IMAGING TOOLS FOR HUMAN SUBDURAL ELECTRODE GRID DATA

by

Andrew Michael Bean

A Thesis Submitted to the Faculty of the

COLLEGE OF OPTICAL SCIENCES

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

2009

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of source is made. Requests for permission for extended quotations from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Andrew Bean

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

_____ 10/30/2009

Russell S. Witte
Professor of Optical Sciences

Date

ACKNOWLEDGEMENTS

I would like to acknowledge Dr. Russ Witte for his guidance. He patiently taught me many necessary technical and research skills. In a subtler manner though, he demonstrated the vision to quickly pinpoint the important questions in many aspects of new and original efforts. These lessons have hopefully prepared me to similarly master the problems I shall help solve.

DEDICATION

Along with the time and effort put herein, this paper is dedicated to my family. Dedication goes first to my wife Jill Sue for lovingly taking care of our life when demands were many. These included -- but were not limited to -- the care of the paper's second dedicatee: our beautiful baby girl Marietta Sue. She rounds out four generations of family for whom I will be forever grateful.

TABLE OF CONTENTS

TABLE OF CONTENTS 5

LIST OF FIGURES 9

LIST OF TABLES 12

ABSTRACT 13

CHAPTER 1 – BACKGROUND AND MOTIVATION 14

Epilepsy 14

Resection 14

Presurgical Assessment 15

Data Sets 16

Spatio-temporal Domain 18

Previous Work 19

CHAPTER 2 - METHODS 21

Overview 21

Initial Processing 23

Dimensions of data set 23

Filter 24

Determine Motion 28

Plot Processing Stages 32

Spectrogram *33*

Image Frame *38*

Spatial arrangement **38**

Values **40**

Spatial output **42**

Movie *44*

Time **44**

Outputs **46**

CHAPTER 3 - APPLICATIONS **49**

Overview *49*

A single epoch of patient 1 ..... *50*

Find all electrodes included in the feature **50**

Spectrogram demonstrates increasing frequency **53**

Quantify movement **55**

Vary dynamic range on a fixed frame **59**

Other epochs from patient 1 ..... *61*

Spatial similarity between all epochs **61**

Spatial shift between two epochs **62**

Patient 2 ..... *65*

Background **65**

Disjoint localizations of two frequency bands **66**

Interesting shape of high frequency spectrogram....68

CHAPTER 4 – CONCLUSION AND FUTURE WORK 71

APPENDIX A – TUTORIAL..... 74

Introduction 74

Convert to binary.....74

Make movie 75

Conclusion.....78

APPENDIX B – MATLAB CODE..... 79

APPENDIX C – MISCELLANEOUS SUPPORTING FEATURES 89

Convert to binary.....89

Motivation.....89

Effect on workflow.....90

Performance of conversion 90

Alternative approaches.....93

Algorithm.....93

Data structure for make movie functionality 95

Overview 95

file Structure.....95

process_data Structure 96

process_movie Structure.....97

60 Hz pre-filter 97

REFERENCES..... 101

LIST OF FIGURES

Figure 1-1: Photograph of Ad-Tech subdural electrode grid used at AHSC. 17
Figure 1-2: Example of electrode grid placement. 17
Figure 2-1: Example to illustrate the Hanning window shape. The rising frequencies are 10 Hz to 20 Hz, and the falling frequencies are 35 Hz to 50 Hz. 24
Figure 2-2: Demonstration of envelope (amplitude of analytic signal) using a modeled raw signal. 25
Figure 2-3: Matlab code for generating the ideal raw signal for envelope demonstration. 26
Figure 2-4: Example from patient data showing the accuracy of the analytic signal. 27
Figure 2-5: Overlay of 20 electrodes during motion. The outlier is dotted. 29
Figure 2-6: The comparison of a non-motion ROI with a motion ROI on two different electrodes. a) From a visual comparison of the time traces, it is clear that the electrodes are detecting a non-neural source during motion. b) The quantitative measure of cross-coherence gives a striking contrast between non-motion and motion. The mean across all frequencies is sufficient to conclude non-motion (0.15) or motion (0.92) for these example ROIs. 31
Figure 2-7: Comparison between non-motion and motion regions. 34
Figure 2-8: Time trace of model, s10and20, created in figure 2-7. 35
Figure 2-9: FFT of data model. This is time independent, so does not describe the times for each frequency spike. 36
Figure 2-10: Spectrogram showing the wideband frequency content that distracts attention from the features. 37
Figure 2-11: Spectrogram showing the time-frequency data as expected. 38
Figure 2-12: Examples of spatial layout in a 5 by 4 grid. a) A single electrode (7). b) Two neighboring electrodes (6 and 7). These neighbors would be evident from the time trace. c) These neighbors (6 and 11) would not be obvious from the time trace. The intuitive spatial layout helps in this case. d) The two electrodes (5 and 6) are consecutively numbered, but are spatially not neighbors. This illustrates a risk of not using a spatial layout. 40
Figure 2-13: A grid size of 8 by 8 demonstrating electrode number overlay. 41
Figure 2-14: Example data to illustrate the display's centroid. a) All four electrodes have the same value, so the centroid is the same distance from each. b) As seen by the electrode values (from bottom right, clockwise: 1, .555, .0118, .0154), the centroid is weighted toward the bottom right. c) Interpolation of the same data from b. It is skewed toward the bottom right. 43
Figure 2-15: Matlab code for creating a data model to illustrate movie. 45

Figure 2-16: Plots of data model for illustrating movie. a) Raw signal, before initial processing. b) Signal ready for movie, after the last stage of initial processing: the dynamic range. The envelope value at each time is plotted for that frame. 45
Figure 2-17: Timeframe of data model used to demonstrate the movie. The interpolation and movement would not be evident from the time traces. 46
Figure 2-18: The mean is taken over all frames and displayed as a single frame, shown here for the model data used in this section. The time text has been replaced by the ROI. A high dynamic range was necessary since the activity moves in the model data. 47
Figure 2-19: The display's centroid and interpolated maximum plots for the model data in figure 2-15. The model was created to move linearly. That property is reflected quantitatively in these data..... 48
Figure 3-1: Time trace clearly showing good brain signal on electrodes 14 and 15. 51
Figure 3-2: Timeline of evenly spaced movie frames to demonstrate the spatial extent..... 52
Figure 3-3: Select frames (not shown in timeline) to demonstrate: a) the expected distribution from the time trace; b) some additional contribution from neighboring electrodes; and c) only neighboring electrodes. 53
54
Figure 3-4: FFT of all 20 electrodes showing strong frequency content around 20 Hz on various electrodes. 54
Figure 3-5: Spectrogram of electrode 14. This shows that the frequency increases with time. 55
Figure 3-6: Each spatial dimension of the display centroid is shown on a behavioral time scale. Between approximately 138 seconds and 140 seconds the centroid's average position is shifted to a specific location. 57
Figure 3-7: The GUI's output plots are shown here for the X direction. This provides detail for the centroid time series in figure 3-6. 58
Figure 3-8: The dyanmic range of a fixed frame is incrementally increased. The spatial significance is that including neighboring electrodes also includes background electrodes. 60
Figure 3-9: The trade-off from increasing the dynamic range for 139.5 is far too much background for 139.37..... 60
Figure 3-10: The mean of all frames for the previous epoch. It is shown here for comparison to the other 6 epochs. 62
Figure 3-11: The mean frame from each of the other 6 epochs. The means all compare similarly to the previous epoch. There is motion noise in some epochs. 63
Figure 3-12: Another epoch (times 16452 – 16453.5) is compared to the previously used epoch (times 138.5 – 139.5), and appears to be shifted toward the right. a) Same epoch as in previous section, repeated here for comparison. Its median distance is 29. b) Another epoch with a shift to the right: median is 33. 64

Figure 3-13: Spectrograms from this epoch (12042 – 12050 shown) revealing various frequency content. a) Electrode 6 shows distinct and strong frequency content around 8 Hz. A dynamic range of 13 dB was used to suppress some of the background. b) Electrode 17 shows distinct but weaker frequency content around 25 Hz. A dynamic range of 25 was used. The maximum dB displayed is the same between both figures, so the relative strength is conveyed. 66
Figure 3-14: The same ROI was imaged in two frequency bands. a) The filter is centered on 8 Hz and occurs mostly around electrode 6. b) The filter is centered on 25 Hz and occurs mostly around electrode 17. The gray circles indicate bad electrodes. It is clear that electrode 11 might have detected seizure activity were it operating correctly. 67
Figure 3-15: Similar interesting looking frequency content exists on electrodes 1 (a) and 6 (b). 68
Figure 3-16: A characteristic frame of the movie produced by bandpass filtering only frequencies from roughly 50 to 100. This is similar to the spatial pattern seen around 8 Hz for this patient. 69
Figure A-1: Parameters for converting the sample text file to binary. 75
Figure A-2: Parameters for making the movie. 78
Figure C-1: The 60 Hz component is clearly visible in both the spectrum and the time trace. The notch filter removes the 60 Hz component. 98
Figure C-2: Spectrum of a 10 second signal before and after the 60 Hz pre-filter. 99
Figure C-3: A zoom into the 60 Hz region shows that the original signal is affected. 99

LIST OF TABLES

Table 2-1: Mapping of electrode number dimension to the two spatial dimensions. This has been described in the text as "unfolding". 40
Table 3-1: List of epochs for patient 1. The program uses the integer number of seconds. The comments are only the quality of the data for demonstration purposes..... 61
Table C-1: Comparison of loading one time sample at a time and one second at a time. 91
Table C-2: Comparison of loading many seconds at a time. 92

ABSTRACT

I developed imaging tools around a Matlab GUI. The tools improve seizure localization with oft-recorded human subdural electrode grid data sets. The tools provide data visualization in the time-frequency and spatio-temporal domains. Spectrograms are built from GUI parameters for visualizing the time-frequency domain. For the spatio-temporal domain, 2-dimensional images of the electrode grid change with time to form a movie. The initial processing includes a bandpass filter, 60 Hz prefilter, Hilbert transform envelope, detection of maximum due to motion noise, dB scaling, and dynamic range. Quantitative output from the spatio-temporal domain includes the coordinates of the display's centroid and the interpolated maximum. Applications with data sets from two patients demonstrate the spatial extent of excitation, propagation of excitation, time-frequency dynamics, and comparisons within and between patient data sets.

Chapter 1 – Background and Motivation

Epilepsy

Epilepsy is a neurological disorder characterized by seizures. Specifically, they must be epileptic seizures: “a transient occurrence of signs and/or symptoms due to abnormal excessive or synchronous neuronal activity in the brain.” (Fisher et al 2005). If the source of the seizures is localized to one or more specific areas, it is called focal epilepsy. Generalized seizures are distributed over the brain. Focal epilepsy is the concern of this project.

Resection

Treatment of epilepsy begins with antiepileptic drugs (AED). However, focal epilepsy is sometimes resistant to medication and requires resection. Resection is a surgical operation for removing the area of the brain responsible for the seizures (the epileptogenic zone). It is important not to remove any brain tissue that contributes to motor, language, or memory.

The epileptogenic zone is a theoretical concept. It is the “area of cortex that is indispensible for the generation of epileptic seizures”. The seizure onset zone is subtly different. It is the area of the cortex that actually generates the seizure. So, when postsurgical analysis determines that the epileptogenic zone is actually larger

than the seizure onset zone, seizures still occur. Nonetheless, the goal of invasive EEG is to localize the seizure onset zone (Rosenow and Lüders, 2001).

Presurgical Assessment

Presurgical assessment begins the process of localization. The following noninvasive imaging modalities are utilized: MRI (Magnetic Resonance Imaging), fMRI (functional Magnetic Resonance Imaging), SPECT (Single Photon Emission Computed Tomography), MEG (Magnetoencephalography), and scalp EEG (Electroencephalography). MRI is most commonly used to locate lesions. fMRI is employed, especially, for investigation of the eloquent cortex responsible for motor, language and memory. Ictal SPECT can be used to locate the seizure onset zone. MEG contributes similarly to scalp EEG, but without the adverse physical effects of the skull. Often video-EEG is the most useful because it allows observation of the patient's behavior throughout the course of many seizures.

All electrophysiological techniques detect electrical brain activity. This is physically appropriate for epileptic seizure localization. Subdural electrodes give better data than the other two electrophysiological techniques: scalp EEG and depth electrodes. Scalp EEG is not adequate in 25% of patients (Spencer et al 1998). It has insufficient temporal resolution at 10 ms and insufficient spatial resolution at 10 cm^2 (Plummer et al 2008). On the other hand, depth electrodes give too fine of resolution to give the proximity of the eloquent cortex (Nair et al 2007). So amongst

the three electrophysiological techniques, subdural electrodes are most commonly used.

The other non-invasive studies are often not accurate enough. They have poorer spatial and temporal resolution than subdural electrodes. fMRI and SPECT have a 3-8 second delay after task initiation (Bargalló 2008). They detect blood-flow changes that follow metabolism, so they have an inherent delay. For the purpose of localization, their results may be non-concordant or inconclusive.

All non-invasive studies contribute by providing important information for the placement of the subdural electrodes over the seizure onset zone. This information is necessary to guide invasive recordings with a clear hypothesis and specific questions.

The concern of the present project is sensitivity of data to brain signals. All other concerns aside (e.g., cost, risk of surgery, medical treatment), subdural electrode data is the most appropriate. "With regard to sensitivity, [invasive EEG] is still the gold standard in detecting seizure onset." (Rosenow and Lüders, 2001).

Data Sets

The subdural electrode data in this project are from invasive video-EEG monitoring at the University of Arizona Health Sciences Center. The electrodes (figure 1-1) are arranged in a rectangular grid array: 5 electrodes by 4 electrodes with 10 mm spacing (Ad-Tech Medical Instrument Corporation, Racine, WI). The sampling rate is 400 Hz. Each electrode is referenced to a common ground electrode

and is thereby called the referential montage. It is a complex data set with 2 spatial dimensions (folded into 1 electrode number dimension) and the time dimension.

Figure 1-1: Photograph of Ad-Tech subdural electrode grid used at AHSC.

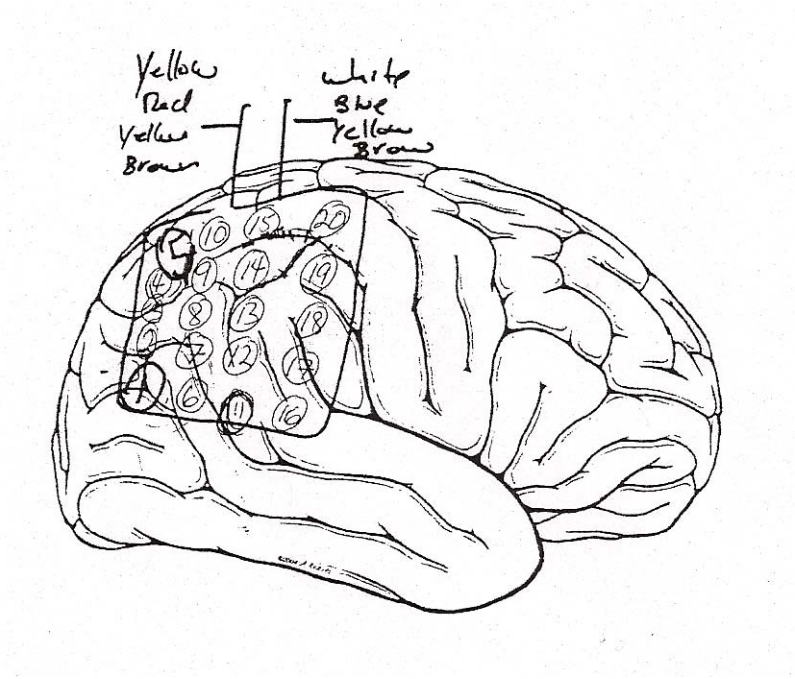

Figure 1-2: Example of electrode grid placement.

The goal of this project is to provide intuitive tools for seizure localization using this data. The native software (TWinLOOK 3.7 by Grass Technologies, West Warwick, RI) presents the data as time traces with some frequency filtering. This project uses the same data, but looks also at the spatio-temporal domain. There is also consideration of the time-frequency domain in the spectrogram.

The data sets are invasive video-EEG. Patient monitoring videos are part of the data set in TWinLOOK. These provide invaluable information. Minimally, it allows behavioral observation during a seizure epoch. This helps confirm that the seizure occurred. It also helps determine the amount of patient motion. Access to the videos has not been implemented in the tools developed in this project. To view TWinLOOK's or other software's patient monitoring videos, the respective native software must be used.

Spatio-temporal Domain

The literature is rich with studies that demonstrate the spatio-temporal nature of brain signals. By providing a background depolarization to a selected cortical region, propagating waves may increase firing probability (Wu et al 2008). An increase in firing probability could be important to the strength of the seizure. Ursino and La Cara (2006) developed a model to simulate ECoG signals. To motivate this model, they list the following spatio-temporal modes of activity: oscillations, synchronism, waves or avalanches. These similar studies should guide the exploration of patient data.

Many of the studies, though, are not human and not epilepsy. The use of voltage sensitive dyes in rats gives excellent results (Wu et al 2008), but is not FDA approved for use in humans. Studies of propagation in the cat visual cortex contribute generally (Witte et al 2007; Arieli et al 1995), but only indirectly apply to human seizures. Some results concern human epilepsy, but utilize depth electrode pairs rather than subdural grid (Chervin et al, 1988). There is, therefore, a need to study the human subdural electrode grid data.

As discussed in the resection section above, the seizure onset zone to be removed is spatial. Time traces do not intuitively convey the relationship among the electrodes for the purpose of seizure localization. The relationships physically exist and are collected in the data. By considering only time traces, there is a loss of information. The resection alone is enough reason to have spatial enabled tools for making this process more intuitive.

Previous Work

Lilly and Cherry (1954) recorded movies of EEG data. They connected amplifiers and glow tubes directly to the electrode leads, and recorded video of the glow tubes. The EEG readings came from depth electrodes in the cerebral acoustic cortex of cats. They analyzed the leading and trailing edges of spatial “figures” for position, direction, time, and velocity.

Analog processing was limited. The above analog system imaged the potential difference between the mean potential of all electrodes and the potential

of each electrode. The potentials were instantaneous average potentials with a one second time constant. The introduction of digital processing was an enormous step forward with its ability to store and process the data (Swartz 1998). These digital capabilities were further utilized to image the spatio-temporal domain: a proof-of-concept script for processing and arranging the time traces of a pre-ictal region of interest into a movie with its original spatial layout (Meade et al, 2008). Further developing tools for visualizing human subdural electrode grid data in the spatio-temporal domain will enable improvement in resection planning and neurological research.

Chapter 2 - Methods

Overview

This project provides a suite of tools. They can be applied to any appropriate data set. The software platform is a Matlab GUI. This GUI, then, becomes familiar to the user for exploring the similarities and variations among different data sets. The movie, for example, is a template for visualizing the spatio-temporal domain across various data sets. This is a more versatile and powerful approach than one-off analysis for each data set.

The goal for the suite of tools is intuitive spatio-temporal imaging and data visualization. The tools are tailored for subdural electrode data from human epileptic seizures. As covered in the introduction, the analyses of these data sets will fill a gap in the spatio-temporal analysis of brain signals. The data sets are recorded with each chronic implantation of subdural electrodes for resection presurgical evaluation, so the tools have the potential of being widely used.

The focus of this project is not, therefore, particular neurological results. Showing some applications is important; so this is accomplished in Chapter 3. The focus, rather, is on generality in the design of the tools. The hope is that in the hands of neurologists, the tools will enable new spatio-temporal neurological results. In such a way, the results will be simpler to achieve and repeat. Discovering new features through data mining will become a task for the neurologist. The tools are

intuitively integrated so that switching amongst them will make data mining more streamlined.

