Presynaptic workflow example

February 7, 2021

$Revision: 1695 $

1 Summary

This example shows a complete presynaptic workflow.

2 Segmentation and analysis of individual datasets (synapses)

Important: Following this procedure will overwrite many of the files given in this example. It is recommended to execute steps in order and compare the newly generated files with the corresponding example files.

2.1 General info

A more detailed description of the relevant computational methods is given in segmentation.pdf .
All relevant data is in examples/presynaptic_example/segmentation/cell_tomo-4/ .
All tasks described below have to be executed, except executing the cluster script.

2.2 Generating a boundary label image file

This step explains how a boundary label file can be generated. Here, synaptic vesicles are labeled in a semi-automated way, while other membranes, cellular structures and organelles are labeled manually.

2.2.1 Prerequisite

- tomogram (3d/tomo.mrc)

2.2.2 Steps - Direct method

- Segment the active zone membrane (viz/disks.raw, label id=2) based on the tomogram
- Segment presynaptic intracellular region (id=3) and cellular structures that should not be used for segmentation (microtubules, mitochondria, non-synaptic vesicles; ids = [35, 47, 72, 74, 75]) (save as viz/disks.raw)
- Segment equatorial circles (disks) of synaptic vesicles (ids = [36, 48, 49, 50, 52, 58, 59, 60, 61] (viz/disks.raw) visualized in the tomogram
- It is advisable that none of the labels generated in the previous steps is present on the faces
- Enter parameters in common/tomo_info.py (template pyto/scripts/tomo_info.py), the following label ids are set:

  - all_ids: all ids present in the labels file, if a label is present in the file, but is not specified in the list, it will be set to 0
  - boundary_ids: label ids for the AZ membrane and synaptic vesicles
  - vesicle ids: synaptic vesicle ids
  - segmentation_region: label id for the cytoplasm, this is the region where tethers and connectors are detected
  - distance_id: id of the AZ membrane
- Run viz/discs_to_balls.py script (template pyto/scripts/discs_to_balls.py) to generate a boundary label file (viz/labels.mrc) where equatorial discs are expanded to full balls

2.2.3 Steps - Iterative method (alternative)

- Segment the active zone membrane (viz/disks_1.raw, label id=2) in the tomogram
- Segment presynaptic intracellular region (id=3) and cellular structures that should not be used for segmentation (microtubules, mitochondria, non-synaptic vesicles; ids = [35, 47, 72, 74, 75] (viz/disks_1.raw)
- Segment equatorial circles (disks) of some synaptic vesicles (ids = [36, 48, 49, 50]) in the tomogram and save the segmentation as viz/disks_1.raw
- Enter parameters in viz/discs_to_balls_1.py script (template pyto/scripts/disks_to_balls.py) and run this script to generate viz/labels_1.raw where the labeled disks are expanded to full balls. Check the script to see how the parameters (especially label ids) are set.
- Enter parameters in viz/blank_1.py (template pyto/scripts/blank.py) and run this script to make a tomogram where the labeled synaptic vesicles are clearly marked (3d/tomo_1.mrc). This make easier the further segmentation of synaptic vesicle equatorial disks.
- Segment equatorial circles (disks) of (some) other synaptic vesicles (ids = [52, 58, 59, 60, 61]) based on the blanked tomogram (tomo_1.mrc). Depending on the segmentation program, labels_1.raw may need to be loaded. Save the segmentation as viz/disks_2.raw.
- Enter parameters in viz/discs_to_balls_2.py (template pyto/scripts/discs_to_balls.py) script and run this script to generate viz/labels-2.mrc where the newly labeled disks are expanded to full balls. Check the script to see how the parameters (especially label ids) are set.
- Continue with iterating blank and discs_to_balls until the final boundary labels file is generated, where all synaptic vesicles are fully labeled

2.2.4 Results

- Final boundary label image file (viz/labels.mrc, or viz/labels_n.mrc)

2.2.5 Notes

Other membrane segmentation tools and procedures can be used. However, the resulting boundary label file has to be consistent with the one provided here. The most important requirements are:

- Boundary label image has to contain integers
- Value 0 is reserved for the background
- Separate, specific values (ids) need to be used to label the following: the active zone region, presynaptic cytoplasm (segmentation region), synaptic vesicles (each vesicle has to have a unique id) and other organelles and cellular structures. These ids should not appear anywhere else in the label file.

