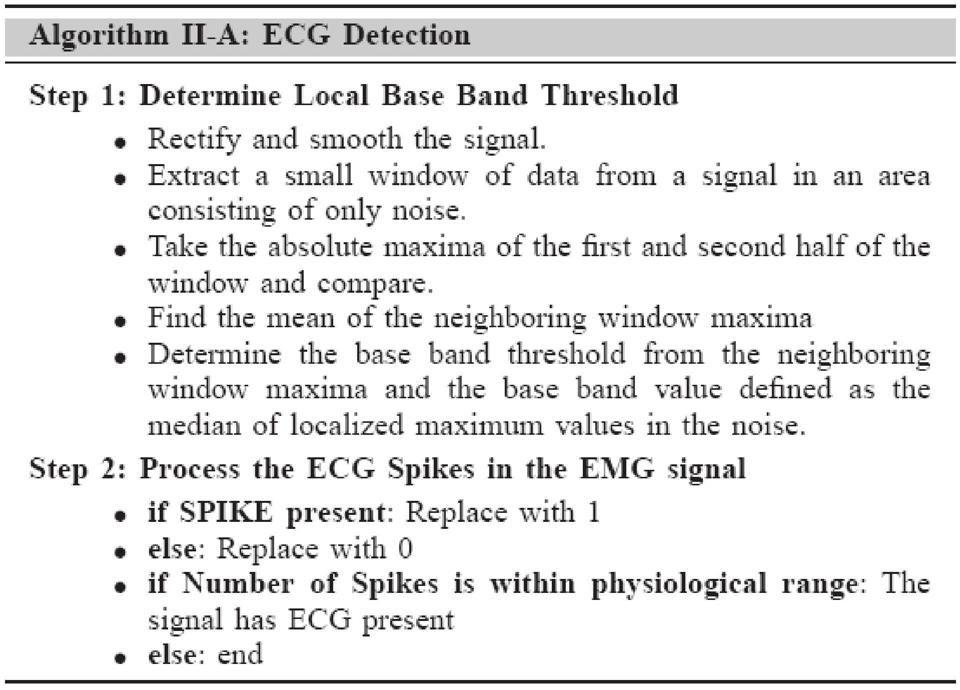


Figure 2. Illustration of the proposed steps in the proposed ECG detection. (a) The initial EMG signal, shown with ECG noise present. (b) The smoothed and rectified transformation of (a). (c) Showing the automatically detected baseline components in red and the remaining unfiltered spikes in blue for the given signal in (a). (d) The final binary threshold of the ECG noise spikes of (a). There are clearly four resulting spikes present in this noise interval.



Algorithm II-A: ECG Detection

Step 1: Determine Local Base Band Threshold

- Rectify and smooth the signal.
- Extract a small window of data from a signal in an area consisting of only noise.
- Take the absolute maxima of the first and second half of the window and compare.
- Find the mean of the neighboring window maxima
- Determine the base band threshold from the neighboring window maxima and the base band value defined as the median of localized maximum values in the noise.

Step 2: Process the ECG Spikes in the EMG signal

- **if SPIKE present**: Replace with 1
- **else**: Replace with 0
- **if Number of Spikes is within physiological range**: The signal has ECG present
- **else**: end

Table 1: Overview Algorithm for Automated ECG Spike Detection

2.2 SIGNAL ALIGNMENT

To accurately identify the ECG noise interference locations for proper removal, we propose a technique to synchronize the raw ECG recordings to the original signal. This allows for the ECG noise spike locations to be accurately determined within the EMG signal that contains the ECG noise. This step is critically important in that it allows for a localized filtering of the ECG contamination in the signal. The synchronization is carried out in a three step process: (i) Alignment of EMG and raw ECG from two separate systems using a synchronization pulse; (ii) Conversion of the raw ECG signal to binary; (iii) Refinement of the alignment to an individually recorded signal.

Before any data sets could be aligned, both ECG and the EMG data sets needed to be oriented in a manner where they can be directly compared. This step is simple, but essential for any kind of data manipulation in multiple data sets. It is assumed that the automation of the program can handle the data in any orientation, therefore in order to convert them into vertical top-down data set; the horizontal data sets were transposed. The inverse of an array (transpose) flips it into the correct orientation. It will be assumed in this paper that any step involving multiple data sets has this orientation fix/check added to it during initialization. It is also important to note that different signals with different acquisition systems don’t have the same sampling frequency. In this case, the EMG signals had a sampling frequency of 2k Hz, while the ECG signals had a sampling frequency of 1k Hz. In order to compare the two signals, the ECG signal had to be digitally processed to have the same sampling frequency. Hence, the ECG signal was interpolated by a factor of the ratio of the sampling frequencies between the two signals (2). Linear 1-D interpolation was performed to increase the sampling rate of the ECG signal.

Due to the fact that the EMG data and the ECG are recorded using two separate systems it is necessary to align the data sets globally. At distinct time intervals electrical pulses are recorded in each system. These electrical pulses will be referred to as manual pulse signal for clarity in the paper. Using these manual pulses, the two systems can be synchronized with one another. It is assumed that the last signal in both data sets represent manual pulses of the two signals. One of the problems that occur in the manual pulses is that it's not a single value indicator. Manual pulse is actually a sharp pulse peak, which indicates start of a recording activity. This means that the front edges of these manual pulse peaks needed to be determined on both of the pulse signals. In order to determine the appropriate front edges of these pulses, the pulses were binarized. This was a soft binarization, as the manual pulses have a clear, sharp profile. The binarization was based off a threshold that was calculated by the maximum of the pulse divided by two. For the specified alignment approach, only the front edge of the pulse is important, therefore no flood-filling or binary refinement was performed. The difference between the two edges of pulses, calculated as $L_{DIFF} = L_{EMG} - L_{ECG,}$ equate to the shifting value required for aligning the pulses.

It is important to note that only the ECG signal is shifted in order to properly align the signal. The EMG signal is untouched, in order to preserve the entire data signal. If L_{DIFF} is positive, then the ECG signal is shifted left. If it is negative, then the ECG signal is shifted right. In this Figure 3 (a) the edges of the synchronization pulses can be seen to be offset from one another. The alignment is performed by shifting the ECG signal to match with the EMG signal. The result of this operation is shown in Figure 3 (b) where the two signal synchronization pulses are correctly aligned.

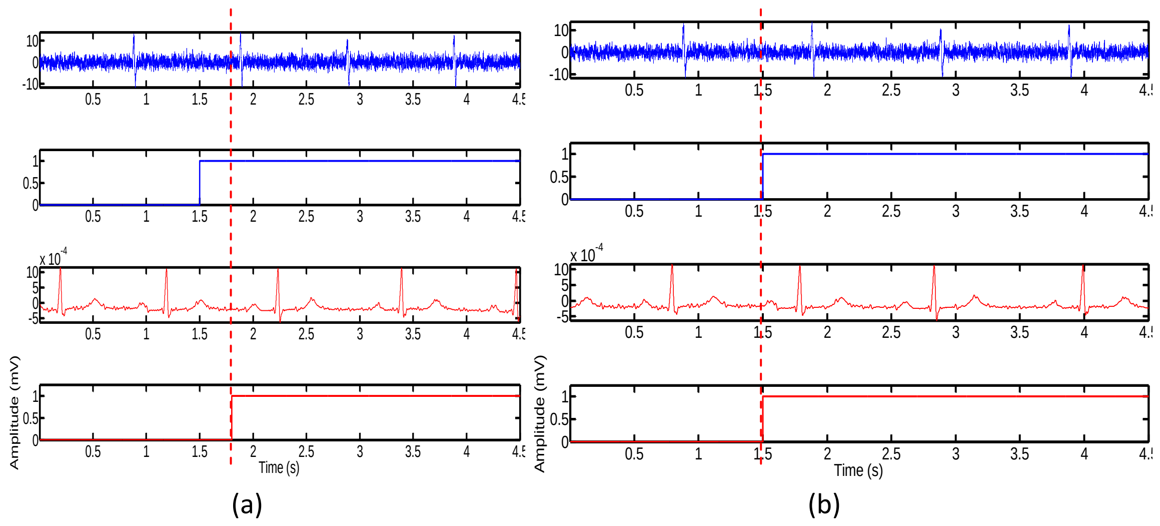


Figure 3. Alignment of the synchronization signal. The EMG signal, containing ECG noise, is shown in blue as the first signal, with its synchronization pulse shown below it. The third signal is the raw ECG signal, with its synchronization pulse shown below it. The dashed line in each figure indicates the extended edge of the location of the synchronization pulse for the ECG data. (a) Shows the two separately recorded signals before alignment. (b) Shows the two signals following alignment. It can be clearly seen that the signals are aligned after the ECG has been shifted to match the EMG signal.

After pulse synchronization the raw ECG signal is converted to a binary representation. The signal is converted to binary such that during the QRS complex of the ECG signal, the binary representation is 1. In the remaining recording information the binary representation is a 0. This process was done through several steps. The first step was to smooth the ECG signal using a standard smoothing average filter with a window size of 10. This was necessary, since it was found that the raw ECG by itself posed a lot of problems in trying to create a threshold model to binarize it. Smoothing the ECG, and removing the rough spikes within it, helped create a smooth spike, which was easier to threshold. After smoothing the ECG, a threshold was determined as ((max of ECG)+(mean of ECG))/2. This threshold equation was developed after quite a bit of trial/error. It is important to note that there is a common theme in developing threshold or models for ECG analysis. The common theme is that due to the high amplitude of spikes, in respect to the baseline of the signal, both the amplitude and the mean of a spike are used to determine a threshold. The one big change in the different threshold processes is the weight assigned to each of the two parameters. Due to the simplicity of the location of ECG spikes within the raw ECG, a 50/50 weight (representative of a pure spike) was the ideal approach in developing the threshold. After determining the threshold, the values that lay above it were considered to be part of active ECG spike (1), and the values below it were considered to be part of the baseline (0). A flood-fill mechanism with 1000 point spacing was used in order to ensure that the same spike wasn't split into two binary forms. As mentioned in the ECG detection algorithm, the flood-fill works by filling in active binary values between neighboring spikes that occur within 1000 points (0.5 seconds). This creates a rectangular binary representation of an ECG spike, instead of multiple dispersed individual spikes. Figure 4 illustrates the conversion of the raw ECG to a binary form.

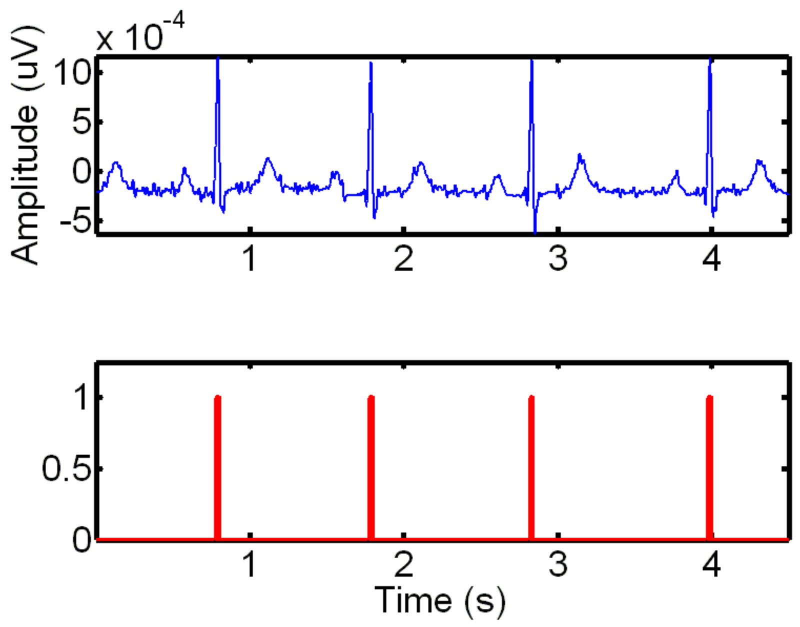


Figure 4. Conversion of a raw ECG signal to binary representation. The original raw ECG signal is shown in the top and the binary representation of this signal is shown in the bottom.

Localized Alignment

In EMG recordings typically multiple recordings are simultaneously acquired. The location of each of these recording sensors is at a different distance from the heart. Therefore, while the pulse synchronization provides a globally acceptable synchronization, it is necessary to refine the ECG alignment for individually recorded signals. This localized alignment of the binary ECG signal is applied uniquely to each recorded signal. To perform this alignment, the ECG spikes in the EMG signal located from the ECG detection algorithm (Section 2.1) were extracted. These spikes were located in an event-free area and served as a landmark for localized alignment. The front edges of the binary representation of the ECG spikes in the raw ECG (Figure 4) were also extracted. In order to calculate the local offset, the difference between the location of one of the ECG spikes found from the ECG detection algorithm, and the front edges of the binary representation of ECG spikes in the raw ECG signal were found. The lowest offset represented the pair that was closest to each other, which meant that ideally this pair should have 0 offset, as they represent the same ECG spike. In order to account for this offset, the raw ECG signal was shifted by the amount of local offset. If the offset was negative, then the signal was shifted right. If it is positive, then the signal was shifted left. It is important to note that in order to improve accuracy, the shift was actually over-shifted in either direction by 35%. This was done in order to account for the discrepancy in the ECG location found by the ECG detection algorithm, which generally located the ECG at its maximum, compared to the ECG found in the binary representation, which located the ECG at its front-edge. The process is demonstrated in Figure 5. In the top of Figure 5, the smoothed EMG signal containing ECG is shown in blue and the binary ECG representation is shown overlaid in red. Note that while the alignment is close to the ECG spike it is slightly offset. The bottom of Figure 5 shows the results following the localized alignment. In this figure the binary ECG representation has been shifted such that it encloses the ECG noise spike in the data. The overall algorithm is summarized in Table 2.

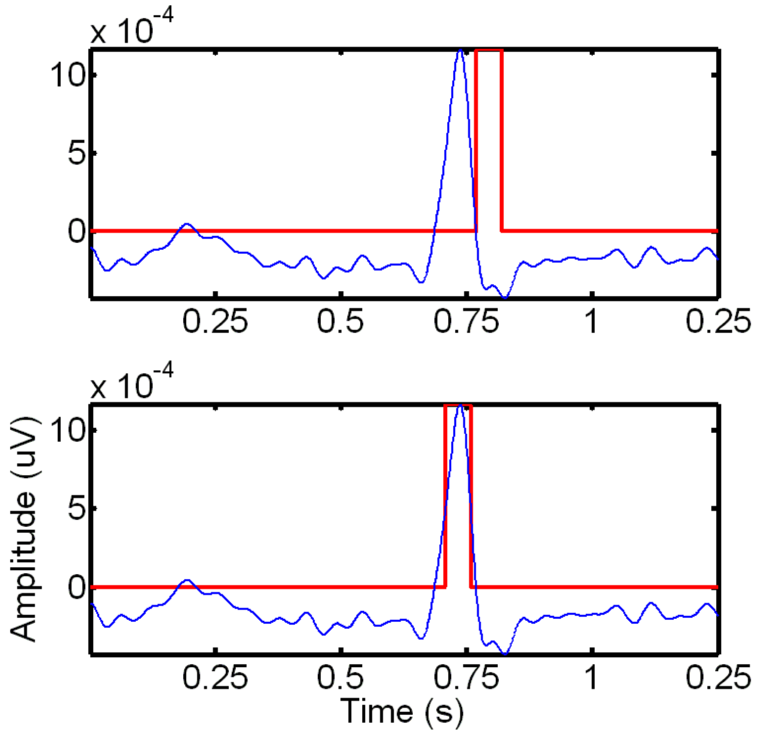
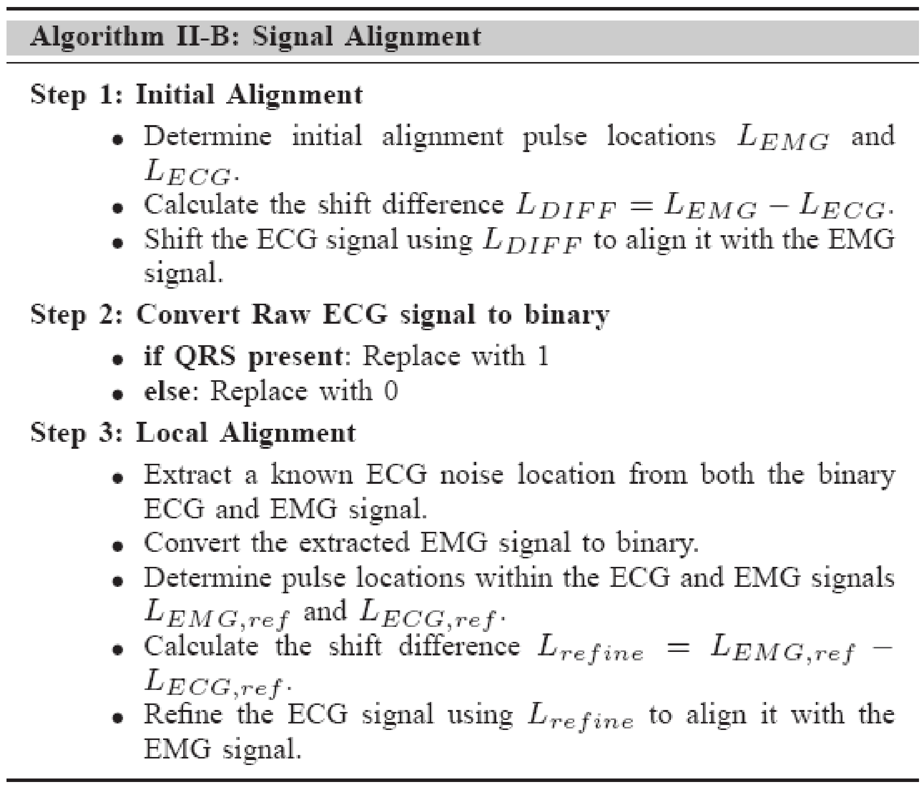


Figure 5. Localized alignment of binary ECG data to an EMG signal. In this figure a window around a single spike is shown before and after the local alignment. The smoothed ECG signal is shown in blue while the binary ECG representation is shown in red. In the top figure the offset between the initial alignment and the individual signal is small, but noticeable. The bottom figure shows the result of the local alignment of the ECG data. The resulting binary ECG representation is aligned to the locations of ECG spikes in the EMG data.



Algorithm II-B: Signal Alignment

```
Step 1: Initial Alignment
    • Determine initial alignment pulse locations $L_{EMG}$ and $L_{ECG}$.
    • Calculate the shift difference $L_{DIFF} = L_{EMG} - L_{ECG}$.
    • Shift the ECG signal using $L_{DIFF}$ to align it with the EMG signal.
Step 2: Convert Raw ECG signal to binary
    • if QRS present: Replace with 1
    • else: Replace with 0
Step 3: Local Alignment
    • Extract a known ECG noise location from both the binary ECG and EMG signal.
    • Convert the extracted EMG signal to binary.
    • Determine pulse locations within the ECG and EMG signals $L_{EMG,ref}$ and $L_{ECG,ref}$.
    • Calculate the shift difference $L_{refine} = L_{EMG,ref} - L_{ECG,ref}$.
    • Refine the ECG signal using $L_{refine}$ to align it with the EMG signal.
```

Table 2: Overview Algorithm for Automated Signal Alignment

2.3 Wavelet Filtering

Morlet Wavelet

To filter the ECG noise data from the EMG signal a filter was designed based on the Morlet Wavelet Decomposition. A Morlet wavelet decomposition is an example of a continuous wavelet transform, which is shown in Equation 1. A continuous wavelet transform allows us to divide a continuous-time function (waveform signal) into wavelets. It offers a very good time and frequency localization, which enables us to separate a signal into different frequency bands represented by sub-wavelets over time. The Ψ represents mother wavelet, which in this case is the Morlet window. This window is used by subtracting a user-defined constant from the wave followed by localization using a Gaussian window. Equation 2 shows the analytical equation of the mother wavelet, Morlet Window. In this equation, v represents the specific wavelet number. The parameter t is the specific time point on the signal. Equation 3 and 4 are used to generally construct the Morlet window from a base wavelet. Specifically, Equation 3 represents the constant subtracted from the base wave, which is used to calculate the number of wavelets required for the ideal deconstruction of the signal. This value, specifically relies on the length of the signal. While Equation 4 represents the normalization factor to create the Morlet window.

$$CWT\frac{\psi}{x}(\tau, s) = \frac{1}{\sqrt{[s]}}\int [x(t)\psi * \left(\frac{t-\tau}{s}]\right) dt$$

EQ 1

$$\textbf{Morlet Window} = \Psi v(t) = Cv\Pi^{\frac{-1}{4}} e^{\frac{-1}{2}t^2} (e^{ivt} - Kv)$$

EQ 2

$$Kv = e^{\frac{-1}{2}v^2}$$

EQ 3

$$Cv = \left(1 + e^{-v^2} - 2e^{\frac{-3}{4}v^2}\right)^{\frac{-1}{2}}$$

EQ 4

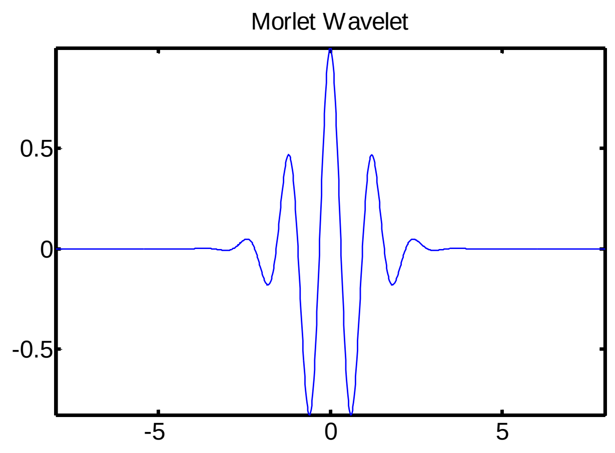
Comparing Morlet with other Wavelet Transforms

As explained previously, the most common way to remove ECG corruption using wavelets is Multi Resolution Analysis (MRA). Multi Resolution Analysis means that instead of filtering the raw waveform, the waveform's wavelets are filtered instead. The reasoning behind using an MRA is to target high frequency based wavelets, which are most susceptible to noise. The low frequency wavelets are deemed as storing vital information for the waveform, hence are not filtered. After the targeted wavelets are filtered, the signal is reconstructed. In order to accomplish this, the waveform is generally separated into wavelets using discrete wavelet transforms such as debauchies, symlet, coiflet, or biorthogonal separation. Each of these wavelet transforms can be modified with a specific amount of vanishing moments (how the function decays towards infinity). Larger amounts of vanishing moments correspond to higher complexity [higher degree polynomial] low/high pass filter used in wavelet deconstruction/reconstruction (Chun-Lin 2010). These higher complex filters also result in higher length filter, which hence reduces resolution of multiple filters used in wavelet analysis.

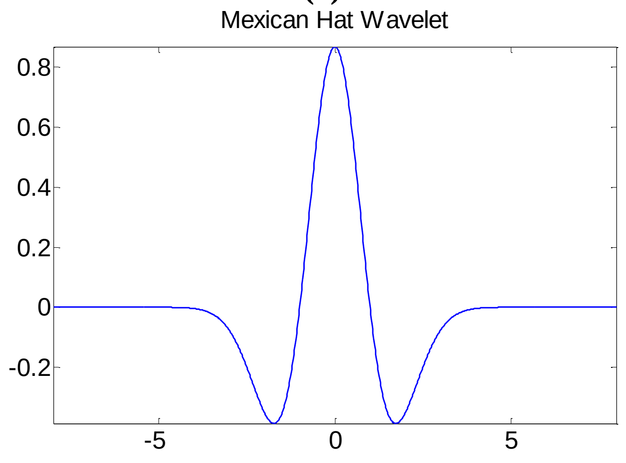
Deconstructing these waveforms using discrete wavelets transform results in two set of wavelets. The first set consists of only one wavelet, which is created through low pass filters that is referred to as approximation. The second set consists of multiple break-down wavelets, which are broken down using high-pass filters. This set of wavelets is referred to as detail wavelets. The amount of decomposed detail wavelets depends on the user's requirements for the level of high frequency deconstruction required for the application. It is assumed that the approximation wavelet is untouched during filtering process, as it is the most sensitive wavelet required for accurate reconstruction. All of the filtering occurs in the detail wavelets, and removing wavelets is also possible with proper scaling procedures. The biggest problem in using this discrete wavelet transform is the fact that ECG spikes generally occur in low frequency cluster, hence have a strong representation in the approximation wavelet. The detail wavelets only contain the bleed-off noise within these ECG cluster. Hence, filtering them only results in smoother ECG spikes, which is pointless filtering process. Discrete wavelet transform is more beneficial for filtering high-frequency electrical noise.

Due to the recent advances in processing speeds and computational power, continuous wavelet transforms (CWT) can be used for deconstructing wavelets as well. It is important to note that performing true continuous wavelet transform is impossible, as there would be infinite wavelets (scalar coefficient) in a deconstruction. Therefore, referring to continuous wavelet transform actually means using CWT with discrete scales. Each continuous wavelet has its own formulas for deriving the discrete scales for deconstruction of a waveform. The advantage of continuous wavelet transform compared to discrete wavelet transform is that low frequency signal can also be differentiated into many different wavelets for precise separation. Meaning, in the deconstruction process there is no distinct different in low frequency separation compared to high frequency separation. This process is ideal for ECG removal application, as the ECG spikes occur in low-frequency clusters, but also contain high frequency noise within the clusters.

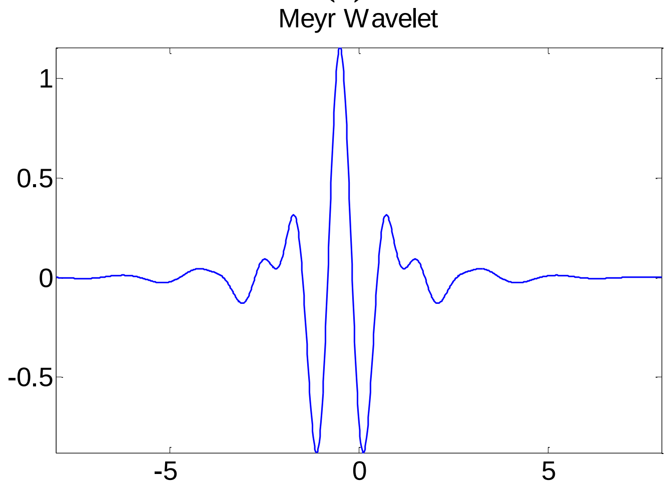
There are three main continuous wavelet transforms used in various applications. These transforms are Mexican Hat, Morlet, and Meyr. Graphical illustration of all three of these wavelets is shown in Figure 6. The Mexican Hat window is computationally fastest for wavelet decomposition, but it is ideal for signals that follow a certain frequency pattern. It is ideal for waveforms such as EEG, which has a repetitive rhythm. Unfortunately, it’s not suitable for EMG signals that have high frequency disturbances from the ECG spikes, and the activity bursts. The frequency disturbances from ECG spikes in the EMG signal has already been discussed. Activity bursts in an EMG signal occur when the motor neurons are active, and are firing at a rapid pace. This results in a higher amplitude EMG signal at a much more rapid frequency. Hence, there is generally a change in frequency characterization in EMG signal, which makes Mexican Hat window not acceptable. Both Morlet and Meyr are designed to be able to effectively decompose waveforms with many frequency disturbances, and no rhythmic pattern. In fact, Meyr window does the better job, because it creates a more accurate reconstruction of the waveform. However, Meyr window is highly computationally intensive, and also requires a scaling function, which is not ideal for this application. Therefore, Morlet window is the best option for efficiently deconstructing and reconstructing EMG signals with ECG corruption.



(a)



(b)



(c)

Figure 6 (a) Graphical representation of the Morlet Wavelet Window. **(b)** Graphical representation of the Mexican Hat Wavelet Window. **(c)** Graphical representation of Meyr Wavelet Window.

Band Stop Filtering

Before doing a wavelet analysis, the signal was prepped with a basic filtering process. In this case, a Band stop filtering was used to process the signal. There are two main steps in performing a band stop filtering. The first step is to obtain the frequency bands, where the spikes are located at. These frequency bands are referred to as start band and stop band. The second step is to effectively use a filter, which removes data at those frequencies, hence effectively removing part of the spike. This is especially useful in large spikes, which have discernible frequency compared to the EMG baseline.

In order to find the start and stop bands, the event free area's frequency composition was analyzed. Specifically, the fast Fourier transform of the signal (FFT) was taken at 2 times the sampling rate (4 kHz).The FFT of the signal was rectified for analysis. This allowed us to find the frequency composition of the signal. Since, the event-free area of the signal is analyzed; ideally there is a flat EMG baseline with huge ECG spikes. Therefore, the FFT should provide a high amplitude clustered spike at the frequency range of 1-2 Hz (60-120 beats per minute). It is not ethical to assume that the ECG spike always occur at 1 Hz, since the heart rate is different for each patient depending on their physical activity level. Therefore, it is crucial to identify an approximate frequency range of the ECG spikes. Since, the default range of fast Fourier transform is from ($-\pi$ to π), the ECG frequency would be read twice (one in positive spectrum, and one in negative spectrum with the same amplitudes). The duplicate values in the negative spectrum were removed from the data set. Figure 7 shows a sample positive spectrum frequency analysis from a fast Fourier transform of the noisy signal.

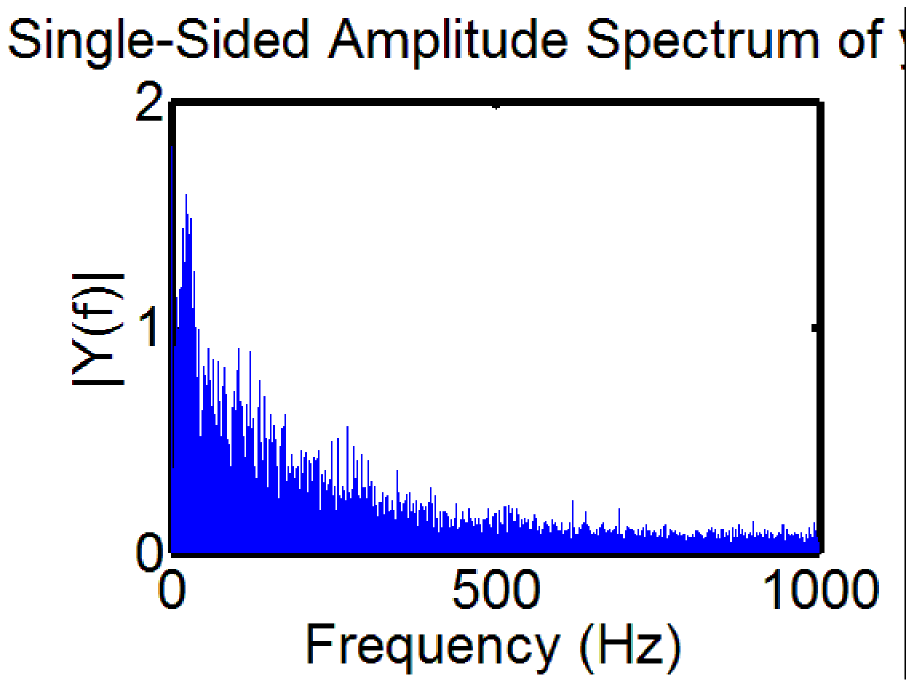


Figure 7 Sample One-Sided Frequency Spectrum Analysis based off Fast-Fourier Transform of an ECG contaminated signal.

Luckily, the process behind finding the cluster of frequency that represented the ECG spikes was similar to finding the ECG spike in the signal. A threshold within the Fourier representation of the signal was determined as one-thirds of the maximum amplitude in the frequency domain. Any values below this level were changed to 0, while values above it were unchanged. The remaining peaks represented a cluster of the target frequency range. Sorting the remaining signal showed the start of the cluster, which represent the start band. The end of the cluster represented the stop band for the frequency of the ECG spike.

