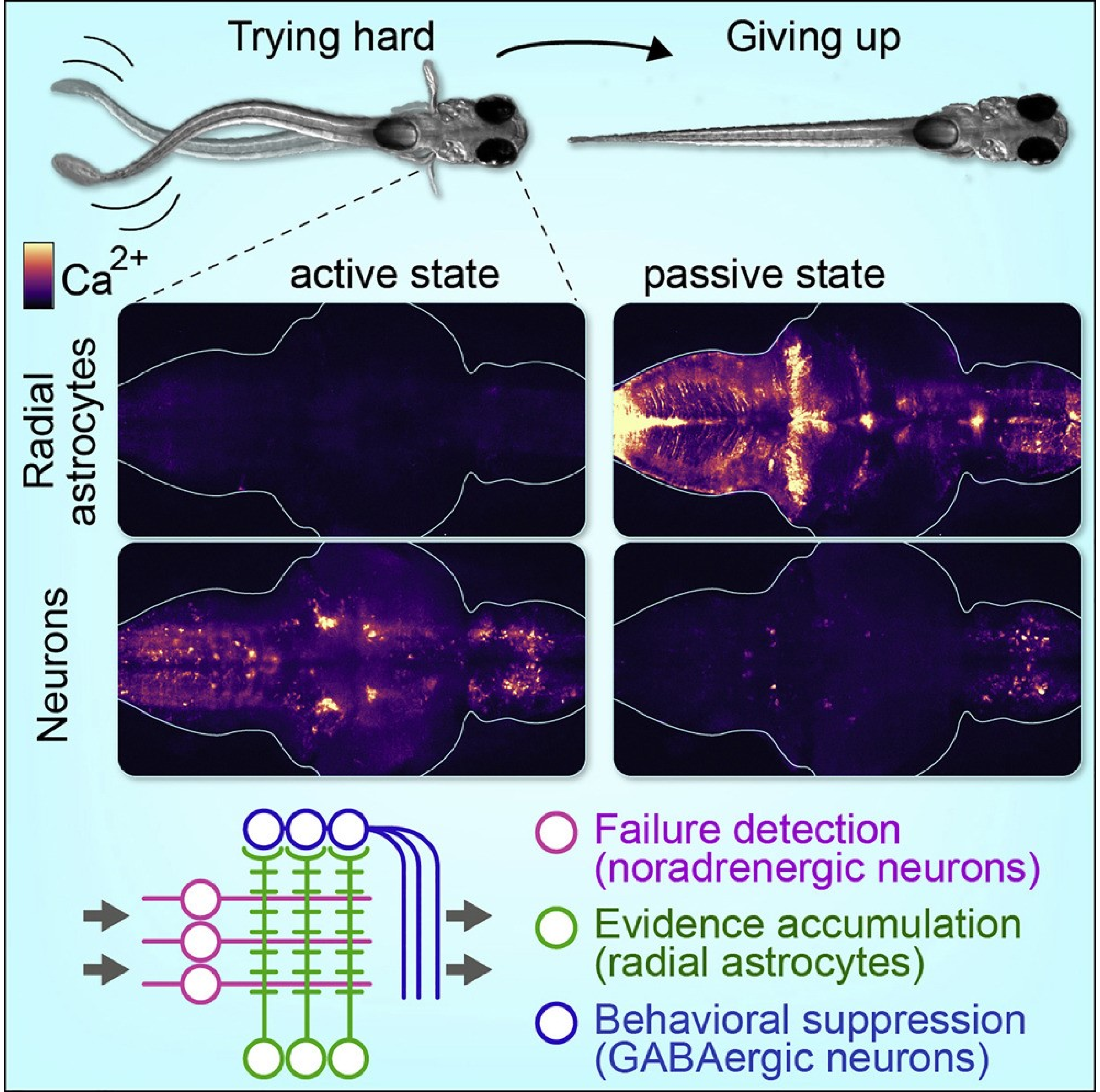
# Volumetric segmentation pipeline

Mika Rubinov

2020-03


Trying hard
Giving up
Ca$^{2+}$
active state
passive state
Radial astrocytes
Neurons
Failure detection (noradrenergic neurons)
Evidence accumulation (radial astrocytes)
Behavioral suppression (GABAergic neurons)

