**Electrophysiology Analysis Optimization using Intan Technologies**

Documentation and Protocol presented

by

Baihan Lin

to

Olavarria Lab

Department of Psychology

in partial fulfillment of the data analysis task

in the subject of

Computational Neuroscience

University of Washington

Seattle, Washington, USA

Apr 2016

**Mentors:**

Dr. Adrian Andelin

Prof. Jaime Olavarria

**Author:**

Baihan Lin

**Abstract:**

The thesis is given birth due to the need to analyze a set of electrophysiology recording data and the curiosity of finding the best way to generate and present the result, and to simplify the entire electrophysiology analysis in a simple automated protocol. Comparing to the previous method, which spends to analyze the data files in our current stage, my method only takes , times faster. I hope this method described in this documentation can offer a more systematic and convenient way in the anlaysis of electrophysiology recording data. I look forward to further improvements if any.

**Comment:**

This documentation is mainly for the purpose of recording my thoughts and attempts in optimizing the data analysis task, therefore not in a standard of submission in any kind.

**Keyword:**

MATLAB, Mac terminal, electrophysiology experiments, Intan Technologies

**Contents:**

**[Abstract …………………………………………………………………………………. 2](#Abstract)**

**[Keywords ……………………………………………………………………………….. 2](#Abstract)**

1. [**Traditional Solution:** (Challenges Faced)……………. ………………………….…**5**](#TraditionalSolutions)
   1. Darking/Contrasting Process
   2. Utilization of Gaussian Blurring via ImageJ or Photoshop
   3. Arbitrary Drawing of line in the V1

1. [**Task 1: Indicating Patch Index** (Java, ImageJ, MATLAB, Patch Analyst) ……….. ...**6**](#IndicatingPI)
   1. Solution 1: Implementing the traditional (via Java or MATLAB)
   2. Solution 2: Patch analysis via Patch Analyst (New possible alternative)
   3. Solution 3: Particle analysis via ImageJ (New possible alternative)
   4. Solution 4: width of line (Improvement on traditional)
   5. Solution 5: Cluster Index (New possible alternative)
   6. Solution 6: Machine Learning (New possible alternative)
2. [**Task 2: Optimizing the data:** (ImageJ, Excel, Java, MATLAB)…………………...... **9**](#Optimizingdata)
   1. Standardization
   2. Smoothing Method (Moving Average):
      1. Excel (Exploration)
      2. Java (Implementation)
      3. MATLAB (Outlook)
3. [**Task 3: Systematization:** (systematize the protocol more efficiently) …………….**13**](#Systematization)
   1. Automatic file reading and processing
   2. Visual user-friendly version
   3. Automatic software operation
   4. Patch Processor (a Software designed by Baihan Lin)
4. [**Method Validation** ………………………………………………………………... **22**](#MethodValidation)
5. [**Future Improvements** ……………………………………………………………. **23**](#FutureImprovements)
6. [**Data Summary** …........................................…………………………………….... **24**](#DataSummary)
   1. Data Charts
   2. Excel Smoothing Plot (as an indication of the validity of SF = 10/364)
   3. ImageJ Plots & Lines (line & line with width of 100)
7. [**Code Summary** (20 Java programs)…….………..………………...…………….. **33**](#ImportantCodes)
   1. **First Attempt** (using TreeMap)
      1. [MapPatchIndexMain.java](#a1) (main method)
      2. [MapPatchIndexSolver.java](#a2) (subprogram)
   2. **Better Solution** (using ArrayList)
      1. [PatchIndexMain.java](#b1) (main method)
      2. [PatchIndexSolver.java](#b2) (subprogram)
   3. **Smooth Factor Finder** (test different smooth factor to find the most proper)
      1. [SmoothPatchIndexMain.java](#c1) (main method)
      2. [PatchIndexSolver.java (subprogram)](#c2)
   4. **Automatic System** (read-all files in a folder)
      1. [AutoPatchIndexMain.java (main method)](#d1)
      2. [PatchIndexSolver.java](#d2) (subprogram)
   5. **Attempt to visualize** (output the system console, failed)
      1. [AutoPatchIndexMainBackUp.java](#e1) (main method)
      2. [PatchIndexSolver.java](#e2) (subprogram)
      3. [ConsoleTextArea.java](#e3) (subprogram)
      4. [LoopedStreams.java](#e4) (subprogram)
   6. **Second Attempt to visualize** (user-friendly, using acm.jar and JComponent)
      1. [VisualPatchIndexMain.java](#f1) (main method)
      2. [VisualPatchIndexSolver.java](#f2) (subprogram)
   7. **Standard Java File** (standardize, exportable to jar, then to exe)
      1. [VisualPatchProcessorMain.java](#g1) (main method) with jar/exe
      2. [VisualPatchIndexSolver.java](#g2) (subprogram)
   8. **Version 2.0** (with excel export, smooth factor and imageJ functions, browse)
      1. [VEPatchProcessorMain.java](#h1) (main method) with jar/exe
      2. [ExcelFileDealer.java](#h2) (subclass)
      3. [PISolver.java](#h3) (subclass)
      4. [ImageJ.java](#h4) (revised subclass)