Once the frequency bands were calculated, a blackman-harris window filter was used to eliminate the signal corresponding to the frequencies within the two frequency bands. Figure 8 shows the sample window overlay of the notch filter, where the central target frequency range is accepted, and the outer frequency ranges are rejected. A blackman-harris window filter is a discrete digital filter. Digital filters work by placing a weight on the signal based on the frequency of the signal. Since, the blackman-harris was a band-stop (notch) filter, the signal within the specified frequencies were rejected. A simple filter would just decimate the signal at the target frequency. This method is not efficient, since this would create a choppy effect within the signal. Many signals also have overlapping frequencies, where the target noise has a broad range of frequency that cannot be accurately detected by the Fourier transform. Therefore, more advanced filters have been developed that have transition bands (bands around the start and stop frequencies of the signal). Instead of just decimating a signal, the signal is slowly destroyed at the transition bands based on a rate determined by the filter. The signal is also not completely destroyed within the frequency range; instead it is attenuated at small amplitude, hence avoiding the choppy effect of a standard filter. The effect and width of the transition bands, attenuation, and other features are determined by the window of the filter. In this case, the blackman-harris is known to have a dull transition period, with a small chance of losing signal due to sharp transition peaks. A signal is generally filtered by convolving it with the signal. Two signals are convolved together by being multiplied in the frequency range. Black-man Harris window equation is shown in Equation 5. N represents the number of samples within the window. The number of samples is calculated from the start and stop frequency bands.

$$w(n) = a_0 - a_1 \cos\left(\frac{2\pi n}{N-1}\right) + a_2 \cos\left(\frac{4\pi n}{N-1}\right)$$

$$a_0 = \frac{1-\alpha}{2}; \quad a_1 = \frac{1}{2}; \quad a_2 = \frac{\alpha}{2} \qquad \textbf{EQ 5}$$

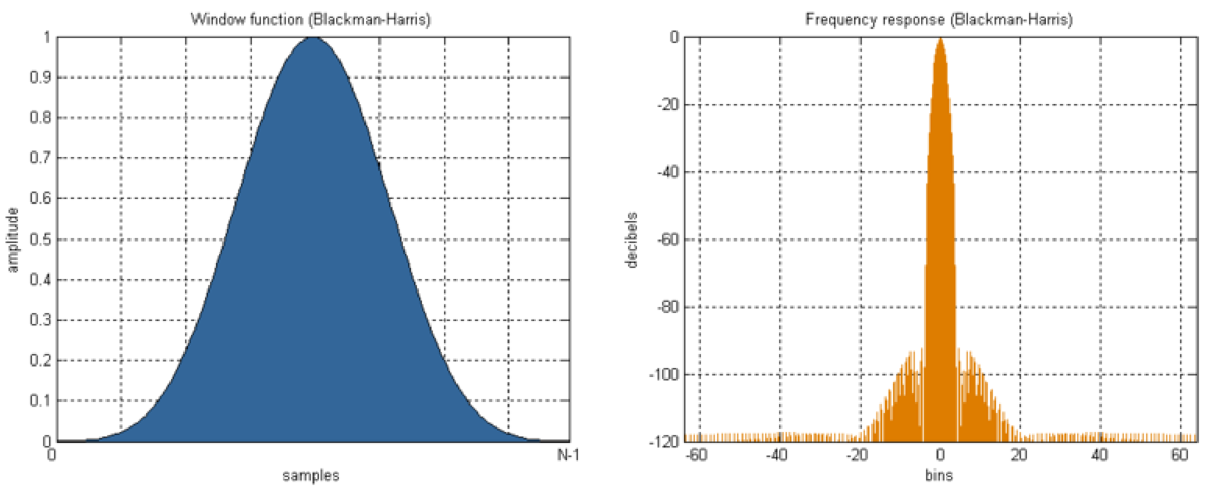


Figure 8 Sample window layout for Black-man Harris Notch Filter.

Wavelet Analysis

One of the primary disadvantages of using a frequency filter is that generally ECG spikes do not follow the same frequency throughout the signal. This happens due to the fact that in many different data recordings, the patient's heart undergoes stressful or relaxing situations, which causes the heart rate of the patient to change during the recording. Almost all of digital filters assume that the frequency of noise (although in a range) typically does not drastically change within the signal. Another disadvantage is the fact that is hard to discern certain frequency ranges as ECG compared to EMG using FFT. These frequencies generally share both EMG and ECG components, hence it is difficult to filter without either over filtering (removing EMG), or under filtering (leaving ECG clusters in the signal). Therefore, a wavelet analysis is more advantageous for the application of removing ECG spikes from the EMG signals.

The primary problem that occurred in using Morlet Wavelet as a transform is that its scales are generally fixed dependent on the length of the signal. This was default set by the transform, in order to obtain the best wavelet decomposition for a sample signal. Any less wavelet representation is under sampling of wavelets, and valuable information is not efficiently split into the different wavelets. Any higher wavelet representation means that the decomposition is over sampled, which means that the information may be duplicated in multiple wavelets. Although, this is an efficient way of representing the decomposition of waveform, it is not ideal for wavelet filtering. It is not ethical to remove specific wavelets within a waveform, if each waveform has a new subset of wavelets scales. Therefore, a standard 128 wavelet scales were chosen for each waveform. In order to customize the Morlet wavelet filter, custom scales of 128 were calculated through Equation 6 (Torrence and Compo 1998). It is important to note that k_σ is the re-defined function of Kv shown in Equation 3 for the custom scales. $\Delta(T)$ is the new time-step for each scale, based on N (the length of the signal), and I (total number of wavelets, 128 in this case). J_0 is used to back-calculate the number of wavelets, to ensure that there is sufficient data points available for 128 wavelet decomposition. Hence, k_σ represents the set of custom scales from the range of $(0toJ_0)$.

$$\begin{aligned}
\Delta(T) &= \frac{\log_2(N)}{I} \\
J_0 &= \frac{\log_2(N)}{\Delta(T)} - 1 \\
k_\sigma &= 2\Delta(t)(2^{\mathbf{0:J_0}\Delta(T)})
\end{aligned}$$

EQ 6

The result of the decomposition of signals is clearly visualized using a scalogram representation. In the scalogram the X-axis represents the samples in the signal, while the Y-axis represents the wavelet number. The energy intensity within a scalogram is calculated at each wavelet scale over time. This energy calculation is shown in Equation 7. Two examples of decomposed signals are shown in Figure 9. Figure 9 (a) shows a signal with ECG noise corruption present and (b) is the resulting scalogram. In this scalogram, it is evident that the two

ECG spikes dominate the energy within the scalogram. These high energy spikes have an energy bleed-off over multiple wavelet scales, which construct the ECG noise in the signal. Similarly, Figure 9 (c) shows a signal without ECG noise corruption present and (d) its resulting scalogram. It can be clearly seen that in a standard EMG signal, all of the wavelets should contain some energy as the EMG signal itself is very random. When the ECG noise corruption is present, the ECG tends to dominate certain bands within the signal.

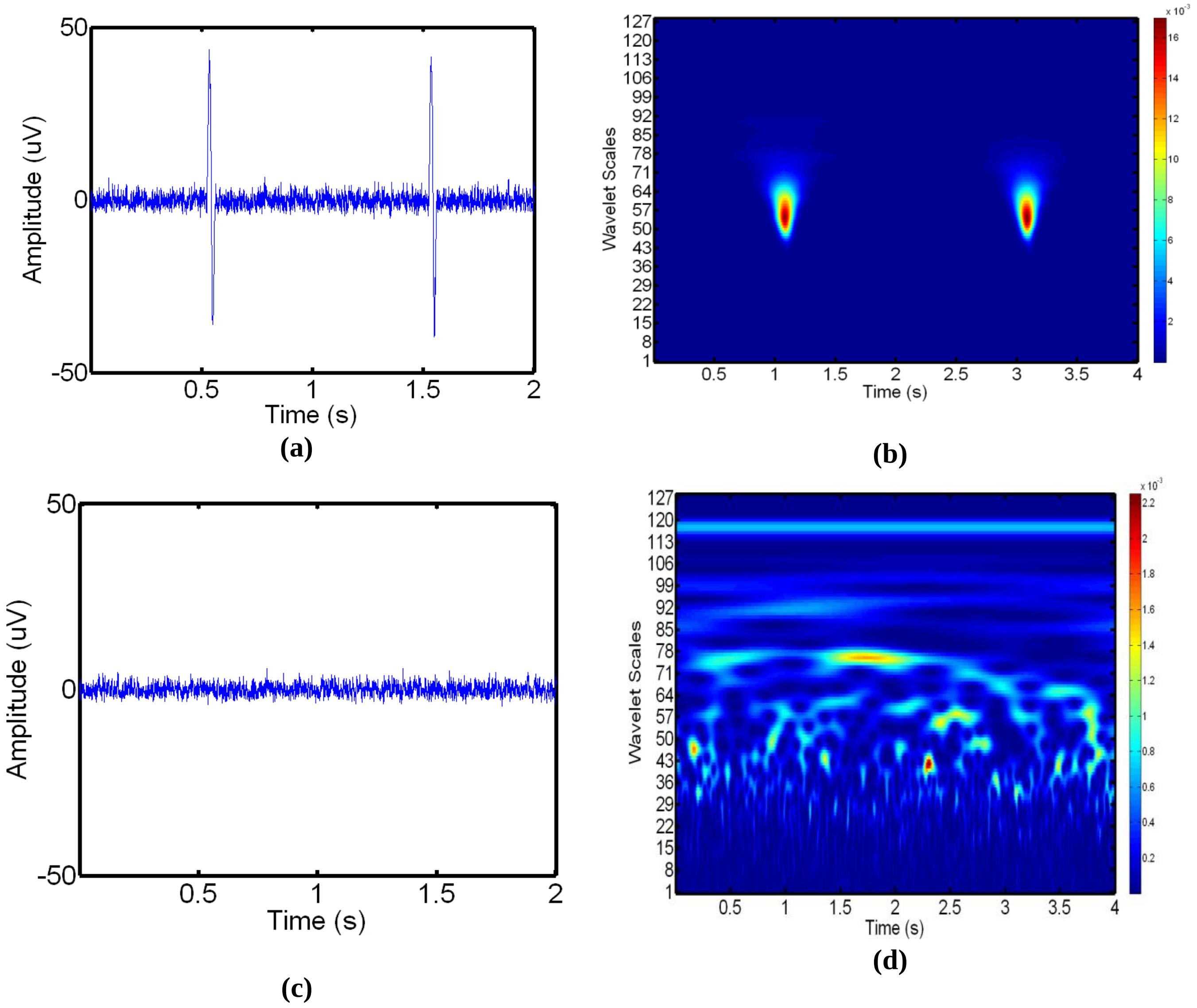


Figure 9. Signal and corresponding scalogram representations. (a) A signal containing ECG corruption and (b) its corresponding scalogram, and (c) a clean signal without ECG corruption

and (d) its corresponding scalogram. Note the distinguishable peaks in the ECG signal and how they translate to the resulting scalogram.

$$S = |coefficient^2|$$

$$SC = \frac{100 * S}{\sum S_i}$$

EQ 7

To remove the ECG noise from the signal a comparison is made between the wavelet decompositions of ECG noise in the EMG signal. From the EMG signal two windows are extracted. The first window is extracted from a known window containing only ECG noise by using the binary ECG representation shown in Figure 4. The second window is extracted from a larger portion of the signal containing both noise and ECG noise spikes. The two windows are then each decomposed into 128 wavelets using the aforementioned Morlet decomposition. A one-to-one cross correlation is then performed on the two sets of 128 wavelets. The 1-D signal cross correlation is defined in Equation 8. A gradient descent approach is used to sequentially remove the wavelets within the EMG signal that has the highest correlation with the wavelets corresponding to the ECG. This gradient descent process is done through a fminbnd function, which returns the range of wavelet scales that when removed, minimizes the energy of the signal. The signal energy is a custom equation that tries to analytically calculate the presence of spikes in a signal. The signal energy is defined in Equation 9. As it is evident, the signal energy has two components. The first component is the basic RMS energy of an equation. Ideally, when the high peak of a signal is removed, the signal RMS will also decrease. Unfortunately, EMG base band removal also decreases the RMS, hence the energy with RMS alone never minimizes during the removal process. Therefore, the second component of the signal energy is the maximum amplitude, which is typically represented by the peak of the QRS complex in an ECG spike. As the ECG spike is removed, the maximum of the short analyzed signal also decreases with the spike. When the process starts removing EMG based wavelets, the maximum does not necessarily decrease (baseband sometimes slightly increases in order to compensate removal). Hence, the change in energy is not dependent on EMG wavelet removal at this point. At this point, the signal energy has been minimized, and the ECG removal process is completed. It is important to note that both components have a 50% weight in the energy analysis, which was found to be ideal after some experimentation.

$$(f \star g)[n] = \sum_{m=-\infty}^{\infty} f^*[m]g[n+m]$$

EQ 8

$$E = \frac{1}{2}\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} + \max_{i=1,\ldots,n} x_i\right)$$

EQ 9

Once the energy of the remaining signal has reached a minimum the removed wavelets are known to contain information relating to the ECG noise. These will be removed from areas containing ECG noise during the local filtering. Figure 10 (a-f) shows the removal process. Figure 10 (a,d) shows the scalogram and contour representation, respectively, of the original signal containing ECG noise. As it is evident, there is highly concentrated energy within 30-40 wavelets that represent the ECG spike in the signal. The energy from the spikes is so dominant that the EMG signal is completely subdued with its relative low energy component. Figure 10 (b,e) shows the wavelets that contain the highest degree of correlation with the ECG noise. Specifically, these figures represent the wavelets that have the highest energy with the highest correlation to ECG noise. These wavelets are removed in the filtering process. Figure 10 (c,f) shows the reconstruction of the signal in (a) with the wavelets identified in (b) removed. Note that the areas which previously represented the ECG noise are no longer present in the filtered signal. The energy of the wavelets are now dominated by the background EMG signal, which has a random distribution of energy as seen on the two graphs, and previously shown. An important aspect to note is that the gap present in the wavelet decomposition shown in (c,f) comes from the removal of the wavelets defined in (b,e). Table 3 shows an algorithmic summary of the wavelet filtering process.

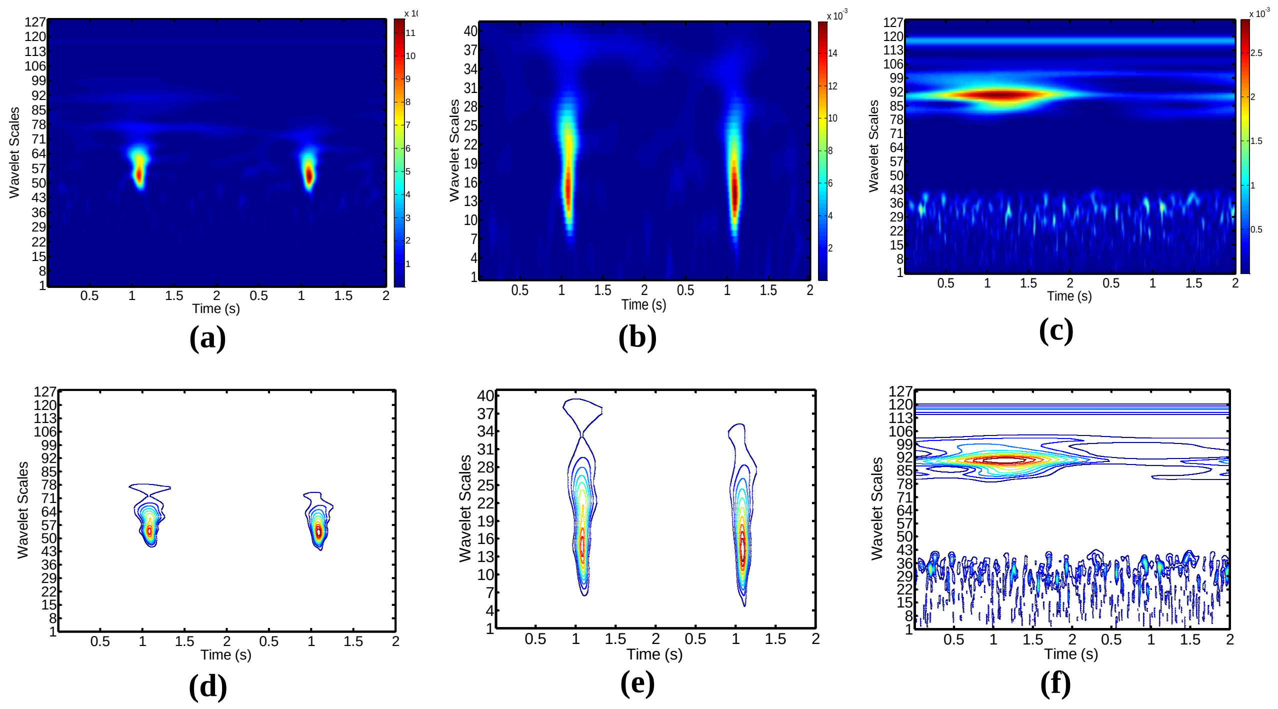
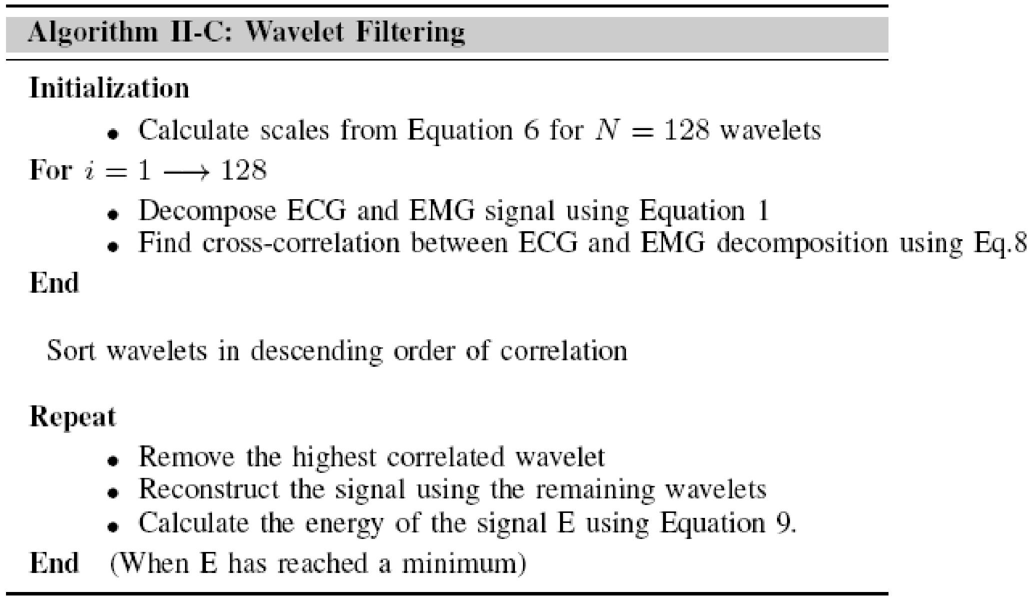


Figure 10. Scalogram and contour representations of the Morlet decomposition of an EMG signal containing ECG noise. (a,d) The original signal containing EMG noise before filtering. (b,e) The wavelets that contain the highest correlation with the ECG noise information. (c,f) The final reconstructed signal with the wavelets in (b,e) removed.



```
Algorithm II-C: Wavelet Filtering
Initialization
    • Calculate scales from Equation 6 for $N = 128$ wavelets
For $i = 1 \longrightarrow 128$
    • Decompose ECG and EMG signal using Equation 1
    • Find cross-correlation between ECG and EMG decomposition using Eq.8
End

Sort wavelets in descending order of correlation

Repeat
    • Remove the highest correlated wavelet
    • Reconstruct the signal using the remaining wavelets
    • Calculate the energy of the signal E using Equation 9.
End  (When E has reached a minimum)
```

Table 3: Overview Algorithm for Wavelet Filtering

2.4 Localized Signal Filtering

The final step is the localized filtering and removal of the ECG noise from the ECG signal. The binary representation of the ECG signal from 2.2 is combined with the selective wavelet filtering from 2.3 to optimally remove the ECG noise from the EMG signal. The localized filtering procedure is completed in four steps: (i) Extract the local window containing the ECG noise from the EMG signal; (ii) Identify the center of the ECG noise spike and the width of the noise spike within the local window, and use this information to refine the window around the ECG noise spike; (iii) Remove the wavelets from this refined window using wavelet correlation; (iv) Reconstruct the window using the filtered wavelets.

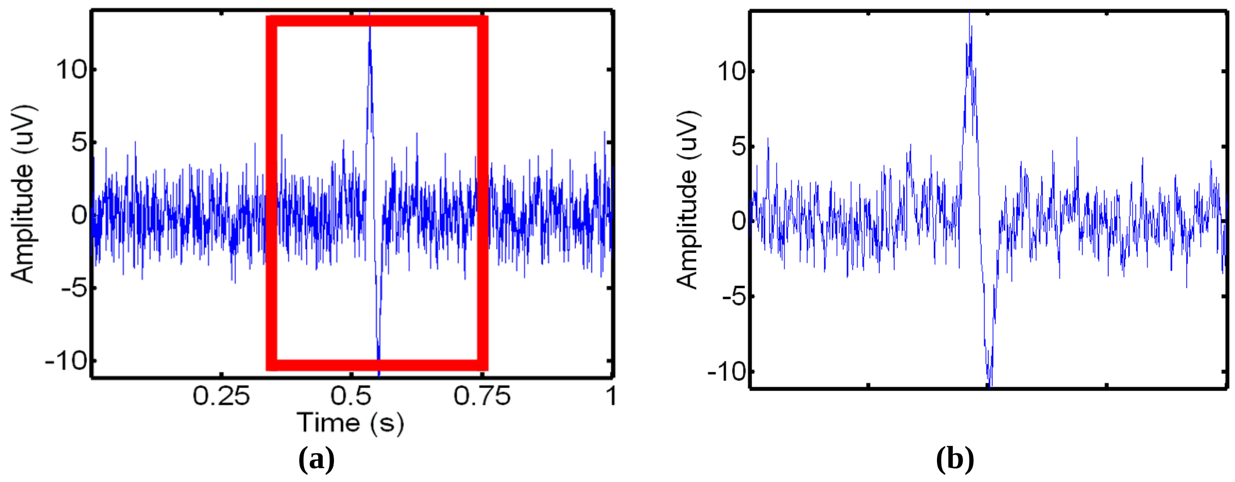
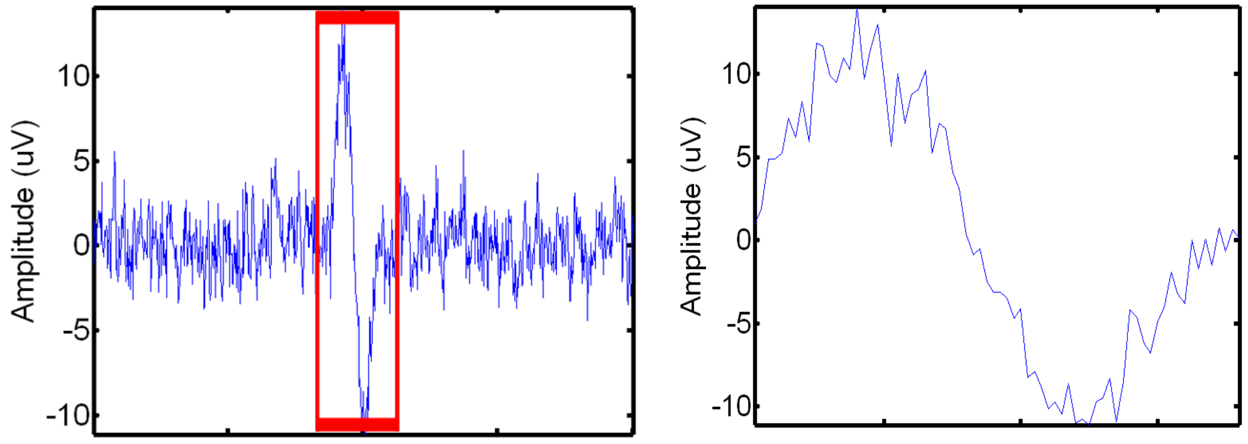


Figure 11. Extraction of the window containing ECG noise from the EMG signal. The signal with the identified window is shown in (a). The extracted window from (a) is shown in (b).



(a) **(b)**

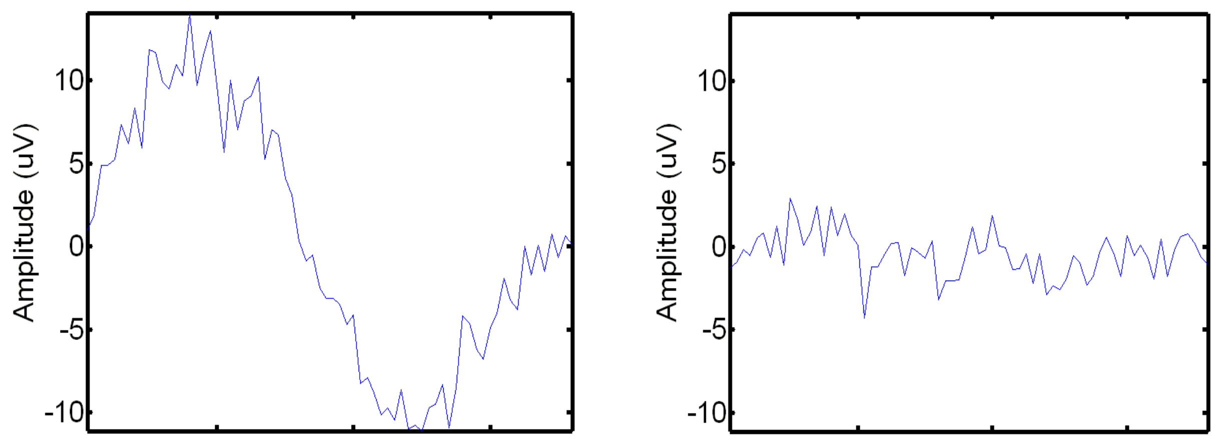
Figure 12. Refinement of the window containing ECG noise from the EMG signal. (a) The initial extracted window. (b) The refined window.

The binary ECG representation is used to extract the individual windows of ECG noise from the corrupted EMG signal. There is one main problem with using the aligned binary representation of ECG in an EMG signal. The binary mask from the raw ECG is based solely on smart thresholding. Therefore, although the location of the ECG spike in an EMG signal is fairly accurate, the width of ECG is based on assumption that the entire QRS complex is larger than the threshold. This assumption is not always true; therefore the binary masks from the binary ECG signal are analyzed. Each ECG binary mask referred to as an ECG pulse is assessed to ensure that it is at least 1000 data points (0.5 seconds) long. If an ECG pulse is not that long, then a new window is created from the start of the pulse. This new window stretches 500 points bi-directionally, hence ensuring that each of the binary mask has at least a 1000 data points. For each ECG spike, the signal inside of the window from the binary mask is extracted. Figure 11 illustrates the local window extraction process.

A main task in preparing the localization functionality is to determine if the specific ECG spike lies during an event. An event is characterized as the area of the signal undergoing active muscular contraction/activity. These parts of the signals have a highly unstable baseline. Due to the high amplitude caused by the activity burst in EMG signal, the ECG spikes are also more unstable and are hidden better in events. The event area of the signal is also the most sensitive to change, therefore even slight over-filtering negatively impacts this area of the signal easily. So, in earlier designs, it was critical to identify the area of the EMG signal with the event area. The filtering in this area of the signal is generally less invasive, in order to preserve as much of the EMG activity signal as possible. In order to find out what areas of the signal has the active EMG event; a mask using the pre-determined event markers was created. Each ECG spike was labeled as whether being part of an event, or being part of event free EMG signal. These labels were determined based on the event mask.

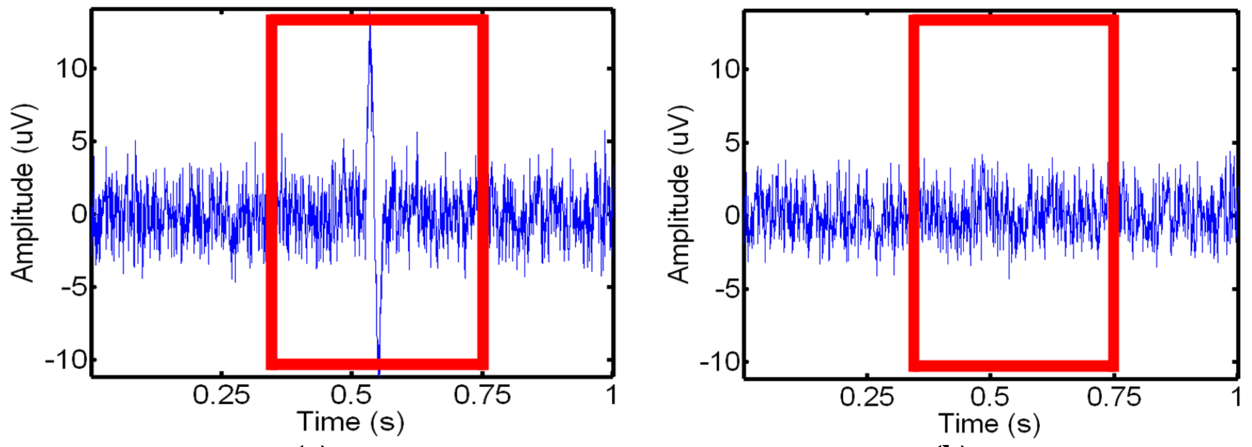
A refinement is performed inside each window to ensure that only the ECG spike is filtered from the data. Within each of these windows the absolute value of this interval is taken for an easier analysis. A smoothing average filter of 10 points is then applied to improve the profile of the window. This smoothing average filter, once again, is a dynamic window that averages 5 neighboring points bi-directionally to create a smoother gradient. The boundary values are not wrapped around, therefore have a shorter window. After the window containing ECG spike is rectified and smoothened, the maximum value in the window is found. This maximum value represents the position of the center of the ECG spike. The window is split in half based off the maximum, since the maximum is assumed to be the center of the ECG spike. The two halves of the window are split once again to create 4 small windows representing ECG. It is important to note that the window sizes are not always equivalent, because the maximum point (center of ECG) does not always lie in the middle of the ECG window. The mean values of the two outside sub-windows are found for refinement calculations. The first step in refinement is to create a simple threshold mask that represents potential EMG bands in the ECG signal. These bands are represented from the mean of the outer window. Any value below this mean is considered to have low energy, and hence labeled as EMG signal. These values are binarized to 0 for the ECG mask, since they are not part of the ECG spike. The remaining values are binarized to 1 to show area of active ECG (QRS complex) spike. This process is repeated for the other window. The second step is to refine the window by iteratively expanding the window outwards bi-directionally from the center of the ECG spike. The window expansion is stopped, when the ECG spike mask hits 0 locating a non QRS-complex of the ECG. Figure 12 (a) demonstrates the local window before refinement, and (b) demonstrates the local window following refinement.

Once the refined window is determined, these windows of areas containing the QRS complex of the ECG spike are extracted and then filtered using the filtering technique explained in section 2.3. This extraction and filtering process is visually shown in Figure 13 (a) and (b) respectively. It is important to note that the neighboring areas outside of the refined window are not extracted or filtered. Hence, ensuring that majority of the EMG signal is not filtered incorrectly, and valuable information is not removed. After filtering the extracted area, this area is then re-inserted into the signal, replacing the un-filtered spike, as shown in Figure 14 (a) and (b). In some rare cases, where the amplitude of the QRS complex is far greater than the EMG signal, the morlet wavelet filter does not effectively separate the ECG spike from EMG signal. Hence, there are small spikes still visible after removal, especially since the filter takes extra care in not removing potentially useful signal. Therefore, the filtered areas of the spikes are post-processed to ensure that the entire spike is removed. To post-process this signal, the energy of the QRS spike and the surrounding EMG is found within the unrefined ECG window. This energy is defined as the same energy calculation used in wavelet analysis. This energy is composed by 50% weight of mean and 50% weight of maximum. Due to the high RMS and maximum in the QRS spike, any remaining spike will have a significantly higher energy than the surrounding EMG signal. The ratio of the energy from QRS spike, and the surrounding EMG signal is calculated. The remaining QRS spike is then rescaled down based off this ratio, in order to ensure that the filtered signal does not damage or cause disturbances in the flow of the signal. It is important to note that this post-processing only occurs in rare circumstances, when the algorithm detects a big difference in the energies of the filtered spike and the surrounding EMG. It is ethical for us to rescale the remaining spike, since the remaining spike does not have ECG frequency wavelets; instead it is remnant of Gaussian noise induced by the high amplitude ECG spikes. Therefore, rescaling the remaining spike instead of removing the frequency composition is an acceptable and effective approach. Table 4 summarizes the algorithm behind localized signal filter.



