Neuron Image Analysis: From raw pics to .csv file

Jesse Cohn[1]

[1]Washington University, Saint Louis

Apr 30, 2021

1 Works for me This protocol is published without a DOI.

Jesse Cohn

ABSTRACT

This protocol describes the pipeline I've been using to analyze images from the InCell of the Kampmann i3N-CRISPRi cell lines. The broad overall steps are:

1. Clean up the folder of images from the InCell (makes a folder per well, with subfolders for Red, Green, Brightfield)
2. Use Fiji/ImageJ to stitch together the images from each well
3. Create a CellProfiler batch file for the stitched images
4. Run the CellProfiler analysis on the cluster
5. Collapse all of the CellProfiler output csv's to a single file for analysis

Some of these steps could likely still be streamlined, I just haven't done it yet.

To follow this pipeline you'll need to be using the same settings on the InCell that I used for imaging (the JCohn_20x_Neurons_2 protocol on the InCell).

If you change any of the settings for these things, there are places in the pipeline that will need tweaking accordingly.

All of the scripts and pipelines needed at various steps are attached to that step or available at: https://github.com/jc6213/CRANIUM . There are also example analysis Jupyter Notebooks there for ease.

PROTOCOL CITATION

Jesse Cohn 2021. Neuron Image Analysis: From raw pics to .csv file. **protocols.io** https://protocols.io/view/neuron-image-analysis-from-raw-pics-to-csv-file-btafnibn

CREATED

Mar 12, 2021

LAST MODIFIED

Apr 30, 2021

PROTOCOL INTEGER ID

48167

Installing a local version of CellProfiler

1 This step will only need to be done the first time you do this. The version of CellProfiler on the cluster is old, so Eric Martin suggested installing a local version to use. To do this, start up an interactive node on the cluster with a good amount of memory (I can't remember how much I used but I remember it failed the first few times so I ended up requesting quite a bit) and run this code:

```
module load python/3.8.8
module load py-pip/19.0.3-python-3.8.8
```

```
module load py-virtualenv/16.4.1-python-3.8.8
module load py-wheel/0.33.1-python-3.8.8
module load jdk

python3 -m venv env
. env/bin/activate

python3 -m pip install numpy
python3 -m pip install cellprofiler
```

If it fails, request more memory in your interactive node and try again. Get in contact with Eric if you're still having trouble with it.

Cleaning up the folder

2 Image your plates with the "**JCohn_20x_Neurons_2.xaqp**" acquisition protocol on the InCell. This should produce a Brightfield, Red, and Green image channel for each image field, with 12 image fields per well (3 rows x 4 columns, captured in a Horizontal Serpentine fashion) of a 96-well plate. Usually you'll have 1 folder per day imaged, and will need to run this script for each of those folders. An example of the InCell output file names for a single field of view would be:

```
A - 01(fld 01 wv 488 - GreenHS).tif
A - 01(fld 01 wv 561 - Orange).tif
A - 01(fld 01 wv TL-Brightfield - Orange).tif
```

Where, for example, "A - 01" is the well, "fld 01" is which field (out of 12), "wv 488" is the wavelength of the fluorescence, and "- GreenHS" is the color name.

It is very important that your images are in this exact format, with these exact colors used, as all of the downstream pipelines rely on parsing these file names.

The folder hierarchy at the moment should be (for simplicity this example uses a well with only 3 fields, each of the real ones would have 12):

Example hierarchy before the WellFolders.py script is run

3 Get the WellFolders.py script: **WellFolders.py**

cd to the directory it is in, and then for each folder run at the command line:

```
python3 WellFolders.py [filepath] [day]
```

