Catamaran Spike Sorting User's Manual

Version 1.3, 27 February 2012

Douglas M. Schwarz
Neurobiology & Anatomy, University of Rochester
douglas.schwarz@rochester.edu

1 Introduction

Catamaran is a MATLAB program to perform classification of neural spike waveforms recorded by tetrodes. It reads the spike waveforms from a Neuralynx NTT file, manipulates them appropriately and produces a plain text file containing the classification of each recorded event. The classification scheme is not sophisticated enough to run unsupervised so interaction by the user is required. A convenient graphical interface is provided.

If this software is used to analyze data that is included in a publication, please cite the following paper:

Schwarz D, Zilany M, Skevington M, Huang N, Flynn B, Carney L. Semi-supervised spike sorting using pattern matching and a scaled Mahalanobis distance metric. Journal of Neuroscience Methods 2012 (to be published).[1]

It contains greater detail on the algorithms implemented in Catamaran.

2 System Requirements

Catamaran was developed on MATLAB version 7.8 (R2009a). It should work on versions as far back as 7.6 (R2008a), though this has not been tested. It will not work on previous versions since it requires the new style classes introduced in R2008a. It also requires the Signal Processing Toolbox and the Statistics Toolbox. There is one processing algorithm that requires the Wavelet Toolbox, but it can be avoided.

The spike waveform classification algorithms can require quite a bit of memory when you have a large number of events to be processed, but this has been mitigated by the use of block processing techniques.

Catamaran is platform independent and has been tested on Windows XP, Windows 7, Macintosh OS X and Ubuntu GNU/Linux.

3 Overview

The problem of sorting action potentials (spikes) by neuron is an ongoing one with each research laboratory developing particular techniques that work for them. We desired sorting software that was relatively fast (taking minutes rather than hours to sort 100,000 events) and did not mind if the clustering could not run unsupervised.

With these principles in mind, Catamaran was developed with the following requirements:

1. Read spikes from Neuralynx NTT files. Assume four waveforms per event.
2. Select a subset of the spikes on which to determine optimal algorithms and parameters.

[1] `http://dx.doi.org/10.1016/j.jneumeth.2012.02.013`

3. Eliminate any spike waveforms that are clipped.
4. Align the spike waveforms.
5. Transform the raw spike waveforms into feature vectors.
6. Provide a number of different transformation algorithms from which to choose.
7. Display some kind of representation of feature vectors to facilitate manual determination of number of clusters.
8. Cluster the feature vectors.
9. Provide a number of clustering algorithms.
10. Allow clusters to be merged.
11. Apply the selected algorithms to the full set of spike waveforms.
12. Save the resulting clustering in some kind of file.
13. Build this functionality into an application with an easy-to-use graphical interface.

From item 2, we see that we have to select a subset of all the spike waveforms because it would be too time consuming to experiment with transformation and clustering algorithms on the whole set. Initially, this subset was taken as a certain number of events (typically 10 or 20 thousand) from the beginning of the set. We then realized that the spike waveforms from a single neuron might change shape during the course of the experiment (maybe two hours in duration) so we needed to sample events uniformly distributed throughout the whole set. However, this doesn't quite work either because one of the criteria for determining if a cluster truely represents a single neuron involves looking at the times between subsequent spikes. We ended up implementing a scheme that selects blocks of contiguous events where the blocks are distributed through the whole set.

4 Operation

The program is easy to use. Install it by placing the files `catamaran.m` (the main program) and `nttfiles.m` (a utility class for reading NTT files) anywhere on your MATLAB path. Start the program by typing `catamaran` at the MATLAB prompt. You can pass in the name of an NTT file,

```
>> catamaran('sample.ntt')
```

or leave it out and then open a file from the File menu:

```
>> catamaran            followed by            File → Open...
```

The system will respond by displaying the main graphical interface shown in Fig. 1.

The transformation algorithm reduces the four waveforms of each event to a small number of values which is the feature vector for that event. (In this example, the feature vectors of RPS have four elements, but algorithms vary and others produce feature vectors of different lengths.)

The display is simply a grid of scatter plots where the elements of the feature vectors are plotted against each other pairwise. For example, the scatter plot in column 1, row 2 is a plot of v_1 vs. v_2, where v_i is the set of all the i^{th} elements. There is no point in plotting the elements against themselves so the plots on the diagonal show histograms of each element instead. (See the MATLAB documentation for the function `gplotmatrix` for additional details.)

This is not a perfect visual representation of the (at least) four-dimensional feature vectors, but it seems to work well enough to determine the number of clusters.

At this point the user is free to experiment with different Transformation and Clustering algorithms and # Clusters settings in order to get the desired result. This example clearly shows two clusters so we set # Clusters to 2 and press Compute. The result is shown in Fig. 2.