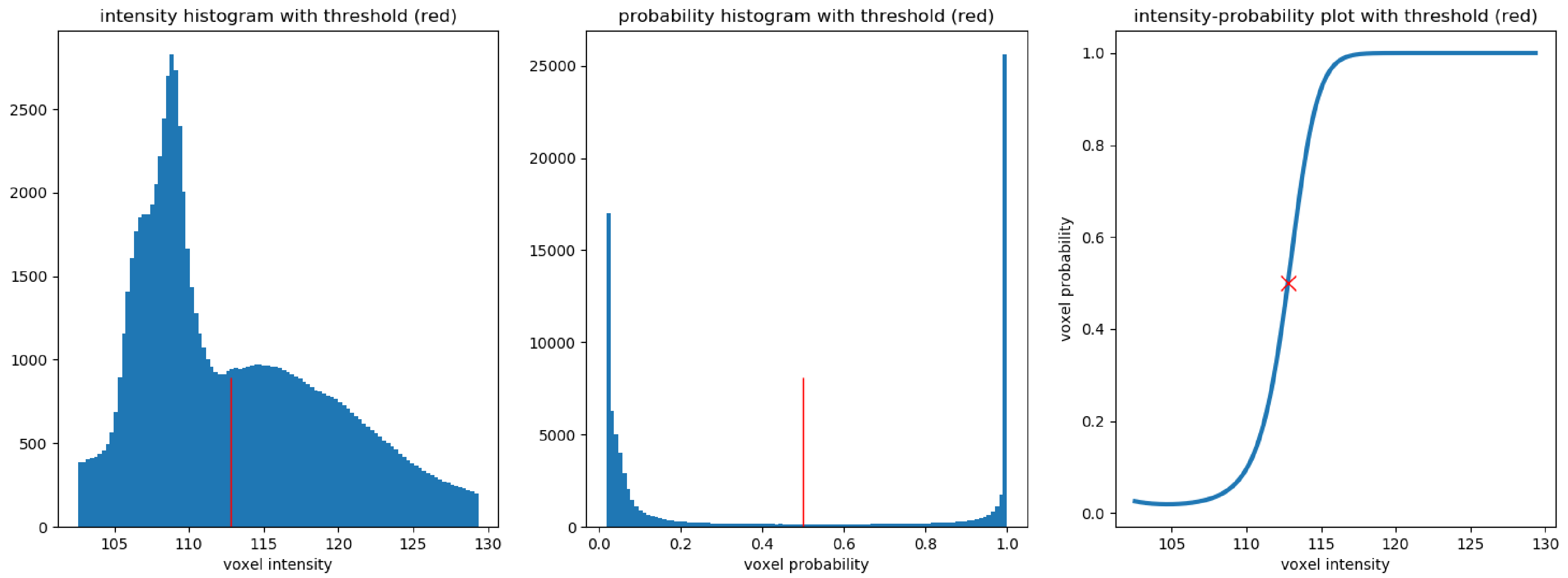
github.com/mikarubi/voluseg

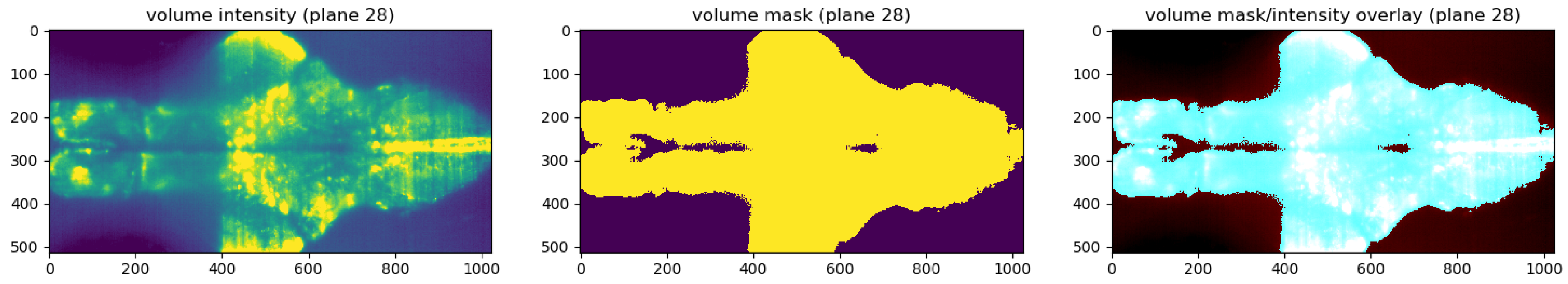
# Volumetric segmentation pipeline

1. Bin/downsample volumes (optional)

2. Register volumes with ANTS (optional)

3. Get average volume and brain mask


intensity histogram with threshold (red)
voxel intensity
probability histogram with threshold (red)
voxel probability
intensity-probability plot with threshold (red)
voxel probability
voxel intensity


volume intensity (plane 28)
volume mask (plane 28)
