<https://github.com/csn-le/wave_clus>

1. Create readable file with .mat format

For example, for data:

A screenshot of a cell phone

Description automatically generated

32 channels are used. Some channels have clear single unit. Some have multi-units.

Get\_spikes will call ‘get\_spikes\_single(filename, par\_input)’

data\_handler = readInData(par); will check if the input data are legit.

*Input file definition*

*From https://github.com/csn-le/wave\_clus/wiki/Input-Files*

*Wave\_clus can read MATLAB files (extension .mat) with continuous data or spikes for clustering spike shapes that have already been detected (e.g. detected on-line by the acquisition system). It should have either a vector named****data****(the continuous signal) or a matrix named****spikes****(nr. of spikes x length of the spike shape) plus a vector****index****with the spike times (in ms).*

Basically, readable mat file includes a data vector (raw data) and an index vector (in ms, denotes time)

data\_handler.par: contains some sorting/detection parameters

This function contains default parameters

function par = set\_parameters()

This can be modified through param.xxx =

The following script allows detection of spikes from Blackrock raw data:

% openNSx('read', 'report') % generate NS6 file.

live\_ch = [1:16 33:48]; % all live channels for 2 16-wire arrays

Fs = 30000;

% This is raw data of c13

% chdat = cell(1, length(live\_ch));

pardata =0;

if pardata ==1

filelist =[];

for i =1:length(live\_ch)

data=double(NS6.Data{1}(i, :)); % data is a necessary vector

index = [1:length(data)]\*1000/Fs; % index is a necessary entry

savefile = ['chdat' num2str(live\_ch(i)) '.mat']; % name of files

save(savefile, 'data', 'index');

filelist{i}=savefile;

end;

end;

% functional\_channels = [2 3 4 6 7 8 9 10 11 12 13 14 33 35 37 43 44 45 46];

functional\_channels = [1:16 33:48];

pos\_detection =[13]; % for channel 13, use positive detection

figure(20); clf;

set(gcf, 'unit', 'centimeters', 'position',[2 2 35 25], 'paperpositionmode', 'auto' )

figure(21); clf;

set(gcf, 'unit', 'centimeters', 'position',[2 2 35 25], 'paperpositionmode', 'auto' )

for i =1:length(functional\_channels);

indx = find(live\_ch==functional\_channels(i));

tosort\_list{i, 1} = ['chdat' num2str(functional\_channels(i)), '.mat'];

param.stdmin = 4;

param.sr = 30000;

if ~isempty(find(pos\_detection==functional\_channels(i)))

param.detection = 'pos';

else

param.detection = 'neg';

end;

param.detect\_fmin = 250; % high pass filter for detection

param.detect\_fmax = 5000; % low pass filter for detection (default 1000)

param.detect\_order = 4; % filter order for detection

param.sort\_fmin = 250; % high pass filter for sorting

param.sort\_fmax = 5000; % low pass filter for sorting (default 3000)

Get\_spikes(tosort\_list,'parallel',false,'par',param);

chspikes = load(['chdat' num2str(functional\_channels(i)) '\_spikes.mat']);

% struct with fields:

% par: [1√ó1 struct]

% threshold: 147.9486

% spikes: [584√ó64 double]

% index: [1√ó584 double]

% psegment: [1√ó100000 double]

% sr\_psegment: 1.6667e+03

load(['chdat' num2str(functional\_channels(i)) '.mat'])

[b\_detect,a\_detect] = ellip(param.detect\_order,0.1,40,[param.detect\_fmin param.detect\_fmax]\*2/param.sr);

data\_detect = filtfilt(b\_detect, a\_detect, data);

figure(20)

subplot(4, 8, indx)

set(gca, 'xlim', [1000 1000+5000], 'ylim', [-800 800], 'nextplot', 'add');

title(['ch#' num2str(functional\_channels(i))])

plot(index, data\_detect)

hold on

line([chspikes.index; chspikes.index], [-50 50], 'color', 'r', 'linewidth', .5)

if ~isempty(find(pos\_detection==functional\_channels(i)))

line(get(gca, 'xlim'), [chspikes.threshold chspikes.threshold], 'color', 'm', 'linewidth', 1)

else

line(get(gca, 'xlim'), -[chspikes.threshold chspikes.threshold], 'color', 'm', 'linewidth', 1)

end;

xlabel('Time (ms)')

ylabel ('Artificial unit')

axis off

figure(21)

subplot(4, 8, indx)

title(['ch#' num2str(functional\_channels(i))])

set(gca, 'xlim', [0 64]/30, 'nextplot', 'add', 'ylim', [-800 600]);

if size(chspikes.spikes, 1)>100

toplot = randperm(size(chspikes.spikes, 1), 100);

else

toplot = [1:100];

end;

waveindex = [1:64]/30;

plot(waveindex, chspikes.spikes(toplot, :), 'r')

xlabel('Time (ms)')

ylabel ('Artificial unit')

axis off

end;

print (20,'-dpng', ['spike\_dect\_all' ])

print (21,'-dpng', ['spike\_wave\_all'])

# Check the quality of spike sorting

Spike sorting is carried out using wave-clus

A screenshot of a cell phone

Description automatically generated

function CheckWaveClusSorting(chname)

