Online Artifact Recognition and Removal in EEG Signals through Wavelet Transform and Independent Component Analysis

A Thesis submitted in fulfilment of the requirements for
the degree of Bachelor of Engineering
in
Electrical and Electronics Engineering

By
Greg Richardson
Student No. s226354

Supervisor:
Prof Friso De Boer
Mirjam Jonkman

Thesis coordinator:
Kamal Debnath
School of Engineering & Information Technology
Faculty of Engineering, Health, Science and the Environment
Charles Darwin University
Darwin

October 2014

CHARLES DARWIN UNIVERSITY
ABSTRACT

**Keywords:** Artifact, Brain Computer Interface (BCI), EEG, Wavelet, Independent Component Analysis (ICA), Feature Recognition, Online.

Electroencephalographic (EEG) signals are signals generated from electrical activity in the brain; these are recorded via a number of electrodes placed across the scalp of a patient. In a Brain Computer Interface (BCI) this technology is utilized and the output data is processed online. EEG signals are highly susceptible to artifact contamination from signals generated by other muscles in the body. These artifacts can often lie in the same frequency range as the brain signals being monitored and, as such, can make readings worthless, as such, these artifacts should be isolated and removed from the signal.

This method uses a combination of the FastICA algorithm and wavelet transforms. First the artifact is observed by identifying a sharp increase in amplitude in the signal (while negating drift). The artifact is then isolated using the FastICA algorithm. Using the relevant independent component (IC), the artifact is recognised by running wavelet transforms using a number of mother wavelets (relating to different potential artifacts). When identified the IC is replaced with a zero vector and the signal is regenerated by performing an inverse FastICA. Artifacts could also be considered features; as such, when a relevant waveform is identified, an output or external function may be triggered prior to removal.
# Table of Contents

List of symbols and abbreviations ........................................................................... 4
List of Figures ............................................................................................................... 5
1. Introduction ............................................................................................................. 6
   1.1. Electroencephalography ................................................................................. 6
   1.2. Artifact Contamination and BAEPs .............................................................. 6
   1.3. Brain Computer Interfaces (BCI) ................................................................. 7
   1.4. Spinal Cord Injuries and Feature Extraction ............................................... 7
2. Project Overview .................................................................................................... 8
3. Literature Review .................................................................................................. 9
4. Methods .................................................................................................................. 11
   4.1. Artifact Observation / Triggering ................................................................. 11
   4.2. Data Storage ................................................................................................ 11
   4.3. Independent Component Analysis (ICA) .................................................... 12
   4.4. Continuous Wavelet Transform (CWT) ....................................................... 21
   4.5. Artifact Identification .................................................................................. 21
   4.6. Artifact Removal ......................................................................................... 22
   4.7. Data Exportation ........................................................................................ 22
5. Simulation Results ................................................................................................ 24
   5.1. Offline Results .............................................................................................. 24
   5.2. Online Results ............................................................................................. 28
6. Recording of Data .................................................................................................. 31
   6.1. Electrode Placement .................................................................................... 31
   6.2. Test Procedure ............................................................................................ 32
   6.3. Data Synchronisation .................................................................................. 32
7. Wavelet generation ............................................................................................... 33
   7.1. Windowing and Extraction ......................................................................... 33
   7.2. Averaging ..................................................................................................... 33
   7.3. Filtering ......................................................................................................... 34
   7.4. Wavelet Generation .................................................................................... 36
8. Experimental System / Propagation Delay ............................................................ 38
   8.1. Delaybox ....................................................................................................... 38
9. Results .................................................................................................................... 41
10. Conclusion ............................................................................................................ 45
11. Future Development ........................................................................................... 46
Bibliography .............................................................................................................. 47
Appendix 1 ................................................................................................................. 49
List of symbols and abbreviations

BAEP - Brainstem auditory evoked potential
BCI – Brain-Computer Interface
CWT – Continuous Wavelet Transform
ECG – Electrocardiography
EEG – Electroencephalography
EMD - Empirical Mode Decomposition
EMG – Electromyography
EOG – Electrooculography
IC – Independent Component
ICA – Independent Component Analysis
MIDI - Musical Instrument Digital Interface
PCA – Principle Component Analysis
SCI – Spinal Cord Injury
List of Figures

All figures are by the author, Greg Richardson, unless otherwise stated.

Figure 1-1: An EEG signal with artifact contamination [1]
Figure 1.4-1: Incidence of SCI per capita vs state of residence [3]
Figure 4.1 1: Triggering System Diagram
Figure 4.2 1: Data Storage System Diagram
Figure 4.3 1: Original Components
Figure 4.3 2: Mixed Signals
Figure 4.3 3: Whitened Data
Figure 4.3 4: Whitening Block
Figure 4.3 5: Projected Density, ICs
Figure 4.3 6: Projected Density, Whitened Mixed Signals
Figure 4.7 1: Artifact Identification Block
Figure 4.7 2: Online System Diagram
Figure 4.7 3: Online System Outputs
Figure 5.1 1: Offline Input (Clean, Contaminated)
Figure 5.1 2: Artifact Identification Through Wavelet Transform
Figure 6.1 1: EEG 10-20 System
Figure 6.1 2: Standard EOG placement
Figure 7.1 1: EEG readings before and during eye blink
Figure 7.1 2: The first 5 eye blink artifacts extracted from Channel 4
Figure 7.2 2: The average waveshape on channel 4
Figure 7.3 1: 50Hz Comb filter magnitude plot
Figure 7.3 2: Filtered eye blink artifact
Figure 7.4 1: Matlab Wavelet Designer showing adapted wavelet alongside averaged model
Figure 7.4 2: Adapted Eye Blink Wavelet Generated
Figure 8 1: Final Experimental System
Figure 8.1 1: Final Experimental System
Figure 8.1 2 Eye blink artifacts on different channels
Figure 9 1: System Input vs Output
Figure 9 2: Captured (windowed) Data vs Rectified Data (following IC removal)
Figure 9 3: Captured Input, Extracted Artifact and Output on Channel 3
Figure 9 4: Presence of Mains Hum on Channel 3 Before and After Cleansing
1. Introduction