volume mask/intensity overlay (plane 28)
0
100
200
300
400
500
0
200
400
600
800
1000

## 4. Cell segmentation

a. Split volume into equally sized blocks

b. Correct plane-acquisition time delays

c. Initialize cell location using pixels that:
    i. have local intensity peaks
    ii. are sufficiently close in space
    iii. are sufficiently highly correlated

# 4. Cell segmentation

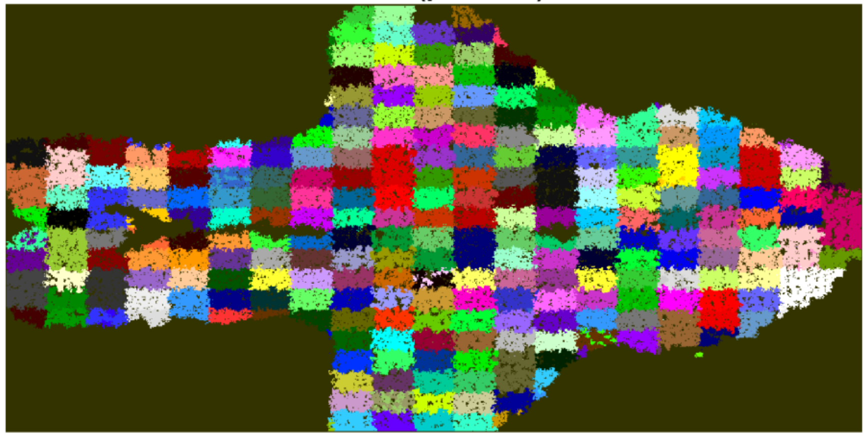
## d. Constrained factorization:

NMF: $V_{n \times t} \approx W_{n \times (c+1)}\ H_{(c+1) \times t}$, where:

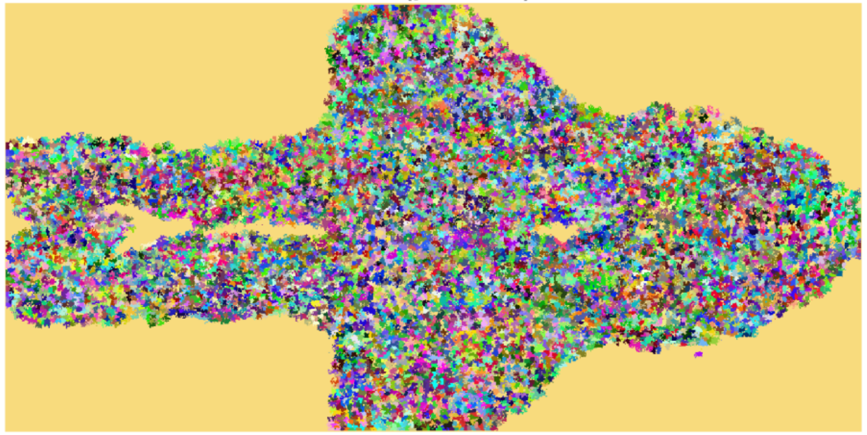
$n \sim 10^7$ pixels; $t \sim 10^4$ timepoints; $c \sim 10^5$ cell segments.

Non-negativity constraints
Hard spatial constraints
Sparseness projection
Background-signal removal

blocks (plane 28)



cells (plane 28)



## 5. Compute baseline

a. Remove duplicate cells

b. Detrend

c. Compute baseline

## Dependencies

[preconfigured on ahrens_lab1]

python: h5py, matplotlib, nibabel, numpy, pandas, scipy, skimage, sklearn

spark

ants for registration (optional)

## Installation

[preinstalled on ahrens_lab1]

```
pip install git+https://github.com/mikarubi/voluseg.git
```

## Usage

1. Generate parameter file

2. Run segmentation

# Generate parameter file

```
# get parameter dictionary and set directories
parameters = voluseg.parameter_dictionary()
parameters['dir_input'] = '/path/to/input_directory'
parameters['dir_output'] = '/path/to/output_directory'
parameters['diam_cell'] = 5.0

# fetch z-resolution, exposure time, and stack frequency
channel_file = os.path.join(parameters['dir_input'], 'ch0.xml')
stack_file = os.path.join(parameters['dir_input'], 'Stack_frequency.txt')
parameters = voluseg.load_metadata(parameters, channel_file, stack_file)

# save parameters
voluseg.step0_process_parameters(parameters)
```

## Run registration

[on ahrens_lab1, after parameter file saved in output directory]

**bsub -n32 voluseg_spark_janelia [nodes] /path/to/output/directory**

```
import os, voluseg
parameters = voluseg.parameter_dictionary()

{registration: medium,                diam_cell: 6.0,
 dir_ants: ,                          dir_input: ,
 dir_output: ,                        ds: 2,
 planes_pad: 0,                       nt: 1000,
 f_hipass: 0,                         f_volume: 2.0,
 n_cells_block: 100,                  n_colors: 1,
 res_x: 0.40625,                      res_y: 0.40625,
 res_z: 5.0,                          t_baseline: 300,
 t_section: 0.01,                     thr_mask: 0.5}
```

```
voluseg.parameter_dictionary??
    registration: medium,    # registration quality: high, medium, low (none or None for no registration)
    diam_cell: 6.0,          # cell diameter (microns)
    dir_ants: ,              # path to ANTs directory
    dir_input: ,             # path to image directory
