function FunImaging\_analysis\_Garrity()

% Original analysis script written by Timothy Wiggin (Leslie Griffith lab)

% Edited by Brian Cary for Garrity Lab

%%%%%%%%%%%%%%%%%%%%% ASSUMPTIONS %%%%%%%%%%%%%%%%%%%%%%.

% 16-bit TIFF file inputs %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Set analysis parameters

CODE\_REVISION\_DATE = '2017-2-28'; % Code version stamp for output

ALIGNMENT\_TOL = 1; %Can mess with if alignment is weird. Lower is

stricter.

%%%%%%%%

% Change based on experimentn

% Are used to calculate maximum change from baseline fit

BASELINE\_FRAMES = 20;

PRE\_BASELINE\_LENGTH = 28;

STIMULUS\_LENGTH = 40;

%%%%%%%%

% Set data read paths

[~,path]=uigetfile('\*.tif','Select folder of tif files');

file\_names = dir(fullfile(path, '\*.tif'));

% Set data save folder

data\_save\_dir = uigetdir(pwd,'Select folder to save data output');

% Prompt for annotation

expt\_title = inputdlg('Provide a brief title for the experiment:',

'Annotation Prompt');

% Prompt for alignment or not

alignyesno = inputdlg('Perform alignment? [y/n]');

%% Read in image data

fprintf(1,'Reading image data:(%03d%%)', 0); % Progress bar

first\_frame = imread(fullfile(path, file\_names(1).name));

frame\_series = zeros(size(first\_frame, 1), size(first\_frame, 2),

length(file\_names));

frame\_series(:,:,1) = first\_frame;

for i = 2:length(file\_names)

fprintf(1,'\b\b\b\b\b%03d%%)', round( (i / length(file\_names))

\* 100)); % Progress bar

frame\_series(:,:,i) = imread(fullfile(path,

file\_names(i).name));

end

fprintf(1,'\n');

%% Correct Drift in the Imagesn

if strcmp(alignyesno, 'y');

disp('Correcting for drift...');

aligned\_images = align\_stacks(frame\_series, ALIGNMENT\_TOL);

else

aligned\_images = frame\_series;

end

%% Select Signal and Background ROIs

disp('Calculating signal intensity and ratio...');

doodle\_image = mean(aligned\_images, 3);

figure;

imshow(doodle\_image);

imshow(doodle\_image/max(doodle\_image(:)));

title('Draw polygon surrounding: Analysis region Of interest');

signal\_mask = roipoly();

close;

figure;

imshow(doodle\_image/max(doodle\_image(:)));

title('Draw polygon surrounding: Background Region');

background\_mask = roipoly();

pause(0.1)

close all;

pause(0.2)

%% Calculate signal to bg Ratio

signal\_mask = double(signal\_mask);

signal\_mask\_size = sum(signal\_mask(:));

background\_mask = double(background\_mask);

background\_mask\_size = sum(background\_mask(:));

signal = zeros(size(aligned\_images,3),1);

background = zeros(size(aligned\_images,3),1);

masked = zeros(size(aligned\_images));

for i = 1:size(aligned\_images,3)

masked(:,:,i) = aligned\_images(:,:,i) .\* signal\_mask;

signal(i) = sum(sum(masked(:,:,i))) / signal\_mask\_size;

temp = aligned\_images(:,:,i) .\* background\_mask;

background(i) = sum(temp(:)) / background\_mask\_size;

end

corrected = signal - background;

scaled\_signal = corrected / mean(corrected(1:BASELINE\_FRAMES));

output = struct('Analysis\_Code\_Version', CODE\_REVISION\_DATE, ...

'Brief\_Experiment\_Title', expt\_title, ...

'Reference\_Image', doodle\_image, ...

'ROI\_Mask', signal\_mask, ...

'BG\_Mask', background\_mask, ...

'Channel\_ROI\_Intensity', signal, ...

'Channel\_BG\_Intensity', background, ...

'Scaled\_Signal', scaled\_signal);

% Display a figure of output

figure

subplot(3,1,1)

hold on

plot(output.Channel\_BG\_Intensity, 'color', 'cyan')

ylabel('Background Intensity')

subplot(3,1,2)

hold on

plot(output.Channel\_ROI\_Intensity, 'color', 'magenta')

ylabel('ROI Intensity')

subplot(3,1,3)

plot(output.Scaled\_Signal, 'color', 'green')

ylabel('Normalized Signal/Background')

%%%%

% Fits a line to baseline data and finds the maximum difference

between

% linear fit and signal

try

y = output.Scaled\_Signal;

y = y(~isnan(y));

x = (1:length(y)).';