[**Bibliography ………………………………………………………………………..... 130**](#Biblography)

[**Special Acknowledgement …...…………………………………………………….... 131**](#SpecialAcknowledgemetn)

**Work Progress:**

**(What, when and how I approached the project)**

I used MATLAB and GitHub to perform my project, because it seems previously I programmed a lot but never kept record for it. Working at night and other irregular hours, I found it sad that these attempts and efforts are sometimes untracked. Thus, I introduced this to ensure my attempts are tracked for my own record.

From the GitHub graph (Figure ), I worked continuously for this new analysis project every week for tens of hours from April 13th 2016 when I was introduced to the traditional method.

From the GitHub punch card (Figure ), it seems I coded for this project spread out the days, mostly at night and early morning. This might be odd but to me, coding at night is the most productive since I found it serene and focusing.

In summary, I put in considerable amount of effort and time into this project, and I really value the trust and responsibilities Prof. Olavarria and Dr. Adrian gave me. I sincerely hope my endeavor and effort do help facilitate the analysis in our lab!

Here is the GitHub repository of all the codes and my progress (Figure ): <https://github.com/doerlbh/OLab_IntanEphys/>

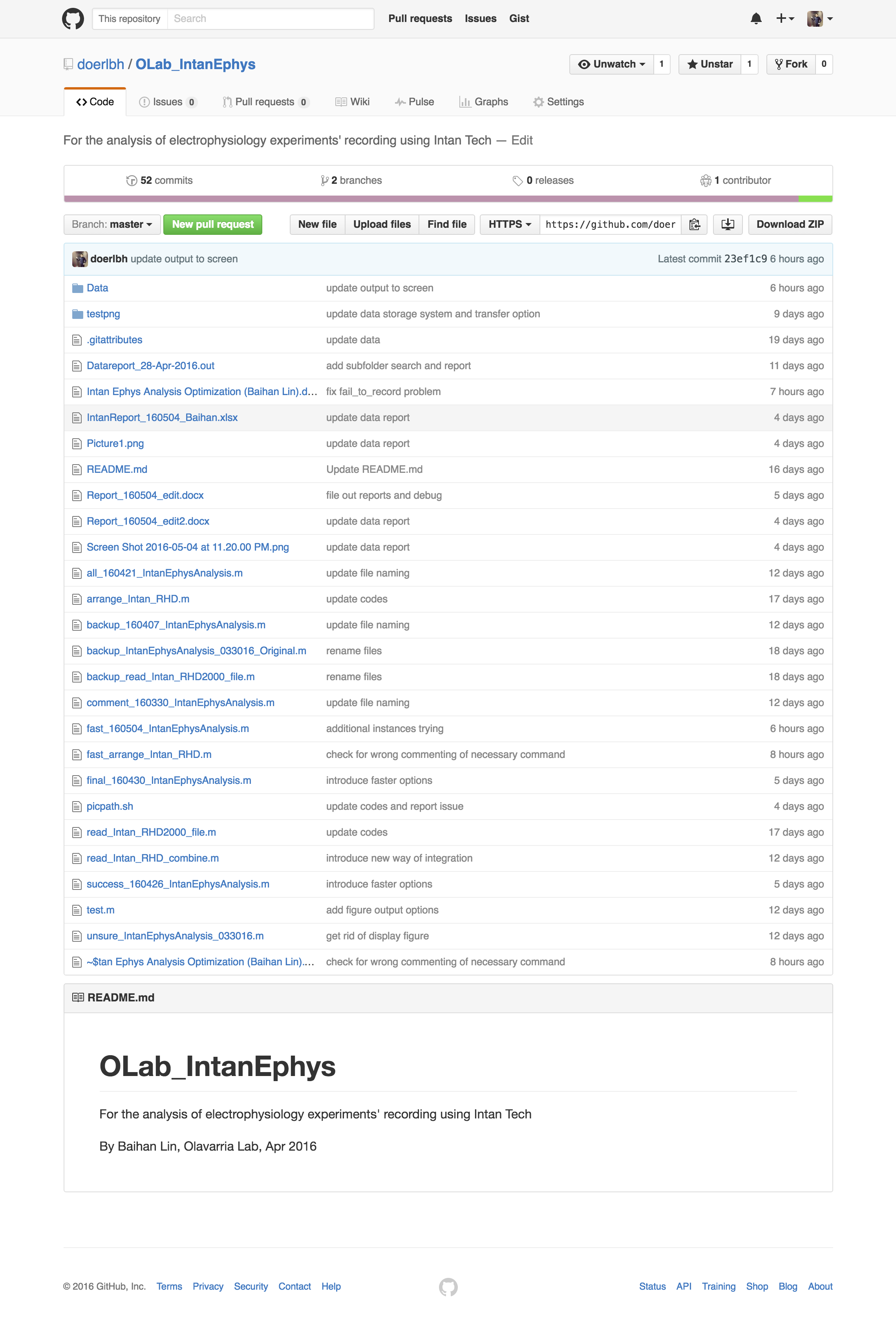
****

Figure . Screenshot of GitHub repository

**Problem Identification:**

**(What to do)**

To study the effect of sensory deprivation on brain development, Using a combination of transneuronal tracing, in situ hybridization for the immediate early gene Zif268 and electrophysiological recordings, our lab recently showed that the primary visual cortex (V1) in pigmented rats has ODCs, and these ODCs correlate with callosal inputs from the opposite hemispheres. Using similar methods, my project aims to understand the effect of monocular deprivation (MD) on the newly discovered system of ODCs in rat visual cortex. However, introducing a new system of electrophysiology, RHD2000 Amplifier Evaluation System by Intan Technologies, we need a new way of analyzing its corresponding data format.

The RHD2000 Amplifier Evaluation System is a modular family of open-source hardware and software that allows users to record biopotential signals from up to 256 low-noise amplifier channels using RHD2000 digital electrophysiology chips from Intan Technologies. As shown in Figure , A USB interface board connects to a host computer via a standard USB cable. Small amplifier boards connect to the interface board via thin, flexible all-digital cables that may be daisy-chained to form robust connections up to 10 meters in length.



Figure , RHD2000 Amplifier Evaluation System

As shown in Figure , Open-source, multi-platform GUI software controls the operation of the amplifiers and streams data to the screen and to disk in real time at user-selected sampling rates from 1 kS/s to 30 kS/s.