Figure 1-1: An EEG signal with artifact contamination [1]

1.1. Electroencephalography
Electroencephalographic (EEG) signals are signals generated from electrical activity in the brain; these are recorded via a number of electrodes placed across the scalp of a patient. A huge advantage of this technology compared to other methods of measuring brain activity (such as electrocorticography) is due to the extremely uninvasive nature of the process; the electrodes used to capture data are placed on the patient’s scalp rather than the surface of their brain. A problem that arises from this uninvasive approach, however, is the relatively low signal to noise ratio in the recording. The brain signals that are being measured are in the microvolt (µV) range and, as such, can be easily masked by background noise such as mains hum from power lines, electrostatic interference from neighbouring electrical devices or movement of sensor cables. Extracranial recordings such as that from an EEG are also highly susceptible to artifact contamination from other muscles in the body including ocular (EOG), cardiovascular (ECG) and skeletal (EMG) muscles.

1.2. Artifact Contamination and BAEPs
The previously mentioned artifact contamination can be a nuisance when trying to conduct EEG-based experiments relating to particular brain signals as they can often lie in the same frequency range as the signals being monitored and, as such, can make readings worthless.

An example of such a test would be an ‘Brainstem auditory evoked potential’-based hearing test where patients (including children) are required to sit motionless for many minutes at a time while numerous auditory stimuli are fed to the patient and their brains response recorded. The problem with this system is that people, particularly children, will get restless during this interval and most likely start moving or looking around, the resulting artifacts contaminate the data and potentially make a number of the readings unusable.

As such, if one were able to accurately and effectively extract and filter out these artifacts, the process would be a lot simpler and fewer samples would need to be taken (making the process faster).
1.3. Brain Computer Interfaces (BCI)
A Brain Computer Interface is, as the name suggests, a device to allow direct communication between a human brain and some external device. These devices usually take the form of a scalp-mounted EEG and some form of online system that allows for processing of real-time inputs. Such devices are often used by physically handicapped people to perform functions they would otherwise be unable to perform.

1.4. Spinal Cord Injuries and Feature Extraction
According to SpinalCure Australia, as many as 12,000 Australians live with some form of spinal cord injury [2]; these individuals suffer from either temporary or permanent limited muscular function in certain areas of their body, many of whom are quadriplegic and have limited to no function in their limbs and torso. For these individuals many tasks are either difficult or impossible to perform without constant assistance, one such task is transportation.

The Australian Institute for Health and Welfare reported in 2010 that the incidence of spinal cord injury per capita was higher in the Northern Territory than anywhere else in Australia, as such, this is a very relevant issue.
2. Project Overview
This project aims to deliver a functional online artifact recognition and extraction suite which, though tested with EEG data (both artificial and recorded), could function with any input data with minimal modification.

The intended method uses a combination of the FastICA algorithm and wavelet transforms; the former is used to isolate and eventually remove the artifact and the latter is to identify the artifact prior to removal. This process is outlined below:

- First the artifact is observed by identifying a sharp increase in amplitude in the signal (while negating drift).
- The artifact is then isolated using the FastICA algorithm.
- Using the relevant IC, the artifact is recognised by running wavelet transforms using a number of mother wavelets (relating to different potential artifacts).
- When identified the IC is replaced with a zero vector and the signal is regenerated by performing an inverse FastICA.
- Artifacts could also be considered features; as such, when a relevant waveform is identified, an output or external function may be triggered prior to removal.

Justification for this system is described in the following literature review.

The project was split into two parts; sections 4 and 5 deal with the initial software development along with offline/online testing with artificial data. Sections 6 to 9 deal with the collection and analysis of real world data along with calibrations made to the system to cater for said data.
3. Literature Review

In the realm of online biosignal analysis, particularly with EEG signals (in the case of brain-computer interfaces), artifact removal and feature extraction is a relatively new and constantly evolving field.

Artifact removal has been achieved through a variety of different methods; from very basic subtractive methods (using a reference signal such as EOG for eye blinks) to more involved methods, such as those utilizing Blind Source Separation (BSS) techniques such as Principle Component Analysis (PCA) [4] and Independent Component Analysis (ICA) [5].

