# BASICS:

## 1 Installation
1.1 Dependencies ................................................. 4

## 2 Inputs
2.1 Input format ..................................................... 5
2.1.1 Directory structure ......................................... 5
2.1.2 Frame ordering .............................................. 5
2.1.3 Recordings with photostim / other artifacts ............... 6
2.2 Different file types ............................................. 6
2.2.1 Tiffs ......................................................... 6
2.2.2 Bruker ....................................................... 6
2.2.3 Mesoscope tiffs ............................................. 6
2.2.4 Thorlabs raw files ......................................... 7
2.2.5 HDF5 files (and *.sbx) .................................. 7
2.2.6 sbx binary files ........................................... 7

## 3 Settings (ops.npy)
3.1 Main settings ................................................... 9
3.2 Output settings .................................................. 10
3.3 Registration ..................................................... 10
3.4 ROI detection ................................................... 11
3.5 Signal extraction ............................................... 12
3.6 Spike deconvolution ............................................ 12
3.7 Channel 2 settings ............................................... 12

## 4 Using the GUI
4.1 Different views and colors for ROI panels ...................... 13
4.1.1 Views ....................................................... 13
4.1.2 Colors ..................................................... 14
4.1.3 Correlations ................................................ 14
4.1.4 Correlations with 1D var .................................. 15
4.1.5 Rastermap / custom ........................................ 15
4.2 Buttons / shortcuts for cell selection ............................ 15
4.2.1 Mouse control ............................................... 15
4.2.2 Keyboard shortcuts ........................................ 15
4.2.3 Multi-cell selection ........................................ 16
4.3 Trace view (bottom row) ....................................... 16
4.4 Classifying cells ............................................... 17
4.4.1 Adding data to a classifier .................................. 17
4.4.2 Building your own classifier ................................. 17
### Visualizing activity
- 4.5

### Manual adding of ROIs
- 4.6

### Merging ROIs
- 4.7

### View registered binary
- 4.8
  - 4.8.1 Z-stack Alignment
- 22

### View registration metrics
- 4.9
- 22

### Outputs
- 5
  - 5.1 MATLAB output
  - 5.2 NWB Output
  - 5.3 Multichannel recordings
  - 5.4 stat.npy fields
  - 5.5 ops.npy fields
- 25

### Multiday recordings
- 6
- 29

### Developer Documentation
- 7
  - 7.1 Versioning
  - 7.2 Testing
    - 7.2.1 Downloading Test Data
    - 7.2.2 Running the tests
- 31

### Frequently Asked Questions
- 8
  - 8.1 Cropped field-of-view
  - 8.2 Deconvolution means what?
  - 8.3 Multiple functional channels
  - 8.4 Z-drift
  - 8.5 No signals in manually selected ROIs
- 33

### Registration
- 9
  - 9.1 Finding a target reference image
  - 9.2 Registering the frames to the reference image
  - 9.3 1. Rigid registration
  - 9.4 2. Non-rigid registration (optional)
  - 9.5 Metrics for registration quality
    - 9.5.1 CLI Script
- 37

### Cell Detection
- 10
  - 10.1 Summary
  - 10.2 SVDs (= PCs) of data
  - 10.3 Sourcery
- 45

### Signal extraction
- 11

### Spike deconvolution
- 12

### suite2p.io package
- 13
  - 13.1 Submodules
  - 13.2 suite2p.io.binary module
  - 13.3 suite2p.io.h5 module
  - 13.4 suite2p.io.nwb module
  - 13.5 suite2p.io.save module
  - 13.6 suite2p.io.sbx module
  - 13.7 suite2p.io.server module
  - 13.8 suite2p.io.tiff module
- 51
13.9 suite2p.io.utils module .................................................. 56
13.10 Module contents .......................................................... 57

14 suite2p.registration package .................................................. 59
14.1 Submodules ........................................................................ 59
14.2 suite2p.registration.bidiphase module ...................................... 59
14.3 suite2p.registration.metrics module ........................................ 59
14.4 suite2p.registration.nonrigid module ...................................... 61
14.5 suite2p.registration.register module ...................................... 64
14.6 suite2p.registration.rigid module .......................................... 66
14.7 suite2p.registration.utils module ........................................ 67
14.8 suite2p.registration.zalign module ........................................ 69
14.9 Module contents .................................................................. 70

15 suite2p.detection package ......................................................... 71
15.1 Submodules ........................................................................ 71
15.2 suite2p.detection.anatomical module ...................................... 71
15.3 suite2p.detection.chan2detect module ..................................... 71
15.4 suite2p.detection.denoise module ......................................... 71
15.5 suite2p.detection.detect module .......................................... 71
15.6 suite2p.detection.metrics module ......................................... 72
15.7 suite2p.detection.sourcery module ........................................ 72
15.8 suite2p.detection.sparsedetect module ................................... 73
15.9 suite2p.detection.stats module ............................................ 75
15.10 suite2p.detection.utils module .......................................... 78
15.11 Module contents .................................................................. 79

16 suite2p.extraction package ....................................................... 81
16.1 Submodules ........................................................................ 81
16.2 suite2p.extraction.dcnn module ............................................ 81
16.3 suite2p.extraction.extract module ....................................... 82
16.4 suite2p.extraction.masks module ...................................... 83
16.5 Module contents .................................................................. 84

17 suite2p.classification package .................................................. 85
17.1 Submodules ........................................................................ 85
17.2 suite2p.classification.classifier module .................................. 85
17.3 suite2p.classification.classify module .................................... 86
17.4 Module contents .................................................................. 86

18 suite2p.gui package ................................................................. 87
18.1 Submodules ........................................................................ 87
18.2 suite2p.gui.buttons module .................................................. 87
18.3 suite2p.gui.classgui module ................................................. 88
18.4 suite2p.gui.drawroi module .................................................. 88
18.5 suite2p.gui.graphics module .................................................. 89
18.6 suite2p.gui.gui2p module ...................................................... 90
18.7 suite2p.gui.io module .......................................................... 90
18.8 suite2p.gui.masks module ..................................................... 91
18.9 suite2p.gui.menus module .................................................... 92
18.10 suite2p.gui.merge module ..................................................... 93
18.11 suite2p.gui.reggui module ................................................... 93
18.12 suite2p.gui.rungui module ................................................... 94
18.13 suite2p.gui.traces module ................................................... 96
18.14 suite2p.gui.utils module .................................................... 96
suite2p is an imaging processing pipeline written in Python 3 which includes the following modules:

- Registration
- Cell detection
- Spike detection
- Visualization GUI

For examples of how the output looks and how the GUI works, check out this twitter thread.

This code was written by Carsen Stringer and Marius Pachitariu. For support, please open an issue.

The reference paper is here. The deconvolution algorithm is based on this paper, with settings based on this paper.

We make pip installable releases of suite2p, here is the pypi. You can install it as pip install suite2p

- modindex
- search
- genindex
CHAPTER
ONE

INSTALLATION

Install an Anaconda distribution of Python – Choose Python 3.x and your operating system. Note you might need to use an anaconda prompt if you did not add anaconda to the path.

1. Download the suite2p repository from GitHub using Git: `git clone https://github.com/MouseLand/suite2p`
2. Open an anaconda prompt / command prompt with conda for python 3 in the path
3. Change the current directory to the suite2p folder: `cd suite2p`
4. Run `conda env create -f environment.yml`
5. To activate this new environment, run `conda activate suite2p`. Afterwards, You should see (suite2p) on the left side of the terminal line.
6. Install suite2p into this environment: `pip install suite2p`
7. Now run suite2p and you're all set.

If you have an older suite2p environment you can remove it with `conda env remove --name suite2p` before creating a new one.

Note you will always have to run `conda activate suite2p` before you run suite2p. Conda ensures mkl_fft and numba run correctly and quickly on your machine. If you want to run jupyter notebooks in this environment, then also `conda install jupyter`.

To upgrade suite2p (package here), run the following in the environment:

```
pip install suite2p --upgrade
```

Common issues

If when running suite2p, you receive the error: `No module named PyQt5.sip`, then try un installing and reinstalling pyqt5

```
pip uninstall pyqt5 pyqt5-tools
pip install suite2p
```

If when running suite2p, you receive an error associated with matplotlib, try upgrading it:

```
pip install matplotlib --upgrade
```

If you are on Yosemite Mac OS, PyQt doesn’t work, and you won’t be able to install suite2p. More recent versions of Mac OS are fine.

The software has been heavily tested on Windows 10 and Ubuntu 18.04, and less well tested on Mac OS. Please post an issue if you have installation problems. The registration step runs faster on Ubuntu than Windows, so if you have a choice we recommend using the Ubuntu OS.
1.1 Dependencies

- rastermap
- pyqtgraph
- PyQt5
- numpy (>= 1.13.0)
- scipy
- h5py
- scikit-learn
- scanimage-tiff-reader
- tifffile
- natsort
- matplotlib (not for plotting (only using hsv_to_rgb and colormap function), should not conflict with PyQt5)
2.1 Input format

This applies to all file types!

2.1.1 Directory structure

suite2p looks for all tiffs/h5 in the folders listed in `ops['data_path']`. If you want suite2p to look in those folders AND all their children folders, set `ops['look_one_level_down']`=True. If you want suite2p to only look at some of the folder’s children, then set `ops['subfolders']` to those folder names.

If you want suite2p to only use specific tiffs in ONE folder, then set the data path to only have one folder (`ops['data_path']=['my_folder_path']`), and name the tiffs you want processed in `ops['tiff_list']`.

See examples in this notebook.

2.1.2 Frame ordering

If you have data with multiple planes and/or multiple channels, suite2p expects the frames to be interleaved, e.g.

- `frame0 = time0_plane0_channel1`
- `frame1 = time0_plane0_channel2`
- `frame2 = time0_plane1_channel1`
- `frame3 = time0_plane1_channel2`
- `frame4 = time1_plane0_channel1`
- `frame5 = time1_plane0_channel2`
- ...

channels are ones-based (channel 1 and 2 NOT 0 and 1).
2.1.3 Recordings with photostim / other artifacts

Photostim and other artifacts require you to exclude these frames during ROI detection. Otherwise there will be “ROIs” that are related to the stimulation, not actually cells. To exclude them, make an array of integers corresponding to the frame times of the photostimulation. Save this array into a numpy array called `bad_frames.npy`:

```python
import numpy as np

bad_frames = np.array([20, 30, 40])
np.save('bad_frames.npy', bad_frames)
```

Put this file into the first folder in your `ops['data_path']` (the first folder you choose in the GUI).

2.2 Different file types

2.2.1 Tiffs

Most tiffs should work out of the box. `suite2p` relies on two external tiff readers: `scanimage-tiff-reader` and `sklearn.external.tifffile`. The default is the scanimage one, but it will use the other one if it errors.

You can use single-page tiffs. These will work out of the box if they end in *.tif or *.tiff. If they have a different ending then use the flag `ops['all_files_are_tiffs'] = True` and the pipeline will assume any files in your folders are tiffs. NOTE that these will be slower to load in and create the binary, so if you're planning on using the pipeline extensively you may want to change your acquisition output.

If you save a stack of tiffs using ImageJ, and it's larger than 4GB, then it won't run through `suite2p` anymore. A workaround is to save as an OME-TIFF in FIJI: "File->save as->OME-TIFF->compression type uncompressed" in FIJI (thanks @kylemxn! see issue here).

If you have old Scanimage tiffs (version <5) that are larger than 2GB, then most tiff readers will not work. @elhananby has recommended this repository for reading the data into matlab (see issue here). After reading it into matlab, you can re-save the tiff in a format that imageJ and `suite2p` can recognize (see matlab tiff writing here).

2.2.2 Bruker

Using Bruker Prairie View system,.RAW files are batch converted to single .ome.tifs. Now, you can load the resulting multiple tif files (i.e. one per frame per channel) to `suite2p` to be converted to binary. This looks for files containing ‘Ch1’, and will assume all additional files are ‘Ch2’. Select “input_format” as “bruker” in the drop down menu in the GUI or set `ops['input_format'] = 'bruker'`.

2.2.3 Mesoscope tiffs

We have a matlab script here for extracting the parameters from scanimage tiffs collected from the Thorlabs mesoscope. The script creates an `ops.json` file that you can then load into the run GUI using the button “load ops file”. This should populate the run GUI with the appropriate parameters. Behind the scenes there are `ops['lines']` loaded and `ops['dy'], ops['dx']` that specify which lines in the tiff correspond to each ROI and where in space each ROI is respectively. `ops['nplanes']` will only be greater than 1 if you collected in multi-plane mode. Once the pipeline starts running, this parameter will change to “nplanes * nrois” and each “plane” is now an ROI from a specific plane. Please open issues if you’re using this and having trouble because it’s not straightforward.
2.2.4 Thorlabs raw files

Christoph Schmidt-Hieber (@neurodroid) has written haussmeister which can load and convert ThorLabs *.raw files to suite2p binary files! suite2p will automatically use this if you have pip installed it (pip install haussmeister).

2.2.5 HDF5 files (and *.sbx)

These should work out of the box, but are less well-tested. Dario Ringach has a utility to convert neurolabware *.sbx files to *.h5 files (see blog post here).

The H5 loading from the GUI now works the same as it always has for tiffs. Select “h5” from the drop-down menu and input the h5 KEY for the data as a string. Now choose the folder with your *.h5 or *.hdf5 files and the pipeline will use all h5 files in that folder. You can use ops[‘look_one_level_down’] to process all subfolders of the data_path.

2.2.6 sbx binary files

Scanbox binary files (*.sbx) work out of the box if you set ops[‘input_format’] = “sbx”.

When recording in bidirectional mode some columns might have every other line saturated; to trim these during loading set ops[‘sbx_ndeadcols’]. Set this option to -1 to let suite2p compute the number of columns automatically, a positive integer to specify the number of columns to trim. Joao Couto (@jcouto) wrote the binary sbx parser.
Here is a summary of all the parameters that the pipeline takes, and its default value.

### 3.1 Main settings

These are the essential settings that are dataset-specific.

- **nplanes**: *(int, default: 1)* each tiff has this many planes in sequence
- **nchannels**: *(int, default: 1)* each tiff has this many channels per plane
- **functional_chan**: *(int, default: 1)* this channel is used to extract functional ROIs (1-based, so 1 means first channel, and 2 means second channel)
- **tau**: *(float, default: 1.0)* The timescale of the sensor (in seconds), used for deconvolution kernel. The kernel is fixed to have this decay and is not fit to the data. We recommend:
  - 0.7 for GCaMP6f
  - 1.0 for GCaMP6m
  - 1.25-1.5 for GCaMP6s
- **fs**: *(float, default: 10.0)* Sampling rate (per plane). For instance, if you have a 10 plane recording acquired at 30Hz, then the sampling rate per plane is 3Hz, so set ops['fs'] = 3.
- **do_bidiphase**: *(bool, default: False)* whether or not to compute bidirectional phase offset from misaligned line scanning experiment (applies to 2P recordings only). suite2p will estimate the bidirectional phase offset from ops['nimg_init'] frames if this is set to 1 (and ops['bidiphase']=0), and then apply this computed offset to all frames.
- **bidiphase**: *(int, default: 0)* bidirectional phase offset from line scanning (set by user). If set to any value besides 0, then this offset is used and applied to all frames in the recording.
- **frames_include**: *(int, default: -1)* if greater than zero, only frames_include frames are processed. Useful for testing parameters on a subset of data.
3.2 Output settings

- **preclassify**: *(float, default: 0.3) (new)* apply classifier before signal extraction with probability threshold of “preclassify”. If this is set to 0.0, then all detected ROIs are kept and signals are computed.

- **save_mat**: *(bool, default: False)* whether to save the results in matlab format in file “Fall.mat”. NOTE the cells you click in the GUI will NOT change “Fall.mat”. But there is a new button in the GUI you can click to resave “Fall.mat” in the “File” window.

- **combined**: *(bool, default: True)* combine results across planes in separate folder “combined” at end of processing. This folder will allow all planes to be loaded into the GUI simultaneously.

- **aspect**: *(float, default: 1.0) (**new**)* ratio of um/pixels in X to um/pixels in Y (ONLY for correct aspect ratio in GUI, not used for other processing)

- **report_time**: *(bool, default: True) (**new**)* whether or not to return a timing dictionary for each plane. Timing dictionary will contain keys corresponding to stages and values corresponding to the duration of that stage.

3.3 Registration

- **do_registration**: *(bool, default: True)* whether or not to run registration

- **align_by_chan**: *(int, default: 1)* which channel to use for alignment (1-based, so 1 means 1st channel and 2 means 2nd channel). If you have a non-functional channel with something like td-Tomato expression, you may want to use this channel for alignment rather than the functional channel.

- **nimg_init**: *(int, default: 200)* how many frames to use to compute reference image for registration

- **batch_size**: *(int, default: 200)* how many frames to register simultaneously in each batch. This depends on memory constraints - it will be faster to run if the batch is larger, but it will require more RAM.

- **maxregshift**: *(float, default: 0.1)* the maximum shift as a fraction of the frame size. If the frame is Ly pixels x Lx pixels, then the maximum pixel shift in pixels will be max(Ly,Lx) * ops['maxregshift'].

- **smooth_sigma**: *(float, default: 1.15)* standard deviation in pixels of the gaussian used to smooth the phase correlation between the reference image and the frame which is being registered. A value of >4 is recommended for one-photon recordings (with a 512x512 pixel FOV).

- **smooth_sigma_time**: *(float, default: 0)* standard deviation in time frames of the gaussian used to smooth the data before phase correlation is computed. Might need this to be set to 1 or 2 for low SNR data.

- **keep_movie_raw**: *(bool, default: False)* whether or not to keep the binary file of the non-registered frames. You can view the registered and non-registered binaries together in the GUI in the “View registered binaries” view if you set this to True.

- **two_step_registration**: *(bool, default: False)* whether or not to run registration twice (for low SNR data). keep_movie_raw must be True for this to work.