    dir_output: ,            # path to output directory
    ds: 2,                   # spatial coarse-graining in x-y dimension
    planes_pad: 0,           # number of planes to pad the volume with (for robust registration)
    nt: 1000,                # number of timepoints to use for cell detection (use all points if nt = 0)
    f_hipass: 0,             # frequency (Hz) for high-pass filtering of cell timeseries
    f_volume: 2.0,           # imaging frequency (Hz)
    n_cells_block: 100,      # number of cells in block
    n_colors: 1,             # number of brain colors (2 in two-color images)
    res_x: 0.40625,          # x resolution (microns)
    res_y: 0.40625,          # y resolution (microns)
    res_z: 5.0,              # z resolution (microns)
    t_baseline: 300,         # interval for baseline calculation (seconds)
    t_section: 0.01,         # exposure time (seconds): time of slice acquisition
    thr_mask: 0.5,           # threshold for volume mask: 0 < thr <= 1 (probability) or thr > 1 (intensity)
```

```
registration: medium,
        # registration quality: high, medium, low (none or None for no registration)
diam_cell: 6.0,
        # cell diameter (microns)
dir_ants: ,
        # path to ANTs directory
dir_input: ,
        # path to image directory
dir_output: ,
        # path to output directory
ds: 2,
        # spatial coarse-graining in x-y dimension
planes_pad: 0,
        # number of planes to pad the volume with (for robust registration)
nt: 1000,
        # number of timepoints to use for cell detection (use all points if nt = 0)
f_hipass: 0,
        # frequency (Hz) for high-pass filtering of cell timeseries
```

```
f_volume: 2.0,
        # imaging frequency (Hz)
n_cells_block: 100,
        # number of cells in block
n_colors: 1,
        # number of brain colors (2 in two-color images)
res_x: 0.40625,
        # x resolution (microns)
res_y: 0.40625,
        # y resolution (microns)
res_z: 5.0,
        # z resolution (microns)
t_baseline: 300,
        # interval for baseline calculation (seconds)
t_section: 0.01,
        # exposure time (seconds): time of slice acquisition
thr_mask: 0.5,
        # threshold for volume mask: 0 < thr <= 1 (probability) or thr > 1 (intensity)
```