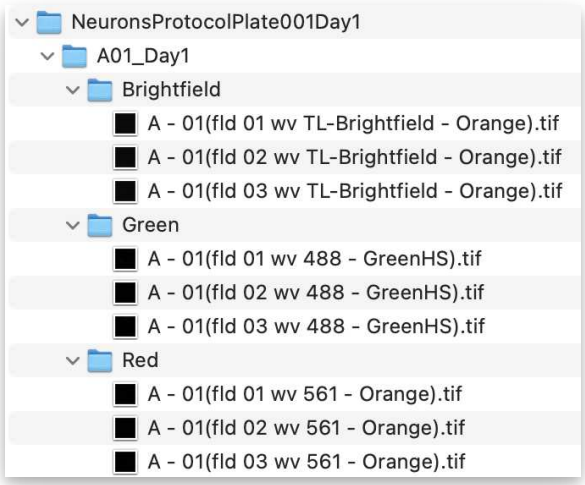
For example, if you had your pictures in:

- /Volumes/DriveName/NeuronsProtocolPlate001Day1
- /Volumes/DriveName/NeuronsProtocolPlate001Day3
- /Volumes/DriveName/NeuronsProtocolPlate001Day5
- /Volumes/DriveName/NeuronsProtocolPlate001Day7

Then you would run the commands:

```
python3 WellFolders.py /Volumes/DriveName/NeuronsProtocolPlate001Day1 1
python3 WellFolders.py /Volumes/DriveName/NeuronsProtocolPlate001Day3 3
python3 WellFolders.py /Volumes/DriveName/NeuronsProtocolPlate001Day5 5
python3 WellFolders.py /Volumes/DriveName/NeuronsProtocolPlate001Day7 7
```

In each folder, the structure should now be:



Example folder hierarchy after WellFolders.py run

This cleanup step isn't strictly necessary for anything, it just helps to keep things a bit more organized, and I wrote the next script (StitchImages.py) to expect this file structure.

Stitching Images

4 With the folders organized, now get the following files:

- **StitchImagesOnGreen.py**
- **FirstStitchGreen.ijm**
- **SecondStitch.ijm**

5 The StitchImagesOnGreen.py script will run ImageJ/Fiji in 'headless' mode, so you have two options:
1) Hard code the location of ImageJ/Fiji executable file into the script itself, on line 56.
2) Pass the location of the executable file with the --imagej flag when you run the script.

The path needs to point to the location of the actual ImageJ/Fiji executable file, not just the package. How I found mine was to go to my applications folder where Fiji is located, then right-click on Fiji, and click "Show Package Contents". The in the 'Contents' folder, and the 'MacOS' subfolder, the ImageJ executable was called "ImageJ-macosx". The path to that file is what you should use.

6 The StitchImagesOnGreen.py script will also call the FirstStitchGreen.ijm and SecondStitch.ijm ImageJ macros, so like above you have two options:
1) Hard code their locations into the script itself on lines 48 and 51.
2) Pass their location with the --firstlocation and --secondlocation flags when you run the script.

7 You'll need to run StitchImagesOnGreen on each of the folders you ran WellFolders on above. A call for StitchImagesOnGreen is in the form:

```
python3 StitchImagesOnGreen.py [path/to/folder] [day] --firstmacrolocation
[path/to/FirstStitchGreen.ijm] --secondmacrolocation [path/to/SecondStitch.ijm] --imagej
[path/to/imagej] --wells [A01 A02 etc.]
```

The --wells flag at the end is an optional flag if you only want to stitch images for some of the wells instead of all of them in the folder. I've never really used it but I included it because hey, why not.

So assuming we've hard-coded in the locations of the ImageJ macros and the ImageJ location, we would run it on our example folders from the last step like so:

```
python3 StitchImagesOnGreen.py /Volumes/DriveName/NeuronsProtocolPlate001Day1 1
python3 StitchImagesOnGreen.py /Volumes/DriveName/NeuronsProtocolPlate001Day3 3
python3 StitchImagesOnGreen.py /Volumes/DriveName/NeuronsProtocolPlate001Day5 5
python3 StitchImagesOnGreen.py /Volumes/DriveName/NeuronsProtocolPlate001Day7 7
```

8 The stitching steps take a while, since the ImageJ/Fiji stitching algorithm takes a bit. It usually takes between 45min - 1hr per folder of pictures.