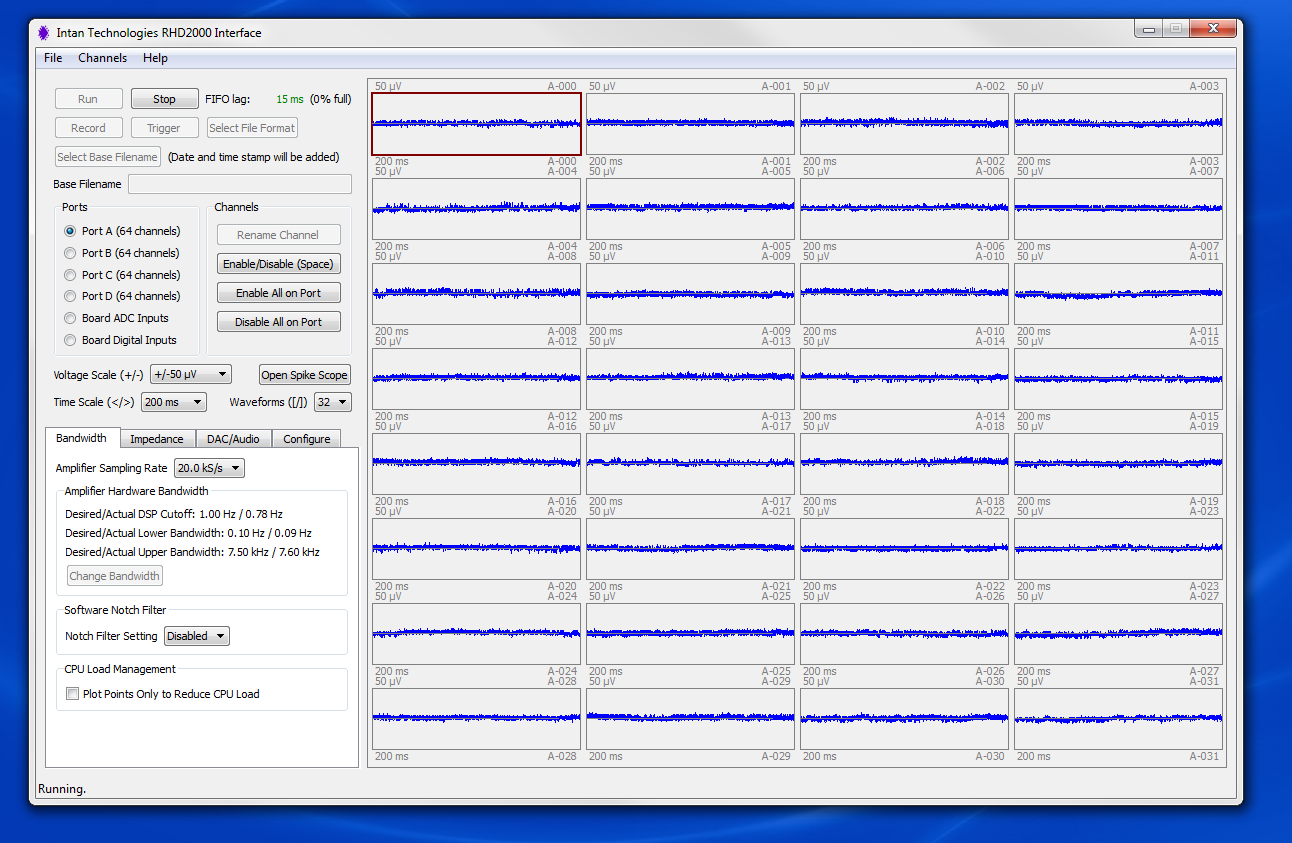


Figure , RHD2000 GUI interface

The RHD2000 Evaluation System allows users to perform the following functions:

* Monitor and record live signals from 16 to 256 low-noise amplifier channels using RHD2000 biopotential amplifier chips.
* Reconfigure amplifier bandwidths and sampling rates from software.
* Measure in situ electrode impedances (both magnitude and phase) at arbitrary frequencies with the click of a button.
* Use eight on-board digital-to-analog converters (DACs) to reconstruct analog waveforms from selected amplifier channels with <1 ms latency.
* Monitor audio of any two amplifier signals using a stereo "line out" jack.
* Record up to eight auxiliary analog inputs and 16 digital inputs synchronized with amplifier data.

These generated binary datasets which cannot be interpreted easily. And we need to use MATLAB in order to decipher the information and perform our customized analysis.

**Traditional Solution:**

**(Efficiency to improve)**

The traditional solution developed by Intan Technologies and Dr. Adrian Andelin consists of two parts: reading the binary datasets and plotting with analysis.

Step 1: read Intan files

Intan files consists of binary information like following (Figure ):