- **reg_tif**: *(bool, default: False)* whether or not to write the registered binary to tiff files

- **reg_tif_chan2**: *(bool, default: False)* whether or not to write the registered binary of the non-functional channel to tiff files

1P registration settings

- **1Preg**: *(bool, default: False)* whether to perform high-pass spatial filtering and tapering (parameters set below), which help with 1P registration

- **spatial_hp**: *(int, default: 42)* window in pixels for spatial high-pass filtering before registration
• **pre_smooth**: *(float, default: 0)* if > 0, defines stddev of Gaussian smoothing, which is applied before spatial high-pass filtering

• **spatial_taper**: *(float, default: 40)* how many pixels to ignore on edges - they are set to zero (important for vignetted windows, for FFT padding do not set BELOW 3*ops[‘smooth_sigma’])

**Non-rigid registration**

• **nonrigid**: *(bool, default: True)* whether or not to perform non-rigid registration, which splits the field of view into blocks and computes registration offsets in each block separately.

• **block_size**: *(two ints, default: [128, 128])* size of blocks for non-rigid registration, in pixels. HIGHLY recommend keeping this a power of 2 and/or 3 (e.g. 128, 256, 384, etc) for efficient fft

• **snr_thresh**: *(float, default: 1.2)* how big the phase correlation peak has to be relative to the noise in the phase correlation map for the block shift to be accepted. In low SNR recordings like one-photon, I’d recommend a larger value like 1.5, so that block shifts are only accepted if there is significant SNR in the phase correlation.

• **maxregshiftNR**: *(float, default: 5.0)* maximum shift in pixels of a block relative to the rigid shift

### 3.4 ROI detection

• **roidetect**: *(bool, default: True)* whether or not to run ROI detect and extraction

• **sparse_mode**: *(bool, default: False)* whether or not to use sparse_mode cell detection

• **spatial_scale**: *(int, default: 0)* what the optimal scale of the recording is in pixels. if set to 0, then the algorithm determines it automatically (recommend this on the first try). If it seems off, set it yourself to the following values: 1 (=6 pixels), 2 (=12 pixels), 3 (=24 pixels), or 4 (=48 pixels).

• **connected**: *(bool, default: True)* whether or not to require ROIs to be fully connected (set to 0 for dendrites/boutons)

• **threshold_scaling**: *(float, default: 5.0)* this controls the threshold at which to detect ROIs (how much the ROIs have to stand out from the noise to be detected). if you set this higher, then fewer ROIs will be detected, and if you set it lower, more ROIs will be detected.

• **max_overlap**: *(float, default: 0.75)* we allow overlapping ROIs during cell detection. After detection, ROIs with more than ops[‘max_overlap’] fraction of their pixels overlapping with other ROIs will be discarded. Therefore, to throw out NO ROIs, set this to 1.0.

• **high_pass**: *(int, default: 100)* running mean subtraction across time with window of size ‘high_pass’. Values of less than 10 are recommended for 1P data where there are often large full-field changes in brightness.

• **smooth_masks**: *(bool, default: True)* whether to smooth masks in final pass of cell detection. This is useful especially if you are in a high noise regime.

• **max_iterations**: *(int, default: 20)* how many iterations over which to extract cells - at most ops[‘max_iterations’], but usually stops before due to ops[‘threshold_scaling’] criterion.

• **nbinned**: *(int, default: 5000)* maximum number of binned frames to use for ROI detection.
3.5 Signal extraction

- **allow_overlap**: (bool, default: False) whether or not to extract signals from pixels which belong to two ROIs. By default, any pixels which belong to two ROIs (overlapping pixels) are excluded from the computation of the ROI trace.
- **min_neuropil_pixels**: (int, default: 350) minimum number of pixels used to compute neuropil for each cell
- **inner_neuropil_radius**: (int, default: 2) number of pixels to keep between ROI and neuropil donut

3.6 Spike deconvolution

We neuropil-correct the trace $F_{out} = F - ops[\text{\textquoteleft}neucoeff\text{\textquoteleft}] \ast F_{neu}$, and then baseline-correct these traces with an $ops[\text{\textquoteleft}baseline\text{\textquoteleft}]$ filter, and then detect spikes.

- **neucoeff**: (float, default: 0.7) neuropil coefficient for all ROIs.
- **baseline**: (string, default \textquoteleft[maximin]\textquoteleft) how to compute the baseline of each trace. This baseline is then subtracted from each cell. \textquoteleft[maximin]\textquoteleft computes a moving baseline by filtering the data with a Gaussian of width $ops[\text{\textquoteleft}sig\_baseline\text{\textquoteleft}] \ast ops[\text{\textquoteleft}fs\text{\textquoteleft}]$, and then minimum filtering with a window of $ops[\text{\textquoteleft}win\_baseline\text{\textquoteleft}] \ast ops[\text{\textquoteleft}fs\text{\textquoteleft}]$, and then maximum filtering with the same window. \textquoteleft[constant\textquoteleft computes a constant baseline by filtering with a Gaussian of width $ops[\text{\textquoteleft}sig\_baseline\text{\textquoteleft}] \ast ops[\text{\textquoteleft}fs\text{\textquoteleft}]$ and then taking the minimum value of this filtered trace. \textquoteleft[constant\_percentile\textquoteleft computes a constant baseline by taking the $ops[\text{\textquoteleft}prctile\_baseline\text{\textquoteleft}]$ percentile of the trace.
- **win_baseline**: (float, default: 60.0) window for maximin filter in seconds
- **sig_baseline**: (float, default: 10.0) Gaussian filter width in seconds, used before maximin filtering or taking the minimum value of the trace, $ops[\text{\textquoteleft}baseline\text{\textquoteleft}] = \text{\textquoteleft}[maximin\textquoteleft or \text{\textquoteleft}[constant\textquoteleft.
- **prctile_baseline**: (float, optional, default: 8) percentile of trace to use as baseline if $ops[\text{\textquoteleft}baseline\text{\textquoteleft}] = \text{\textquoteleft}[constant\_percentile\textquoteleft].

3.7 Channel 2 settings

- **chan2_thres**: threshold for calling an ROI “detected” on a second channel
Once you’ve run the processing, you can open the output stat.npy file from the GUI. This allows you to explore the data in depth both spatially and in time. In addition you can classify ROIs as ‘cells’ or ‘NOT cells’ (left or right side of screen) and train a classifier to automatically identify the cells as one of these two classes. Note that these categories do not have to be ‘cells’ and ‘NOT cells’, they could be ‘boutons’ and ‘NOT boutons’, we just chose to say ‘cells’ because that’s the most common ROI studied.

You can now drag and drop your stat.npy files into the GUI!

4.1 Different views and colors for ROI panels

4.1.1 Views

To turn off ROIs in views 2-4, uncheck *ROIs on*

1. *ROIs*: ROIs only are drawn
2. *mean img*: mean image is shown in background
3. *mean img (enhanced)*: mean image filtered with a min-max filter shown in background
4. *correlation map*: map of correlated pixels shown in background
5. *mean img (non-functional)*: the non-functional mean image shown in background (if nchannels=2)
4.1.2 Colors

Randomly colored ROI view is the default view. The ROIs in the random view are colored between purple and yellow, with red reserved for ROIs assigned to be RED based on the non-functional channel (you can change the threshold for calling a cell RED with the number next to the chan2 prob button). The other color views color the ROIs based on their statistics. The values of those statistics are shown in the colorbar below the buttons.

Here is more info about the less explanatory views:

4.1.3 Correlations

In correlation color view, the selected cell’s activity (or the mean of the selected cells’ activities) is correlated with the activity of all the other ROIs. The ROIs are colored according to these correlations. The bin in which to compute the correlations can be chosen (in units of frames). The default bin size is the number of frames per second (ops['fs']).

If a 1D external variable is loaded, then the corr with 1D var button is activated. The cells are then colored according to their correlation with the external variable. The bin size is determined by the box next to the correlations button.
4.1.4 Correlations with 1D var

You can load an external stimulus or behavioral trace (1D) using “File - Load behavior or stim trace (1D only)”. The GUI expects a *.npy file that is the same length as the data in time (F.shape[1] from “F.npy”). You can then look at the correlation of each cell with this trace. And it will be plotted along with the cell traces if you select multiple cells or in the “Visualize” menu.

4.1.5 Rastermap / custom

Rastermap: Click ‘Visualize selected cells’ in the Visualizations menu and run rastermap on the cells. The selected cells (which could be all cells on LEFT or RIGHT) will then be colored based on their position in the rastermap.

Custom map: Use ‘Load custom hue’ in the Visualizations menu to load a *.npy file with the same number of values as ROIs (length of stat) and these values will become the hues of the cells (scaled to between 0 and 1) for the HSV map. If you do rastermap after this then the colors will change and vice versa this will overwrite the rastermap colors.

4.2 Buttons / shortcuts for cell selection

4.2.1 Mouse control

- double left click = returns to full view in ALL PLOTS
- left click = select cell
- left click + CTRL = select multiple cells
- left click + drag = moves field of view
- right click = flip selected cell(s) from left<->right, – or if clicked in trace view, will open up “export” option
- scroll wheel = zoom in and out

4.2.2 Keyboard shortcuts

- Esc = returns to full view
- Delete = removes box from draw selection from window For the letters, just press the letter (do not capitalize)
- O = turn of ROIs in non-ROI view
- Q-U = different views (can change saturation with slider)
- A-M = different color maps
- Left and right keys = cycle between cells of same panel
- Up Key = flip selected cell to other panel
- Alt+Enter = merge selected ROIs
- note you can also ask the GUI to auto-suggest merges with the Merge>Auto-suggest merges window *
4.2.3 Multi-cell selection

You can select multiple cells by holding down CTRL while left-clicking on cells. If you are in ‘cells’ or ‘NOT cells’ view (not ‘both’ view), then several buttons for multi-cell selection activate.

The draw selection button activates a box that you can drag and resize to select multiple cells. To delete the box, click the Delete key. select top n selects n=X top neurons from the current colormap. For instance, in ‘skew’ view, select top n will select the most skewed neurons. In ‘correlation’ view, it will choose the most correlated neurons with the currently selected neuron.

4.3 Trace view (bottom row)

When one cell is selected, the fluorescence, neuropil and deconvolved traces are shown for the chosen cell in the bottom row of the GUI. When multiple cells are selected, you can choose what type of traces to view with the drop-down menu in the lower left:

- F: fluorescence
- Fneu: neuropil fluorescence
- F - 0.7*Fneu: corrected fluorescence
- deconvolved: deconvolution of corrected fluorescence

You can resize the trace view with the triangle buttons (bigger = , smaller = ). If multiple cells are selected, you can vary how much the traces overlap with the +/- buttons.

You can select as many cells as you want, but by default only 40 of those will be plotted. You can increase or decrease this number by changing the number in the box below max # plotted.
You can hide the fluorescence, neuropil and/or the deconvolved traces by toggling the checkboxes or using the keys as follows:

Deconvolved - N key Neuropil - B Key Fluorescence - V Key

## 4.4 Classifying cells

suite2p comes with a built-in classifier (based on our own manual curation of GCaMP6s imaging of cells in cortex). The default classifier is initialized as the built-in classifier, but can be modified by the user.

After running suite2p, the cells are automatically classified by the default classifier (at the time of running the pipeline), and these cell probabilities are shown as the colors in the classifier view. You can then further manually curate this data (flipping cells left and right depending on your criteria).

### 4.4.1 Adding data to a classifier

You can add this manually curated data to an already built classifier:

1. Load a classifier by going to the “Classifier” menu and clicking “Load”. Choose the default classifier, or load another classifier that you’ve built and saved with the from file option.

2. Click the add current data to classifier button. This will either overwrite the classifier file that is loaded, or you can specify a file location for the classifier with this newly added data.

### 4.4.2 Building your own classifier

Go to the “Classifier” menu and click “Build”. A window will pop up and in the window you can add datasets as training samples for the classifier. Click the Load iscell.npy button and add an iscell.npy file. You can add as many as you like, then click build classifier, and it will ask you to specify a file location for the new classifier. Then you can load the classifier that you built into the GUI, or you can save it as your default classifier.

## 4.5 Visualizing activity

Go to the “Visualizations” menu and click “Visualize selected cells”. If only one ROI is selected, then all ROIs in that view (cell or not cell) will be plotted. Otherwise the selected cells are plotted. You can sort the neurons by their principal component weights, or by our algorithm rastermap by clicking the compute buttons. Once you click the compute buttons, they will be grayed out, because you can’t compute them again (they won’t change). The plot below shows a mesoscope recording sorted by rastermap. You can change between sorting by rastermap and by the PCs by using the drop-down menu.
The red box allows you to zoom in on specific segments of the recording. You can move it by dragging the mouse when in the box, or with the arrow keys. You can resize it by using the diamond handles on the sides of the box, or by holding down the shift key and using the arrow keys.

If you click the *show selected cells in GUI* button, then the cells surrounded by the red box will show up as white in the GUI.
4.6 Manual adding of ROIs

You can add ROIs in the File>Manual labelling. You MUST keep the binary file for the computing of the mask's activity across time. When you save and exit the ROIs will be added to the *.npy files as the first N ROIs (where N is the number that you drew).
4.7 Merging ROIs

You can merge selected ROIs (multi-select with CTRL) by pressing ALT+ENTER, or get suggested merges in the “Merge ROI” menu. The merged ROIs then must be saved before you close the GUI to write the new ROIs to the *.npy files. Each merged ROI is appended to the end of the list of ROIs (in stat), and the ROIs that were merged to create it are in the key ‘imerge’. Note in the stat file and other files the original ROIs (that create the ROI) are NOT removed so that you retain the original signals and original suite2p output. In the GUI ROI view the merged ROIs are shown.

The merging of fluorescence is done by taking the mean of the selected cells’ fluorescences. The list of merges are available in the stat for you to choose alternative strategies for combining signals.

4.8 View registered binary

Open the “Registration” menu and click “View registered binary”. A window will pop up with the binary file loaded (first row) along with the registration shifts (second row), and the fluorescence of a selected ROI (third row). If ops['keep_movie_raw']=1, then both the unregistered and registered binaries will be shown in the first row. You can select an ROI by typing in the ROI number in the upper right.

You can zoom in and out on any of the plots. The shift plot and the fluorescence plot have linked x-axes. To return to full view, double-click on the plot that you want to recenter.

When not playing the movie, you can click on the shift plot and the fluorescence plot to go to a specific point in time in the movie. You can also seek through the movie by clicking the slide bar. The left and right arrow keys will move the slide bar incrementally. The space bar will pause and play the movie.
You can also view all the masks, and go from cell to cell by clicking on them.
4.8.1 Z-stack Alignment

You can also now load a Z-stack into this view. You can compute the z-position of the recording across time IF you keep the registered binary file. It expects the Z-stack to be a tiff with number of planes by number of pixels in Y by number of pixels in X. The results of the correlation between z-stack and each frame are saved in `ops['zcorr']` which is number of planes (in Z) x number of frames. The GUI smooths this matrix across Z and then takes the max and plots the max across time in the third row.

4.9 View registration metrics

Open the “Registration” menu and click “View registration metrics”. A window will pop up with `ops['regDX']` and `ops['regPC']` plotted. The `ops['regPC']`’s are computed by taking the principal components of the registered movie. `ops['regPC'][0,0,:,:]` is the average of the top 500 frames of the 1st PC, `ops['regPC'][1,0,:,:]` is the average of the bottom 500 frames of the 1st PC. `ops['regDX']` quantifies the movement in each PC (iPC) by registering `ops['regPC'][0,iPC,:,:]` and `ops['regPC'][1,iPC,:,:]` to the reference images and computing the registration shifts.

The first plot in the upper left shows the magnitude of the shifts (both rigid and non-rigid) in the PCs (`ops['regDX']`). The second row of plots are meant to help explore the direction of the PC. The first image is the “difference” between the top and the bottom of the PC. The second image is the “merged” image of the top and bottom of the PC. The third image allows you to flip between the top and bottom PCs using the “play” button.

The left and right arrow keys will change the PC number (or you can type in a number). The space bar will pause and play the movie.
The example below shows a movie that has been rigid registered but not non-rigid registered. The metrics suggest that non-rigid registration should also be performed on this recording.
F.npy: array of fluorescence traces (ROIs by timepoints)
Fneu.npy: array of neuropil fluorescence traces (ROIs by timepoints)
spks.npy: array of deconvolved traces (ROIs by timepoints)
stat.npy: list of statistics computed for each cell (ROIs by 1)
ops.npy: options and intermediate outputs (dictionary)
iscell.npy: specifies whether an ROI is a cell, first column is 0/1, and second column is probability that the ROI is
a cell based on the default classifier

All can be loaded in python with numpy

```python
import numpy as np

F = np.load('F.npy', allow_pickle=True)
Fneu = np.load('Fneu.npy', allow_pickle=True)
spks = np.load('spks.npy', allow_pickle=True)
stat = np.load('stat.npy', allow_pickle=True)
ops = np.load('ops.npy', allow_pickle=True)
ops = ops.item()
iscell = np.load('iscell.npy', allow_pickle=True)
```

### 5.1 MATLAB output

If 'save_mat'=1, then a MATLAB file is created Fall.mat. This will contain ops, F, Fneu, stat, spks and iscell. The “iscell” assignments are only saved ONCE when the pipeline is finished running. If you make changes in the GUI to the cell assignments, ONLY iscell.npy changes. To load a modified iscell.npy into MATLAB, I recommend using this package: npy-matlab. Alternatively there is a new save button in the GUI (in the file menu) that allows you to save the iscell again to the Fall.mat file.
5.2 NWB Output

If `ops['save_NWB']`=1, then an NWB file is created `ophys.nwb`. This will contain the fields from `ops` and `stat` required to load back into the GUI, along with `F`, `Fneu`, `spks` and `iscell`. If the recording has multiple planes, then they are all saved together like in combined view. See fields below:

- **stat:** `stat['ypix']`, `stat['xpix']` (if multiplane `stat['iplane']`) are saved in ‘pixel_mask’ (called ‘voxel_mask’ in multi-plane).
- **ops:** ‘meanImg’, ‘max_proj’, ‘Vcorr’ are saved in Images ‘Backgrounds_k’ where k is the plane number, and have the same names. optionally if two channels, ‘meanImg_chan2’ is saved.
- **iscell:** saved as an array ‘iscell’

`F`, `Fneu`, `spks` are saved as roi_response_series ‘Fluorescence’, ‘Neuropil’, and ‘Deconvolved’.

