Workflows

Vladan Lucic

February 7, 2021

Max Planck Institute of Biochemistry
Email: vladan@biochem.mpg.de

$Revision: 1690 $

1 Summary

This document contains specific instructions and parameters for the hierarchical connectivity segmentation, segment analysis of individual experiments (datasets) and the statistical analysis of multiple experiments. A more general description of concepts and procedures, as well as some details that are omitted here are covered in the Segmentation and the segment analysis guide and the Analysis of multiple datasets guide.

2 Presynaptic

2.1 Purpose

To segment and analyze complexes interlinking vesicles (connectors) and linking vesicles to the active zone membrane (tethers). Synaptic vesicles and the active zone membrane constitute boundaries.

One synapse constitutes one experiment (dataset). Most often there is one synapse per tomogram, but it is possible to have more. The main parts of this procedure are:

- Detection of tethers and connectors by segmentation and analysis of individual datasets
- Statistical analysis of multiple datasets

2.2 Presynaptic example

An example of the complete presynaptic workflow is given in Pyto/examples/presynaptic_example/ directory and the steps are explained in presynaptic_example.pdf.

2.3 File organization

2.3.1 Segmentation and analysis of individual datasets

- Each experiment (presynapse) should have a separate directory. Typically, all experiment directories are located within one directory (for example called 'tomo')
- Individual experiment directories should have clear, meaningful names, ideally these should be the same as the unique experiment identifiers (see below) such as 'ctrl_1', 'ctrl_2', 'snc_tko_1', 'kcl_5'. Each of these should contain the following subdirectories:
  - 3d/, viz/ (or something like this): Tomograms and boundary label files
  - common/: contains tomo_info.py
  - One directory for each task: regions, vesicles, layers, connectors and clustering

2.3.2 Statistical analysis of multiple datasets

- Analysis directory (called 'analysis') should be next to tomo. It should contain the following subdirectories :
  - catalogs: catalog files, one for each tomo / synapse, their names should be tomo identifiers
  - work: statistical analysis between experimental groups

2.3.3 Common

- All script files used here are located in pyto/scripts directory. They need to be copied to their final destination and edited, as explained below.

2.4 Prerequisites

2.4.1 Tomogram

Contains one or more biological objects of interest, should be in mrc or em format.

2.4.2 Boundary label file

This is a label file that defines all labels (also called material or region). It is an image file with integer datatype where the positioning and size of exactly corresponds to the tomogram. It can be in mrc, em or raw format (no header, only data array).

This file has to contain distinct labels for each of the following:

- active zone membrane
- synaptic vesicles (each vesicle should have a unique label)
- cytoplasm (segmentation region).

The active zone membrane and synaptic vesicles together constitute boundaries.

The segmentation region has to make a direct contact with the boundaries, because the segmentation starts from the segmentation region voxels that directly contact the boundary. Also, the segmentation region should exclude the space where complexes of interest can not be located (because the space is occupied by another organelle, for example).

It is important that each vesicle has a unique label id and that no other voxels are labeled by the same vesicle label id. Other organelles and similar structures may also be labeled.

2.5 Segmentation and analysis of individual datasets (synapses)

The following tasks comprise segmentation, the segment analysis and other types of analysis that are performed on individual datasets. It is recommended that they are executed in the order they are listed below.

2.5.1 General notes

- All tasks import common parameters from tomo_info.py. These common parameters should not be explicitly defined in the scripts (but imported from tomo_info.py).
- Parameters that specify image shape, data type, byte order and array order do not need to be defined for images in mrc or em format because they are determined from their headers. However, if these parameters are specified in scripts, they will override the header values.
- Parameters that are not mentioned may be left at the default values.
- It is preferable that the same scripts (having the same parameters) are used for all datasets.
- It is important to note that pixel size is not used at all for the segmentation and analysis of individual datasets. It is introduced only in the analysis of multiple datasets.

2.5.2 Set common parameters

In order to avoid entering same parameters in multiple scripts, the common parameters are entered in common/tomo_info.py.