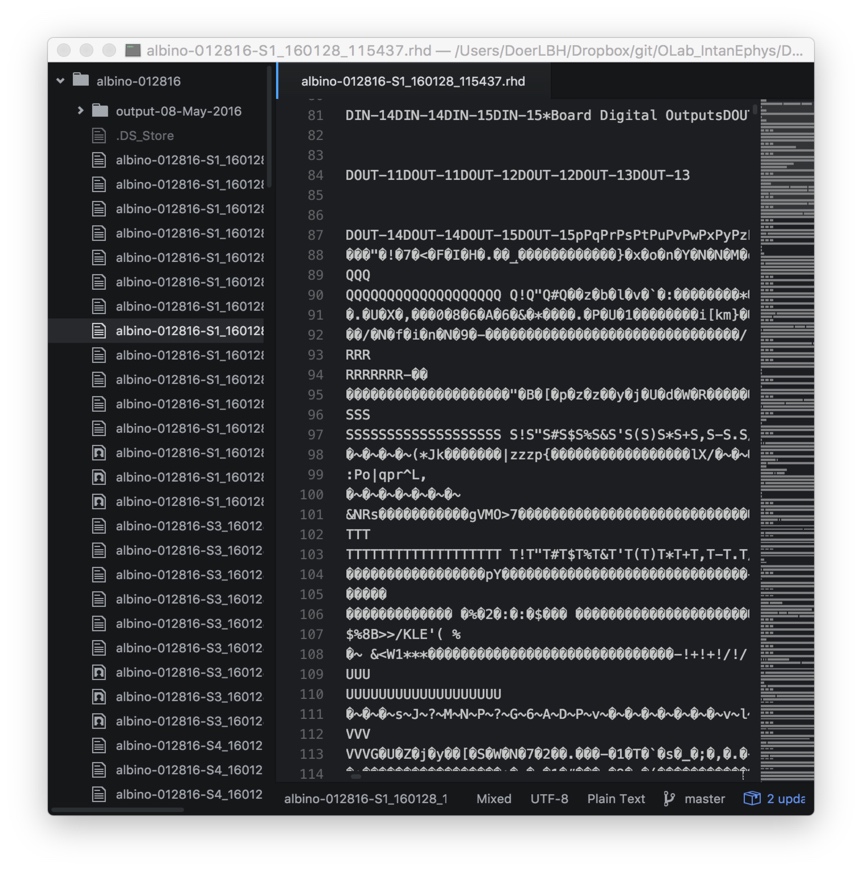
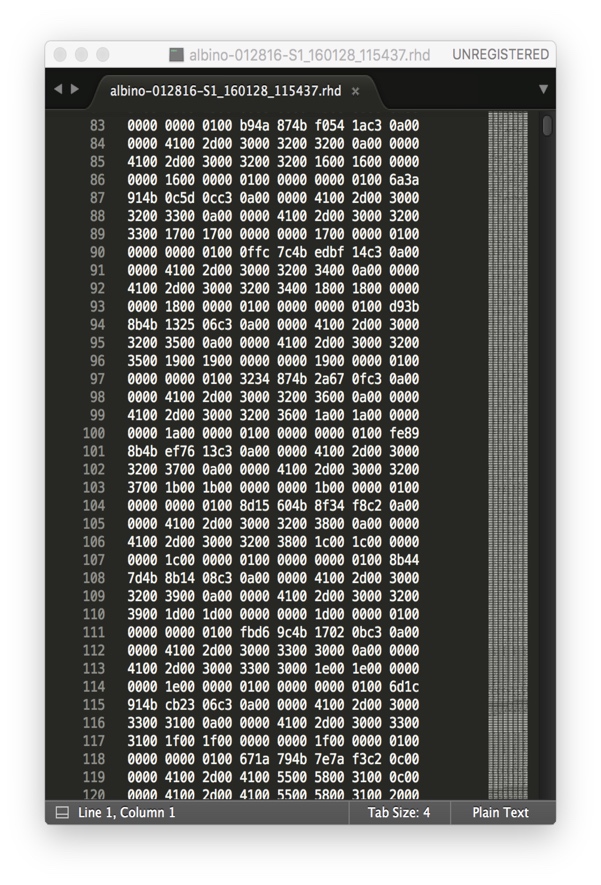