The benefits will be to both patient care and research. For patient care, the resections will only become more accurate with intuitive and capable spatial tools. For research, these same tools can be applied in a repeatable manner to existing data sets. The hope is that these tools will motivate and enable new discoveries; perhaps similar to other spatio-temporal discoveries in different types of neurological data

This chapter will describe the project. First, the initial processing of the data will be covered. This will illustrate the approach for making use of the complex raw data. After the description of the initial processing, the spectrogram will be explained. The spectrogram informs the choice of initial processing parameters, and is utilized before the image formation. Next, the 2 dimensional formation of a single image frame will be outlined. Lastly, there is a section on the inclusion of time to combine the images into a movie. Data models were created for each of these sections to help emphasize the important points.

Initial Processing

Dimensions of data set

As mentioned in Chapter 1, the data physically exists in 3 dimensions: 1 temporal and 2 spatial. However, it is stored and viewed digitally as only 2 dimensions: 1 temporal and 1 electrode numbering. The columnar spatial dimension is concatenated together to form just one dimension: electrode number.

The dimensions are used in both orders depending on the need. As concerns the file input/output, it is best to order the dimensions of the Matlab matrix as time (number of rows) by electrode number (number of columns). The reason is that files are read and written serially left to right then top to bottom. In order for the file to be written sequentially, a single time sample is written left to right; and each consecutive time sample is written downward. Viewing the data as a time trace has electrodes listed vertically and time progressing horizontally. So, when viewing the data, the dimensions are ordered electrode number by time.

The initial processing begins with raw data. It is necessary to convert the data into the expected binary files (see appendix C). The dimensions of the binary file match the dimensions of the raw data file. This is an operation that is performed only once for each data set. Before beginning each session with the data set, then, the binary file's path is simply loaded. This also loads the data set's metadata; for example, sampling rate, electrode spacing, etc. With these loaded, the initial processing can begin on the raw data.

The first parameter to define is the region of interest (ROI). This is a time region. There is no defined spatial region of interest since the entire spatial region is imaged. The ROI is simply defined as a begin time and an end time. The ROI is used for the movie and all plots.

Filter

Selecting only a certain range of frequencies is essential for processing brain signals. Therefore, a bandpass frequency filter is used. The band of frequencies to be passed is set on the GUI. The shape of the bandpass is fixed. A Hanning window is constructed with four frequency values. The window is extended and fixed at unity between the second and third frequencies. The first frequency specifies the frequency at which it increases from zero, and the fourth frequency specifies the frequency at which it returns to zero. The Hanning window is preferred over the rectangular window because of the latter's edge effects and subsequent ringing.

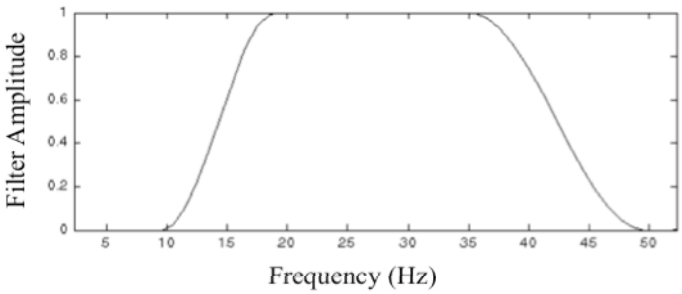

Figure 2-1: Example to illustrate the Hanning window shape. The rising frequencies are 10 Hz to 20 Hz, and the falling frequencies are 35 Hz to 50 Hz.

The envelope of the oscillating electrode signals is needed. The image needs to display the signal power instead of the oscillating signal. Displaying just the

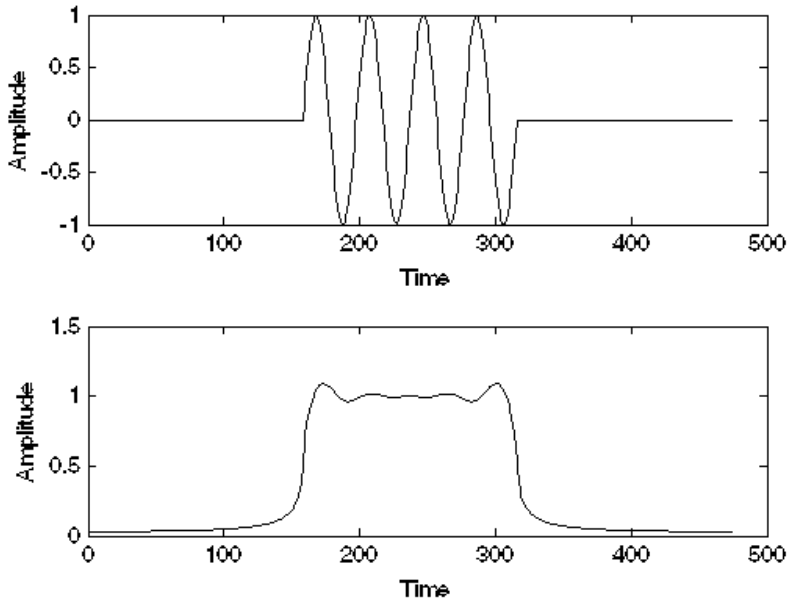

Figure 2-2: Demonstration of envelope (amplitude of analytic signal) using a modeled raw signal.

absolute value is not adequate because it does not accurately reflect the signal's power.

The envelope is attained using the Hilbert transform. It is useful for many purposes; e.g., state variables, analytic amplitude, analytic phase, polar plots, and temporal power spectral density. In the present case, it is being used to compute the analytic amplitude. This provides an appropriate envelope for brain dynamics (Freeman 2007). The wavelet transform has also been used. The Hilbert transform

is chosen largely for its relative simplicity over wavelets. In the analysis of phase synchronization, the two approaches were shown to be fundamentally equivalent (Le Van Quyen et al 2001).

There are multiple methods for computing the Hilbert transform of a digitized signal. For example, the Matlab function *hilbert()* returns the analytic signal. This is a much more simple approach if no filtering is performed. In our case, however, we can attain the analytic signal and perform the filter in one step. In the filter used (figure 2-1) all negative frequencies are outside the bandpass filter. The result of filtering out negative frequencies is the analytic signal (Bracewell 1978). For figure 2-2, an ideal raw signal was constructed. The negative frequencies were set to zero. Then, the inverse Fourier transform is shown. In this case, the pass-band filter was not applied for the purpose of illustrating just the analytic signal.

```
>> x = 0:1/(2*pi):8*pi;
>> s = sin(x);
>> a = zeros(1, 158*3);
>> a(158:158*2-1) = s;
>> subplot(2,1,1); plot(a);
>> A = fftshift(fft(a));
>> Aa = zeros(1,474);
>> Aa(1:474/2-1) = 0;
>> Aa(474/2:474) = 2*A(474/2:474);
>> aa = ifft(Aa);
>> subplot(2,1,2); plot(abs(aa));
```

Figure 2-3: Matlab code for generating the model raw signal for envelope demonstration.

In figure 2-4, patient data was used instead of simulated data. The filter parameters are 10 Hz to 15 Hz for the rising edge, and 21 Hz to 26 Hz for the falling edge. The figure illustrates that the 60 Hz noise is filtered out in the 138.0 to 138.4 region. It also illustrates that the slow frequencies are filtered out in the 139.5 to 140.0 region. Most importantly it shows that the analytic signal forms an excellent envelope of the filtered signal. This same data is used to demonstrate the applications in Chapter 3.

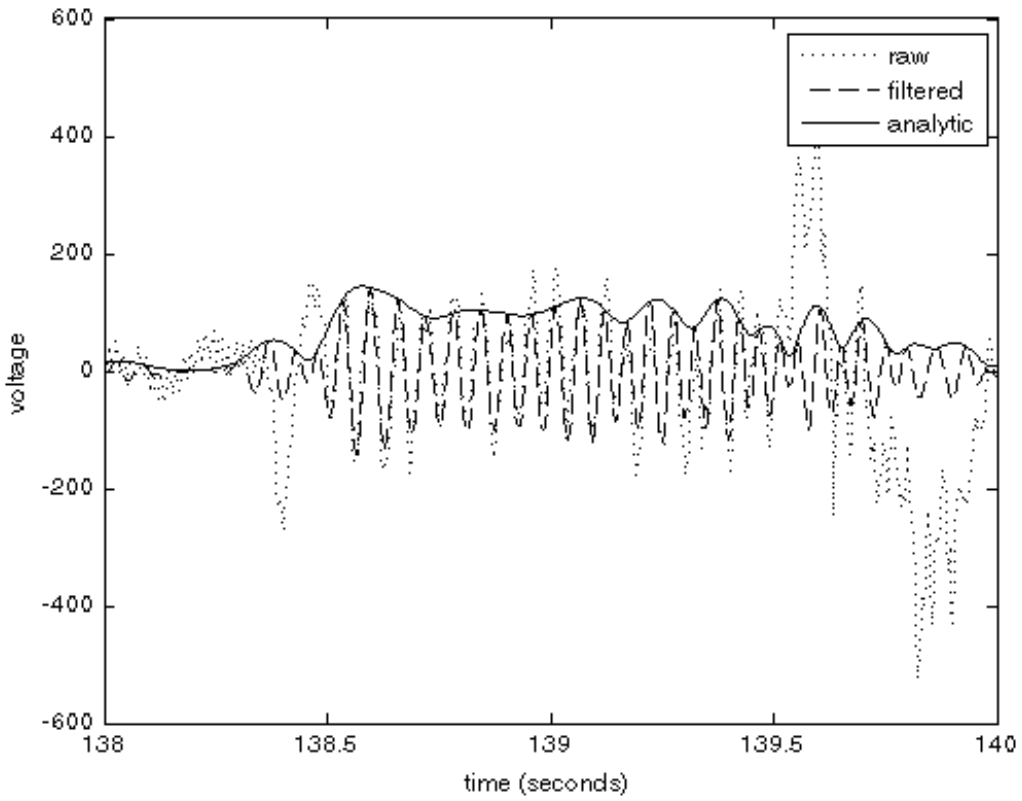

Figure 2-4: Example from patient data showing the accuracy of the envelope using the analytic signal.

The negative frequencies are important to consider for the filtering because, as discussed, the exclusion of them easily provides the analytic signal. For signal spectrums, on the other hand, negative frequencies are symmetric with the positive frequencies. This is a result of the real numbered signal. Therefore, the negative frequencies will not be typically shown in figures.

A 60 Hz noise prefilter has been implemented as well. This is applied independent of the bandpass filter and Hilbert transform. For example, it alone can be applied to the raw signal time traces as a noise reduction effort. The implementation chosen for this project is a standard digital notch filter. See Appendix C for more details.

The ability to load and save these parameters is important. This capability is provided on the GUI in a simple manner. A dropdown list contains the names of the built-in and custom filters. These filters are stored in a subdirectory (of the GUI files' root directory) called *ProcessParams*. The filter parameters are stored as variables in Matlab workspace files. Built-in parameter files, for example, are named after the common frequency bands: delta, theta, alpha, beta, gamma, epsilon.

Determine Motion

Motion noise causes high amplitude signals that are not brain activity. To determine if the region around a given time value is motion, the coherence amongst the electrode pairs is evaluated. If greater than 75% of the electrodes (e.g., 15 out of

20) are 80% coherent, then the region is determined to be motion. The region is centered on the time value and has a length equal to that of one tenth of the ROI.

The coherence is easily detected by visual inspection of the time trace. In figure 2-5, all but one electrode change together. The changes across electrodes are often in phase even when the amplitude varies. This can be seen when they are placed upon each other on the same plot. Computationally, though, they are looked at in pairs.

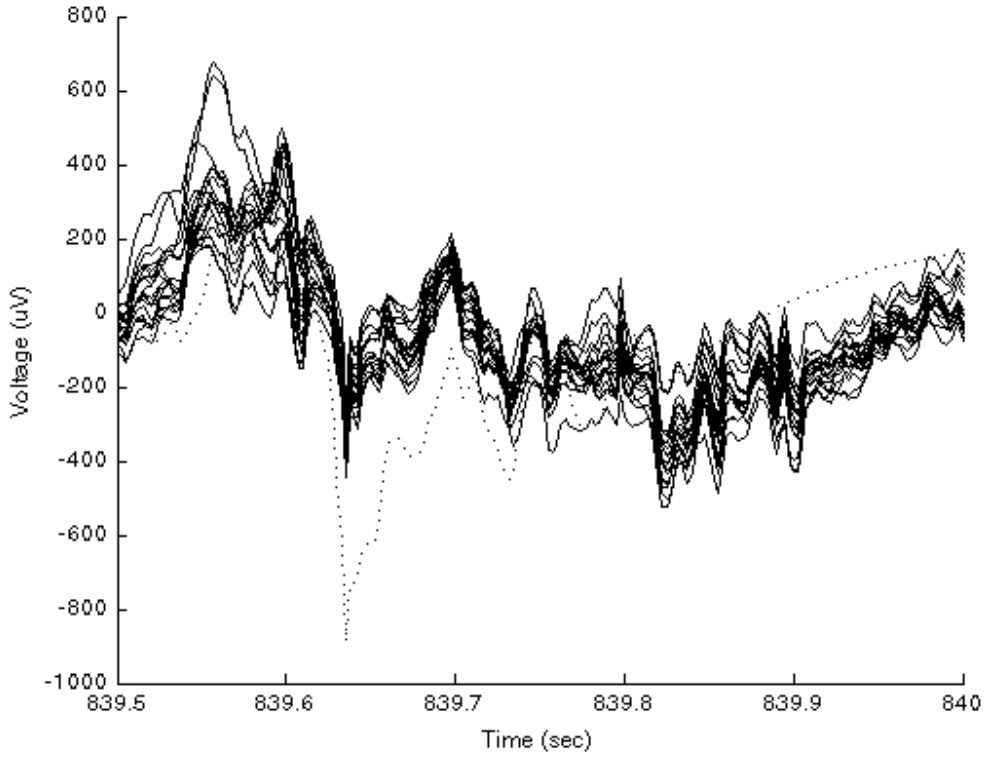

Figure 2-5: Overlay of 20 electrodes during motion. The outlier is dotted.

For N electrodes, the number of cross-coherence calculations is:

$$(N^2 - N) / 2$$

There are N^2 electrode pairs to start with. N of those pairs are an electrode with itself, which is always 1, so N can be subtracted. Then, this is divided by 2 because the operation is commutative. So, for 20 electrodes, there are 190 calculations. This large number of calculations limits the number of regions that can be tested. For a long ROI, therefore, longer regions will be tested to keep down the total number of calculations.

The results are stored in a matrix. The N by N matrix has ones on the diagonal, and is symmetric about the diagonal. All values are between 0 and 1. The first step is to use the power spectral densities to compute the vector:

This is a vector of coherence values by frequency. So, each pair of electrodes will have an entire vector of values. This is too much data and needs to be summarized. An average will suffice since, in the motion case, the coherence values are high across all frequencies (see figure 2-6). These values are returned by the custom function *calcmeancoh()* in the N by N matrix named *meancoh* (see Appendix B).

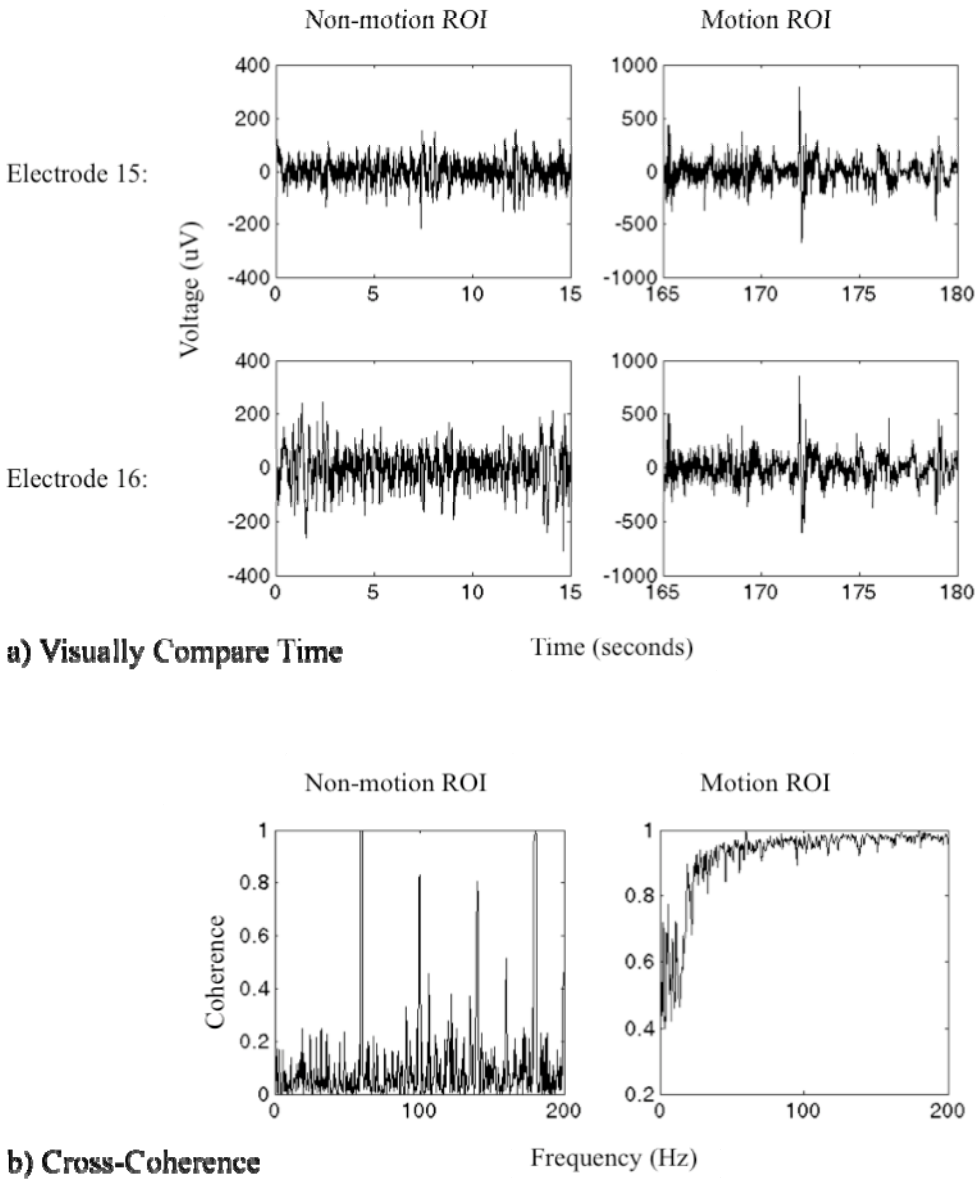

Figure 2-6: The comparison of a non-motion ROI with a motion ROI on two different electrodes. a) A visual comparison of the time traces shows that they match. This implies that the electrodes are detecting a non-neural source during motion. b) The quantitative measure of cross-coherence gives a striking contrast between non-motion and motion. The mean across all frequencies is sufficient to conclude non-motion (0.15) or motion (0.92) for these example ROIs.

Finally, to determine the percent coherent for one electrode with all other electrodes, the values of the N by N *meancoh* matrix are averaged for the electrode's

column or row. If the average is above 80% for 75% of the electrodes, then it is concluded as motion. The numbers were determined empirically and could be explored further (see Future Work).

Plot Processing Stages

The "Processing Stages" panel of the GUI gives the user the opportunity to visualize each stage. The choice of plots includes:

- Signal
- FFT
- Filtered
- Hilbert
- dB Scaled
- Dynamic Range

The functionality is straightforward. In addition to selecting which stage to plot, the only other choice is the electrode. There is a list with each electrode listed, and an additional choice for "all". When "all" is selected, all electrodes are place on one plot. The midline of each is separated evenly in the vertical direction by the maximum value across all electrodes. The x-axis is labeled with the appropriate value and unit (e.g., time or frequency). The y-axis is not labeled with the value and unit; but, rather, is labeled with the range of electrode numbers. Note that for each stage, including the raw signal, the 60 Hz pre-filter will be applied if it is selected.