Subtractive and regression-based artifact removal methods use a reference signal which accurately represents the artifact which is then scaled or otherwise manipulated before being removed from the source signals. For example, in the case of EEG readings, one would likely use an electrode around the eye of the patient to receive accurate EOG readings which would contain eye blink waveshapes; this signal is then manipulated and removed from the EEG signals (this process can be performed in either the time or frequency domain). However the problem with these techniques is that many, if not all, of these reference signals are not uncorrelated with the input signals [6]. Thus, by using these methods, relevant data would be removed from the sources along with the artifact. For example, since EOG readings are taken from the patients face (relatively close to the brain) much of the information read from the frontal EEG electrodes (Fp1, Fp2, F7, F8, etc) would also be present in the EOG reading. This information would be attenuated and potentially lost when using a subtraction/regression based artifact removal technique.

Lagerlund [4] uses a principle component analysis (PCA) based method for the removal of artifacts from an EEG signal (particularly eye-blink artifacts). This method separates a set of mixed signals into a set of linearly uncorrelated ‘principal components’ through finding orthogonal components with the largest possible variance from the remaining data, the component which most accurately represents the artifact is then removed and the data is reconstructed. Although this process is more effective than many earlier methods, it is limited in that it cannot completely separate artifacts from the remaining data due to how unlikely it is that the clean data and artifacts are spatially orthogonal. This discrepancy is particularly apparent when the artifact and brain signals have similar amplitudes. As such, this method would not be suitable for a general artifact-removal process. This method also requires electrodes to be placed around the patient’s body, making this a less comfortable procedure.

Soomro [5] demonstrates a method for removal of eye-blinks in EEG signals through use of Empirical Mode Decomposition (EMD) to extract an approximate shape of the eye-blink artifact which is then added to the original data as a source to ‘guide’ the fastICA algorithm to remove the artifact from each signal. Though this method showed strong results (correlation coefficient of 0.871 between clean and rectified signals), it was limited by computational complexity (claiming the method would not be suited for online applications) as well as in scope; the method only analysed one source for contamination (Fp1) and used only very simple, relatively large magnitude waveshapes as artifacts.

From the above, it is shown that the ICA process is quite proficient at extracting artifacts from contaminated sources, even when said artifacts are close to the amplitude of the source signals, as such, this process will be utilized in this project. However, a system that is not explored in any of the above methods is the identification of these artifacts.
One method to identify waveshapes, as shown in [7], is through the use of wavelet analysis with custom-defined wavelets. Using this, one can find where, and at what scale a waveform is present in a signal.

In this project, wavelets will be defined based on characteristic shapes and frequencies present in artifacts. These wavelets will be used to identify said artifact in a signal.
4. Methods

4.1. Artifact Observation / Triggering

All of the muscular artifacts we are looking at in this project have amplitudes much greater than the monitored signals (such as EEG signals), as such, one can deduce that, when a large increase in amplitude occurs, some form of artifact is present.

However, many signals, including those from an EEG are susceptible to large amounts of drift. By simply triggering whenever the amplitude of a certain signal is above a threshold, drift may cause the system to repeatedly trigger. To combat this; a dual average is used consisting of a short-time averager and a long-time averager that are differenced.

Since FIR averagers act as low-pass filters and this system uses two of such which are differenced, they can calibrated such that low-frequency components (such as drift) are negated.

The short averager quickly reacts to a sharp change in amplitude whereas the long averager takes longer to react. As such, when the difference between the averagers is quite high, an artifact is present.

![Triggering System Diagram](image)

Figure 4.1-1: Triggering System Diagram

4.2. Data Storage

The incoming stream of data (consisting of simulated EEG signals, artifacts and low-frequency drift signals) is stored in a 512-point, dynamically updating discrete shift register.

When an artifact is detected by the aforementioned triggering system, the output trigger is delayed by 256 points, this allows the whole artifact to be stored in the discrete shift register before the contents of the DSR are sampled and held. This system is shown below:
This newly stored data will then be processed using the ICA and wavelet analysis methods explored in the following sections.

4.3. **Independent Component Analysis (ICA)**

Independent Component Analysis is a popular and widely used method for removing artifacts from EEG readings. ICA decomposes a number of input signals into statistically independent components. One commonly-used implementation of ICA is the FastICA algorithm created by Aapo Hyvärinen.

The FastICA is split into three functions; centring of data, data whitening and component extraction. These functions are summarised below:

4.3.1. **Data Centring**

The FastICA process requires each signal to have zero mean (centred data). To accomplish this, the mean of each component of the input matrix $x$ is subtracted from the signal:

$$x - E[x] \rightarrow x$$

4.3.2. **Data Whitening**

The signal vectors must now be transformed such that each of the resulting components is uncorrelated to the others with a variance of one. This is known as whitening the data (as it transforms the input vector into a white noise vector).

The covariance matrix of a correlated dataset, such as $x$, is non-diagonal;

$$E(xx^T) = \Sigma$$

This covariance matrix can be estimated as:

$$\Sigma \approx \frac{XX^T}{n}$$

And, through eigenvalue decomposition:

$$\Sigma = EDE^T$$

Where the columns of $E$ are the eigenvectors of $\Sigma$, and $D$ is a diagonal matrix containing its eigenvalues.