## Manual execution

```
filename_parameters = os.path.join(dir_output, 'parameters.pickle')

parameters = voluseg.load_parameters(filename_parameters)

voluseg.step1_process_images(parameters)

voluseg.step2_align_images(parameters)

voluseg.step3_mask_images(parameters)

voluseg.step4_detect_cells(parameters)

voluseg.step5_clean_cells(parameters)
```

## Pipeline output: parameters

- ‘prepro.output’
  - spark configuration
  - parameters
  - run time

- ‘parameters.pickle’
  - parameter dictionary, load with:
  - parameters = voluseg.load_parameters(filename_parameters)
  - dictionary required as input to individual pipeline steps

## Pipeline output: quality control

- ‘mask_plots’
  - directory of average volume plane images
  - brain mask superimposed on brain volume
  - can be used to assess goodness of brain masks

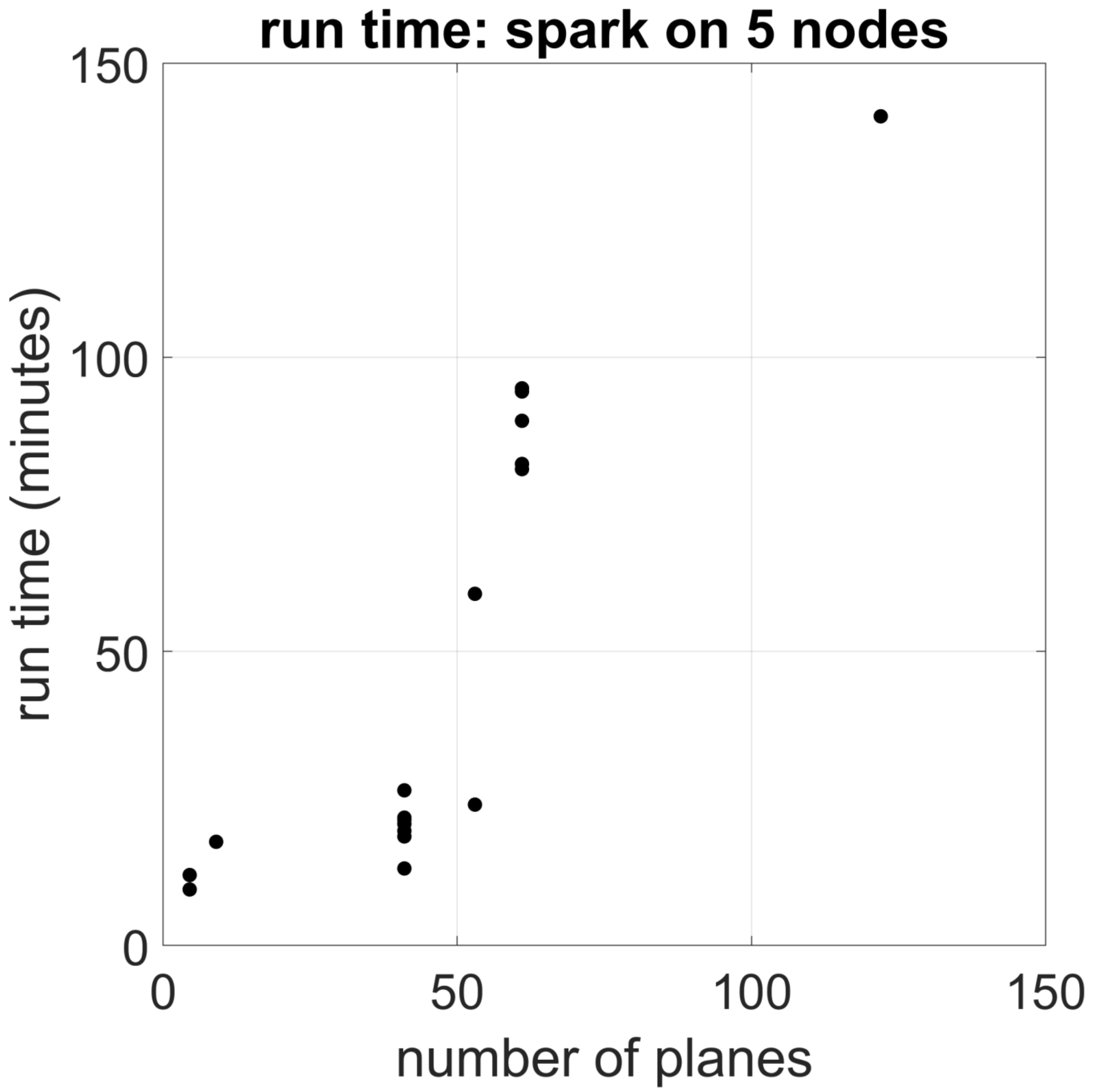
- `transforms’ directory
  - affine transforms for individual volumes
  - can be used to assess movement of individual volumes