Spectrogram

A spectrogram is a time-frequency image of the data. It is important with brain signals to be aware of the frequency content. However, the frequency content gives limited information without knowledge of the times at which it occurs. The spectrogram displays time along the x-axis as usual. The y-axis values are the possible frequencies. A colormap value is displayed at each pixel to represent the strength of each frequency at each time. It is an indispensible tool for visualizing the frequency content of a time series.

The spectrogram tool implemented on the GUI utilizes the Matlab function *spectrogram()*. No initial processing is applied to the data before the spectrogram is computed. The user selects the electrode for which the spectrogram will be displayed. Also, text boxes on the GUI allow control over the computation and display. The *window* variable can be entered directly by the user. The step parameter on the GUI allows the *noverlap* variable (number of overlapping segments) to be entered indirectly by the user. The overlap, therefore, is the step value subtracted from the window value. The frequency axis is fixed from 0 to 100 Hz.

For display, the user enters the dynamic range and the maximum value. The dynamic range allows the user to isolate a feature of interest from the background noise. The maximum value serves multiple purposes. First the maximum value allows the user to specify a value that saturates the display to isolate a feature of

interest from a large value (e.g., motion). Also, the maximum value can be set to be constant across all electrodes. In this way, if the user is switching between the spectrogram of different electrodes, the display maximum will be consistent. There is a button to set the maximum value of the display to the maximum PSD value over all electrodes. As opposed to saturating high values, some electrodes' spectrograms will not contain the maximum value.

To demonstrate the display, a model has been created. There are two features at different frequencies with different start and end times. The Matlab code in figure 2-7 was used to create the variable, and to display the spectrogram in the same way as the GUI:

```
>> samp = 400; %sampling rate
>> s10and20 = zeros(1, 6*samp);
>> s10and20(2*samp:4*samp) = sin(2*pi*10*(2:1/samp:4));
>> s10and20(3*samp:5*samp) = s10and20(3*samp:5*samp) +
sin(2*pi*20*(3:1/samp:5));
>> s10and20_fft = fftshift(abs(fft(s10and20)));
>> [S,F,T,P] = spectrogram(s10and20, 250, 125, 1:100, samp);
>> figure
>> imagesc(0:1/samp:6, F, 10*log10(abs(P)));
>> set(gca, 'ydir', 'norm');
>> colormap(hot);
>> Pmax = max(max(10*log10(abs(P))));
>> caxis([Pmax-15 Pmax]);
```

Figure 2-7: Matlab code for generating the model raw signal for the spectrogram

The data is kept simple and is not a realistic model of brain waves. The data includes two features at distinct frequencies: 10 Hz and 20 Hz. From the time traces (see figure 2-8), the frequency content is not evident.

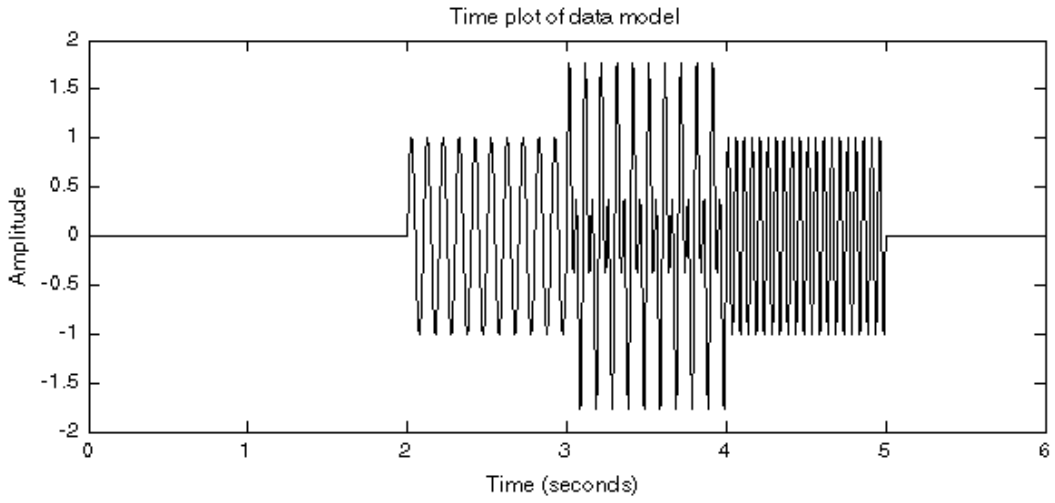

Figure 2-8: Time trace of model, s10and20, created in figure 2-7.

It is clear from figure 2-9 that the FFT does not contain information about the times at which each frequency occurs. For this, we need time-frequency analysis. The spectrogram is a time-frequency analysis tool. The features start and stop abruptly so there is wideband frequency content. Brain waves will be similar, except that the wideband frequency content is from background signals and noise. Nonetheless, it is clear that the frequency content is strongest at the desired frequencies (see figure 2-9).

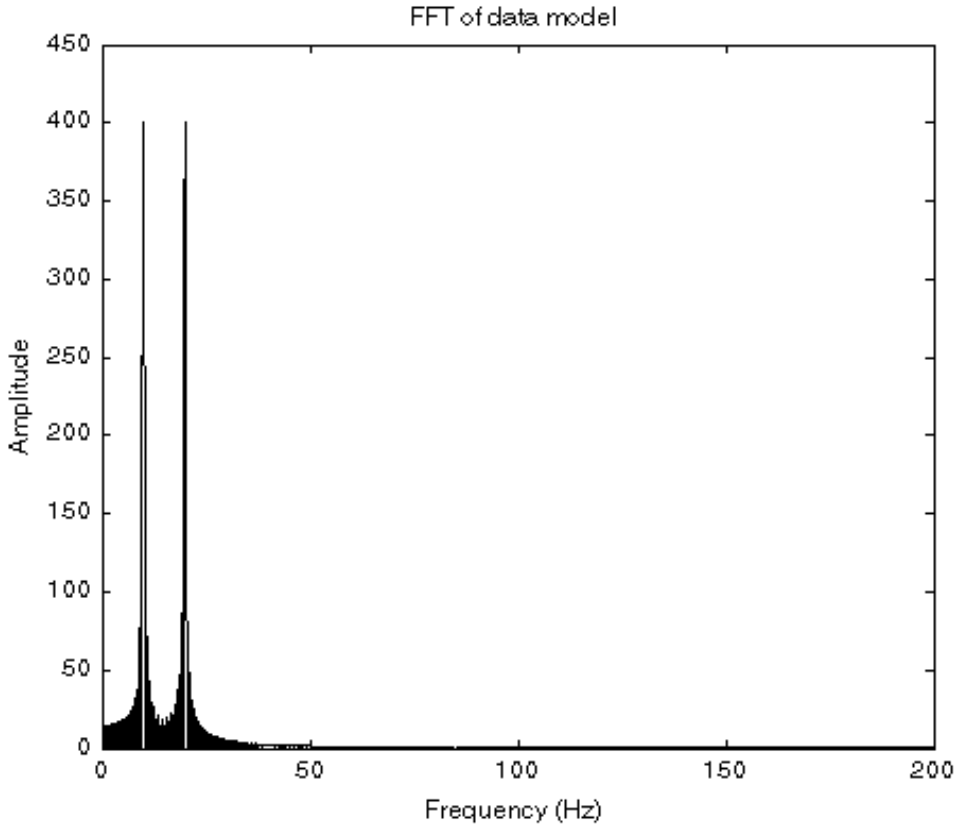

Figure 2-9: FFT of data model. This is time independent, so does not describe the times for each frequency spike.

Figure 2-10 shows the spectrogram of the model. This is the direct result of the code in figure 2-7 before the dynamic range is applied using the *caxis()* function. With the dynamic range applied the features are more evident (see figure 2-11). The value chosen, 15 dB, was found by trial and error to give the most aesthetic appeal. The goal is to get the best possible resolution in both frequency and time.

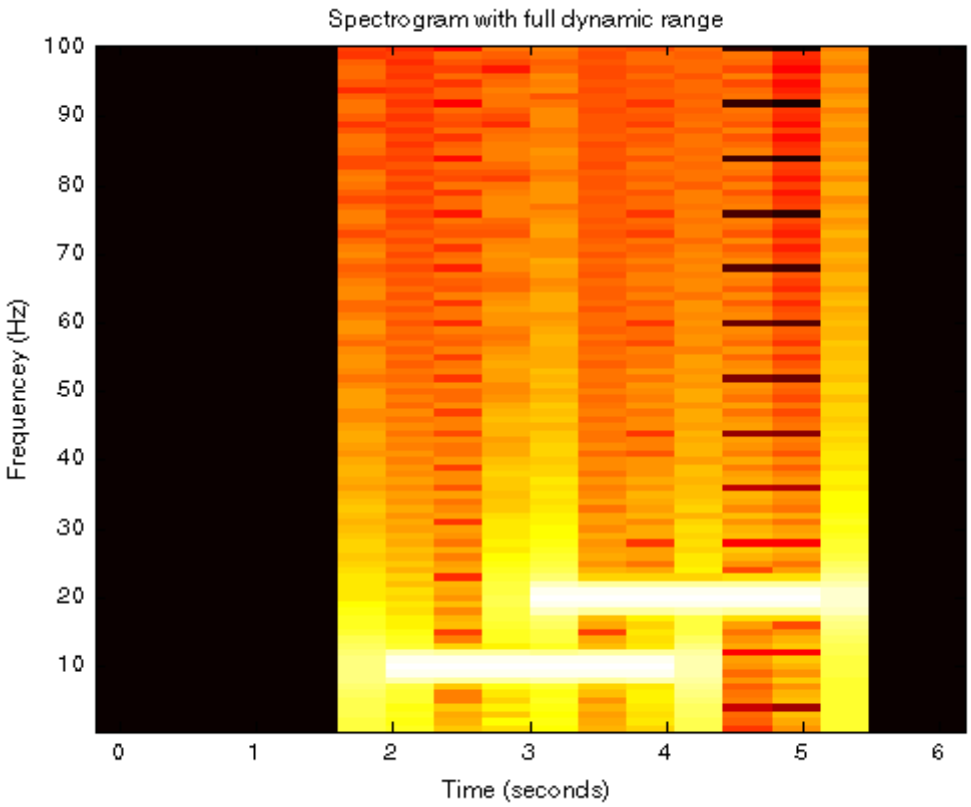

Figure 2-10: Spectrogram showing the wideband frequency content that distracts attention from the features.

For frequency resolution, a larger window is better because it samples the frequencies more completely. A larger window, though, poorly localizes changing frequencies. A smaller step will apply the window to more time frames, thereby giving better time resolution. So, each of these parameters are available for the user to attain the most appropriate image for the particular data.

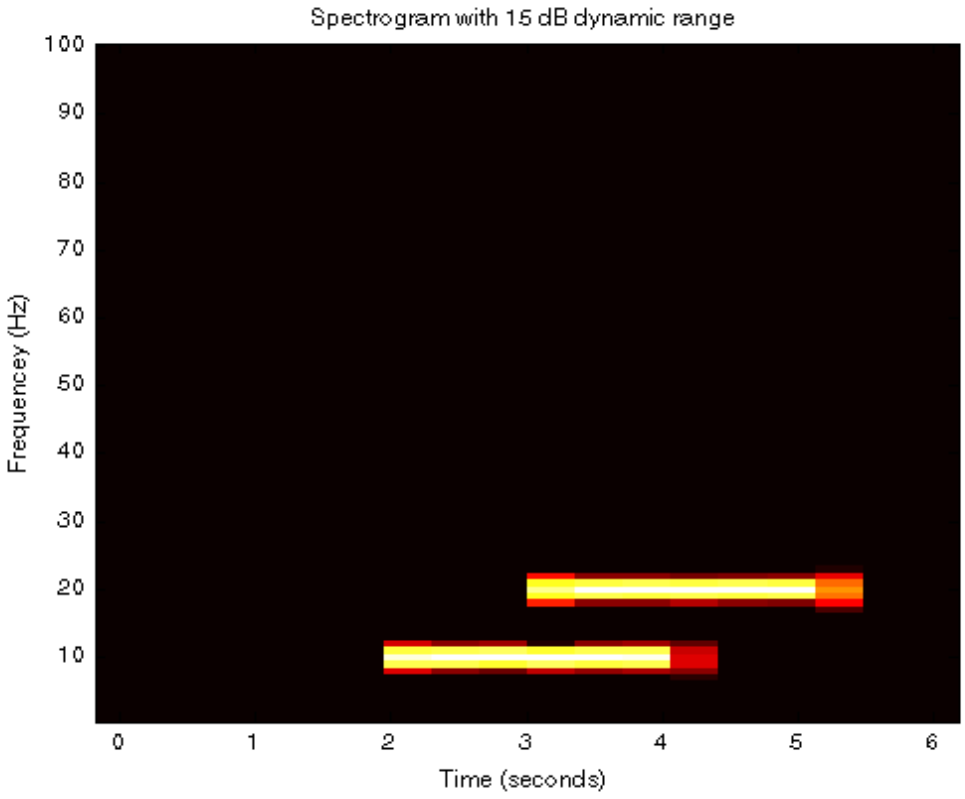

Figure 2-11: Spectrogram showing the time-frequency data as expected. The limited dynamic range isolates the features.

Image Frame

Spatial arrangement

The crux of this project is imaging the original spatial arrangement of the electrodes. There is a wealth of spatial information that is not intuitively revealed by the time traces. As discussed in the Initial Processing section, the electrode number dimension is the concatenation, or folding, of the two spatial dimensions. The aim of

the image frame, therefore, is to unfold the electrode number dimension into its original two spatial dimensions.

This section will discuss only a single frame of the movie. The formation of the entire movie is covered in the next section. This section will not consider realistic brain data. The data used here is modeled to demonstrate particular spatial concepts. In this section, *step_size* (discussed in more detail in the next section) is assumed to be 1. This defines the frame to be 1 time sample. Therefore, the dimensions of the data in this section are 1 by number of grid rows by number of grid columns; or, before unfolding, 1 by number of electrodes.

The loop to unfold the electrode number dimension is simple. It is a double loop with the outer index as the row dimension index and the inner index as the column dimension index. The unfolding assignment simply sets the value at the row_index and column_index of the matrix to the value at the electrode number given by:

(row_index – 1) * number_columns + column_index

So, with a 3 by 2 electrode grid, the mapping is given in table 1. For the convenience of the user, the GUI provides an option to overlay the electrode numbers. This is shown in figure 2-13, along with a common grid size.

row_index	column_index	electrode_number
1	1	1
1	2	2
1	3	3
2	2	4
2	1	5
2	3	6

Table 2-1: Mapping of electrode number dimension to the two spatial dimensions. This has been described in the text as "unfolding".

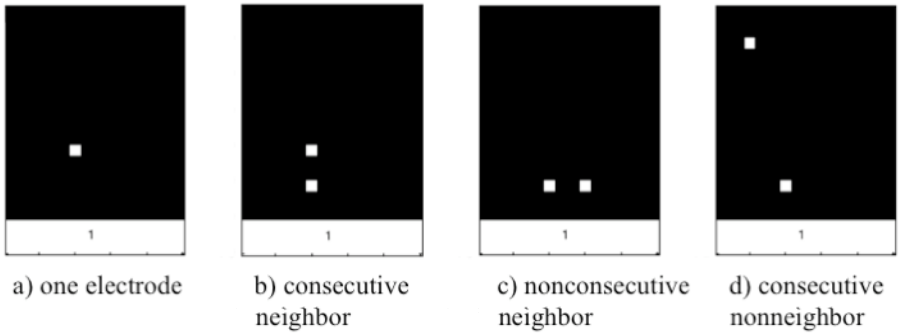

Figure 2-12: Examples of spatial layout in a 5 by 4 grid. a) A single electrode (7). b) Two neighboring electrodes (6 and 7). These neighbors would be evident from the time trace. c) These neighbors (6 and 11) would not be evident from the time trace. The intuitive spatial layout helps in this case. d) The two electrodes (5 and 6) are consecutively numbered, but are spatially not neighbors. This illustrates a risk of not using a spatial layout.

Values

The value of the display matrix element is incorporated into the image as a color value. The built-in Matlab colormap “hot” is used. If inhibition was also being determined and displayed, a custom colormap “hotcold” would be used (see Future

Work). Either way, the y-axis value of the time trace gets mapped to an intensity value on the display.

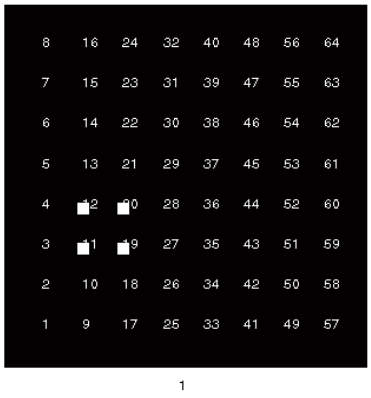

Figure 2-13: A grid size of 8 by 8 demonstrating electrode number overlay.

GUI parameters for maximum and dynamic range give precise control of how matrix values are mapped to color values. Entering a maximum allows the user to: set the maximum for the plot from the actual data, manually enter a lower maximum to saturate high values, or set the maximum for the plot according to the highest non-motion maximum from the data (see the Initial Processing section). The dynamic range, then, is the dB value below the maximum at which to display 0 as black. In that sense, it is a threshold operation. The dynamic range is applied after the operations of interpolation and frame step size. These parameters allow the user the control to visually extract the feature.

Interpolation is an important spatial operation to have available. A GUI checkbox controls whether it is applied or not. Cubic interpolation is used. The interpolation does not take into account any other physiologic information (see Future Work). For example, it is based only on the amplitude of the values, not on any delays in phase. Nonetheless, as demonstrated in figures throughout this paper, it greatly assists the user's ability to visualize the electrode data.

Spatial output

Thus far the spatial capabilities have been purely visual. There is also quantitative output. The two implemented outputs are: the display's centroid and interpolated maximum. Each of the outputs are a single x and y point for each frame. Further analysis can be performed on these outputs to lead to more spatio-temporal understanding (see the Applications chapter).

To exclude background signals and noise, the display centroid's calculation is applied after the dynamic range. This has the effect of a threshold to allow only the data of interest. In this way, the dynamic range has a significance beyond just the display aesthetics. It also makes this centroid more intuitive. For one, it is a "what you see is what you get" result: the placement matches the displayed signal. Also, the threshold value is applied automatically without an additional parameter to set. In general, a method must be used to exclude background and noise from the

centroid. Otherwise the centroid is biased toward all electrodes, which is the center of the image.

A centroid includes information from the entire extent of the activity. It will be weighted toward a higher value electrode. An example of this can be seen between figures 2-14a and 2-14b. In figure 2-14b, the bottom right electrode is given a higher value than the other electrodes, and thus the centroid is closer. The interpolated maximum (not shown separately) is a more simple measure. It is simply the point with the highest value. It often differs from the display's centroid since it does not take into account the spatial extent of the activity. The interpolated maximum may be more visually appealing because it may be simpler to compare to the image. See the next section for time plots of these values.

Figure 2-14: Example data to illustrate the display's centroid. a) All four electrodes have the same value, so the centroid is the same distance from each. b) As seen by the electrode values (from bottom right, clockwise: 1, .555, .0118, .0154), the centroid is weighted toward the bottom right. c) Interpolation of the same data from b. It is skewed toward the bottom right.

The capability exists on the GUI to image a single frame within the ROI. This is useful for quickly testing a set of initial processing parameters, or for outputting a particularly interesting frame. More commonly, however, the frames of a ROI are generated sequentially to form a movie. This is the topic of the next section.

Movie

Time

In this section, the time dimension will be added. In the previous section, fixing time developed the spatial details. That procedure is followed for each consecutive time in the movie. This is accomplished by the custom function *makemovie()*. The initial processing is applied so that each time sample has a value. The movie uses the same data as the processing stage plot titled "Dynamic Range".