5.3 Multichannel recordings

Cells are detected on the `ops['functional_chan']` and the fluorescence signals are extracted from both channels. The functional channel signals are saved to `F.npy` and `F_neu.npy`, and non-functional channel signals are saved to `F_chan2.npy` and `Fneu_chan2.npy`.

5.4 stat.npy fields

- **ypix:** y-pixels of cell
- **xpix:** x-pixels of cell
- **med:** (y,x) center of cell
- **lam:** pixel mask \( \sum(\text{lam} \ast \text{frames}[\text{ypix, xpix,:}]) = \text{fluorescence} \)
- **npix:** number of pixels in ROI
- **npix_norm:** number of pixels in ROI normalized by the mean of npix across all ROIs
- **radius:** estimated radius of cell from 2D Gaussian fit to mask
- **aspect_ratio:** ratio between major and minor axes of a 2D Gaussian fit to mask
- **compact:** how compact the ROI is (1 is a disk, >1 means less compact)
- **footprint:** spatial extent of an ROI’s functional signal, including pixels not assigned to the ROI; a threshold of 1/5 of the max is used as a threshold, and the average distance of these pixels from the center is defined as the footprint
- **skew:** skewness of neuropil-corrected fluorescence trace
- **std:** standard deviation of neuropil-corrected fluorescence trace
- **overlap:** which pixels overlap with other ROIs (these are excluded from fluorescence computation)
- **ipix_neuropil:** pixels of neuropil mask for this cell

Here is example code to make an image where each cell (without its overlapping pixels) is a different “number”:
stat = np.load('stat.npy')
ops = np.load('ops.npy').item()

im = np.zeros((ops['Ly'], ops['Lx']))

for n in range(0, ncells):
    ypix = stat[n]['ypix'][~stat[n]['overlap']]
xpix = stat[n]['xpix'][~stat[n]['overlap']]
im[ypix, xpix] = n+1

plt.imshow(im)
plt.show()

(There is no longer ipix like in the matlab version. In python note you can access a 2D array like X[ys, xs] = lam. In Matlab, this would cause a broadcast of all the pairs of ys and xs, which is why ipix = ys + (xs-1) * Ly was a useful temporary variable to have around for linear indexing into arrays. In Python, the equivalent ipix would be ipix = yx + xs * Lxy.)

5.5. ops.npy fields

This will include all of the options you ran the pipeline with, including file paths. During the running of the pipeline, some outputs are added to ops.npy:

- **reg_file**: location of registered binary file
- **Ly**: size of Y dimension of tiffs/h5
- **Lx**: size of X dimension of tiffs/h5
- **nframes**: number of frames in recording
- **yrange**: valid y-range used for cell detection (excludes edges that were shifted out of the FOV during registration)
- **xrange**: valid x-range used for cell detection (excludes edges that were shifted out of the FOV during registration)
- **refImg**: reference image used for registration
- **yoff**: y-shifts of recording at each timepoint
- **xoff**: x-shifts of recording at each timepoint
- **corrXY**: peak of phase correlation between frame and reference image at each timepoint
- **meanImg**: mean of registered frames
- **meanImgE**: a median-filtered version of the mean image
- **Vcorr**: correlation map (computed during cell detection)
- **filelist**: List of the image file names (e.g. tiff) that were loaded, in the order that Suite2p processed them.
- **date_proc**: Date and time that the analysis was run.
In the matlab version of suite2p, Henry Dalgleish wrote the utility “registers2p” for multiday alignment, but it has not been ported to python.

I recommend trying to run all your recordings together (add all the separate folders to data_path). This has worked well for people who have automated online registration on their microscope to register day by day (scanimage 2018b (free) offers this capability). I highly recommend checking this out - we have contributed to a module in that software for online Z-correction that has greatly improved our recording quality.

However, if there are significant non-rigid shifts between days (angle changes etc) and low SNR then concatenating recordings and running them together will not work so well.

In this case, (if you have a matlab license) here is a package written by Adam Ranson which is based on similar concepts as ‘registers2p’ by Henry Dalgleish that takes the output of suite2p-python directly: https://github.com/ransonar/ROIMatchPub.
CHAPTER
SEVEN

DEVELOPER DOCUMENTATION

7.1 Versioning

There’s a rare issue that developers may face when calling `suite2p --version` on their command line. You may get an incorrect version number. To fix this issue, one should use the following command:

```
$ git fetch --prune --unshallow
```

7.2 Testing

Before contributing to Suite2P, please make sure your changes pass all our tests.

7.2.1 Downloading Test Data

To run the tests (located in the `tests` subdirectory of your working `suite2p` directory), you’ll first need to download our test data. Suite2p depends on dvc to download the test data.

**Note:** Before testing, make sure you have dvc and pydrive2 installed. Navigate to the suite2p directory and use the following command to install both dvc and pydrive2.

```
$ pip install -e .[data]
zsh users should use the following:
$ pip install -e .docs
```

Use the following command to download the test data into the `data` subdirectory of your working `suite2p` directory.

```
$ dvc pull
```
7.2.2 Running the tests

Tests can then be easily run with the following command:

$ python setup.py test

If all the tests pass, you’re good to go!
8.1 Cropped field-of-view

Some issues on this: #273, #207, #125.

Why does this happen? suite2p crops the field-of-view so that areas that move out of view on the edges are not used for ROI detection and signal extraction. These areas are excluded because they are not always in the FOV - they move in and out and therefore activity in these regions is unreliable to estimate.

suite2p determines the region to crop based on the maximum rigid shifts in XY. You can view these shifts with the movie in the “View registered binary” window. If these shifts are too large and don’t seem to be accurate (low SNR regime), you can decrease the maximum shift that suite2p can estimate by setting ops[‘maxregshift’] lower than its default (which is 0.1 ≈ 10% of the size of the FOV). suite2p does exclude some of the large outlier shifts when computing the crop, and determines the threshold of what is an “outlier” using the parameter ops[‘th_badframes’]. Set this lower to increase the number of “outliers”. These “outliers” are labelled as ops[‘badframes’] and these frames are excluded also from ROI detection.

You can add frames to this list of ops[‘badframes’] by creating a numpy array (0-based, the first frame is zero) and save it as bad_frames.npy in the folder with your tiffs (if you have multiple folders, save it in the FIRST folder with tiffs, or if you have subfolders with ‘look_one_level_down’ it should be in the parent folder). See this page inputs for more info.

8.2 Deconvolution means what?

There is a lot of misinformation about what deconvolution is and what it isn’t. Some issues on this: #267, #202, #169

TLDR: Deconvolution will NOT tell you how many spikes happened in a neuron - there is too much variability in the calcium signal to know that. Our deconvolution has NO sparsity constraints and we recommend against thresholding the output values because they contain information about approximately how many spikes occurred. We found that using the raw deconvolved values gave us the most reliable responses to stimuli (as measured by signal variance).

See this figure from our review paper for reference:
Figure 4. The limits of spike deconvolution. (a) Recorded fluorescence trace in blue, and reconstructions from spike deconvolution (red) and from direct regression on spike times (yellow). (b) Ground truth spike times recorded by simultaneous electrophysiology [47]. (c) Spike deconvolution result, and the correlation with ground truth in bins of 10, 40 and 160 ms (non-negative, using the OASIS implementation [48]). (d) Regression on ground truth spike times to obtain “optimal” amplitudes. (e) Fluorescence traces aligned to spike times and baseline subtracted at 0 timelag. (f) Variability of single-spike “optimal” amplitudes from GT regression sets the limit of possible spike deconvolution performance. (g) Correlation of binned ground truth spike trains with deconvolved and “optimal” amplitudes. At large bin sizes, deconvolution saturates the possible maximal performance. (h) Failures of deconvolution at small bin sizes correspond to ambiguities of spike timing on the order of 100 ms, reflected in the shape of the cross-correlogram between deconvolved spike trains and ground truth.
Long answer (mostly from #267):

There is an unknown scaling factor between fluorescence and # spikes, which is very hard to estimate. This is true both for the raw dF, or dF/F, and for the deconvolved amplitudes, which we usually treat as arbitrary units. The same calcium amplitude transient may have been generated by a single spike, or by a burst of many spikes, and for many neurons it is very hard to disentangle these, so we don’t try. Few spike deconvolution algorithms try to estimate single spike amplitude (look up “MLspike”), but we are in general suspicious of the results, and usually have no need for absolute numbers of spikes.

As for the question of thresholding, we always recommend not to, because you will lose information. More importantly, you will treat 1-spike events the same as 10-spike events, which isn’t right. There are several L0-based methods that return discrete spike times, including one we’ve developed in the past, which we’ve since shown to be worse than the vanilla OASIS method (see our Jneurosci paper). We do not use L1 penalties either, departing from the original OASIS paper, because we found that hurts in all cases (see Jneurosci).

How do you compare across cells then if these values are arbitrary to some extent?

If you need to compare between cells, you would usually be comparing effect sizes, such as tuning width, SNR, choice index etc. which are relative quantities, i.e. firing rate 1 / firing rate 2. If you really need to compare absolute firing rates, then you need to normalize the deconvolved events by the F0 of the fluorescence trace, because the dF/F should be more closely related to absolute firing rate. Computing the F0 has problems in itself, as it may sometimes be estimated to be negative or near-zero for high SNR sensors like gcamp6 and 7. You could take the mean F0 before subtracting the neuropil and normalize by that, and then decide on a threshold to use across all cells, but at that point you need to realize these choices will affect your result and interpretation, so you cannot really put much weight on them. For these reasons, I would avoid making statements about absolute firing rates from calcium imaging data, and I don’t know of many papers that make such statements.

### 8.3 Multiple functional channels

If you have two channels and they both have functional activity, then to process both you need to run suite2p in a jupyter notebook. Here is an example notebook for that purpose: multiple_functional_channels.ipynb

### 8.4 Z-drift

It’s not frequently asked about but it should be :) 

In the GUI in the “View registered binary” window you can now load in a z-stack and compute the z-position of the recording across time.

Scanimage now can do z-correction ONLINE for you!
**8.5 No signals in manually selected ROIs**

If you circle an ROI in the manual selection GUI on top of another ROI and ops['allow_overlap'] is 0 or False, then that ROI will have no activity because it has no non-overlapping pixels. You can change this after processing with:

```python
import numpy as np
np.load('ops.npy', allow_pickle=True).item()
np.save('ops_original.npy', ops)
ops['allow_overlap'] = True
np.save('ops.npy', ops)
```

Thanks @marysethomas, see full issue here: #651,
You can register your frames using the first channel of the recording, or using the second channel. Say your first channel shows GCaMP and your second channel shows td-Tomato, you might want to use the second channel for registration if it has higher SNR. If so, set `ops['align_by_chan']=2`. Otherwise, leave `ops['align_by_chan']=1` (default).

### 9.1 Finding a target reference image

To perform registration, we need a reference image to align all the frames to. This requires an initial alignment step. Consider we just took the average of a subset of frames. Because these frames are not motion-corrected, the average will not be crisp - there will be fuzzy edges because objects in the image have been moving around across the frames. Therefore, we do an initial iterative alignment procedure on a random subset of frames in order to get a crisp reference image for registration. We first take `ops['nimg_init']` random frames of the movie. Then from those frames, we take the top 20 frames that are most correlated to each other and take the mean of those frames as our initial reference image. Then we refine this reference image iteratively by aligning all the random frames to the reference image, and then recomputing the reference image as the mean of the best aligned frames.

The function that performs these steps can be run as follows (where `ops` needs the `reg_file`, `Ly`, `Lx`, and `nimg_init` parameters):

```python
def register.pick_initial_reference(ops):
    # Perform initial alignment and find initial reference image
    refImg = ...
    return refImg
```

Here is an example reference image on the right, compared to just taking the average of a random subset of frames (on the left):
If the reference image doesn't look good, try increasing `ops['nimq_init']`.

### 9.2 Registering the frames to the reference image

Once the reference image is obtained, we align each frame to the reference image. The frames are registered in batches of size `ops['batch_size']` (default is 200 frames per batch).

We first perform rigid registration (assuming that the whole image shifts by some (dy,dx)), and then optionally after that we perform non-rigid registration (assuming that subsegments of the image shift by separate amounts). To turn on non-rigid registration, set `ops['nonrigid']=True`. We will outline the parameters of each registration step below.

### 9.3 1. Rigid registration

Rigid registration computes the shifts between the frame and the reference image using phase-correlation. We have found on simulated data that phase-correlation is more accurate than cross-correlation. **Phase-correlation** is a well-established method to compute the relative movement between two images. Phase-correlation normalizes the Fourier spectra of the images before multiplying them (whereas cross-correlation would just multiply them). This normalization emphasizes the correlation between the higher frequency components of the images, which in most cases makes it more robust to noise.

**Cross-correlation**
You can set a maximum shift size using the option `ops['maxregshift']`. By default, it is 0.1, which means that the maximum shift of the frame from the reference in the Y direction is $0.1 \times \text{ops['Ly']}'$ and in X is $0.1 \times \text{ops['Lx']}'$ where Ly and Lx are the Y and X sizes of the frame.

After computing the shifts, the frames are shifted in the Fourier domain (allowing subpixel shifts of the images). The shifts are saved in `ops['yoff']` and `ops['xoff']` for y and x shifts respectively. The peak of the phase-correlation of each frame with the reference image is saved in `ops['corrXY']`. 
You can run this independently from the pipeline, if you have a reference image (ops requires the parameters non-rigid=False, num_workers, and maxregshift):

```python
maskMul, maskOffset, cfRefImg = register.prepare_masks(refImg)
refAndMasks = [maskMul, maskOffset, cfRefImg]
aligned_data, yshift, xshift, corrXY, yxnr = register.phasecorr(data, refAndMasks, ops)
```

(see bioRxiv preprint comparing cross/phase here)

## 9.4 2. Non-rigid registration (optional)

If you run rigid registration and find that there is still motion in your frames, then you should run non-rigid registration. Non-rigid registration divides the image into subsections and computes the shift of each subsection (called a block) separately. Non-rigid registration will approximately double the registration time.

The size of the blocks to divide the image into is defined by `ops['block_size'] = [128, 128]` which is the size in Y and X in pixels. If Y is the direction of line-scanning for 2p imaging, you may want to divide it into smaller blocks in that direction.

![many overlapping squares]

Each block is able to shift up to `ops['maxregshiftNR']` pixels in Y and X. We recommend to keep this small unless you're in a very high signal-to-noise ratio regime and your motion is very large. For subpixel shifts, we use Kriging interpolation and run it on each block.

Phase correlation of each block:
In a low signal-to-noise ratio regime, there may be blocks which on a given frame do not have sufficient information from which to align with the reference image. We compute a given block’s maximum phase correlation with the reference block, and determine how much greater this max is than the surrounding phase correlations. The ratio between these two is defined as the $\text{snr}$ of that block at that given time point. We smooth over high $\text{snr}$ blocks and use bilinear interpolation to upsample create the final shifts:
We then use bilinear interpolation to warp the frame using these shifts.

9.5 Metrics for registration quality

The inputs required for PC metrics are the following fields in ops: nframes, Ly, Lx, reg_file. Your movie must have at least 1500 frames in each plane for the metrics to be calculated. You can run on the red channel (ops['reg_file_chan2']) if use_red=True. The outputs saved from the PC metrics are ops['regDX'], ops['tPC'] and ops['regPC'].

```python
from suite2p.registration import metrics
ops = metrics.get_pc_metrics(ops, use_red=False)
```

ops['tPC'] are the time courses of each of the principal components of the registered movie. Note the time-course is not the entire movie, it’s only the subset of frames used to compute the PCs (2000-5000 frames equally sampled throughout the movie).

ops['regPC'] are computed from the spatial principal components of the registered movie. ops['regPC'][0,0,:,:] is the average of the top 500 frames of the 1st PC, ops['regPC'][1,0,:,:] is the average of the bottom 500 frames of the 1st PC. ops['regDX'] quantifies the movement in each PC (iPC) by registering ops['regPC'][0, iPC,:,:] and ops['regPC'][1,iPC,:,:] to the reference image ops['refImg'] (if available, if not the mean of all the frames is used as the reference image) and computing the registration shifts.

Here’s a twitter thread with multiple examples.
### 9.5.1 CLI Script

Suite2p provides a CLI (Command-Line Interface) script that calculates the registration metrics for a given input tif and outputs some statistics on those metrics. You can use this script to determine the quality of registration and tune your registration parameters (e.g., determine if non-rigid registration is necessary).

To run the script, use the following command:

```
$ reg_metrics <INSERT_OPS_DATA_PATH> # Add --tiff_list <INSERT_INPUT_TIF_FILENAME_HERE>.tif to select a subset of tifs
```

Once you run the `reg_metrics` command, registration will be performed for the input file with default ops parameters and an output similar to the following will be shown:

```
# Average NR refers to the average nonrigid offsets of the blocks for a PC
# Max NR refers to the max nonrigid offsets of the blocks for a PC
Plane 0:
Avg_Rigid: 0.000000 Avg_Average NR: 0.028889 Avg_Max NR: 0.120000
Max_Rigid: 0.000000 Max_Average NR: 0.044444 Max_Max NR: 0.200000
```

For each `nplane`, these statistics (Average and Max) are calculated across PCs on the offsets found in `ops['regDX']`. If the registration works perfectly and most of the motion is removed from the registered dataset, these scores should all be very close to zero.

**Important:** Make sure to also inspect the registered video to check the quality of registration. You can see an example of how this is done in the GUI [here](#).

You may notice that upon visual inspection, the registered video may look fine/contain little motion even if the statistics are not close to zero. You should always visually check the registration output and prioritize what your eyes say over what the CLI script reports.

**Note:** All suite2p registration settings can be modified in this CLI script. Just pass the setting with its value as an optional argument. For instance,

```
$ reg_metrics path_to_data_tif --nplanes 2 --smooth_sigma 1.2
```

runs the script with `ops['nplanes'] = 2` and `ops['smooth_sigma'] = 1.2`. You can see all the arguments `reg_metrics` takes with the following command:

```
$ reg_metrics --help
```
10.1 Summary
The cell detection algorithm consists of reducing the dimensionality of the data (principal components computation), smoothing spatial principal components, finding peaks in these components, and extending ROIs spatially around these peaks. On each iteration of peak extraction, the neuropil is estimated from large spatial masks and subtracted from the spatial components. This is to improve cell detection and to help avoid extracting neuropil components with large spatial extents.