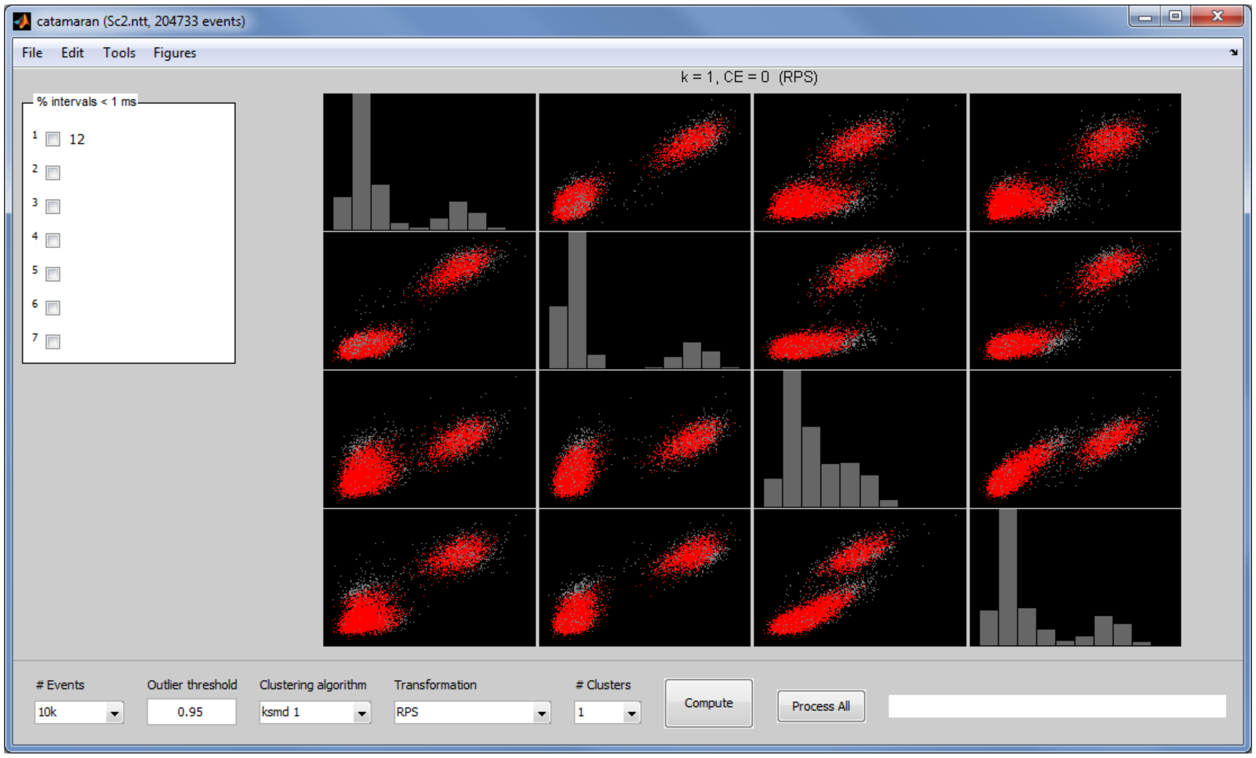


Figure 1: Catamaran Graphical Interface with data

Note the white box labeled "% intervals < 1 ms". This tool allows you to rate the quality of each cluster. The number label is the percentage of first order intervals that are less than 1 ms, which is one indicator of cluster quality. If that value is less than approximately 0.5% then there is a good chance that the cluster contains spikes from a single neuron. By clicking in the check boxes a button appears which, when pressed, rotates through the rating choices of "Good", "Excellent" and "Multi-unit". These settings have no bearing on any of the clustering or transformation algorithms, but are merely recorded in the output file.

A number of additional tools (available in the Tools menu) have been provided to help determine cluster quality. See section 5.3 for details.

Once the user is satisfied, a press of Process All will apply the chosen algorithms to the full set of events.

5 Menus

The term *current folder* refers to the parent folder of the current NTT file.

5.1 File menu

Open. . .
: Select and open an NTT file. Once selected, it will be opened and displayed immediately.

Save As. . .
: Save the completed clustering of the whole set of events into a plain text file. The format consists of 11 header lines followed by the cluster numbers, one per line. Here is an excerpt:

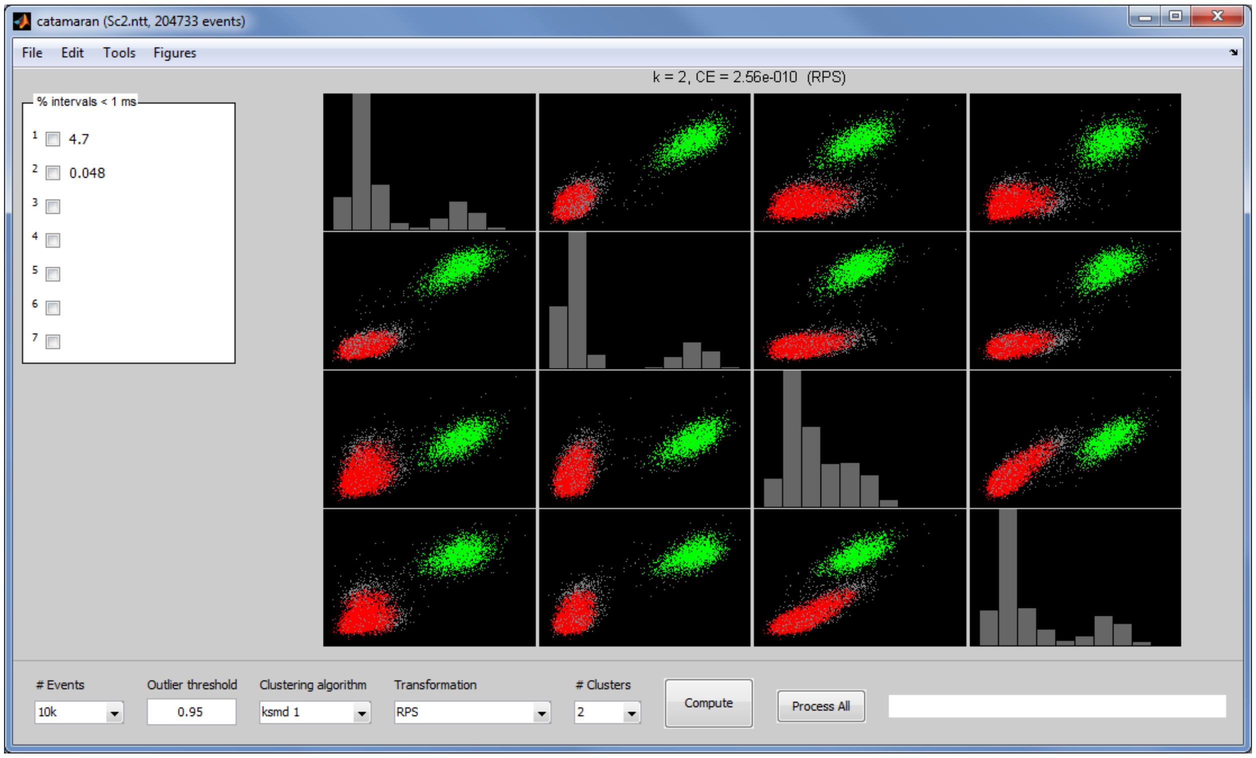


Figure 2: Catamaran Graphical Interface with clustered data

```
% Number of clusters = 4
% Time = 20-Jan-2011 14:14:53
% Transformation algorithm = RPS
% Clustering algorithm = ksmd 1
% Cluster 1 is unrated
% Cluster 2 is unrated
% Cluster 3 is unrated
% Cluster 4 is unrated
% Cluster 5
% Cluster 6
% Cluster 7
1
2
...
```

Show in Explorer (Windows only)
: Open the current folder in Windows Explorer.

