An object-oriented analysis pipeline

Yixiang Wang 07/01/2018 Version 0.0.1

0. Required libraries

   wholeBrainDX-master

   CEDMATLAB (spike2 library)

   If you want to do fast non-rigid registration, recommend:

   DEMON (a fast non-rigid registration library, not necessary)

1. Classes (modules)

***Description of a specific class/method(function)/property can be viewed in matlab. For example, you can simply type:**

'doc Integration' (documentation of class Integration)

'movieData.fetchAverageImage' (doc of method/function fetchAverageImage under class movieData)

to get more details.

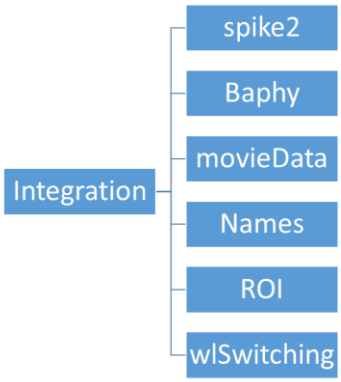
Superclasses:

spike2, Baphy, movieData, Names, ROI, wlSwitching

Subclass:

Integration

Configuration:



Integration class **inherits properties and methods** from spike2, Baphy, movieData, Names, ROI, wlSwitching.

If a method is static, which means you do not need to initiate an instance (object) for using it, you can just call this method by 'className.methodName(Input variables)'.

If a method is not static, you need an instance first. In this case, you need to first construct an instrance and then use 'instanceName.methodName(Input variables)' to call the method.

Most of methods are static in this framework

For example:

1.To generate lists of filenames of .tif/.mat/.m files under current folder, you can type 'Integration.fileDetector()'.

2.To do analysis on current movie, you need to make an Integration instance (object), then do analysis on this instance. Say you create an instance named 'IntgA', to run pre-processing pipeline on this instrance using the method(function) 'prePipe', type 'IntgA.prePipe()' .

When you create an Integration object, the program will first construct the corresponding spike2, baphy, movieData, Names, ROI, wlSwitching sub-objects (if specified). Module(s) can be waived if it's not required for the analysis or the corresponding files are not provided.

3.

An example of using this pipeline

First, you need a separate pipeline function (here 'audPipe(param)'), feed in the function with a struct 'param'. I copy my script I used on HPC here:

```
addpath(genpath('/gpfs/ysm/scratch60/yw545/Command'));
cd('/gpfs/ysm/scratch60/yw545/YW_070218');
%Asign the param struct
param.spacialFactor = 2; %For downsampling
param.flag = 1; %Flag for different processing procedures
param.rgd_flag = 1; %Flag for doing/not doing rigid registration
audPipe(param);
```

The function audPipe.m looks like this:

```
function audPipe(param)

    %If run on the HPC, use slurm to change the current directory
    try
        currentFolder = 'E:\Yixiang\New scripts\test';
        cd(currentFolder);
    catch
        disp('No such directory...Running on pwd...')
    end

    spacialFactor = param.spacialFactor; %For downsampling
    flag = param.flag; %Flag for different processing procedures
    rgd_flag = param.rgd_flag; %Flag for doing/ not doing rigid registration

    %Detect movie/baphy/spike2/roi files
    Integration.fileDetector();

    %Convert spike2 raw data to .mat files
    Integration.Spike2Matlab(cd);
    % Integration.Spike2Matlab(cd,'E:\CEDMATLAB\CEDS64ML');
```

```
    filelist = readtext('files.txt',' ');
    nmov = size(filelist,1);
    IdxInAll = cell(nmov,1);
    IdxInAll_1 = cell(nmov,1);
    IdxInAll_2 = cell(nmov,1);

    %Process each movie sequentially
    for f = 1:nmov
        [Idx,Idx1,Idx2] =
      Integration.processMovies(f,flag,spacialFactor,nmov,rgd_flag);
        IdxInAll{f,1} = Idx;
        IdxInAll_1{f,1} = Idx1;
        IdxInAll_2{f,1} = Idx2;
    end

    %Do averaging across all movies
    Integration.doAveragingAcrossMovies(flag,IdxInAll,IdxInAll_1,IdxInAll_2);

end
```

This function will call the `Integration.processMovies` method iteratively. This method can handle different scenarios (wavelength switching or single channel recording), but ultimately it will call the `prePipe` method/function under the Integration class. You should decide how exactly you'd like to process your data in the `prePipe` method/function. For example, you can apply different filters in sequence using methods from the Integration class:

```
%Correct photobleaching
A_corrct = Integration.bleachCorrection(obj.A);
disp('Photobleaching corrected!');

% Decide whether do rigid registration or not
if reg_flag
    if ~exist('Fixed_frame.mat','file')
        A_fixed = A_corrct(:,:,1);
        save('Fixed_frame.mat','A_fixed');
    else
        load('Fixed_frame.mat');
    end
    A_corrct = movieData.movieRigReg(A_fixed,A_corrct);
end

%Apply ROI mask(s)
A_ROI = Integration.ApplyMask(A_corrct,obj.ROIData);
clear A_corrct

%Downsampling
A_DS = Integration.downSampleMovie(A_ROI,spacialFactor);
clear A_ROI

%SVD denosing of down-sampled A
de_A = Integration.roiSVD(A_DS);
disp('SVD denosing is done')
```

```
disp('')
clear A_DS

%Top-hat filtering
TH_A = Integration.TopHatFiltering(de_A);
disp('Top-head filtering is done');
disp('')
clear de_A

%Check flag to decide whether to generate frequency/volume maps
if obj.flag
    %Integration.FreqColorMap(TH_A,filename,obj);
    Integration.FreqVoluColorMap(TH_A,filename,obj.nmov);
end

%Impose dFOverF to top-hat filtered matrix
TH_A = Integration.grossDFoverF(TH_A);
disp('Gloabal dFoverF is done')
disp(' ')

%Gaussian smoothing
Ga_TH_A = Integration.GauSmoo(TH_A,2); %set sigma = 2
disp('Gaussian smoothing is done');
disp(' ')
%clear TH_A

%Save filtered matrix
outputFolder = fullfile(currentFolder,obj.outputFolder);
mkdir(outputFolder);
save(fullfile(outputFolder,checkname),'Ga_TH_A','-v7.3');

%Binary thresholding of pre-processed A
[BW_ppA,~] = Integration.bwThresholding(Ga_TH_A);
clear GA_TH_A;

%Generate connected component
Integration.GenerateCC(TH_A,BW_ppA,filename)
clear TH_A BW_ppA

disp(['Preprocessing done: ' filename]);
disp('')
```

For the auditory project, one just need to put all the raw data in a folder:
.tif movie files .roi/.zip ROI files .m baphy input records
.smr (spike2 raw data) later will be automatically converted to .mat files

The pipeline will detect all these files and run through each movies and combine information from its corresponding baphy and spike2 files, do all kinds of alignment/sorting and apply different filters, and ultimately output filtered matrix, connected components/regionprop info, and tonotopic maps. For other projects, you might want write you own pipeline method/function.