Since $\Sigma$ is real, $E$ is orthogonal and, as such, $E^{-1} = E^T$:

$$E^T\Sigma E = D$$
By definition, the covariance matrix of an uncorrelated dataset ‘y’ is a diagonal matrix:

\[ E\{YY^T\} = D_1 \]

To make the components of ‘x’ uncorrelated, it must be transformed such that x=>y, as such:

\[ y = A \times X \]

\[ E\{YY^T\} = D = E\{AXA^TX^T\} \]

\[ E\{AXA^TX^T\} \approx \frac{AXA^TX^T}{n} = AA^T \Sigma \]

Thus:

\[ E^T \Sigma E = AA^T \Sigma \]

By pre-multiplying each side by \( \Sigma^T \)

\[ \Sigma^T E^T \Sigma E = \Sigma^T AA^T \Sigma \]

\[ E^T E = AA^T \]

\[ A = E^T \]

Thus, make the components of X uncorrelated, it must be pre-multiplied by the transpose of the eigenvectors of its covariance matrix \( \Sigma \). [8]

A whitened set of data has an identity covariance matrix;

\[ E\{\tilde{X}\tilde{X}^T\} = I \]

Thus, to whiten our existing set of data, a transformation must be determined such that D => I:

By definition:

\[ D^{-1}D = I \]

And that:

\[ D^{-1} = D^{-\frac{1}{2}} \times D^{-\frac{1}{2}} = D^{-\frac{1}{2}} \times I \times D^{-\frac{1}{2}} \]

Combining this with the above:

\[ D^{-\frac{1}{2}} \times I \times D^{-\frac{1}{2}} \times D = I \]

\[ D^{-\frac{1}{2}} \times D \times D^{-\frac{1}{2}} = I \]

Combining with the equation shown above, \( E^T \Sigma E = D \), we get:

\[ D^{-\frac{1}{2}} \times E^T \Sigma E \times D^{-\frac{1}{2}} = I \]
Now, define a new variable $Z = A_W \times X$, where $A_W$ is a transform matrix that whitens the data. $E(ZZ^T) = I$

By using the same steps as above, we arrive to the conclusion:

$$A_W = ED^{-1/2}E^T = A^T D^{-1/2}A$$

Therefore, the whitened data is given by:

$$ED^{-1/2}E^T x \rightarrow x$$

To demonstrate the effects of this process graphically, below are two randomly generated waveshapes plotted against each other such that each component of the first waveshape defines the y-coordinate of a point and each component of the second defines the x-coordinate:

![Figure 4.3-1: Original Components](image)

The two waveshapes are then combined together in two different configurations, $Z_1 = 1.5W_1 + 0.4W_2$ and $Z_2 = 2W_1 + 5W_2$; these signals were then plotted against each other in the same way as before:
Graphically this has rotated and distorted the ‘shape’ of the original signals.

The mixed dataset is then whitened and, as before, the two whitened components are plotted against each other:

As can be seen, the shape of the signals has returned to the shape of the original components (with some form of scaling) but is still rotated, as such, it can be concluded that to extract the individual components, this data must be transformed in such a way that it is rotated back to its original form, this is performed in the next section.

Below is a diagram of the whitening block in the Simulink program:
The code for the ‘Whitening’ block is below:

```matlab
function [Xwh, mu, invMat, whMat] = whiten(X)

mu = mean(X);  % Finds the mean of each signal of the data
X = bsxfun(@minus, X, mu);  % Subtracts the mean from each signal

covMat = cov(X);  % Determines the covariance matrix of X, E{X*X'}=EDE'
[V,D] = eig(covMat);  % Determines matrix of eigenvectors (V) and diagonal matrix of eigenvalues (D)
whMat = V*D^(-1/2)*V';  % Determines whitening matrix such that whMat*X = Whitened Data
invMat = inv(whMat);  % Determines dewhitenning matrix such that invMat*(Whitened Data) = X
Xwh = whMat*X';  % Whitened Data
end
```

4.3.3. **Component Extraction (Determining Mixing Matrices)**

The data is now uncorrelated, it must now be separated into its separate components. The purpose of this section to determine a transformation matrix that will transform the data in such a way that each row depicts an independent component (IC) rather than a complete signal.

To use independent component analysis we assume that all separate components have a non-Gaussian probability density function and that signals (consisting of mixtures of ICs) have approximately Gaussian probability density functions, using the same graphical example as in the previous section this can be shown as follows:
As can be seen, the distribution of points projected on each axis is certainly non-Gaussian (the points are nearly linearly distributed across the axis, whereas the whitened mixed signals have a significantly more Gaussian (bell-shaped) distribution:

As can be seen, the distribution of points projected on each axis is certainly non-Gaussian (the points are nearly linearly distributed across the axis, whereas the whitened mixed signals have a significantly more Gaussian (bell-shaped) distribution:
Thus the IC extraction algorithm must determine a transformation matrix that reduces the Gaussianity of these projections. To accomplish this, the mixing matrix vectors are randomly selected and then repeatedly mutated by running each component of the vector through the first and second derivative of some nonlinearity function $f(u)$.

Pseudocode for this process is shown below:

For an $N$-channel input $X$ of length $M$

With $N$ unique components (true for most practical cases, though it could also be any value less than $N$)

Function outputs mixing matrix $W$ which transforms the input $X$ into the ICs that make it up

$\epsilon$ is some stopping criterion.

g(u) and $g'(u)$ are the first and second derivative of some nonlinearity function $f(u)$

For $p = 1$ to $N$

$W_p = \text{random vector of length } N$

While change in $W_p > \epsilon$

$W_p^+ = \frac{1}{M} X g(W_p T X) - \frac{1}{M} g'(W_p T X) 1 W_p$

$W_p^+ = W_p^+ - \sum_{j=1}^{p-1} W_p T W_j$

$W_p = \frac{W_p^+}{||W_p^+||}$

End

End

$W = \begin{bmatrix} W_1 \\ \vdots \\ W_N \end{bmatrix}$

[9]

The algorithm used in the Simulink program differs slightly to the above in that it finds all of the mixing matrix components simultaneously (in parallel). However, the process is essentially the same.