Show in Finder (Macintosh only)
: Open the current folder in the Finder.

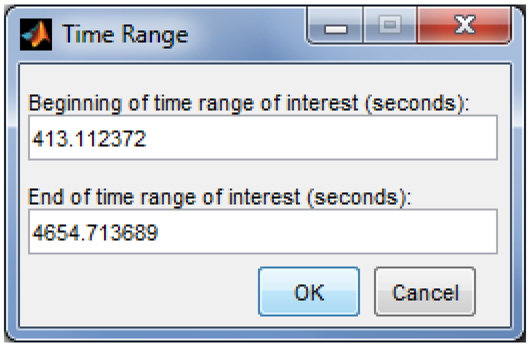
Open Notes...
: Open the file `Notes.txt` in the current folder with a simple text editor. If it doesn't already exist it will be created. This text editor is modal (meaning you must close it to interact with any other windows) and `Notes.txt` is updated immediately when closed.

Close Catamaran
: Close the application. You will be presented with a confirmation dialog.

5.2 Edit menu

Time Span...

Sometimes it is desired to limit the analysis to a span of time less than the full time range of the data. This limited time span can be set with the Time Span dialog (shown in Fig. 3). The initial values indicate the full range of the data in seconds; enter the restricted range which must be contained entirely within the full range.



Beginning of time range of interest (seconds): 413.112372

End of time range of interest (seconds): 4654.713689

Figure 3: Time Span dialog

Preferences...

Open the Preferences dialog (shown in Fig. 4). There are two preferences items and their values are saved between sessions.

Default Input Folder

Specify the default folder for opening an NTT file when Catamaran is first started. (Subsequent open operations will start in the current folder.) You can type a folder location or pick one by pressing Browse... This is useful if you have a standard place in your file system where you store all your NTT files.

AutoSave

AutoSave will cause the clustering results text file to be saved automatically when the Process All function has completed. The file will be saved in the current folder and is assigned a name which matches the NTT file, but with the extension changed to "`.clu`". *Subsequent processing of the same NTT file will cause the cluster file to be overwritten.*

If AutoSave is turned off, the cluster file will only be saved when Save As... is selected from the File menu.

5.3 Tools menu

These tools are also available from a popup menu by right-clicking on the gray area surrounding the scatter plots.

Detach plot

Copy the current scatter plot grid to a new figure. Once there, you can save it, zoom into plots, etc.

Plot spike waveforms

For each cluster, create a figure which displays the spike waveforms assigned to that cluster. In addition to the edit box, the arrow and number pad keys can be used to move through the population of spikes. See Fig. 5.

Wavehist all

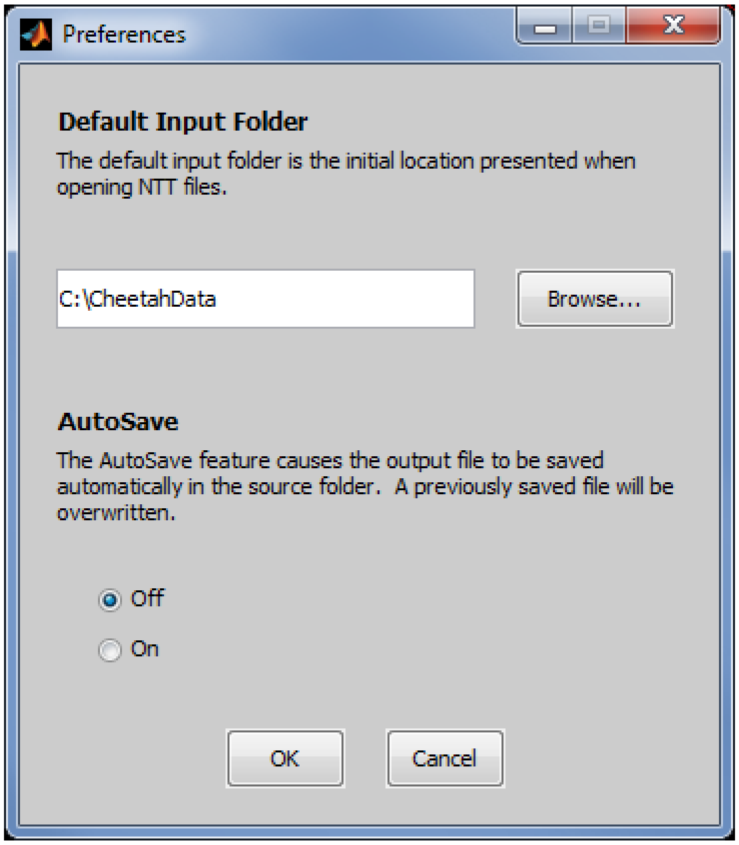
Produce a visualization of a histogram at each sample time for all the waveforms of all events. See Fig. 6.

Wavehist by cluster

Same as Wavehist all except produce a separate figure for each cluster.

Wavehist array

Same as Wavehist by cluster except all plots are on a single figure. See Fig. 7.


Preferences

Default Input Folder

The default input folder is the initial location presented when opening NTT files.

C:\CheetahData

AutoSave

The AutoSave feature causes the output file to be saved automatically in the source folder. A previously saved file will be overwritten.

[✓ Off] On

Figure 4: Preferences dialog

1st order interval histogram array

Plot histograms of first order intervals (time between subsequent spikes within and between clusters). Because of the refractory period of neurons, if you have correctly clustered the spikes from individual neurons you do not expect to see intervals less than about 1 ms. Conversely, if you see that a significant number of the intervals are less than 1 ms it is likely that the cluster contains spikes from more than one neuron. In the sample plot in Fig. 8, we can see that 4.7% of the intervals in cluster 1 are less than 1 ms whereas for cluster 2 the amount is only 0.048%. Therefore, we can conclude that cluster 1 is probably multi-unit and cluster 2 may be from a single neuron.

Merge clusters

Sometimes it is difficult to get the clustering algorithms to behave the way you'd like, especially when the clusters are close together or are different in size. It can be useful to specify a greater number of clusters than you think are really there and then merge some of them afterwards. That is the purpose of the Cluster Merge Tool. A sample is shown in Fig. 9.

The tables contain cluster entanglement (CE) values. Serving as a measure of how close two clusters are, those values are often sufficient to determine which clusters should be merged. Of course, the color names match the cluster colors in the scatter plots so that can help, also.