10.2 SVDs ( = PCs) of data
Before computing the principal components of the movie, we bin the data such that we have at least as many frames to take the SVD of as specified in the option `ops['navg_frames_svd']`. The bin size will be the maximum of `nframes/ops['navg_frames_svd']` and `ops['tau'] * ops['fs']` (the number of samples per transient). We then bin the movie into this bin size and subtract the mean of the binned movie across time. Then we smooth the movie in Y and X with a gaussian filter of standard deviation `sig = ops['diameter']/10`. The we normalize the pixels by their noise variance. The noise variance is variance of each pixel in the movie across time (at least 1e-10). Then we compute the covariance of the movie (`mov @ mov.T`). Then we compute the SVD of the covariance and keep the top `ops['nsvd_for_roi']` spatial components (components that are Y x X).

The function that performs this is `celldetect2.getSVDdata` and it requires the ops described above, and Ly, Lx, yrange, xrange, and a reg_file location.

10.3 Sourcery
After the spatial components are found, we perform an iterative algorithm to find the cells in the components. Each iteration consists of the following steps:

1. **Smoothing of spatial components**: The components are smoothed with a Gaussian filter in Y and X with standard deviation `sig = ops['diameter']` (this matrix is called `us`). Note that diameter can be a list (for unequal pixel/um in Y and X). Next the mean of the squared smoothed components is computed. The mean of the squared un-smoothed components is also computed. The correlation map is defined as the element-wise division of the smoothed components by the unsmoothed components. The function that computes the correlation map is `celldetect2.getVmap`.

2. **Detection of peaks in correlation map**: On each iteration, up to 200 peaks are extracted from the correlation map. These are the largest remaining peaks such that they are greater than the threshold, which is set to be proportional to the median of the peaks in the whole correlation map: `ops['threshold_scaling'] *`
The initial activity code for this newly detected peak is the value of \( \text{us} \) (Gaussian smoothed PCs) at this peak. This is a vector of values across the PCs (nPCs in length).

3. **ROI extension**: The ROI is iteratively extended around its currently defined pixels +/- 1 in each direction. First, the new pixel weights (\( \text{lam} \)) of the extended ROI are computed. The weights \( \text{lam} \) are the unsmoothed PCs projected into the code dimension. The pixels that are greater than \( \max(\text{lam})/5 \) are kept. The \( \text{lam} \)'s are normalized to be unit norm. The new code is recomputed from the new weights, and is the unsmoothed PCs projected onto the \( \text{lam} \) weights. Then this extension procedure is repeated until no pixels are greater than \( \max(\text{lam})/5 \).

4. **Neuropil computation**: Now that the new codes are computed, the neuropil is estimated. We set spatial basis functions for the neuropil, which are raised cosines that tile the FOV. The parameter \( \text{ops['ratio_neuropil']} \) determines how big you expect the neuropil basis functions to be relative to the cell diameter (\( \text{ops['diameter']} \)). The default is 6. This results in a tiling of 7x7 raised cosines if your FOV is 512x512 pixels and your diameter is 12 pixels. For one-photon recordings, we recommend setting \( \text{ops['ratio_neuropil']} \) to 2 or 3. Next we perform regression to compute the contribution of the neuropil on the PCs, and we subtract the estimated neuropil contribution from the \( U \) PCs. And these steps are repeated until the stopping criterion is reached.

**Stopping criterion**: The number of cells detected in the first iteration is defined as \( N_{\text{first}} \). The cell detection is stopped if the number of cells detected in the current iteration is less than \( N_{\text{first}}/10 \) or if the iteration is the last iteration (defined by \( \text{ops['max_iterations']} \)).

**Refinement step**: We remove masks which have more than a fraction \( \text{ops['max_overlap']} \) of their pixels overlapping with other masks. Also, if \( \text{ops['connected']}=1 \), then only the connected regions of ROIs are kept. If you are looking for dendritic components, you may want to set \( \text{ops['connected']}=0 \).
CHAPTER
ELEVEN

SIGNAL EXTRACTION
Our spike deconvolution in the pipeline is based on the OASIS algorithm (see OASIS paper). We run it with only a non-negativity constraint - no L0/L1 constraints (see this paper for more details on why).

We first baseline the traces using the rolling max of the rolling min. Here is an example of how the pipeline processes the traces (and how to run your own data separately if you want):

```python
# compute deconvolution
from suite2p.extraction import dcnv
import numpy as np

tau = 1.0  # timescale of indicator
fs = 30.0  # sampling rate in Hz
neucoeff = 0.7  # neuropil coefficient
# for computing and subtracting baseline
baseline = 'maximin'  # take the running max of the running min after smoothing with
sig_baseline = 10.0  # in bins, standard deviation of gaussian with which to smooth
win_baseline = 60.0  # in seconds, window in which to compute max/min filters

ops = {'tau': tau, 'fs': fs, 'neucoeff': neucoeff,
       'baseline': baseline, 'sig_baseline': sig_baseline, 'win_baseline': win_baseline}

# load traces and subtract neuropil
F = np.load('F.npy')
Fneu = np.load('Fneu.npy')
Fc = F - ops['neucoeff'] * Fneu

# baseline operation
Fc = dcnv.preprocess(
    F=Fc,
    baseline=ops['baseline'],
    win_baseline=ops['win_baseline'],
    sig_baseline=ops['sig_baseline'],
    fs=ops['fs'],
    prctile_baseline=ops['prctile_baseline'])

# get spikes
spks = dcnv.oasis(F=Fc, batch_size=ops['batch_size'], tau=ops['tau'], fs=ops['fs'])
```


13.1 Submodules

13.2 suite2p.io.binary module

class suite2p.io.binary.BinaryFile(Ly, Lx, read_filename, write_filename=None)
   Bases: object

   bin_movie(bin_size, x_range=None, y_range=None, bad_frames=None, reject_threshold=0.5)
      Returns binned movie that rejects bad_frames (bool array) and crops to (y_range, x_range).

      Parameters
         • bin_size (int) – The size of each bin
         • x_range (int, int) – Crops the data to a minimum and maximum x range.
         • y_range (int, int) – Crops the data to a minimum and maximum y range.
         • bad_frames (int array) – The indices to not include.
         • reject_threshold (float) –

      Returns frames – The frames

      Return type  nImg x Ly x Lx

   close()
      Closes the file.

      Return type  None

static convert_numpy_file_to_suite2p_binary(from_filename, to_filename)
   Works with npz files, pickled npy files, etc.

      Parameters
         • from_filename (str) – The npy file to convert
         • to_filename (str) – The binary file that will be created

      Return type  None

property data:  numpy.ndarray
   Returns all the frames in the file.

      Returns frames – The frame data

      Return type  nImg x Ly x Lx
iter_frames(batch_size=1, dtype=<class 'numpy.float32'>)
Iterates through each set of frames, depending on batch_size, yielding both the frame index and frame data.

Parameters
- **batch_size** (int) – The number of frames to get at a time
- **dtype** (np.dtype) – The numpy data type that the data should return as

Yields
- **indices** (array int) – The frame indices.
- **data** (batch_size x Ly x Lx) – The frames

ix(indices, is_slice=False)
Returns the frames at index values “indices”.

Parameters
- **indices** (int array) – The frame indices to get
- **is_slice** (bool, default False) – if indices are slice, read slice with “read” function and return

Returns **frames** – The requested frames

Return type len(indices) x Ly x Lx

property n_frames: int
total number of frames in the read_file.

Return type int

property nbytes
total number of bytes in the read_file.

property nbytesread
number of bytes per frame (FIXED for given file)

read(batch_size=1, dtype=<class 'numpy.float32'>)
Returns the next frame(s) in the file and its associated indices.

Parameters
- **batch_size** (int) – The number of frames to read at once.
- **frames** (batch_size x Ly x Lx) – The frame data

Return type Optional[[ndarray, ndarray]]

sampled_mean()
Returns the sampled mean.

Return type float

property shape: Tuple[int, int, int]
The dimensions of the data in the file

Return type Tuple[int, int, int]

Returns
- **n_frames** (int) – The number of frames
- **Ly** (int) – The height of each frame
- **Lx** (int) – The width of each frame
property size: int
Returns the total number of pixels

Returns size
Return type int

write(data)
Writes frame(s) to the file.

Parameters data (2D or 3D array) – The frame(s) to write. Should be the same width and height as the other frames in the file.

Return type None

class suite2p.io.binary.BinaryFileCombined(LY, LX, Ly, Lx, dy, dx, read_filenames)
Bases: object

close()
Closes the file.

Return type None

iter_frames(batch_size=1, dtype=<class 'numpy.float32'>)
Iterates through each set of frames, depending on batch_size, yielding both the frame index and frame data.

Parameters
• batch_size (int) – The number of frames to get at a time
• dtype (np.dtype) – The numpy data type that the data should return as

Yields
• indices (array int) – The frame indices.
• data (batch_size x Ly x Lx) – The frames

property n_frames: int
total number of frames in the read_file.

Return type int

property nbytes
total number of bytes in the read_file.

property nbytesread
number of bytes per frame (FIXED for given file)

read(batch_size=1, dtype=<class 'numpy.float32'>)
Returns the next frame(s) in the file and its associated indices.

Parameters
• batch_size (int) – The number of frames to read at once.
• frames (batch_size x Ly x Lx) – The frame data

Return type Optional[Tuple[ndarray, ndarray]]

suite2p.io.binary.binned_mean(mov, bin_size)
Returns an array with the mean of each time bin (of size ‘bin_size’).

Return type ndarray

suite2p.io.binary.from_slice(s)
Creates an np.arange() array from a Python slice object. Helps provide numpy-like slicing interfaces.
Return type: Optional[ndarray]

suite2p.io.binary.temporary_pointer(file)
context manager that resets file pointer location to its original place upon exit.

### 13.3 suite2p.io.h5 module

suite2p.io.h5.h5py_to_binary(ops)
finds h5 files and writes them to binaries

**Parameters**
- **ops** (dictionary) – ‘nplanes’, ‘h5_path’, ‘h5_key’, ‘save_path’, ‘save_folder’,

**Returns**
- **ops** – ‘Ly’, ‘Lx’, ops[‘reg_file’] or ops[‘raw_file’] is created binary

**Return type**
dictionary of first plane

### 13.4 suite2p.io.nwb module

suite2p.io.nwb.nwb_to_binary(ops)
convert nwb file to binary (experimental)

**Parameters**
- **ops** (dictionary) – requires ‘nwb_file’ key optional keys ‘nwb_driver’, ‘nwb_series’
  ‘look_one_level_down’

**Returns**
- **ops** – ops[‘reg_file’] or ops[‘raw_file’] is created binary assigns keys ‘Ly’, ‘Lx’,
  ‘tiffreader’, ‘first_tiffs’, ‘frames_per_folder’, ‘nframes’, ‘meanImg’, ‘meanImg_chan2’

**Return type**
dictionary of first plane

suite2p.io.nwb.read_nwb(fpath)
read NWB file for use in the GUI

suite2p.io.nwb.save_nwb(save_folder)
convert folder with plane folders to NWB format

### 13.5 suite2p.io.save module

suite2p.io.save.combined(save_folder, save=True)
Combines all the folders in save_folder into a single result file.

can turn off saving (for gui loading)

Multi-plane recordings are arranged to best tile a square. Multi-roi recordings are arranged by their dx,dy physical
localization. Multi-plane / multi-roi recordings are tiled after using dx,dy.

suite2p.io.save.compute_dydx(ops)
suite2p.io.save.save_mat(ops, stat, F, Fneu, spks, iscell, redcell)
13.6 suite2p.io.sbx module

suite2p.io.sbx.sbx_to_binary(ops, ndeadcols=-1, ndeadrows=0)
finds scanbox files and writes them to binaries


Returns ops – ‘Ly’, ‘Lx’, ops[‘reg_file’] or ops[‘raw_file’] is created binary

Return type dictionary of first plane

13.7 suite2p.io.server module

suite2p.io.server.send_jobs(save_folder, host=None, username=None, password=None, server_root=None, local_root=None, n_cores=8)
send each plane to compute on server separately

add your own host, username, password and path on server for where to save the data

suite2p.io.server.ssh_connect(host, username, password, verbose=True)
from paramiko example

suite2p.io.server.unix_path(path)

13.8 suite2p.io.tiff module

suite2p.io.tiff.generate_tiff_filename(functional_chan, align_by_chan, save_path, k, ichan)
Calculates a suite2p tiff filename from different parameters.

Parameters

- functional_chan (int) – The channel number with functional information
- align_by_chan (int) – Which channel to use for alignment
- save_path (str) – The directory to save to
- k (int) – The file number
- wchan (int) – The channel number.

Returns filename

Return type str

suite2p.io.tiff.mesoscan_to_binary(ops)
finds mesoscope tiff files and writes them to binaries


Return type dictionary of first plane
suite2p.io.tiff.ome_to_binary(ops)
converts ome.tif to *.bin file for non-interleaved red channel recordings assumes SINGLE-PAGE tiffs where first channel has string ‘Ch1’ and also SINGLE FOLDER

Parameters ops (dictionary) – keys nplanes, nchannels, data_path, look_one_level_down, reg_file

Returns ops – creates binaries ops[‘reg_file’] assigns keys: tiffreader, first_tiffs, frames_per_folder, nframes, meanImg, meanImg_chan2

Return type dictionary of first plane

suite2p.io.tiff.open_tiff(file, sktiff)
Returns image and its length from tiff file with either ScanImageTiffReader or tifffile, based on ‘sktiff’

Return type Tuple[Union[TiffFile, ScanImageTiffReader], int]

suite2p.io.tiff.save_tiff(mov, fname)
Save image stack array to tiff file.

Parameters
• mov (nImg x Ly x Lx) – The frames to save
• fname (str) – The tiff filename to save to

Return type None

suite2p.io.tiff.tiff_to_binary(ops)
finds tiff files and writes them to binaries


Return type dictionary of first plane

suite2p.io.tiff.use_sktiff_reader(tiff_filename, batch_size=None)
Returns False if ScanImageTiffReader works on the tiff file, else True (in which case use tifffile).

Return type bool

13.9 suite2p.io.utils module

suite2p.io.utils.find_files_open_binaries(ops1, ish5=False)
finds tiffs or h5 files and opens binaries for writing


Returns ops1 – adds fields ‘filelist’, ‘first_tiffs’, opens binaries

Return type list of dictionaries

suite2p.io.utils.get_h5_list(ops)
make list of h5 files to process if ops[‘look_one_level_down’], then all h5’s in all folders + one level down

suite2p.io.utils.get_sbx_list(ops)
make list of scanbox files to process if ops[‘subfolders’], then all tiffs ops[‘data_path’][0] / ops[‘subfolders’] / *.sbx if ops[‘look_one_level_down’], then all tiffs in all folders + one level down TODO: Implement “tiff_list” functionality
suite2p.io.utils.get_suite2p_path(path)
    Find the root suite2p folder in the path variable

    Return type  Path

suite2p.io.utils.get_tif_list(ops)
    make list of tiffs to process if ops['subfolders'], then all tiffs ops['data_path'][0] / ops['subfolders'] / *
    .tif if ops['look_one_level_down'], then all tiffs in all folders + one level down if ops['tiff_list'], then
    ops['data_path'][0] / ops['tiff_list'] ONLY

suite2p.io.utils.init_ops(ops)
    initializes ops files for each plane in recording

    Parameters ops (dictionary) – 'nplanes', 'save_path', 'save_folder', 'fast_disk', 'nchannels',
    'keep_movie_raw' + (if mesoscope) 'dy', 'dx', 'lines'

    Returns ops1 – adds fields 'save_path0', 'reg_file' (depending on ops: 'raw_file', 'reg_file_chan2',
    'raw_file_chan2')

    Return type  list of dictionaries

suite2p.io.utils.list_files(froot, look_one_level_down, exts)
    get list of files with exts in folder froot + one level down maybe

suite2p.io.utils.list_h5(ops)

suite2p.io.utils.search_for_ext(rootdir, extension='tif', look_one_level_down=False)

13.10 Module contents
SUITE2P. REGISTRATION PACKAGE

14.1 Submodules

14.2 suite2p.registration.bidiphase module

suite2p.registration.bidiphase.compute(frames)
   Returns the bidirectional phase offset, the offset between lines that sometimes occurs in line scanning.
   
   Parameters frames (frames x Ly x Lx) – random subsample of frames in binary (frames x Ly x Lx)

   Returns bidiphase – bidirectional phase offset in pixels

   Return type int

suite2p.registration.bidiphase.shift(frames, bidiphase)
   Shift last axis of ‘frames’ by bidirectional phase offset in-place, bidiphase.

   Parameters
   
   • frames (frames x Ly x Lx) –
   
   • bidiphase (int) – bidirectional phase offset in pixels

   Return type None

14.3 suite2p.registration.metrics module

suite2p.registration.metrics.bin_median(mov, window=10)
suite2p.registration.metrics.corr_to_template(mov, tmpl)
suite2p.registration.metrics.filt_parallel(data, filt, num_cores)
suite2p.registration.metrics.filt_worker(inputs)
suite2p.registration.metrics.get_flow_metrics(ops)
   get farneback optical flow and some other stats from normcorre paper
suite2p.registration.metrics.get_pc_metrics(ops, use_red=False)
   Computes registration metrics using top PCs of registered movie

   movie saved as binary file ops[‘reg_file’] metrics saved to ops[‘regPC’] and ops[‘X’] ‘regDX’ is nPC x 3 where X[:,0] is rigid, X[:,1] is average nonrigid, X[:,2] is max nonrigid shifts ‘regPC’ is average of top and bottom frames for each PC ‘tPC’ is PC across time frames
Parameters

- **use_red (bool, optional)** – default False, whether to use ‘reg_file’ or ‘reg_file_chan2’

Returns **ops** – The same as the ops input, but will now include ‘regPC’, ‘tPC’, and ‘regDX’.