At the end, you should have a folder called "**Stitched_Images_[day]**" in each of the folders you passed above, and the folder should contain stitched images with a file format of "**[well]_[color]_Stitched.tif**", with a Red and a Green image for each well. **Properly stitched images should have a file size of about 87-89 MB.**

8.1 Occasionally this may not make properly stitched images for some wells. You can usually tell quickly because the filesize will be significantly different from 87-89 MB. In that case you can try the "StitchImagesOnBF" script, which uses the Brightfield channel instead of the Green channel to do the stitching:

- **StitchImagesOnBF.py**
- **FirstStitchBF.ijm**

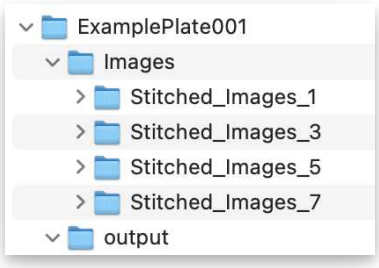
and the same 2nd stitch macro from above. Do the same process as above but pointing the firstmacro location to FirstStitchBF.ijm

Making the CellProfiler Batch file

9 At this point, we're ready to analyze these images with CellProfiler. What we'll be doing is making a "Batch" file with CellProfiler, then uploading everything to the cluster, and running the actual CellProfiler analysis on the cluster.

CellProfiler has documentation about how to make these "Batch" files here: http://cellprofiler-manual.s3.amazonaws.com/CellProfiler-3.0.0/help/other_batch.html. It's missing a few pieces of information, and is also a bit outdated, so I'll go over what needs to be done in this section too.

10 Here you'll move the Stitched_Image folders we made in the last section into a new folder. Make a folder with the following structure:



File hierarchy for making the CellProfiler batch file

The Stitched Images folders are the ones we made in the step above.

11 Get the pipeline file for CellProfiler to use: **JC_CRANIUM_Pipeline_v2.cppipe**

Now open up CellProfiler, go to File -> Import -> Pipeline From File... and load the **JC_CRANIUM_Pipeline.cpipe** pipeline.

12 Drag the 'Images' folder (that contains all of the Stitched_Images subfolders) into the Images module at the very top where it says "Drop files and folders here"

13 The "Metadata", "NamesAndTypes", and "Groups" modules should recognize and group everything appropriately based on the folder and filenames, so there shouldn't be anything you need to do in those modules.

Go to the "NamesAndTypes" module, click "Update" and **make a note of how many image sets there are (at the very bottom of the CellProfiler window it should say "Found __ imaging sets**"), we'll need this number later when we run this on the cluster.

14 Down in the bottom left of the CellProfiler window, click on the **"Output Settings"** button.

For **"Default Input Folder:"** put the absolute path to the "Images" folder.
Example:

```
/Volumes/SeagateExpansionDrive/ExamplePlate001/Images
```

For **"Default Output Folder:"** put the absolute path to the "output" folder.
Example:

```
/Volumes/SeagateExpansionDrive/ExamplePlate001/output
```

15 Next click on the "**CreateBatchFiles**" module.

For **Local Root Path:** put all of the path preceding the main parent folder name. So based on the last step, we'd put:

```
/Volumes/SeagateExpansionDrive/
```

and for **Cluster Root Path:** put the directory on the cluster you intend to move these files to. For example, I put everything into my main folder on scratch, which is:

```
/scratch/rmlab/1/Jesse/
```

The idea here is that the Batch module will take all of the file paths for your images, and replace the "Local Root Path" with the "Cluster Root Path" so that the file paths will correctly point to the right places on the cluster.

16 In the bottom left, click on the "**Analyze Images**" button.

This will take a sec, and then it should create a file called "**Batch_Data.h5**" in the output folder.

Getting things ready on the cluster

17 Now we'll move the whole folder to the cluster. Be sure you're moving it to the exact directory you specified above as the "Cluster Root Path".

