Optical Imaging Data

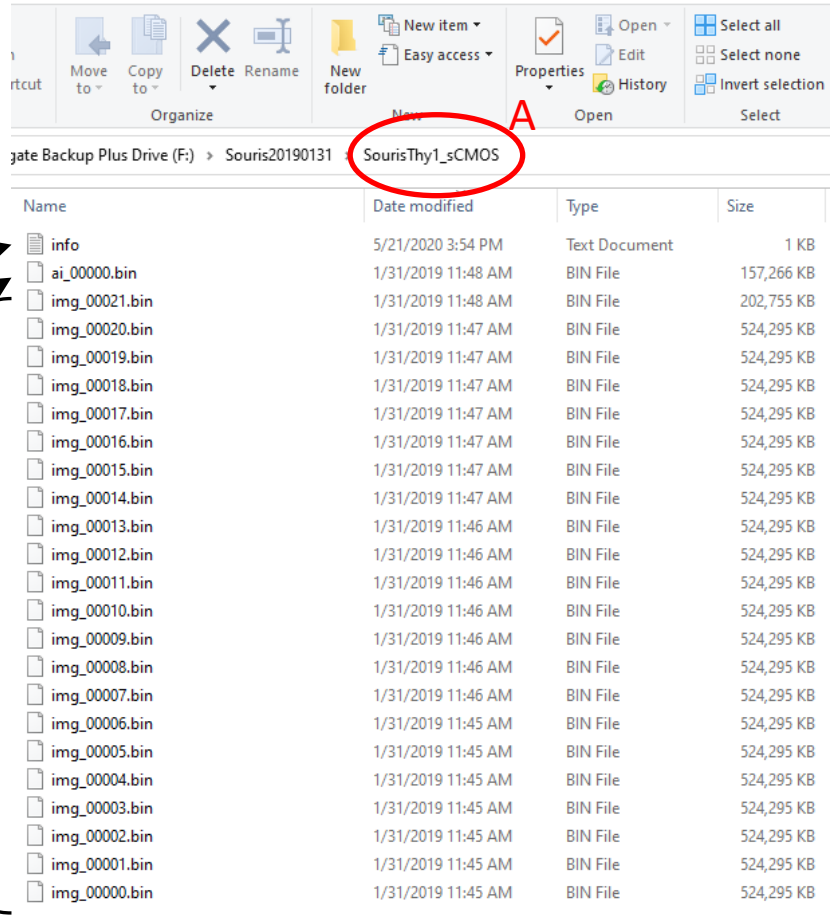
Labeo Technologies, September 2020

Recommended Hardware

- Any recent CPU should work without any issues.
- 32 GB of RAM.
- SSD Hard drive.
- The analysis software uses Matlab and toolboxes: wavelength & signal processing.

How raw data is organized:

- Each recordings have their own folder [A]
- In this folder, there will be 3 different type of files:
  - **Info.txt**: contains all the parameters of the system for that specific recording
  - **ai_#####.bin**: binary file containing analog inputs
  - **img_#####.bin**: binary file containing images



Update Notes:

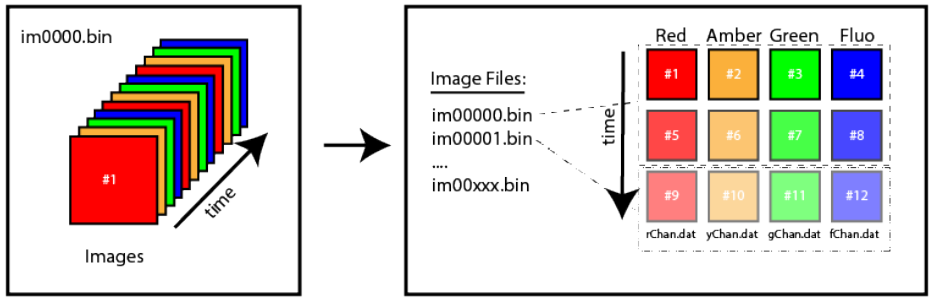
- The names of the files generated by ImagesClassification changed in a recent update. Be aware of this when using the following scripts, there might be errors due to this change.
  - Old file names: rChan.dat, gChan.dat, yChan.dat, fChan.dat and sChan.dat
  - New file names: red.dat, green.dat, yellow.dat, fluo_xx.dat and speckle.dat