To indicate which clusters are to be merged simply click on the cell(s) in the upper table at their intersection. To make multiple selections hold down the Control key. In the sample table, we can see that the values of $CE_{1,2}$ and $CE_{3,4}$ are much larger than the other values in the table strongly suggesting that clusters 1 and 2 should be merged and also clusters 3 and 4. Merge by pressing Merge Selected Clusters, after which the tool appears as in Fig. 10. New superclusters will be labeled with the numbers that are the smallest of the original component cluster numbers and the other numbers will be left unused. This is reflected in the lower table which now contains placeholder values (NaN = not-a-number) in cells belonging to unused cluster numbers. If all the

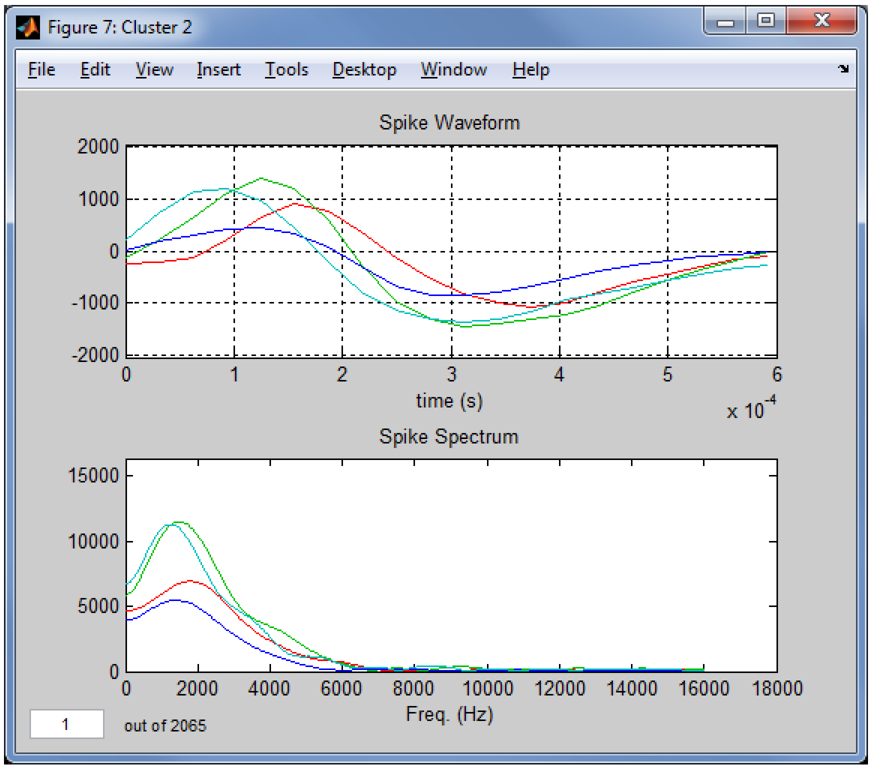


Figure 5: Plot spike waveforms

remaining values are small (just one in this example) then the merging was probably warranted.

The colors of the clusters in the scatter plots are updated, also.

There is a quirk in how tables are implemented in MATLAB. If you close the Cluster Merge Tool and then open it again, it will not highlight the selected cells in the upper table. It is usually best to press Unselect All and Merge Selected Clusters in order to start over.

Sequence plots
: To see if the feature vectors change with position in the temporal sequence you can use this tool to plot the elements of the feature vectors in sequential order.

Time plots
: These plots are similar to the Sequence plots except that the feature vectors are plotted as a function of the event time.

Stationarity plot
: This is yet another way to view the feature vectors as a function of time. In this plot, a principal components analysis is performed and the first two principal components are plotted versus time in a 3-D scatter plot. By rotating the plot around you can determine if the characteristics of the spikes are changing with time. See Fig. 11.

5.4 Figures menu

Restore Default Position
: Initially, the application is sized to nearly fill the default screen, but it can be resized and moved. This new position is saved when the application is closed. Selecting this item will return the application to its default size and position.

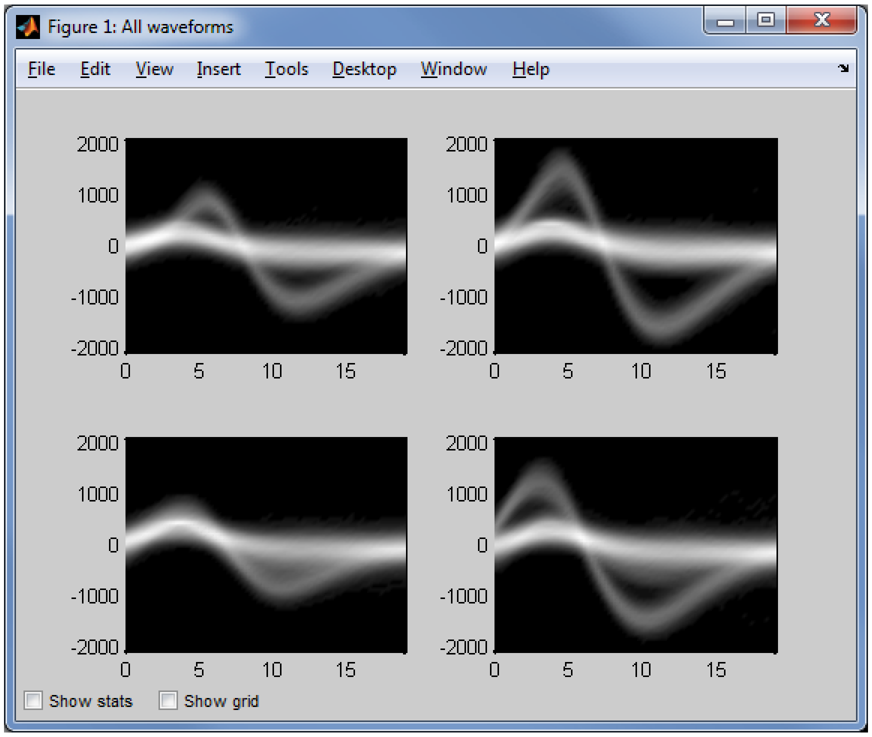


Figure 6: Wavehist all

Close All Child Figures
: In the process of exploring the clustering results it is common to open many additional figures using the supplied tools. Selecting this item will close them all at once.