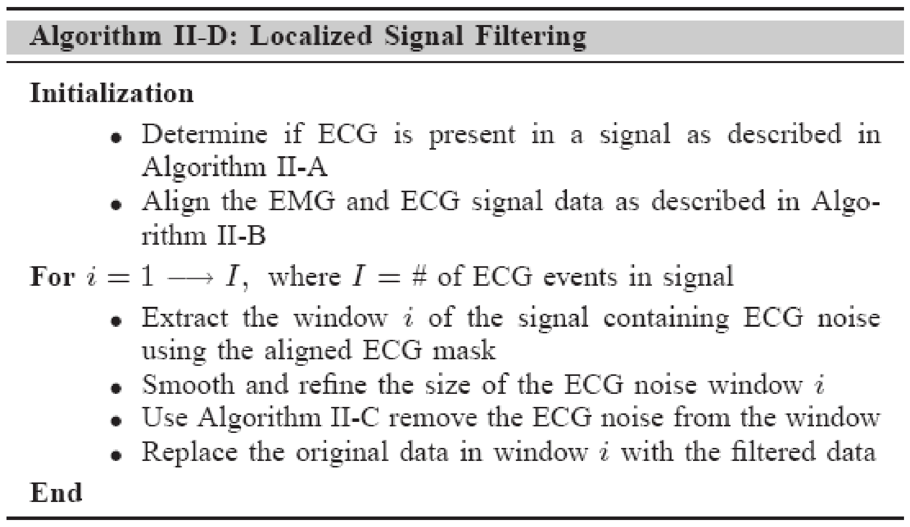
(a) **(b)**

Figure 13. Wavelet filtering of the window containing ECG noise from the EMG signal. (a) The window before wavelet filtering. (b) The window following wavelet filtering.



(a) (b)

Figure 14. Removal of ECG noise from the EMG signal. The extracted window is highlighted for visual clarity. (a) The initial signal. (b) The final signal after local filtering has been applied.



```
Algorithm II-D: Localized Signal Filtering

Initialization
    • Determine if ECG is present in a signal as described in Algorithm II-A
    • Align the EMG and ECG signal data as described in Algorithm II-B
For $i = 1 \longrightarrow I$, where $I$ = # of ECG events in signal
    • Extract the window $i$ of the signal containing ECG noise using the aligned ECG mask
    • Smooth and refine the size of the ECG noise window $i$
    • Use Algorithm II-C remove the ECG noise from the window
    • Replace the original data in window $i$ with the filtered data
End
```

Table 4: Overview Algorithm for Localized Signal Filtering

III. Experimental Results

3.1 Validation on Synthetic Phantoms

Type I Phantom Generation

Electro Cardiogram spikes were added through the help of pre-developed ECG simulation software in Matlab (Karthik 2003). Since, this approach uses Fourier series to simulate the ECG spikes; it was an easier task in modifying it to match the profile of the ECG spikes found in an EMG signal Figure 15 (a). In order to appropriately modify the ECG spike's profile, the P-wave, Q-wave, and even the QRS complex were suppressed. The S-wave profiled the negative spectrum of the ECG spike, and the T-wave profiled the positive spectrum of the ECG spike. The duration of these waves were reduced to 1/20th of a second, which matches the physiology of ECG spikes in an EMG signal. The S-T interval was also removed in order to smooth transition from depolarization to re-polarization of an ECG spike. Figure 15 (b) shows the simulated ECG spikes at specified amplitude. The benefit of using Fourier series to simulate the ECG spikes is the ability of completely modifying the ECG spikes to match any profile that is required. This freedom in simulation helped ensure that the filtering software was robust enough to handle many different profiles of ECG spikes. For example, the modified ECG simulation approach allowed me to modify the positive, and the negative amplitude of the ECG spike. S-wave represented the positive amplitude of the ECG spike, while the T-wave represented the negative amplitude of the ECG spike. This allowed me to be able to simulate top-heavy ECG or bottom-heavy ECG. This was an important step to validate the filtering approach, since some muscles in real data have top-heavy ECG corruption, while others have bottom-heavy ECG corruption. It is fairly rare to find a completely even/symmetric ECG spikes in an EMG signal.

Two more discrete benefits of using a mathematical modeled Fourier series instead of real ECG is the ability to invert the ECG spikes, and the ability to change heart rate easily. Since, the ideal simulated ECG spikes are symmetric around the 0 baseline; it is fairly easy to invert the signal. Multiplying the signal with a negative mask inverts the signal, hence inverting the ECG. This is important, since some ECG spikes have shown to be inverted in some cases in real signal. Changing the heart rate of the ECG is also fairly easy, since the Fourier series is frequency based modeling. By increasing the frequency of the Fourier waves, the heart rate can be effectively increased. However, due to the complex nature of multiple Fourier series, it is imperative to keep the duration of the waves proportional to each other, in order to create the overall ECG signal. The duration of each wave, and the interval amongst each other was fine tuned to create the ECG spike shown in [Figure B]. Huge disruption in proportion of these durations messes with the entire ECG spike.

In order to test the proposed approach, simulated (phantom) signals were generated. The flat baseline for the raw EMG was created through the use of EMG LAB software (McGill, Lateva et al. 2005). This provided a flat unit EMG band that could be manipulated to develop a more realistic model. I was able to re-scale this flat band to any value, using a set multiplier. This band was rescaled to a physiologically relevant +/- 5 mV amplitude. In order to add EMG burst activity that occurs during muscular contraction, I decided to use a sinusoidal model. At first, the EMG band was split into seven separate windowed areas. The contraction bursts were placed in even window area of the signal. This was done in order to create room for three different muscular contractions. This was important in order to ensure that there was sufficient amount of non-event EMG in-between the spikes, where the ECG detection algorithm runs. Although, the window sizes were equal for spacing issues, the EMG burst length still was partially randomized to ensure that each contraction had a different size. In, order to simulate the profile of the burst, a sinusoidal phase was used. A sine wave of the length of the target contraction was created. In order to create a different shape for each contraction, a random sinusoidal phase in the domain of { $\frac{\pi}{random}$ to π } was used. To properly amplify the EMG signal into the sinusoidal profile, a signal mask was needed to be created. The signal mask consisted of the EMG baseband signal at the area of the EMG contraction, and the rest of the signal consisted of null (0). An inverted signal mask was also necessary to create, which had the signal everywhere but the area of the contraction. The area of contraction burst consisted of null (0) values in this inverted signal mask. The sinusoidal wave created previously was multiplied by the signal mask to amplify the signal into a sinusoidal profile (hence creating a burst) at the contraction area only. This signal was then added to the inverted signal mask, which represented the standard EMG baseline signal at the areas outside of the EMG contractions. The amplitude of the sinusoidal wave effectively determined the EMG contraction amplitude. In order to further randomize these clustered contractions, a randomized weight of 0.5 - 1 was multiplied onto the signal at each point in the contraction area. The algorithm ensured that there was sufficient space between the contractions, and that the contractions had a randomized burst profile as seen in true signals. Figure 15 (c) shows the raw simulated EMG signal with a single contraction.

In order to add the corrupted ECG spikes onto the EMG signal, I had to ensure that the sampling frequency was the same in both of the signals. I used 1-D linear interpolation to match the low sampling frequency of the simulated ECG to a realistic sampling frequency of 2k Hz that is present in the EMG. The EMG signal was also centered around the 0 baseline in order to ensure that the ECG spike amplitudes would not get altered after the addition of the two signals (due to baseline shift). The EMG signal was centered around the 0 baseline by shifting the baseline offset found in the event free area of the signal. This baseline offset was found by averaging the first 5000 points of the signal. In, order to corrupt this EMG signal, the simulated ECG spikes were added to the simulated EMG. Figure 15 (d) shows the corrupted Type I phantom EMG signal.

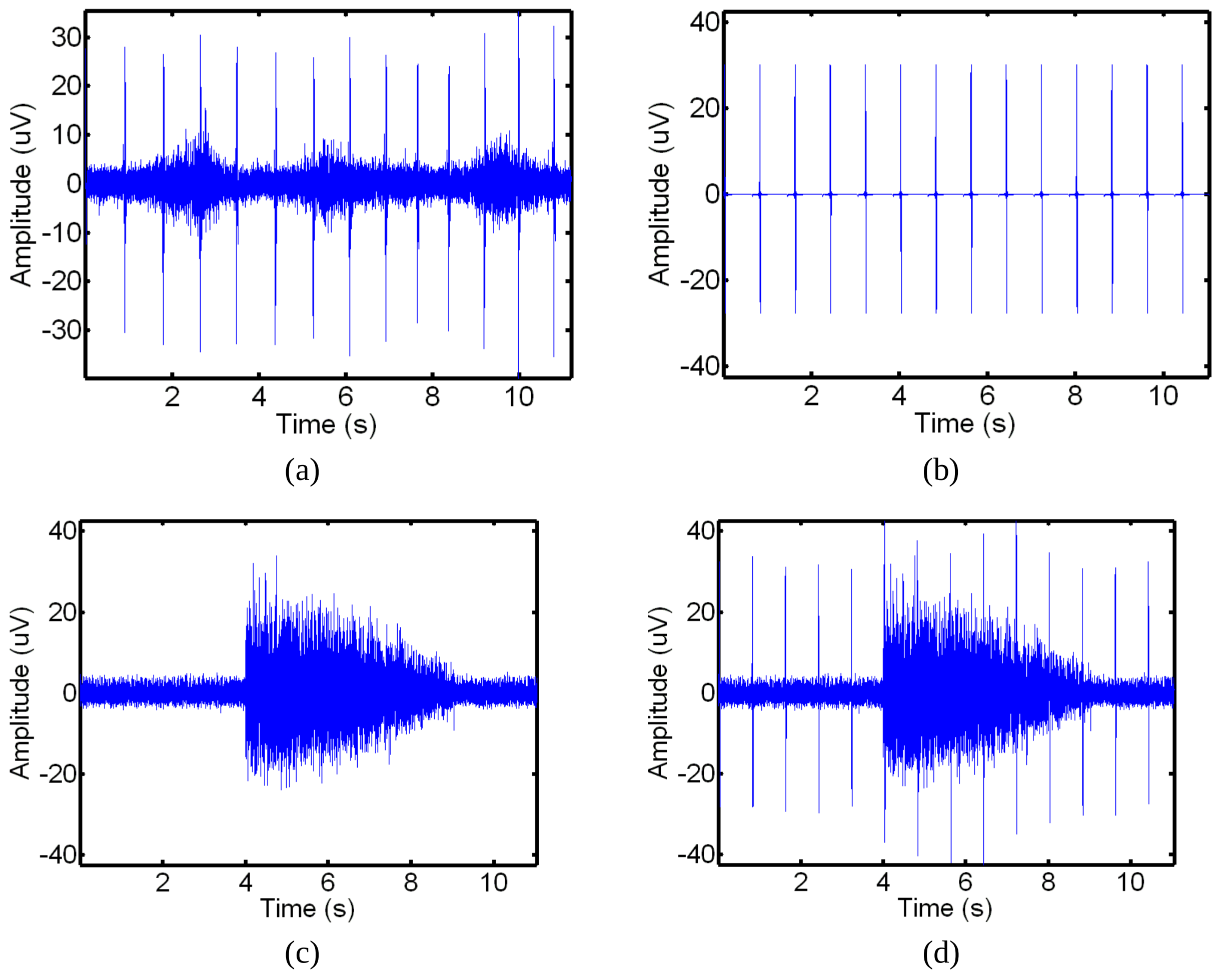


Figure 15 (a) Example of a real EMG signal with real ECG corruption. (b) Generated Phantom ECG spikes. The phantom ECG spikes were modeled based off realistic ECG corruption experienced by EMG signals as shown in (a). (c) Generated Phantom EMG signal based off sinusoidal amplification of EMG baseline. (d) EMG signal in (c) is added to the ECG spikes from (b) in order to generate the corrupted Type I Phantom signal.

Type II Phantom Generation

Due to the relative simplicity of phantom EMG contraction, based off sinusoidal amplification, a Type II phantom was also generated for a more realistic model. Type II phantom consists of real EMG signal that has been physiologically supported as containing no ECG corruption, and simulated (phantom) ECG signals. Hence, the real EMG signal serves as the clean original signal used in data analysis. It has been physiologically supported that EMG signals from the lower limbs, such as the leg muscles do not have ECG corruption in the raw signal (Liu, Wang et al. 2008). Hence, these signals can be used as the clean EMG signal in the validation process. The advantage of using real EMG signal over phantom EMG signal is a more realistic definition of event-based contraction shape. However, the size or the length of the contractions cannot be controlled using non-phantom (real) EMG signals.

There were minimal steps required to prepare the EMG signal for generating the corrupted phantom signal. The first step is to ensure that the imported real EMG signal is centered over the x-axis. This is extremely important, because as shown previously, the simulated ECG spikes are centered over the x-axis. If both signals are not properly centered, then the positive and negative amplitude of ECG corruption will not be accurate when the two signals are combined for corruption process. In order to center the EMG signal, the mean of the first 5000 points is calculated. These first 5000 points are assumed to be event-free and part of the base-band. Hence, ideally, the mean of these points should be 0, if the signal is properly centered. The actual mean represents the offset that needs to be shifted in order to obtain a centered position. The entire signal is subtracted by this offset, in order to obtain the desired shift in the signal. So, if the offset is positive, then the signal is shifted down to compensate. If the offset is negative, then the signal is shifted up to compensate accordingly.

The second main step is to match the ECG signal length with the pre-determined real EMG length. In order to add two signals together, they have to be the same length. In a Type I phantom generation process, both the simulated EMG and the simulated ECG lengths can be easily controlled. However, in the Type II phantom generation, the EMG signal length depends on the imported real EMG signal, which can change depending on the signal itself. Therefore, the EMG signal length is determined in seconds, based on the sampling frequency of the signal. This length in seconds is then used to generate the appropriate phantom ECG signal. The phantom ECG signal is then added onto the real EMG signal to corrupt the signal. Due to small shifts in offsets from signal manipulation, the final step is to re-center the final corrupted signal using the same steps described above. Figures 16 (a) and (b) show the clean and corrupted Type II phantoms respectively.

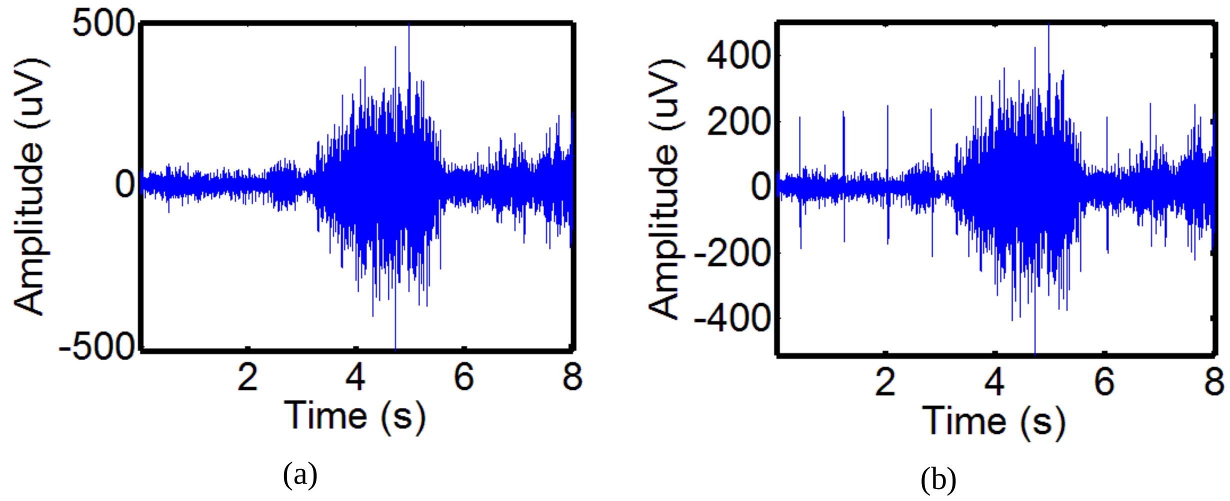


Figure 16 (a) Real EMG signal known to have no ECG corruption. This signal is used as a reference clean signal for a more realistic Type II phantom signal. (b) Phantom ECG corruption is added to the EMG signal, in order to generate the corrupted Type II phantom signal.

3.2 Validation Metrics

<u>Reduction of the RMS in a signal</u>

The Root Mean Square (RMS) is a well known statistical measure in signal processing. Within windows that contain the presence of ECG, the magnitude of the RMS before and after removal is a good indicator of the improvement in a signal. The RMS is defined for a signal, x, of length n as shown in Equation 10. The RMS error is then calculated as shown in Equation 11, where x_{orig} is the original signal and x_{proc} is a processed signal (e.g. a corrupted signal or a filtered signal). In other words, this is a direct form of error calculation between the original (clean signal) and the filtered/corrupted signal. It is a good tool to assess how close our filtered signal is to the original signal.

$$x_{rms} = \sqrt{\frac{1}{n}(x_1^2 + x_2^2 + \cdots + x_n^2)} \qquad \textbf{EQ 10}$$

$$E_{rms} = |x_{orig} - x_{proc}| \qquad \textbf{EQ 11}$$

It is additionally important to examine this error within areas that do not contain ECG noise. This provides a measure of the change in the areas that do not contain ECG during processing. While a local filtering technique has no change in areas that do not contain ECG, global filtering will have an impact within these areas and can lead to unnecessary signal damage and removal of critical information. In validation on phantoms, the percent change can be calculated for the recovery of the original signal from the corrupted signal. The percentage change is calculated as shown in Equation 12, where $E_{rms,filt}$ is the RMS error of a filtered signal and $E_{rms,corrupt}$ is the RMS error of the synthetically corrupted signal. Within an ECG noise window the a positive percentage change is an improvement, while outside an ECG noise window a positive percentage change indicates the introduction of additional noise into the signal. Within ECG windows, a positive reduction in RMS of the signal means that the ECG spike is successfully removed or reduced.

$$\%\text{Change} = 100 * \left(1 - \frac{E_{rms,filt}}{E_{rms,corrupt}}\right) \qquad \textbf{EQ 12}$$

Variance-indexed strain homogeneity

Restoring local signal variance is an additional important parameter. In addition to restoring the overall signal energy, restoring the variance indicates that the local properties of the signal are accurately recovered. To analyze the local signal variance, a window of width 7 is progressively moved over the signal from the beginning to the end. Within this window the variance in the signal is computed and stored for each location. The stored matrix of these variances is referred to as the variance profile of the signal. By examining the percentage change in the variance profile between signals, it can be determined how similar the homogeneity of the signals is to one another. Error in variance is calculated as shown in Equation 13, where V_{orig} is the variance matrix of the original signal and V_{proc} is the variance matrix of the processed signal (e.g. the corrupted signal or the filtered signal).

For the phantom signals, the percent change can is calculated between an original signal and a processed signal. The percentage change is calculated as shown in Equation 14, where $E_{var,filt}$ is the variance error of a filteredsignal and $E_{var,corrupt}$ is the variance error of the synthetically corrupted signal. Similar to the percentage change of RMS, within an ECG noise window the a positive percentage change is an improvement, while outside an ECG noise window a positive percentage change indicates the introduction of additional noise into the signal.

It is important to note that not all signals are evenly divisible by a window of width 7. The middle of the window was appointed as the fourth point, with three neighboring points on both sides. Therefore, the boundary conditions were handled accordingly. The local variance of the first three points was appointed as the local variance of the fourth point. The local variance of the last three points was appointed as the local variance of the [last-3] (fourth to last) point. In order to find a representation of the variances within ECG spikes, the calculated variances at points lying within an ECG spike were averaged together. In order to find the total variance of signal lying outside of the ECG spike, the local variance of points lying outside of ECG spikes were averaged together. There was also an event mask created in the signal, which located areas of event oriented EMG signal, and event free signal. This mask was used to identify whether each window (within ECG spike or outside of ECG spike) was located in an event or not in an event. It helped identify data for a more in-depth analysis. The identification for the window was determined based on the front-edge of the target window. So, if the front edge of the window was labeled as being part of an event, then the whole window was labeled as an event-based window of data. In order to compensate for the windows located on the edges of real event markers, the event marker window was extended an extra 3000 points bi-directionally. Therefore, only spikes those are truly located in a flat-baseline event-free areas of the signal are counted as being event-free.

$$E_{var} = |V_{orig} - V_{proc}| \qquad \textbf{EQ 13}$$

$$100 * \left(1 - \frac{E_{var,filt}}{E_{var,corrupt}}\right) \qquad \textbf{EQ 14}$$

3.3 Visual Results

The first step in identifying the effects of the proposed filtering approach is to visually compare the results from filtering the phantom signal. As explained in Section 3.1, Type I and Type II phantom signals were generated for data analysis. Figure 17 shows the effect of the filtering process on Type I phantom signals. Figure 17 (a) shows the original Type I phantom clean EMG signal. Figure 17 (b) shows the signal after corruption. It is important to note that the height of ECG changes over the signal, due to the varying EMG in the contraction phase of the signal. Figure 17 (c) shows the effects of global filtering on the EMG signal. All of the ECG spikes are visually eliminated from the signal. Closely inspecting the EMG burst of this signal clearly shows that the signal has lost the smooth edges in the burst profile. A lot of the finer details in the EMG are also lost after mass-global filtering of the signal. However, the shape of the burst remains intact due to the simplicity of the sinusoidal profile of the phantom EMG burst signal. Figure 17 (d) shows the effects of the local filtered EMG signal; it is clearly evident that the ECG spikes are also efficiently removed in this signal. Referencing this with the original signal (a), the edges seem smooth, and the EMG profile seems to be recovered without introducing significant error. The shape of the EMG signal is completely preserved, and there is very little loss in content in the signal after the filtering process.

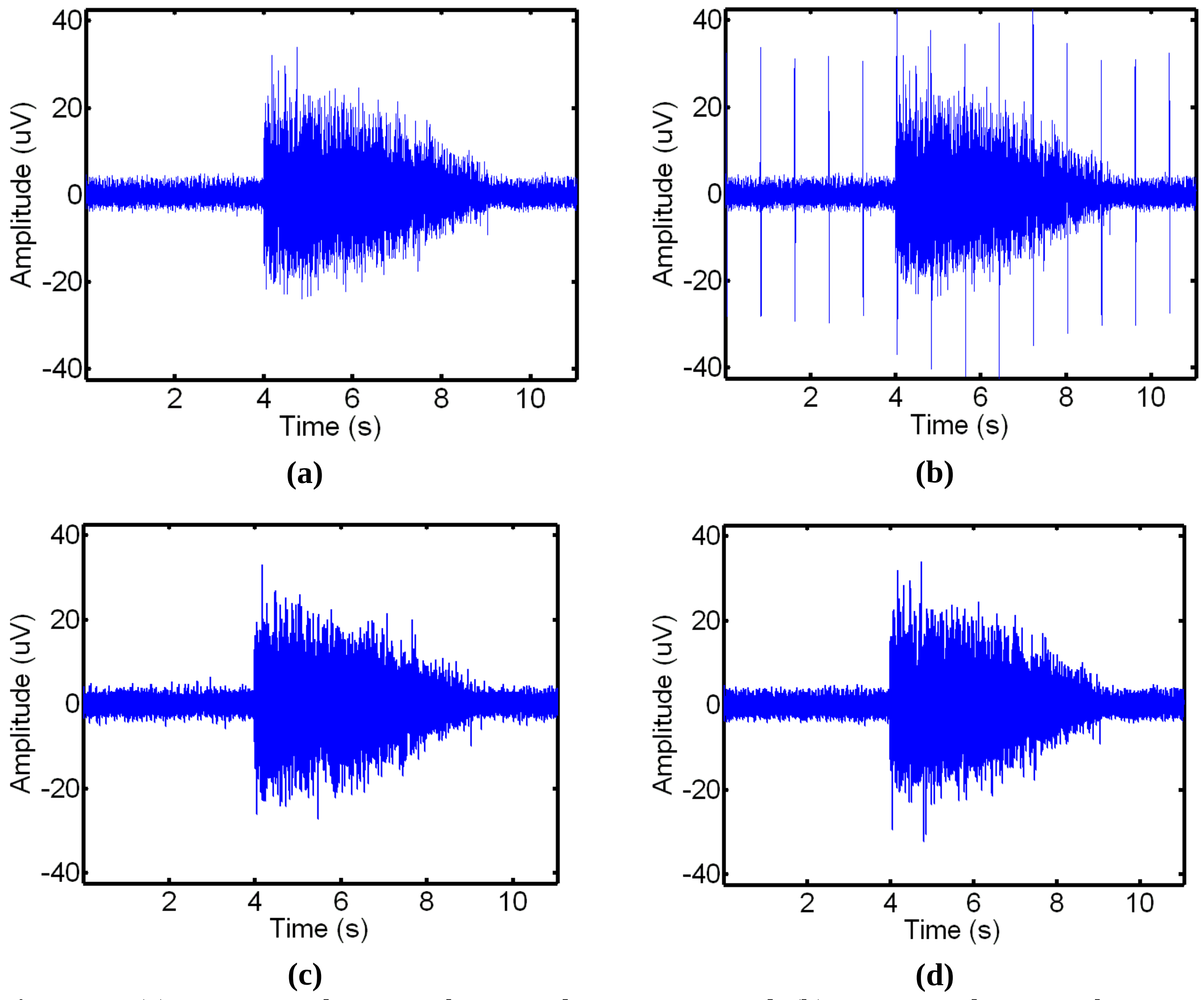
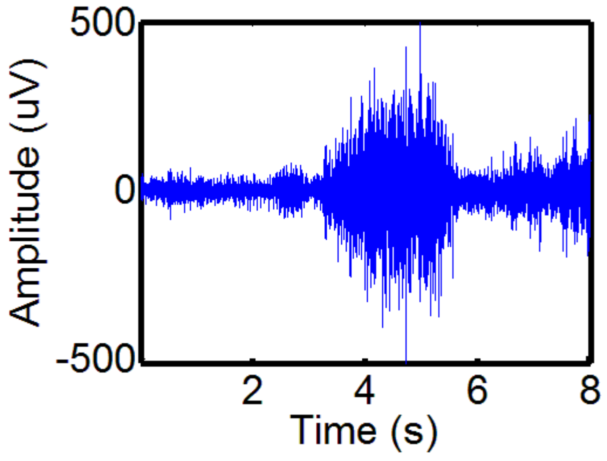
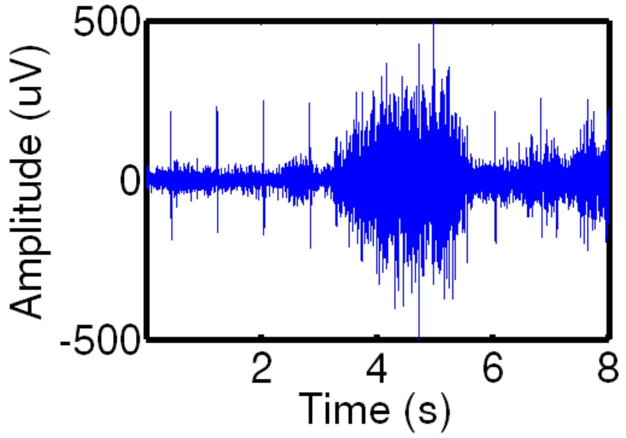


Figure 17 (a) A generated Type I phantom clean EMG signal, (b) a corrupted Type I phantom EMG signal, (c) the signal in (b) after global wavelet filtering has been applied, (d) the signal in (b) after the proposed filtering method has been applied. The ECG noise spikes have been significantly reduced in the filtered version of the signal, while the original signal's envelop has been preserved.

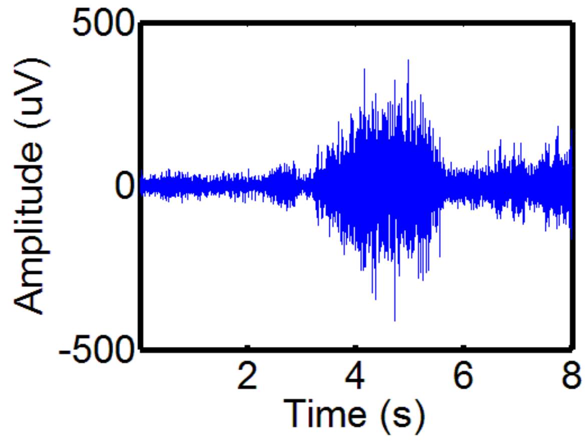
The effect of filtering is more visually prevalent in Type II phantom signals, because these are realistic physiological based EMG signals. These EMG signals are real in-vivo signals that were manually corrupted. Therefore, the shape of the EMG bursts is complex, and is shaped like a true frequency burst for a signal. Figure 18 shows the effect of the filters on the Type II phantom signals. Figure 18 (a) shows the original clean Type II phantom, and the corrupted signal is shown in Figure 18 (b). The shape of the burst in the signal is jagged and randomized (not simulated). Hence, the ECG spikes are hidden fairly well in the EMG event (b). This is a prime real-life example, where the ECG is not clearly visible or modeled in the entire signal. The global filtering process, as shown in Figure 18 (c) is able to successfully visibly remove the ECG corruption from the signal. However, it also has seemed to change the shape of the EMG signal by compressing it down. The slope of the start of the EMG burst profile, which is essential in quantifying the muscular activity, has been compromised by global filtering. This slope has been substantially decreased to show a visual change from the original signal. The proposed approach, as shown in Figure 18 (d), also visually eliminates all traces of ECG artifact shown in Figure 18 (b). More importantly, it also does not change or compromise the burst profile that is shown in the original signal (Figure 18 (a)). This is an important visual benefit of using the proposed localized approach rather than the standard global filtering approach. The algorithm's ability to narrow its filter down to the true ECG spike is attested by the non-removal or modification of the EMG's innate spikes that are present in the burst profile.



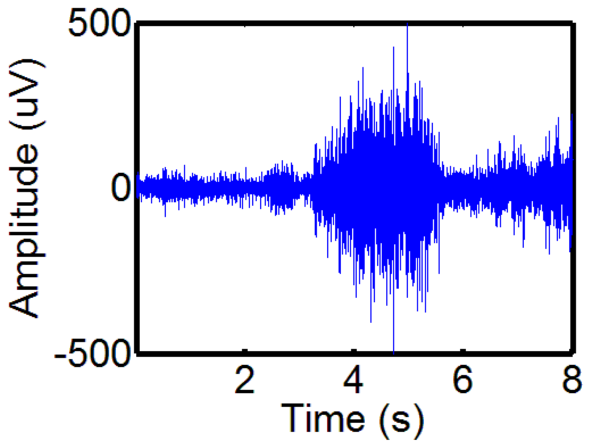
(a)



(b)



(c)



(d)

Figure 18 (a) A generated Type II phantom clean EMG signal, (b) a corrupted Type II phantom EMG signal, (c) the signal in (b) after global wavelet filtering has been applied, (d) the signal in (b) after the proposed filtering method has been applied.

3.4 Analytical Results on Phantom Signals

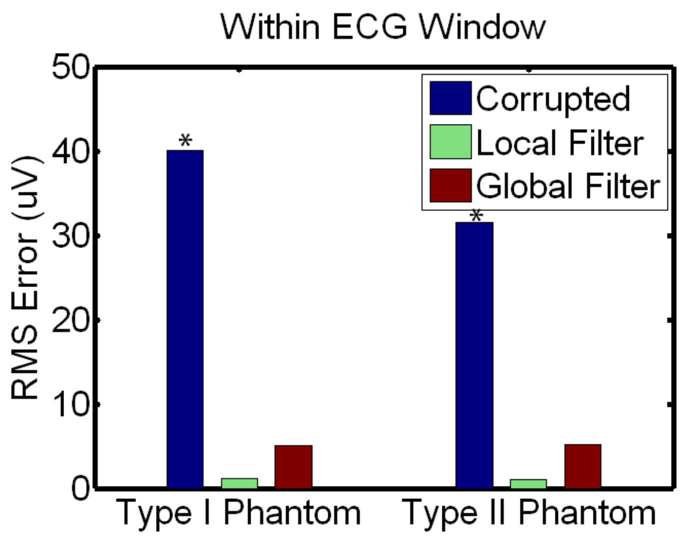
Analytical Results on the Overall Phantom Signal

Visual results provide an easy conceptual representation for expressing the advantages of the proposed localized filtering approach. However, the visual results are subjective analysis, which cannot be used alone as a basis for expressing advantages of an approach. Hence, the validation metrics shown in Section 3 were used to analyze both Type I and Type II phantom signals. Analytical results provides concrete basis to assess the effects of the filter.