P = polyfit(x(1:PRE\_BASELINE\_LENGTH),y(1:PRE\_BASELINE\_LENGTH),

1);

yfit = P(1)\*x +P(2);

drug\_time = PRE\_BASELINE\_LENGTH:(PRE\_BASELINE\_LENGTH +

STIMULUS\_LENGTH);

max\_change = max(abs(yfit(drug\_time)-y(drug\_time)));

fprintf('The max change in signal from baseline is %.3f \n',

max\_change)

catch

disp('baseline or stimulus length does not fit. No max change

calculated.');

end

%%%%

%% Write the data to disk

%fprintf(1,'Writing output to disk:(%03d%%)', 0); % Progress bar

out\_fdl = strcat('Analysis\_', expt\_title, '\_', datestr(now, 29) );

mkdir(data\_save\_dir,out\_fdl{1});

save(fullfile(data\_save\_dir, out\_fdl{1}, 'analysis\_output.mat'),

'output');

%imwrite(uint16(frame\_series(:,:,1)), fullfile(pwd, out\_fdl{1},

'original\_frames.tif'));

% Write a csv with the roi,bg, and signal data

fid = fopen(fullfile(data\_save\_dir, out\_fdl{1}, [out\_fdl{1},

'.csv']), 'w');

fprintf(fid, '%s,', 'ROI');

fprintf(fid, '%s,', 'BG');

fprintf(fid, '%s\n', 'Scaled Signal');

fclose(fid);

csvdata(:,1) = output.Channel\_ROI\_Intensity;

csvdata(:,2) = output.Channel\_BG\_Intensity;

csvdata(:,3) = output.Scaled\_Signal;

dlmwrite(fullfile(data\_save\_dir, out\_fdl{1}, [out\_fdl{1}, '.csv']),

csvdata, '-append');

% Save alignment video if alignment was performed

if strcmp(alignyesno, 'y')

imwrite(uint16(aligned\_images(:,:,1)), fullfile(data\_save\_dir,

out\_fdl{1}, 'aligned\_image.tif'));

%imwrite(uint16(masked(:,:,1)), fullfile(pwd, out\_fdl{1},

'masked.tif'));

for i = 2:size(aligned\_images, 3)

fprintf(1,'\b\b\b\b\b%03d%%)', round( (i /

size(aligned\_images, 3)) \* 100)); % Progress bar

%imwrite(uint16(frame\_series(:,:,i)), fullfile(pwd,

out\_fdl{1}, 'original\_frames.tif'), 'WriteMode' , 'append');

%imwrite(uint16(aligned\_images(:,:,i)), fullfile(pwd,

out\_fdl{1}, 'aligned\_image.tif'), 'WriteMode' , 'append');

end

end

fprintf('\n');

disp('Done!');

end

%%%%% alignment function. Recursively matches frames.

function [aligned\_images] = align\_stacks(images\_to\_align,

ALIGNMENT\_TOL)

[optimizer, metric] = imregconfig('multimodal');

Rfixed = imref2d(size(images\_to\_align));

half\_stack = round(size(images\_to\_align,3) / 2);

[aligned\_images] = align\_half(images\_to\_align(:,:,1:half\_stack),

images\_to\_align(:,:,half\_stack + 1:end));

function [a\_images] = align\_half(image\_A, image\_B)

% Check if the A stack is aligned to itself

tform = imregtform(image\_A(:,:,end) , image\_A(:,:,1) ,

'translation' , optimizer , metric);

if(pdist2([0 0], tform.T(3,1:2)) > ALIGNMENT\_TOL)

% If there is only one frame in the stack, it is

already aligned to itself and this will never execute

ha\_st = round(size(image\_A,3) / 2);

[image\_A] = align\_half(image\_A(:,:,1:ha\_st),

image\_A(:,:,ha\_st + 1: end));

end

% Check if the B stack is aligned to itself

tform = imregtform(image\_B(:,:,end) , image\_B(:,:,1) ,

'translation' , optimizer , metric);

if(pdist2([0 0], tform.T(3,1:2)) > ALIGNMENT\_TOL)

% If there is only one frame in the stack, it is

already aligned to itself and this will never execute

ha\_st = round(size(image\_B,3) / 2);

[image\_B] = align\_half(image\_B(:,:,1:ha\_st),

image\_B(:,:,ha\_st + 1: end));

end

% The A and B stacks are now internally aligned. Check if

the A

% and B stacks are aligned with one another

tform = imregtform(image\_B(:,:,1) , image\_A(:,:,1) ,

'translation' , optimizer , metric);

if(pdist2([0 0], tform.T(3,1:2)) > ALIGNMENT\_TOL)

% Not aligned, apply the transformation to the entire B

stack

for i = 1:size(image\_B,3)

image\_B(:,:,i) = imwarp(image\_B(:,:,i), tform,

'OutputView', Rfixed);

end

end

% The A and B stacks are now internally aligned and aligned

with

% one another. Concatonate and return them.

a\_images = cat(3, image\_A, image\_B);

end

end