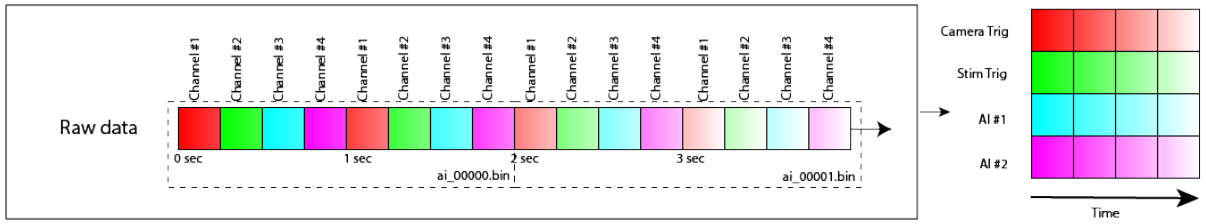
Opening data from Matlab

- An open source library with the basic opening/filtering functions is supplied by Labeotech.
- This library can:
  - Open and classify the different channels
  - Compute HbO and HbR from the non-fluorescent channels (*to be released soon*)
  - Do the hemodynamic correction on fluorescence data
  - Compute $\Delta F/F$ for fluorescence channels
  - Compute blood flow from speckle acquisitions

Data organization in .bin files:

- img files:
  - Channels are interlaced during recordings.
  - Each file contains X images (depending on the size of the region of interest, this number can change. Files will always be around 500 MB)
  - Once the first file is full, a new img_ file is created, until the end of the acquisition.
- ai files:
  - The data pattern on those file is a bit more complex.
  - Analog inputs are saved once every seconds (and at the end of the acquisition)
  - The transmission of the data from each AI channel is interlaced when received from the acquisition board.
  - For each transmission, the new chunk of data is stacked at the end of the previous one. Once the file size limit is reach, a new ai_#####.bin file is created.





Pre-Analysis (to separate channels):

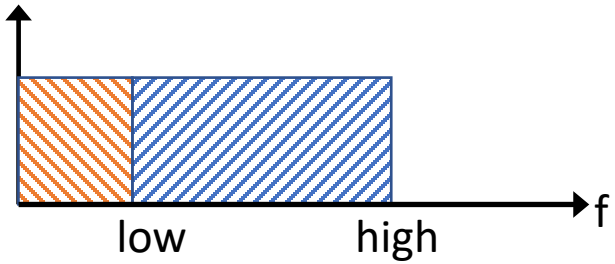
Fonction to put each recorded channel in its own file:

ImagesClassification(*sDataFolder*, sSaveFolder, *iSpatialBinning, iTemporalBinning, bIgnoreStim, bROI*);

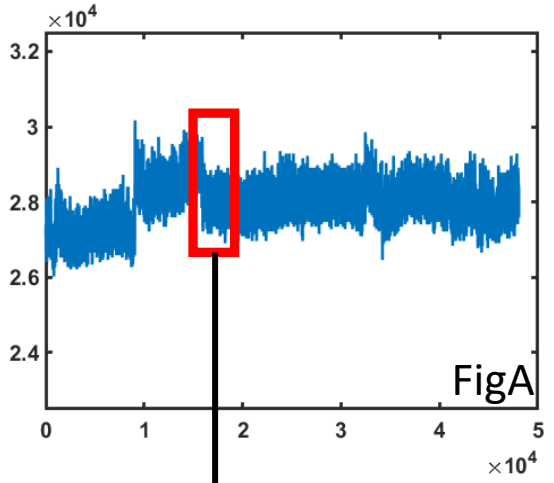
1. First parameter is the path to the folder containing data to be analysed;
2. Second is the path to where to save the results.
3. Third parameter is for the spatial binning option (set to 2 for a 2x2 binning; 1 for no binning or 1x1.);
4. Fourth parameter is for temporal binning (through averaging). Set to 1 for no binning, and to any integer number for binning (i.e.: 3 for a 3 frames averaging)
5. Fifth parameter is to tell the software to consider the acquisition as a Resting state protocol (meaning without stimulation). (set to 1 to ignore stimulation signal; set to 0 to do as required based on the stimulation signal).
6. Sixth parameter is to be able to set a post-acquisition Region of Interest. It can help to reduce the size of an acquisition. Set to 0 to ignore.