Below is the block diagram for the ‘component extraction’ block used in this project:
The code for this block is as follows:
The gfpica function is a heavily abridged version of Aapo Hyvärinen’s original fpica function used in his fastICA algorithm [9]. The function itself is quite involved and will not be described in detail here, though it essentially follows the pseudocode presented above.

4.3.4. Implementation

In terms of this system, the input data is the 512-point, 5-channel output sampled from the discrete shift register described in section 4.2 (refer to above diagrams).

The output should consist of the independent components separated; one of these components should be the artifact with minimal contamination from the original simulated signals.

It should be noted that, in the original implementation of this software, EEG signals were simulated using channels of randomly generated Gaussian white noise. Since ICA separates
sources by maximising non-Gaussianity, these signals cannot be separated. In real world applications, the observed signals would likely not be Gaussian. [10]

4.4. Continuous Wavelet Transform (CWT)

The continuous wavelet transform essentially determines how ‘present’ a certain wavelet is in a signal at different points in time. The wavelet is then scaled and the process is repeated. The original wavelet is known as the mother wavelet, and the shifting is performed in accordance to a certain shifting function. The process is described mathematically below:

Let $x(t)$ be the function (or signal) that is being analysed, and $\psi(t)$ describe the mother wavelet. The wavelet transform at scale $a$ and time-shift $b$ is given by:

$$X_{\omega}(a, b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} x(t) \psi^* \left( \frac{t - b}{a} \right) dt$$

Where $*$ denotes the complex conjugate.

It should be noted that, for a set scaling factor $a$, the continuous wavelet transform is simply the convolution between $x(t)$ and $\psi \left( \frac{-t}{a} \right)$ scaled by a factor $\frac{1}{\sqrt{|a|}}$.

By selecting or, in the case of this project, designing a mother wavelet that is characteristic of a certain artifact, one can determine whether or not said artifact is present in a signal and, if so, when it occurs.

This technique is implemented as described in the following section.

4.5. Artifact Identification

At this point, the data containing the artifact-ridden signals has been stored and separated into its independent components through ICA.

Using the mixing matrix found through fastICA, it can be determined which IC was most prominent in the signals (which can be assumed to be the artifact component) and, on top of that, which channel it was most prominent in.

Knowing which channel in which the artifact was most defined also gives hints as to what sort of artifact it is. For example; if input data was from an EEG and the channel with the most artifact contamination was the Fp1 (at the front of the skull) it could be assumed that the artifact in question would likely be from an EOG signal such as an eye-blink or eye movement, whereas if the artifact was most prominent on the FT7 it may be caused by a jaw tensing or ears moving.

As such, depending on which track is most contaminated by the artifact, wavelet transforms with a number of mother wavelets (based on the likely artifact candidates) are performed on the separated component, if one of these transforms has a peak value over some set threshold, it is assumed that it is that artifact that was contaminating the data and an output is triggered accordingly. If none of the artifacts appear likely, nothing is triggered.
4.6. Artifact Removal

To remove the artifact, the IC containing said artifact is simply replaced by a zero vector and the original signal is reconstructed using the new set of ICs and the mixing matrix found previously.

4.7. Data Exportation
Before exportation, the entire ‘clean’ signals must be reconstructed; this consists of the input stream when no artifact is present and the rectified data following artifact removal being connected. To accomplish this, another function block was created, this is shown below:

The code for the ‘Data Store’ block is explained below:

```matlab
function yy = dStore(tr1, en1, xStr, xOut)
    global g_Buffer; %Assign a global variable to act as a buffer to store future values of yy
    ytemp = zeros(5,1); %Predefine output length as 5x1
    bufflen = length(g_Buffer(:,1)); %Determine length of the buffer
    if tr1 %if a new set of rectified data is incoming...
        g_Buffer = abs(xOut); %assign the incoming data to the buffer
    else
        g_Buffer = [g_Buffer(:,2:bufflen) zeros(5,1)]; %Otherwise, shift the buffered data forward one step
        if en1 %While there is still rectified data to output...
            ytemp(:) = g_Buffer(:,1)'; %set the output as the first (current) buffer location
        else
            ytemp(:) = abs(xStr); %otherwise, set the output as the time delayed input signal
        end
    end
    yy = ytemp; %output ytemp
end
```

**Figure 4.7-1: Data Storage Block**
5. Simulation Results

5.1. Offline Results

First, to test the effectiveness of the artifact extraction algorithm, a five-channel, two second EEG recording was imported into matlab and used as the ‘clean’ input. A predefined waveshape was then added to each track at varying scales, the shape was added at 0.4 seconds to simulate artifact contamination.

Below are the ‘Clean’ and ‘Artifact Contaminated’ data sets represented graphically, please note that each colour is a different EEG channel:

![Figure 5.1-1: Offline Input (Clean, Contaminated)](image)

The resulting signal was then fed through the same algorithm used in the online system.