Do this with rsync with the -r flag to move all the subfolders also. To continue our example:

```
rsync -r /Volumes/SeagateExpansionDrive/ExamplePlate001
jessecohn@htcf.wustl.edu:/scratch/rmlab/1/Jesse
```

18 The other two things we'll need are a 'lookup' file to tell each instance of CellProfiler which image sets to analyze, and the sbatch job submission. These are just text files so you can either write them on your computer and move them onto the cluster with rsync or make them in place with vim or nano.

The way I like to do this is to have each instance of CellProfiler run on a node with 25GB of memory, and to analyze at most 10 images each. I've never had this run out of memory before. So each line of the lookup file will be telling each instance of cell profiler which Image sets to analyze in the space-delimited format "**First Last**". For instance if you had 45 image sets, your .txt lookup file would just be:

```
1 10
11 20
21 30
31 40
41 45
```

For this example I'd save this as ExamplePlate001_lookup.txt

18.1 If you have a lot of image sets, you can use python to quickly make the lookup file. First enter the python interactive mode:

```
python3
```

18.2 Change the following code to make the file in the right directory and with a useful name, and change the "total" variable to reflect the total number of image sets you have:

```
with open('/scratch/rmlab/1/Jesse/ExamplePlate001_lookup.txt', 'w') as file:
    total = 45
    starts = list(range(1, (total + 1), 10))
    stops = list(range(10, total, 10)) + [total]
    for start, stop in zip(starts, stops):
        _ = file.write('{} {}\n'.format(start, stop))
```

18.3 Exit the python interactive mode:

```
exit()
```

You should now have the lookup file in the place you specified in the above code.

Submitting the job and getting the results

19 Next, for the job submission file, make a .sh file with this as the body:

```
#!/bin/bash

#SBATCH --cpus-per-task=1
#SBATCH --mem=25000
#SBATCH --array=1-20%8

module load python/3.8.8
module load py-virtualenv/16.4.1-python-3.8.8
module load jdk

read startimage stopimage < <(sed -n ${SLURM_ARRAY_TASK_ID}p ExamplePlate001_lookup.txt )

. /home/jessecohn/env/bin/activate
cellprofiler -p /scratch/rmlab/1/Jesse/ExamplePlate001/output/Batch_data.h5 -c -r -f
${startimage} -l ${stopimage}
```

Also can download here: **CellProfilerCall.sh**

Several parts will need editing:

```
#SBATCH --array=1-20%8
```

Change this line to have however many lines are in your lookup file (which you can get with wc -l on your lookup file). Like if you had **30** lines in your lookup file, it would become **#SBATCH --array=1-30%8** (the %8 just specifies to only run 8 instances of CellProfiler at a time).

```
read startimage stopimage < <(sed -n ${SLURM_ARRAY_TASK_ID}p ExamplePlate001_lookup.txt )
```

Change "ExamplePlate001_lookup.txt" to point to your lookup file.

```
. /home/jessecohn/env/bin/activate
cellprofiler -p /scratch/rmlab/1/Jesse/ExamplePlate001/output/Batch_data.h5 -c -r -f
${startimage} -l ${stopimage}
```

The first line of this section will need to be edited to point to the env where you have CellProfiler. If you installed it using the code from the very first step of the protocol, this will probably just be a matter of replacing 'jessecohn' with your cluster user name.

Then in the second line, replace '/scratch/rmlab/1/Jesse/ExamplePlate001/output/Batch_data.h5' with the path to your Batch_data.h5 file.

20 Submit the sbatch file.

```
sbatch CellProfilerCall.sh
```

21 This will take a while to run, usually a few hours. The output folder should start to fill up with .csv files in the format of "**[well]_[day]_output.csv**", one for each well on each day. Once all instances of CellProfiler are done running and all of the .csv files have accumulated, you can concatenate them all into one final file by running this command in the output directory:

```
awk 'FNR==1 && NR!=1{next;}{print}' *.csv > ExamplePlate001_Final.csv
```

That's it! The data should be ready to analyze.