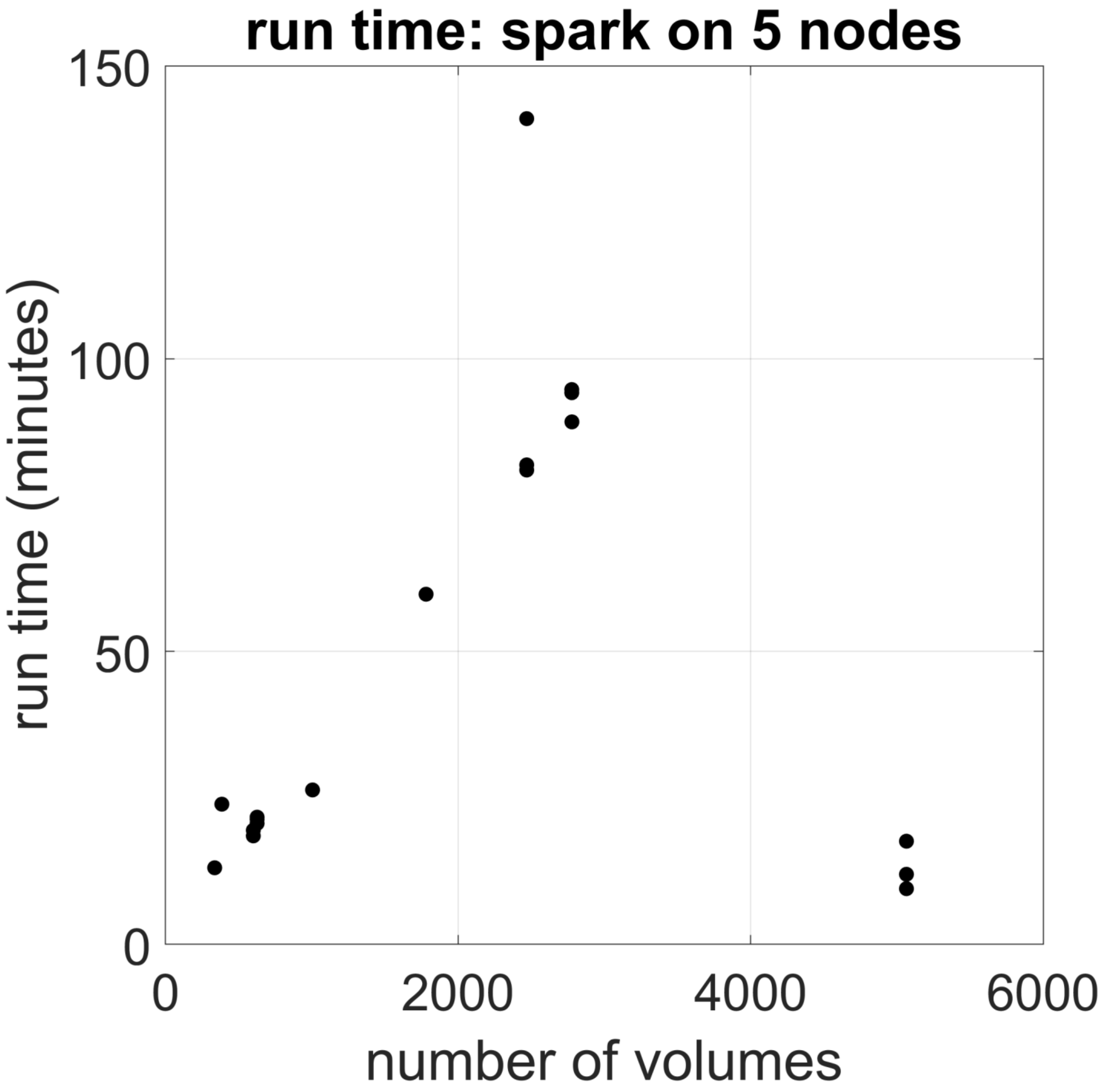
## Pipeline output: ‘volume0.hdf5’

- ‘background’: estimated background fluorescence
- ‘block_valids’: indices of blocks used for segmentation
- ‘block_xyz0/1’: min/max block xyz coordinates
- ‘n_blocks’: total number of blocks
- ‘n_voxels_cells’: approximate number of voxels in each cell
- ‘thr_intensity’: brain-mask intensity threshold
- ‘thr_probability’: brain-mask probability threshold
- ‘timepoints’: indices of timepoints used for cell segmentation
- ‘timeseries_mean’: volume-mean timeseries
- ‘volume_mean/mask/peak’: volume mean/mask/local peak intensity

## Pipeline output: ‘cells0_clean.hdf5’

- ‘background’: estimated background fluorescence
- ‘cell_baseline’: computed cell baselines
- ‘cell_timeseries’: detrended [+ optionally filtered] cell timeseries
- ‘cell_timeseries_raw’: ‘raw’ cell_timeseries (direct output of segmentation)
- ‘cell_weights’: cell spatial footprints (spatial NMF components)
- ‘cell_x/y/z’: cell x, y, z coordinates
- ‘n/t’: number of cells/timepoints
- ‘volume_id’: cell ids represented on a volume
- ‘volume_weight’: cell spatial footprints represent on a volume
- ‘x/y/z’: x, y, z volume dimensions


run time: spark on 5 nodes
run time (minutes)
number of planes
0
50
100
150
0
50
100
150



run time: spark on 5 nodes
150
100
50
0
run time (minutes)
0
2000
4000
6000
number of volumes