It was first fed into a fastICA algorithm which separated the signal into its independent components; the component which defined the artifact was identified, along with the channel in which it was post present:
From here, the artifact identification and location process begins, to accomplish this; a single-scale wavelet transform is performed on the extracted artifact IC, the mother wavelet for this process was generated using the primary spectral components of the artifact waveshape which are then windowed. The peak location of the resulting function defines where the wavelet is most accurately represented in the data. As in the figure below, the transform identified the artifact at approximately t=0.4 seconds, which, since it was defined as such, is correct. Since the peak is quite pronounced (Y = 410) it would be correct to presume that the artifact was correctly identified.

```matlab
% Artifact Identification
nwlet = fliplr(wlet); % reverse wavelet vector (x(t) => x(-t))
wtIC = conv(ICart,nwlet); % converge the artifact IC with the reversed wavelet, this is the same as performing a unit-scale wavelet transform.

% Artifact Locating
 [~,conloc] = max(wtIC.^2); % Find peak of transformed data, this is the location, in points of the artefact + the length of the wavelet
conloc = conloc - length(wlet); % computes the position of the artifact
```
After this, the section of data is cut out from the IC and the data is recompiled through the inverse mixing matrix. Below are graphs of the signal in the time and frequency domain before and after rectification for one of the five signals.
Online Artifact Recognition and Removal

Original data, before contamination

Contaminated data, following addition of artifact

Rectified data, following artifact removal
As can be seen in the above graphs, the system mostly restores the data to its original state. The frequency response between 0-40 Hz (Delta – Early Gamma band) only varies significantly in the Theta band.

### 5.2. Online Results

The simulation was run with a looping EEG signal as input (the same as used in the offline tests above). Below is the layout of the Simulink design along with a number of the inputs/outputs visualised:

![Online System Diagram](image1)

As can be in scope 3, the correct artifact is identified in both cases. The data relevant to the first artifact (for one channel) is shown graphically below:

![Online System Outputs](image2)
As can be seen in the above frequency spectrums, the artifact removal process removes most of the components from the artifact contamination, keeping most defining components of the original signal (peaks around 13, 19, 35Hz). There is also an apparent change in amplitude for the overall signal; however, this is primarily due to the attenuation of higher frequency components such as the mains hum around 50Hz rather than the relevant low frequency components (Delta – Early Gamma band):
Though the effect of this IC removal on the desired data is minimal, it is still present. This does not only affect the region containing the artifact but also the surrounding regions within the window. The effect of this could be minimised by reducing the size of the window which is used to capture and store the artifact-ridden data, this way any important data lost due to slight cross-contamination of clean signal in the ‘Artifact’ will be localised only to the section originally containing the artifact. However, this would require knowledge of the artifacts that are going to be present and, as such, should be calibrated when the real-world input data is known along with averaged artifact shapes. The capturing and analysis of these artifact waveshapes will be explored in the following sections.
6. Recording of Data
The following sections will detail the movement from artificially-generated input signals to real-time real-world data.

The purpose of this section is to detail the EEG data collection process including electrode placement, the test procedure and data synchronisation:

6.1. Electrode Placement
For testing a number of positions were used in the placement of electrodes; two of these were used in all tests and the others varied with the subject. The following electrode locations relate to either the 10-20 EEG system or standard EOG placement, shown below:

![Figure 6.1-1: EEG 10-20 System [11]](image)
All tests contained the following two points:

- Fp1 – Close to source for eyeblinks, high level of contamination
- C5 – Temporal scalp location, high contamination from jaw clenching [13]

For each test subject, the following points were used interchangeably:

- Cz – Just to demonstrate propagation of signals between points on the scalp
- EVOG+/EVOG- - Used for the clean extraction of eye blink artifacts

Although the above points were selected, the system would work with any combination of electrode positions given that the triggering system be calibrated to detect artifacts on said positions.

These electrodes were sent through an amplifier and ADC before being read into Simulink and saved.

### 6.2. Test Procedure

The test procedure was quite simple; the patient would be asked to sit motionless in a dark room with the EEG equipment set up with electrode placement stated above. The recording process would then begin and the patient would be asked to perform a specific action, such as an eye blink, every time they heard a loud noise (caused by myself knocking on the window to the room).

Most subjects were not video recorded as to not make them feel uncomfortable. However, the last trial was recorded in 60fps using an LG G2 mobile phone.

### 6.3. Data Synchronisation

For most subjects, there was no form of data synchronisation used; since the artifacts were very distinct and visible, they were manually extracted for analysis (details of this in the below section).

For the final subject, a camera was set up and synchronised with EEG data using a small flashing light on a Freetronics Leostick (based on the Arduino Leonardo) which was triggered at the same time as the data recording began. By collating the video and EEG data one could determine which waveshapes corresponded to which extracranial muscle activity.
7. Wavelet generation
This section will detail the process involved in developing an artifact wavelet, in this case, an eye blink wavelet.

The process is split into the following components:

- Windowing and extraction
- Averaging
- Filtering
- Wavelet Generation

These steps will be explored in the following sections:

7.1. Windowing and Extraction
The first step involves determining at which points in time a subject was performing the action that causes the artifact in question. To do this, a video of the subject (myself) was recorded along with the EEG data which were synced through the use of a trigger signal (a blinking LED). When the subject performs the action on video, the associated EEG data can be extracted.