- Labels should not touch tomogram sides

The most common reason for errors is the label values are not assigned properly.

2.3 Setting common parameters

If not done during the previous step, common parameters heve to be entered in common/tomo_info.py (template pyto/scripts/tomo_info.py)

2.4 Regions script

Basic greyscale statistics for the whole tomogram and for the labeled parts are calculated in this step.

2.4.1 Steps

- Run regions/regions.py script (template pyto/scripts/vesicles.py). Typically, no parameters need to be changed from the default values

2.4.2 Results

- The results are stored both in regions/tomo_regions.dat (plain text) and regions/tomo_regions.pkl (Python pickle file). The text file is used to quickly check the results. For more complicated analysis of the results, the pickle file can be loaded into a Python session.

2.5 Vesicles script

Basic greyscale, morphological and localization analysis of synaptic vesicles. Includes separate analyses for vesicle membrane, lumen and the whole vesicles.

2.5.1 Steps

- Enter parameters to regions/regions.py script (template pyto/scripts/vesicles.py). Typically, only membrane thickness (membrane_thick) need to be adjusted
- Execute the vesicles script

2.5.2 Results

Results are saved in the following files (all in regions/):

- tomo_vesicles.dat: All results in a plain text file
- tomo_vesicles.pkl, tomo_mem.pkl, tomo_lum.pkl: Pickle file results for whole vesicles, membranes and lumens, respectively. Used for the analysis of multiple datasets.

2.6 Layers script

Makes layers inside the presynaptic terminals that start at and are parallel to the AZ membrane.

2.6.1 Steps

- Enter parameters to layers/layers.py script (template pyto/scripts/layers.py). Typically, no parameters need to be changed from the defaults
- Execute the layers script

2.6.2 Results

Results are saved in the following files (all in regions/):

- labels_layers.dat: All results in a plain text file, used also for the analysis of multiple datasets.
- layers.mrc: layers image

2.7 Connectors script

Detects, analyzes and classifies tethers that link synaptic vesicles to the active zone membrane and connectors that interlink synaptic vesicles.

2.7.1 Steps

- Enter parameters to connectors/connectors.py script (template pyto/scripts/connectors.py). Typically, the only parameter that need to be changed from the default value relates to the classification by volume (class_3_volumes). The exact values depend on the pixel size and on the expected volume
- Execute the script

2.7.2 Results

Results are saved in the following files (all in connectors/):

- tomo_thr-*.dat: Single threshold results in a plain text file
- tomo_thr-*.pkl: Single threshold results in a pickle file, used for the analysis of multiple datasets.
- tomo_thr-*.mrc: Single threshold segmentation image
- tomo.dat, tomo.pkl, tomo.mrc: Combined results at all thresholds before classification in a plain text file, pickle file and as an hierarchical segmentation image, respectively.
- tomo_new_[connectors, tethers]_[small, good, big].[dat, pkl, mrc]: Combined results at all thresholds after classification for all classes obtained as a combination of classification by contacted ids (tethers, connectors) and by volume (small, good, big). The extension determines whether the file is a plain text file, pickle file or segmentation image.

2.8 Clusters script

Clusters vesicles and connectors based on distance (hierarchical clustering) and on connectivity.

2.8.1 Steps

- Enter parameters to connectors/clusters.py script (template pyto/scripts/clusters.py). The most important is:
  - in_seg_file_name: name of the pickle file generated by the connectors script that contain connectors that are considered for clustering here.
- Typically, among other parameters, only those related to hierarchical clustering may need to be changed from the defaults. These are found in the following sections:
  - Hierarchical clustering of boundaries (vesicles): distance-based clustering of vesicles
  - Hierarchical clustering of connectors: distance-based clustering of connectors
- Execute the script

2.8.2 Results

Results are saved in the following files (all in clusters/), the root is typically derived from the in_seg_file_name parameter):

- <root>_cluster_bound.dat: Resulting clusters for different clusterings of boundaries (vesicles)
  - Vesicle clusters by connectivity (via connectors)
  - Vesicle distance based hierarchical clustering
  - Vesicle clusters dual to hierarchical connector clusters, that is vesicle clusters comprising those vesicles that contact connectors within hierarchical clusters
- <root>_cluster_conn.dat: Resulting clusters for different clusterings of connectors
  - Connector clusters by connectivity (via vesicles)
  - Connector distance based hierarchical clustering
  - Connector clusters dual to hierarchical vesicle clusters, that is connector clusters comprising those connectors that contact vesicles within hierarchical clusters
- <root>_conn_cluster.dat
- <root>_cluster.pkl

In addition, two other pickle files are generated:

- <root>_bound_distances.plk: distance between all pairs of boundaries (vesicles)
- <root>_conn_distances.pkl: distance between all pairs of connectors

These are meant to be reused when this script is executed multiple times (with different hierarchical clustering parameters, for example), because calculating distances is the most computationally demanding part her. Of course, if vesicles of connectors are changed, the previously calculated distances should not be used.