% chname can ben chdat2

% raw data

raw = load(['chdat' chname]); % if chname is 2, then load('chdat2.mat')

% spike detection

spkall = load(['chdat' chname '\_spikes.mat']); % load spike times from detection, including all that pass thredhold

% par: [1√ó1 struct]

% threshold: [151.8461 150.4016 150.6069 150.6656]

% spikes: [11979√ó64 double]

% index: [1√ó11979 double]

% psegment: [1√ó100000 double]

% sr\_psegment: 1.6667e+03

spksort = load(['times\_chdat' chname '.mat'])

% spikes: [953√ó64 double]

% cluster\_class: [953√ó2 double]

% par: [1√ó1 struct]

% gui\_status: [1√ó1 struct]

% Temp: 13

% forced: [1√ó953 logical]

% inspk: [953√ó11 double]

% ipermut: [1√ó953 double]

% spksort.cluster\_class(:, 1) is the class definition, 0 unsorted, 1 sorted

figure(23); clf

set(gcf, 'unit', 'centimeters', 'position',[2 2 20 16], 'paperpositionmode', 'auto' )

ha=axes;

tdur=5;

tmax = floor(raw.index(end)/1000-tdur);

toplot=1;

ha2 = subplot(2, 2, [3]);

set(ha2, 'nextplot', 'add')

% plot sorted spikes

clusters = unique(spksort.cluster\_class(:, 1));

nclusters = length(clusters);

allcolors = varycolor(3);

allcolors = [

0 0 0

0 0 1

1 0 0

0 1 0

allcolors];

for k =1:nclusters-1

if size(spksort.spikes(spksort.cluster\_class(:, 1)== k, :), 1)>20

spkwave\_avg = mean(spksort.spikes(spksort.cluster\_class(:, 1)== k, :), 1);

plot(spkwave\_avg, 'color', allcolors(k+1, :), 'linewidth', 1);

end;

end;

axis tight

while toplot

ha1 = subplot(2, 2, [1 2]);

set(ha1, 'ylim', [-800 800], 'nextplot', 'add')

xlabel('Time (s)')

ha3 = subplot(2, 2, 4);

set(ha3, 'xlim', [0 25], 'nextplot', 'add')

xlabel('Inter-spike interval (ms)')

ylabel ('Counts')

tbeg = randperm(tmax, 1);

tend =[tbeg+tdur];

index\_raw=find(raw.index>=tbeg\*1000 & raw.index<=tend\*1000);

[b\_detect,a\_detect] = ellip(4,0.1,40,[250 10000]\*2/30000); % high pass

data\_plot\_hp = filtfilt(b\_detect, a\_detect, raw.data(index\_raw)); % band pass 2-200 hz

axes(ha1)

plot(raw.index(index\_raw)/1000, data\_plot\_hp, 'k');

% plot detection

index\_detection = find(spkall.index>=tbeg\*1000 & spkall.index<=tend\*1000);

spk\_peaks = min(spkall.spikes(index\_detection, :), [], 2);

plot(spkall.index(index\_detection)/1000, spk\_peaks, 'mo','markersize', 3)

% Either way, wave\_clus generates a file times\_filename.mat, with a variable cluster\_class of two columns: the first column is the class of the spike and the second one is the spike time in ms.

spk\_peaks\_max = prctile(min(spksort.spikes(find(spksort.cluster\_class(:, 1) ~= 0), :),[], 2), 1);

for k =1:nclusters-1

if size(spksort.spikes(spksort.cluster\_class(:, 1)== k, :), 1)>20

axes(ha1)

spk\_time\_sort = (spksort.cluster\_class(find(spksort.cluster\_class(:, 1) == k), 2)); % in ms

spk\_peaks\_sort = prctile(spksort.spikes(find(spksort.cluster\_class(:, 1) == k), :)', 5);

if ~isempty(find(spk\_time\_sort>=tbeg\*1000 & spk\_time\_sort<=tend\*1000))

plot([spk\_time\_sort(spk\_time\_sort>=tbeg\*1000 & spk\_time\_sort<=tend\*1000)/1000 spk\_time\_sort(spk\_time\_sort>=tbeg\*1000 & spk\_time\_sort<=tend\*1000)/1000]', [spk\_peaks\_max-100 spk\_peaks\_max],...

'linewidth', 2, 'color', allcolors(k+1, :));

end;

% ISI histogram

edges = [0:1:25];

[ncounts{k}] = histcounts(diff(spk\_time\_sort), edges);

axes(ha3)

plot(edges(2:end), ncounts{k}, 'color', allcolors(k+1, :), 'linewidth', 1)

end;

end;

set(ha1, 'ylim', [-800 800], 'nextplot', 'add', 'xlim', [tbeg tend])

reply = input('Keep showing Y/N [Y]:', 's');

if isempty(reply) || strcmp(reply, 'Y')|| strcmp(reply, 'y')

toplot =1;

else

toplot=0;

end;

end;

reply = input('Save this pic? Y/N [Y]:', 's');

if isempty(reply) || strcmp(reply, 'Y')|| strcmp(reply, 'y')

print (gcf,'-dpng', ['waveclus\_chdat' chname])

end;

A close up of a device

Description automatically generated