Return type **dict**

suite2p.registration.metrics.local_corr(mov, batch_size, num_cores)

computes correlation image on mov (nframes x pixels x pixels)

suite2p.registration.metricsoptic_flow(mov, tmpl, nflows)

optic flow computation using farneback

suite2p.registration.metrics pc_register(pclow, pchigh, bidi_corrected, spatial_hp=None, pre_smooth=None, smooth_sigma=1.15, smooth_sigma_time=0, block_size=(128, 128), maxregshift=0.1, maxregshiftNR=10, reg_1p=False, snr_thresh=1.25, is_nonrigid=True, bidiphas offset=0, spatial_taper=50.0)

register top and bottom of PCs to each other

Parameters

- **pclow (float, array)** – average of bottom of spatial PC: nPC x Ly x Lx
- **pchigh (float, array)** – average of top of spatial PC: nPC x Ly x Lx
- **bidi_corrected (bool)** – whether to do bidi correction.
- **spatial_hp (int)** – high-pass filter window size for the spatial dimensions
- **pre_smooth (int)** – low-pass filter window size for the spatial dimensions
- **smooth_sigma (int)** – see registration settings
- **smooth_sigma_time (int)** – see registration settings
- **block_size (int, int)** – see registration settings
- **maxregshift (float)** – see registration settings
- **maxregshiftNR (int)** – see registration settings
- **reg_1p (bool)** – see 1Preg settings
- **snr_thresh (float)** – signal to noise threshold to use.
- **is_nonrigid (bool)** –
- **bidiphas offset (int)** –
- **spatial_taper (float)** –

Returns **X** – nPC x 3 where X[:,0] is rigid, X[:,1] is average nonrigid, X[:,2] is max nonrigid shifts

Return type **float array**

suite2p.registration.metrics pclowhigh(mov, nlowhigh, nPC, random_state)

Compute mean of top and bottom PC weights for nPC’s of mov

computes nPC PCs of mov and returns average of top and bottom

Parameters
• mov (frames x Ly x Lx) – subsampled frames from movie
• nlowhigh (int) – number of frames to average at top and bottom of each PC
• nPC (int) – number of PCs to compute
• random_state – a value that sets the seed for the PCA randomizer.

Returns
• pclow (float, array) – average of bottom of spatial PC: nPC x Ly x Lx
• pchigh (float, array) – average of top of spatial PC: nPC x Ly x Lx
• w (float, array) – singular values of decomposition of mov
• v (float, array) – frames x nPC, how the PCs vary across frames

14.4 suite2p.registration.nonrigid module

suite2p.registration.nonrigid.block_interp(ymax1, xmax1, mshy, mshx, yup, xup)
interpolate from ymax1 to mshy to create coordinate transforms

Parameters
• ymax1 –
• xmax1 –
• mshy (Ly x Lx) – meshgrid in y
• mshx (Ly x Lx) – meshgrid in x
• yup (nimg x Ly x Lx) – y shifts for each coordinate
• xup (nimg x Ly x Lx) – x shifts for each coordinate

suite2p.registration.nonrigid.calculate_nblocks(L, block_size=128)
Returns block_size and nblocks from dimension length and desired block size

Parameters
• L (int) –
• block_size (int) –

Return type Tuple[int, int]

Returns
• block_size (int)
• nblocks (int)

suite2p.registration.nonrigid.getSNR(cc, lcorr, lpad)
Compute SNR of phase-correlation.

Parameters
• cc (nimg x Ly x Lx) – The frame data to analyze
• lcorr (int) –
• lpad (int) – border padding width

Returns snr
suite2p, Release 0.7.2

Return type float

```
suite2p.registration.nonrigid.make_blocks(Ly, Lx, block_size=(128, 128))
```
Computes overlapping blocks to split FOV into to register separately

**Parameters**
- $Ly$ (int) – Number of pixels in the vertical dimension
- $Lx$ (int) – Number of pixels in the horizontal dimension
- $block\_size$ (int, int) – block size

**Returns**
- $yblock$ (float array)
- $xblock$ (float array)
- $nblocks$ (int, int)
- $block\_size$ (int, int)
- $NRsm$ (array)

```
suite2p.registration.nonrigid.map_coordinates(I, yc, xc, Y)
```
In-place bilinear transform of image ‘I’ with ycoordinates $yc$ and xcoordinates $xc$ to $Y$

**Parameters**
- $I$ (Ly x Lx) –
- $yc$ (Ly x Lx) – new y coordinates
- $xc$ (Ly x Lx) – new x coordinates
- $Y$ (Ly x Lx) – shifted I

**Return type** None

```
suite2p.registration.nonrigid.phasecorr(data, maskMul, maskOffset, cfRefImg, snr_thresh, NRsm, xblock, yblock, maxregshiftNR, subpixel=10, lpad=3)
```
Compute phase correlations for each block

**Parameters**
- $data$ (nimg x Ly x Lx) –
- $maskMul$ (ndarray) – gaussian filter
- $maskOffset$ (ndarray) – mask offset
- $cfRefImg$ – FFT of reference image
- $snr\_thresh$ (float) – signal to noise ratio threshold
- $NRsm$ –
- $xblock$ (float array) –
- $yblock$ (float array) –
- $maxregshiftNR$ (int) –
- $subpixel$ (int) –
- $lpad$ (int) – upsample from a square +/- lpad

**Returns**
• $y_{max1}$
• $x_{max1}$
• $c_{max1}$

`suite2p.registration.nonrigid.phasecorr_reference(refImg0, maskSlope, smooth_sigma, yblock, xblock)`

Computes taper and fft'ed reference image for phasecorr.

**Parameters**

- `refImg0` (array) –
- `maskSlope` –
- `smooth_sigma` –
- `yblock` (float array) –
- `xblock` (float array) –

**Returns**

- `maskMul`
- `maskOffset`
- `cRefImg`

`suite2p.registration.nonrigid.shift_coordinates(data, yup, xup, mshy, mshx, Y)`

Shift data into yup and xup coordinates

**Parameters**

- `data` (nimg x Ly x Lx) –
- `yup` (nimg x Ly x Lx) – y shifts for each coordinate
- `xup` (nimg x Ly x Lx) – x shifts for each coordinate
- `mshy` (Ly x Lx) – meshgrid in y
- `mshx` (Ly x Lx) – meshgrid in x
- `Y` (nimg x Ly x Lx) – shifted data

`suite2p.registration.nonrigid.transform_data(data, nbblocks, xblock, yblock, ymax1, xmax1, bilinear=True)`

Piecewise affine transformation of data using block shifts ymax1, xmax1

**Parameters**

- `data` (nimg x Ly x Lx) –
- `nbblocks` ((int, int)) –
- `xblock` (float array) –
- `yblock` (float array) –
- `ymax1` (nimg x nbblocks) – y shifts of blocks
- `xmax1` (nimg x nbblocks) – y shifts of blocks
- `bilinear` (bool (optional, default=True)) – do bilinear interpolation, if False do nearest neighbor

**Returns** Y – shifted data
Return type float32, nimg x Ly x Lx

suite2p.registration.nonrigid.upsample_block_shifts(Lx, Ly, nbblocks, xblock, yblock, ymax1, xmax1)
upsample blocks of shifts into full pixel-wise maps for shifting
this function upsamples ymax1, xmax1 so that they are nimg x Ly x Lx for later bilinear interpolation

Parameters
• Lx (int) – number of pixels in the horizontal dimension
• Ly (int) – number of pixels in the vertical dimension
• nbblocks ((int, int)) –
• xblock (float array) –
• yblock (float array) –
• ymax1 (nimg x nbblocks) – y shifts of blocks
• xmax1 (nimg x nbblocks) – y shifts of blocks

Returns
• yup (nimg x Ly x Lx) – y shifts for each coordinate
• xup (nimg x Ly x Lx) – x shifts for each coordinate

14.5 suite2p.registration.register module

suite2p.registration.register.compute_crop(xoff, yoff, corrXY, th_badframes, badframes, maxregshift, Ly, Lx)
determines how much to crop FOV based on motion
determines badframes which are frames with large outlier shifts (threshold of outlier is th_badframes) and it
excludes these badframes when computing valid ranges from registration in y and x

Parameters
• xoff (int) –
• yoff (int) –
• corrXY –
• th_badframes –
• badframes –
• maxregshift –
• Ly (int) – Height of a frame
• Lx (int) – Width of a frame

Returns
• badframes
• yrange
• xrange
suite2p.registration.register.compute_reference(ops, frames)
computes the reference image
picks initial reference then iteratively aligns frames to create reference

Parameters
- **ops** (*dictionary*) – need registration options
- **frames** (*3D array, int16*) – size [nimg_init x Ly x Lx], frames to use to create initial reference

Returns **refImg** – size [Ly x Lx], initial reference image
Return type 2D array, int16

suite2p.registration.register.compute_reference_masks(refImg, ops=None)

suite2p.registration.register.enhanced_mean_image(ops)
computes enhanced mean image and adds it to ops
Median filters ops['meanImg'] with 4*diameter in 2D and subtracts and divides by this median-filtered image to return a high-pass filtered image ops['meanImgE']

Parameters **ops** (*dictionary*) – uses ‘meanImg’, ‘aspect’, ‘spatscale_pix’, ‘yrange’ and ‘xrange’

Returns **ops** – ‘meanImgE’ field added
Return type dictionary

suite2p.registration.register.pick_initial_reference(frames)
computes the initial reference image
the seed frame is the frame with the largest correlations with other frames; the average of the seed frame with its top 20 correlated pairs is the initial reference frame returned

Parameters **frames** (*3D array, int16*) – size [frames x Ly x Lx], frames from binary

Returns **refImg** – size [Ly x Lx], initial reference image
Return type 2D array, int16

suite2p.registration.register.register_binary(ops, refImg=None, raw=True)
main registration function
if ops is a list of dictionaries, each will be registered separately

Parameters
- **refImg** (*2D array (optional, default None)*) –
- **raw** (*bool (optional, default True)*) – use raw_file for registration if available, if False forces reg_file to be used

Return type dictionary

suite2p.registration.register.register_frames(refAndMasks, frames, ops=None)
register frames to reference image

Parameters

14.5. suite2p.registration.register module
• **refImg** (2D array (optional, default None)) –
• **raw** (bool (optional, default True)) – use raw_file for registration if available, if False forces reg_file to be used


Return type **dictionary**

```python
suite2p.registration.register.shift_frames(frames, yoff, xoff, yoff1, xoff1, ops=None)
```

### 14.6 suite2p.registration.rigid module

**suite2p.registration.rigid.apply_masks(data, maskMul, maskOffset)**

Returns a 3D image ‘data’, multiplied by ‘maskMul’ and then added ‘maskOffset’.

Parameters

- **data** (nImg x Ly x Lx) –
- **maskMul** (ndarray) –
- **maskOffset** (ndarray) –

Returns **maskedData**

Return type **nImg x Ly x Lx**

```python
suite2p.registration.rigid.compute_masks(refImg, maskSlope)
```

Returns maskMul and maskOffset from an image and slope parameter

Parameters

- **refImg** (Ly x Lx) – The image
- **maskSlope** –

Return type **Tuple[ndarray, ndarray]**

Returns

- **maskMul** (float array)
- **maskOffset** (float array)

```python
suite2p.registration.rigid.phasecorr(data, cfRefImg, maxregshift, smooth_sigma_time)
```

compute phase correlation between data and reference image

Parameters

- **data** (int16) – array that’s frames x Ly x Lx
- **maxregshift** (float) – maximum shift as a fraction of the minimum dimension of data (min(Ly,Lx) * maxregshift)
- **smooth_sigma_time** (float) – how many frames to smooth in time

Return type **Tuple[int, int, float]**

Returns

- **ymax** (int) – shifts in y from cfRefImg to data for each frame
- **xmax** (int) – shifts in x from cfRefImg to data for each frame
- **cmax** (float) – maximum of phase correlation for each frame
suite2p.registration.rigid.phasecorr_reference(refImg, smooth_sigma=None)
Returns reference image fft’ed and complex conjugate and multiplied by gaussian filter in the fft domain, with
standard deviation ‘smooth_sigma’ computes fft’ed reference image for phasecorr.

Parameters
refImg (2D array, int16) – reference image
Returns cfRefImg
Return type 2D array, complex64

suite2p.registration.rigid.shift_frame(frame, dy, dx)
Returns frame, shifted by dy and dx

Parameters
- frame (Ly x Lx) –
- dy (int) – vertical shift amount
- dx (int) – horizontal shift amount
Returns frame_shifted – The shifted frame
Return type Ly x Lx

14.7 suite2p.registration.utils module

suite2p.registration.utils.combine_offsets_across_batches(offset_list, rigid)
suite2p.registration.utils.complex_fft2(img, pad_fft=False)
Returns the complex conjugate of the fft-transformed 2D array ‘img’, optionally padded for speed.

Parameters
- img (Ly x Lx) – The image to process
- pad_fft (bool) – Whether to pad the image
Return type ndarray

suite2p.registration.utils.convolve(mov, img)
Returns the 3D array ‘mov’ convolved by a 2D array ‘img’.

Parameters
- mov (nImg x Ly x Lx) – The frames to process
- img (2D array) – The convolution kernel
Returns convolved_data
Return type nImg x Ly x Lx

suite2p.registration.utils.fft2(data, size=None)
compute fft2 over last two dimensions using pytorch size (padding) is not used
suite2p.registration.utils.gaussian_fft(sig, Ly, Lx)
gaussian filter in the fft domain with std sig and size Ly,Lx

Parameters
- sig –
- Ly (int) – frame height
- Lx (int) – frame width
suite2p, Release 0.7.2

**Returns** `fhg` – smoothing filter in Fourier domain

**Return type** `np.ndarray`

suite2p.registration.utils.*ifft2*(data, size=None)
compute ifft2 over last two dimensions using pytorch size (padding) is not used

suite2p.registration.utils.*kernelD*(xs, ys, sigL=0.85)
Gaussian kernel from xs (1D array) to ys (1D array), with the ‘sigL’ smoothing width for up-sampling kernels, (best between 0.5 and 1.0)

**Parameters**
- `xs` (ndarray) –
- `ys` (ndarray) –
- `sigL` (float) –

**Return type** `ndarray`

suite2p.registration.utils.*kernelD2*(xs, ys)

**Parameters**
- `xs` (int) –
- `ys` (int) –

**Return type** `ndarray`

suite2p.registration.utils.*mat_upsample*(lpad, subpixel=10)
upsampling matrix using gaussian kernels

**Parameters**
- `lpad` (int) –
- `subpixel` (int) –

**Returns**
- `Kmat` (np.ndarray)
- `nup` (int)

suite2p.registration.utils.*meshgrid_mean_centered*(x, y)
Returns a mean-centered meshgrid

**Parameters**
- `x` (int) – The height of the meshgrid
- `y` (int) – The width of the meshgrid

**Return type** `Tuple[ndarray, ndarray]`

**Returns**
- `xx` (int array)
- `yy` (int array)

suite2p.registration.utils.*spatial_high_pass*(data, N)
high pass filters data over axis=1,2 with window N

**Parameters**
• data (Ly x Lx) – The image to smooth.
• N (int) – The window size

Returns smoothed_data – The smoothed frame

Return type Ly x Lx

suite2p.registration.utils.spatial_smooth(data, window)
Spatially smooth data using cumsum over axis=1,2 with window N

Parameters
• data (Ly x Lx) – The image to smooth.
• window (int) – The window size

Returns smoothed_data – The smoothed frame

Return type Ly x Lx

suite2p.registration.utils.spatial_taper(sig, Ly, Lx)
Returns spatial taper on edges with gaussian of std sig

Parameters
• sig –
• Ly (int) – frame height
• Lx (int) – frame width

Returns

Return type maskMul

suite2p.registration.utils.temporal_smooth(data, sigma)
Returns Gaussian filtered ‘frames’ ndarray over first dimension

Parameters
• data (nImg x Ly x Lx) –
• sigma (float) – windowing parameter

Returns smoothed_data – Smoothed data

Return type nImg x Ly x Lx

14.8 suite2p.registration.zalign module

suite2p.registration.zalign.compute_zpos(Zreg, ops)
compute z position of frames given z-stack Zreg

Parameters
• Zreg (3D array) – size [nplanes x Ly x Lx], z-stack
• ops (dictionary) – ‘reg_file’ <- binary to register to z-stack, ‘smooth_sigma’, ‘Ly’, ‘Lx’, ‘batch_size’

Returns
• ops_orig
• zcorr
suite2p.registration.zalign.register_stack(Z, ops)

Parameters

- Z
- ops (dict)

Returns

- Zreg (nplanes x Ly x Lx) – Z-stack
- ops (dict)

14.9 Module contents
SUITE2P.DETECTION PACKAGE

15.1 Submodules

15.2 suite2p.detection.anatomical module

15.3 suite2p.detection.chan2detect module

suite2p.detection.chan2detect.cellpose_overlap(stats, mimg2)
suite2p.detection.chan2detect.correct_bleedthrough(Ly, Lx, nblk, mimg, mimg2)
suite2p.detection.chan2detect.detect(ops, stats)
suite2p.detection.chan2detect.intensity_ratio(ops, stats)
    compute pixels in cell and in area around cell (including overlaps) (exclude pixels from other cells)
suite2p.detection.chan2detect.quadrant_mask(Ly, Lx, ny, nx, sT)

15.4 suite2p.detection.denoise module

suite2p.detection.denoise.pca_denoise(mov, block_size, n_comps_frac)

15.5 suite2p.detection.detect module

suite2p.detection.detect.detect(ops, classfile=None)
suite2p.detection.detect.select_rois(ops, mov, dy, dx, Ly, Lx, max_overlap=True, sparse_mode=True, do_crop=True, classfile=None)
15.6 suite2p.detection.metrics module

15.7 suite2p.detection.sourcery module

suite2p.detection.sourcery.circleMask(d0)
creates array with indices which are the radius of that x,y point

Parameters
\( d0 \) – (patch of \((-d0,d0+1)\) over which radius computed

Returns
• \( rs \) – array \((2*d0+1,2*d0+1)\) of radii
• \( dx \) – indices in \( rs \) where the radius is less than \( d0 \)
• \( dy \) – indices in \( rs \) where the radius is less than \( d0 \)

suite2p.detection.sourcery.connected_region(stat, ops)

suite2p.detection.sourcery.create_neuropil_basis(ops, Ly, Lx)
computes neuropil basis functions

Parameters
• \( ops \) – ratio_neuropil, tile_factor, diameter, neuropil_type
• \( Ly \) (int) –
• \( Lx \) (int) –

Returns basis functions (pixels x nbasis functions)

Return type S

suite2p.detection.sourcery.drawClusters(stat, ops)

suite2p.detection.sourcery.extendROI(ypix, xpix, Ly, Lx, niter=1)

suite2p.detection.sourcery.getSVData(mov, ops)

suite2p.detection.sourcery.getSVDproj(mov, ops, u)

suite2p.detection.sourcery.getStU(ops, U)

suite2p.detection.sourcery.getVmap(Ucell, sig)

suite2p.detection.sourcery.get_connected(Ly, Lx, stat)
grow i0 until it cannot grow any more

suite2p.detection.sourcery.get_stat(ops, stats, Ucell, codes, frac=0.5)
computes statistics of cells found using sourcery

Parameters
• \( Ly \) –
• \( Lx \) –
• \( d0 \) –
• \( mPix \) ((pixels,ncells)) –
• \( mLam \) ((weights,ncells)) –
• \( codes \) ((ncells,nsvd)) –
• \( Ucell \) ((nsvd,Ly,Lx)) –
Returns assigned to stat: ipix, ypix, xpix, med, npix, lam, footprint, compact, aspect_ratio, ellipse

Return type stat

suite2p.detection.sourcery.iter_extend(ypix, xpix, Ucell, code, refine=-1, change_codes=False)

suite2p.detection.sourcery.localMax(V, footprint, thres)
find local maxima of V (correlation map) using a filter with (usually circular) footprint

Parameters
• V –
• footprint –
• thres –

Returns i,j

Return type indices of local max greater than thres

suite2p.detection.sourcery.localRegion(i, j, dy, dx, Ly, Lx)
returns valid indices of local region surrounding (i, j) of size (dy.size, dx.size)

suite2p.detection.sourcery.minDistance(inputs)

suite2p.detection.sourcery.morphOpen(V, footprint)
computes the morphological opening of V (correlation map) with circular footprint

suite2p.detection.sourcery.pairwiseDistance(y, x)

suite2p.detection.sourcery.postprocess(ops, stat, Ucell, codes)

suite2p.detection.sourcery.r_squared(xp, yp, xpix, ypix, diam_y, diam_x, estimator=<function median>)

suite2p.detection.sourcery.sourcery(mov, ops)

suite2p.detection.sourcery.sub2ind(array_shape, rows, cols)

15.8 suite2p.detection.sparsedetect module

class suite2p.detection.sparsedetect.EstimateMode(value)
    Bases: enum.Enum

    An enumeration.