A model has been developed to illustrate the movie. Figure 2-15 shows the code used. A random component has been added to the model to simulate background neural signals. Figure 2-16 shows the raw time traces and the result of the initial processing. Finally, figure 2-17 shows selected frames forming a timeline of the movie. The data is much more intuitive when arranged spatially.

```
>> move_across = 20*(rand(800,20)-.5);
>> for i=1:200
move_across(i+200,2) = move_across(i+200,2) + (200 - i) *
sin(2*pi*(10/400)*i);
move_across(i+200,7) = move_across(i+200,7) + i *
sin(2*pi*(10/400)*i);
```

```
move_across(i+400,7) = move_across(i+400,7) + (200 - i) *
sin(2*pi*(10/400)*i);
move_across(i+400,12) = move_across(i+400,12) + i *
sin(2*pi*(10/400)*i);
end
>> fid_out = fopen('../work/seizures/sim-moveacross.bin', 'wb');
>> fwrite(fid_out, move_across', 'int16');
>> fclose(fid_out);
```

Figure 2-15: Matlab code for generating the model raw signal for the movie.

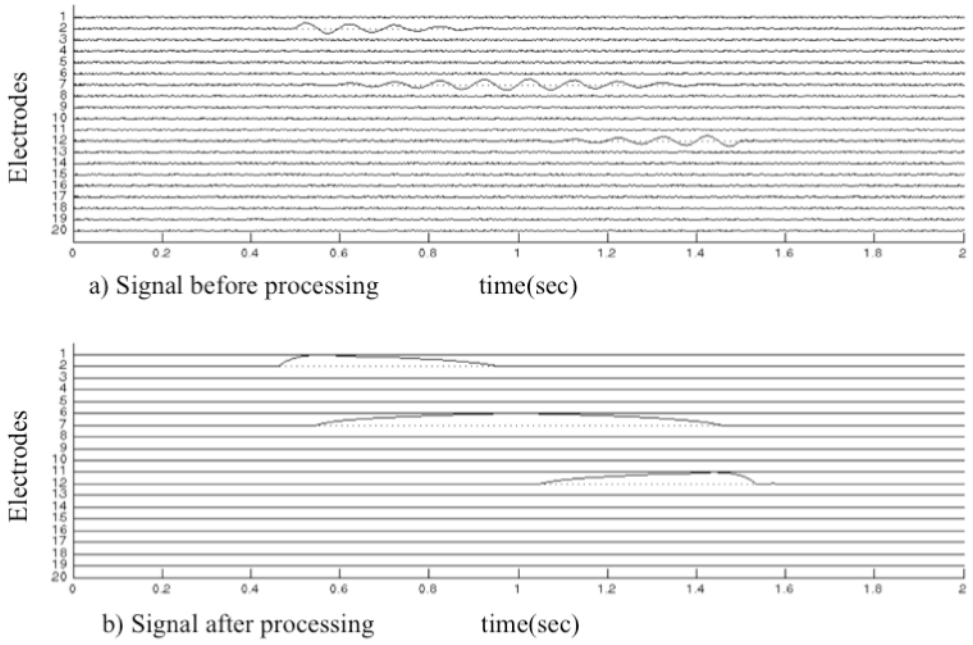

Figure 2-16: Plots of data model for illustrating movie. a) Raw signal, before initial processing. b) Signal ready for movie, after the last stage of initial processing: the dynamic range. The envelope value at each time is plotted for that frame.

The time dimension length can be shortened by setting the variable *step_size* to a number greater than 1. Before imaging the consecutive times, the values from *step_size* number of time samples are averaged. This is important for a long ROI. For

example, a 30 frames per second (fps) movie of 10 seconds of data at 400 Hz would take over 2 minutes to play. For exploring data, it might be better to lose some time resolution in favor of shorter playback. With a *step_size* of 10, a 2-minute movie would take 12 seconds to play. It might also be desired to control the number of frames. For example, the number of frames of the movie can be matched to the number of bins in the spectrogram.

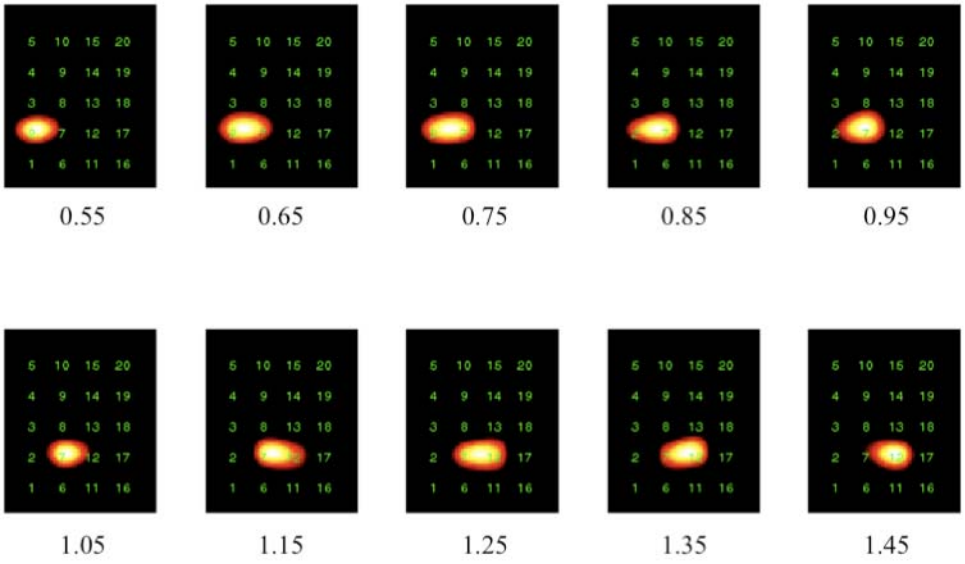

Figure 2-17: Timeframe of the data model used to demonstrate the movie. The interpolation and movement are not evident from the time traces.

Outputs

To enable playback, the most important output is a variable to store the movie itself. While the movie is rendered, each frame is stored. Then, when clicked, a button on the GUI passes the 3-dimensional matrix to the function *mplay*. This

program has the familiar controls for play/pause, fast forward/rewind, frame advance, etc. The matrix is also available for further analysis from the Matlab base workspace. There is also an option on the GUI for the user to display a "mean frame". This is simply the average of each pixel over the ROI. Figure 2-18 shows the mean frame for the model data used in this chapter.

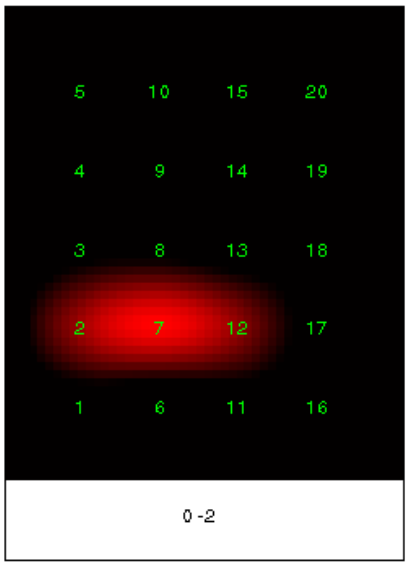

Figure 2-18: The mean is taken over all frames and displayed as a single frame, shown here for the model data used in this section. The time text has been replaced by the ROI. A high dynamic range was necessary since the activity moves in the model data.

Each individual frame (see previous section) computes the display's centroid and interpolated maximum. While rendering the movie, the respective coordinates are added to a matrix with the time dimension. This allows further analysis of the data. It also allows a plot of each of the two time series (figure 2-19). For example, from this time plot it is easy to compute that the signal moves at 20 mm/sec. The

electrode spacing is used to set the y-axis values (see the Convert to Binary section in Appendix A). The velocity conclusion is correct according to the model data in figure 2-15. An application to the display's centroid of real data is in Chapter 3.

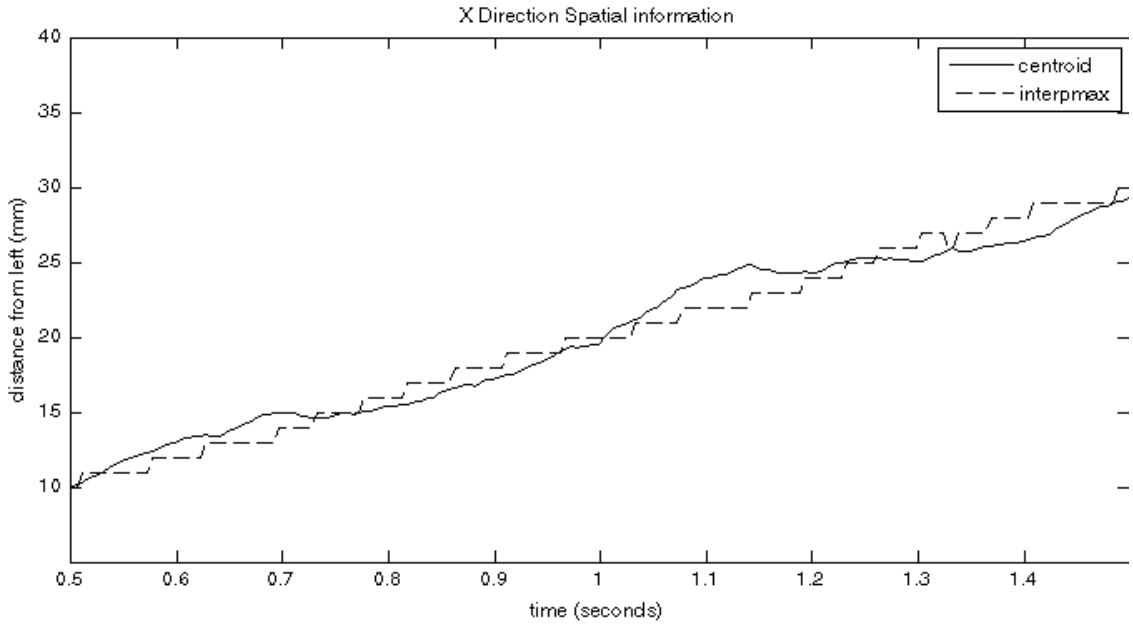

Figure 2-19: The display's centroid and interpolated maximum plots for the model data used in this chapter. The model was created to move linearly. That property is reflected quantitatively in these data.

Chapter 3 - Applications

Overview

The tools available from this project have been described. Modeled data has been created to demonstrate the progression of how the raw data is used. In this chapter the tools will be applied to various patient data sets to reveal some spatio-temporal and time-frequency properties.

The reader must be forewarned that the author is not a neuroscientist. The following applications do not provide any conclusions for resection planning, for research, or for any other purpose. The applications presented here simply use these real data sets to demonstrate two important points. First, the suspected spatio-temporal and time-frequency properties do exist in the data. Second, the tools developed here help analyze those properties. The next step is for trained neuroscientists to apply the tools along with their knowledge.

Data sets from two patients are used. The first section focuses on just one epoch of the first patient's data set. This epoch contains a particularly clean feature for the sake of demonstration. For example, there are less adverse affects from the motion artifacts in this chosen epoch. The second section introduces the application of the tools for data mining across many epochs of the same patient. The video monitoring shows similar motor behavior in each seizure. The video monitoring does not contribute beyond the time identification of each seizure. The third section

demonstrates generality by showing applications to the different patterns in a second patient's data set. Again, the video monitoring is consistent across all seizures of this patient. These applications sufficiently demonstrate the potential usefulness of the tools developed in this project.

A single epoch of patient 1

Find all electrodes included in the feature

This is a simple demonstration of the benefits of visualization in the spatial dimension. The raw voltage time traces can show the obvious electrodes. Through filtering, they will stand out even more. However, with visualization of the spatial dimension, the underlying signal's extent is more apparent and intuitive.

In figure 3-1, using only voltage versus time, electrodes 14 and 15 clearly stand out. It is not intuitive to establish the spatial extent. From this view, it seems conceivable that electrode 9 has a stronger signal. Still, the frequency content is not as clean as electrodes 14 and 15. It is also of smaller amplitude -- as may be other neighboring electrodes.

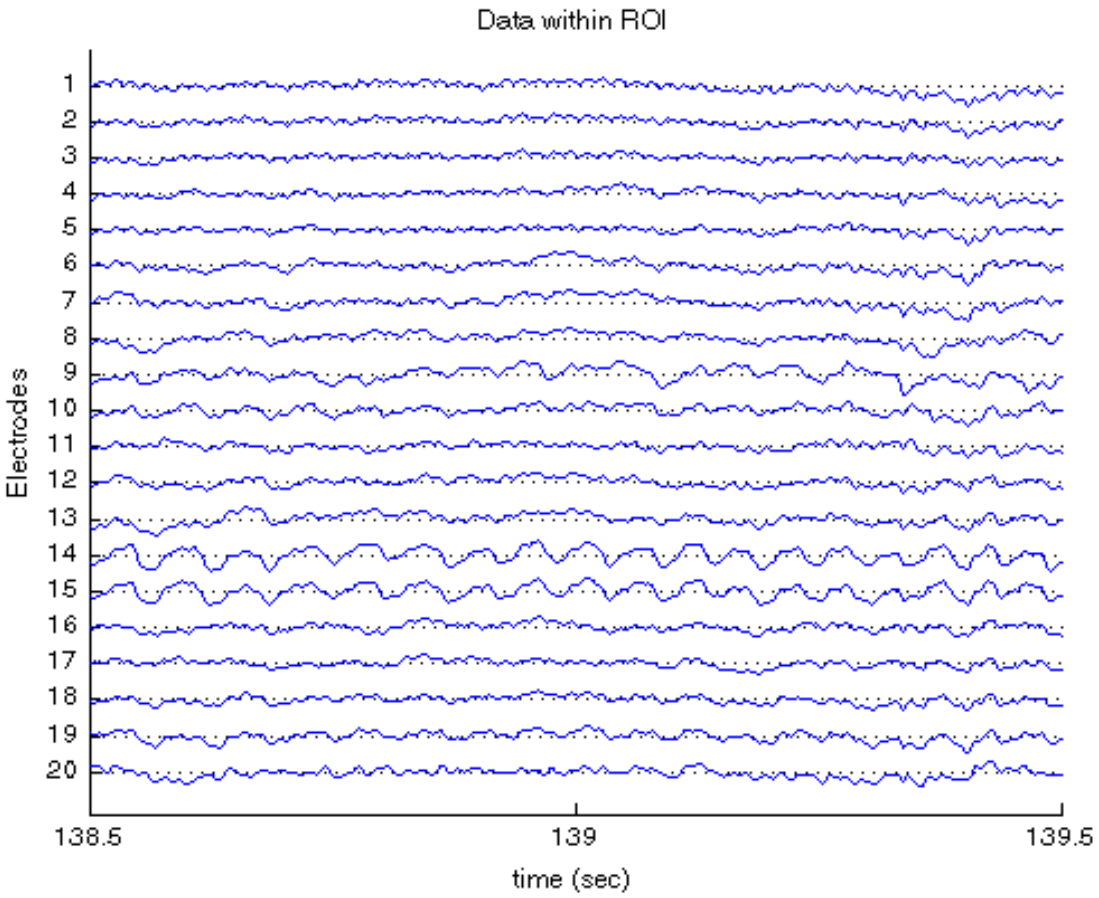

Figure 3-1: Time trace showing good brain signal on electrodes 14 and 15.

In figure 3-2, using the spatial dimension in each frame of the movie, the detection of lower amplitude electrodes becomes clear. The spatial extent is larger than just two electrodes. Also, the shape of the spatial extent changes spatially due to the lower amplitudes.

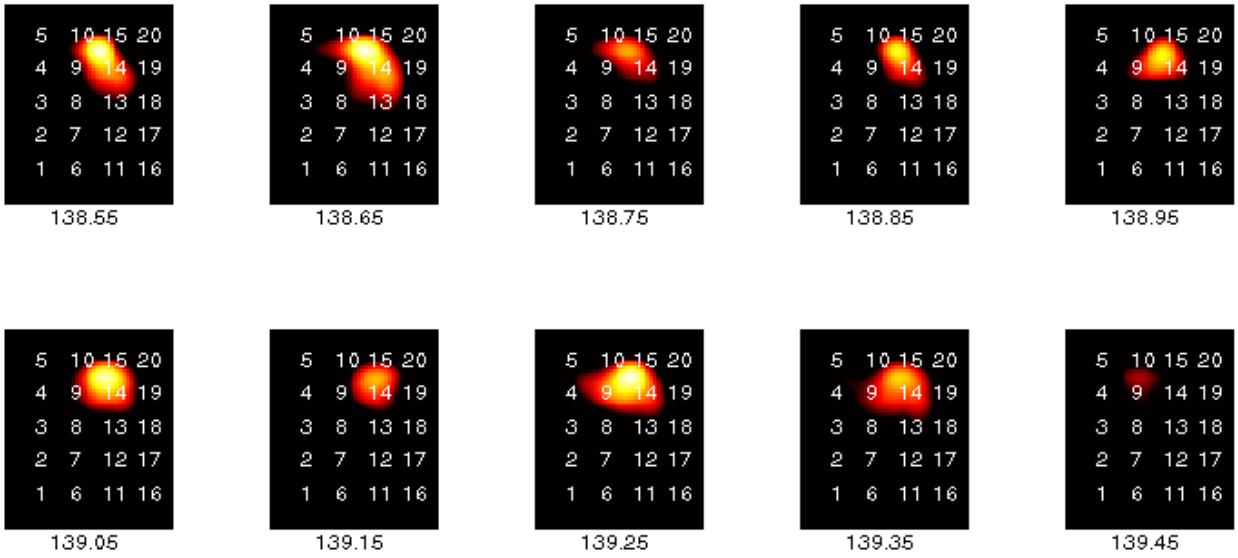

Figure 3-2: Timeline of evenly spaced movie frames to demonstrate the spatial extent.

The goal of figure 3-2 is to show an evenly spaced timeline. There are a few other select frames of interest. Figure 3-3a shows what might be expected from the time traces: the signal on 14 and 15 being approximately equal, and therefore the interpolated signal is equally between them. Figure 3-3b shows the signal interpolated across the space of four electrodes. This is also clear in many frames of figure 3-2. In figure 3-3c, the signal is stronger on 9 and 10 than on 14 and 15. Therefore, it is interpolated between 9 and 10. This particular time would have been especially difficult to pick out from the time trace.

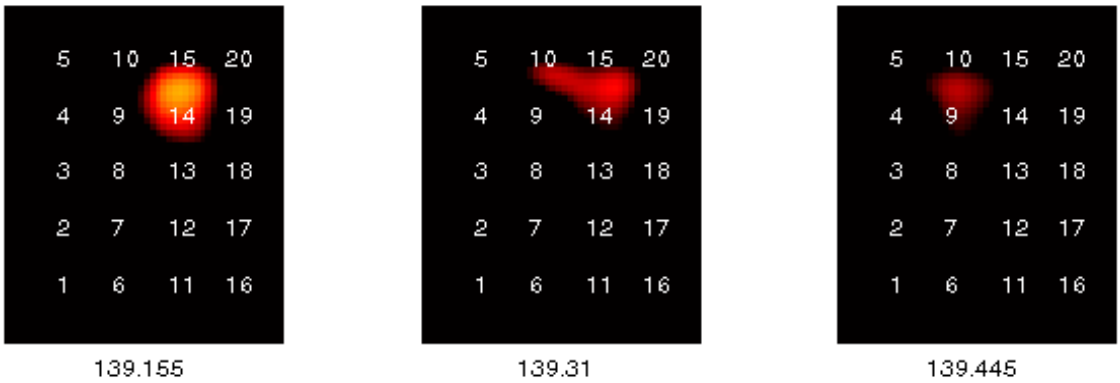

Figure 3-3: Select frames (not shown in timeline) to demonstrate: a) the expected distribution from the time trace; b) some additional contribution from neighboring electrodes; and c) only neighboring electrodes.

This application of the tools shows the benefit of utilizing the spatial dimension. With the electrodes arranged linearly, spatial properties are not clear. The signal strength can be visualized by adding the dimension of display intensity. The horizontal and vertical axes, then, show the original two dimensional arrangement of the electrode grid array. Thus, there is a more intuitive appeal.