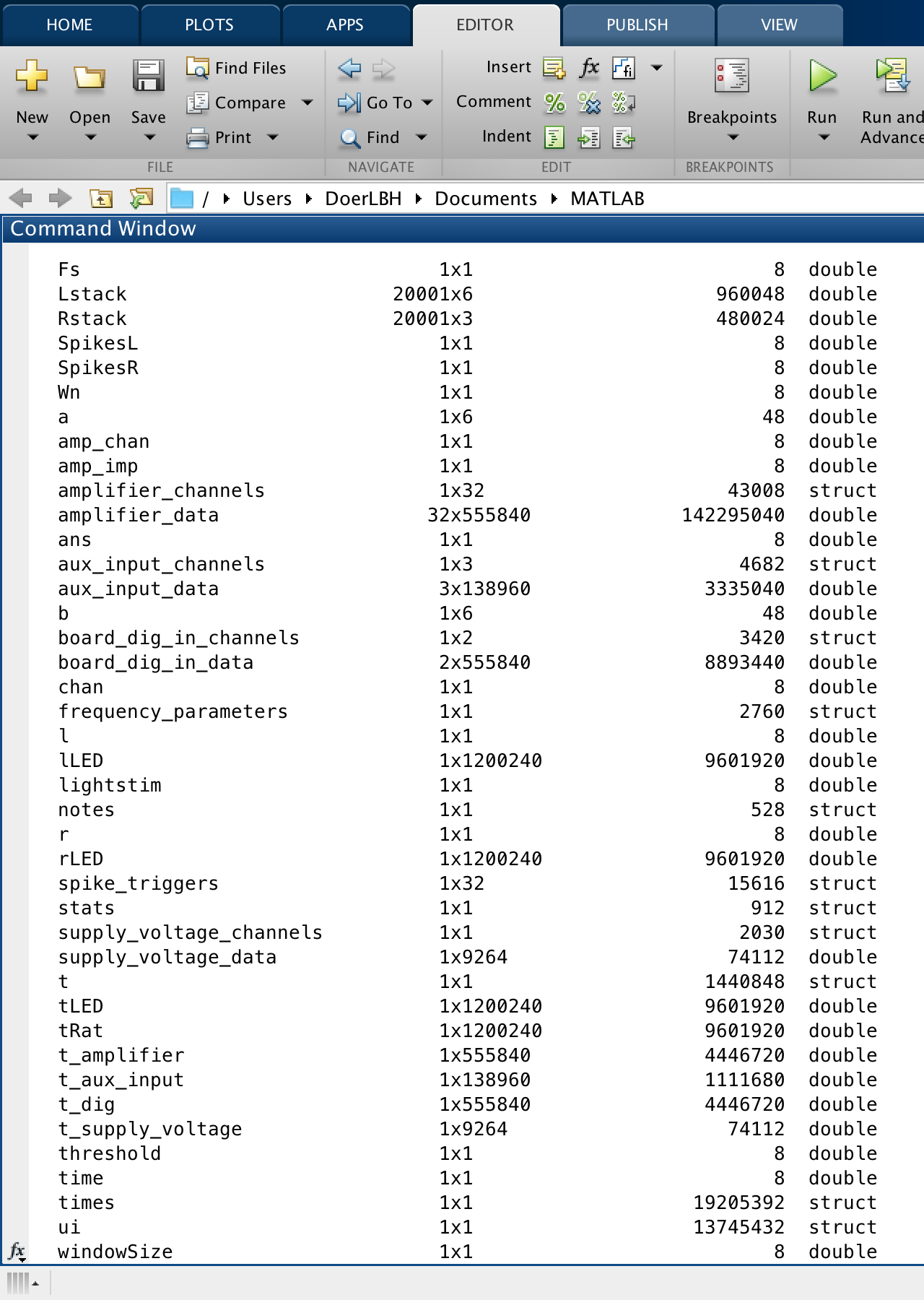
 