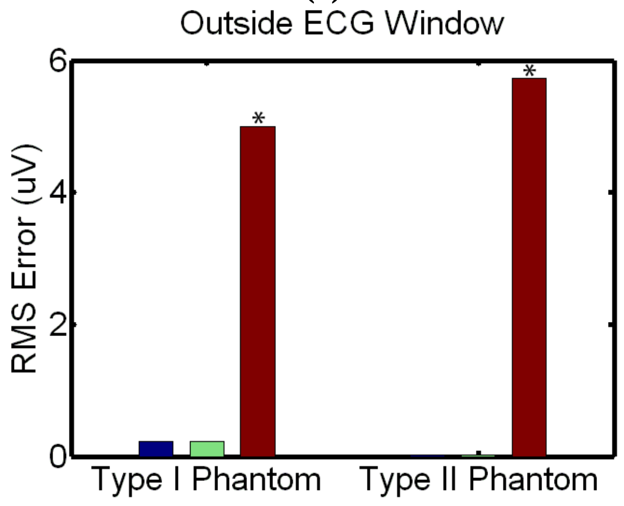
To validate the technique, the phantom signals were processed using the proposed filtering technique. Figure 19 shows the results of the filtering using both a global and local filtering. The global filtering is the wavelet removal technique applied without localization to the individual ECG noise windows. The corrupted error is the absolute error (AE) in the specified validation metric between the corrupted signal and the original clean signal. The absolute error (AE) in the local filter and the global filtered signals are also calculated based off the original (clean) signal. Signals with significantly large error, calculated with a paired T-test (p-value 0.05), are labeled with an asterisk (*).

Figure 19 (a) shows the Root-Mean-Squared error analysis in both Type I and Type II phantom analysis within the ECG windows. Both of these phantom signals have significantly large error in the corrupted signals. This error is attributed to the ECG spikes introduced in the corruption process. The proposed local filtering approach significantly reduced the RMS error within the ECG window, hence producing a 97% and a 96.7% improvement in Type I and Type II phantom signals respectively. On the other hand, the global filtering only provides an 87.4% and 83.7% improvement in RMS error within the ECG windows. The error induced from the filtering process is depicted in Figure 19 (b), which shows the RMS error outside of the ECG windows. Ideally, the filtering process should not alter the EMG composition outside of the ECG windows. The error induced from the proposed local filtering process is 0% in both the Type I and Type II phantom, while the error induced from global filter is significantly large (>200%) error. This essentially shows the short comings of the global filtering process, which adds significantly large error in the native EMG signal during the filtering process.

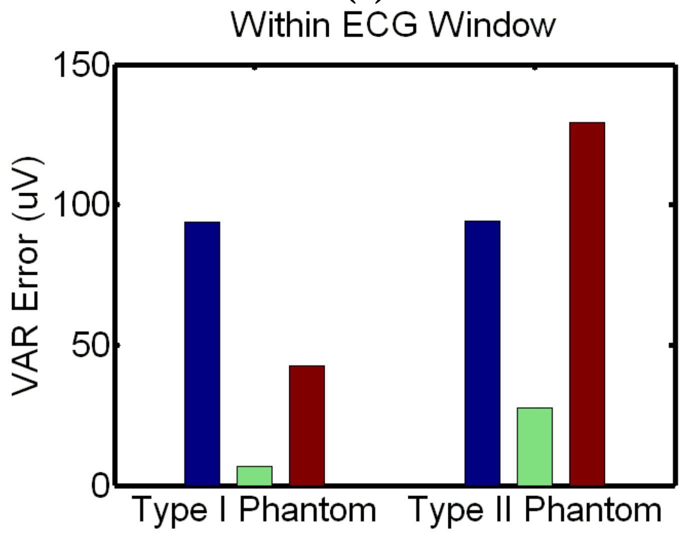
Figure 19 (c) shows the filtering effect on variance-indexed strain homogeneity within the ECG windows in the signal. This validation metric, although important, is fairly sensitive to accurate reconstruction of signal. It ensures that the homogeneous rhythm of the signal is not greatly altered in the filtering process, which is generally overlooked visually. The local filter does an excellent job in preserving the homogeneity of the signal with a 92.6% and a 70.6% improvement in Type I and Type II phantom signals. There is a slightly less improvement rate in Type II phantom signals, because of the complexity of homogeneity in real EMG signals. The global filter does an extremely poor job at preserving the homogeneity of the signal in the ECG windows of the signal after filtering. There is only a 54.7% improvement in the Type I phantom signal. There is even a -37.4% improvement in the Type II phantom signal. This means that the global filter deteriorated the homogeneity of the signal inside of the ECG window after filtering it. Not surprisingly, the global filter also added large homogeneity error outside of the ECG windows in both Type I and Type II phantom signals (>200%). This large error in homogeneity can be attributed to the complexity of the homogeneity of native EMG signal. Hence, accurately reconstructing EMG signal with the same homogeneity is extremely difficult. Hence, this supports the importance of developing an accurate method to target the QRS complex of the ECG spikes in localizing the filtering process.


Within ECG Window
RMS Error (uV)
Corrupted
Local Filter
Global Filter
*
*
Type I Phantom
Type II Phantom
0
10
20
30
40
50

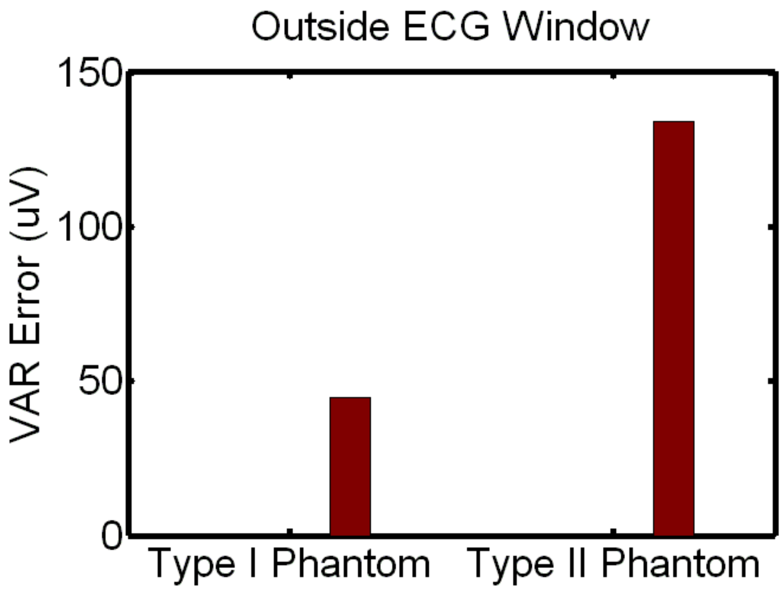
(a)


Outside ECG Window
RMS Error (uV)
*
*
Type I Phantom
Type II Phantom
0
2
4
6

(b)


Within ECG Window
VAR Error (uV)
Type I Phantom
Type II Phantom
0
50
100
150

(c)



(d)

Figure 19 (a) RMS metric error within ECG windows in Type I phantom and Type II phantom. This graph shows that Local Filter does a better job than global filter at significantly reducing the ECG corruption within the ECG signal. (b) RMS metric error outside of ECG windows. This figure shows that Global Filter introduces significantly high error as opposed to the proposed Local Filter in areas of EMG signal outside of ECG spikes. (c) Variance-indexed strain homogeneity metric error inside of ECG windows. This figure shows that the proposed local filter has a lot higher efficiency in preserving the signal homogeneity compared to the global filter within ECG windows. Global filter deteriorates the signal homogeneity in the Type II phantom. (d) Variance-indexed strain homogeneity metric error outside of ECG windows. The global filter introduces large magnitude error in homogeneity of the signal outside of areas of ECG spikes, while the proposed local filter preserves the signal homogeneity outside of ECG windows. The asterisk (*) represents significantly high error compared to the original signal, using a p-value of 0.05

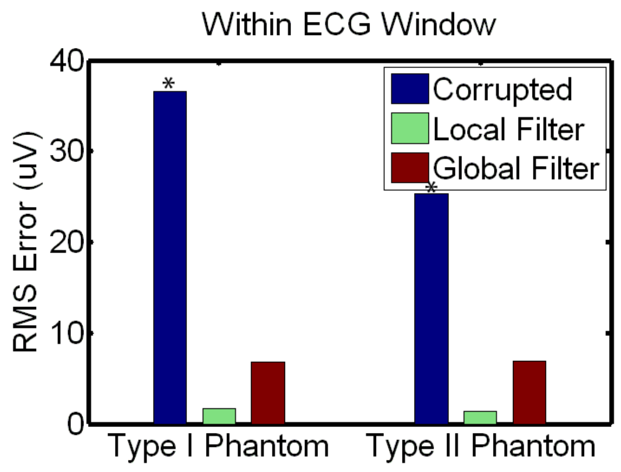
Analytical Results on Event Area Phantom Signals

Most in-vivo EMG signals have activity bursts in the signal that show muscular activity. These bursts, as explained before, tend to make ECG detection and removal much harder due to their complex shape and frequency composition. Additionally, these bursts tend to be the most critical component of the signal, when performing analysis on the EMG signal, as they are direct representation of magnitude for muscular activity. Hence, it is critical to further separate the analysis of the filter into effects on areas of the signal, which contains the event oriented bursts, and event-free areas of the signal (standard baseline EMG signal). Ideally, the proposed localized filter approach will behave similarly on both areas of the signal, further demonstrating its robustness.

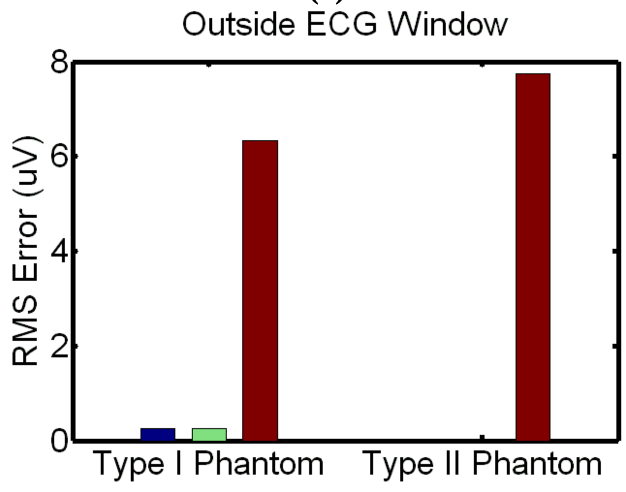
Figure 20 (a) shows the analysis on the validation metrics for only the event portion of the signal. Within events, the RMS error produced from the ECG spikes was significantly reduced by both the local and the global filter. The local filter had a 95.3% improvement, while the global filter only had an 81.4% improvement in the Type I phantom signals. Comparing these values to the overall analysis seems to be almost identical. This is a strong support for claiming that there is no bias in the overall analysis of the filter's efficiency from the flat non-event areas of the signal. Type II phantoms also show similar results in the signal, where the local filter shows a 94.4% improvement, and the global filter only introduces a 72.5% improvement. In the critically sensitive area, such as an event burst, this difference in % improvement for reduction of error caused by ECG spikes is a critical factor in determining the benefits of the local filter.

The biggest problem that was shown earlier was the large external artifact produced by the global filter in areas outside of ECG windows. This innate EMG area is extremely sensitive in an event, as it determines the shape of the EMG burst. Modification of height or the layout of the surrounding EMG signal leads to a visually modified signal. As it has been discussed before, the filters that especially visually show a change in the signal are generally deemed unacceptable in any manner. Therefore, it is critical that the local proposed filtering approach keeps the error outside of ECG windows inside of event as minimized as possible. Figure 20 (b) shows the RMS analysis in these areas outside of ECG windows. This analysis is once again extremely similar to the ones shown in the overall signal. The Global filter introduces a large magnitude of RMS error in areas outside of ECG window (>200%) compared to the local filter. Not surprisingly, there is higher error from the global filter in Type II phantom signals inside of the event than outside of the event. The Type II phantoms are a more realistic representation of in-vivo signal, since it has real EMG contractions. These real EMG contractions that are not mathematically modeled give the global filter a lot harder time in avoiding introducing error. Therefore, it is essential to note that the global filter efficiency is altered based on the realism of the phantom signal, while the local filter is completely robust for any kind of signal it encounters.

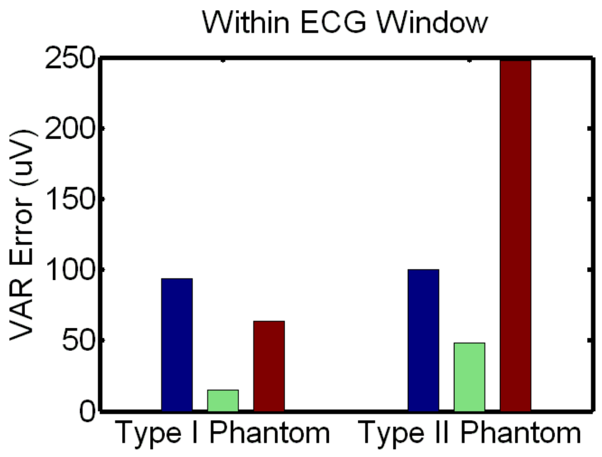
Examining the homogeneity of these signals, as shown by the 7-point local variance, is also an important asset in determining the efficiency of the two filters. As explained earlier, accurately reconstructing the homogeneity of the raw EMG signal is extremely difficult. Hence, local filter which only modifies the ECG spike tends to perform a lot better in this validation metric. This effect is even more prominent in event area of the signal, which has complex and varying EMG. Figure 20(c) depicts the effect on homogeneity of the signal in areas of the event signal within ECG windows. In the Type I phantom signal, the local filter provides an 84% improvement to the signal homogeneity. This improvement is far greater than the global filter’s 31.6% improvement to the homogeneity of the signal after removing the ECG spike. Looking at the Type II phantom, with a more complicated and realistic event shape, the global filter once again deteriorates the signal homogeneity after removing the ECG. Specifically, the global filter has a -147.6% improvement. This deterioration is an unacceptable process compared to the local filter’s 52% improvement in homogeneity. These numbers once again support the claim that the local filter is able to handle different kinds of realistic signal, a lot better than the global filter. This signal homogeneity deterioration is compounded by the global filter in areas outside of ECG windows. Figure 20 (d) shows the introduction of homogeneity error by the global filters in areas outside of the ECG windows (>200%). These analyses, once again, have the same results as the overall analyses of the signal. This means that the overall signal analyses portrays an accurate representation for the effect of the filter on the sensitive event burst of the signal. The overall analysis is not biased by the flat event-free intervals between contraction based events. Hence, the overall analysis discussed in other sections also represents the same effect in the important event area of the signal.



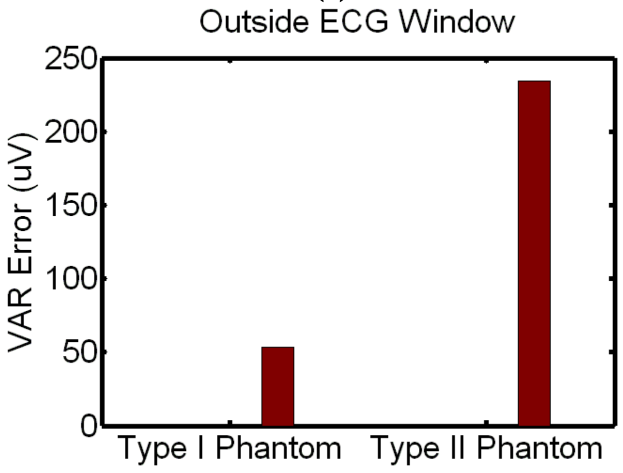
(a)



(b)



(c)



(d)

Figure 20 The following analysis is done on areas of EMG-burst of signals produced by muscular activity events. (a) RMS metric error within ECG windows in event areas of the signal Type I and Type II phantoms. This graph shows that Local Filter does a better job than global filter at significantly reducing the ECG corruption within the ECG signal. (b) RMS metric error outside of ECG windows in event areas of the signal. This figure shows that Global Filter introduces high error as opposed to the proposed Local Filter in areas of EMG signal outside of ECG spikes. However, this error is not significant, due to the varying deviations introduced from different kinds of bursts. (c) Variance-indexed strain homogeneity metric error inside of ECG windows. This figure shows that the proposed local filter has a lot higher efficiency in preserving the signal homogeneity compared to the global filter within ECG windows inside events. Global filter deteriorates the signal homogeneity in the Type II phantom, and does a poor job at preserving the homogeneity in Type I phantom. (d) Variance-indexed strain homogeneity metric error outside of ECG windows. The global filter introduces large magnitude error in homogeneity of the signal outside of areas of ECG spikes, while the proposed local filter preserves the signal homogeneity outside of ECG windows. The asterisk (*) represents significantly high error compared to the original signal, using a p-value of 0.05. Overall, the

local filter does a much better job at performance compared to the global filter in the sensitive event areas of the signal.

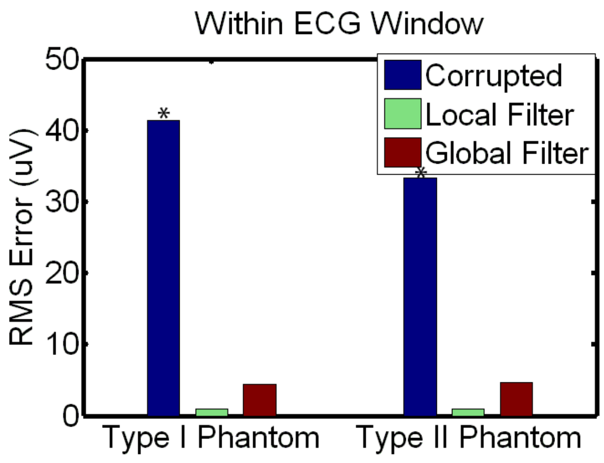
Analytical Results on Event-Free Area Phantom Signals.

The event free area of the signal is also an important asset to individually analyze. The event free area of the signal is generally composed of a long, low amplitude flat band of EMG signal. The effect from the global filter is generally harder to visually assess in this area of the signal, since it leads to a change in amplitude of the band. There is generally no change in shape of the signal, hence the global filter generally seems to be an acceptable filter, and since the ECG spikes are visually removed. However, these areas of the signal also have physiological significance, since certain muscles have specific pattern of relaxation that are critical for certain studies. Therefore, small changes in amplitude of the EMG in the relaxed event-free area of the signal also corrupt the data set entirely.

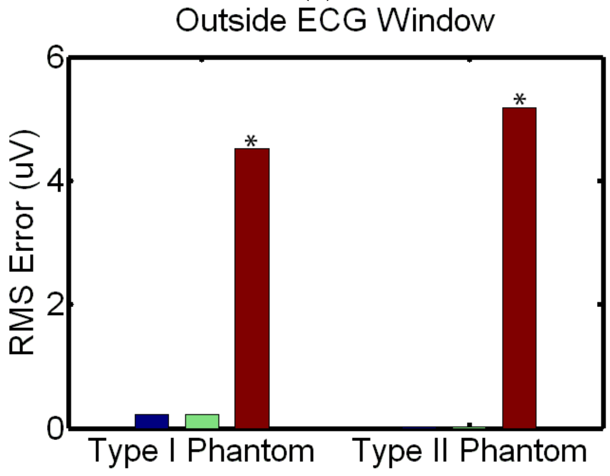
Figure 21 (a) depicts the effect of global and local filters on the RMS error of the signal within ECG windows. Not surprisingly, both the local and the global filters significantly reduce the ECG corruption in both Type I and Type II phantom signals. The local filter has a 97.6% and 97.1% improvement in the two signals. The global filter, on the other hand, has an 89.3% and an 86% improvement. Not surprisingly, both of the filters are more efficient in event-free flat areas of EMG signal. The global filter, which has an easier time locating the proper ECG spikes in the flat area of signal, operates a lot better at event-free areas than event areas. This once again claims the support previously made that the efficiency of most global filters solely rely on the target signal, and are not very robust.

Surprisingly, the global filter still introduces significantly large RMS error in areas outside of ECG. Global filters tend to visually perform better in flat baseline signal, but as shown in Figure 21 (b) there is significantly high RMS error introduced by the global filter. This introduction to error is considered significant, due to the low deviation in this area of the signal. Hence, it's easier to support statistical significance on this flat area of the signal. The global filter consistently introduced large amount of RMS error in this flat event-free area of signal, by potentially modifying the amplitude of the EMG signal in the process of filtration. Although, the magnitude of this error is around 5 uV, in relative comparison it is significantly higher. This error (>200%) from the global filter is once again extremely large compared to the error introduced from the local error. Overall, the RMS error trend is the same as the overall trend, and the event-oriented trend.

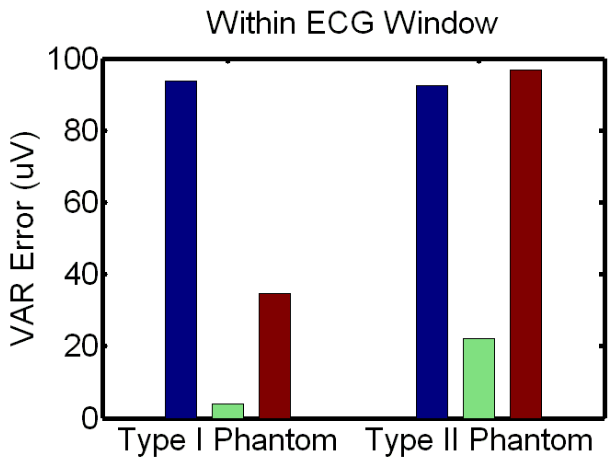
Variance-indexed homogeneity is also shown in Figure 3.2.2 (c) for event-free area of the signal. This analysis follows suit with the other homogeneity analysis explained previously. To reiterate the point, it is hard to reconstruct a signal with accurate homogeneity as the original signal. The local filter has a 95.6% improvement, while the global filter has a measly 64.3% improvement. This once again occurs due to the fact that the local filter narrows its filtering down to the QRS complex of the ECG spike, which has high error in homogeneity anyways. The global filtering removes the neighboring EMG signal, which introduces error in homogeneity to counter-act the error reduced from ECG removal. In the Type II phantom, where there are random EMG spikes in the event-free area compared to the flat baseline in the Type I phantom, the global filter expectedly deteriorates the homogeneity (-4.6%). This deterioration is not as high as what is observed in event area of the signal, because the homogeneity of the event-free area of the signal is generally a lot less random than the event area of the signal. Hence, it is easier to reconstruct the signal with the same homogeneity in the event-free area of the signal than the event-area of the signal. Finally, Figure 3.2.2 (d) shows once again that the global filter also introduces error in areas outside of ECG window due to its globalize filtering process. Overall, it is apparent that the local filter outperforms the global filter in robustness, accuracy of ECG removal, and introduction of processing artifact. The proposed local filter is the ideal choice for sensitive data, such as EMG, where small corruption can lead to inaccurate analysis. It was also found that the local filter outperforms the global filter in both event and event-free areas of the signal. The global filter also performs worse in the crucially important event area of the signal than the event-free area of the signal.



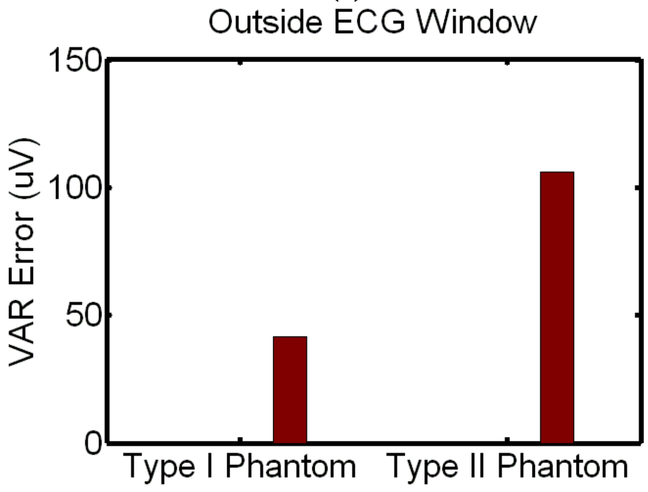
(a)



(b)



(c)



(d)

Figure 21 Validation metric error analyses in areas of event-free signal. (a) RMS error for Type I and Type II phantom signal within ECG windows. (b) RMS error for Type I and Type II phantom signals in areas outside of ECG windows. (c) Local Strain-Indexed Homogeneity error within ECG windows of the event-free signal. (d) Homogeneity error outside of ECG windows of event-free signal. The asterisk (*) represents significantly high error compared to the original signal, using a p-value of 0.05. Overall, the proposed local filter significantly reduces the RMS error produced by ECG spikes. Alternatively, the global filter introduces significant RMS error outside of ECG windows.

3.5 Analytical Test for Robustness

In this text, it has been frequently mentioned that the local filter is more robust than the global filter. It has already been shown in Section 3.4, that the local filter handles Type II phantom signals that are more realistic, far better than the global filter. In order to assess the determining factor that influences the robustness of the filters, the effect of certain factors were analyzed. These factors were ECG: EMG ratio, heart rate, and invert. ECG: EMG ratio is the relation of height of ECG to the EMG signal. Specifically, this height of EMG is the height of the EMG event contractions. The baseline EMG in the event-free area of the signal is ¼ the height of the EMG event contractions. There are four ratios that were analyzed: 4, 2, 1, 0.5, and 0.25. This means that at 0.25 ratio, the ECG spikes are partially hidden in the EMG band, since they are of the same height. At this ratio, the ECG spikes are completely hidden from the EMG event contractions. The second factor, which is heart rate, is very straightforward. In realistic signals, the heart rate varies from the activity that the patient is doing during the event. Heart rate is also not universal to every patient. In order to assess if the heart rate is the limiting factor for the robustness of the filters, heart rates of 60, 80, and 110 were analyzed. Finally, certain filters search for pattern recognition, and have an easier time finding ECG when it is not inverted. In realistic signals, ECG may seem inverted due to the lead-configuration setup. Therefore, the final assessment is to ensure that the proposed filter is not affected by the inverted ECG spike.

In order to perform this analysis, phantom signals of all combinations of these three varying factors were analyzed. Each combination was repeated three times for statistical significance, and an Analysis of Variance (ANOVA) was performed on the result to analyze the effect of the factors on the Local Improvement and Global Improvement in RMS (within ECG windows). The global error (RMS outside of ECG windows) was also analyzed. The raw data can be found in the Appendix for reference. Table 5 shows the ANOVA test on the Local Improvement effect. Factors having a p-value of less than 0.05 means that there is significant interaction between the factor and the effect (Local Improvement). In this case, both ECG:EMG ratio and the heart rate significantly affect the Local Improvement. Comparing this to the global improvement, shown in Table 6, ECG: EMG ratio plays a significant role in both of the filter's effectiveness.

```
Analysis of Variance for Local Improvement, using Adjusted SS for Tests

Source                              DF    Seq SS    Adj SS   Adj MS       F      P
ECG:EMG Ratio                        4  20824.19  20824.19  5206.05  818.35  0.000
Heart Rate                           2     42.31     42.31    21.16    3.33  0.043
Inverted                             1      6.64      6.64     6.64    1.04  0.311
ECG:EMG Ratio*Heart Rate             8    142.35    142.35    17.79    2.80  0.011
ECG:EMG Ratio*Inverted               4     43.47     43.47    10.87    1.71  0.160
Heart Rate*Inverted                  2     24.66     24.66    12.33    1.94  0.153
ECG:EMG Ratio*Heart Rate*Inverted    8     76.79     76.79     9.60    1.51  0.173
Error                               60    381.70    381.70     6.36
Total                               89  21542.11
```

Table 5: ANOVA test on Local Improvement. P-value < 0.05 shows that the specific factor has significant effect on the Local Improvement. Factors ECG: EMG Ratio and the Heart Rate both significantly affect the Local Improvement. The interaction between these two factors also significantly affects the Local Improvement

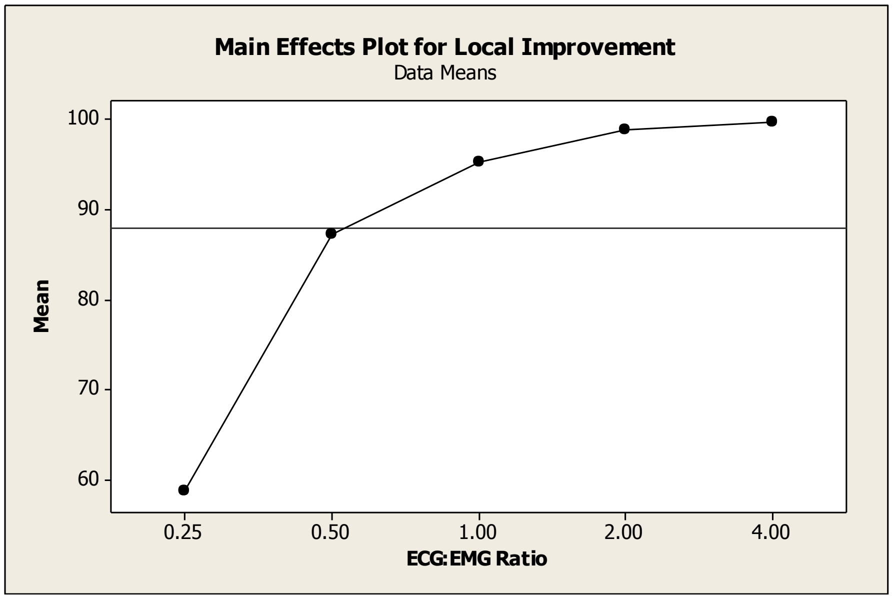
```
Analysis of Variance for Global Improvement, using Adjusted SS for Tests

Source                              DF  Seq SS  Adj SS  Adj MS       F      P
ECG:EMG Ratio                        4  506544  506544  126636  385.79  0.000
Heart Rate                           2    1810    1810     905    2.76  0.072
Inverted                             1      12      12      12    0.04  0.852
ECG:EMG Ratio*Heart Rate             8    6297    6297     787    2.40  0.026
ECG:EMG Ratio*Inverted               4     592     592     148    0.45  0.771
Heart Rate*Inverted                  2    1735    1735     868    2.64  0.079
ECG:EMG Ratio*Heart Rate*Inverted    8    4622    4622     578    1.76  0.103
Error                               60   19695   19695     328
Total                               89  541308
```

Table 6: ANOVA test on Global Improvement. P-value < 0.05 shows that the specific factor has significant effect on the Local Improvement. Factors ECG: EMG Ratio, and the interaction between ECG:EMG and heart rate significantly affects the Global Improvement

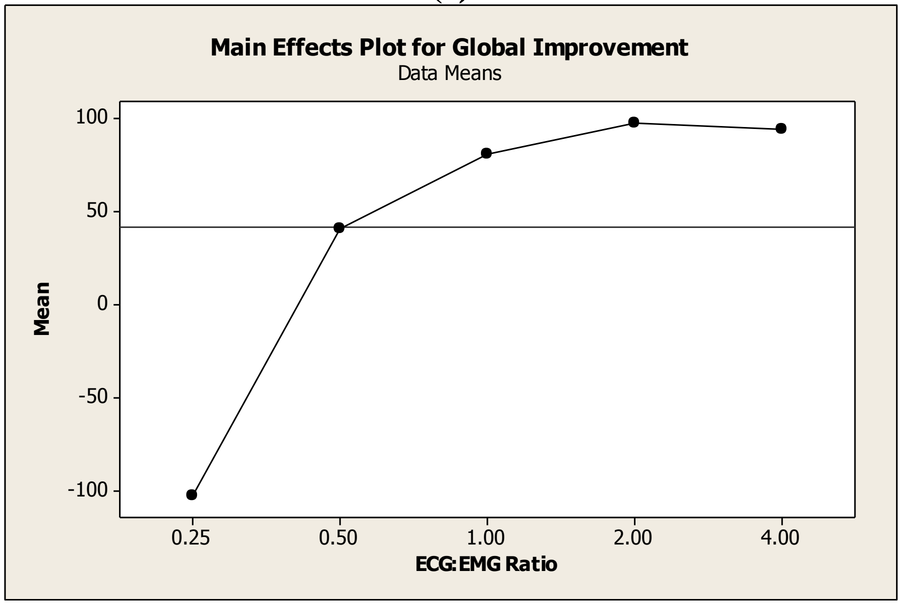
In order to assess how the ECG: EMG ratio plays a significant role in determining the effectiveness of the filter; its particular effect was analyzed. Figure 22 (a) shows the main effect plot for ECG: EMG ratio on the proposed local filter. The pattern of effectiveness makes complete sense, in where the filter struggles against the low ECG: EMG ratio compared to the high ECG: EMG ratio. This makes sense, because it is a lot harder to effectively locate the QRS complex of ECG spikes that are well hidden in the EMG signal. Remember, at these low ratio spikes, the ECG is completely hidden in the event-area of the signal, and is even partially hidden in the EMG band. It is important to note that the ECG detection algorithm was still able to determine that there is ECG present at such low ratio, and was able to effectively locate them for proper alignment. It is a bit interesting to note the drastic drop of effectiveness from 0.5 to 0.25 ratio of ECG: EMG. This, once again, is attributed to the fact that the EMG baseband is at 0.25 ratio as well. So, at 0.5 ratio, the ECG is not hidden in the vast area of event-free EMG base band, and is effectively removed by the filter. At 0.25 ratio, however, effectively removing these ECG spikes is a bit harder.