Spectrogram demonstrates increasing frequency

The above application of the GUI requires proper filter frequencies. This application uses the same data to demonstrate how the frequency content can be determined, and that the frequency content is not constant in time. In fact, it slowly increases over the region of interest.

A spectrogram is not necessary to determine the frequency content over the entire region of interest. A Fourier Transform can, and often is, utilized for that purpose. That is, the Fast Fourier Transform (FFT) shows the frequency content without regard to the

times at which it occurs. So, in figure 3-4 electrodes 14 and 15, as well as electrodes 9 and 10 (and some other electrodes to lesser extents) have frequency content around 20 Hz. However, the times at which these frequencies occur is not clear until the spectrogram is considered.

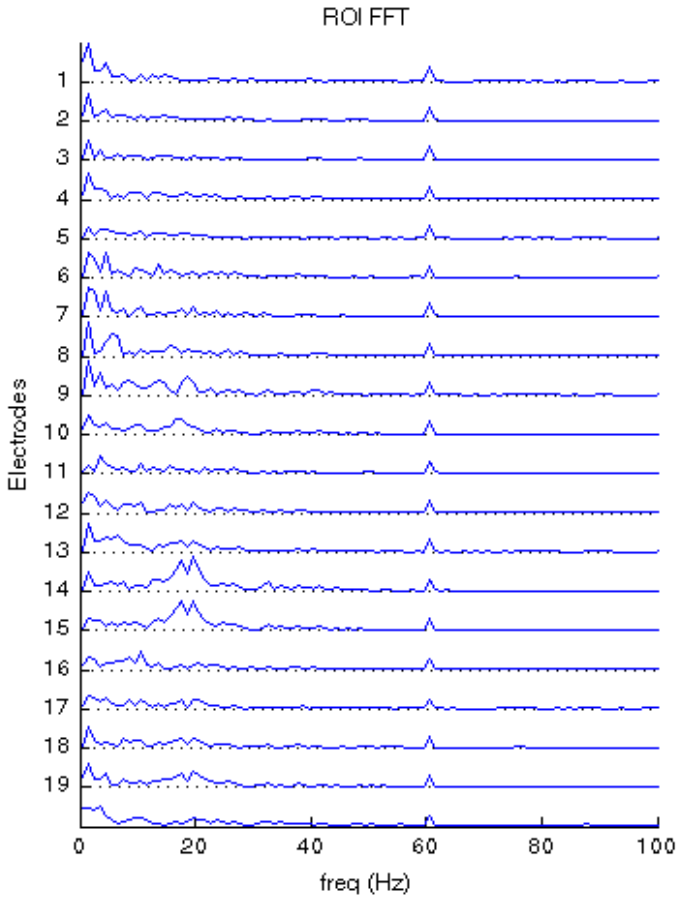

Figure 3-4: FFT of all 20 electrodes showing strong frequency content around 20 Hz on various electrodes, mostly 9, 10, 14, and 15.

The spectrogram of electrode 14 in figure 3-5 shows necessary agreement with figure 3-4; but, as stated above, figure 3-4 is not sufficient to describe the dependence on time. The spectrogram shows that the frequency content starts at 138 and ends at 140. It

also shows that there is some low frequency content at each end.

For this data it also shows that between 138.5 and 139.5 the frequency is increasing. The increase can be quantified. On the frequency axis the increase is seen to be approximately 2 Hz between 139.5 and 138.5. This increase, therefore, could be described as 2 Hz per second. This may or may not have a physiological interpretation or importance.

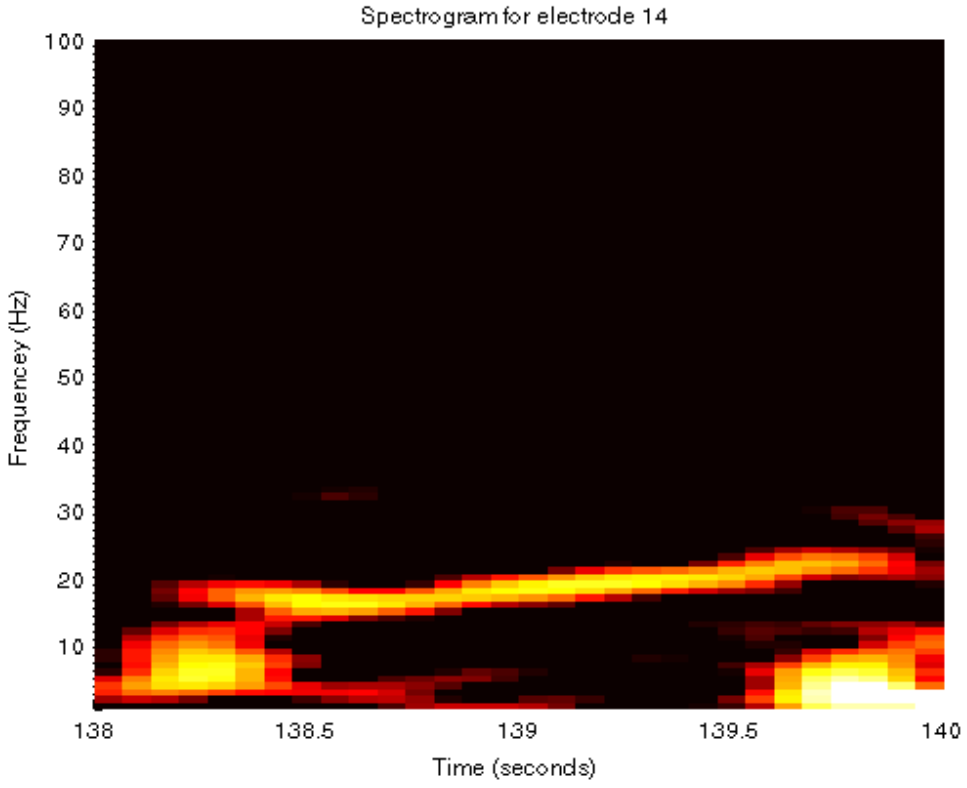

Figure 3-5: Spectrogram of electrode 14. This shows that the frequency increases with time.

Quantify movement

After the signals are arranged spatially, more data can be derived. Some examples

are: the centroid, the interpolated maximum, gradients, spreading depolarization, and pattern recognition. The GUI can assign the spatial-time matrix in the base workspace, so many of these examples are possible as further analysis. The first two examples are built-in and can be assigned to the base workspace or plotted directly (see Chapter 2, Movie Outputs).

There is a Matlab figure for each of X and Y. Each is an overlay of the display's centroid and the interpolated maximum. Figure 3-7 shows just the plot for X. The abscissa is the time in the region of interest. The values on the plot's y-axis are the image X coordinates in mm. Note that the plot for Y is more intuitive since the distances are vertical in both the movie and the plot. The distance axis is accurate according to the parameter *electrode_spacing* specified during the conversion to binary. In this case, the spacing is 10 mm and the electrodes are at the multiples of 10. The origin is referenced at the bottom left of the array.

The display's centroid and interpolated maximum clearly differ. The time resolution of the centroid is finer than the time resolution of the interpolated maximum because the centroid depends on the entire spatial extent rather than a single point. There are multiple discontinuities in the interpolated maximum because multiple spatially disjoint values can become bigger than one another.

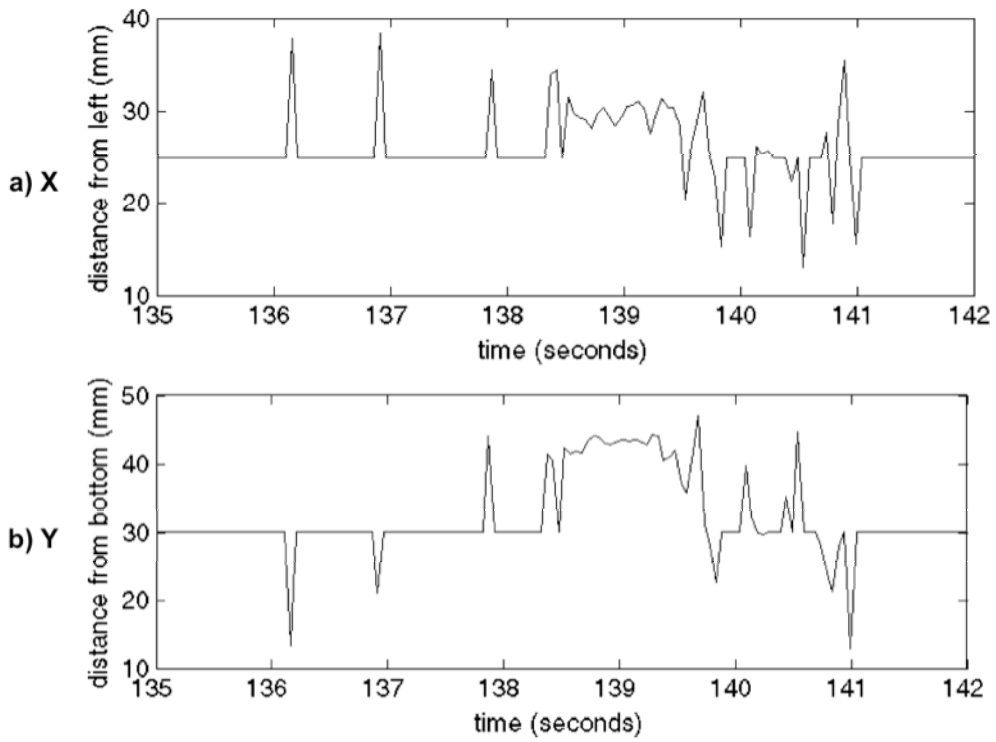

Figure 3-6: Each spatial dimension of the display centroid is shown on a behavioral time scale. Between approximately 138 seconds and 140 seconds the centroid's average position is shifted to a specific location.

The first inspection of the potential spatial movement of the signal occurs on a behavioral time scale (figure 3-6). For most of the ROI, the signal is all background so the display's centroid is in the middle of the frame. Between approximately 138 and 140 the centroid changes location. Therefore, there is likely a neurological feature at that time. There are some short single sample movements that could be filtered out to emphasize the feature.

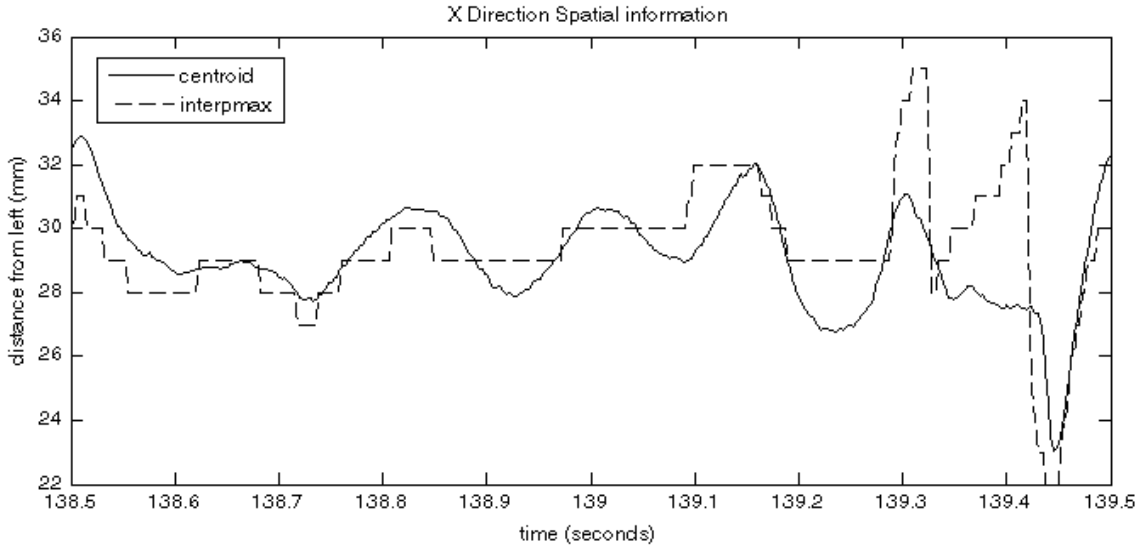

Figure 3-7: The GUI's output plots are shown here for the X direction. This provides detail for the centroid time series in figure 3-6.

The shorter time frame with just the feature is explored for further detail (figure 3-7). Figure 3-7 can be verified against the timeline of frames in figure 3-5. The most compelling movement occurs between the two frames for the times 139.35 and 139.45. Qualitatively, there is a right to left movement. In the centroid's X plot, the value indeed decreases. In fact, at those times, we can see that the centroid moves from about 28 mm to about 23 mm. In this timeframe, then, the velocity is 50 mm/sec.

Over most of the region of interest, though, it oscillates with X. The oscillations may be of concern to an epileptologist because oscillations organize spatial phase distributions (Wu et al 2008). The oscillation in figure 3-7 is approximately 6 Hz. A neurologist could consider this result. Also, the centroid time series could be treated as its own data set for further processing: spatial frequency analysis, derivatives, etc.

Vary dynamic range on a fixed frame

In the application above, it is difficult to determine how much signal was present on electrodes 14 and 15 when the display's centroid was only present near electrodes 9 and 10 in frame 139.45. For example, it would be important to know if the signal is completely inhibited on electrodes 14 and 15. The dynamic range is fixed at 10 to optimize the display across all frames of the region of interest. For this frame in particular, though, it is important to explore the dynamic range.

Figure 3-8 shows larger dynamic ranges. At a dynamic range of 13, electrode 14 is becoming visible. At a dynamic range of 14, electrode 15 is becoming visible with some areas that are not of interest. One dB more, at a dynamic range of 15, two additional electrodes (8 and 17) are moving the centroid downward. It becomes spatially significant that electrodes 14 and 15 are almost as weak as some of the less interesting background electrodes.

This not only shows the spatial relationship between electrodes' signal strengths, it also helps confirm the choice of 10 for dynamic range. Even with a dynamic range of 13, there is barely an increase in the contribution from electrode 14. Moreover, the rest of the frames in the region of interest would suffer. For example, at the nearby time of 139.37, a dynamic range of 13 results in too much background as shown in figure 3-9.

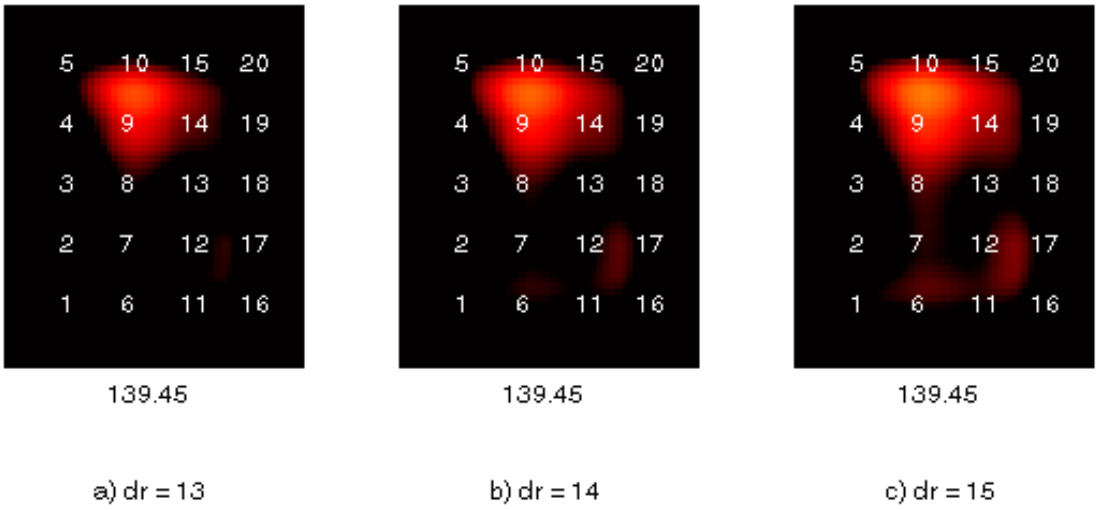

Figure 3-8: The dyanmic range of a fixed frame is incrementally increased. The spatial significance is that including neighboring electrodes also includes background electrodes.

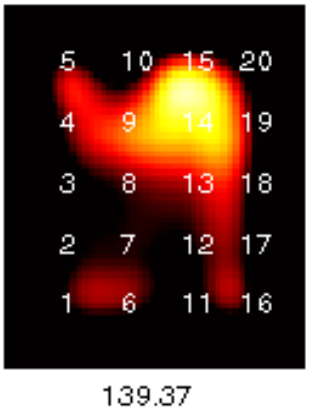

Figure 3-9: The trade-off from increasing the dynamic range for 139.5 is far too much background for 139.37.

The GUI enables the exploration of a single frame. There is a textbox for entering a single frame's time. Then, each parameter can be adjusted and just the single frame is rendered. So, in the first case, the time of 139.45 was held constant while the dynamic range was varied. Then, leaving the dynamic range fixed at 13, the time was set to 139.37 to see what that frame looks like. In the latter case the entire movie could have been generated to check all frames.

Other epochs from patient 1

Spatial similarity between all epochs

The above examples were illustrated with the same epoch. This patient has six other epochs from a different day. These were notated as events during monitoring. Table 1 shows the author's summary of them. All epochs occur in the same data set. The entire data set was, as always, converted to a binary file. In this way, the investigator can quickly switch between epochs. It is as simple as entering new ROI times. There is no delay between each ROI, and the parameters all remain unchanged.

number	hh:mm:ss	seconds	comment about data
1	00:22:44	1424	Worst
2	01:45:34	6364	Average
3	02:20:58	8488	Average; too much background
4	03:15:09	11739	Good; looks like inhibition
5	04:33:43	16453	Great; propagation, small noise
6	05:30:42	19872	Good

Table 3-1: List of epochs for patient 1. The program uses the integer number of seconds. The comments are only the quality of the data for demonstration purposes.

The other epochs have been explored to confirm spatial uniformity. Table 1 specifies each epoch in the data set with a comment about the usefulness of the epoch. Some are not adequate for detailed analysis because of background signals/noise or motion artifacts. This is not to say that they are entirely useless. Each epoch has enough data to visually confirm similarity with each other. Figures 3-10 and 3-11 show the mean frames (see Chapter 2, Movie) for all epochs.

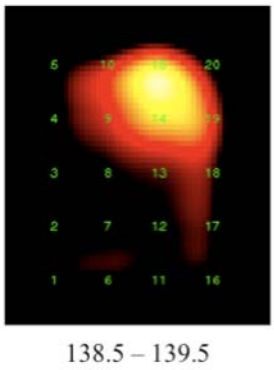

Figure 3-10: The mean of all frames for the previous epoch. It is shown here for comparison to the other 6 epochs.

Spatial shift between two epochs

The epoch around 16453 seconds (04:33:43) has the combination of good quality data and an interesting spatial feature. As displayed in figure 3-12, the second epoch occurs further to the right. The figure shows this hypothesis qualitatively. For a quantitative measure, the medians were computed. The median is used instead of the

mean to give the distance cutoff at which half of the brain activity is on each side. The respective medians are 29 mm and 33 mm from the left.