3 Statistical analysis of multiple datasets

3.1 General info

A more detailed description of the relevant computational methods is given in analysis.pdf .

All relevant data is in examples/presynaptic_example/analysis/ .

3.2 Prerequisite

In this part, results from the segmentation and analysis of individual datasets (above) are used. In order to mimic the real case where multiple datasets are analyzed, the result files obtained by the segmentation and analysis of individual datasets (previous section, files under segmentation/cell_tomo-4/) that are relevant for further processing are simply copied to parallel directories (cell_tomo-2, -6 and -7).

3.3 Making catalogs

Each experiment has to have a corresponding catalog file that contains the location of result files generated by the segmentation and analysis of individual datasets (above).

All catalog files should reside in analysis/catalogs/.

3.3.1 Steps

- Use files from catalog/directory. They are already set for experimental groups (treatments) 'ctrl' and 'treated', and for experiment identifiers 'ctrl_1', 'ctrl_5', 'treated_2' and 'treated_5' (correspond to datasets cell_tomo-4, cell_tomo-7, cell_tomo-2 and cell_tomo-6, in this order). Catalog files are copied from pyto/scripts/presynaptic_catalog.py and have all parameters already set to the values appropriate for this example. These include pixel size, file paths to single dataset result files, and some other properties.

3.3.2 Bach generation of catalog files

Bach generation of catalog files is more complicated, but may save time when a large number of catalogs need to be created.

- Start from the catalog file for one experiment (dataset cell_tomo-4, experiment 'ctrl_1')
- Execute script analysis/work/make_catalog-2.py to create catalog file treated_2.py (for experiment 'treated_2')
- Modify and execute analysis/work/make_catalog-2.py to generate other catalog files. The required modifications can be deduced from the comparison of catalogs ctrl_1.py, treated_2.py and the provided analysis/work/make_catalog-2.py. All modifications are in the parameter section.

3.4 Making structure-specific pickles

In this task, the results generated by all segmentation and analysis tasks of all individual datasets, as well as the additional dataset-specific information specified in catalogs, are combined to make a set of structure-specific pickles.

3.4.1 Steps

- Use the provided analysis/work/work.py (copied and edited from pyto/scripts/presynaptic_stats.py)
- All parameters are set already. Specifically, variables identifiers, categories and references were modified to correspond to this example. Other variables are not relevant for this task, but are used at a later point (see below)
- Note: the highest bin limit value in variable layer_distance_bins was changed from the default (250 nm) to 160 nm because the dataset used for this example is smaller in size than the standard synaptic tomograms.
- Execute the following python commands (in ipython, jupyter or related) in analysis/work/ :

```
> import work
> from work import *
> work.main(individual=True, save=True)
```

3.4.2 Results

- The resulting structure specific pickle files are saved in the same directory.
- Their names are 'sv.pkl', 'tether.pkl', 'conn.pkl','layer.pkl' and 'cluster.pkl', unless they were changed by modifying the corresponding variables in work.py script

3.5 Statistical analysis

Here are instruction for viewing data from all experiments (saved in the structure specific pickles), presenting data in graphs and calculating statistical significance between the experimental groups.

In order to better present the functionality, structure specific pickles from another (real) project are used for in this task.

Files used here are under analysis_big/:

- analysis_big/catalog/: Catalog files
- analysis_big/work/: Structure specific pickles, presynaptic analysis file (work.py) and the analysis notebook (work.ipynb)

3.5.1 Steps and results

All steps and results (graphs and tables) are shown in the presynaptic analysis notebook (work.ipynb). In short, these consist of:

1. Read and preprocess the data from the structure specific pickles
2. Plot and statistically compare between group properties such as: vesicle location, size and greyscale, connector morphology, localization and the relationship to vesicles, and clustering of vesicles and connectors.