![Figure 7.1-1: EEG readings before and during eye blink](image)

7.2. Averaging
After extracting each set of EEG data relating to the action being investigated, each set of data should be shifted and windowed such that each stage of the artifact match up on the x axis. For eye blinks this was easy; each waveform was windowed such that the local maxima were centred.
After shifting and windowing, the waveshapes should be averaged such that you end up with an average waveshape for each channel:

7.3. Filtering
The next step involves processing these average waveshapes such that all, or most, of the EEG remnants and noise (such as mains hum) are removed, revealing the raw artifact waveshape.
To do this, one must filter the data. There are many ways to do this, with varying levels of complexity, but for the sake of this project, quite simple methods were utilized:

In Australia, the frequency of transmitted mains power is 50Hz. The electromagnetic fields emanating from power transmission lines can induce a current in neighbouring wires (such as the long cables used to connect the EEG electrodes to the amplifier in this experimental setup). As such, one can often, if not always, find some 50Hz (plus harmonics) noise in recorded electric signals. Since one cannot quantify the effect the mains hum has on the readings; the most effective method of mitigating the noise is by simply comb filtering 50Hz and its harmonics. The transfer function for such a filter takes the form:

$$H(z) = b \frac{1 - z^{-n}}{1 - az^{-n}}$$

Where $n$ is the order of the filter (which defines the number of notches, or the number of harmonics filtered), $b$ and $a$ are positive scalars which define the bandwidth of the filter. Below is the magnitude plot of a comb filter with $n=10$ and quality factor = 35, $f_0 = 50$Hz:

![Magnitude plot of a comb filter](image)

**Figure 7.3-1: 50Hz Comb filter magnitude plot**

Now that the 50Hz noise has been removed, all other noise can be filtered out however is seen fit. In the case of the eye blink, everything over 50Hz was hard filtered with a lowpass filter, Below is the average waveshape following filtration:
After this filtration there would likely still be some remnants of EEG signals in the waveshape, but the effect of these signals, again, is quite hard to quantify. However, since a large number of eye blink waveshapes were averaged, the majority of these EEG signals would have been attenuated.

7.4. Wavelet Generation
Now that an averaged model for the artifact has been developed, a wavelet must be designed to identify said artifact. To facilitate wavelet transform and analysis, the mother wavelet must satisfy a number of criteria including that it must have both zero mean and a square norm of one, as well as being windowed such that its amplitude begins and ends at 0. Rather than modifying the waveform to satisfy these conditions manually, the matlab Wavelet Toolbox was utilized; the artifact from the cleanest channel (in this case, channel 4) was input into the Wavelet Designer;
The following wavelet was generated:

This wavelet will be used to identify eye blinks in the final software. The same process would be used to develop wavelets for all other artifacts being monitored.
8. Experimental System / Propagation Delay
The following section will detail the system used to process the experimental data.

Most of the experimental system as a whole remained very similar to the simulation system used in 5.2. However, the following changes were made:

- Signal generation was replaced by the input signal alone
- The sampling frequency was reduced from 1024Hz to the medical standard of 512Hz (though a higher frequency could have easily been utilized while remaining within computational constraints)
- To compensate for the longer artifact duration, the capturing window was increased from 512 points to 768 points.
- To account for the propagation delay of signals between points on the scalp, each signal was delayed by a varying amount, allowing eye blink artifacts to line-up correctly. It should be noted that this was a temporary solution and that, if future developments were made; this delay subsystem would be automated, allowing for the accurate detection of any artifact.

Below is a diagram of the overall system:

![Figure 8-1: Final Experimental System](image)

As can be seen on the above diagram, the input signal is fed into a subsystem titled ‘delaybox’. The function of this box is described below:

8.1. Delaybox
It was determined in the experimentation conducted that there is a propagation delay of artifacts between electrodes, in other words; there is a time delay between when an artifact is detected at the front of the head and when it is detected at the back of the head.

This misalignment of artifacts causes a reduction in precision when removing contamination through the ICA process. As such, a solution should be made such that, depending on which artifact is incoming, the signals are shifted in such a way that … due to time constraints and limitations in the existing extraction/identification process, such a universal system could not be
implemented, instead, since eye-blinks have been the primary focus up until this point, delays were implemented to combat the propagation of eye blink signals between sources.

Below is a diagram displaying the contents of the ‘Delaybox’ subsystem:

![Diagram of the Delaybox subsystem](image)

Figure 8.1-1: Final Experimental System

The delay constants were found by comparing the averaged eye blink artifacts on each channel (averaging process described in section 7). This data was analysed through two different methods; originally the maxima for each channel was found and differenced to find the time delay between each channel. However, due to contamination from other signals, one could argue that the maxima were not necessarily a good comparison point; so instead, the eye blink wavelet developed above was convolved with each of the channels; the peak values were found and differenced to give the propagation delays.

Both methods yielded similar results which are shown above (delay of 8/512 seconds between channels 1 and 4, 21/512 seconds between 2 and 4, 8/512 seconds between 3 and 4).

In the below figure, one can see the average eye blink artifact waveforms from each channel following filtration and normalization.
The ‘max’ function in matlab was then used to find the location of the peaks for each waveform, which are then differenced to find the propagation delays.
9. Results

After implementing the new wavelets and calibrating the triggering subsystem, the software was run with data captured as per the procedure described in section 6 and the settings described in section 8.