Comparing this data set to the Global filter's effectiveness shows the true robustness of the local filter. Although both filter share the same pattern of robustness, the Improvement % from the local filter is far superior to the Improvement % from the global filter at every ECG: EMG ratio. There is even a more drastic fall in improvement % of the global filter compared to the local filter. Specifically, at 0.50 ratio, the local filter still improves the signal by 88%, while the global filter improves the signal by only 48% (almost half the improvement %!). The most essential note of comparison is that at no ECG:EMG ratio, does the local filter ever degrade the signal. Even, at the low ratio of 0.25, the local filter still improves the signal by 60% by removing the remaining small spikes in the signal. The global filter, on the other hand, actually harms the signal. At 0.25 ratio, the global filter has a -100% improvement in the RMS of the signal. This is primarily caused by the fact that the global filter has no means of identifying the location of the ECG spikes, unlike the local filter. The global filter may filter the EMG spikes by accident, and hence deteriorate the signal. As previously explained, unlike the homogeneity, the RMS is fairly insensitive to small changes. Hence, a -100% improvement means that the signal is completely destroyed by the signal. It would be better to keep the signal non-filtered than to filter the signal with a global filter at this ratio. Hence, the effectiveness and robustness of the proposed local filter is no match for the standard global filter.

An interesting, final note from the global filter's effectiveness is the cusp shown in Figure 22 (b). This means that the global filter has a very narrow range of ECG: EMG ratio that it can filter at its highest efficiency. Unlike the local filter, which keeps performing better at higher ratios, the global filter actually starts to perform worse at ECG: EMG ratio of 4 compared to 2. This has been visually supported, and occurs because the global filter starts to over-filter or under-filter the signal at high amplitude ECG spikes. This effect can be further observed by examining the effect of ECG: EMG ratio on the addition of RMS Error (outside of ECG windows) from the Global filter, as shown in Figure 23. In this figure, it is clearly evident, that the global filter introduces the least error in ECG: EMG ratio of 2 and this error actually increases at higher ratios. Local filter, on the other hand, has a more controlled process in eliminating the ECG spikes, due to targeting the QRS complex, and therefore does not encounter this kind of problem. Specifically, the local filter only targets the QRS complex, and therefore has better information on the remaining ECG during its iterative process in removing the ECG wavelet components. Therefore, global filter has a far narrow range of ECG: EMG ratios it can effectively handle compared to the local filter. Hence, the local filter is far more robust than the global filter.



(a)



(b)

Figure 22 (a) The effect of ECG: EMG ratio on the local filter's RMS improvement (%). (b) The effect of ECG: EMG ratio on the global filter's RMS improvement (%). It is clearly evident that although both of the filters have similar reaction to the changes in the Ratio, the local filter handles the ratios far better than the global filter.

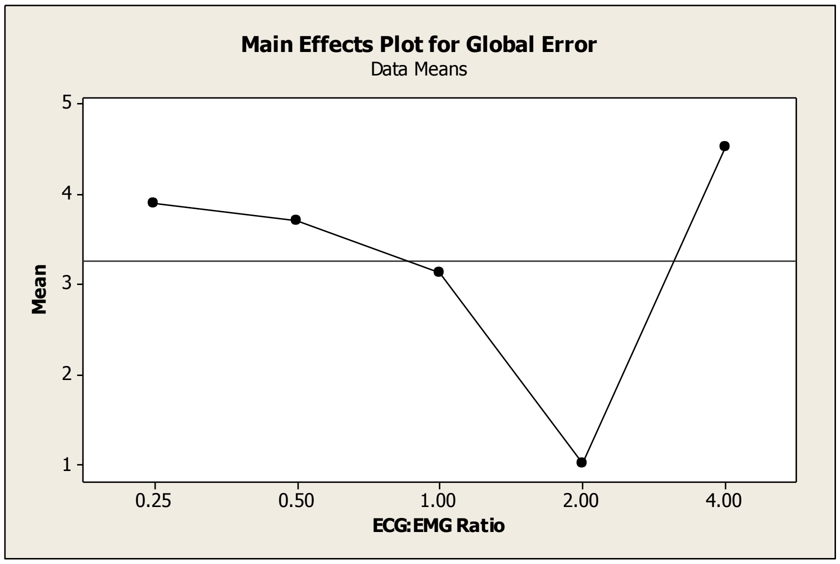


Figure 23: The effect of ECG: EMG ratio on the introduction of error (outside of ECG windows) from the Global Filter

The ANOVA test on Table 5 also showed that the interaction between ECG:EMG ratio and the heart rate also significantly impacted the effectiveness of the local filter. This is a bit surprising, because the heart rate should not influence the effectiveness, since each ECG spike has its own window in the local filter. In order to find out how heart rate Therefore, the interaction between the two factors were plotted in Figure 24. The interaction plot clearly shows that there is in-fact no connection between the effectiveness of the local filter and the interaction between the heart rate and the ECG: EMG ratio. The only ratio, where heart rate seems to play a significant role is the 0.25 ratio. Since, there is no linear trend in the effect of heart rate on the effectiveness at this low ratio, this change can be attributed to the inaccuracy in finding the QRS complex in the ECG window at this low of ratio. The higher the heart rate, the more ECG windows present, and the higher the likelihood that there is an inaccurate ECG removal. Hence, there is discrepancy in the efficiency of the local filter at this low ratio.

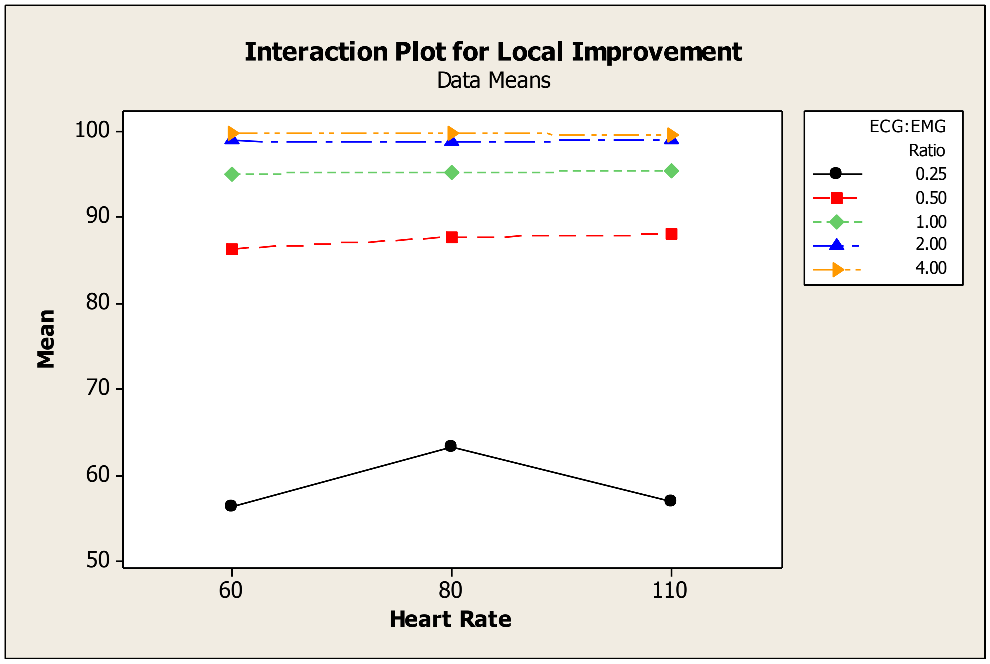


Figure 24: Interaction plot of the effect from the heart rate and the ECG: Ratio on the local filter. There is only significant interaction at ECG:EMG ratio of 0.25, therefore the interaction of the two factors overall do not actually influence the effectiveness of the filter together.

3.6 In-vivo Results

To test the efficiency of the algorithms, the algorithms needed to be tested on real data. One of the key components of the algorithm is its robustness in handling as many types of signals as possible. Hence, data set collected by Frazier Rehab Institute included EMG signals from the different muscle groups in the torso. Additionally, the patient was required to perform different tasks during the collection procedure, which caused the EMG pattern to differentiate based on the specific muscle groups used. Different tasks included standing, sitting, coughing, breathing heavily, etc. The main goal was to obtain as vastly different types of EMG signals as possible to ensure that the algorithm is robust enough to handle the varying different kinds of clinical assessed EMGs. The signals with ECG detected were effectively filtered by both the local and the global filter. An electro-physiologist expert visually looked at the overlay of the results from both of the filters, and deemed the results produced by the local filter as an acceptable form of the filtered signal. He deemed the results from the global filter as an unacceptable filtering process, since it introduced vast amount of error from the filtering process.

Analytically observing the effects of these filters allow us to observe the effect of both filtering processes on the signal through the validation metrics. Table 7 shows the raw data analysis from the validation metrics in many different kinds of EMG data sets. It is critical to note that these validation metrics are the raw analysis, and not the error analysis shown in the phantom signals. Error analysis is not feasible, due to the lack of a clean original signal, which can be compared with the filtered signal. Statistical analysis can also not be done on this data, since each EMG signal data set has a different ECG height, and profile. Hence, the RMS of the signal before and after removal is affected by the signal itself, and not the removal process.

Generally, the height and the profile of the ECG are accounted by comparing it to the original clean signal in the phantom analysis. This table however does show the negative effects of the global filtering process. As, previously shown in the phantom analysis, the outer areas (areas outside of ECG windows) are typically unaffected by the local filter. Therefore, the validation metrics from the local filter in outer areas of ECG windows is similar to the original clean signal. Comparing these values to the outer values shown by the global filter shows an extremely large magnitude of error introduced by the global filter in areas outside of the ECG windows. Hence, global filter deteriorates the signal, and essentially makes it unusable for further assessment and analysis.

			Local Filter				Global Filter			
ID	**Subject**	**Sig. #**	**RMS Inner**	**RMS Outer**	**VAR Inner**	**VAR Outer**	**RMS Inner**	**RMS Outer**	**VAR Inner**	**VAR Outer**
1	R032L_Supine MEP5sec	15	**69.386**	**0**	**60.788**	**0**	**83.148**	**123.40**	**71.988**	**74.726**
2	R032L_Supine MEP5sec	16	**58.300**	**0.0481**	**38.880**	**0.0783**	**58.468**	**29.784**	**16.967**	**41.838**
3	R032L_Supine MEP5sec	6	**65.267**	**0**	**61.425**	**0**	**45.208**	**5.5623**	**16.465**	**41.801**
4	R032L_Supine MEP5sec	5	**35.761**	**0.9405**	**9.6682**	**1.2996**	**56.044**	**68.969**	**44.780**	**77.687**
5	R068H_Supin eMIP5sec	16	**52.070**	**0.0152**	**17.098**	**0.0153**	**56.951**	**13.077**	**21.366**	**11.123**
6	R068H_Supin eMIP5sec	15	**60.933**	**0.2170**	**43.647**	**0.2462**	**73.782**	**50.884**	**57.033**	**46.044**
7	R058N_Supin eSpirometry	16	**28.543**	**1.7029**	**21.202**	**1.9926**	**59.359**	**93.581**	**68.489**	**228.66**
8	R068H_Supin eMIP5sec	6	**52.684**	**0.0721**	**35.557**	**0.0595**	**63.331**	**34.155**	**45.449**	**26.699**
9	R061H_Supin eMIP5sec	16	**76.104**	**0**	**77.494**	**0**	**91.667**	**381.91**	**90.180**	**359.91**
10	R061H_Supin eMEP	16	**74.894**	**0**	**75.543**	**0**	**90.004**	**348.03**	**88.028**	**314.40**
11	R058N_Supin eMEP	16	**74.695**	**0.0210**	**72.488**	**0.0286**	**82.396**	**101.87**	**80.374**	**120.54**
12	R058N_Supin eMEP	15	**46.679**	**0.0594**	**28.232**	**0.1069**	**49.351**	**14.505**	**38.054**	**41.592**

13	R058N_Supin eMEP	5	**55.332**	**0.0887**	**30.654**	**0.1224**	**66.127**	**47.390**	**41.431**	**37.995**
14	R058N_Supin eMEP	6	**31.714**	**0.5464**	**27.944**	**0.0373**	**61.991**	**108.23**	**80.552**	**438.14**
15	R032L_Supine Spirometry	6	**59.674**	**0.2248**	**47.498**	**0.3091**	**78.899**	**116.19**	**69.217**	**118.03**
16	R032L_Supine Spirometry	15	**67.270**	**0.0693**	**59.594**	**0.0818**	**84.185**	**155.20**	**76.021**	**115.43**
17	R032L_Supine Spirometry	16	**55.647**	**0**	**51.396**	**0**	**53.470**	**22.728**	**38.619**	**21.648**
18	R058N_Sitting Spirometry	5	**66.015**	**0.0537**	**57.797**	**0.0721**	**73.981**	**54.907**	**73.888**	**132.62**
19	R058N_Sitting Spirometry	15	**64.679**	**0.1241**	**54.534**	**0.1584**	**77.067**	**55.523**	**67.324**	**61.461**
20	R058N_Sitting Spirometry	16	**80.688**	**0.1098**	**83.088**	**0.1647**	**88.911**	**168.15**	**93.497**	**305.00**

Table 7 Shows the raw in-vivo validation metric data analysis of different EMG signals. These validation metric analyses are not error analyses, since the original clean signal is not present. The local filter generally does a very good job at removing ECG spikes in all the different kind of EMG signal, while introducing minimal error in areas outside of ECG spikes. The global filter introduces large magnitude of error in areas outside of ECG windows, hence deteriorating the signal, and making it unusable for assessment.

IV Conclusion and Future Work

EMG signals from the trunk muscles provide valuable clinical information that due to ECG corruption from the heart cannot be accurately assessed. Current filtering approaches do not provide adequate ECG removal, and do not effectively remove all of the ECG corruption from EMG signals. The proposed framework has been shown to significantly reduce and eliminate ECG corruption within EMG signals. This approach is a post-digital filter, which allows the filtering of previously collected signals with ECG corruption already present. Additionally, the proposed localized filtering approach has the benefit of not introducing significant error into the native EMG signal during the filtering process, and performing more robust at different types of EMG signals unlike comparable global filtering techniques. The local filter also has an advantage that it never deteriorates or degrades the signal, unlike the comparable global filter, making it preferable for clinical applications.

Future design work includes trying to improve the efficiency of the localized filter by improving the ability to locate the QRS complex at extremely low amplitude ECG (rare). This should improve the overall accuracy of the filter, while eliminating its main weakness. Other future analysis work includes making direct comparisons with the currently used solutions, such as the Butterworth filter, discrete wavelet transform techniques (Debauchies), and other advanced ECG removal filter gating techniques. Other validation metrics that can be used are the profiles of Power Spectrum Density (PSD), custom signal strength, etc. Although, most of these previously used validation metrics hide the negative effects of global filter, by assessing the shape of the signal and not the direct contents within the signal.

REFERENCES

Bartolo, A., C. Roberts, et al. (1996). "Analysis of diaphragm EMG signals: comparison of gating vs. subtraction for removal of ECG contamination." Journal of Applied Physiology **80**(6): 1898-1902.

Bloch, R. (1983). "Subtraction of electrocardiographic signal from respiratory electromyogram." Journal of Applied Physiology **55**(2): 619-623.

Carre, P., H. Leman, et al. (1998). "Denoising of the uterine EHG by an undecimated wavelet transform." Biomedical Engineering, IEEE Transactions on **45**(9): 1104-1113.

Chun-Lin, L. (2010). A Tutorial of the Wavelet Transform, NTUEE, Taiwan.

Clancy, E. A., S. Bouchard, et al. (2001). "Estimation and application of EMG amplitude during dynamic contractions." Engineering in Medicine and Biology Magazine, IEEE **20**(6): 47-54.

Costa Junior, J. D., D. D. Ferreira, et al. (2010). Reducing electrocardiographic artifacts from electromyogram signals with independent component analysis. Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE.

DeVivo, M. J. (2010). "Epidemiology of spinal cord injury." Spinal Cord Medicine Principles and Practice. Demos Medical Publishing: New York, NY: 78-84.

Drake, J. D. M. and J. P. Callaghan (2006). "Elimination of electrocardiogram contamination from electromyogram signals: An evaluation of currently used removal techniques." Journal of Electromyography and Kinesiology **16**(2): 175-187.

Hof, A. L. (2009). "A simple method to remove ECG artifacts from trunk muscle EMG signals." Journal of Electromyography and Kinesiology **19**(6): e554-e555.

Hyvärinen, A., J. Karhunen, et al. (2001). Independent component analysis, Wiley-interscience.

Karthik, R. (2003). ECG simulation using MATLAB.

Liu, X., S. Wang, et al. (2008). "The sensory and motor representation of synchronized oscillations in the globus pallidus in patients with primary dystonia." Brain **131**(6): 1562-1573.

Lu, G., J.-S. Brittain, et al. (2009). "Removing ECG noise from surface EMG signals using adaptive filtering." Neuroscience Letters **462**(1): 14-19.

Mallat, S. G. (1989). "A theory for multiresolution signal decomposition: the wavelet representation." Pattern Analysis and Machine Intelligence, IEEE Transactions on **11**(7): 674-693.

Marque, C., C. Bisch, et al. (2005). "Adaptive filtering for ECG rejection from surface EMG recordings." Journal of Electromyography and Kinesiology **15**(3): 310-315.

McGill, K. C., Z. C. Lateva, et al. (2005). "EMGLAB: An interactive EMG decomposition program." Journal of Neuroscience Methods **149**(2): 121-133.

Redfern, M. S., R. E. Hughes, et al. (1993). "High-pass filtering to remove electrocardiographic interference from torso EMG recordings." Clinical Biomechanics **8**(1): 44-48.

Rodríguez-Carreño, I., A. Malanda-Trigueros, et al. (2006). "Filter design for cancellation of baseline-fluctuation in needle EMG recordings." Computer Methods and Programs in Biomedicine **81**(1): 79-93.

Schweitzer, T. W., J. W. Fitzgerald, et al. (1979). "Spectral analysis of human inspiratory diaphragmatic electromyograms." Journal of Applied Physiology **46**(1): 152-165.

Taelman, J., S. Van Huffel, et al. (2007). Wavelet-Independent Component Analysis to remove Electrocardiography Contamination in surface Electromyography. Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE.

Thomas, C. K., E. Y. Zaidner, et al. (1997). "Muscle Weakness, Paralysis, and Atrophy after Human Cervical Spinal Cord Injury." Experimental Neurology **148**(2): 414-423.

Torrence, C. and G. P. Compo (1998). "A Practical Guide to Wavelet Analysis." Bulletin of the American Meteorological Society **79**(1): 61-78.

van Boxtel, A. (2001). "Optimal signal bandwidth for the recording of surface EMG activity of facial, jaw, oral, and neck muscles." Psychophysiology **38**(1): 22-34.

von Tscharner, V. (2000). "Intensity analysis in time-frequency space of surface myoelectric signals by wavelets of specified resolution." Journal of Electromyography and Kinesiology **10**(6): 433-445.

von Tscharner, V., B. Eskofier, et al. (2011). "Removal of the electrocardiogram signal from surface EMG recordings using non-linearly scaled wavelets." Journal of Electromyography and Kinesiology **21**(4): 683-688.

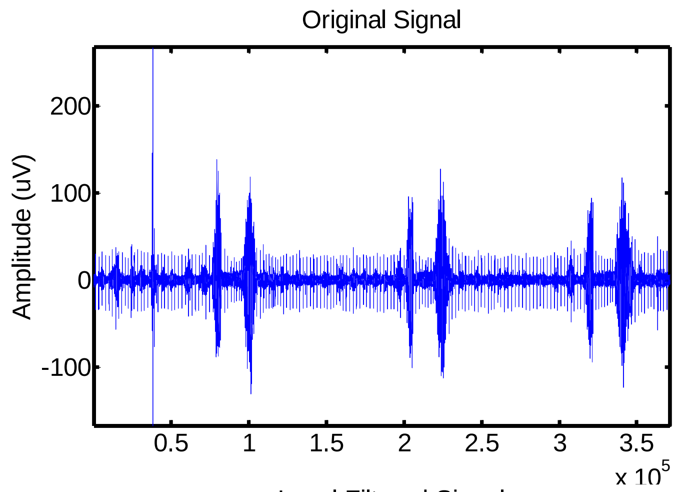
Weidong, Z. and J. Gotman (2004). Removal of EMG and ECG artifacts from EEG based on wavelet transform and ICA. Engineering in Medicine and Biology Society, 2004. IEMBS '04. 26th Annual International Conference of the IEEE.

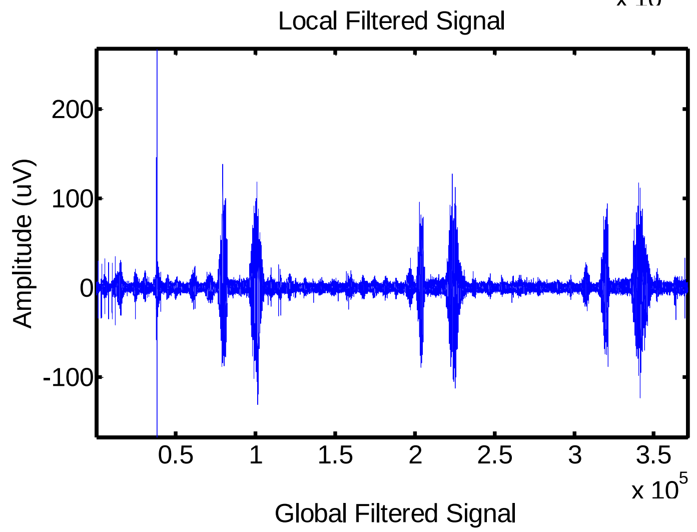
Yuancheng, D., W. Wolf, et al. (2000). "New aspects to event-synchronous cancellation of ECG interference: an application of the method in diaphragmatic EMG signals." Biomedical Engineering, IEEE Transactions on **47**(9): 1177-1184.

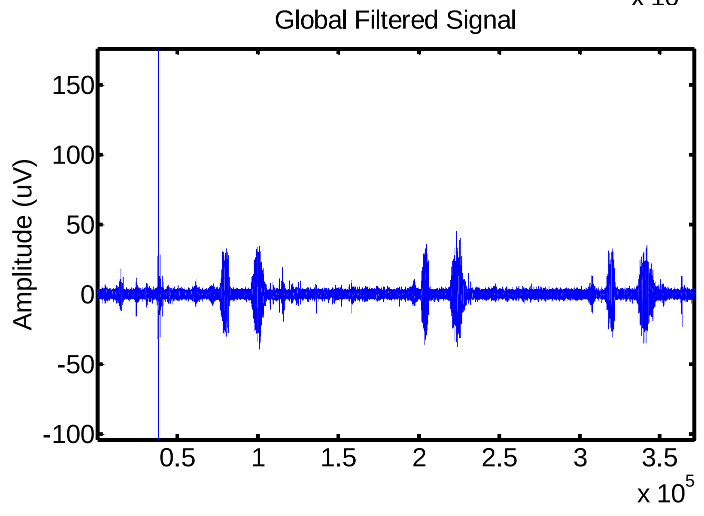
Zschorlich, V. R. (1989). "Digital filtering of EMG-signals." Electromyogr Clin Neurophysiol **29**(2): 81-86.

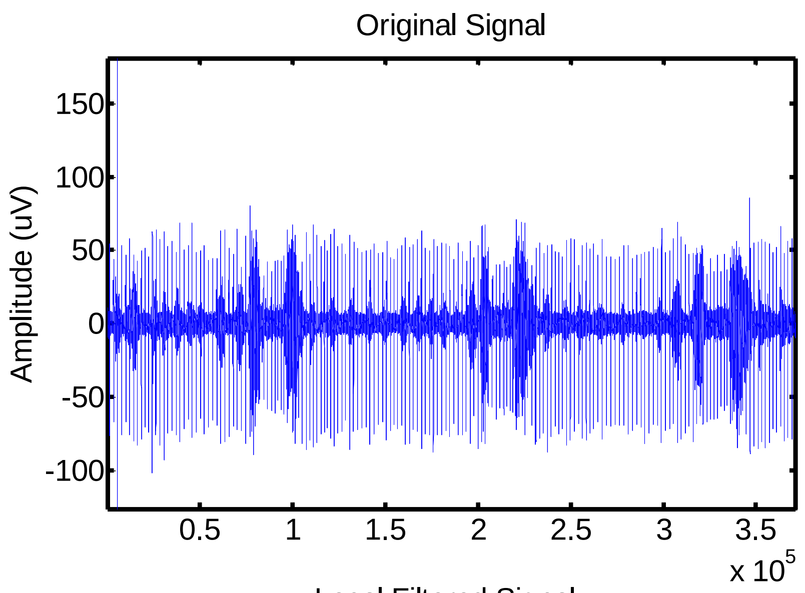
APPENDIX I

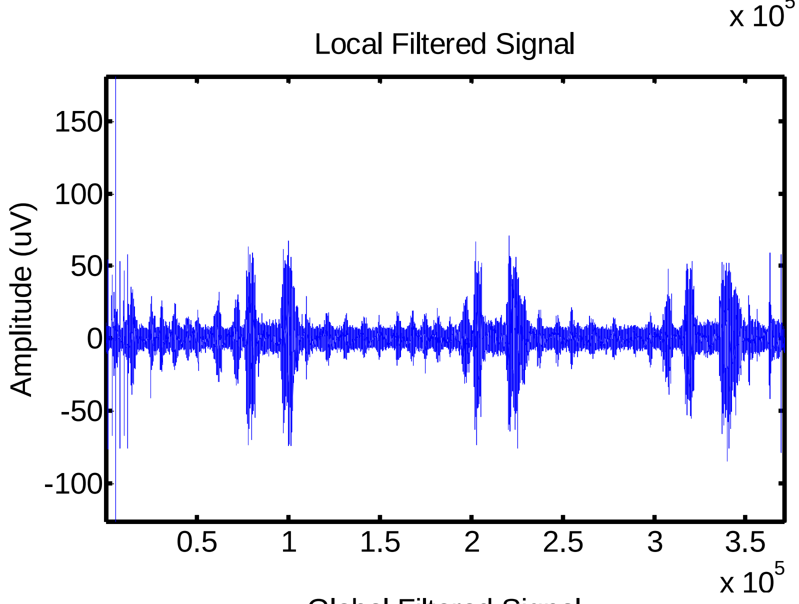
Visual Results from Real In-Vivo Data.

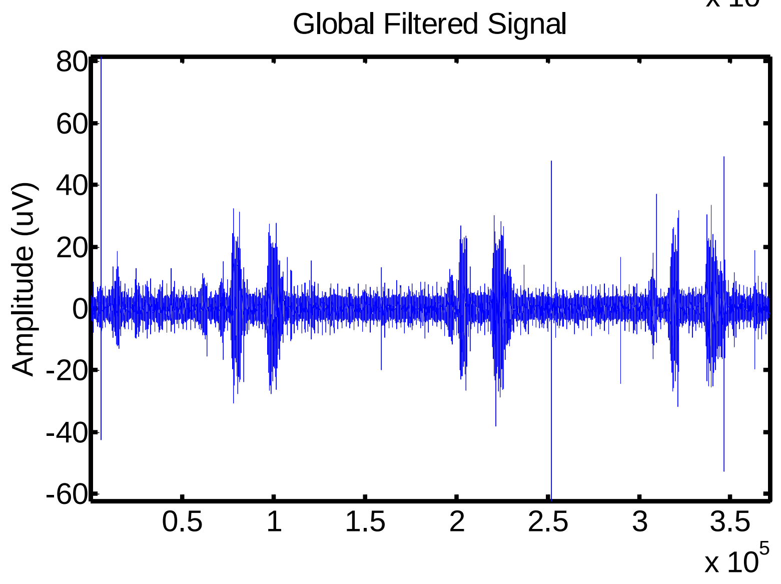


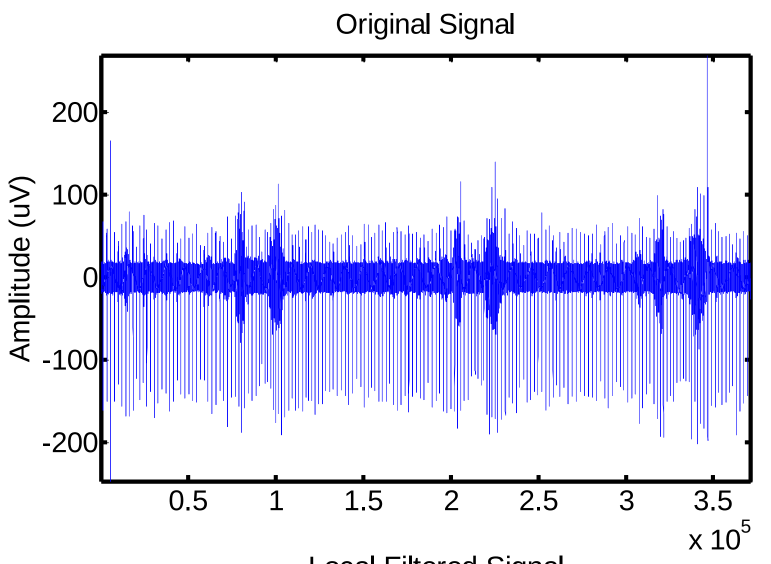


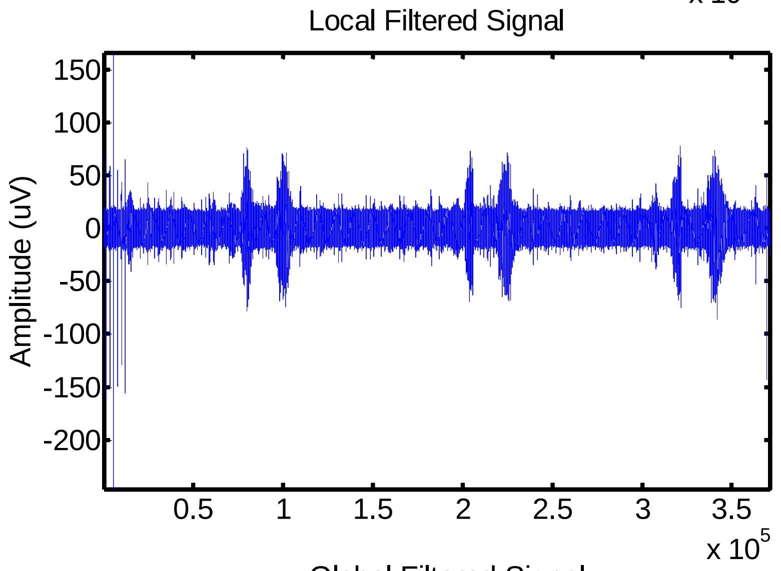


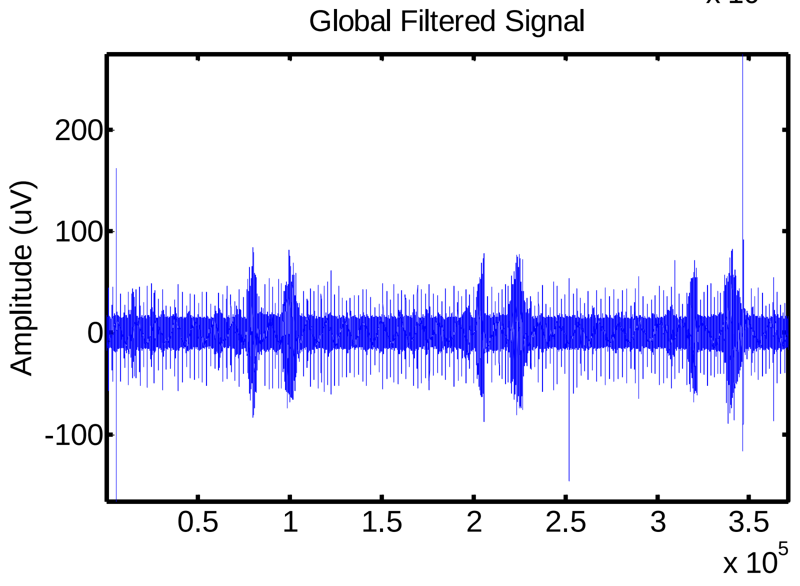

Original Signal
Amplitude (uV)
150
100
50
0
-50
-100
0.5
1
1.5
2
2.5
3
3.5
x 10^5