- `image_file_name`: tomogram file name
- `labels_file_name`: boundary label file
- `labels_data_type`: data type of the boundary label file (should be set to None for mrc and em formats)
- `all_ids`: ids of all labels in the label file that should be considered
- `boundary_ids`: ids of of all boundaries
- `vesicle_ids`: ids of all vesicles
- `segmentation_region`: id of the segmentation region

2.5.3 Labeling spherical vesicles an a semi-automated way (optional)

To label spherical vesicles, it is sufficient to segment the equatorial planes using a manual labeling software (such as Amira). executing script disc_to_balls.py expands the equatorial planes (termed discs) to spheres (balls). The parameter to be set is:

- `discs_file_name`: discs (equatorial planes) file name

In order to avoid manually segmenting multiple equatorial planes of the same vesicle, it may be useful to mark the already labeled vesicles using the blank.py script, by repeating the following cycle until all vesicles are labeled (see the presynaptic example) :

- Label some equatorial planes
- Convert the labeled discs to balls (disc_to_balls.py script)
- Blank the balls (labeled vesicles) in the tomogram (blank.py script)

2.5.4 Regions script

Execution of this scripts calculates the basic greyscale statistics and morphological characteristics of the structures defined in the boundary labels file (includes boundaries and the segmentation region). Strictly speaking it is not necessary, but it is recommended to run this script, because the results are useful to set the segmentation threshold range, and for troubleshooting.

1. Copy script pyto/scripts/regions.py to regions/regions.py. Typically, no parameters need to be changed from the default values.
2. Run regions/regions.py script

Results:

- _regions.dat: human readable file containing all results
- .pkl: pickle file containing all results

It is recommended to visually inspect the results for obvious problems.

2.5.5 Vesicles script

This script (found in pyto/scripts/vesicles.py) is used to determine basic greyscale, morphology and localization info for vesicles. Provides info similar similar to the regions script.

1. Copy script pyto/scripts/vesicles.py to vesicles/vesicles.py script
2. Set `distance_id`: typically label id of the active zone membrane
3. Set `membrane_thick`: membrane thickness in pixels, used to separate vesicle membrane from lumen
4. Check is some of the vesicles are too large or too small for synaptic vesicles (40 nm diameter). Remove those from vesicle_ids and boundary_ids (tomo_info.py)
5. Execute vesicles/vesicles.py script

The results are saved in:

- _vesicles.dat: human readable file containing all results
- _vesicles.pkl: pickle file containing results for complete vesicles (membrane and lumen together)
- _mem.pkl: pickle file containing results for vesicle membranes
- _lum.pkl: pickle file containing results for vesicle lumens

It is recommended to visually inspect the results for obvious problems.

2.5.6 Layers script

Makes thin layers, that is 1 pixel thick regions parallel to the active zone membrane. The layers are made on on the cytoplasm (segmentation region) and over vesicles. More generally, they can be defined in respect to any other boundary.

1. Copy pyto/scripts/layers.py to layers/layers.py
2. Set `boundary_id_1` to the active zone membrane id, `boundary_id_2` to None, `layers_mask_id` to a list containing the active zone membrane label id
3. `layer_thickness = 1` (pixel), number of layers (num_layers) should be such that all layers span at least 250 nm.
4. `num_extra_layers = 0` and other parameters do not play any role in this case
5. Execute layers/layers.py script

The execution of this script creates the following output files:

- .dat: human readable file containing results
- .mrc: layers image file

It is recommended to visually inspect the layers image as well as the results for obvious problems.

2.5.7 Connectors script

This is the main segmentation and segment analysis script (found in pyto/scripts/connectors.py). The following parameters have to be defined:

1. Copy pyto/scripts/connectors.py to connectors/connectors.py

2. Threshold range (`threshold`) should be chosen as follows: There should be no connectors (2-bound segments) at the lowest threshold. The highest threshold should be little lower than the highest threshold where connectors are formed. This can be adjusted by finding the lowest greyscale level where there's only noise (by visual inspection of your tomo and by using the segmentation region greyscale statistics). The number of threshold steps should be at least 20 (counting only those at which connectors are detected). Threshold range should be determined for each tomo, but the method to determine the range has to be the same for all tomos.