6 Main Panel Controls

6.1 # Events

When exploring different ways of transforming and clustering a large number of events (perhaps hundreds of thousands) it would be far too tedious to operate on the whole set. Therefore, Catamaran samples the events in the NTT file and chooses the number of events set by this control. One of the criteria used to determine if the algorithms are doing what is desired involves looking at time intervals between subsequent spikes. For this reason, the events are not simply sampled uniformly through the full set, but are sampled in blocks of contiguous events where the blocks are uniformly distributed through the set. If the selected number of events is N then the number of blocks and the number of events per block are both approximately equal to $\sqrt{N}$.

6.2 Transformation

The purpose of the transformation algorithm is to reduce the raw spike waveforms into feature vectors. A feature vector is a set of statistics computed from the four waveforms in each event and there are many schemes used to do this. For example, if you were to take the peak value of each waveform, those four numbers would form a feature vector. Several different transformations from waveform space to feature space are implemented and described below.

RPS (Repolarization slope)
: Good spike waveforms will tend to have a strong transition from positive to negative in the repolarization region

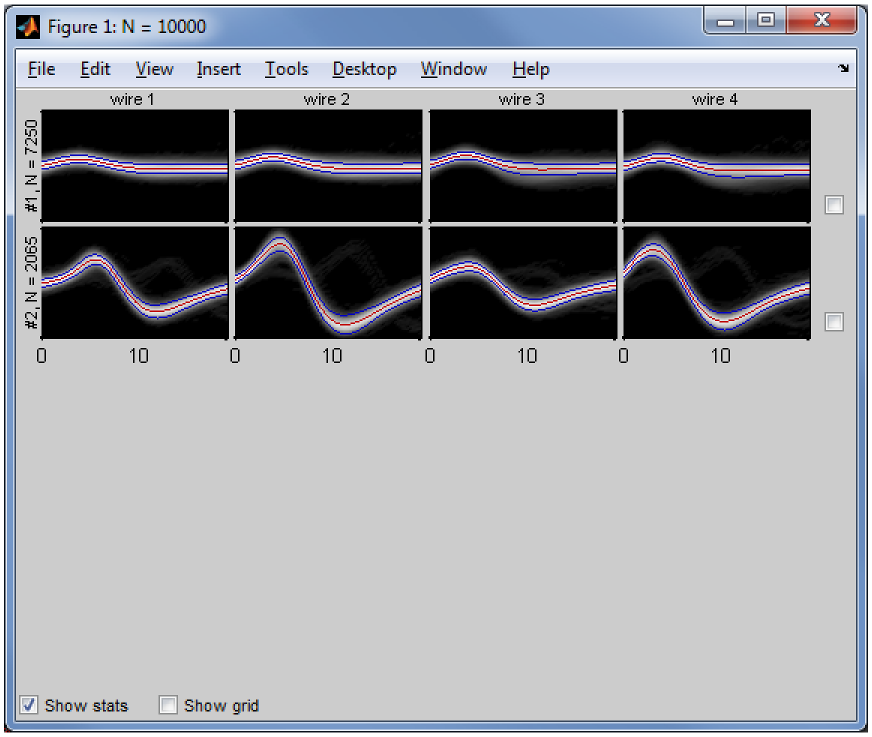


Figure 7: Wavehist array

of the waveform. This algorithm does a pattern match to search for this edge and returns the value of the best match to the pattern. The pattern used is

$$P = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & -1 & -1 & -1 & -1 \end{bmatrix} \tag{1}$$

which essentially means we are looking for a sustained positive region followed by a sustained negative region.

The pattern matching is done by flipping the pattern to create a convolution kernel,

$$K = \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2}$$

and then convolving this kernel with each spike waveform. The value of the convolution at each point indicates how well the waveform matches the pattern there. By taking the maximum of these values we compute the best match over the whole waveform. It is not necessary to determine the location of the best match, merely the amplitude. We do this for each waveform of the tetrode to obtain a 4-element feature vector.

$$\begin{aligned} v_1 &= \max\left[\{s_{1,n}\} * \{K_n\}\right] \\ v_2 &= \max\left[\{s_{2,n}\} * \{K_n\}\right] \\ v_3 &= \max\left[\{s_{3,n}\} * \{K_n\}\right] \\ v_4 &= \max\left[\{s_{4,n}\} * \{K_n\}\right] \end{aligned}$$

where $\{s_{i,n}\}$ is the set of samples of the i^{th} waveform in an event. This is repeated for each of N events to produce an $N \times 4$ matrix.

RPS-

This is the same as RPS except with the sign inverted so that the algorithm is looking for a positive slope region.

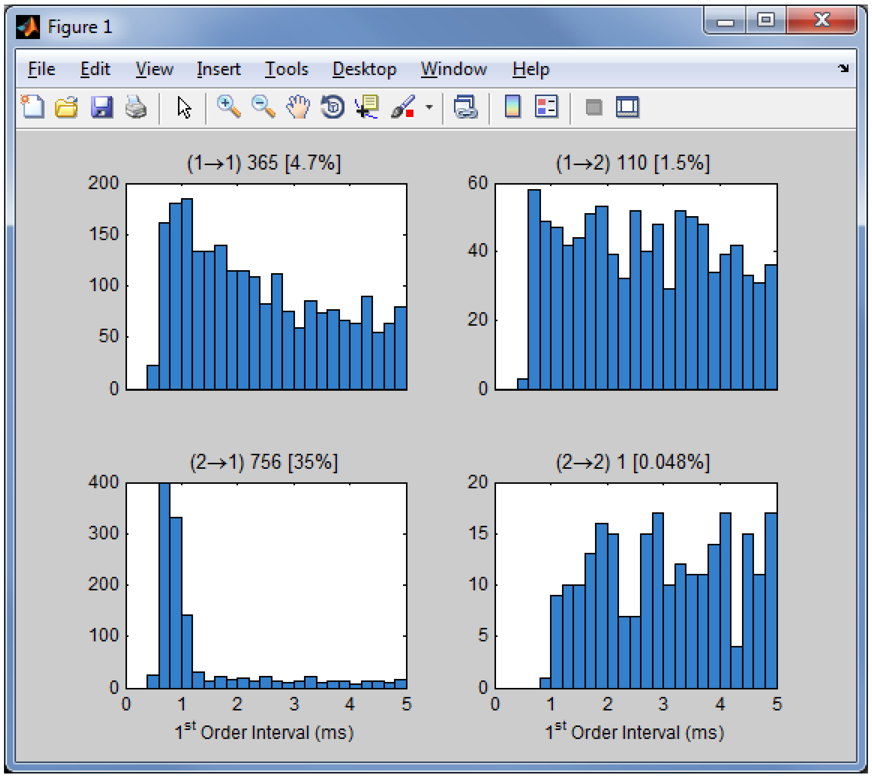


Figure 8: 1st order interval histogram array

RPS2

This is similar to RPS except each waveform is forced to be sampled at the same time value rather than picking the point of maximum negative slope of each spike waveform.