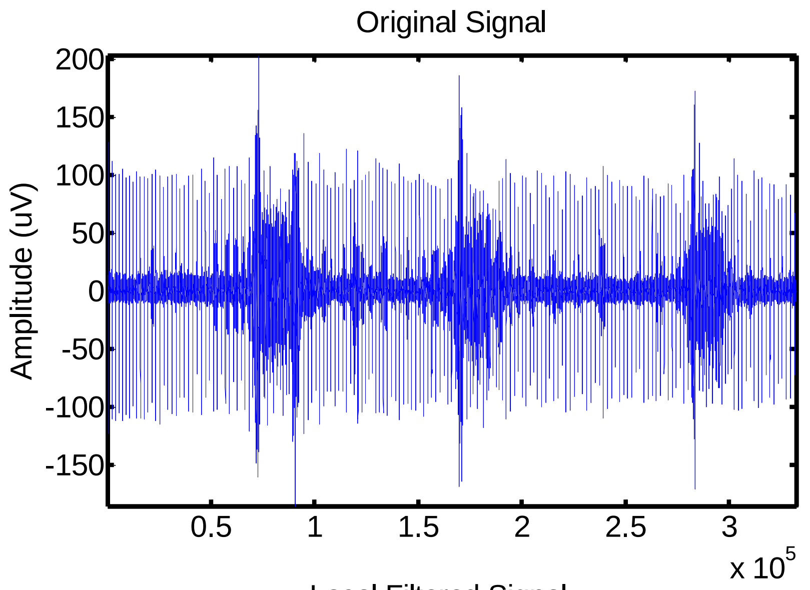

Local Filtered Signal
Amplitude (uV)
150
100
50
0
-50
-100
0.5
1
1.5
2
2.5
3
3.5
x 10^5

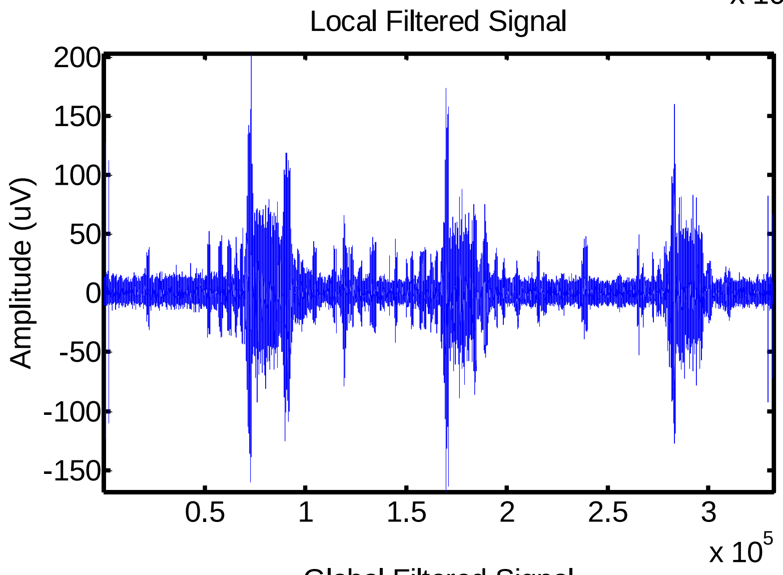

Global Filtered Signal
Amplitude (uV)
80
60
40
20
0
-20
-40
-60
0.5
1
1.5
2
2.5
3
3.5
x 10^5

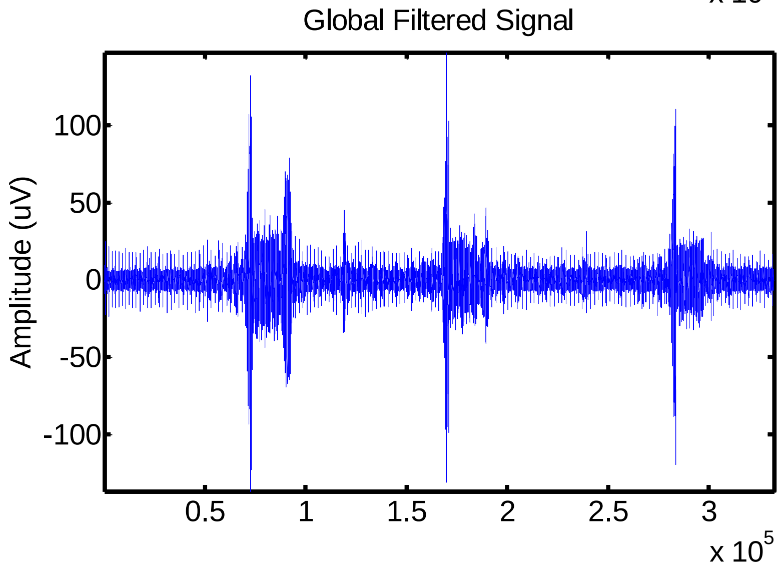

Original Signal
Amplitude (uV)
200
100
0
-100
-200
0.5
1
1.5
2
2.5
3
3.5
x 10^5

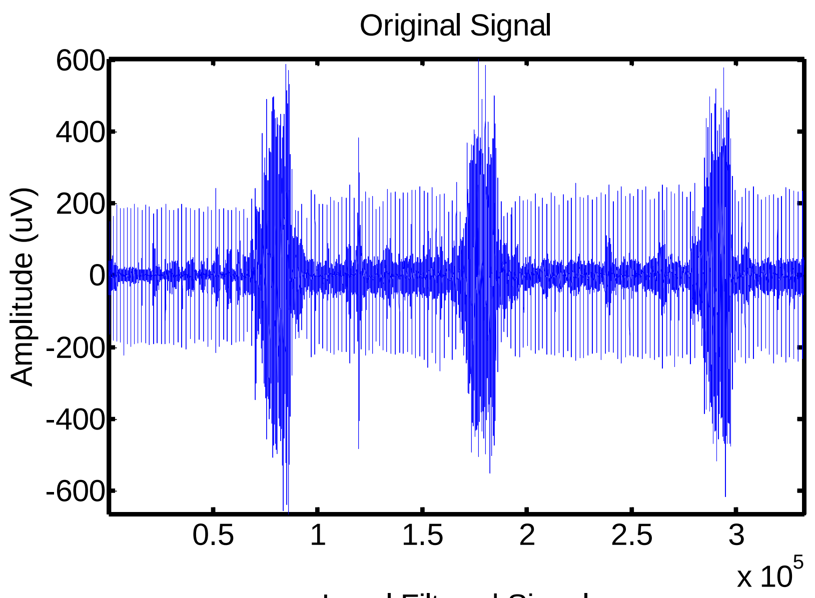

Local Filtered Signal
Amplitude (uV)
150
100
50
0
-50
-100
-150
-200
0.5
1
1.5
2
2.5
3
3.5
x 10^5

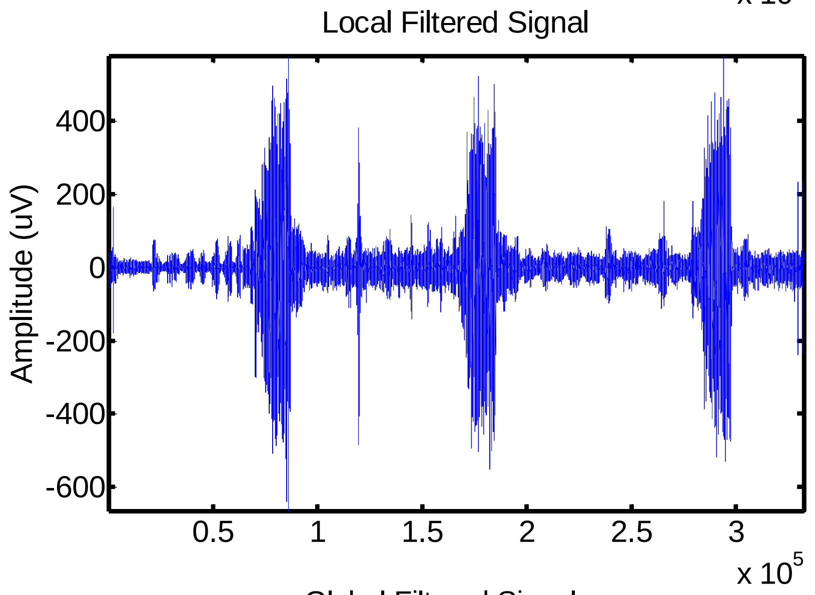

Global Filtered Signal
Amplitude (uV)
200
100
0
-100
0.5
1
1.5
2
2.5
3
3.5
x 10^5

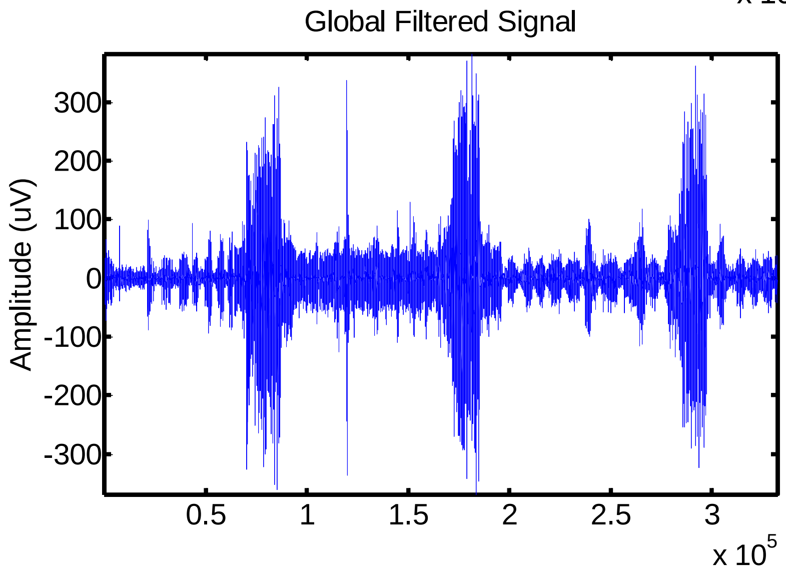

Original Signal
Amplitude (uV)
200
150
100
50
0
-50
-100
-150
0.5
1
1.5
2
2.5
3
x 10^5

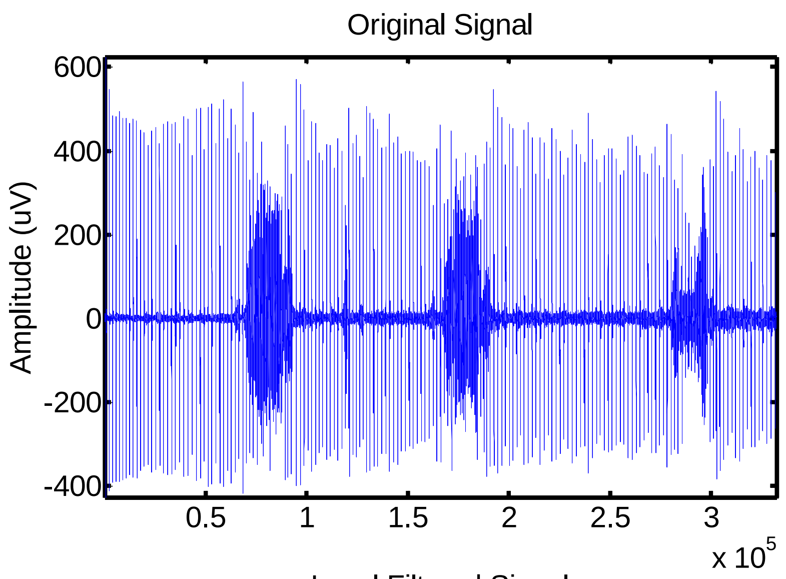

Local Filtered Signal
Amplitude (uV)
200
150
100
50
0
-50
-100
-150
0.5
1
1.5
2
2.5
3
x 10^5

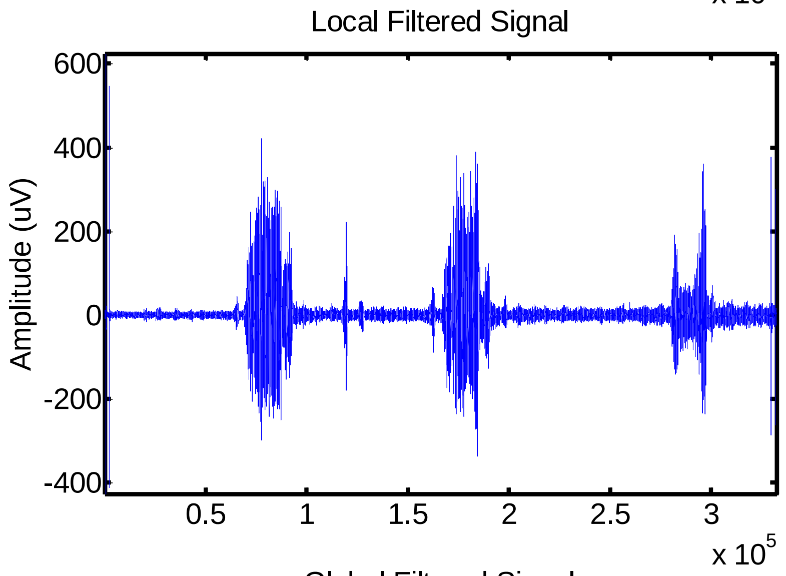

Global Filtered Signal
Amplitude (uV)
100
50
0
-50
-100
0.5
1
1.5
2
2.5
3
x 10^5

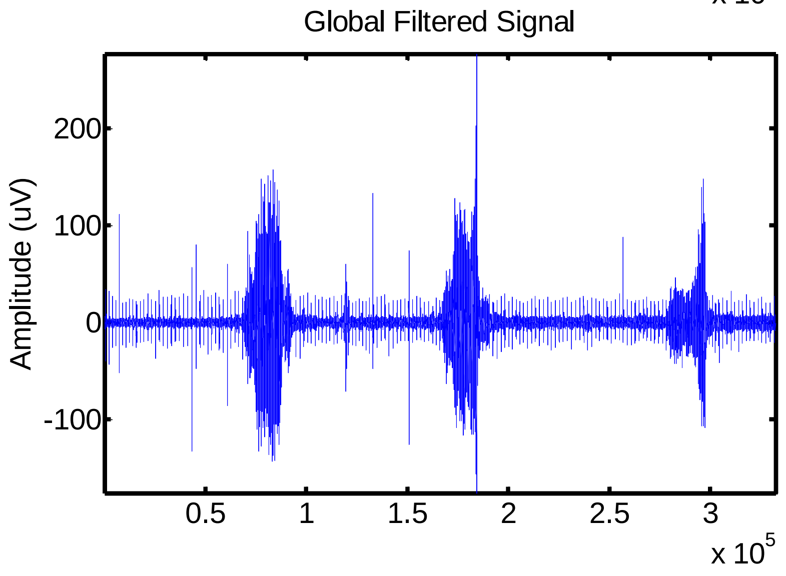

Original Signal
Amplitude (uV)
600
400
200
0
-200
-400
-600
0.5
1
1.5
2
2.5
3
x 10^5

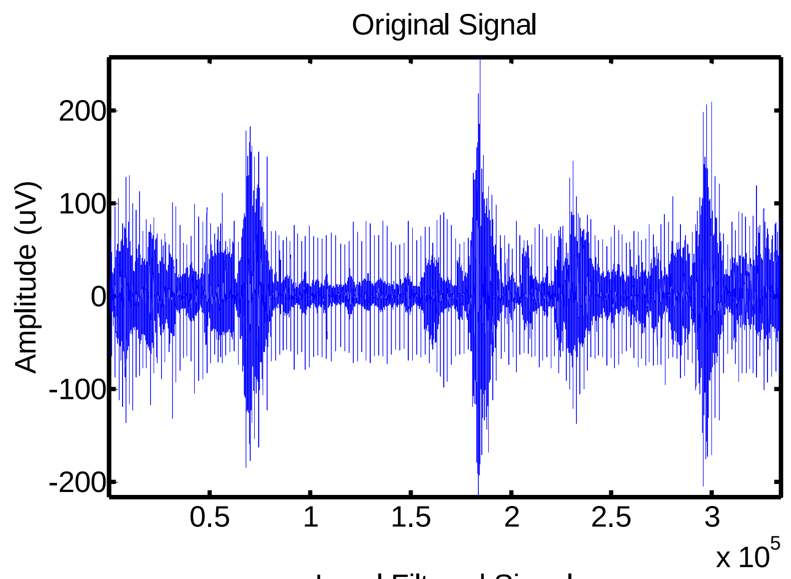

Local Filtered Signal
Amplitude (uV)
400
200
0
-200
-400
-600
0.5
1
1.5
2
2.5
3
x 10^5

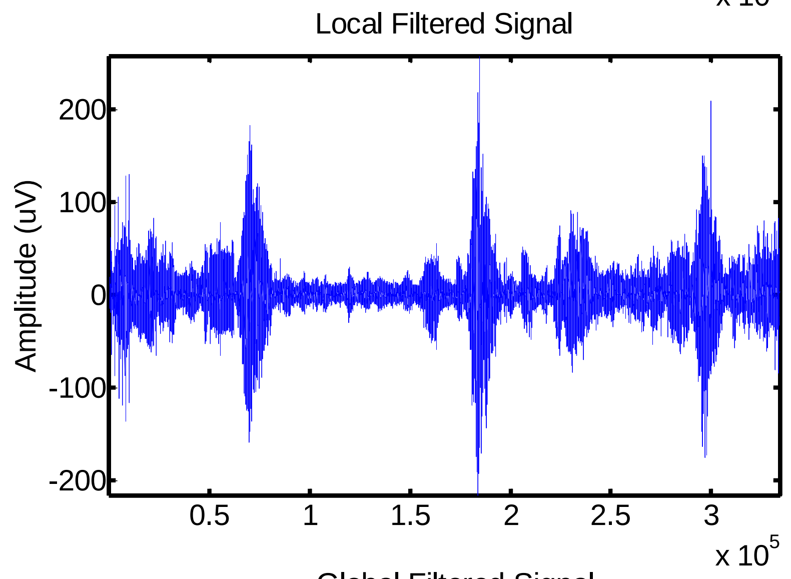

Global Filtered Signal
Amplitude (uV)
300
200
100
0
-100
-200
-300
0.5
1
1.5
2
2.5
3
x 10^5

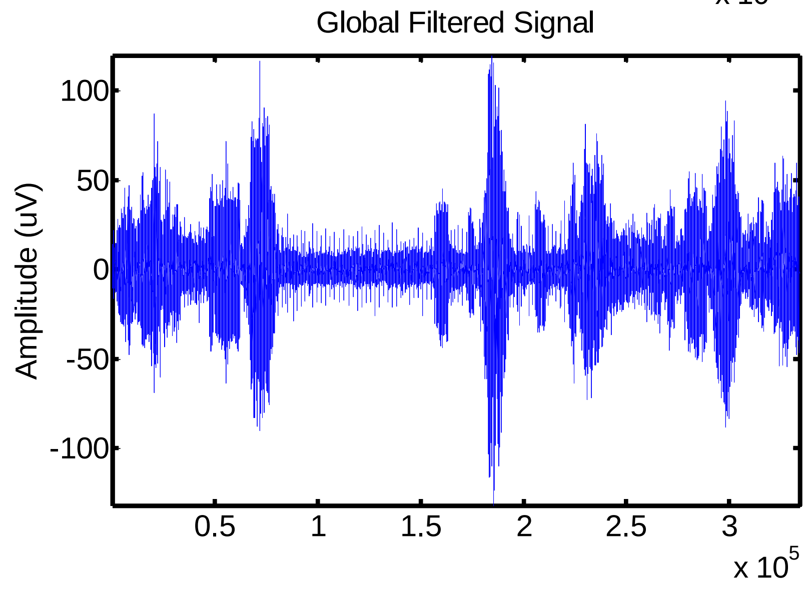

Original Signal
Amplitude (uV)
600
400
200
0
-200
-400
0.5
1
1.5
2
2.5
3
x 10^5

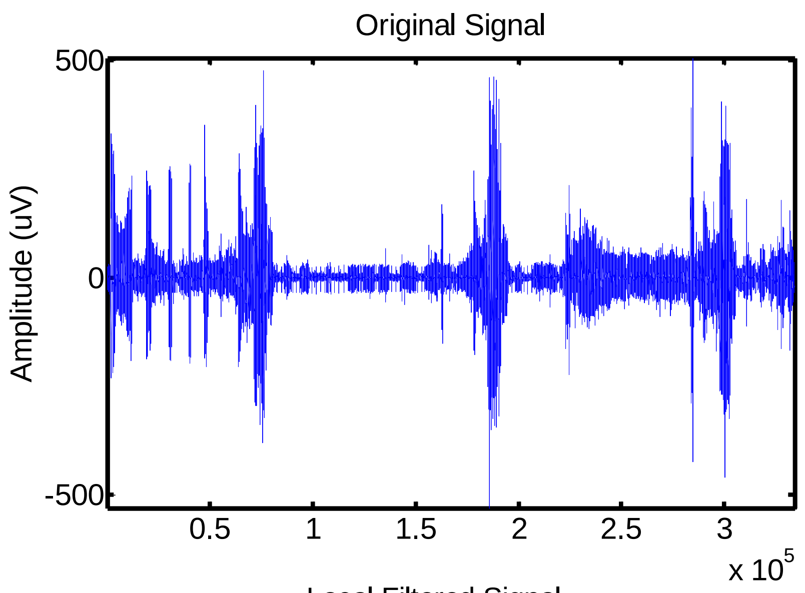

Local Filtered Signal
Amplitude (uV)
600
400
200
0
-200
-400
0.5
1
1.5
2
2.5
3
x 10^5

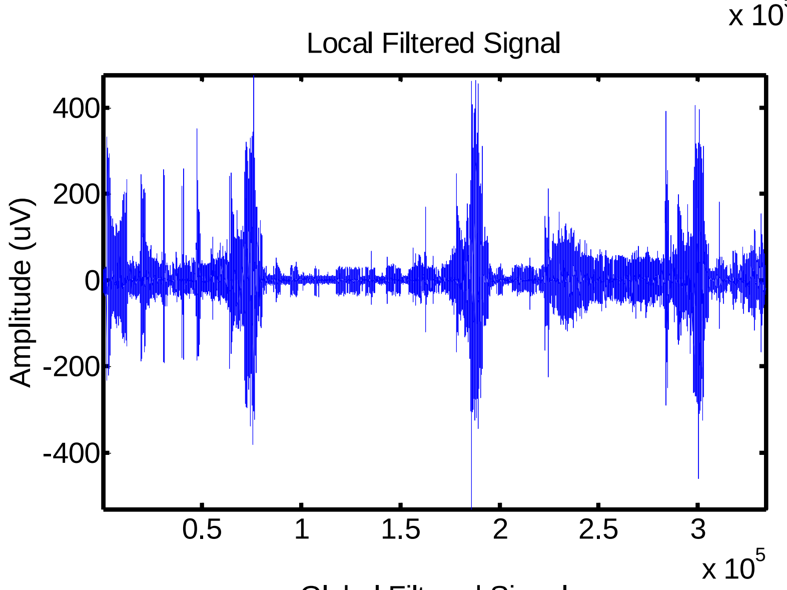

Global Filtered Signal
Amplitude (uV)
200
100
0
-100
0.5
1
1.5
2
2.5
3
x 10^5

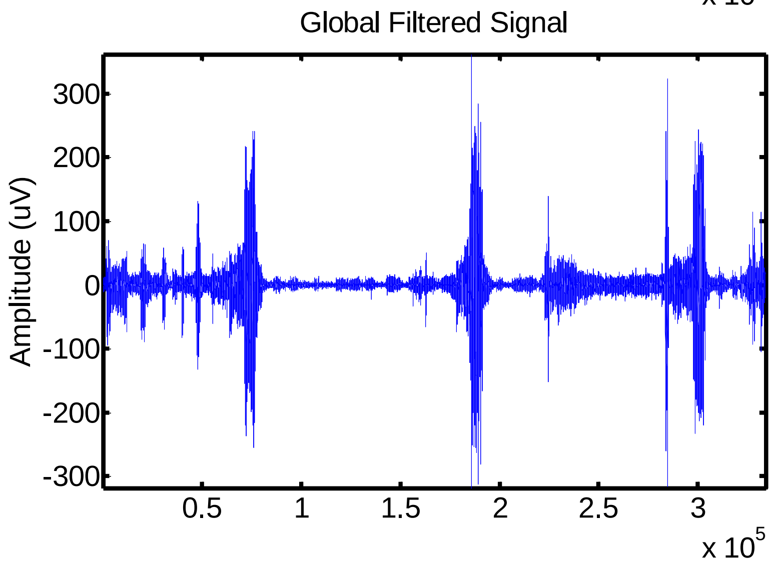

Original Signal
Amplitude (uV)
200
100
0
-100
-200
0.5
1
1.5
2
2.5
3
x 10^5

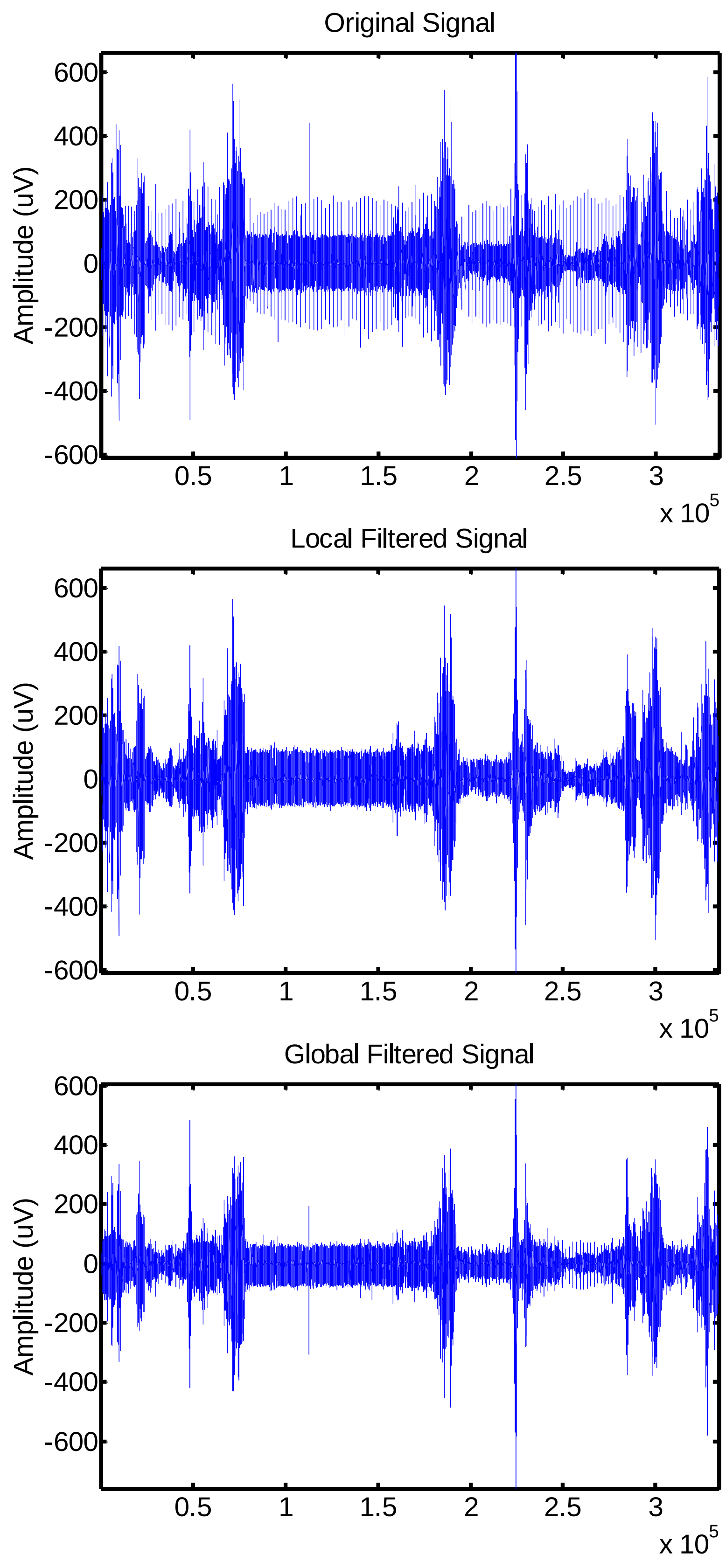

Local Filtered Signal
Amplitude (uV)
200
100
0
-100
-200
0.5
1
1.5
2
2.5
3
x 10^5

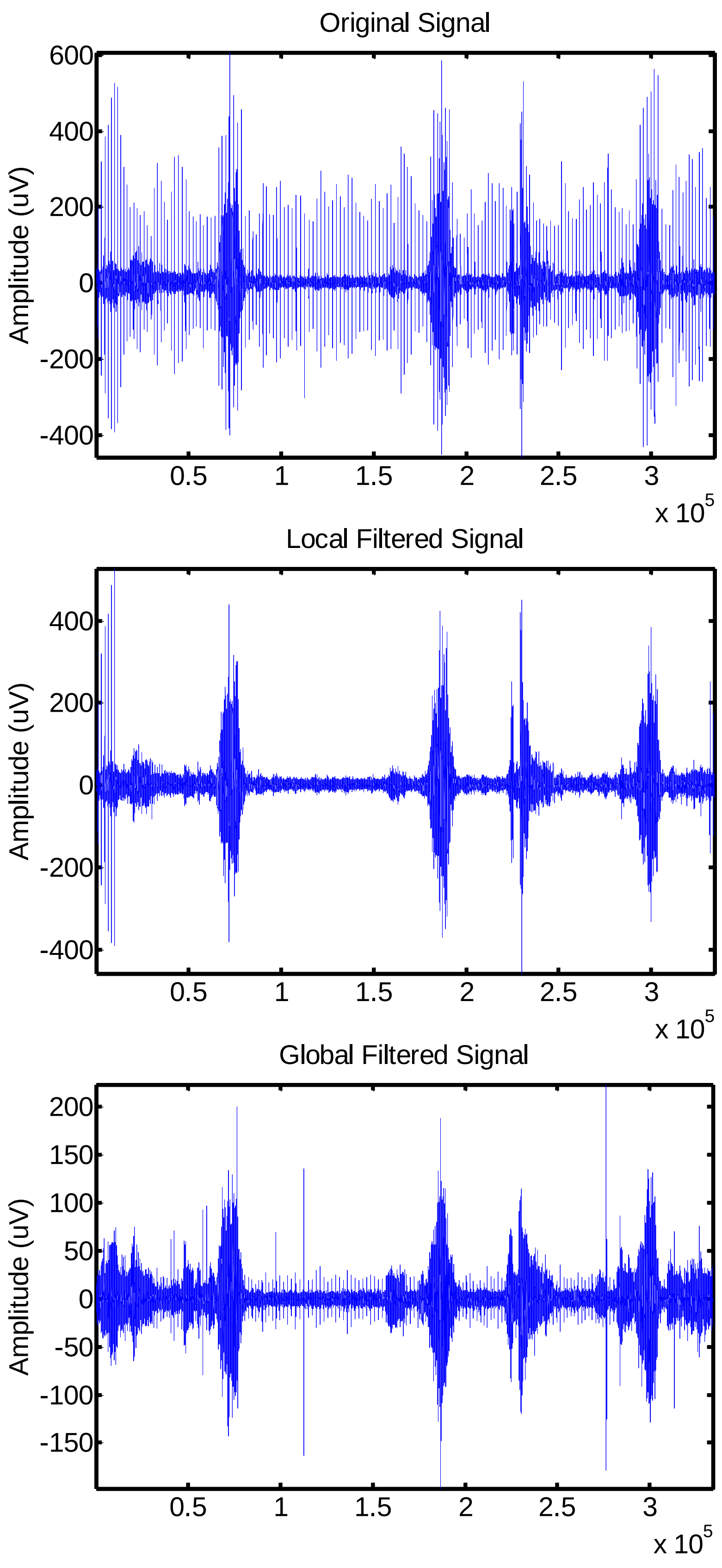

Global Filtered Signal
Amplitude (uV)
100
50
0
-50
-100
0.5
1
1.5
2
2.5
3
x 10^5

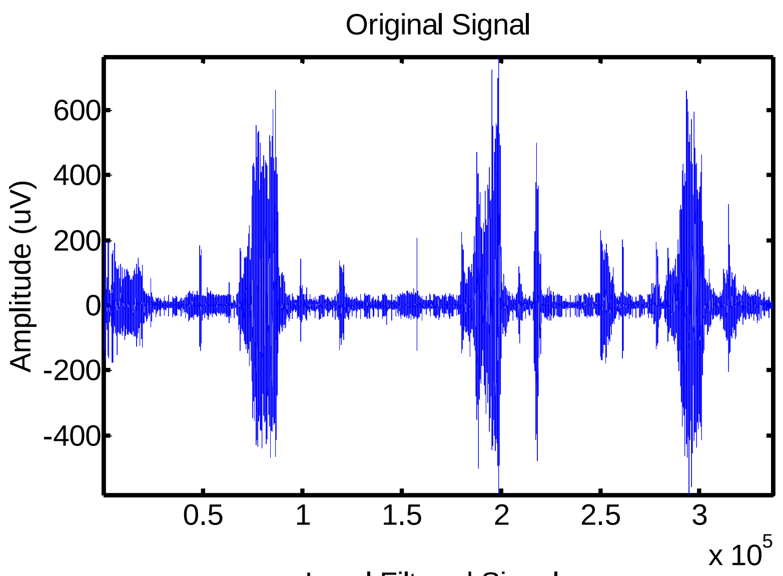

Original Signal
Amplitude (uV)
500
0
-500
0.5
1
1.5
2
2.5
3
x 10^5

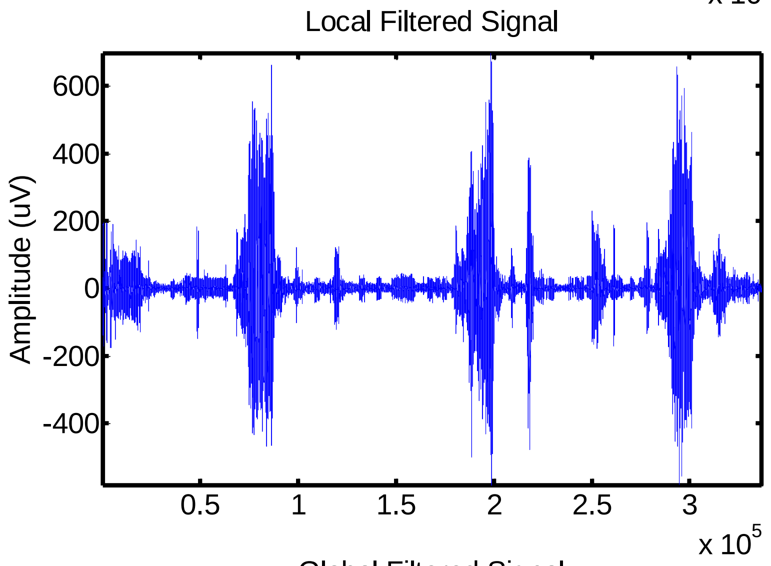

Local Filtered Signal
Amplitude (uV)
400
200
0
-200
-400
0.5
1
1.5
2
2.5
3
x 10^5