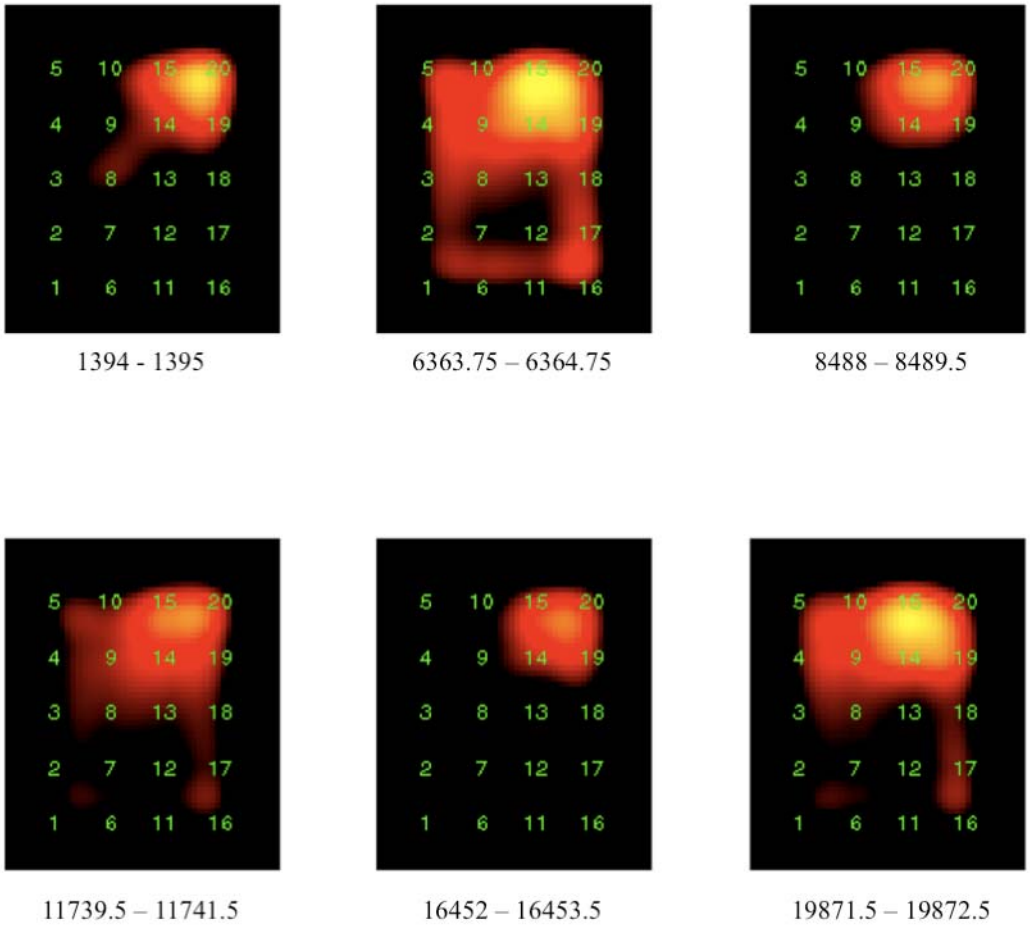

Figure 3-11: The mean frame from each of the other 6 epochs. The mean frames all compare similarly to the previous epoch. The motion noise in 3 epochs is evidenced by some signal on more or almost all electrodes.

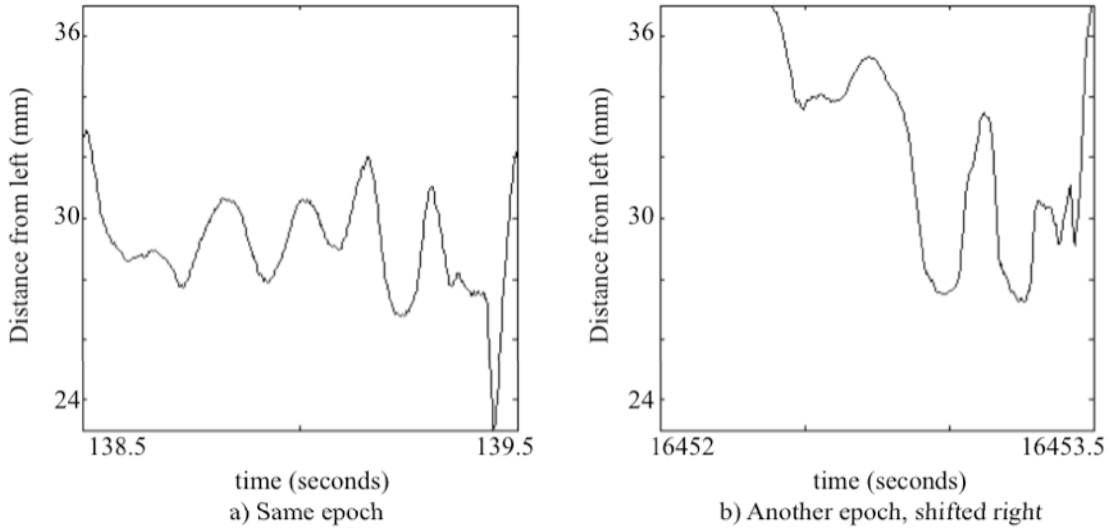

Figure 3-12: Another epoch (times 16452 – 16453.5) is compared to the previously used epoch (times 138.5 – 139.5), and appears to be shifted toward the right. a) Same epoch as in previous section, repeated here for comparison. Its median distance is 29. b) Another epoch with a shift to the right: median is 33.

Using the display's centroid has some disadvantages. The calculation depends on low background signals. Since the centroid is dependent on the threshold determined by the display's dynamic range, the dynamic range must be adjusted to include only the feature of interest. For example, a background signal on the opposite corner of the grid will bias the centroid. However, if the background signal is as strong as the feature, removing it by lowering the dynamic range is difficult. The display's centroid time series is available in the base workspace for additional manipulation when necessary. Another difficulty is that there is no reference between the epochs, so the choice of ROI must be done visually to match the similar features of each epoch.

Patient 2

Background

The applications thus far have used the data of one patient. Patient 2 has different features in the seizures: the focuses are in different spatial locations, the features vary in time duration, and they occur in differing frequency bands. The data collection is also different for this patient due to 3 electrodes that appear to not be operating correctly. These properties are demonstrated in this section.

Some background for this data set is known. The patient's history includes noninvasive video-EEG monitoring that demonstrated seizures of right temporal onset. It also included multifocal interictal activity that required a non-invasive work-up (MRI and PET). These led to invasive monitoring. The monitoring includes a right temporal grid, and left temporal strips. Only the grid data is utilized here.

The first step is to establish the initial processing parameters. One seizure event was annotated to occur at 11:44:50, or 12060 seconds. The ROI at 12042 – 12050 is imaged. The signal on electrode 6 is the starting point, so a spectrogram was created (figure 3-13a). It reveals strong frequency content at about 8 Hz over almost the entire ROI. The Hanning window frequency parameters (see Chapter 2) were set to 4, 6, 10, and 12 Hz for this band. Even though the maximum across all electrodes was 53 dB, the movie was created with a maximum dB display of 45 dB to emphasize

the activity on electrode 6. The movie was set to have a dynamic range of 15 dB. Selected frames from the resulting movie are displayed in figure 3-14a.

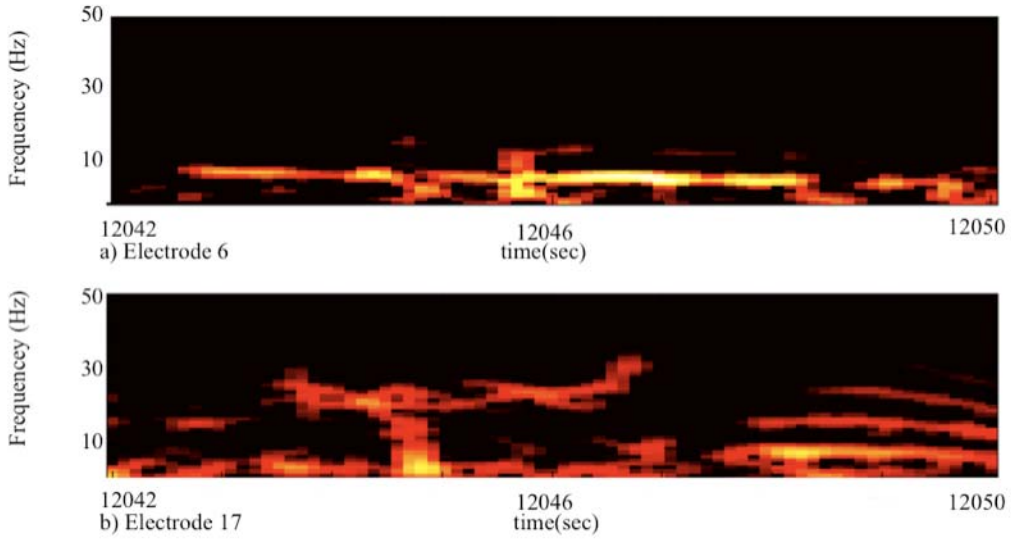

Figure 3-13: Spectrograms from this epoch (12042 – 12050 shown) revealing various frequency content. a) Electrode 6 shows distinct and strong frequency content around 8 Hz. A dynamic range of 13 dB was used to suppress some of the background. b) Electrode 17 shows distinct but weaker frequency content around 25 Hz. A dynamic range of 25 was used. The maximum dB displayed is the same between both figures, so the relative strength is conveyed.

Disjoint localizations of two frequency bands

Figure 3-14a shows that some activity occurred on other electrodes, for example electrode 17. Its spectrogram was also generated (figure 3-13b). Electrode 17 has an apparently different feature. It occurs around 25 Hz. It is shorter in time, and weaker in amplitude than the 8 Hz feature around electrode 6. Figure 3-14b shows

selected frames from a movie with a filter tailored to the 25 Hz frequency content. The Hanning window frequency parameters were set to 17, 20, 30, and 33 Hz for this band. All other parameters remained the same.

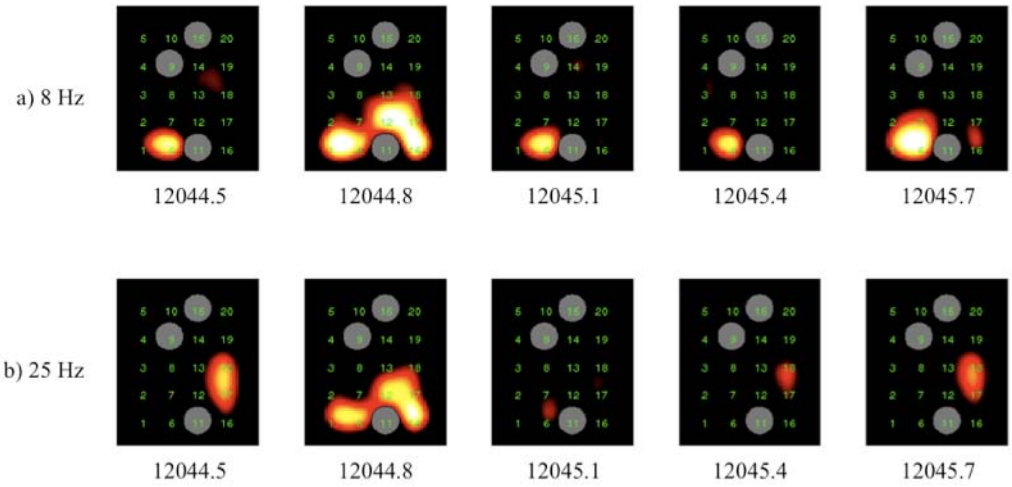

Figure 3-14: The same ROI was imaged in two frequency bands. a) The filter is centered on 8 Hz and occurs mostly around electrode 6. b) The filter is centered on 25 Hz and occurs mostly around electrode 17. The gray circles indicate bad electrodes. It is clear that electrode 11 might have detected seizure activity were it operating correctly.

The set of bad electrodes is another important property of this data set. Electrodes 9, 11, and 15 do not contain brain signal. This is determined by visual inspection of their time traces. However, from only the time traces, the spatial significance is lost. In the spatial arrangement (figure 3-14), the significance is clear. Electrodes 9 and 15 are likely not important. Electrode 11, on the other hand, shows a potential loss of important information. In the second frame, 12044.8 seconds, one could hypothesize that the area under electrode 11 has at least some signal. So, in this case, the spatial dimension shows the significance of

bad data rather than connecting good data.

Interesting shape of high frequency spectrogram

As seen in the above applications to this data set, there is a lot of time-frequency content. This section contains another example from the epoch around

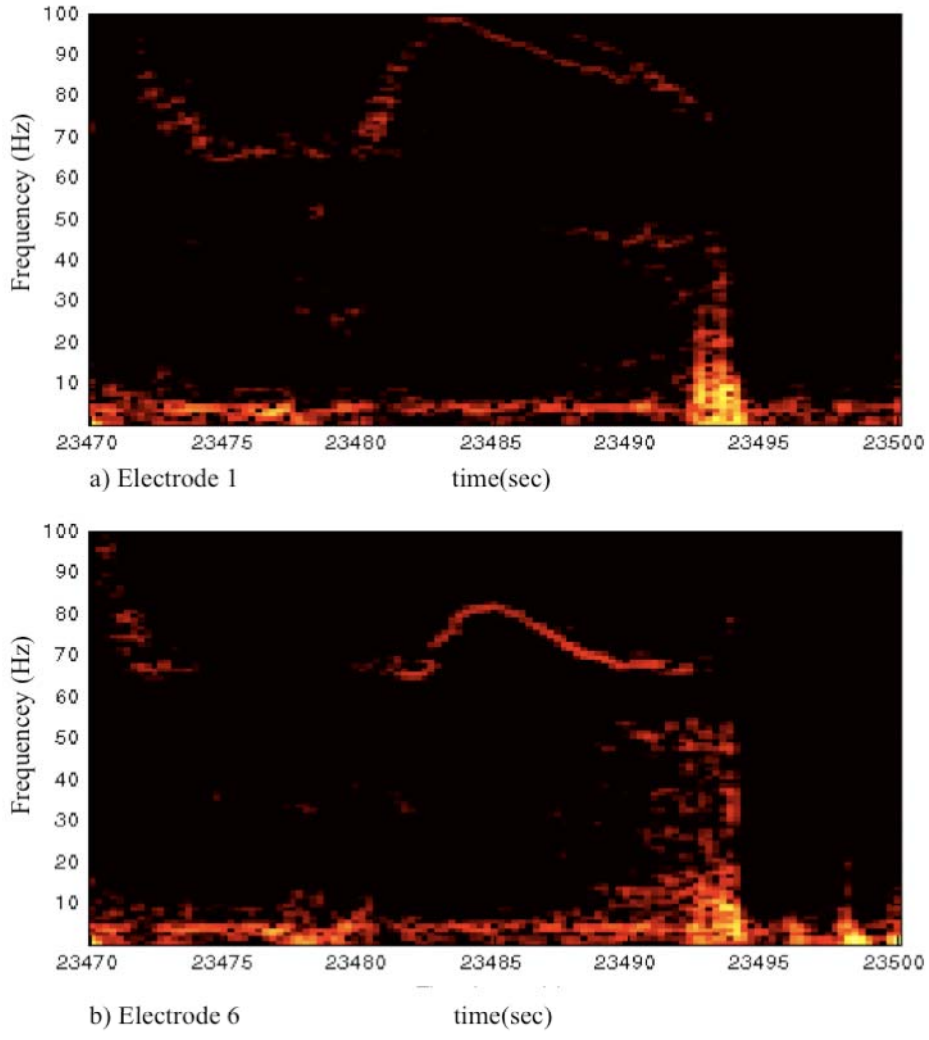

Figure 3-15: Interesting looking frequency content exists on electrodes 1 (a) and 6 (b).

the event annotated at 23490 seconds (20:25:21). This application is one result of data mining on the behavioral time frame of 23470 to 23500.

The time-frequency pattern shown in figure 3-15 looks distinctive. It is similar in shape on electrodes 1 and 6. It occurs pre-ictal and there is no change in behavior evident from the video monitoring. The rather abrupt stop at the end of (time = 23495 seconds) may also be worth some investigation. The important question for this pattern is whether it is indeed neural data.

The next reasonable step provided by this suite of tools is to explore this frequency band spatially. This would be motivated by trying to determine on which other electrodes this pattern might occur. Toward this end, a filter was set for the

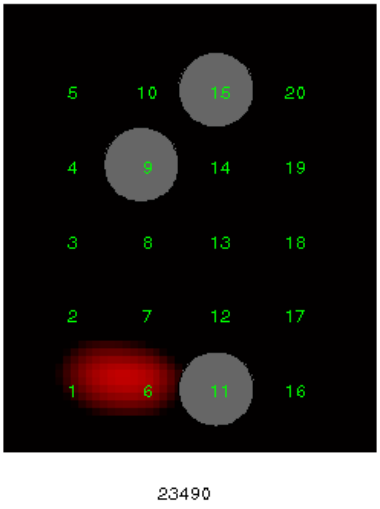

Figure 3-16: A characteristic frame of the movie produced by bandpass filtering only frequencies from roughly 50 to 100. This is similar to the spatial pattern seen around 8 Hz for this patient.

frequencies between 50 Hz and 100 Hz. The result is the familiar pattern that this patient's data often demonstrates. One highly characteristic frame (figure 3-16) shows the figure between electrodes 1 and 6.

Chapter 4 – Conclusion and Future Work

The tools developed in this project have been shown to enable spatio-temporal and time-frequency analysis. This analysis has not been sufficiently performed on subdural human epilepsy data. When it is performed, it will improve resection presurgical assessment and general research.

The tools have been developed for a neuroscientist. So the design assumes that the user can generally identify neural signals on a time trace. From there, the tools intuitively help reveal spatio-temporal properties of the data. As with most software, some training and practice will lead to better familiarity. Then, application to any data set will lead to repeatable results.

Each data set is explored for spatio-temporal properties that are not evident from the time traces. The first step toward this end is initial processing with a frequency filter, an envelope, dB scaling, and proper dynamic range. Time-frequency information is presented in the spectrogram to assist this process. The spatial arrangement of the electrodes into their original layout is the main goal. Various new time series are computed to avail quantitative data. The data produced by each stage can be made available to the Matlab workspace for almost endless analysis.

The future work can be broken down into four categories: test and fix bugs; improve existing functionality; improve with new functionality; apply to data sets. Testing and fixing bugs is self-explanatory, but its importance in the software development cycle cannot be overstated.

The second category includes as-yet identified improvements to existing functionality and many that have been identified. The convert to binary GUI (see Appendix C) accepts only tab separated files, but could be modified to allow the user to choose the file format. The bandpass frequencies could be specified by the two band frequencies and the rising/falling edges separately. The interface could also construct it more intuitively with a graphical component. The 60 Hz prefilter could interpolate the spectrum so that it doesn't go to zero unnaturally. The motion determination parameters (number of electrodes, percent coherent, and region length) could be optimized on more data sets. Also, it could be applied to more than just testing a maximum. These could be welcome improvements for the user.

The third category is new functionality altogether. A true centroid of the data, rather than the display's centroid as implemented, could be possible with some additional processing. An interpolation model could be developed to give better results than cubic interpolation. For example, a delay between similar signals on two electrodes could imply some information about the source's placement between them. A known background signal could be used to infer inhibition. Both of those, then, could be imaged in meaningful ways. Lastly, from an interface point of view, the potential exists for integration between time traces and movie so that a vertical line follows the time traces as the movie frames progress. These are only a few of the possibilities that could be built upon this framework.

The last category is not work on the tools, rather with the tools. There are numerous possible plans to proceed. The simplest plan, of course, is to use the tools

for data mining on an existing data set. On the other hand, a bigger effort could be organized to determine the availability of, and collect, data sets to see how they compare spatially. In either case, a subject matter expert who is familiar with neurological signals should perform the analyses. It is the hope of the author that many interesting properties are revealed.

Appendix A – Tutorial

Introduction

The aim of this tutorial is to introduce the basic and necessary workflow to view a data set in the spatio-temporal domain. Sample data is provided for this purpose.

Convert to binary

Open Matlab.

Set the directory to the “makemovie” directory.

Type in the command window “gui_convert_binary”.

Click “Select Input File”.

Choose “makemovie\sample_moveacross.txt”.

Enter “Header Lines” = 0.

Enter “Multiplier “= 1.

Click “Select Output File”.

Select your working directory, preferably not the “makemovie” directory.

Click “Open”.

Under “Enter File”, replace “No Base Name entered” with “sample_moveacross”.

Enter “Sampling Rate” = 400.

Enter “Grid Dimensions” = 5 by 4.

Enter “Electrode Spacing” = 10.

Leave Start Time as default 0s.

Click “Run Conversion”.

Observe the progress tracking, and the message box to notify completion.