RPS2-

This is the same as RPS2 except with the sign inverted.

PCA all wires

This algorithm consists of taking each of the four waveforms from a tetrode, concatenating them into one long waveform and then performing principal components analysis (PCA) on the whole set of these long waveforms. The first four principal components are returned.

RPS/PCA

This is the same as RPS followed by application of principal components analysis. All four principal components are retained so this isn't inherently different from RPS alone, but sometimes helps with visualization.

FFT1/PCA

For each waveform we compute the Discrete Fourier Transform (using the FFT algorithm), keep the real and imaginary parts of the first frequency point, perform PCA on those 8 features and keep the first 4 principal components.

FFT2/PCA

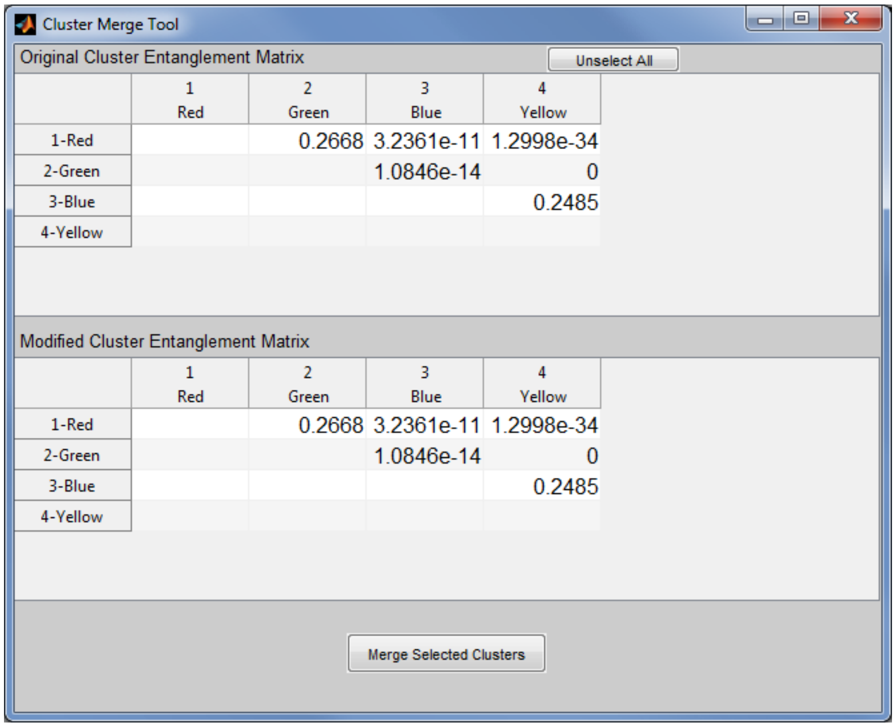
Same as FFT1/PCA except we keep the first and second frequency points.

FFT3/PCA

Same as FFT1/PCA except we keep the first three frequency points.

DCT/PCA

For each waveform we compute the Discrete Cosine Transform (DCT), concatenate the 4 waveforms into one and then perform PCA on the whole set over all events, keeping the first 5 principal components.