    Estimated = 'estimated'

    Forced = 'FORCED'

suite2p.detection.sparsedetect.add_square(yi, xi, lx, Ly, Lx)
return square of pixels around peak with norm 1

Parameters
• yi (int) – y-center
• xi (int) – x-center
• lx (int) – x-width
• Ly (int) – full y frame
• Lx (int) – full x frame

Returns
• \( y_0 \) (array) – pixels in y
• \( x_0 \) (array) – pixels in x
• \( \text{mask} \) (array) – pixel weightings

suite2p.detection.sparsedetect.\texttt{estimate\_spatial\_scale}(I)

\textbf{Return type} \ int

suite2p.detection.sparsedetect.\texttt{extendROI}(ypix, xpix, Ly, Lx, niter=1)
extend ypix and xpix by niter pixel(s) on each side

suite2p.detection.sparsedetect.\texttt{extend\_mask}(ypix, xpix, lam, Ly, Lx)
extend mask into 8 surrounding pixels

suite2p.detection.sparsedetect.\texttt{find\_best\_scale}(I, spatial\_scale)
Returns best scale and estimate method (if the spatial scale was forced (if positive) or estimated (the top peaks).

\textbf{Return type} \ Tuple[int, \texttt{EstimateMode}]

suite2p.detection.sparsedetect.\texttt{iter\_extend}(ypix, xpix, mov, Lyc, Lxc, active\_frames)
extend mask based on activity of pixels on active frames ACTIVE frames determined by threshold

\textbf{Parameters}
• \( \text{ypix} \) (array) – pixels in y
• \( \text{xpix} \) (array) – pixels in x
• \( \text{mov} \) (2D array) – binned residual movie [nbinned x Lyc*Lxc]
• \( \text{active\_frames} \) (1D array) – list of active frames

\textbf{Returns}
• \( \text{ypix} \) (array) – extended pixels in y
• \( \text{xpix} \) (array) – extended pixels in x
• \( \text{lam} \) (array) – pixel weighting

suite2p.detection.sparsedetect.\texttt{multiscale\_mask}(ypix0, xpix0, lam0, Lyp, Lxp)

suite2p.detection.sparsedetect.\texttt{neuropil\_subtraction}(mov, filter\_size)
Returns movie subtracted by a low-pass filtered version of itself to help ignore neuropil.

\textbf{Return type} \ None

suite2p.detection.sparsedetect.\texttt{sparsery}(mov, high\_pass, neuropil\_high\_pass, batch\_size, spatial\_scale, threshold\_scaling, max\_iterations, yrange, xrange, percentile=0)
Returns stats and ops from ‘mov’ using correlations in time.

\textbf{Return type} \ Tuple[Dict[str, Any], List[Dict[str, Any]]]

suite2p.detection.sparsedetect.\texttt{square\_convolution\_2d}(mov, filter\_size)
Returns movie convolved by uniform kernel with width ‘filter\_size’.

\textbf{Return type} \ ndarray

suite2p.detection.sparsedetect.\texttt{two\_comps}(mpix0, lam, Th2)
check if splitting ROI increases variance explained

\textbf{Parameters}
• \( \text{mpix0} \) (2D array) – binned movie for pixels in ROI [nbinned x npix]
• **lam** *(array)* – pixel weighting
• **Th2** *(float)* – intensity threshold

**Returns**
• **vrat** *(array)* – extended pixels in y
• **ipick** *(tuple)* – new ROI

### 15.9 suite2p.detection.stats module

```python
class suite2p.detection.stats.EllipseData(mu, cov, radii, ellipse, dy, dx)
Bases: tuple

  property area
  property aspect_ratio: float
      Return type float

  property cov
      Alias for field number 1

  property dx
      Alias for field number 5

  property dy
      Alias for field number 4

  property ellipse
      Alias for field number 3

  property mu
      Alias for field number 0

  property radii
      Alias for field number 2

  property radius: float
      Return type float

class suite2p.detection.stats.ROI(ypix, xpix, lam, med, do_crop, rsort=array([0.0, 1.0, 1.0, ...,
  42.42640687, 42.42640687, 42.42640687]))
Bases: object


  do_crop: bool

  classmethod filter_overlappers(ros, overlap_image, max_overlap)
      returns logical array of rois that remain after removing those that overlap more than fraction max_overlap from overlap_img.

      Return type List[bool]

  fit_ellipse(dy, dx)

      Return type EllipseData
```

### 15.9. suite2p.detection.stats module
classmethod from_stat_dict(stat)

Return type ROI

classmethod get_mean_r_squared_normed_all(rois, first_n=100)

Return type ndarray

classmethod get_n_pixels_normed_all(rois, first_n=100)

Return type ndarray

classmethod get_overlap_count_image(rois, Ly, Lx)

Return type ndarray

get_overlap_image(overlap_count_image)

Return type ndarray

lam: numpy.ndarray

property mean_r_squared: float
    Return type float

property mean_r_squared0: float
    Return type float

property mean_r_squared_compact: float
    Return type float

med: numpy.ndarray

property n_pixels: int
    Return type int

property npix_soma: int
    Return type int

ravel_indices(Ly, Lx)
    Returns a 1-dimensional array of indices from the ypix and xpix coordinates, assuming an image shape Ly x Lx.

    Return type ndarray

rsort: numpy.ndarray = array([0. , 1. , 1. , ..., 42.42640687, 42.42640687, 42.42640687])

property solidity: float
    Return type float

property soma_crop: numpy.ndarray
    Return type ndarray
class method stats_dicts_to_3d_array(stats, Ly, Lx, label_id=False)

Outputs a (roi x Ly x Lx) float array from a sequence of stat dicts. Convenience function that repeatedly calls ROI.from_stat_dict() and ROI.to_array() for all rois.

Parameters

- **stats** (List of dictionary 'ypix', 'xpix', 'lam') –
- **Ly** (y size of frame) –
- **Lx** (x size of frame) –
- **label_id** (whether array should be an integer value indicating ROI id or just 1 (indicating presence of ROI)) –

`to_array(Ly, Lx)`

Returns a 2D boolean array of shape (Ly x Lx) indicating where the roi is located.

Return type ndarray

suite2p.detection.stats.aspect_ratio(width, height, offset=0.01)

Return type float

suite2p.detection.stats.count_overlaps(Ly, Lx, ypixs, xpixs)

Return type ndarray

suite2p.detection.stats.distance_kernel(radius)

Returns 2D array containing geometric distance from center, with radius ‘radius’

Return type ndarray

suite2p.detection.stats.filter_overlappers(ypixs, xpixs, overlap_image, max_overlap)

returns ROI indices that remain after removing those that overlap more than fraction max_overlap from overlap_img.

Return type List[bool]

suite2p.detection.stats.fitMVGaus(y, x, lam0, dy, dx, thres=2.5, npts=100)

computes 2D gaussian fit to data and returns ellipse of radius thres standard deviations. :param y: pixel locations in y :type y: float, array :param x: pixel locations in x :type x: float, array :param lam0: weights of each pixel :type lam0: float, array

Return type EllipseData

suite2p.detection.stats.mean_r_squared(y, x, estimator=<function median>)

Return type float

suite2p.detection.stats.median_pix(ypix, xpix)

suite2p.detection.stats.norm_by_average(values, estimator=<function mean>, first_n=100, offset=0.0)

Returns array divided by the (average of the ‘first_n’ values + offset), calculating the average with ‘estimator’.

Return type ndarray
suite2p.detection.stats.roi_stats(stat, dy, dx, Ly, Lx, max_overlap=None, do_crop=True)

computes statistics of ROIs:

:param stat: 'ypix', 'xpix', 'lam'
:type stat: dictionary

:param diameters: (dy, dx)
:type diameters: dictionary

Return type dictionary

15.10 suite2p.detection.utils module

suite2p.detection.utils.downsample(mov, taper_edge=True)

Returns a pixel-downsampled movie from ‘mov’, tapering the edges of ‘taper_edge’ is True.

Parameters

- mov (nImg x Ly x Lx) – The frames to downsample
- taper_edge (bool) – Whether to taper the edges

Returns The downsampled frames
Return type filtered_mov

suite2p.detection.utils.hp_gaussian_filter(mov, width)

Returns a high-pass-filtered copy of the 3D array ‘mov’ using a gaussian kernel.

Parameters

- mov (nImg x Ly x Lx) – The frames to filter
- width (int) – The kernel width

Returns filtered_mov – The filtered video
Return type nImg x Ly x Lx

suite2p.detection.utils.hp_rolling_mean_filter(mov, width)

Returns a high-pass-filtered copy of the 3D array ‘mov’ using a non-overlapping rolling mean kernel over time.

Parameters

- mov (nImg x Ly x Lx) – The frames to filter
- width (int) – The filter width

Returns filtered_mov – The filtered frames
Return type nImg x Ly x Lx

suite2p.detection.utils.mask_iou(masks_true, masks_pred)

return best-matched masks

Parameters

- masks_true (ND-array (int)) – where 0=NO masks; 1,2… are mask labels
- masks_pred (ND-array (int)) – ND-array (int) where 0=NO masks; 1,2… are mask labels

Returns

- iou (float, ND-array) – array of IOU pairs
- preds (int, ND-array) – array of matched indices
- iou_all (float, ND-array) – full IOU matrix across all pairs
suite2p.detection.utils.mask_stats(mask)
    median and diameter of mask

suite2p.detection.utils.match_masks(iou)

suite2p.detection.utils.square_mask(mask, ly, yi, xi)
    crop from mask a square of size ly at position yi, xi

suite2p.detection.utils.standard_deviation_over_time(mov, batch_size)
    Returns standard deviation of difference between pixels across time, computed in batches of batch_size.
    Parameters
    • mov (nImg x Ly x Lx) – The frames to filter
    • batch_size (int) – The batch size
    Returns filtered_mov – The statistics for each pixel
    Return type Ly x Lx

suite2p.detection.utils.temporal_high_pass_filter(mov, width)
    Returns hp-filtered mov over time, selecting an algorithm for computational performance based on the kernel width.
    Parameters
    • mov (nImg x Ly x Lx) – The frames to filter
    • width (int) – The filter width
    Returns filtered_mov – The filtered frames
    Return type nImg x Ly x Lx

suite2p.detection.utils.threshold_reduce(mov, intensity_threshold)
    Returns standard deviation of pixels, thresholded by 'intensity_threshold'. Run in a loop to reduce memory footprint.
    Parameters
    • mov (nImg x Ly x Lx) – The frames to downsample
    • intensity_threshold (float) – The threshold to use
    Returns Vt – The standard deviation of the non-thresholded pixels
    Return type Ly x Lx

15.11 Module contents
16.1 Submodules

16.2 suite2p.extraction.dcnv module

suite2p.extraction.dcnv.oasis($F, batch_size, tau, fs$)
computes non-negative deconvolution
no sparsity constraints

Parameters
- $F$ (float, 2D array) – size [neurons x time], in pipeline uses neuropil-subtracted fluorescence
- $batch_size$ (int) – number of frames processed per batch
- $tau$ (float) – timescale of the sensor, used for the deconvolution kernel
- $fs$ (float) – sampling rate per plane

Returns $S$ – size [neurons x time], deconvolved fluorescence

Return type float, 2D array

suite2p.extraction.dcnv.oasis_matrix($F, v, w, t, l, s, tau, fs$)
spike deconvolution on many neurons parallelized with prange

suite2p.extraction.dcnv.oasis_trace($F, v, w, t, l, s, tau, fs$)
spike deconvolution on a single neuron

suite2p.extraction.dcnv.preprocess($F, baseline, win_baseline, sig_baseline, fs, prctile_baseline=8$)
preprocesses fluorescence traces for spike deconvolution
baseline-subtraction with window ‘win_baseline’

Parameters
- $F$ (float, 2D array) – size [neurons x time], in pipeline uses neuropil-subtracted fluorescence
- $baseline$ (str) – setting that describes how to compute the baseline of each trace
- $win_baseline$ (float) – window (in seconds) for max filter
- $sig_baseline$ (float) – width of Gaussian filter in seconds
- $fs$ (float) – sampling rate per plane
• **prctile_baseline** *(float)* – percentile of trace to use as baseline if using *constant_prctile* for baseline

**Returns**  
F – size [neurons x time], baseline-corrected fluorescence

**Return type**  
float, 2D array

### 16.3 suite2p.extraction.extract module

**suite2p.extraction.extract.create_masks_and_extract**(ops, stat, cell_masks=None, neuropil_masks=None)

creates masks, computes fluorescence, and saves stat, F, and Fneu to .npy

**Parameters**


- **stat** *(array of dicts)* –

**Returns**

- **ops** *(dictionary)*

- **stat** *(list of dictionaries)* – adds keys ‘skew’ and ‘std’

**suite2p.extraction.extract.enhanced_mean_image**(ops)

computes enhanced mean image and adds it to ops

Median filters ops[‘meanImg’] with 4*diameter in 2D and subtracts and divides by this median-filtered image to return a high-pass filtered image ops[‘meanImgE’]

**Parameters**

- **ops** *(dictionary)* – uses ‘meanImg’, ‘aspect’, ‘spatscale_pix’, ‘yrange’ and ‘xrange’

**Returns**

- **ops** – ‘meanImgE’ field added

**Return type**  
dictionary

**suite2p.extraction.extract.extract_traces**(ops, cell_masks, neuropil_masks, reg_file)

extracts activity from reg_file using masks in stat and neuropil_masks

computes fluorescence F as sum of pixels weighted by ‘lam’ computes neuropil fluorescence Fneu as sum of pixels in neuropil_masks

data is from reg_file ops[‘batch_size’] by pixels: .. code-block:: python

```python
F[n] = data[:, stat[n][‘ipix’]] @ stat[n][‘lam’]  
Fneu = neuropil_masks @ data.T
```

**Parameters**


- **cell_masks** *(list)* each is a tuple where first element are cell pixels (flattened), and second element are pixel weights normalized to sum 1 (lam)

- **neuropil_masks** *(list)* each element is neuropil pixels in (Ly*Lx) coordinates GOING TO BE DEPRECATED: size [ncells x npixels] where weights of each mask are elements

- **reg_file** *(io.BinaryFile object)* io.BinaryFile object that has iter_frames(batch_size=ops[‘batch_size’]) method

**Returns**

- **F** *(float, 2D array)* – size [ROIs x time]

- **Fneu** *(float, 2D array)* – size [ROIs x time]
suite2p.extraction.extract.extract_traces_from_masks(ops, cell_masks, neuropil_masks)
extract fluorescence from both channels
also used in drawroi.py
suite2p.extraction.extract.matmul_neuropil(Fi, data, neuropil_ipix, neuropil_npix)
suite2p.extraction.extract.matmul_traces(Fi, data, cell_ipix, cell_lam)

16.4 suite2p.extraction.masks module

suite2p.extraction.masks.create_cell_mask(stat, Ly, Lx, allow_overlap=False)
creates cell masks for ROIs in stat and computes radii

Parameters
- stat (dictionary ‘ypix’, ‘xpix’, ‘lam’) –
- Ly (y size of frame) –
- Lx (x size of frame) –
- allow_overlap (whether or not to include overlapping pixels in cell masks) –

Return type Tuple[ndarray, ndarray]
Returns
- cell_masks (len ncells, each has tuple of pixels belonging to each cell and weights)
- lam_normed

suite2p.extraction.masks.create_cell_pix(stats, Ly, Lx, lam_percentile=50.0)
Returns Ly x Lx array of whether pixel contains a cell (1) or not (0).
lam_percentile allows some pixels with low cell weights to be used, disable with lam_percentile=0.0

Return type ndarray

suite2p.extraction.masks.create_masks(ops, stats)
create cell and neuropil masks

suite2p.extraction.masks.create_neuropil_masks(ypixs, xpixs, cell_pix, inner_neuropil_radius, min_neuropil_pixels, circular=False)
creates surround neuropil masks for ROIs in stat by EXTENDING ROI (slower if circular)

Parameters cellpix (2D array) – 1 if ROI exists in pixel, 0 if not; pixels ignored for neuropil computation

Returns neuropil_masks – each element is array of pixels in mask in (Ly*Lx) coordinates

Return type list
16.5 Module contents
CHAPTER

SEVENTEEN

SUITE2P.CLASSIFICATION PACKAGE

17.1 Submodules

17.2 suite2p.classification.classifier module

class suite2p.classification.classifier.Classifier(classfile=None, keys=None)
    Bases: object
    ROI classifier model that uses logistic regression

    Parameters
    • classfile (string (optional, default None)) – path to saved classifier
    • keys (list of str (optional, default None)) – keys of ROI stat to use to classify

load(classfile, keys=None)
    data loader
    saved classifier contains stat with classification labels

    Parameters
    • classfile (string) – path to saved classifier
    • keys (list of str (optional, default None)) – keys of ROI stat to use to classify

predict_proba(stat)
    apply logistic regression model and predict probabilities
    model contains stat with classification labels

    Parameters
    • stat (list of dicts) – needs self.keys keys

run(stat, p_threshold=0.5)
    Returns cell classification thresholded with ‘p_threshold’ and its probability.