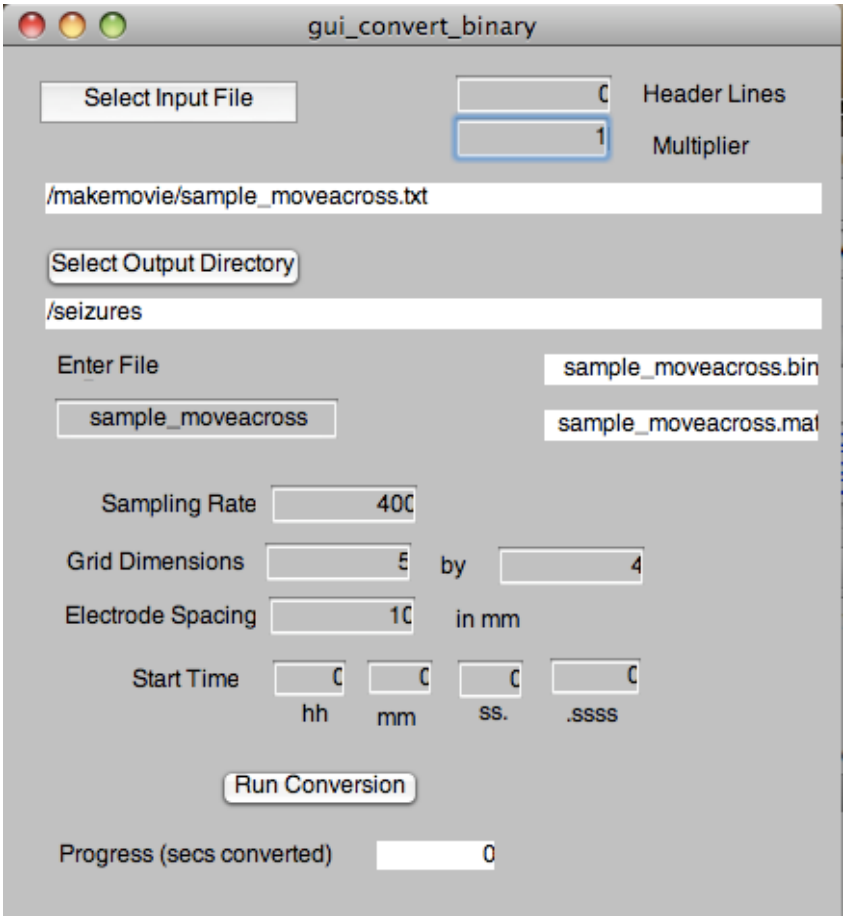

Figure A-1: Parameters for converting the sample text file to binary.

Make movie

Open Matlab.

Set the directory to the “makemovie” directory.

Type in the command window: “gui_makemovie”.

In the “Load Data” panel, click “Select File”.

Select the file created in the previous section: “sample_moveacross.bin”.

Leave the default “Begin Time” = 0.

Enter “Finish Time” = 2.

In the “Processing Stages” panel:

Click “Plot” with the defaults selected “Signal” and “All”.

Notice the signals on electrodes 2, 7, and 12.

Close the signal plot window.

In the Spectrogram panel:

Click “Over all electrodes”. Notice the value is set automatically in “Maximum”.

Select “Electrode” = 7.

Click “Spectrogram”.

Notice the feature is around 10 Hz.

Close the spectrogram window.

In the “Processing Stages” panel:

Select “FFT”.

Select “All”.

Click “Plot”.

Close the fft all window.

Select “7”.

Click “Plot”.

Close the fft for electrode 7 window.

In the “Parameters” panel:

Set the Hanning filter centered at 10 Hz: 6, 8, 12, 14.

Click the button: “Calculate ROI Max”.

In the “Processing Stages” panel:

Successively plot the stages: “Filtered”, “Hilbert”, “dB Scaled”, and “Dynamic Range”.

In the “Make Micro Movie” panel:

Enter “Step Size” = 4.

Check “XY Plots”.

Click “Make Movie”.

Please wait while the movie renders.

Close the plot (it now contains just the final frame).

Click “Playback”.

Use the intuitive mplay button panel to explore the movie.

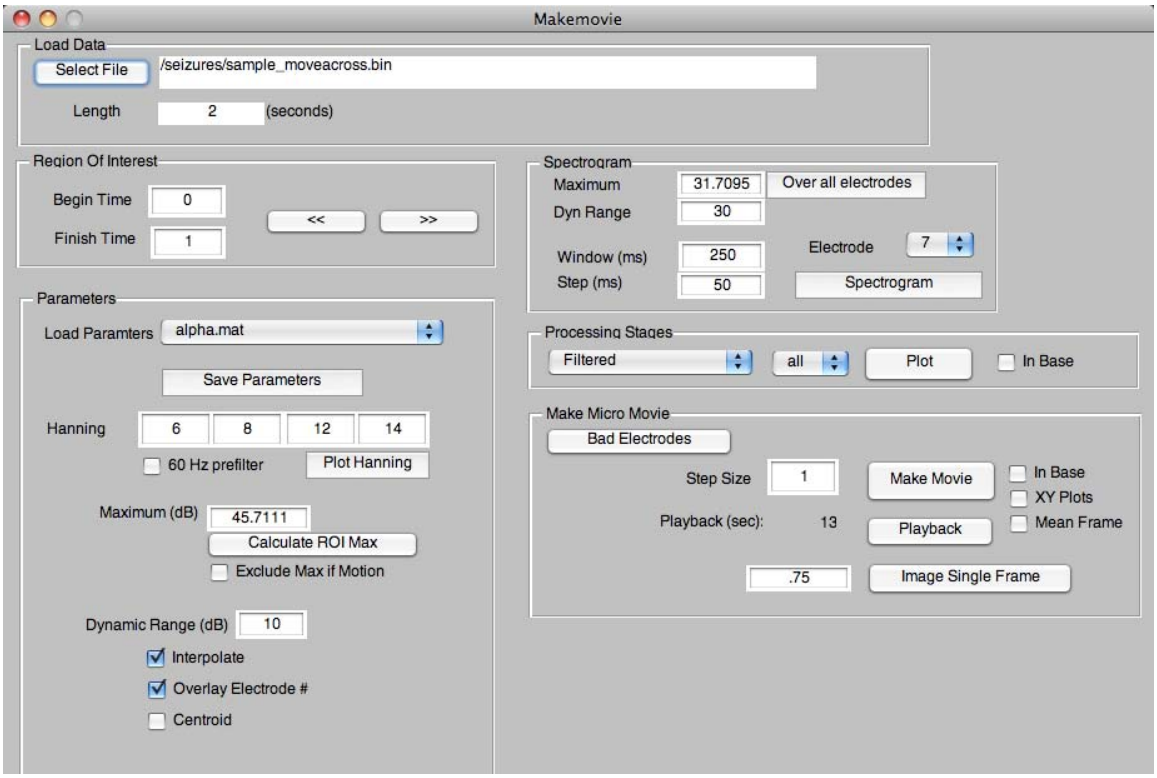

Figure A-2: Parameters for making the movie.

Conclusion

The skills attained through this tutorial are the necessary skills for exploring a data set. The body of the thesis should be referenced for further capabilities and details.

Appendix B – Matlab Code

All files necessary for the execution of either GUI are under one main directory. This directory is titled “makemovie”. There are two subdirectories. First, the directory “MoviePlayer” contains all the files necessary for the GUI to utilize the function *mplay()* for movie playback. Second, the directory “ProcessParams” contains all the Matlab workspaces that are used to define the built-in and custom filters (see Chapter 2, Initial Processing). The total disk space is under 500 KB.

Description and definition of each file

The purpose of this appendix is to describe the code for each file. The files are listed alphabetically to make reference easier.

- calcmeancoh.m

  Calculates a matrix of pairwise coherence values for all electrodes. The matrix is applied to the determination of motion.

```
function meancoh = calcmeancoh(electrode_data, sample, ...
   check_length, sampling_rate, num_electrodes)
```

- create_hanning.m

  Returns the vector by which to multiple the fft to filter the data.

```
function out=create_hanning(f1,f2,f3,f4,freq_ss)
```

- electrode_array_filter.m

Applies the filter to each electrode.

```
function [electrode_data_ifft_filter, electrode_data_fft] = ...
    electrode_array_filter(electrode_data, sampling_rate, ...
    f1, f2, f3, f4)
```

- electrode_array_process_data.m

  Applies 60 Hz and bandpass filters, and returns the data in the dB scale.

  ```
  function roi_data = electrode_array_process_data(struct_makemovie)
  ```

- electrode_array_read.m

  Reads and shapes correctly the binary data set.

  ```
  function roi_data = electrode_array_read(struct_makemovie)
  ```

- gui_convert_binary.m

  Contains all the functions necessary to support the GUI.

- gui_makemovie.m

  Contains all the functions necessary to support the GUI.

- hotcold.m

  Matlab colormap by R. Witte used for displaying inhibition (not implemented).

  ```
  function h = hotcold(m)
  ```

- image_movie_frame.m

  Arrange electrode grid spatially, interpolate, and generate quantitative output.

```
function struct_movie_frame = image_movie_frame(...
    movie_data, frame_index, frame_time, step_size, ...
    array_dim, interpolate, electrode_spacing, ...
    maximum_value, minimum_value, ...
    overlay_numbers, bad_electrodes, centroid, ...
    preserve_sign)
```

- ismotion.m

  Determines if the result of calcmeancoh() is motion by checking the number of electrodes above a certain percentage.

```
function bool_result = ismotion(meancohmatrix)
```

- makemovie.m

  Loops through ROI to create a movie of each image frame.

```
function bool_result = ismotion(meancohmatrix)
```

- notch60.m

  Applies FIR notch filter.

```
function filtered = notch60(signal, samp_freq)
```

- plot20.m

  Plots any number of electrode time series (not just 20) on the same y-axis.

```
function plot20(signal, axis_signal, plottitle, ...
    yaxislabel, xaxislabel, newfigure)
```

- sample_moveacross.txt

  Sample data for the tutorial (see Appendix A).

Code for selected files

```
function roi_data = electrode_array_process_data(struct_makemovie)

%% Authorship

%this function was written by Andy Bean
%between May 2008 and October 2009
%continuation of code written by Jeff Meade
%between August 2007, and May 2008
%at the University of Arizona, Tucson
%in collaberation with Dr. Russell Witte

% change namespace
file = struct_makemovie.param.file;
process_data = struct_makemovie.param.process_data;

%% Get the data for the movie

roi_data = electrode_array_read(struct_makemovie);

%% 60 Hz filter if applicable
if process_data.filter60
    roi_data = notch60(roi_data, file.sampling_rate);
end

%% Frequency filter if applicable

if size(process_data.filter_freqs, 2) == 4
    f1 = process_data.filter_freqs(1);
    f2 = process_data.filter_freqs(2);
    f3 = process_data.filter_freqs(3);
    f4 = process_data.filter_freqs(4);
    roi_data = electrode_array_filter(roi_data, ...
        file.sampling_rate, f1, f2, f3, f4);
end

%% Convert filtered data to dB scale if applicable

if process_data.dB_scale

    %we do not wish to display the full dB range of the data,
    %but rather only observe the largest dB values,
    %because large values correspond with
    %significant brain activity. the variable 'threshold'
    %indicates what range of
    %dB we wish to look at (dynamic range).

    %we also wish to preserve the sign of the data for
    %additional structural patturn analysis,
    %which is included in the last line of this code

    data_db = 20*log10(abs(roi_data));

    %scale is set by gui
    %data_db = data_db .* ...
```

```
    %   (data_db > (process_data.maximum_db - process_data.threshold));

    if process_data.preserve_sign
        roi_data = data_db .* sign(real(roi_data));
    else
        roi_data = data_db;
    end

end
```

```
function [movie_frame spaced_movie_matrix centroid_frame
interpmax_frame] = makemovie(struct_makemovie)

%% Authorship

%this function was written by Andy Bean
%between May 2008 and October 2009
%continuation of code written by Jeff Meade
%between August 2007, and May 2008
%at the University of Arizona, Tucson
%in collaberation with Dr. Russell Witte

%% Comments

%this program will use the raw ecog electrode data and create a
%spatial-temporal movie of the activity within a user defined time
%selection

%the current data set(s) are from a 20-element electrode array arranged
in
%a 5x4 grid. the 1st electrode is the bottom left and the numbers
%increase going up the column until the 20th electrode is the top
%right. the following illistration shows this

%   [05][10][15][20]
%   [04][09][14][19]
%   [03][08][13][18]        ELECTRODE GRID
%   [02][07][12][17]
%   [01][06][11][16]


%% Make structure variables easier to work with

% change namespace
file = struct_makemovie.param.file;
process_data = struct_makemovie.param.process_data;
process_movie = struct_makemovie.param.process_movie;
roi_time = struct_makemovie.param.roi_time;

% notch60, filter, db, etc.
movie_data = electrode_array_process_data(struct_makemovie);
%movie_data = zscore(movie_data, 0, 2);

%% Make the movie file (interpolation built into this code)

%note: the movie file has a maximum number of frames of 999, so keep
```

```
this
%in mind when specifying begin time, end time, and step size

%**** CHECK FOR number of frames < 999 ?

%set the time frame we are interested in looking at
index_begin = 1;
index_finish = floor( (roi_time(2) - roi_time(1)) * file.sampling_rate
);
num_frames = floor((index_finish-index_begin +
1)/process_movie.step_size);
%movie_frame = cell(num_frames, 1);
spaced_movie_matrix = zeros(num_frames, ...
    (file.array_dim(1) + 1) * file.electrode_spacing, ...
    (file.array_dim(2) + 1) * file.electrode_spacing);
centroid_frame = zeros(num_frames, 3);
interpmax_frame = zeros(num_frames, 2);
figure(5)

%minimum_db = 0;
minimum_db = process_data.maximum_db - process_data.threshold;

%(temporarily implemented without user control)
aviobj = avifile ( 'test.avi', 'fps', 30 );

for frame=1:num_frames

    struct_movie_frame = image_movie_frame(...
        movie_data, ...
        frame, ...
        roi_time(1) + ( (( frame-1) * process_movie.step_size) ...
                                    / file.sampling_rate ), ...
        process_movie.step_size, ...
        file.array_dim, ...
        process_movie.interpolate, ...
        file.electrode_spacing, ...
        process_data.maximum_db, ...
        minimum_db, ...
        process_movie.overlay_numbers, ...
        process_movie.bad_electrodes, ...
        process_movie.centroid, ...
        process_data.preserve_sign);

    %build up the movie file
    movie_frame(frame) = getframe;

    aviobj = addframe ( aviobj, movie_frame(frame) );

    %build up spatial data
    spaced_movie_matrix(frame,:,:) = ...
        struct_movie_frame.spaced_image_matrix;

    centroid_frame(frame, 1) = struct_movie_frame.centroid_x;
    centroid_frame(frame, 2) = struct_movie_frame.centroid_y;
    centroid_frame(frame, 3) = struct_movie_frame.centroid_value;

    if process_movie.interpolate
        interpmax_frame(frame, 1) = struct_movie_frame.interpmax_x;
        interpmax_frame(frame, 2) = struct_movie_frame.interpmax_y;
```

```
    end
end

aviobj = close ( aviobj );
```

```
function struct_movie_frame = image_movie_frame(...
    movie_data, ...
    frame_index, ...
    frame_time, ...
    step_size, ...
    array_dim, ...
    interpolate, ...
    electrode_spacing, ...
    maximum_value, ...
    minimum_value, ...
    overlay_numbers, ...
    bad_electrodes, ...
    centroid, ...
    preserve_sign)

%this function was written by Andy Bean
%between May 2008 and October 2009
%continuation of code written by Jeff Meade
%between August 2007, and May 2008
%at the University of Arizona, Tucson
%in collaberation with Dr. Russell Witte

%define some initial variables (interpolation code)
x_image_lim = (array_dim(2) + 2) * electrode_spacing;
y_image_lim = (array_dim(1) + 2) * electrode_spacing;

% Xe,Ye is the location of the electrode signal
% the + 1 is to add a margin around the image
% the spacing can be proportional with the electrode size
xe = 0:electrode_spacing:x_image_lim - electrode_spacing;
ye = 0:electrode_spacing:y_image_lim - electrode_spacing;
[Xe,Ye] = meshgrid(xe, ye);

% Xi,Yi is all the points that will take interpolated signal
xi = 1:1:x_image_lim - electrode_spacing;
yi = 1:1:y_image_lim - electrode_spacing;
[Xi,Yi] = meshgrid(xi, yi);

nonspaced_image_matrix = zeros(array_dim(2) + 2,array_dim(1) + 2);

j_index_begin = 1;


i=1;
si = j_index_begin + (step_size)*(frame_index-1);
ei = j_index_begin + (step_size)*frame_index - 1;
%convert from double to integer
si = floor(si);
ei = floor(ei);

%this code will average the frames together specified by the step size
for row=2:(array_dim(2) + 1)
```

```
    for col=2:(array_dim(1) + 1)
        nonspaced_image_matrix(row,col) = ...
            mean(movie_data(si:ei,i));
        i=i+1;
    end
end


if interpolate
    %interpolate the 4x5 to 40x50
    %(temporary to test usefulness; if kept, it should be a parameter)
    global interpolate_method;
    if isempty(interpolate_method)
        interpolate_method = 'cubic';
    end
    spaced_image_matrix = interp2(Xe,Ye,nonspaced_image_matrix',...
                                  Xi,Yi,interpolate_method);
else
    %correct spacing but don't interpolate
    %  still must flip left/right (row and col switched)
    spaced_image_matrix = zeros(y_image_lim - electrode_spacing, ...
                                x_image_lim - electrode_spacing);
    for row=1:array_dim(2)
        for col=1:array_dim(1)
            spaced_image_matrix(...
                col*electrode_spacing-1:col*electrode_spacing+1,...
                row*electrode_spacing-1:row*electrode_spacing+1) ...
                = nonspaced_image_matrix(row+1,col+1);
        end
    end
end

%add spaced_image_matrix to return structure
struct_movie_frame.spaced_image_matrix = flipud(spaced_image_matrix);

%set up this matrix for modification for display
display_image_matrix = spaced_image_matrix;


%find max as displayed in movie (interpolated or not)
[interpmax_y interpmax_x] = ...
    find(spaced_image_matrix == max(max(spaced_image_matrix)));
%add to struct
if isempty(interpmax_x)
    struct_movie_frame.interpmax_x = NaN;
else
    struct_movie_frame.interpmax_x = interpmax_x;
end
if isempty(interpmax_y)
    struct_movie_frame.interpmax_y = NaN;
else
    struct_movie_frame.interpmax_y = interpmax_y;
end
%add to movie
%display_image_matrix(round(interpmax_x), round(interpmax_y)) = 0;


%apply threshold after interpolating
%(used to be after centroid)
```

```
display_image_matrix = display_image_matrix .* ...
    (display_image_matrix > minimum_value);


%calculate centroid
image_for_centroid = display_image_matrix;
xc = 1:((array_dim(2) + 1) * electrode_spacing);
yc = 1:((array_dim(1) + 1) * electrode_spacing);
[Xc Yc] = meshgrid(xc,yc);
mass_image = sum(sum(image_for_centroid));
centroid_x = sum(sum(Xc .* image_for_centroid)) / mass_image;
centroid_y = sum(sum(Yc .* image_for_centroid)) / mass_image;
centroid_value = max(max(movie_data(si:ei,:)));

if isnan(centroid_x) || isnan(centroid_y)
    centroid_x = (array_dim(2)+1) * electrode_spacing / 2;
    centroid_y = (array_dim(1)+1) * electrode_spacing / 2;
    centroid_value = 1;
end

%add centroid to struct
struct_movie_frame.centroid_x = centroid_x;
struct_movie_frame.centroid_y = centroid_y;
struct_movie_frame.centroid_value = centroid_value;

%add footer for time
display_image_matrix = cat(1, ...
    ones(electrode_spacing, x_image_lim - electrode_spacing)...
    .* maximum_value, display_image_matrix);

%show image
imagesc(display_image_matrix,[minimum_value,maximum_value]);
axis image;
set(gca, 'YDir', 'normal');
set(gca,'Visible', 'off');

%add centroid to image
if centroid
    dia = 5 * centroid_value / maximum_value;
    circx = centroid_x - dia/sqrt(2);
    circy = centroid_y + electrode_spacing - dia/sqrt(2);
    rectangle('Position', [circx circy dia dia], ...
              'Curvature', [1 1], 'EdgeColor', 'yellow');
end

%show time on image
text((x_image_lim - electrode_spacing)/2, electrode_spacing/2, ...
    num2str(frame_time),  'horizontalAlignment', 'left', ...
    'verticalAlignment', 'middle');

%gray out bad electrodes
for be_index = 1:length(bad_electrodes)
    %find actual row and col indices (but rectangle accounts for time
    %footer)
    real_row = ceil(bad_electrodes(be_index)/array_dim(1));
    real_col = bad_electrodes(be_index) - (real_row - 1) * array_dim(1);

    circle_dia = floor(electrode_spacing);
    rectangle('Position', ...
```

```
        [real_row*electrode_spacing-floor(circle_dia/2)-1, ...
        (real_col+1)*electrode_spacing-floor(circle_dia/2)-1, ...
        circle_dia, circle_dia], ...
        'Curvature', [1,1], 'FaceColor', [.4 .4 .4]);
    %blusih greay [0 .5 .6]
    %dark ray [[.4 .4 .4]
end

%show electrode numbers on image?
if overlay_numbers
    for row=1:array_dim(2)
        for col=1:array_dim(1)
            text(row*electrode_spacing, ...
                (col+1)*electrode_spacing, ...
                num2str( ((row-1)*array_dim(1)) + col ), ...
                'Color', 'green',  'horizontalAlignment', 'center', ...
                'verticalAlignment', 'top');
        end
    end
end

%set colormap
if preserve_sign
    colormap(hotcold(256));
else
    colormap(hot(256));
end
```

Appendix C – Miscellaneous supporting features

Convert to binary

Motivation

Due to the amount of data being processed, it is important for the functions to be efficient. The first step in good efficiency is the input file. The input, of course, will be the output of some other software. This output will commonly be an ASCII file. A binary file is more desirable to work with, so ASCII will be converted to binary.