Figure . .rhd files are shown in binary (left) or hex (right), either unreadable

Thus, with R



Fs

Lstack

Rstack

SpikesL

SpikesR

Wn

a

amp\_chan

amp\_imp

amplifier\_channels

amplifier\_data

ans

aux\_input\_channels

aux\_input\_data

b

board\_dig\_in\_channels

board\_dig\_in\_data

chan

frequency\_parameters

l

lLED

lightstim

notes

r

rLED

spike\_triggers

stats

supply\_voltage\_channels

supply\_voltage\_data

t

tLED

tRat

t\_amplifier

t\_aux\_input

t\_dig

t\_supply\_voltage

threshold

time

times

ui

windowSize

amplifier\_channels 1x32 43008 struct

amplifier\_data 32x555840 142295040 double

aux\_input\_channels 1x3 4682 struct

aux\_input\_data 3x138960 3335040 double

board\_dig\_in\_channels 1x2 3420 struct

board\_dig\_in\_data 2x555840 8893440 double

frequency\_parameters 1x1 2760 struct

notes 1x1 528 struct

spike\_triggers 1x32 15616 struct

supply\_voltage\_channels 1x1 2030 struct

supply\_voltage\_data 1x9264 74112 double

t\_amplifier 1x555840 4446720 double

t\_aux\_input 1x138960 1111680 double

t\_dig 1x555840 4446720 double

t\_supply\_voltage 1x9264 74112 double

amplifier\_data 32x555840 142295040 double

aux\_input\_data 3x138960 3335040 double

board\_dig\_in\_data 2x555840 8893440 double

supply\_voltage\_data 1x9264 74112 double

t\_amplifier 1x555840 4446720 double

t\_aux\_input 1x138960 1111680 double

t\_dig 1x555840 4446720 double

t\_supply\_voltage 1x9264 74112 double

/Users/DoerLBH/Dropbox/git/OLab\_IntanEphys/Data/test

%% Begin with opening file to be analyzed

clear all %Starting a new analysis so we want to eliminate all old variables

close all

%

%First we import the data using:

%read\_Intan\_RHD2000\_file = Opens the MATLAB file browser UI to locate the

%file of interest. Afterward it reads header info and establishes basic

%variables from the .rhd file

%

read\_Intan\_RHD2000\_file

%

%From function info: % Reads Intan Technologies RHD2000 data file generated by evaluation board

% GUI. Data are parsed and placed into variables that appear in the base

% MATLAB workspace. Therefore, it is recommended to execute a 'clear'

% command before running this program to clear all other variables from the

% base workspace.

%check what variables have been imported, especially if youre unsure

%whether accessory amplifier channels were disabled or not during recording

%% Establishing some basic variables from values pulled in by above function

amplifier\_channels(1) %Channel data is being collected on (on

%preamplifier this would be 'A-004'

%Above command gives data output about the channel on which data is being

%collected

%% Variable name changes below to simplify:

tRat = t\_amplifier; %time variable for ephys data

tLED = t\_dig; %time variable for LED data

ui.ratData = amplifier\_data(1,:);

lLED = board\_dig\_in\_data(1,:);

rLED = board\_dig\_in\_data(2,:);

% %% Check variables look right-- plot should be identical to last one

% figure % for spike detection

% hold on

% plot(tRat, ui.ratData,'blue')

% plot(tLED,lLED,'red') %max makes red lines continue across top half of vertical axis

% plot(tLED,rLED,'green')

% xlabel 'time (s)'

% ylabel 'amplitude (A.U.)'