    Return type
    ndarray

save(filename)
    save classifier to filename

    Return type
    None
17.3 suite2p.classification.classify module

suite2p.classification.classify(classify(stat, classfile, keys=('npix_norm', 'compact', 'skew')))

Returns array of classifier output from classification process.

17.4 Module contents


18.1 Submodules

18.2 suite2p.gui.buttons module

class suite2p.gui.buttons.QuadButton(bid, Text, parent=None)
    Bases: PyQt5.QtWidgets.QPushButton
        custom QPushButton class for quadrant plotting requires buttons to put into a QButtonGroup (parent.quadbtns)
        allows only 1 button to pressed at a time
            press(parent, bid)

class suite2p.gui.buttons.SizeButton(bid, Text, parent=None)
    Bases: PyQt5.QtWidgets.QPushButton
        buttons to make trace box bigger or smaller
            press(parent)

class suite2p.gui.buttons.TopButton(bid, parent=None)
    Bases: PyQt5.QtWidgets.QPushButton
        selection of top neurons
            press(parent)
            top_selection(parent)
suite2p.gui.buttons.make_cellnotcell(parent)
        buttons for cell / not cell views at top
suite2p.gui.buttons.make_quadrants(parent)
        make quadrant buttons
suite2p.gui.buttons.make_selection(parent)
        buttons to draw a square on view
18.3 suite2p.gui.classgui module

class suite2p.gui.classgui.ListChooser(Text, parent=None)
    Bases: PyQt5.QtWidgets.QDialog
    apply_class(parent)
    build_classifier(parent)
    exit_list()
    load_cell()
    load_text()
    save_default(parent)
suite2p.gui.classgui.activate(parent, inactive)
suite2p.gui.classgui.add_to(parent)
suite2p.gui.classgui.class_activated(parent)
suite2p.gui.classgui.class_default(parent)
suite2p.gui.classgui.class_file(parent)
suite2p.gui.classgui.class_masks(parent)
suite2p.gui.classgui.disable(parent)
suite2p.gui.classgui.load(parent, name)
suite2p.gui.classgui.load_classifier(parent)
suite2p.gui.classgui.load_data(parent, keys, trainfiles)
suite2p.gui.classgui.load_default_classifier(parent)
suite2p.gui.classgui.load_list(parent)
suite2p.gui.classgui.load_s2p_classifier(parent)
suite2p.gui.classgui.make_buttons(parent, b0)
suite2p.gui.classgui.reset_default(parent)
suite2p.gui.classgui.save(parent, train_stats, train_iscell, keys)
suite2p gui.classgui.save_list(parent)
suite2p.gui.classgui.save_model(name, train_stats, train_iscell, keys)

18.4 suite2p.gui.drawroi module

class suite2p.gui.drawroi.ROIDraw(parent)
    Bases: PyQt5.QtWidgets.QMainWindow
    add_ROI(pos=None)
    check_proc(event)
    closeEvent(self, QCloseEvent)
    close_GUI()
create_masks_of_cells(mean_img)
keyPressEvent(self, QKeyEvent)
mouse_moved(pos)
normalize_img_add_masks()
plot_clicked(event)
plot_trace()
proc_ROI()

class suite2p.gui.drawroi.ViewButton(bid, Text, parent=None)
    Bases: PyQt5.QtWidgets.QPushButton
    press(parent, bid)

suite2p.gui.drawroi.masks_and_traces(ops, stat_manual, stat_orig)
    main extraction function inputs: ops and stat creates cell and neuropil masks and extracts traces returns: F (ROIs x time), Fneu (ROIs x time), F_chan2, Fneu_chan2, ops, stat F_chan2 and Fneu_chan2 will be empty if no second channel

class suite2p.gui.drawroi.sROI(iROI, parent=None, pos=None, diameter=None, color=None, yrange=None, xrange=None)
    Bases: object
    draw(parent, imy, imx, dy, dx)
    position(parent)
    remove(parent)

18.5 suite2p.gui.graphics module

suite2p.gui.graphics.ROI_index(ops, stat)
    matrix Ly x Lx where each pixel is an ROI index (-1 if no ROI present)

class suite2p.gui.graphics.TraceBox(parent=None, border=None, lockAspect=False, enableMouse=True, invertY=False, enableMenu=True, name=None, invertX=False)
    Bases: pyqtgraph.graphicsItems.PlotItem.PlotItem.PlotItem
    mouseDoubleClickEvent(self, QGraphicsSceneMouseEvent)
    zoom_plot()

class suite2p.gui.graphics.ViewBox(parent=None, border=None, lockAspect=False, enableMouse=True, invertY=False, enableMenu=True, name=None, invertX=False)
    Bases: pyqtgraph.graphicsItems.ViewBox.ViewBox.ViewBox
    mouseClickedEvent(ev)
    mouseDoubleClickEvent(self, QGraphicsSceneMouseEvent)
    mouseDragEvent(ev, axis=None)
    zoom_plot()

suite2p.gui.graphics.init_range(parent)
18.6 suite2p.gui.gui2p module

class suite2p.gui.gui2p.MainWindow(statfile=None)
    Bases: PyQt5.QtWidgets.QMainWindow
    ROI_position()
    ROI_remove()
    ROI_selection()
    ROIs_on(state)
    dragEnterEvent(self, QDragEnterEvent)
    dropEvent(self, QDropEvent)
    ichosen_stats()
    keyPressEvent(self, QKeyEvent)
    make_buttons()
    make_graphics(b0)
    mode_change(i)
        changes the activity mode that is used when multiple neurons are selected or in visualization windows like rastermap or for correlation computation!
        activityMode = 0 : F 1 : Fneu 2 : F - 0.7 * Fneu (default) 3 : spks
        uses binning set by self.bin
    number_chosen()
    plot_clicked(event)
        left-click chooses a cell, right-click flips cell to other view
    roi_text(state)
    select_cells(ypix, xpix)
    top_number_chosen()
    update_plot()
    zoom_cell(state)
    zoom_to_cell()
    suite2p.gui.gui2p.run(statfile=None)

18.7 suite2p.gui.io module

suite2p.gui.io.enable_views_and_classifier(parent)
suite2p.gui.io.export_fig(parent)
suite2p.gui.io.load_NWB(parent)
suite2p.gui.io.load_again(parent, Text)
suite2p.gui.io.load_behavior(parent)
suite2p.gui.io.load_custom_mask(parent)
suite2p.gui.io.

**load_dialog**(parent)

**load_dialog_NWB**(parent)

**load_dialog_folder**(parent)

give stat.npy path and load all needed files for suite2p

**load_folder**(parent)

**load_proc**(parent)

**load_to_GUI**(parent, basename, procs)

**make_masks_and_enable_buttons**(parent)

**load_files**(name)

**load_folder**(parent)

**load_proc**(parent)

**load_to_GUI**(parent, basename, procs)

**make_masks_and_enable_buttons**(parent)

18.8 suite2p.gui.masks module

class 

**suite2p.gui.masks.ColorButton**(bid, Text, parent=None)

Bases: PyQt5.QtWidgets.QPushButton

**press**(parent, bid)

**suite2p.gui.masks.add_roi**(parent, n, i)

add roi n to view i

**suite2p.gui.masks.beh_masks**(parent)

**suite2p.gui.masks.chan2_masks**(parent)

**suite2p.gui.masks.chan2_prob**(parent)

**suite2p.gui.masks.cmap_change**(parent)

**suite2p.gui.masks.corr_masks**(parent)

**suite2p.gui.masks.custom_masks**(parent)

**suite2p.gui.masks.draw_colorbar**(colormap='hsv')

**suite2p.gui.masks.draw_masks**(parent)

creates RGB masks using stat and puts them in M0 or M1 depending on whether or not iscell is True for a given ROI:

- **param ops:** mean_image, Vcorr
- **param stat:** xpix, ypix
- **param iscell:** vector with True if ROI is cell

**outputs:** M0: ROIs that are True in iscell M1: ROIs that are False in iscell

**suite2p.gui.masks.flip_for_class**(parent, iscell)

**suite2p.gui.masks.flip_plot**(parent)

**suite2p.gui.masks.flip_roi**(parent)

flips roi to other plot there are 3 levels of overlap so this may be buggy if more than 3 cells are on top of each other

**suite2p.gui.masks.hsv2rgb**(cols)
suite2p, Release 0.7.2

suite2p.gui.masks.init_masks(parent)
creates RGB masks using stat and puts them in M0 or M1 depending on whether or not iscell is True for a given ROI.
:param ops: mean_image, Vcorr
:param stat: xpix, ypix, xext, yext
:param iscell: vector with True if ROI is cell
:param ops_plot: plotROI, view, color, randcols

outputs: M0: ROIs that are True in iscell M1: ROIs that are False in iscell

suite2p.gui.masks.istat_hsv(istat)
suite2p.gui.masks.istat_transform(istat, colormap='hsv')
suite2p.gui.masks.make_buttons(parent, b0)
color buttons at row b0
suite2p.gui.masks.make_chosen_ROI(M0, ypix, xpix, v)
suite2p.gui.masks.make_chosen_circle(M0, ycirc, xcirc, col, sat)
suite2p.gui.masks.make_colorbar(parent, b0)
suite2p.gui.masks.make_colors(parent)
suite2p.gui.masks.plot_colorbar(parent)
suite2p.gui.masks.plot_masks(parent, M)
suite2p gui.masks.rastermap_masks(parent)
suite2p.gui.masks.redraw_masks(parent, ypix, xpix)
redraw masks after roi added/removed
suite2p gui.masks.remove_roi(parent, n, i0)
removes roi n from view i0
suite2p gui.masks.rgb_masks(parent, col, c)

18.9 suite2p.gui.menus module

suite2p gui.menus.classifier(parent)
suite2p gui.menus.mainmenu(parent)
suite2p gui.menus.manual_label(parent)
suite2p gui.menus.mergebar(parent)
suite2p gui.menus.plugins(parent)
suite2p gui.menus.regPC_window(parent)
suite2p gui.menus.reg_window(parent)
suite2p gui.menus.registration(parent)
suite2p gui.menus.run_suite2p(parent)
suite2p gui.menus.suggest_merge(parent)
suite2p gui.menus.vis_window(parent)
suite2p gui.menus.visualizations(parent)
18.10 suite2p.gui.merge module

```python
class suite2p.gui.merge.LineEdit(key, parent=None):
    Bases: PyQt5.QtWidgets.QLineEdit
    get_text()
    set_text(ops)

class suite2p.gui.merge.MergeWindow(parent=None):
    Bases: PyQt5.QtWidgets.QDialog
    compute_merge_list(parent)
    do_merge(parent)
    set_merge_list(parent, goodind)
    suggest_merge(parent)
suite2p.gui.merge.apply(parent)
suite2p.gui.merge.distance_matrix(parent, ilist)
suite2p.gui.merge.do_merge(parent)
suite2p.gui.merge.merge_activity_masks(parent)
```

18.11 suite2p.gui.reggui module

```python
class suite2p.gui.reggui.BinaryPlayer(parent=None):
    Bases: PyQt5.QtWidgets.QMainWindow
    add_masks()
    add_raw()
    add_red()
    add_zstack()
    cell_chosen()
    cell_mask()
    compute_z(parent)
    createButtons(parent)
    fitToWindow()
    go_to_frame()
    jump_to_frame()
    keyPressEvent(self, QKeyEvent)
    load_zstack()
    make_masks()
    next_frame()
    number_chosen()
```
open()
openCombined(save_folder)
openFile(filename, fromgui)
on_open_combined()
pause()
plot_clicked(event)
plot_trace()
plot_zcorr()
setup_views()
start()
updateButtons()
updateFrameSlider()
zoom_image()
class suite2p.gui.reggui.PViewer(parent=None)
    Bases: PyQt5.QtWidgets.QMainWindow
    createButtons()
    keyPressEvent(self, QKeyPressEvent)
nex_frame()
onen()
onenFile(filename)
pause()
plot_clicked(event)
plot_frame()
start()
zoom_plot()

suite2p.gui.reggui.subsample_frames(ops, nsamps, reg_loc)

18.12 suite2p.gui.rungui module

class suite2p.gui.rungui.LineEdit(k, key, parent=None)
    Bases: PyQt5.QtWidgets.QLineEdit
    get_text(intkeys, boolkeys)
    set_text(ops)
class suite2p.gui.rungui.OpsButton(bid, Text, parent=None)
    Bases: PyQt5.QtWidgets.QPushButton
    press(parent, bid)
class suite2p.gui.rungui.RunWindow(parent=None)
    Bases: PyQt5.QtWidgets.QDialog
add_batch()
add_ops()
bin_folder()
clean_script()
compile_ops_db()
create_buttons()
finished()
    finished(self, int) [signal]
get_folders()
get_h5py()
load_db()
load_ops(name=None)
parse_inputformat()
remove_ops()
reset_ops()
revert_default_ops()
run_S2P()
save_default_ops()
save_folder()
save_ops()
save_text()
started()
stderr_write()
stdout_write()
stop()
class suite2p.gui.rungui.TextChooser(parent=None)
    Bases: PyQt5.QtWidgets.QDialog
    exit_list()

class suite2p.gui.rungui.VerticalLabel(text=None)
    Bases: PyQt5.QtWidgets.QWidget
    paintEvent(self, QPaintEvent)
18.13 suite2p.gui.traces module

- `suite2p.gui.traces.collapse_scale(parent)`
- `suite2p.gui.traces.collapse_trace(parent)`
- `suite2p.gui.traces.deconv_on(parent)`
- `suite2p.gui.traces.expand_scale(parent)`
- `suite2p.gui.traces.expand_trace(parent)`
- `suite2p.gui.traces.make_buttons(parent, b0)`
- `suite2p.gui.traces.nc_chosen(parent)`
- `suite2p.gui.traces.neuropil_on(parent)`
- `suite2p.gui.traces.plot_trace(parent)`
- `suite2p.gui.traces.traces_on(parent)`

18.14 suite2p.gui.utils module

- `suite2p.gui.utils.boundary(ypix, xpix)`
  - returns pixels of mask that are on the exterior of the mask
- `suite2p.gui.utils.circle(med, r)`
  - returns pixels of circle with radius 1.25x radius of cell (r)

18.15 suite2p.gui.views module

- `class suite2p.gui.views.RangeSlider(parent=None, *args)`
  - Bases: PyQt5.QtWidgets.QSlider
  - A slider for ranges.
    - This class provides a dual-slider for ranges, where there is a defined maximum and minimum, as is a normal slider, but instead of having a single slider value, there are 2 slider values.
    - This class emits the same signals as the QSlider base class, with the exception of valueChanged
    - Found this slider here: https://www.mail-archive.com/pyqt@riverbankcomputing.com/msg22889.html and modified it
  - `high()`
  - `level_change()`
  - `low()`
  - `mouseMoveEvent(self, QMouseEvent)`
  - `mousePressEvent(self, QMouseEvent)`
  - `mouseReleaseEvent(self, QMouseEvent)`
  - `paintEvent(self, QPaintEvent)`
  - `setHigh(high)`
setLow(low)

```python
class suite2p.gui.views.ViewButton(bid, Text, parent=None):
    Bases: PyQt5.QtWidgets.QPushButton
    
    custom QPushButton class for quadrant plotting requires buttons to put into a QButtonGroup (parent.viewbtns)
    allows only 1 button to pressed at a time
    
    press(parent, bid)
```

```python
suite2p.gui.views.init_views(parent)
make views using parent.ops


assigns parent.views
```

```python
suite2p.gui.views.make_buttons(parent)
view buttons
```

```python
suite2p.gui.views.plot_views(parent)
set parent.view1 and parent.view2 image based on parent.ops_plot['view']
```

### 18.16 suite2p.gui.visualize module

```python
class suite2p.gui.visualize.NeuronSlider(parent=None):
    Bases: suite2p.gui.visualize.RangeSlider

    level_change()
```

```python
class suite2p.gui.visualize.RangeSlider(parent=None, *args):
    Bases: PyQt5.QtWidgets.QSlider
    
    A slider for ranges.

    This class provides a dual-slider for ranges, where there is a defined maximum and minimum, as is a normal slider, but instead of having a single slider value, there are 2 slider values.

    This class emits the same signals as the QSlider base class, with the exception of valueChanged

    Found this slider here: https://www.mail-archive.com/pyqt@riverbankcomputing.com/msg22889.html and modified it
    
    high()
    
    level_change()
    
    low()

    mouseMoveEvent(self, QMouseEvent)

    mousePressEvent(self, QMouseEvent)

    mouseReleaseEvent(self, QMouseEvent)

    paintEvent(self, QPaintEvent)

    setHigh(high)

    setLow(low)
```

```python
class suite2p.gui.visualize.SatSlider(parent=None):
    Bases: suite2p.gui.visualize.RangeSlider
```