In ASCII format, a number is made up of a series of characters. So, in memory, a number is of varying length depending on the number of digits and the sign (i.e., a negative sign adds an ASCII character). Consequentially, each line of data (in this case, one sample from the electrode array) will be of varying length. This makes reading the file inefficient because a loop must process each individual character to get to a particular position in the file. To get to the beginning of the Nth line, the loop must identify N-1 end of line characters. Then, to get to the Mth electrode within the line, a separate loop must identify M-1 delimiter characters (e.g., space, tab, comma). The latter is not as much of a performance issue, but is less elegant.

In binary format, on the other hand, each number occupies the same amount of storage. So, in memory, a number is always the same length independent of the

number of digits or the sign. This avoids the need for looping through each ASCII character. It allows the program to read the Mth character on the Nth line by simply reading the number in the following memory position:

N * (number of electrodes) + M

The requirement for the user to change the ROI is the main motivation for efficiency. For example, a common operation will be to explore successive time ranges of the same length. Another example is to explore varying lengths of ROI. Changing the ROI in either way would especially be a problem on long data sets or at the end of data sets. In order to not interrupt the user, the transition should be accomplished as quickly as possible.

Effect on workflow

The trade-off is an extra step required before using the software. This extra step, though, is also an opportunity to collect metadata (e.g., sampling rate, electrode grid dimensions). An added benefit for the programming of the make movie GUI is that it can be simpler because the data is standardized before it is utilized.

Performance of conversion

Any performance statistics are based on a workstation computer with the following specifications: 2.2 GHz Intel Core 2 Duo with 2 GB of 667 MHz DDR2 SDRAM running Mac OS X version 10.5.7. The Matlab version is 2008b.

The most important consideration is not running out of RAM. It would certainly be easiest and most efficient to load the entire input, convert the entirety, and write the entire output. In general, though, the user's workstation may not be capable enough, and the file sizes could be far too large. Most built-in imports seem to be designed to convert an entire file at once, so this custom conversion is necessary. On the other hand, clearing the memory after each line takes too long to run. The interval length of 1 sample requires too much file input/output, so it's not efficient.

A good trade-off between minimizing the amount of RAM for each interval and minimizing the number of file I/O (input/output) operations is a 1 second interval of data loaded for each loop. An experiment was run to compare a 1 second interval to an interval of one sampling rate of data. The following table summarizes conversion times for 2400 seconds of data with 20 electrodes at 400 Hz.

Load Interval Length (sec)	**Conversion Time (sec) for 2400 seconds of data**
1/sampling rate	120
1	25

Table C-1: Comparison of loading one time sample at a time and one second at a time.

A 1 second interval has additional benefits. For the sake of programming, it is natural to consider data in 1 second intervals. Likewise, all data is likely to be originally captured in 1 second intervals.

Loading longer amounts of data before clearing memory didn't significantly improve performance. This was tested on 5790 seconds of data with 20 electrodes at 400 Hz (a 242.2 MB ASCII file). The following table summarizes the conversion times. Neither improvement justifies the added complexity of programming.

Load Interval Length (sec)	**Conversion Time (sec) for 5790 seconds of data**
1	58
10	56
100	56

Table C-2: Comparison of loading many seconds at a time.

Note that the discrepancy between conversion rates with 1 second interval (2400 in first table versus 5790 seconds in second table) can be easily explained. The two sets of experiments were run at different times, so the processes running on the computer were different. Therefore, the times can vary. Between the different interval times within each experiment, on the other hand, the load on the computer was constant.

The binary number's data type is integer instead of double. The precision of the data only requires one decimal point, so the many decimal points allowed by double would be wasted. The visualization functions assume units of microvolts, so the numbers are large enough to be stored as integers without losing precision. Note that if the original data is given in millivolts, a multiplier of 1000 can be specified before conversion. Although, the integer data type does give a 20% improvement in conversion time, this is not a compelling enough reason. More compelling is the resulting binary file size. The double precision files will be 4 times larger in size – this could be greater in size than the original ASCII file.

Alternative approaches

The choice to require the user to convert the data before using the tool has alternatives. For one, the ROI could be converted each time it changed. Clearly, that would not allow the user to work freely and uninterrupted. Instead, at the beginning of each session, the user could choose a larger window within which all their various possible ROIs would fit. This approach would be acceptable, but is only slightly different than just converting and saving the whole data set at once ahead of time.

Algorithm

The following describes the program after the “Convert” button is clicked.

- The metadata is collected into local variables. Some of these are

necessary for the conversion, and all will be saved when conversion is complete and successful.

- The format_line variable (for use in the C-style fget function) is set
- The multiplier is set (1 by default)
- Open input file for read only
- Open output file for write as binary
- Ignore the specified (0 by default) number of header lines
- Initialize the progress tracking capabilities
    - Set the second count to 0
    - Set the text box to “Starting...”
- Loop without condition
    - if end of file, break loop
    - increment second count variable
    - if second count is a multiple of 100
        - Set the progress text box to the number of seconds
        - pause for a hundredth of a second
    - read one second of data from ASCII files into a vector variable
    - write the binary data as int16
- Set the final progress
- Close both input and output files

- Save the metadata with the same file name as the binary file, but with the .mat file extension
- Display a message box to user indicating that it is finished

Data structure for make movie functionality

Overview

Many parameters are needed by the functions. Function parameter lists can become unwieldy. Instead, a nested structure was designed to accommodate this need. There are three top-level structures, and one top-level vector. The three top-level structures are named *file*, *process_data*, and *process_movie*. The top-level vector, *roi_time*, simply defines the region of interest by the start and end times.

file Structure

This is the input file's metadata. It is entered at the same time that the data is converted to binary. It is stored in a structure, *file*, in a Matlab workspace. By convention, the Matlab workspace file has the same name as the binary file, except with the .mat file extension.

The data stored in this structure is important for both binary conversion and for the makemovie GUI functionality. Below is a list of each variable stored in the *file* structure.

- *sampling_rate*: without specifying this there would be no reference for the spectral analysis and no reference to time
- *array_dim*: these two numbers specify the rows and columns of the rectangular electrode array
- *electrode spacing*: this number allows the makemovie algorithm to layout the electrode array spatially; it is particularly important for the interpolation functionality
- *start_time*: 4 numbers to give the user the ability to refer to a real-world hh:mm:ss.ssss clock time (not fully implemented)

process_data Structure

The user enters various important parameters for processing the input brain signal data into a power signal that can be displayed.

- *filter_freqs*: a hanning band-pass frequency filter is used, and these four numbers specify the band of the filter
- *filter60*: this is a true/false flag for applying a FIR notch filter centered at 60 Hz
- *maximum_db:* the data will have a maximum itself, but this allows the user to specify a different maximum to be used by the image scaling
- *threshold*: this is the dB range from *maximum_db* within which the data is not set to zero for display
- *preserve_sign*: a true/false flag to allow the polarity of the real part of the

signal to be re-applied after all other processing is completed; there might be additional information in the signals polarity at various electrodes (not fully implemented)

process_movie Structure

These are parameters used in constructing the matrix to be imaged as a frame of the movie.

- *step_size*: a multiplier used for averaging a number of samples into a single frame; commonly used to shorten the movie play length
- *interpolate*: a true/false flag for whether the blank space between electrodes (specified by *file.electrode_spacing*) will be interpolated
- *overlay_numbers*: this functionality allows the user to see the electrode number as it was originally laid out while the frame(s) are being displayed

60 Hz pre-filter

One of the most apparent sources of noise is the 60 Hz electronics noise. It can be a problem for simply looking at the raw data. When there is no brain signal, the time trace should show an almost constant 0 instead of a small oscillating signal. Figure C-1 shows the spectrum of the signal demonstrating the 60 Hz component, the original flat signal in time, and signal after removing the 60 Hz noise. In addition to time traces, the 60 Hz noise can also be a problem for the movie if left unfiltered. For these reasons, the option is available to remove it.

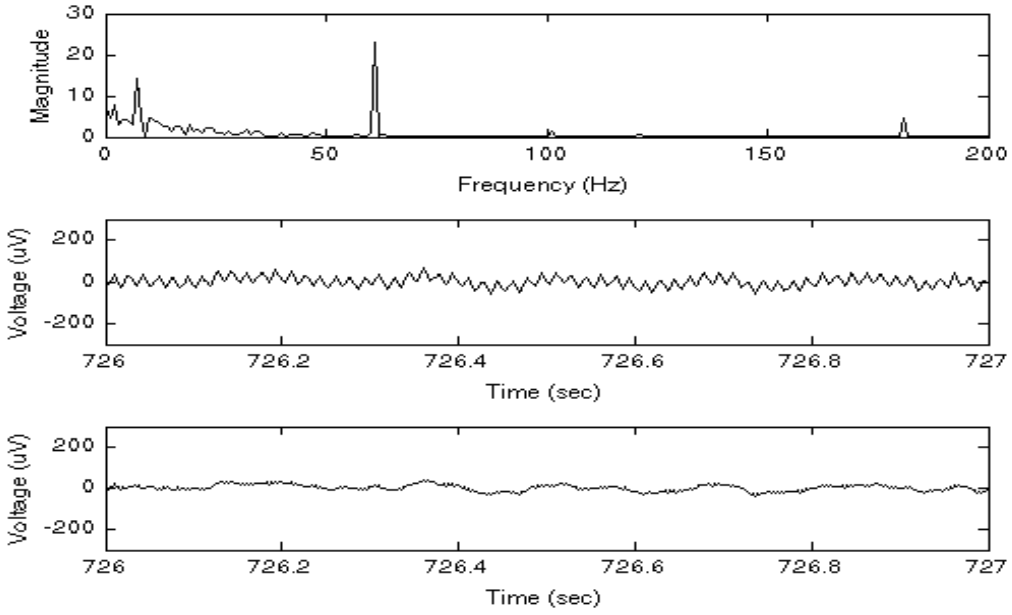

Figure C-1: The 60 Hz component is clearly visible in both the spectrum and the time trace. The notch filter removes the 60 Hz component.

Note that for many brain signals, 60 Hz falls outside the passband filter. So, often, there is no need to additionally pre-filter it. In this case, the option exists to not pre-filter.

Figure C-2 shows the performance of the 60 Hz pre-filter in the frequency domain. The large spike at 60 Hz is clearly gone. However, it does affect the average signal spectrum. This is seen more clearly in figure C-3. Most of these brain signals have some, but very little, signal power at all frequencies.

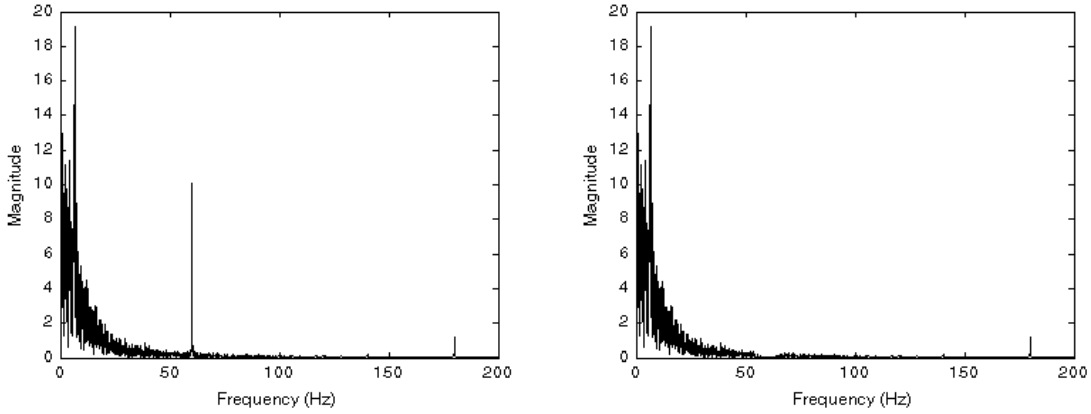

Figure C-2: Spectrum of a 10 second signal before and after the 60 Hz pre-filter.

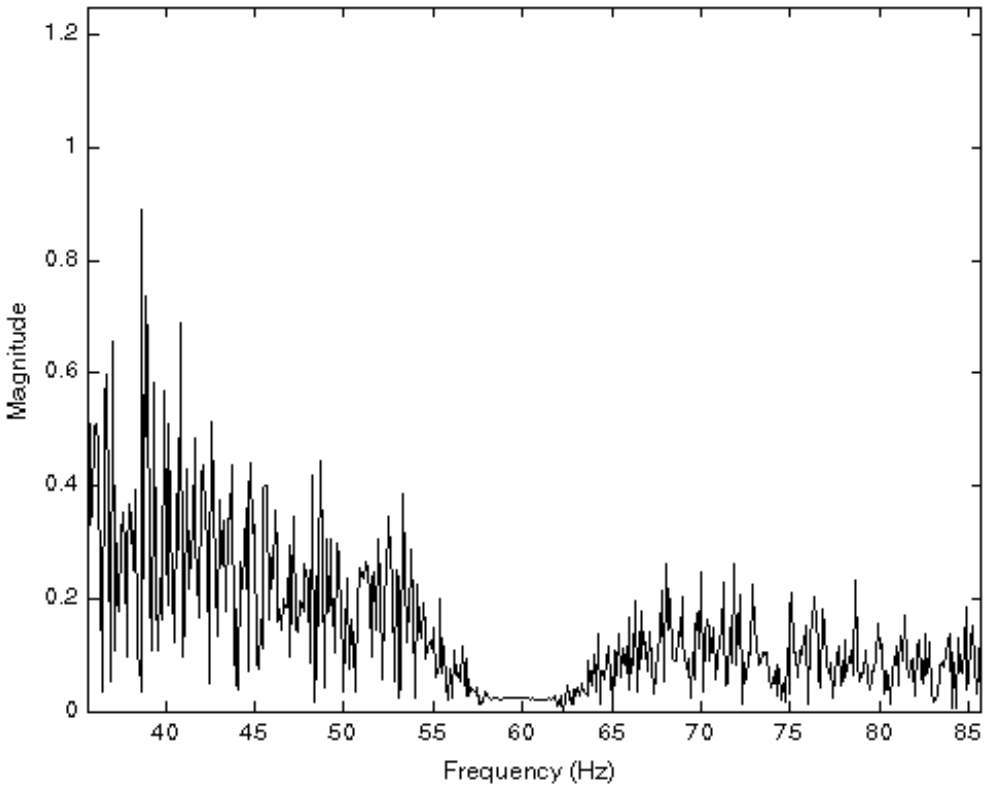

Figure C-3: A zoom into the 60 Hz region shows that the original signal is affected.

The built-in Matlab function *filter()* is used to apply the digital notch filter. The numerator and denominator coefficient vectors are calculated to match a Butterworth filter.

The Butterworth window is created using the built-in Matlab function *butter()*. The chosen fixed parameters for this filter are third order between frequencies 55 Hz and 65 Hz.

References

Arieli A, Shoham D, Hildesheim R and Grinvald A 1995 Coherent spatiotemporal patterns of ongoing activity revealed by real-time optical imaging coupled with single-unit recording in the cat visual cortex *J. Neurophysiol.* **73(5)** 2072-93

Bargalló N 2008 Functional magnetic resonance: New applications in epilepsy *European J. Radiology* **67** 401-408

Bracewell R N 1978 *The Fourier Transform and Its Applications, 2nd Ed* (New York, NY: McGraw-Hill) pp 267-70

Chervin R D, Pierce P A and Connors B W 1988 Periodicity and Directionality in the Propagation of Epileptiform Discharges Across Neocortex *J. Neurophysiol.* **60(5)** 1695-1713

Fisher R S, Boas WvE, Blume W, Elger C, Genton P, Lee P and Engel J 2005 Epileptic Seizures and Epilepsy: Definitions Proposed by the International League Against Epilepsy (ILAE) and the International Bureau for Epilepsy (IBE) *Epilepsia* **46(4)** 470-472

Freeman W J 2007 Hilbert transform for brain waves *Scholarpedia* **2(1)** 1338

Le Van Quyen M, Foucher J, Lachaux J-P, Rodriguez E, Lutz A, Martinerie J and Varela F 2001 Comparison of Hilbert transform and wavelet methods for the analysis of neuronal synchrony *J. Neurosci. Meth.* **111** 83-98

Lilly J C and Cherry R 1954 Surface movement of click responses from acoustic cerebral cortex of cat: leading and trailing edges of a response figure *J. Neurophsiol.* **17** 521-32

Meade J, Hutzler R L, Weinand M E and Witte R S 2008 Spatial and temporal mapping of seizures in an epilepsy patient using a chronic subdural grid electrode *University of Arizona College of Optical Sciences Industrial Affiliates Meeting* Poster

Nair D R, Burgess R, McIntyre C C and Lüders H 2008 Chronic subdural electrodes in the management of epilepsy *Clin. Neurophysiol.* **119(1)** 11-28

Plummer C, Harvey A S and Cook M 2008 EEG source localization in focal epilepsy: Where are we now? *Epilepsia* **49(2)** 201-218

Rosenow F and Lüders H 2001 Presurgical evaluation of epilepsy *Brain* **124** 1683-1700

Spencer SS, Guimaraes P, Shewmon A 1998 Intracranial electrodes In: *Epilepsy: a comprehensive textbook* ed Engel J, Pedley TA (New York, NY: Lippincott-Raven) pp 1719-48

Swartz, B E 1998 The advantages of digital over analog recording techniques *Electroencephalography and Clin. Neurophysiol.* **106(2)** 113-117

Ursino M and La Cara G E 2006 Travelling waves and EEG patterns during epileptic seizure: Analysis with an integrate-and-fire neural network .*I of Theoretical Biol.* **242(1)** 171-187

Witte R S, Rousche P J and Kipke D R 2007 Fast wave propagation in auditory cortex of an awake cat using a chronic microelectrode array *J. Neural Eng.* **4** 68-78

Wu J W, Huang X and Zhang C 2008 Propagating Waves of Activity in the Neocortex: What They Are, What They Do *Neuroscientist* **14(5)** 487-502