The data from the fifth subject (myself) was fed into the system, below is a sample of this data along with the output of the system, note the time delay between input and output being around 1.5 seconds or 769 samples (the size of the window +1):

![Figure 9-1: System Input vs Output](chart)

To closer examine and assess the artifact removal system, we will extract the latest set of data from the discrete shift register and the ICA artifact removal block. These two sets of signals are shown below:
It should be noted that this is not the standard window that is used; normally the artifact would be positioned towards start of the window. This window (with the artifact centralised) was used to give the reader a better view of the captured artifact.

However, one can see quite clearly that, with the exception of a perceived change of amplitude, the artifact has been removed quite cleanly. The below figure shows the input, the extracted artifact and the output of one of the channels in this system, along with their frequency spectra:
Although there is no ‘clean’ reference, one can see that much of the low frequency contamination (between 0 and 10 Hz) has been attenuated.

As noted previously, we again experience a perceived change in amplitude in some of the channels (notably channels 3 and 4) caused by an increased contamination due to mains hum (caused by the 50Hz noise being out of phase between channels, causing the ICA algorithm to neglect it as a common waveshape). The effect of this can be seen in the following figure:
As can be seen, this method is equally effective at removing artifact contamination in real world data as in the artificial data being used in section 5.

The system does have its drawbacks however, it was determined that real world artifacts last significantly longer than those created for use in section 5. As such, there was a significantly longer delay between input and output making this system impractical for time-sensitive applications.
10. Conclusion
The goal of this project was to identify and remove artifacts from a number of input sources. The method chosen to accomplish this was a combination of Independent Component Analysis for artifact extraction and Wavelet Analysis for artifact identification.

As shown in the above sections, the program developed adequately accomplishes what it set out to do; it removes artifacts that are present in a number of signals (either simulated signals or captured EEG signals) with linear scaling of amplitude (and to some extent, time shifting) between channels, leaving very little contamination in the resulting reconstructed data. It also identifies, quite accurately, the extracted waveform based on a list of ‘known’ artifact shapes.

However, although the system can remove the artifacts quite precisely, it requires said artifacts to be windowed and processed in their entirety, meaning the minimum delay between the input and output is the length of the longest artifact (with some artifacts, such as eye blinks, lasting for over a second). This means that the system would not be suitable for time-sensitive application such as the driving of an electric wheelchair (it would be dangerous to wait over a second for their machine to stop in an emergency situation).

Though the system as a whole is not suitable for all the applications suggested in the introduction, a number of components could be extracted or built upon for use in other systems. Examples of this are listed in the following section.

Though this document shows a bias towards brain-computer interfacing applications, this technology could also be used in a number of other areas. For example, with slight modification one could use the technology to dynamically mix instruments in a live setting when one such instrument is playing too loudly, or remove cross-contamination in hexaphonic guitar pickups when one string is plucked too heavily, thus improving guitar-MIDI tracking.
11. Future Development

Though the program accomplishes all goals that were set, there are a number of additional features that could be added to expand its functionality. The first, as hinted above, would be to extract artifacts that have been time-delayed slightly between signals, this would account for delayed response in certain signals as is the case with many EEG artifacts [13].

Another problem (that was noted in section 5.2) surrounds the fact that the ICA algorithm can extract a maximum number of independent components equal to the number of input signals, as such, there is often some cross-contamination between the ‘artifact’ IC and sections of desired clean data, this data is then lost when the artifact IC is removed. This problem could possibly be minimised by adding an additional data channel that consists of an approximated model of the artifact shape. This approach will have to be investigated for practicality, and processes for creating such an average model will need to be studied (one of which may be the Empirical Mode Decomposition (EMD) component of the Hilbert-Huang transform).

Another possible addition to this thesis would be in directly comparing artifact extraction methods quantatively. This would be done by recording a clean set of data, inducing an artifact (which has been identified and averaged in the method described above), removing said artifact using the method being investigated and comparing the output with the ‘true’ clean signal.

One modification that would be interesting to investigate is in the use of adaptive mixing matrices for component extraction. In the current system, to separate independent components a mixing matrix is generated through the FastICA algorithm. This code is run every time an artifact is detected which is very computationally intensive and requires the system to be windowed (to compare channels) thus generating a large delay between input and output. However, as stated previously in this document, the only component that is significantly distributed between the channels is the artifact itself (which is also scaled to some degree) therefore, the mixing matrix would be relatively similar between artifacts of the same nature. Taking this into consideration, one could develop a system with adaptive mixing matrices; after detecting the incoming artifact, a relevant matrix ‘template’ is imported and its values adapted based on the incoming signal. This would cut down delay time significantly and allow for much higher sampling rates.

As suggested above, another system that could be investigated is the pre-detection of artifacts (prior to removal); as shown in the results, the artifacts were clearly visible on a number of channels even before extraction, this would suggest that the artifacts could be detected without having to first remove them from the data. This would mean that a number of systems could be implemented prior to artifact removal including automated channel delaying or adaptation of mixing matrix, both of which would require knowledge of the incoming artifact.

A number of improvements could be made in the code to improve efficiency; one of such would be to move all the high-point calculations (such as convolutions and ICA mixing matrix determination) to the GPU. Again, these changes would need to be assessed for merit.
Bibliography


[Accessed 1 May 2014].

Appendix 1
Non-scaled outputs

Another set of rectified data, non-scaled