### 18.16. suite2p.gui.visualize module
level_change()

class suite2p.gui.visualize.Slider(bid, parent=None)
    Bases: PyQt5.QtWidgets.QSlider
    level_change(parent, bid)

class suite2p.gui.visualize.VerticalLabel(text=None)
    Bases: PyQt5.QtWidgets.QWidget
    paintEvent(self, QPaintEvent)

class suite2p.gui.visualize.VisWindow(parent=None)
    Bases: PyQt5.QtWidgets.QMainWindow
        LINE_position()
        PC_on(plot)
        PReturn()
        ROI_position()
        THRES_position()
        activate()
        compute_map()
        finished()
        keyPressEvent(self, QKeyEvent)
        neural_sorting(i)
        plot_clicked(event)
        plot_traces()
        roi_range(roi)
        select_cells()
        sort_time()
        stderr_write()
        stdout_write()

18.17 Module contents
suite2p.classification, 86
suite2p.classification.classifier, 85
suite2p.classification.classify, 86
suite2p.detection, 79
suite2p.detection.chan2detect, 71
suite2p.detection.denoise, 71
suite2p.detection.detect, 71
suite2p.detection.sourcery, 72
suite2p.detection.sparsedetect, 73
suite2p.detection.stats, 75
suite2p.detection.utils, 78
suite2p.extraction, 84
suite2p.extraction.dcnv, 81
suite2p.extraction.extract, 82
suite2p.extraction.masks, 83
suite2p.gui, 98
suite2p.gui.buttons, 87
suite2p.gui.classgui, 88
suite2p.gui.drawroi, 88
suite2p.gui.graphics, 89
suite2p.gui.gui2p, 90
suite2p.gui.io, 90
suite2p.gui.masks, 91
suite2p.gui.menus, 92
suite2p.gui.merge, 93
suite2p.gui.reggui, 93
suite2p.gui.rungui, 94
suite2p.gui.traces, 96
suite2p.gui.utils, 96
suite2p.gui.views, 96
suite2p.gui.visualize, 97
suite2p.io, 57
suite2p.io.binary, 51
suite2p.io.h5, 54
suite2p.io.mwb, 54
suite2p.io.save, 54
suite2p.io.sbx, 55
suite2p.io.server, 55
suite2p.io.tifl, 55
suite2p.io.utils, 56
suite2p.registration, 70
suite2p.registration.bidiphase, 59
suite2p.registration.metrics, 59
suite2p.registration.nonrigid, 61
suite2p.registration.register, 64
suite2p.registration.rigid, 66
suite2p.registration.utils, 67
suite2p.registration.zalign, 69
collapse_trace() (in module suite2p.gui.traces), 96
ColorButton (class in suite2p.gui.masks), 91
combine_offsets_across_batches() (in module suite2p.registration.utils), 67
combined() (in module suite2p.io.save), 54
compile_ops_db() (suite2p.gui.rungui.RunWindow method), 95
complex_fft2() (in module suite2p.registration.utils), 67
compute() (in module suite2p.registration.bidiphaseregister), 59
crop() (in module suite2p.registration.register), 64
crop_dydx() (in module suite2p.io.save), 54
compute_map() (suite2p.gui.visualize.VisWindow method), 98
compute_masks() (in module suite2p.registration.rigid), 66
compute_merge_list() (suite2p.gui.merge.MergeWindow method), 93
compute_reference() (in module suite2p.registration.register), 64
compute_reference_masks() (in module suite2p.registration.register), 65
compute_z() (suite2p.gui.reggui.BinaryPlayer method), 93
compute_zpos() (in module suite2p.registration.zalign), 69
connected_region() (in module suite2p.detection.sourcery), 72
convert_numpy_file_to_suite2p_binary() (suite2p.io.binary.BinaryFile static method), 51
cov() (in module suite2p.registration.utils), 67
corr_masks() (in module suite2p.gui.masks), 91
corr_to_template() (in module suite2p.registration.metrics), 59
correct_bleedthrough() (in module suite2p.detection.chan2detect), 71
count_overlaps() (in module suite2p.detection.stats), 77
cov() (suite2p.detection.stats.EllipseData property), 75
create_buttons() (suite2p.gui.rungui.RunWindow method), 95
create_cell_mask() (in module suite2p.registration.utils), 83
create_cell_pix() (in module suite2p.registration.utils), 83
create_masks() (in module suite2p.registration.utils), 83
create_masks_and_extract() (in module suite2p.registration.utils), 82
create_masks_of_cells() (suite2p.gui.drawroi.ROIDraw method), 88
custom_masks() (in module suite2p.gui.masks), 91
data (suite2p.io.binary.BinaryFile property), 51
detector() (in module suite2p.detection.chan2detect), 71
detect() (in module suite2p.detection.detect), 71
disable() (in module suite2p.gui.classgui), 88
distance_kernel() (in module suite2p.detection.stats), 77
distance_matrix() (in module suite2p.detection.detect), 71
do_crop() (suite2p.detection.stats.ROI attribute), 75
do_merge() (in module suite2p.detection.detect), 71
do_merge() (suite2p.detection.detect.detect), 93
downsample() (in module suite2p.detection.stats), 95
dragEnterEvent() (suite2p.gui.gui2p.MainWindow method), 90
draw() (suite2p.gui.drawroi.sROI method), 89
draw_colorbar() (in module suite2p.gui.masks), 91
draw_masks() (in module suite2p.gui.masks), 91
drawClusters() (in module suite2p.detection.detect), 71
dropEvent() (suite2p.gui.gui2p.MainWindow method), 90
dx() (suite2p.detection.stats.EllipseData property), 75
dy() (suite2p.detection.stats.EllipseData property), 75
ellipse (suite2p.detection.stats.EllipseData property), 75
EllipseData (class in suite2p.detection.stats), 75
enable_views_and_classifier() (in module suite2p.gui.io), 90
enhanced_mean_image() (in module suite2p.extraction.extract), 82
enhanced_mean_image() (in module suite2p.registration.register), 65
estimate.spatial.scale() (in module suite2p.detection.sparsedetect), 74
Estimated() (suite2p.detection.sparsedetect.EstimateMode attribute), 73
EstimateMode (class in suite2p.detection.sparsedetect), 73
make_colorbar() (in module suite2p.gui.masks), 92
make_colors() (in module suite2p.gui.masks), 92
make_graphics() (suite2p.gui.gui2p.MainWindow method), 90
make_masks() (suite2p.gui.reggui.BinaryPlayer method), 93
make_masks_and_enable_buttons() (in module suite2p.io.io), 91
make_quadrants() (in module suite2p.gui.buttons), 87
make_selection() (in module suite2p.gui.buttons), 87
manual_label() (in module suite2p.gui.menus), 92
map_coordinates() (in module suite2p.registration.nonrigid), 62
mask_ious() (in module suite2p.detection.utils), 78
mask_stats() (in module suite2p.detection.utils), 79
masks_and_traces() (in module suite2p.gui.drawroi), 89
mat_upsample() (in module suite2p.registration.utils), 68
match_masks() (in module suite2p.detection.utils), 79
matmul_neuropil() (in module suite2p.extraction.extract), 83
matmul_traces() (in module suite2p.extraction.extract), 83
mean_r_squared (suite2p.detection.stats.ROI property), 76
mean_r_squared() (in module suite2p.detection.stats), 77
mean_r_squared0 (suite2p.detection.stats.ROI property), 76
mean_r_squared_compact (suite2p.detection.stats.ROI property), 76
med (suite2p.detection.stats.ROI attribute), 76
median_pix() (in module suite2p.detection.stats), 77
merge_activity_masks() (in module suite2p.gui.merge), 93
mergebar() (in module suite2p.gui.menus), 92
MergeWindow (class in suite2p.gui.merge), 93
meshgrid_mean_centered() (in module suite2p.registration.utils), 68
mesoscan_to_binary() (in module suite2p.io.tiff), 55
minDistance() (in module suite2p.detection.sourcery), 73
mode_change() (suite2p.gui.gui2p.MainWindow method), 90

Module
suite2p.classification, 86
suite2p.classification.classifier, 85
suite2p.classification.detect, 79
suite2p.detection, 79
suite2p.detection.chan2detect, 71
suite2p.detection.denoise, 71
suite2p.detection.detect, 71
suite2p.detection.sourcery, 72
suite2p.detection.sparsedetect, 73
suite2p.detection.stats, 75
suite2p.detection.utils, 78
suite2p.extraction, 84
suite2p.extraction.dcnv, 81
suite2p.extraction.extract, 82
suite2p.extraction.masks, 83
suite2p.gui, 98
suite2p.gui.buttons, 87
suite2p.gui.classgui, 88
suite2p.gui.drawroi, 88
suite2p.gui.graphics, 89
suite2p.gui.gui2p, 90
suite2p.gui.io, 90
suite2p.gui.masks, 91
suite2p.gui.menus, 92
suite2p.gui.merge, 93
suite2p.gui.reggui, 93
suite2p.gui.rungui, 94
suite2p.gui.traces, 96
suite2p.gui.utils, 96
suite2p.gui.views, 96
suite2p.gui.visualize, 97
suite2p.io, 57
suite2p.io.binary, 51
suite2p.io.h5, 54
suite2p.io.nwb, 54
suite2p.io.save, 54
suite2p.io.sbx, 55
suite2p.io.server, 55
suite2p.io.tiff, 55
suite2p.io.utils, 56
suite2p.registration, 70
suite2p.registration.bidiphase, 59
suite2p.registration.metrics, 59
suite2p.registration.nonrigid, 61
suite2p.registration.register, 64
suite2p.registration.rigid, 66
suite2p.registration.utils, 67
suite2p.registration.zalign, 69
morphOpen() (in module suite2p.detection.sourcery), 73
mouseMoved() (suite2p.gui.drawroi.ROIDraw method), 89
mouseClickEvent() (suite2p.gui.graphics.ViewBox method), 89
mouseDoubleClickEvent() (suite2p.gui.graphics.TraceBox method), 89
mouseDoubleClickEvent() (suite2p.gui.graphics.ViewBox method), 89
mouseDragEvent() (suite2p.gui.graphics.ViewBox method), 89
save_default() (suite2p.gui.classgui.ListChooser method), 88
save_default_ops() (suite2p.rungui.RunWindow method), 95
save_folder() (suite2p.rungui.RunWindow method), 95
save_iscell() in module suite2p.gui.io, 91
save_list() in module suite2p.gui.classgui, 88
save_mat() in module suite2p.gui.io, 91
save_merge() in module suite2p.gui.io.save, 54
save_model() in module suite2p.gui.classgui, 88
save_nwb() in module suite2p.io.nwb, 54
save_ops() (suite2p.gui.rungui.RunWindow method), 95
save_redcell() in module suite2p.gui.io, 91
save_text() (suite2p.rungui.RunWindow Method), 95
save_tiff() in module suite2p.io.tif, 56
sbx_to_binary() in module suite2p.io.sbx, 55
search_for_ext() in module suite2p.io.utils, 57
select_cells() (suite2p.gui.classgui.MainWindow method), 90
select_cells() (suite2p.gui.visualize.VisWindow method), 98
select_rois() in module suite2p.detection.detect, 71
send_jobs() in module suite2p.io.server, 55
set_merge_list() (suite2p.gui.merge.MergeWindow method), 93
set_text() (suite2p.gui.merge.LineEdit method), 93
set_text() (suite2p.rungui.LineEdit method), 94
setHigh() (suite2p.gui.views.RangeSlider method), 96
setHigh() (suite2p.gui.visualize.RangeSlider method), 97
setLow() (suite2p.gui.views.RangeSlider method), 96
setLow() (suite2p.gui.visualize.RangeSlider method), 97
setup_views() (suite2p.gui.rungui.BinaryPlayer method), 94
shape (suite2p.io.binary.BinaryFile property), 52
shift() in module suite2p.registration.bidiphase, 59
shift_coordinates() in module suite2p.registration.nonrigid, 63
shift_frame() in module suite2p.registration.rigid, 67
shift_frames() in module suite2p.registration.register, 66
size (suite2p.io.binary.BinaryFile property), 52
SizeButton (class in suite2p.gui.buttons), 87
Slider (class in suite2p.gui.visualize), 98
solidity (suite2p.detection.stats.ROI property), 76
soma_crop (suite2p.detection.stats.ROI property), 76
sort_time() (suite2p.gui.visualize.VisWindow method), 98
sourcery() in module suite2p.detection.sourcery, 73
sparsery() in module suite2p.detection.sparsedetect, 74
spatial_high_pass() in module suite2p.registration.utils, 68
spatial_smooth() in module suite2p.registration.utils, 69
spatial_taper() in module suite2p.registration.utils, 69
square_convolution_2d() in module suite2p.detection.sparsedetect, 74
square_taper() in module suite2p.registration.utils, 69
sROI (class in suite2p.gui.drawroi), 89
ssh_connect() in module suite2p.io.server, 55
standard_deviation_over_time() in module suite2p.detection.utils, 79
start() (suite2p.gui.reggui.BinaryPlayer method), 94
start() (suite2p.gui.reggui.PCViewer method), 94
started() (suite2p.gui.rungui.RunWindow method), 95
stats_dicts_to_3d_array() (suite2p.detection.stats.ROI class method), 76
stderr_write() (suite2p.gui.rungui.RunWindow method), 95
stdout_write() (suite2p.gui.rungui.RunWindow method), 95
stdout_write() (suite2p.gui.rungui.RunWindow method), 95
start() (suite2p.gui.rungui.RunWindow method), 95
sub2ind() in module suite2p.detection.sourcery, 73
subsamp下来在（module suite2p.reggui), 94
suggest_merge() in module suite2p.gui.menus, 92
suggest_merge() (suite2p.gui.merge.MergeWindow method), 93
suite2p.classification module, 86
suite2p.classification.classifier module, 85
suite2p.classification.classify module, 86
suite2p.detection module, 79
suite2p.detection.chan2detect module, 71
suite2p.detection.denoise module, 71
suite2p.detection.detect module, 71
suite2p.detection.sourcery module, 72
suite2p.detection.sparsedetect module, 73
suite2p.detection.stats
  module, 75
suite2p.detection.utils
  module, 78
suite2p.extraction
  module, 84
suite2p.extraction.dcnv
  module, 81
suite2p.extraction.extract
  module, 82
suite2p.extraction.masks
  module, 83
suite2p.gui
  module, 98
suite2p.gui.buttons
  module, 87
suite2p.gui.classgui
  module, 88
suite2p.gui.drawroi
  module, 88
suite2p.gui.graphics
  module, 89
suite2p.gui.gui2p
  module, 90
suite2p.gui.io
  module, 90
suite2p.gui.masks
  module, 91
suite2p.gui.menus
  module, 92
suite2p.gui.merge
  module, 93
suite2p.gui.reggui
  module, 93
suite2p.gui.rungui
  module, 94
suite2p.gui.traces
  module, 96
suite2p.gui.utils
  module, 96
suite2p.gui.views
  module, 96
suite2p.gui.visualize
  module, 97
suite2p.io
  module, 57
suite2p.io.binary
  module, 51
suite2p.io.h5
  module, 54
suite2p.io.mwb
  module, 54
suite2p.io.save
  module, 54
suite2p.io.server
  module, 55
suite2p.io.tifff
  module, 55
suite2p.io.utils
  module, 56
suite2p.registration
  module, 70
suite2p.registration.bidiphase
  module, 59
suite2p.registration.metrics
  module, 59
suite2p.registration.nonrigid
  module, 61
suite2p.registration.register
  module, 64
suite2p.registration.rigid
  module, 66
suite2p.registration.utils
  module, 67
suite2p.registration.zalign
  module, 69

temporal_high_pass_filter() (in module suite2p.detection.utils), 79
temporal_smooth() (in module suite2p.registration.utils), 69
temporary_pointer() (in module suite2p.io.binary), 54
TextChooser (class in suite2p.gui.rungui), 95
THRES_position() (suite2p.gui.visualize.VisWindow method), 98
threshold_reduce() (in module suite2p.detection.utils), 79
tiff_to_binary() (in module suite2p.io.tifff), 56
to_array() (suite2p.detection.stats.ROI method), 77
top_number_chosen() (suite2p.gui.gui2p.MainWindow method), 90
top_selection() (suite2p.gui.buttons.TopButton method), 87
TopButton (class in suite2p.gui.buttons), 87
TraceBox (class in suite2p.gui.graphics), 89
traces_on() (in module suite2p.gui.traces), 96
transform_data() (in module suite2p.registration.nonrigid), 63
two_comps() (in module suite2p.detection.sparsedetect), 74

unix_path() (in module suite2p.io.server), 55
update_plot() (suite2p.gui.gui2p.MainWindow method), 90
updateButtons() (suite2p.gui.reggui.BinaryPlayer method), 94
updateFrameSlider() (suite2p.gui.reggui.BinaryPlayer method), 94
upsample_block_shifts() (in module suite2p.registration.nonrigid), 64
use_sktiff_reader() (in module suite2p.io.tiff), 56

V
VerticalLabel (class in suite2p.run.gui), 95
VerticalLabel (class in suite2p.run.gui.visualize), 98
ViewBox (class in suite2p.run.gui.graphics), 89
ViewButton (class in suite2p.run.gui.drawroi), 89
ViewButton (class in suite2p.run.gui.views), 97
vis_window() (in module suite2p.run.gui.menus), 92
visualizations() (in module suite2p.run.gui.menus), 92
VisWindow (class in suite2p.run.gui.visualize), 98

W
write() (suite2p.run.binary.BinaryFile method), 53

X
xpix (suite2p.detection.stats.ROI attribute), 77

Y
ypix (suite2p.detection.stats.ROI attribute), 77

Z
zoom_cell() (suite2p.run2p.MainWindow method), 90
zoom_image() (suite2p.run2p.BinaryPlayer method), 94
zoom_plot() (suite2p.run.gui.graphics.TraceBox method), 89
zoom_plot() (suite2p.run.gui.graphics.ViewBox method), 89
zoom_plot() (suite2p.run.reggui.PCViewer method), 94
zoom_to_cell() (suite2p.run2p.MainWindow method), 90