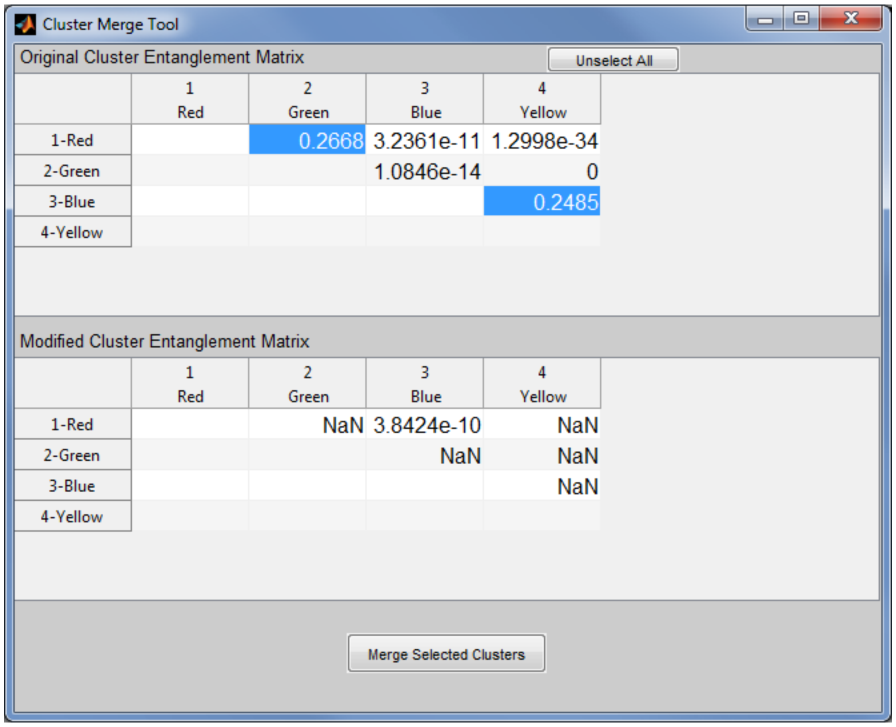
Original Cluster Entanglement Matrix

	1 Red	2 Green	3 Blue	4 Yellow
1-Red		0.2668	3.2361e-11	1.2998e-34
2-Green			1.0846e-14	0
3-Blue				0.2485
4-Yellow				

Modified Cluster Entanglement Matrix

	1 Red	2 Green	3 Blue	4 Yellow
1-Red		0.2668	3.2361e-11	1.2998e-34
2-Green			1.0846e-14	0
3-Blue				0.2485
4-Yellow				

Figure 9: Cluster Merge Tool



Original Cluster Entanglement Matrix

	1 Red	2 Green	3 Blue	4 Yellow
1-Red		0.2668	3.2361e-11	1.2998e-34
2-Green			1.0846e-14	0
3-Blue				0.2485
4-Yellow				

Modified Cluster Entanglement Matrix

	1 Red	2 Green	3 Blue	4 Yellow
1-Red		NaN	3.8424e-10	NaN
2-Green			NaN	NaN
3-Blue				NaN
4-Yellow				

Figure 10: Cluster Merge Tool after merging

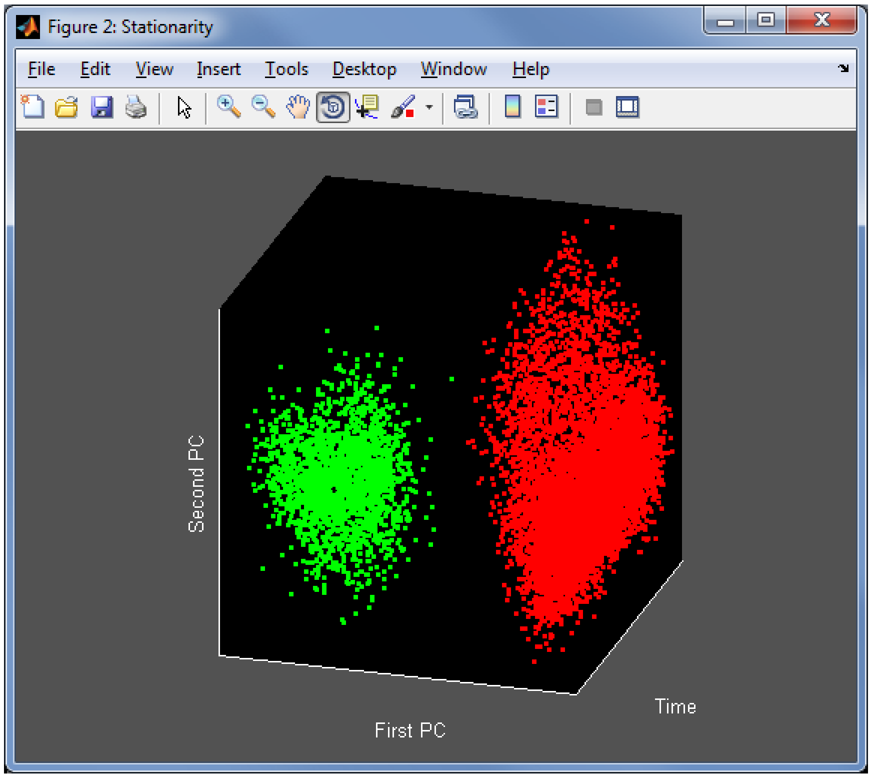


Figure 11: Stationarity plot

FFT/KS
: We compute the DFT, keeping the first two frequency points (real and imaginary parts) producing 16 values per event. Then we compute the Kolmogorov-Smirnov (KS) test statistic for each of the 16 columns in that matrix. The KS statistic is a measure of how different from normal the test distibution is. We retain the 4 columns with the highest values of the KS statistic, i.e., the 4 columns that are the most non-normal.

DCT/KS
: Same as FFT/KS except we compute the DCT instead of the DFT.

KS only
: We perform no transformations at all, but simply compute the KS statistic for each time value in the set of all 4 waveforms over all events. We pick the 4 columns that are the most non-normal.

Bimodality4
: Bimodality is a measure of how bimodal a distribution is. Like KS only, we perform no transformations, but simply compute the bimodality statistic of the distibutions of values at each time point in the set of all 4 waveforms. We retain the 4 time points that are the most bimodal. Probably only useful when only two clusters are present.

(RPS + FFT1)/PCA
: A combination of the features from RPS and the first frequency component of the DFT, followed by PCA. The first 6 principal components are kept.

Wavelets/KS
: This algorithm performs a 4-level wavelet decomposition followed by computation of the KS statistic of the resulting coefficients. The 6 most non-normal components are retained.

6.3 # Clusters

This control sets the number of clusters as determined visually from the scatter plots. The program can handle up to seven clusters. This value can also be changed with the up and down arrow keys.

6.4 Clustering algorithm

The algorithms using `ksmd` or `kmeans` are the fastest with ksmd 1 giving the best performance overall, but there are many instances in which Classic gives a better result.

Classic

This algorithm is called Classic only because it is the first one implemented. The others were developed much later.

The idea of this algorithm is connected with the concept of *cluster entanglement*. Cluster entanglement (CE) is a measure of how close two clusters are to each other. A large value means that two clusters are relatively close (perhaps "touching") and a small value means they are well separated.

This algorithm starts out by fitting a mixed Gaussian model to the array of feature vectors. That gives us an initial set of cluster centers. We then optimize the locations of the centers so that we minimize the cluster entanglement. Since this optimization is computationally intensive (read "slow") we perform it on no more than 5,000 points. Finally, we use the resulting clusters as a seed population in an algorithm much like *k*-means except that Euclidean distance is replaced by Mahalanobis distance.

ksmd α (values of α from 0–2)

This algorithm is a variant on *k*-means which uses scaled Mahalanobis distance (the "smd" in the name) for the distance metric. It calls an underlying program, `ksmd`, with a scaling parameter equal to α. This α parameter controls the degree to which the distance of each point is scaled by the volume of the cluster to which it is being measured. This prevents a large cluster from "swallowing" a smaller one that is nearby which can happen when $\alpha = 0$. Larger values of α will increase the likelihood of separating a small cluster near a large one. It uses the initialization algorithm, *k*-means++, and is much faster than Classic. Even though the final metric is the same as that used by Classic, they don't give the same results due to the initialization scheme.

kmeans

Standard *k*-means algorithm. Not recommended, included just for reference.

6.5 Outlier threshold

It is common in statistical analyses to make an attempt to identify statistical outliers and Catamaran has that capability. Once clustering has taken place, feature vectors near the periphery of each cluster can be marked. The outliers from all clusters are plotted in gray in the scatter plots.