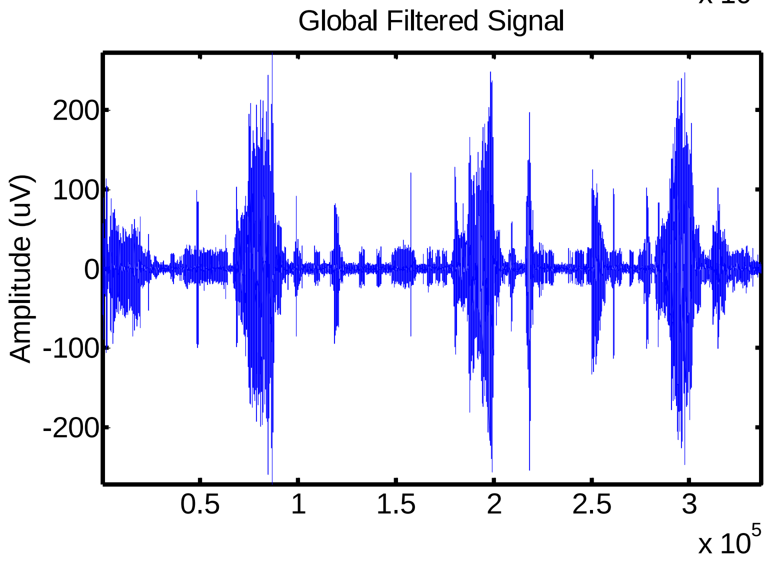

Global Filtered Signal
Amplitude (uV)
300
200
100
0
-100
-200
-300
0.5
1
1.5
2
2.5
3
x 10^5

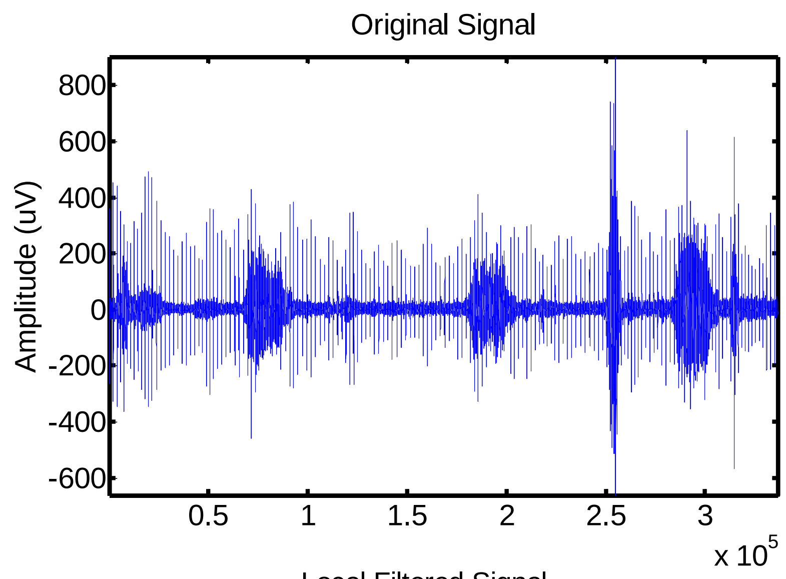

Original Signal
Amplitude (uV)
600
400
200
0
-200
-400
-600
0.5
1
1.5
2
2.5
3
x 10^5
Local Filtered Signal
Amplitude (uV)
600
400
200
0
-200
-400
-600
0.5
1
1.5
2
2.5
3
x 10^5
Global Filtered Signal
Amplitude (uV)
600
400
200
0
-200
-400
-600
0.5
1
1.5
2
2.5
3
x 10^5

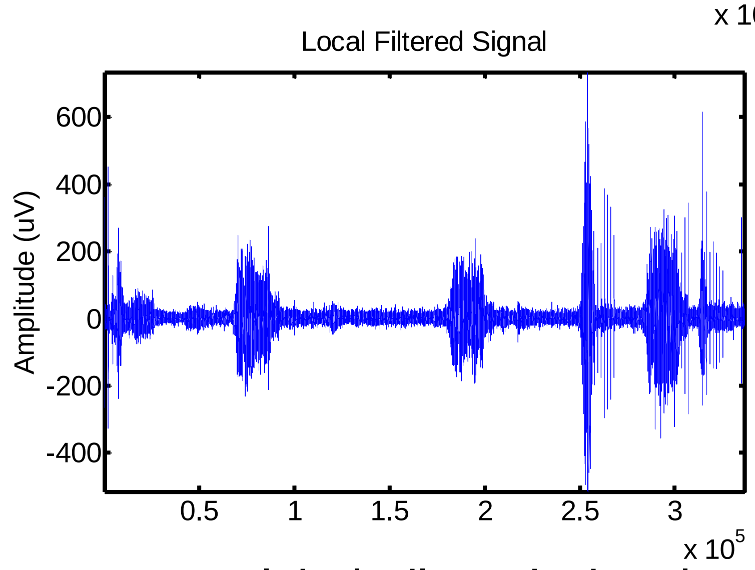

Original Signal
Amplitude (uV)
600
400
200
0
-200
-400
0.5
1
1.5
2
2.5
3
x 10^5
Local Filtered Signal
Amplitude (uV)
400
200
0
-200
-400
0.5
1
1.5
2
2.5
3
x 10^5
Global Filtered Signal
Amplitude (uV)
200
150
100
50
0
-50
-100
-150
0.5
1
1.5
2
2.5
3
x 10^5

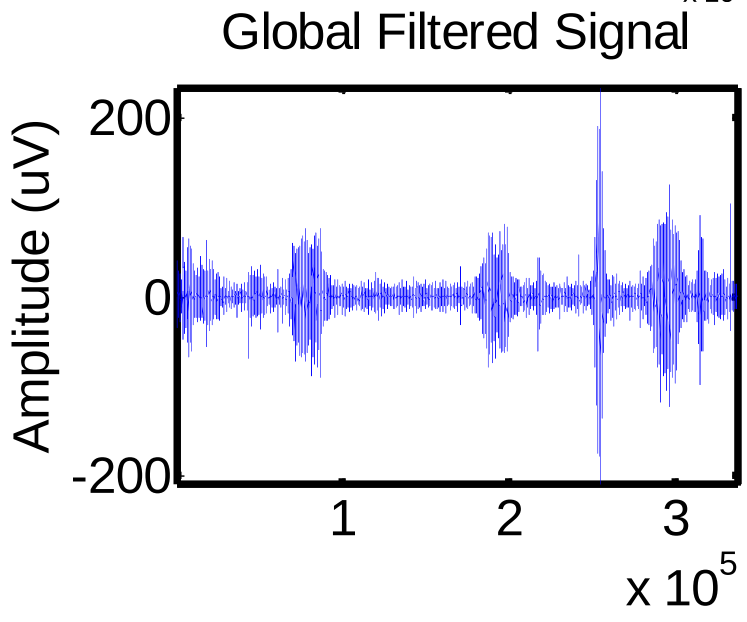

Original Signal
Amplitude (uV)
600
400
200
0
-200
-400
0.5
1
1.5
2
2.5
3
x 10^5

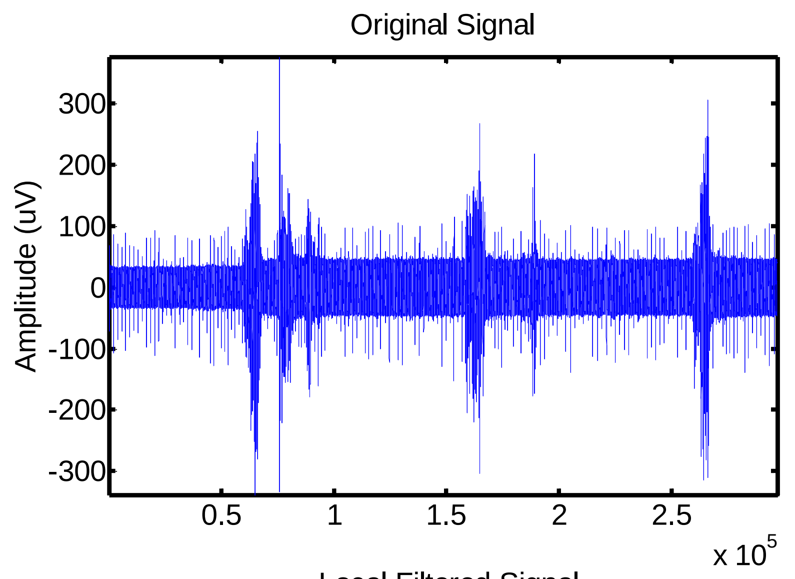

Local Filtered Signal
Amplitude (uV)
600
400
200
0
-200
-400
0.5
1
1.5
2
2.5
3
x 10^5

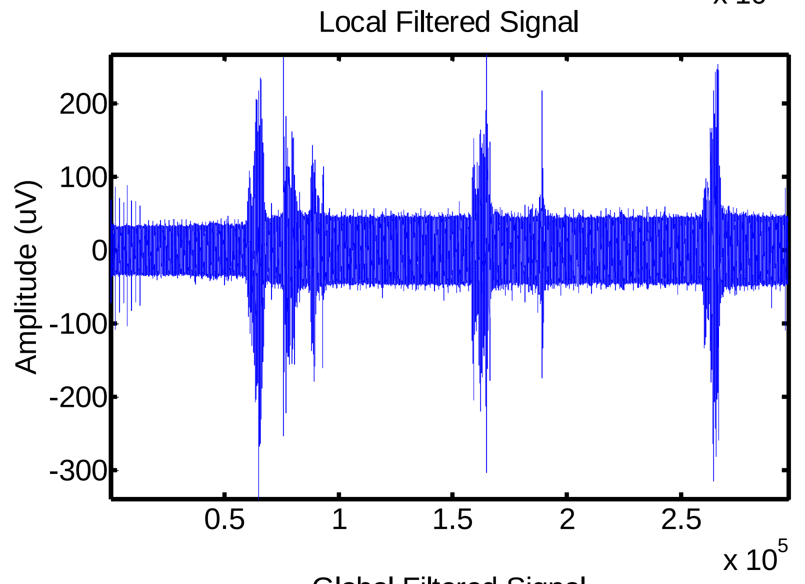

Global Filtered Signal
Amplitude (uV)
200
100
0
-100
-200
0.5
1
1.5
2
2.5
3
x 10^5

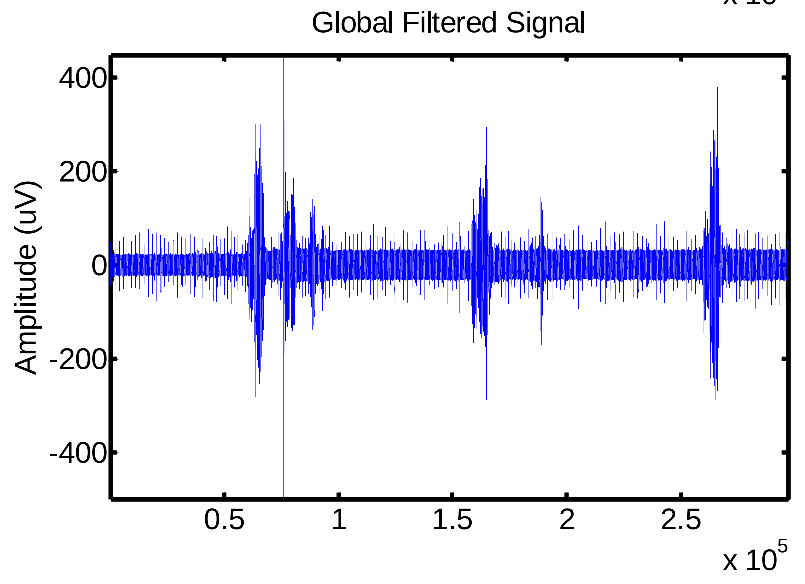

Original Signal
Amplitude (uV)
800
600
400
200
0
-200
-400
-600
0.5
1
1.5
2
2.5
3
x 10^5

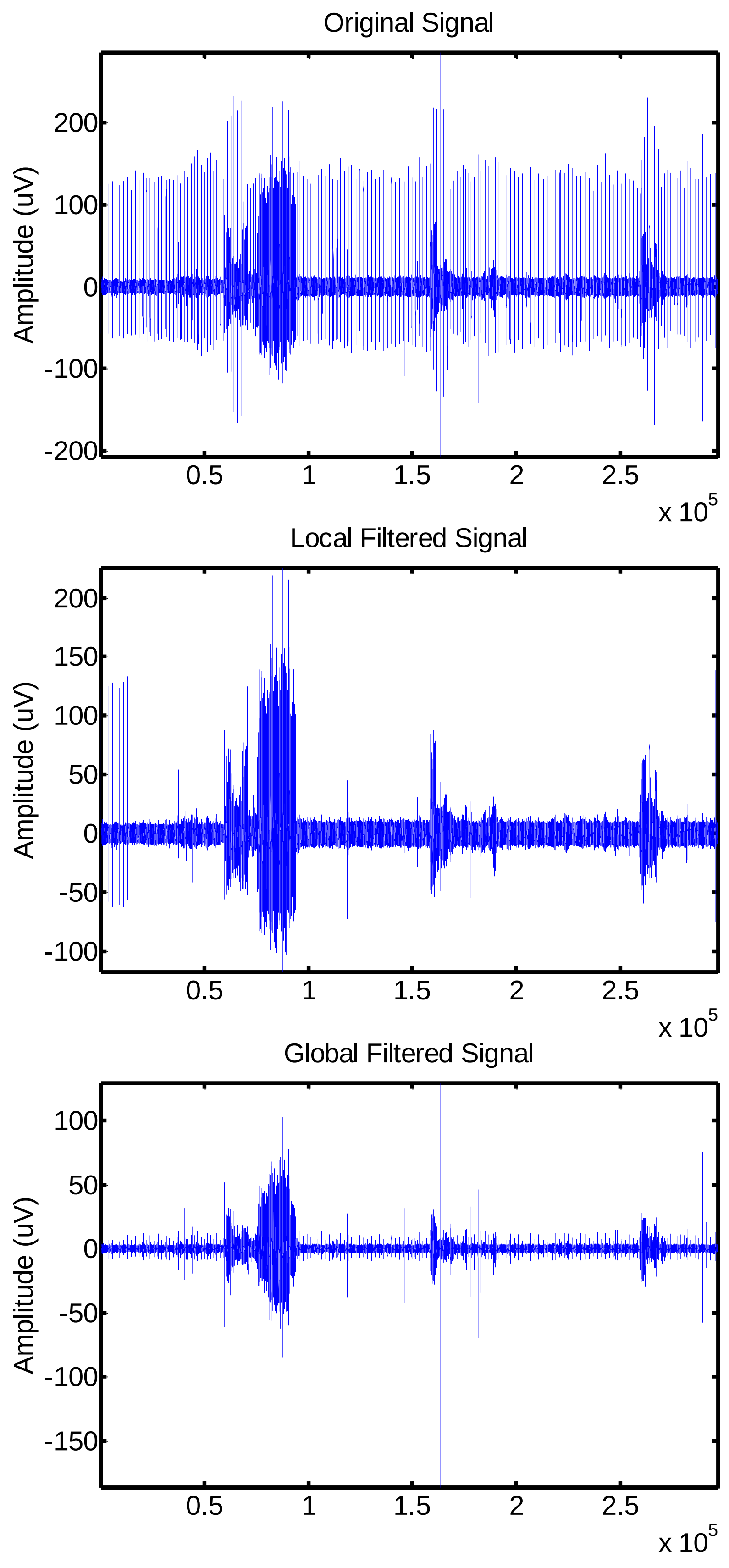

Local Filtered Signal
Amplitude (uV)
600
400
200
0
-200
-400
0.5
1
1.5
2
2.5
3
x 10^5

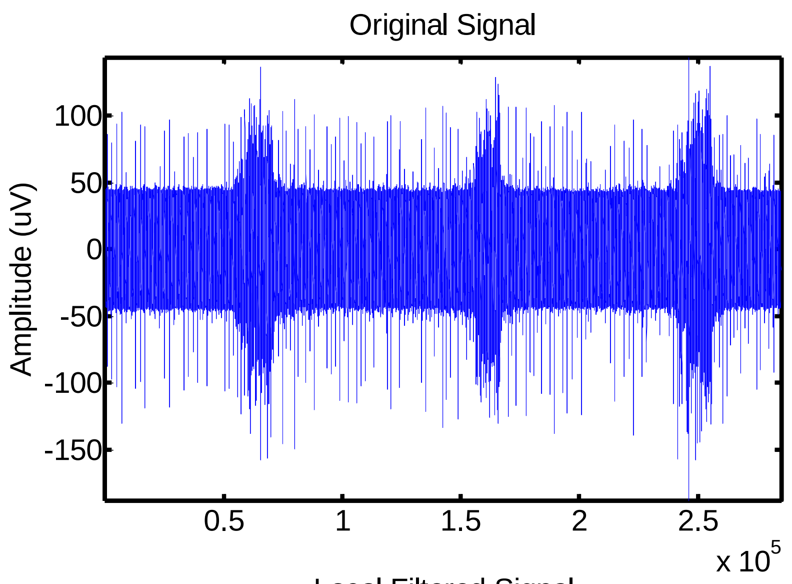

Global Filtered Signal
Amplitude (uV)
200
0
-200
1
2
3
x 10^5

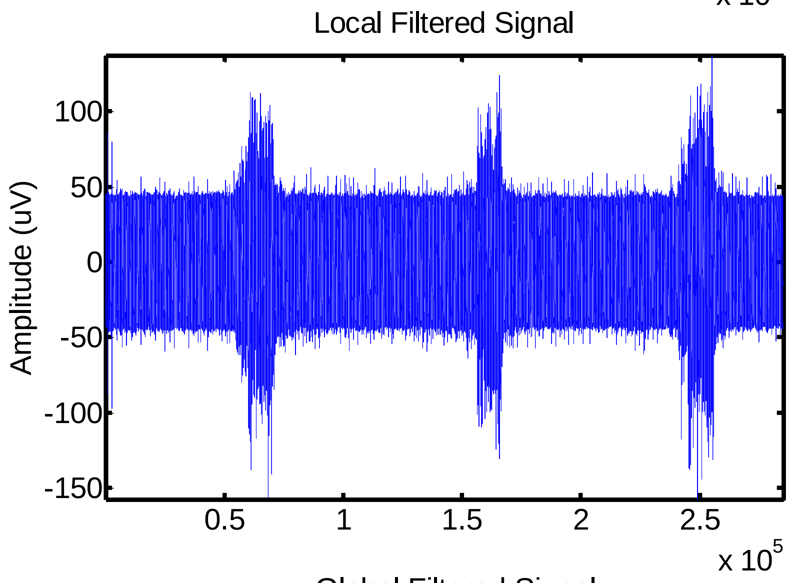

Original Signal
Amplitude (uV)
300
200
100
0
-100
-200
-300
0.5
1
1.5
2
2.5
x 10^5

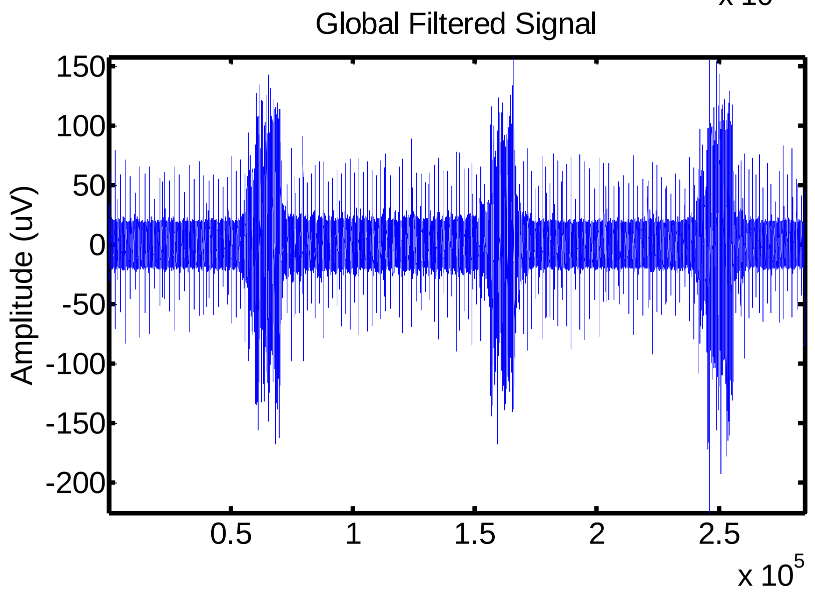

Local Filtered Signal
Amplitude (uV)
200
100
0
-100
-200
-300
0.5
1
1.5
2
2.5
x 10^5


Global Filtered Signal
Amplitude (uV)
400
200
0
-200
-400
0.5
1
1.5
2
2.5
x 10^5


Original Signal
Amplitude (uV)
200
100
0
-100
-200
0.5
1
1.5
2
2.5
x 10^5
Local Filtered Signal
Amplitude (uV)
200
150
100
50
0
-50
-100
0.5
1
1.5
2
2.5
x 10^5
Global Filtered Signal
Amplitude (uV)
100
50
0
-50
-100
-150
0.5
1
1.5
2
2.5
x 10^5


Original Signal
Amplitude (uV)
100
50
0
-50
-100
-150
0.5
1
1.5
2
2.5
x 10^5


Local Filtered Signal
Amplitude (uV)
100
50
0
-50
-100
-150
0.5
1
1.5
2
2.5
x 10^5


Global Filtered Signal
Amplitude (uV)
150
100
50
0
-50
-100
-150
-200
0.5
1
1.5
2
2.5
x 10^5

APPENDIX II

Matlab Code

The following raw code is presented for informative purposes only. No part of this code may be directly or indirectly used without an expressed consent from Nihit Bajaj, Matt Nitzken, or Dr. Ayman El-Baz. Certain functions were left out, and slightly modified to protect the originality of the code.

Main Function:

```
function [A_PROC signalecgSHIFT globalfilteredsignal] =
ecgAlignFilter3(A,ECG,signalFreq,ecgFreq,eventMatrix,k1)

    clc; close all;
    %enforce signal matrix dimensions (preventative global error handler)
    if (size(A,1)>size(A,2)); A = A'; end
    if (size(ECG,1)>size(ECG,2)); ECG = ECG'; end

    %align all the ecg signals
    [A_ECG, eventMarkers] =
rectifyDRIVER(A,ECG,signalFreq,ecgFreq,eventMatrix,k1);
    A_PROC = zeros(size(A));

    for i = 1:size(A,1);
        fprintf('Processing signal %d...\n',i);
        signal = A(i,:);
        signalecgSHIFT = A_ECG(i,:);

        center = eventMarkers(3) - round((eventMarkers(3) -
eventMarkers(2))/2);
        window = center-10000:center+10000;
        [hasECG] = ecgDetAltN(signal,eventMarkers);
%          [hasECG] = ecgDetAltN(abs(signal(window)))
%          [~, hasECG] = ecgDetAlt(signal,signalFreq,window);

        if (hasECG)
            fprintf('ECG Removal ACTIVE...\n');
            [filteredSignal_BSF] = bsfFiltering(signal,signalFreq,window);
            [filteredSignal] = filterDIRECTGD(filteredSignal_BSF);
            fprintf('ECG SELECT ACTIVE...\n');
            [processedSignal] =
filterECGSELECT2(signal,filteredSignal,signalecgSHIFT,eventMarkers);
            globalfilter = filteredSignal;
        else
            processedSignal = signal;
            globalfilter = signal;
        end

        globalfilteredsignal(i,:) = globalfilter;
        A_PROC(i,:) = processedSignal;
    end
    A_PROC = A_PROC'; %transpose array
end

%Recent updates:
%--------------
% - pushed all event marker controls down into rectifyDRIVER
% - lots of changes to core functions, doesn't suffer from filter bleeding
% anymore
```

Function Responsible for Creating Binary Masks of ECG:

```
function [A_ECG, eventMarkers] =
rectifyDRIVER(A,ECG,signalFreq,ecgFreq,eventMatrix,k1)

    %enforce signal unity
    if (size(A,1)>size(A,2)); A = A'; end
    if (size(ECG,1)>size(ECG,2)); ECG = ECG'; end
    fprintf('ECG Global Align ACTIVE...\n');

    %perform point shift calculations internally
    if (k1 == 0) %if k1 == 0 then assume the event markers are already the
correct bin format
        eventMarkers = eventMatrix; %pass the value through untouched
    else
        eventMatrix = getMEParse(eventMatrix); %clean the eventMatrix
variable (note this is a very unstable variable...)
        eventMarkers = round((eventMatrix - (k1/signalFreq))*signalFreq);
%find the starting bins of events
    end

    %declare variables
    ecgDATA = ECG(1,:); %this is the ECG data
    ecgPULSE = ECG(2,:); %this is the ECG pulse
    A_ECG = zeros(size(A)); %primary storage matrix for aligned ecg signals
    noSigs = size(A,1); %the number of signals we are processing
    alignMark = eventMarkers(3); %the primary alignment marker for an event
%     if (alignMark < 30000); alignMark = eventMarkers(3); end %error handle
(if second check fails the data is corrupt)
    signalPULSE = A(end,:); %the last row must always be the pulse

    %this is the master shift
    [ecgSHIFT] =
rectifyINIT(signalPULSE,ecgPULSE,ecgDATA,signalFreq,ecgFreq); %we rectify the
main bulk signal

%     figure;
    for i = 1:noSigs
        signalDATA = A(i,:);
        [sigSTRT] = rectifyECG(signalDATA,eventMarkers); %we need the start
of the data signal
        if (sigSTRT ~= 0)
            [signalecgSHIFT] = rectifyFINE(ecgSHIFT,signalDATA,sigSTRT);
            A_ECG(i,:) = signalecgSHIFT;
        else
```

```
            A_ECG(i,:) = ecgSHIFT;
        end
    end

end

%Recent updates:
%--------------
% modified for k1 passthrough
%
% moved getMEParse inside this function to subhandle the event markers and
% allow for proper passthrough
```

Function Responsible for automatically obtaining the time stamps for events.

```
function [eventMatrix] =  getMEParse(me, overload)

    melen = length(me);
    if (~exist('overload','var'))
        overload = 3;
    end
    marks = floor(melen/overload);

    if (marks == 2)
        eventMatrix = me;
    else
        eventMatrix = zeros(1,overload*2);
        for i = 1:overload
            idx0 = (2*i)-1;
            idx1a = (4*i)-3;
            idx1b = (4*i);
            eventMatrix(idx0) = me(idx1a);
            eventMatrix(idx0+1) = me(idx1b);
        end
    end
end
```

Specific Function that converts ECG from the ECG signal into binary mask through smart thresholding.

```
function [ecgSHIFT] =
rectifyINIT(signalPULSE,ecgPULSE,ecgDATA,signalFreq,ecgFreq)

    if (nargin<4)
        signalFreq = 2000;
        ecgFreq = 2000;
    end

    %rescale the ecg data
    scalar = ceil(signalFreq./ecgFreq);
    ecgPULSE = interp(ecgPULSE,scalar);
    ecgDATA = interp(ecgDATA,scalar);
    %binarize the signals
    spMax = max(signalPULSE);
    epMax = max(ecgPULSE);
    signalPULSE(signalPULSE < (spMax/2)) = 0;
    signalPULSE(signalPULSE ~= 0) = 1;
    ecgPULSE(ecgPULSE < (epMax/2)) = 0;
    ecgPULSE(ecgPULSE ~= 0) = 1;

    %grab pulse syncs
    spSTRT = find(signalPULSE==1,1);
    epSTRT = find(ecgPULSE==1,1);
    shift = spSTRT - epSTRT;

    %this is our output signal
    ecgSHIFT = zeros(1,length(signalPULSE));
    if (shift <= 0)
        if (shift==0); shift = 1; end
        temp = ecgDATA(abs(shift):end);
        if (length(temp) > length(ecgSHIFT))
            temp = temp(1:length(ecgSHIFT));
            ecgSHIFT = temp;
        else
            ecgSHIFT(1:length(temp)) = temp;
        end
    elseif (shift > 0)
        ecgSHIFT(abs(shift)+1:end) = ecgDATA(1:(length(ecgSHIFT)-
abs(shift)));
    end

    ecgSHIFT = binarizeECG(ecgSHIFT);
    ecgSHIFT = pulseTrack(ecgSHIFT,1000);
%     plot(ecgSHIFT); pause
end

function [ecgDATA] = binarizeECG(ecgDATA)

    myData = ecgDATA; %pull abs
    window = 10; %this is a rigid filter window (should never need to be
changed though)
    mySmoothData = zeros(1,length(myData)); %create storage var

    for i = 1:length(myData) %iterate data
        myWindow = i-window:i+window; %generate a dynamic window
        if(myWindow(1) < 1); myWindow = myWindow + (1-myWindow(1)); end
%ensure lower boundary conditions
        if(myWindow(end) > length(myData)); myWindow = myWindow +
(length(myData)-myWindow(end)); end %ensure upper boundary conditions

        movAvg = mean(myData(myWindow)); %calculate the running average
        mySmoothData(i) = movAvg; %store it
    end

    ecgDATA = mySmoothData;
%     plot(ecgDATA); pause
    thresh = (max(ecgDATA)+mean(ecgDATA))/2;
    %strip the data
    ecgDATA(ecgDATA < thresh) = 0;
    ecgDATA(ecgDATA ~= 0) = 1;
end


function [ecgOUT] = pulseTrack(ecgDATA,pulseCTRL)

    flood = 0; %initiate a flood fill
    pulse = 0; %indicate pulse is active
    pulseTRK = 0; %pulse tracker parameter
    pulseS0 = 0; %pulse storage
    pulseS1 = 0;

    ecgOUT = zeros(size(ecgDATA));

    for i = 1:length(ecgDATA)
        if (pulse)
            pulseTRK = pulseTRK + 1;
            if (pulseTRK >= pulseCTRL)
                pulse = 0;
                flood = 1;
            elseif (ecgDATA(i))
                pulseS1 = i;
                pulseTRK = 0;
            end
        elseif (flood)
            ecgOUT(pulseS0:pulseS1) = 1;
            flood = 0;
        else
            if (ecgDATA(i))
                pulseS0 = i;
                pulseTRK = 0;
                pulse = 1;
            end
        end
    end
end
```

Function Responsible for Outputting the location of ECG spike in EMG signal for alignment purposes.

```
function [sigSTRT] = rectifyECG(signalDATA,eventMarkers)


 [hasECG  SpikeLocation] = ecgDetAltN(signalDATA,eventMarkers);

     if ~(hasECG);
        sigSTRT = 0;
    else
        sigSTRT = SpikeLocation(3); %idx(index) + offset;
    end
```

```
end
```