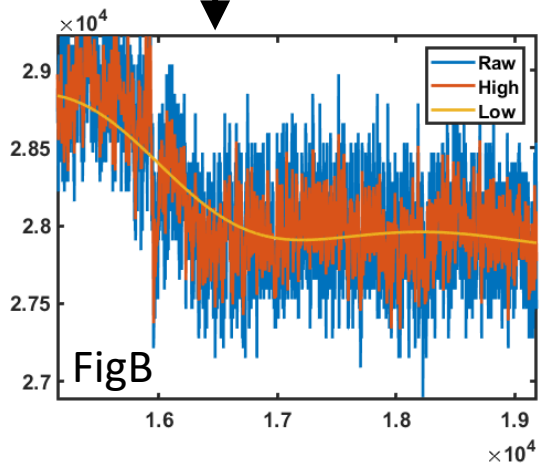
Normalization filtering

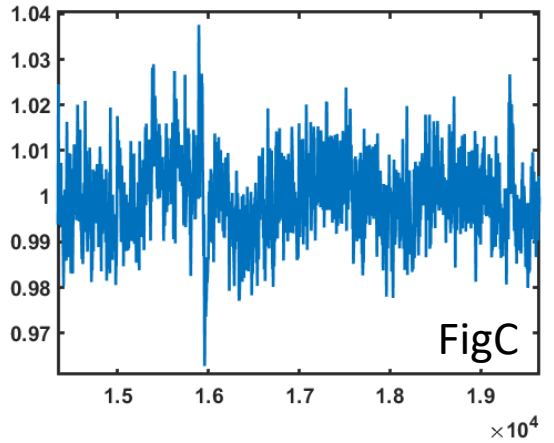
- Let suppose a temporal signal from one pixel (figA)
- The filter use two frequencies to normalize the signal:
  - A low frequency filter
  - A high frequency filter



- In the case of the pixel presented before, we would get these two signals (figB)
- The normalized output would then be has presented in (figC)







Normalization:

(Must be runned after ImagesClassification.m)

To do the intrinsic channel normalisation (see following slide for details)

NormalisationFiltering(*FolderData, FileData, lowFreq, highFreq, bDivide*)

1. Path of the folder containing the data file to be normalized.
2. Can be either: the name of the file to be normalized or the matrix containing the data.
3. Low frequency cut-off
4. High frequency cut-off
5. If set to 1: the returned data will be of the form: $Data_{HF}/Data_{LF}$
   Otherwise: $Data_{HF} - Data_{LF}$

Ex: dat = NormalisationFiltering(pwd, ‘red’, 1/120, 1, 1);

Ex2: dat = NormalisationFiltering(pwd, dat, 0, 3, 0);

Hemodynamic correction

(Must be runned after ImagesClassification.m)

<u>To do the hemodynamic correction on Fluorescence</u>

HemoCorrection(*Folder, varargin*):

1- Folder: Folder contaning the dataset to work with.

2- Varargin a)-> if empty: a dialog box will be prompt to ask user which channels to use to do the correction.

b) -> cell array of string: to specify which channels to use. (Red, Green and/or Yellow)

Ex: HemoCorrection(pwd, {'Red', 'Green'});

<u>To do the fluorescence normalisation:</u>

Dat = NormalisationFilter(*FolderData, FileData, lowFreq, highFreq, bDivide)*

- See slides 8 and 9 for explanations.
- Ex: dat = NormalisationFilter(pwd, ‘fluo_475’, 0.3, 3, 1);

How to run analysis (for Blood Flow):

(Must be runned after ImagesClassification.m)

1. Ana_Speckle(*sFolderName, []*)
    1. First parameter is the path to the folder containing data;
    2. Second parameter is not used at this time, must be set to [];

How to see results:

- Manually:
  - In Matlab (change the working directory to where the data is stored), then:

```
Info = matfile('fluo_475.mat');
fid = fopen('fluo_475.dat');
dat = fread(fid, inf, 'single');
fclose(fid);
dat = reshape(dat, Info.datSize(1,1), Info.datSize(1,2),[]);
Imagesc(dat(:,:,1)); %or any other frame #
```

  - This can be done for any .dat file:
    - sChan.dat or speckle.dat for speckle
    - rChan.dat or red.dat, gChan.dat or green.dat and yChan.dat or yellow.dat for any of the three colors channels
    - fChan.dat (or fluo_xx.dat) for fluorescence

Example:

- Step 1: Pre-analyse

```
ImagesClassification(pwd, pwd, 1, 1, 0, 0);
```

- Step 2: Hemodynamic correction

```
dat = HemoCorrection(pwd,{'red','green','yellow'});
```

- Step 3: Normalization

```
Dat = NormalisationFiltering(pwd, dat, 0.3, 3,1);
```

- Step 4: Visualisation

```
figure;
for ind = 1:size(dat,3)
        imagesc(dat(:,:,ind),[0.95 1.05]);
        title(int2str(ind));
        pause(0.1);
end
```