The outlier threshold is nominally the fraction of points to remain if the cluster shape were normally distributed. It's not, of course, but it's a useful approximation. The Mahalanobis distance of each feature vector from the cluster is computed and compared with an absolute distance computed from the dimensionality of the data. Points with a Mahalanobis distance greater than this value are marked as outliers. They are marked in the output file by negating their cluster identification number. For example, if a point is computed as belonging to cluster 3, but is also an outlier, it will get a cluster number of -3.

6.6 Compute

After making changes to any of the other settings you must press this button to recompute the scatter plots and the clustering. You can also press the Enter key.

6.7 Process All

Once you are satisfied with the performance of the chosen algorithms on the sample data, press this button to apply those algorithms and settings to all the events in the NTT file. This may take a while so the progress is reported in the white message window to the right of this button.

It is important to note that the program does not load all the data and perform another transformation and clustering all at once. That would take longer and use much more memory. Rather, it processes the events in blocks of 10,000 events at a time. Furthermore, it uses the initial results as seed clusters. After each block has been transformed, the distances of each feature vector from the seed clusters are computed and each event is assigned to the nearest seed cluster. The distance metric is the same as that used for the chosen clustering algorithm.

6.8 Message display

Displays information about time-consuming computations.

7 Processing

Every time a new NTT file is read and clustered, Catamaran performs the following processing steps:

1. Read spike waveforms from NTT file.
2. Identify clipped waveforms.
3. Align the spikes.
4. Transform spike waveforms into feature space.
5. Cluster the feature vectors.
6. Plot representation of feature vectors.

When Process All has been pressed, Catamaran must do the following for each block of 10,000 events:

1. Read spike waveforms from NTT file.
2. Identify clipped waveforms.
3. Align the spikes.
4. Transform spike waveforms into feature space.
5. Cluster the feature vectors.
6. Merge specified clusters.
7. Plot representation of feature vectors.
8. Compile clustering results.

7.1 Spike Alignment

Events are recorded by the Neuralynx hardware whenever the signal level on any one of the four electrodes exceeds a set threshold. Each event consists of a recording of all four electrodes for approximately 2 ms, starting a bit before the trigger. Because of noise, the trigger point might be reached at different phases of the spike waveforms. This causes the waveforms not to line up if they are all plotted together. Some of the processing algorithms are insensitive to this random phase variation (notably RPS), but most are highly sensitive to it. As such, we have made an effort to align the spikes. It's a difficult task because of the low signal-to-noise ratio.

Our approach is a simple one:

1. Interpolate the raw waveforms to quadruple the sampling rate.
2. Low-pass filter those waveforms.
3. Find the location of the peak of each filtered waveform.
4. Compute the average peak location over all waveforms.
5. Compute the time differences between where each waveforms peak and the average peak location.
6. Shift interpolated waveforms by those time differences.
7. Downsample back to original sample rate.

For a demonstration of the efficacy of this technique see Fig. 12.

Notice how much more well defined the wavehist display is when the spike alignment is enabled.

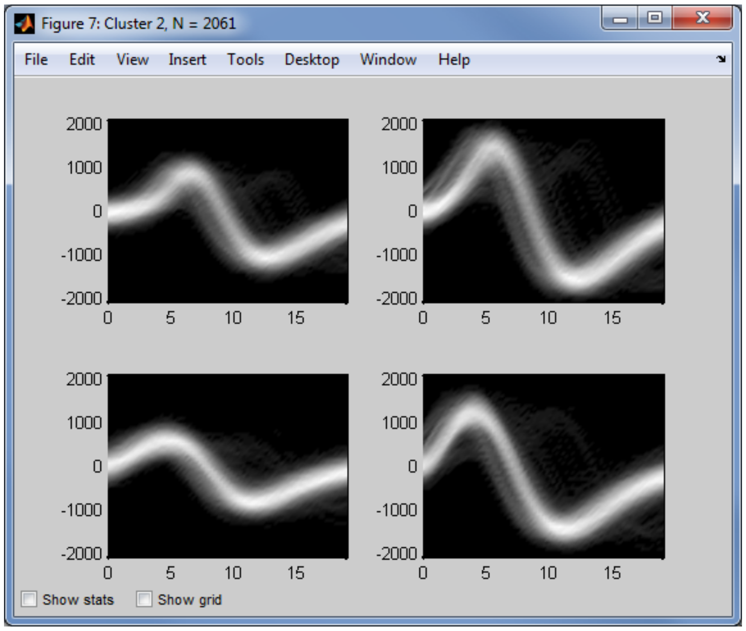


Figure 7: Cluster 2, N = 2061
File Edit View Insert Tools Desktop Window Help
2000
1000
0
-1000
-2000
0 5 10 15
Show stats
Show grid

(a) Alignment disabled

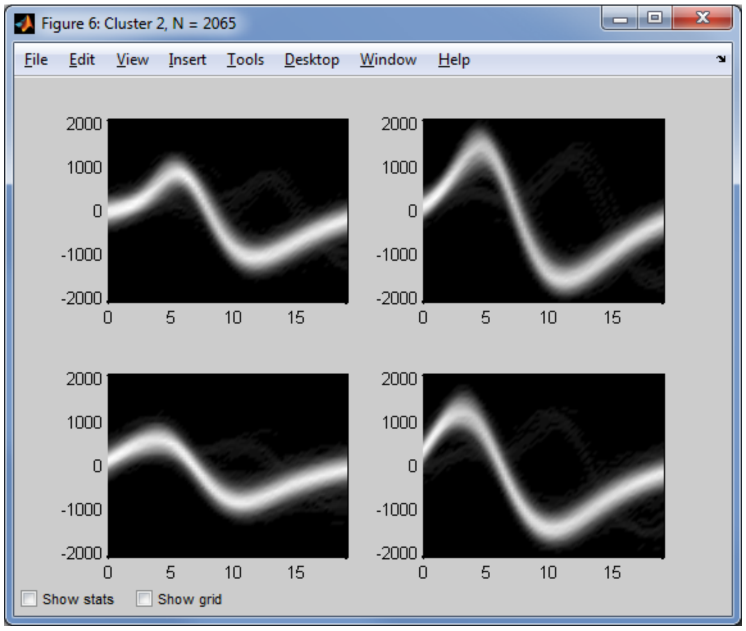


Figure 6: Cluster 2, N = 2065
File Edit View Insert Tools Desktop Window Help
2000
1000
0
-1000
-2000
0 5 10 15
Show stats
Show grid

(b) Alignment enabled

Figure 12: Spike alignment