% legend('Raw Data', 'Left Eye LED','Right Eye LED')

%% filter data

Wn = 300/10000; % Normalized cutoff frequency

[b,a] = butter(5,Wn,'high');

ui.ratData = filtfilt(b,a,ui.ratData);

%% invert signal data for thresholding

ui.ratData = ui.ratData.\*(-1);

%% Setting threshold for spikes and finding light ON times

threshold = 20;

Fs = 20000; %amplifier\_sample\_rate

windowSize = Fs \* 0.05; %creates our time interval by taking the 20k

% sampling rate at which the data was collected and converts it to

% timestamps collected every millisecond, in other words the value of

% windowSize is 1 ms.

% Finding all spikes in recording:

ui.spikes = diff(ui.ratData > threshold) > 0.1;

%% Plot Again with spikes showing & correct time axis now:

t=length(tRat)-1

figure % for spike detection

hold on

plot(tRat(1:t), ui.ratData(1:t),'blue')

plot(tRat(1:t), ui.spikes\*max(ui.ratData),'black')

plot(tLED(1:t),lLED(1:t)\*80,'green') %max makes red lines continue across top half of vertical axis

plot(tLED(1:t),rLED(1:t)\*80,'red')

xlabel 'time (s)'

ylabel 'amplitude (A.U.)'

legend('Raw Data', 'Spikes', 'Left Eye LED','Right Eye LED')

%% Count spikes during each LED stimulation

SpikesL = sum(ui.spikes.\*lLED(1:end-1))

SpikesR = sum(ui.spikes.\*rLED(1:end-1))

%% This creates a whole lot of extra light related variables, but unsure if they are actually useful

lightstim = 199; %change?

ui.leftLEDon = diff(lLED < -lightstim)>0.1;

ui.rightLEDon = diff(rLED < -lightstim)>0.1;

% times.leftLED = find(leftLED == 500);

% times.rightLED = find(rightLED == 500);

%rewrite:

times.lLEDon = find(lLED == 1);

% Gives all time points that left LED is on

%result is a vector 1 x 80299

times.lLEDoff = find(lLED == 0);

% Gives all time points that left LED is off

%result is a vector 1 x 1119941

times.rLEDon = find(rLED == 1);

times.rLEDoff = find(rLED == 0);

% the above code will break out the time points when the LED is on for each

% side and when it is off. Next step:

% Need to ask it to count how many times ui.spikes takes place

% during each LEDon segment

%% gets light "on" times into one array

times.lLEDstart = times.lLEDon(diff(times.lLEDon)>Fs\*0.05);

%results in three specific time points for this LED

times.rLEDstart = times.rLEDon(diff(times.rLEDon)>Fs\*0.05);

% this turns the times.xLEDon into a list of the points when the LED turned

% on \*\*\*Use this for making a raster plot\*\*\*\*

%results in two specific time points for this LED

%% create raster plots-Left Stim

% Error: Subscript indices must either be real ve integers or

% logicals.

for l = 1:length(times.lLEDstart)

% collects window of data each time the light stimulus initiated

windowSize = round(Fs\*0.5); % window size in samples

ui.Lrastercell{l} = ui.spikes(times.lLEDstart(l) - windowSize:times.lLEDstart(l) + windowSize);

end

%In original script, variable that's equivalent to 'times.lLEDon' is e.g.

% a 23x1 double that includes only the start times for light turning

% on. Maybe be better to use times.lLEDstart?

%times.lLedon in this script gives the chunks when led was on, aka a

%1x80299 double

t.Lraster = transpose((1:length(ui.Lrastercell{1}(:)))/Fs);

t.Lraster = repmat(t.Lraster,1,length(times.lLEDstart));

% creates time vector for raster

times.Lrasterlight = 1:(length(times.lLEDstart));

t.Lrasterlight = ones(1,(length(times.lLEDstart)))\*windowSize/Fs;

% creates dashed line for indicating stim onset on raster plot

ui.Lraster = horzcat(ui.Lrastercell{:});

% concatenates cell array into a double

Lstack = repmat(1:length(times.lLEDstart),length(t.Lraster),1);