Function Responsible for Detecting ECG in an EMG signal, and storing locations of ECG spikes for alignment.

```
function [hasECG  SpikeLocation] = ecgDetAltN(sig,eventMarkers)

    hasECG = 0;

    for i = 1:1:2
       contractionlocation = i;
        if(hasECG==0)
            signal = sig(eventMarkers(2*i):eventMarkers((2*i)+1));
            startPoint = eventMarkers(2*i);
            [hasECG SpikeLoc startbuffer window] = ecgDetAltSub(signal);

            if(hasECG)
            [r c] = size(SpikeLoc);

            if(r < 2)
              SpikeLoc = SpikeLoc(1,:);
            else
              SpikeLoc = SpikeLoc(2,:);
            end

            multiplier = SpikeLoc(1,1);
            SpikeLoc = SpikeLoc(2:end);
            SpikeLoc(SpikeLoc==0)  = [];
            SpikeLocation = SpikeLoc + (startPoint-1) + (startbuffer-1) +
(window*(multiplier-1));
            else
                SpikeLocation = 1;
            end
        end
    end
end


function [hasECG SpikeLocation outwind window] = ecgDetAltSub(signal)

    wind = 10000;
    outwind = wind;
    hasECG = 0;
    hECG = zeros(1,50);
    counter = 0;
    done = 0;
    window = 10000;
    error = 1;
    halcounter = floor(((length(signal)-outwind)./window)./2);

   signal = abs(signal);

    if((length(signal)-wind) < window*10)
```

```
        passes = floor((length(signal)-wind)./((5/2)*window));
    else
        passes = 3;
    end

    while(done==0)

       if(length(find(hECG>0)) > passes)
            hasECG = 1;
            done = 1;
        elseif(wind+window >= length(signal)-1)
            done = 1;                          %Could not find ECG in the signal
            NumberofSpikes = 0;
        end

       if(done==0)
            sig = signal(wind:wind+window);
            wind = wind + window;
            counter = counter + 1;

            signal2 = SmoothingAverage(sig);

            medianpoint = round(length(signal2)./2);

            max1 = max(signal2(1:medianpoint));
            max2 = max(signal2(medianpoint:length(signal2)));
            bband = BaseBandCalculation(signal2(1:end));

            absolutemax = max([max1 max2]);
            meanmax = mean([max1 max2]);

            cond = 0;
            if(abs(max1-max2) < (4*bband))
                cond = 1;
            elseif((max1 > 25*bband)&&(max2 > 25*bband))
                cond = 1;
            end

            if((meanmax >= 1.75*bband) &&(cond==1))

                spikes = (signal2 > (meanmax*0.6));


                [NumberofSpikes validation spikel] = ValidateSpikes(spikes);

                 if(((NumberofSpikes > 2) && (NumberofSpikes < 10))&&
(validation==1))
                    hECG(counter) = 1;
                    if(error==1)


                    spikel2 = SpikeCheck2(spikel);
                    SpikeLocation(1,:) = [counter spikel2];
                    error = 0;
                    end
                 end

            end



            if(((hECG(counter)==1) && (NumberofSpikes>=4)) &&
((NumberofSpikes <=6) &&(validation==1)))
              [r c] = size(SpikeLocation);
                if((r < 2)|| ((counter >=halcounter-1)&&(counter <=
halcounter+1)))
                    if(r > 1)
                        SpikeLocation(2,:) = [];
                    end

                    spikelt = SpikeCheck2(spikel);

                    if(length(SpikeLocation) > (length(spikelt)+1))
                        spikelt(end+1:end+(length(SpikeLocation)-
length(spikelt))-1) = 0;
                    elseif(length(SpikeLocation) < (length(spikelt)+1))
                        SpikeLocation(end+1:end+(length(spikelt)-
length(SpikeLocation))+1) = 0;
                    end
                    SpikeLocation(2,:) = [counter spikelt];
                end

            end
        end
    end

    if(error==1)
        SpikeLocation = zeros(1,5);
    end

end


function [output validation spikelocation] = ValidateSpikes(input)

    validation = 0;
    for i = 1:length(input)
        if(input(i) > 0)
            input(i+1:i+200) = 0;
            i = i + 199;
        end
    end

    spikelocation = find(input > 0);
    output = length(spikelocation);

    medianpoint = round(length(input)./2);
    valid1 = length(find(input(1:medianpoint) > 0));
    valid2 = length(find(input(medianpoint:end) > 0));

    valdiff = abs(valid1 - valid2);

    if((valdiff < 3)||((valid1 > 1)&&(valid2 > 1)))
        validation = 1;
    end

    [r c] = size(spikelocation);
    if(r > c)
        spikelocation = spikelocation';
    end
end


function [output] = BaseBandCalculation(signal)
    relativemax = zeros(1,round(length(signal)./100));
    counter = 1;

    for i = 1:100:length(signal)-500
        relativemax(counter) = max(signal(i:i+100));
        counter = counter + 1;
    end

    relativemax(relativemax==0) = [];

    output = median(relativemax);


end


function [output] = SmoothingAverage(signal)

    window = 10;
    half = round(window./2);
    signal2 = signal;

    for i = (half+1):1:(length(signal)-half)
        meanval = mean(signal((i-half):1:(i+half)));
        signal2(i) = meanval;
    end

    output = signal2;
end


function [output] = SpikeCheck2(input)

    difference = diff(input);
    input(find(difference < 750)+1) = [];
    output = input;
```

```
end
```

Function Responsible for pulse alignment of the signals.

```
function [signalecgSHIFT] = rectifyFINE(ecgSHIFT,signalDATA,sigSTRT)

    %grab pulse syncs

    idx = find(ecgSHIFT==1);
    idxD = diff(idx);
    index = find(idxD>10)+1;
    index = idx(index);
    [val idx] = min(abs(index-sigSTRT));
    epSTRT = index(idx);
    shift = sigSTRT - epSTRT;

    pulse0 = find(ecgSHIFT==0);
    pulse1 = find(ecgSHIFT==1,1);
    pulseS = find(pulse0 > pulse1,1);
    pulseS = pulse0(pulseS);
    pulsewidth = pulseS-pulse1;

    %this is our output signal
    signalecgSHIFT = zeros(1,length(signalDATA));
    if (shift <= 0)
        shift = shift-(round(pulsewidth.*0.15));
        temp = ecgSHIFT(abs(shift):end);
        if (length(temp) > length(signalecgSHIFT))
            temp = temp(1:length(signalecgSHIFT));
            signalecgSHIFT = temp;
        else
            signalecgSHIFT(1:length(temp)) = temp;
        end
    elseif (shift > 0)
        shift = shift+(round(pulsewidth.*0.15));
        signalecgSHIFT(abs(shift)+1:end) = ecgSHIFT(1:(length(ecgSHIFT)-
abs(shift)));
    end
end
```

Function Responsible for Band Stop filtering as a preparation for the wavelet filtering.

```
function [filteredSignal] = bsfFiltering(signal,freq,noiseInterval)

    [bsLow bsHigh] = bsfGetBands(signal(noiseInterval),freq);
    [filteredSignal] = bsfFS1FS2(signal,freq,bsLow,bsHigh);

end

function [bsLow bsHigh] = bsfGetBands(filteredSignal,fs)

    fourSignal = abs(fft(filteredSignal,2*fs)); %take FFT of signal
    fourSignal = fourSignal(1:round(length(fourSignal)/2)); %nuke the second
half of the fft

    myMax = max(fourSignal); %get the max
    modSignal = fourSignal - (myMax/3); %make fft noise negative
    modSignal(modSignal<0) = 0; %kill anything that is negative
%     plot(modSignal)

    %allocate variables
    clusterStart = [];
    clusterEnd = [];
    searching = 1;
    continuing = 0;
    giveUp = fs/20; %giveup is our sampling rate over 20

    for i = 1:length(modSignal) %search the signal
        if(searching) %we have no current cluster and are looking for one
            if (modSignal(i)>0) %we found data, this is the start of a
cluster
                continuing = 1; %we will now extend the cluster
                searching = 0; %we are no longer searching
                clusterStart(end+1) = i; %this is the beginning of the
cluster @ i
                potentialEnd = i; %this COULD be the end at i if it is a
spike
                giveUp = fs/20; %reset giveUp
            end
        elseif(continuing) %we are currently trying to track a cluster
            if (modSignal(i)>0) %we found an additional peak belonging to the
cluster
                giveUp = fs/20; %reset giveUp
                potentialEnd = i; %this now may be the new end
            else
                giveUp = giveUp - 1; %we found nothing, begin giving up
            end

            if (giveUp <= 0) %we have given up our continued search
                continuing = 0; %no longer continuing this search
                searching = 1; %start a new search
                clusterEnd(end+1) = potentialEnd; %our potential end is now
our true end
            end
        end
    end

    if (length(clusterEnd)<length(clusterStart)); clusterEnd(end+1) =
length(modSignal); end

    centerPeak = clusterStart(1) + round((clusterEnd(1)-clusterStart(1))/2);
%this is the center of the first cluster (aka the noise)
    spreadPeak = 2*round((clusterEnd(1)-clusterStart(1))/2); %double the
cluster width

    if (spreadPeak < 5); spreadPeak = 10; end %account for collapse
    bsLow = centerPeak-spreadPeak; %lower band
    bsHigh = centerPeak+spreadPeak; %higher band
end

% [noiseStart noiseStop] = intelliNoise(signal,noiseLen,freq)
% RMCA - Smart noise creation algorithm
%
% Inputs:
%   signal - [1D double] - a 1 dimensional signal (noise reference)
%   fs - [int] - the frequencyt of the sampled waveform
%   fs1 - [int] - the frequency of start-stop band
%   fs2 - [int] - the frequency of stop-stop band
%
% Outputs:
%   filteredSignal - [1D double] - a 1 dimensional signal
% ---------------------------------------------------

function [filteredSignal] = bsfFS1FS2(signal,fs,fs1,fs2)

    % LPF Paramters
    fp1 = fs1 - 0.1;
    fp2 = fs2 + 0.1;
    Fs1 = fs1./fs;
    Fs2 = fs2./fs;
    Fp1 = fp1./fs;
    Fp2 = fp2./fs;
    F0 = (Fs1 + Fs2)./2;
    Fp = 0.5.*(Fs2 - Fs1);
    Fs = 0.5.*(Fp2 - Fp1);

    Fc = (Fp + Fs)./2;
    C = 7.96;

    NoOfSamples = round(C./Fc); %get number of samples
    if ((NoOfSamples/2)==fix(NoOfSamples./2)); NoOfSamples = NoOfSamples +1;
end %ensure sample count is odd
    N = (-(NoOfSamples-1)/2):1:((NoOfSamples-1)/2); %make range of N
    w_n = harrisWin7(NoOfSamples); %get harris window for samples

    h_LPF = impulseResponseLPF(w_n,Fc); %to calculate the impuse response of
LPF

    n = 0;
    h_BSF = zeros(1,length(N)); %allocate BSF
    for i = N %perform calculation
        n = n+1;
        if i == 0
            h_BSF(n) = 1 - 2.*cos(2.*F0.*pi.*i).*h_LPF(n);
        else
            h_BSF(n) = - 2.*cos(2.*F0.*pi.*i).*h_LPF(n);
        end
    end

    Y_C = conv(signal,h_BSF); %perform convolution
    filteredSignal = Y_C(fix(NoOfSamples./2):1:length(Y_C)-
fix(NoOfSamples./2+1)); %get filtered signal
end

function h_LPF = impulseResponseLPF(w_n,Fc)
    n=0;
    N = length(w_n);
    for i = -(N-1)/2:1:(N-1)/2
        n = n+1;
        if (i==0)
            h(n) = 2.*Fc; %To calculate h[n]
        else
            h(n) = ((2.*Fc).*(sin(2.*Fc*pi*i)))./(2.*Fc.*pi.*i);
        end
    end
    h_LPF = h.*w_n;
end

function w_n = harrisWin7(NoOfSamples)
    b0 = 0.355768;
    b1 = 0.487396;
    b2 = 0.144232;
    b3 = 0.012604;

    i=0;
    for n = 0:1:(NoOfSamples)-1
        i = i+1;
        w_n(i) = b0 - b1.*cos(2.*n.*pi./(NoOfSamples-1)) +
b2.*cos(4.*n.*pi./(NoOfSamples-1)) - b3.*cos(6.*n.*pi./(NoOfSamples-1));
    end
end
```

Function Responsible for Wavelet Filtering

```
function [filteredSignal] = filterDIRECTGD(signal)

%     [filteredSignal] = bsfFiltering(signal,freq,1:10000);
    [noiseEnd] = fminbnd(@(x)
gradDecentEnd(x,signal),70,95,optimset('TolX',1,'MaxIter',8,'Display','off'))
; %perform gradient descent analysis (use anon ref to eliminate globals)
    noiseEnd = round(noiseEnd);
    [filteredSignal] = preWaveFilt(signal,noiseEnd);
%     [filteredSignal] = bsfFiltering(filteredSignal,freq,1:10000);

end

function [myEnergy] = gradDecentEnd(noiseEnd,signal)

    signal = signal(1:round(length(signal)/4));
    noiseEnd = round(noiseEnd); %round our input (we are using a tolerance of
1 so dec is worthless)
    [filteredSignal] = preWaveFilt(signal,noiseEnd);
    [filteredSignal] = fixSignalScaling(signal, filteredSignal); %fix scaling
    [myEnergy] = getMyEnergy(signal, filteredSignal); %get energy
```

```
end

function [filteredSignal] = preWaveFilt(signal,noiseEnd)

    noOfWaves = 128; %number of wavelets to use
    mySignalEnergy = getMyEnergy(signal, signal); %the current signal energy

    [filteredSignal] = preWaveReject(signal, noOfWaves, noiseEnd);
    [filteredSignal] = fixSignalScaling(signal, filteredSignal);
    mySignalEnergy(end+1) = getMyEnergy(signal, filteredSignal); %get the new
energy

    processing = 1; %engine is hot!
    iter = 0;
    maxIter = 5;
    while (processing)
        previousSignal = filteredSignal; %store our last signal
        [filteredSignal] = preWaveReject(filteredSignal, noOfWaves,
noiseEnd);
        mySignalEnergy(end+1) = getMyEnergy(signal, filteredSignal); %get the
new energy

        iter = iter+1;

        if (mySignalEnergy(end) > mySignalEnergy(end-1)) || (iter >= maxIter)
            processing = 0;
            filteredSignal = previousSignal;
        end
    end
end

function [filteredSignal] = preWaveReject(signal, noOfWaves, noiseEnd)

    [C] = waveDecompose(signal,noOfWaves); %decompose the whole signal

    killList = zeros(1,size(C.cfs,1));
    killList(30:noiseEnd) = 1;

    C.cfs(killList==1,:) = [];
    C.scales(killList==1) = [];

    filteredSignal = icwtft(C);
end

% [cwtstruct] = decomposeSignal(signal,noOfWaves)
% RMCA - Smart noise creation algorithm
% Inputs:
%   signal - [1D double] - a 1 dimensional signal (noise reference)
%   noOfWaves - [int] - desired number of wavelets to decompose the signal
into
%
% Outputs:
%   noiseStart - [int] - the integer of the noise start
%   noiseStop - [int] - the integer of the noise stop
% ---------------------------------------------------

function [cwtstruct] = waveDecompose(signal,noOfWaves)

    myLen = length(signal); %get the no of data points in our signal
    myWavelet = 'morl'; %we will use morlet decomposition by default

    dt = 1; %out time step is 1 (while sampling may actually not be 1, this
is a time hack)
    s0=2*dt; %our starting timestep is 2 times the beginning of our time
    % dS=0.4875; %default dS value for morlet
    dS = log2(myLen)/(noOfWaves); %back calculate the oversampling rate (ONLY
FOR MORLET!!!!)

    J1 = fix(log2(myLen)/dS)-1; %recalculate the new number of waves (this
should equal noOfWaves)
    criticalJ1 = fix(log2(myLen)/0.4875)-1; %calculate the critical number of
waves
    if (J1 < criticalJ1); fprintf('WARNING :: You are undersampling this
signal!'); end %check for undersampling
    scales = s0*2.^((0:J1)*dS); %calculate the actual scale values (override
the defaults!)

    cwtstruct = cwtft(signal,'scales',scales,'wavelet',myWavelet); %decompose
the wavelet using hacked values for scales

    % to recompile the signal simply use:
    % xrec = icwtft(cwtstruct);

end
```

Function Responsible for localizing the filtering process:

```
function [processedSignal] =
filterECGSELECT2(signal,filteredSignal,signalecgSHIFT,eventMarkers)

    idx = find(signalecgSHIFT==1); %get locations where ecg is active
    idxD = diff(idx); %find differences between values of ecg
    index = find(idxD>10)+1; %extract the initial edges of each ECG
    altindex = find(idxD<10); %extract the values relating to the width of
pulses
    ecgINDEX = idx(index); %pull the locations of ecg pulses in real data
    pulses = length(ecgINDEX); %find the numer of pulses
    pulseWidth = ceil(length(altindex)/pulses); %find the average width of a
pulse
    if (pulseWidth < 500); pulseWidth = 250; end  %change pulsewidth back to
500.
%     pulseWidth = 1000; %come forward and backward off an edge by a width of
2000

    %form event markers
    OFFSET = 3000; %push a forced offset into markers
    [eventPULSE] = generateEventPulse(eventMarkers,length(signal),OFFSET);
%generate a pulse marker for only events

    processedSignal = signal; %duplicate signal
    for i = 2:pulses-1 %ignore first and last ecg (avoids runoff boundary
conditions)
        t0 = ecgINDEX(i); %index of the start edge of an ecg pulse
        bin1 = signal(t0-(1*pulseWidth):t0+(1*pulseWidth)); %pull the bin
containing the ecg
        bin1P = filteredSignal(t0-(1*pulseWidth):t0+(1*pulseWidth)); %pull
the bin containing the filtered ecg

        [filteredbin] = filterMODSIGNAL4(eventPULSE(t0),bin1,bin1P);
        processedSignal(t0-(1*pulseWidth):t0+(1*pulseWidth)) = filteredbin;
%inject filtered data back into main data stream
    end
end

function [eventPULSE] = generateEventPulse(eventMarkers,signalLen,OFFSET)

    eventMarkers = round(eventMarkers); %push eventMarkers into INTEGER bins
    eventPULSE = zeros(1,signalLen); %create a pulse framework (default = all
0s)
    events = floor(length(eventMarkers)/2); %find the number of events
present (assume that any odd events are runoff errors)

    for i = 1:events %iterate through events
        startIDX = (2*i)-1; %set a shifting start index
        endIDX = startIDX + 1; %set a shifting end index
        eventPULSE(eventMarkers(startIDX)-OFFSET:eventMarkers(endIDX)+OFFSET)
= 1; %between the shifting indices is an event (mark it)
    end
end


%Changes
%------------------------
% - Dynamic binning has been removed
% - aggression control has been removed (control dynamically calculated in
% current filterMODSIGNAL X function.
% - removed static callbacks


function [output] = filterMODSIGNAL4(event,rawsignal,filteredsignal)

    %currently there is no difference between inside/outside event
    %processing. The legacy separation has been left in case it is needed.
    if(event==0) %if inside noise
        [output] = smartFilter(rawsignal,filteredsignal);
    else %if inside event
        [output] = smartFilter(rawsignal,filteredsignal);
    end
end

function [output] = smartFilter(input,filteredInput)
```

```
    Contents are Hidden
end

function [output] = iceFilter(input)

    %enforce matrix size
    if (size(input,1)>size(input,2)); input = input'; end
    myData = abs(input); %pull abs
    window = 10; %this is a rigid filter window (should never need to be
changed though)
    mySmoothData = zeros(1,length(myData)); %create storage var

    for i = 1:length(myData) %iterate data
        myWindow = i-window:i+window; %generate a dynamic window
        if(myWindow(1) < 1); myWindow = myWindow + (1-myWindow(1)); end
%ensure lower boundary conditions
        if(myWindow(end) > length(myData)); myWindow = myWindow +
(length(myData)-myWindow(end)); end %ensure upper boundary conditions

        movAvg = mean(myData(myWindow)); %calculate the running average
        mySmoothData(i) = movAvg; %store it
    end

    [~, idx] = max(mySmoothData); %find the max (this is our real ecg center
    %now for some tricky stuff
    int1 = mySmoothData(1:idx); %separate the first half of the data
    int2 = mySmoothData(idx+1:end); %separate the second half
    int1s = int1(1:round(length(int1)/2)); %now split the first half again
    int2s = int2(round(length(int2)/2):end); %split second again as well
    mean1 = mean(int1s); %pull the mean of the first quarter (its not
actually a quarter, its just a theoretical quarter)
    mean2 = mean(int2s); %do the same for the last quarter
    %ok, now to create an actual window :)
    int1(int1<mean1) = 0; %kill the first half thats lower than our noise
mean
    int1(int1~=0) = 1; %push the rest to 1
    int2(int2<mean2) = 0; %kill the second half thats lower than our noise
mean
    int2(int2~=0) = 1; %push the rest to 1
    [idx1] = find(fliplr(int1)==0,1); %flip the first half matrix and pull
off the first 1
    [idx2] = find(int2==0,1); %pull off the first one of the second half

    output = zeros(1,length(input)); %create a mask storage
    output(idx-idx1:idx+idx2) = 1; %using our center and detected edges force
a mask over the ECG
end
```

Function Responsible for creating Type I Phantom Signals

```
function [plotx csignal nsignal ecgsignal marker] =
Signal_Simulation(numberofcontractions,emgamplitude,heartrate,amphigh,amplow,
inv,randomize)
```

```
EMGBaseBandAmp = 25;

marker = zeros(1,numberofcontractions.*2);
[emgx emgy] = EMG_Simulator(EMGBaseBandAmp);
[ecg] = ECG_Wave2(heartrate,amphigh,amplow,inv);

[emgh emgw] = size(emgy);
[ecgh ecgw] = size(ecg);

if(emgw > ecgw)
    ecg(ecgw+1:emgw) = 0;
else

    xdiff = emgx(2)-emgx(1);
    endsample = ((ecgw-emgw)*xdiff) + emgx(emgw);
    emgx(emgw+1:ecgw)= emgx(emgw)+xdiff:xdiff:endsample;

    while(length(emgy) < ecgw)

        emgy = [emgy emgy];

    end
end



emgy = emgy(1:ecgw);
emgy = awgn(emgy,1);
emgnoise = emgy./5;
emgnoise = awgn(emgnoise,5);

if(nargin > 0)
    numb = numberofcontractions + 1;
else
    numb = round(rand()*3)+3;
end
numbercontractions = numb-1;

index = round(length(emgx)/numb);

markindex = 1;
for i = 1:1:numbercontractions
    contractionlength = round(rand()*12000)+12000;

    index2 = index*i;
    randstart = (ceil((rand()*8)+0.01))+1;
    emgamp = round(emgamplitude./(6.*(EMGBaseBandAmp/10)));
    randamp = floor(emgamp*0.75);

    amplitude = round(rand()*randamp) + (emgamp-randamp);
```

```
        emgspike = amplitude*sin((pi/randstart):(pi-
(pi/randstart))/(contractionlength-1):pi);
        emgspike = awgn(emgspike,5);

        if(randomize==1)
            randomcluster = rand(1,length(emgspike));
            randomcluster(randomcluster < 0.5) = 1;

            emgspike = emgspike.*randomcluster;

        end

        emgcontraction = [zeros(1,index2-1) emgspike];

        marker(markindex) = index2;
        markindex = markindex + 1;

        ll = length(emgcontraction);

        marker(markindex) = ll;
        markindex = markindex + 1;

        emgcontraction2 = [emgcontraction zeros(1,length(emgx)-ll)];


        emgg = emgy.*emgcontraction2;
        emgy = emgg + emgy;


        clear emgcontraction emgcontraction2 randomcluster

    end

    signal = ecg(1:ecgw) + emgy(1:ecgw);
    emgx = emgx-1;

    shift = sum(signal(1:5000))/5000;
    signal = signal-shift;

    plotx = [1:1:length(emgx)];

    csignal = emgy(1:ecgw);
    nsignal = signal;
    ecgsignal = ecg(1:ecgw);
end
```

Function Responsible for Creating ECG component of the Phantom Signals

Note: Most of this function is taken from the free library by Karthik. It has been appropriately referenced in the text.

```
function [output] =
ECG_Wave2(heartrate,amplitudeh,amplitudel,inverted,signallength)

if(nargin < 5)
    signallength = 140; %x-axis length in seconds
end


samplingrate = 2000; %sampling rate in Hz
amplitude = mean([amplitudeh amplitudel]);

x=0.01:(1/samplingrate):signallength;

      li=30/heartrate;

      %p-wave specifications
      a_pwav = 0.01;
      %a_pwav=amplitude./6.4; %amplitude
      d_pwav=0.03;           %duration
      t_pwav=0.16;           %p-r interval

      %q-wave specifications
      a_qwav=amplitude./64;  %amplitude
      d_qwav=0.033;          %duration 0.033
      t_qwav=0.166;          %DO NOT CHANGE

      %QRS specifications
      a_qrswav=amplitude./500;   %amplitude of QRS complex %=amplitude
      d_qrswav=0.01;        %Duration of QRS complex

      %s-wave specifications
      %a_swav=amplitude./6.4;
      a_swav = amplitudel; %amplitude
      d_swav=0.02;             %duration
      t_swav=0.06;              %DO NOT CHANGE 0.09

      %t-wave specifications
      a_twav=amplitudeh; %amplitude    %=amplitude./4.5714
      d_twav=0.021;             %duration %0.051
      t_twav=0.000;               %s-t interval

      %u-wave specifications
      a_uwav=amplitude./500; %amplitude
      d_uwav=0.00001;              %duration  %0.00001;
      t_uwav=0.433;              %DO NOT CHANGE



 pwav=p_wav(x,a_pwav,d_pwav,t_pwav,li);


 %qwav output
 qwav=q_wav(x,a_qwav,d_qwav,t_qwav,li);
 %qrswav output
 qrswav=qrs_wav(x,a_qrswav,d_qrswav,li);

 %swav output
 swav=s_wav(x,a_swav,d_swav,t_swav,li);


 %twav output
 twav=t_wav(x,a_twav,d_twav,t_twav,li);


 %uwav output
 uwav=u_wav(x,a_uwav,d_uwav,t_uwav,li);

 %ecg output
 ecg=pwav+qrswav+twav+swav+qwav+uwav;
if(inverted==1)
    ecg = ecg.*-1;
end

shift = mean(ecg(500:1000));
output = ecg - shift;
%output = awgn(output,1);
end




function [qrswav]=qrs_wav(x,a_qrswav,d_qrswav,li)
    l=li;
    a=a_qrswav;
    b=(2*l)/d_qrswav;
    n=2000;
    qrs1=(a/(2*b))*(2-b);
    qrs2=0;
    for i = 1:n
        harm=(((2*b*a)/(i*i*pi*pi))*(1-cos((i*pi)/b)))*cos((i*pi*x)/l);
        qrs2=qrs2+harm;
    end
    qrswav=qrs1+qrs2;
end


function [pwav]=p_wav(x,a_pwav,d_pwav,t_pwav,li)
    l=li;
    a=a_pwav;
    x=x+t_pwav;
    b=(2*l)/d_pwav;
    n=100;
    p1=1/l;
    p2=0;
    for i = 1:n
```

```
        harm1=(((sin((pi/(2*b))*(b-(2*i))))/(b-
(2*i))+(sin((pi/(2*b))*(b+(2*i))))/(b+(2*i)))*(2/pi))*cos((i*pi*x)/l);
        p2=p2+harm1;
    end
    pwav1=p1+p2;
    pwav=a*pwav1;
end


function [qwav]=q_wav(x,a_qwav,d_qwav,t_qwav,li)
    l=li;
    x=x+t_qwav;
    a=a_qwav;
    b=(2*l)/d_qwav;
    n=100;
    q1=(a/(2*b))*(2-b);
    q2=0;
    for i = 1:n
        harm5=(((2*b*a)/(i*i*pi*pi))*(1-cos((i*pi)/b)))*cos((i*pi*x)/l);
        q2=q2+harm5;
    end
    qwav=-1*(q1+q2);
end


function [swav]=s_wav(x,a_swav,d_swav,t_swav,li)
    l=li;
    x=x-t_swav;
    a=a_swav;
    b=(2*l)/d_swav;
    n=100;
    s1=(a/(2*b))*(2-b);
    s2=0;
    for i = 1:n
        harm3=(((2*b*a)/(i*i*pi*pi))*(1-cos((i*pi)/b)))*cos((i*pi*x)/l);
        s2=s2+harm3;
    end
    swav=-1*(s1+s2);

end


function [twav]=t_wav(x,a_twav,d_twav,t_twav,li)
    l=li;
    a=a_twav;
    x=x-t_twav-0.045;
    b=(2*l)/d_twav;
    n=100;
    t1=1/l;
    t2=0;
    for i = 1:n
        harm2=(((sin((pi/(2*b))*(b-(2*i))))/(b-
(2*i))+(sin((pi/(2*b))*(b+(2*i))))/(b+(2*i)))*(2/pi))*cos((i*pi*x)/l);
        t2=t2+harm2;
    end
    twav1=t1+t2;
    twav=a*twav1;

end


function [uwav]=u_wav(x,a_uwav,d_uwav,t_uwav,li)
    l=li;
    a=a_uwav;
    x=x-t_uwav;
    b=(2*l)/d_uwav;
    n=100;
    u1=1/l;
    u2=0;
    for i = 1:n
        harm4=(((sin((pi/(2*b))*(b-(2*i))))/(b-
(2*i))+(sin((pi/(2*b))*(b+(2*i))))/(b+(2*i)))*(2/pi))*cos((i*pi*x)/l);
        u2=u2+harm4;
    end
    uwav1=u1+u2;
    uwav=a*uwav1;
end
```

Function Responsible for Creating the EMG component of Phantom Signal

```
function [outputx outputy] = EMG_Simulator(amplitude)
% clc
% clear all
% close all

    startIndex = 1;
    endIndex = 2300;

         dataI = importdata('');

        datax = dataI.data(:,1);
        datay = dataI.data(:,2);

        datax = transpose(datax);
        datay = transpose(datay);
        indicator = 0;
        while indicator ==0

            [r c] = size(find(diff(datax)==0));

            if(c ==0)
                indicator = 1;
            end

            if(indicator==0)

               ind = find(diff(datax)==0);
                datax(ind(:)) = datax(ind(:))-0.00000001;
            end
        end

            xi = 1:1/2000:dataI.data(endIndex,1);
            yi = interp1(datax,datay,xi);

            difference = max(yi)-min(yi);
            ampratio = difference./amplitude;
            yi = yi./ampratio;
            difference = max(yi)-min(yi);
            shift = fix(difference./2);

            outputx = xi;
            outputy = yi-shift;
end
```

Function Responsible for creating Type II Phantoms:

```
function [plotx csignal nsignal ecgsignal marker] =
Invivo_Simulation(inputemg,heartrate,amphigh,amplow,inv)


load '';
    index = inputemg;

    emgy = A(:,index);
    emgx = 1:1:length(emgy);


    if(exist('ME'))
      eventMatrix = getMEParse(ME); %clean the eventMatrix variable (note
this is a very unstable variable...)
       marker = round((eventMatrix - (k1/2000))*2000); %find the starting
bins of events
    elseif(exist('DPOINT'))
        marker = generateEventMarkers(A(:,21),3,20000);
    else
                                     %Guesses Potential Event Markers
%       contractionlength  = round(length(emgy)./10);
%       stp = round(length(emgy)./8);   %Contraction Start Point
%
%       marker(1) = stp;
%       marker(2) = stp + contractionlength;
%       marker(3) = stp.*4;
%       marker(4) = marker(3) + contractionlength;
%       marker(5) = stp.*7;
%       marker(6) = marker(5) + contractionlength;
%
%         marker = 0;


         marker = generateEventMarkers(A(:,21),3,20000);
    end
    emgshift = mean(emgy(1:5000));
    emgy = emgy - emgshift;

    signallength = ceil(length(emgy)./2000)+1;

    [ecg] = ECG_Wave2(heartrate,amphigh,amplow,inv,signallength);


    [ecg] = ecg(1:emgx(end));


    emgy = emgy';
    signal = ecg + emgy;


    shift = sum(signal(1:5000))/5000;
    signal = signal-shift;

    plotx = emgx;

    csignal = emgy;
    nsignal = signal;
    ecgsignal = awgn(ecg,5);
end
```