% creates transform for stacking windowed data for the raster plot

%% create raster plots-Right Stim

% Error: Subscript indices must either be real ve integers or

% logicals.

for r = 1:length(times.rLEDstart)

% collects window of data each time the light stimulus initiated

windowSize = round(Fs\*0.5); % window size in samples

ui.Rrastercell{r} = ui.spikes(times.rLEDstart(r) - windowSize:times.rLEDstart(r) + windowSize);

end

t.Rraster = transpose((1:length(ui.Rrastercell{1}(:)))/Fs);

t.Rraster = repmat(t.Rraster,1,length(times.rLEDstart));

% creates time vector for raster

times.Rrasterlight = 1:(length(times.rLEDstart));

t.Rrasterlight = ones(1,(length(times.rLEDstart)))\*windowSize/Fs;

% creates dashed line for indicating stim onset on raster plot

ui.Rraster = horzcat(ui.Rrastercell{:});

% concatenates cell array into a double

Rstack = repmat(1:length(times.rLEDstart),length(t.Rraster),1);

% creates transform for stacking windowed data for the raster plot

%% Before running next part, need to find how many times light goes on,

% do this by checking the variable "times.lLEDstart" and "times.rLEDstart".

% It will either show the exact values for the start times or will indicate

% how many different light on times there are, if trials exceeds ~5.

times

%% this number needs to be input as last value in reshape function below:

ui.LrasterStack = reshape(ui.Lraster,20001,3);

ui.RrasterStack = reshape(ui.Rraster,20001,2);

%% Check plot to verify reshape has been applied appropriately to LEFT data:

figure % creates raster plot

plot(t.Lraster,ui.LrasterStack+Lstack);

hold on

plot(t.Lrasterlight,times.Lrasterlight,'-.black');

hold off

ylabel 'trial number'

xlabel 'time (s)'

%% Check plot to verify reshape has been applied appropriately to RIGHT data:

figure % creates raster plot

plot(t.Rraster,ui.RrasterStack+Rstack);

hold on

plot(t.Rrasterlight,times.Rrasterlight,'-.black');

hold off

ylabel 'trial number'

xlabel 'time (s)'

%% Lastly, get spike averages for each eye

stats.spikes.Laveon = sum(sum(ui.LrasterStack(windowSize:end,:)))/length(times.lLEDstart);

% % calculates average number of spikes after light turned on

stats.spikes.Laveoff = sum(sum(ui.LrasterStack(1:windowSize,:)))/length(times.lLEDstart);

% % calculates average number of spikes preceding light onset

stats.spikes.Laveon

stats.spikes.Laveoff

%% Lastly, get spike avwerages for each eye

stats.spikes.Raveon = sum(sum(ui.RrasterStack(windowSize:end,:)))/length(times.rLEDstart);

% % calculates average number of spikes after light turned on

stats.spikes.Raveoff = sum(sum(ui.RrasterStack(1:windowSize,:)))/length(times.rLEDstart);

% % calculates average number of spikes preceding light onset

stats.spikes.Raveon

stats.spikes.Raveoff

**Bibliography:**

Intan Technologies, <http://intantech.com/RHD2000_evaluation_system.html>

Driscoll, T. A. (2009). *Learning MATLAB*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Feng, J. (2004). *Computational neuroscience: Comprehensive approach*. Boca Raton: Chapman & Hall/CRC.

**Special Acknowledgement:**

Thank Dr. Adrian Andelin for giving me the opportunity to analyze her data! In order to present her the most convincing data, I am lucky to review, utilize and summarize my ability of information searching, computational modeling and conception actualization. In the entire process of developing ideas into software, I learnt a lot from the questions and reflections at every step and had the fortune to enjoy the excitement of tackling the challenges one by one! Her tolerance on the deadline and encouragement on the difficulty give me the strongest fortitude to proceed! Her usual strict and meticulous style forces me to think deeper and striver closer to perfectionism!

Thank Prof. Jaime Olavarria for introducing me with insightful lectures into this dynamic field of biopsychology and neuroscience and letting me getting involved in the challenging projects of his warm lab! (It is my honor to be able to be peer TA in his class next quarter! Thank him to give me courage to challenge Psychology Honors Program next quarter!)

Thank my friends in the lab, Adrian and Melissa who support me with sunshine! Adrian never fails to encourage me whenever I make any mistake or face some challenge. He taught me how to flexibly tackle incidents in research but still gave me chance to shoulder experimental rats.

Thank University of Washington for giving us the platform to scientifically explore practical academic problems interdisciplinarily!

I will continue the voyage of exploring the infinite realm of neuroscience in my academic career fearlessly.

Baihan Lin

April 2016