3. Boundary contact conditions: `n_boundary = 2`, `count_mode = 'exact'`

4. Set `distance_id`: typically label id of the active zone membrane

5. The first classification (classification 1 in the script) should be used to obtain a proper segmentation (to cut the hierarchical tree). It is necessary to do this classification to obtain a proper segmentation, that is to eliminate overlapping segments. The default value (`class_1_mode = 'new'`) means that the segments are detected at the lowest threshold at which they appear.

6. The values of other parameters can be left at the default values. Depending on the application, changing some might improve the results.

7. Execute connectors/connectors.py script

The execution of this script creates the following output files:

- *_thr-*.dat, *_thr-*.pkl, *_thr-*.mrc: Segmentation results in a human readable (.dat) and pickle (.pkl) formats, as well as the segmetnation image (.mrc)
- Files not containing 'thr' nor 'new' (.dat and .pkl): Results from segmentations at all thresholds together
- *_new*: Final segmentation

It is recommended to visually inspect the output segmentation images as well as the results (.dat) files to check for obvious problems.

2.5.8 Further classification by the connectors script (optional)

In addition to the segmentation and analysis, the segmentation script can be used to classify the detected segments. A single run of the segmentation script can perform segmentation, analysis and classification, in this order (the default way). The following parameters need to be set:

1. The second classification (classification 2 in the segmentation script) separates tethers (link vesicles to the active zone membrane) and connectors (interconnect vesicles), `class_2_ids` needs to be a list that contains the active zone membrane id

2. It is recommended to use classification by volume. However, the exact values depend on the pixel size. For example, the upper limit was set to 150 for 2.73 nm and to 600 for 1.37 nm pixel size.

Alternatively these classifications can be done by after the initial segmentation. In general, segmented (and possibly classified) connectors can be further classified by the connectors script

1. The pickle file containing the segmentation (and analysisis) needs to be specified (parameter `segmentation_file_name`)

2. To avoid redoing segmentation: `do_segmentation_flag = False` and `segmentation_file_name` is set to a segmentation pickle obtained in the previous step.

3. Classifications that were done already should not be repeated

4. Other classification(s) can be added or commented out

2.5.9 Clustering script (optional), defines the following clusterings

1. Connectivity-based clustering of vesicles: A vesicle cluster is defined as subset all vesicles that are mutually interconnected by connectors
2. Hierarchical spatial distance based clustering of vesicles. Various linkages and cutting methods are defined. These are also compared with the connectivity based clusters.
3. Hierarchical spatial distance based clustering of connectors. Various linkages and cutting methods are defined.

2.6 Statistical analysis of multiple datasets

2.6.1 Make catalog files

There has to be a catalog file for each experiment. All catalog files have to reside in the same directory (such as analysis/catalog/).

1. Chose a unique identifier for each experiment. For example, use the experimental condition and a number: 'wt_1', 'wt_ 2', 'protein-x_ko_1', 'untreated_1', 'compound-y_1', 'compound-y_4', ... This can be completely unrelated to the corresponding segmentation directory.
2. Chose experimental condition (treatment) names. Following the identifiers given above, the names could be 'wt, 'protein-x_ko', 'untreated', 'compound-y'.
3. For each experiment, copy pyto/scripts/presynaptic_catalog.py to categories/ . It is recommended that the catalog file name is the identifier with .py suffix.
4. In each catalog file, set the following variables:
   - `identifier`: unique experiment identifier
   - `treatment`: specifies experimental condition
   - file paths of dataset specific pickles; these can be absolute or relative to the catalogs/ directory. It is strongly recommended to use the default variable names (those specified in pyto/scripts/presynaptic_catalog.py: `tethers_file, connectors_file, sv_file, sv_membrane_file, sv_lumen_file, layers_file, clusters_file.`
   - `pixel_size`: pixel size in nm
   - Other properties can be added to the catalog file. It is strongly recommended that all catalog files pertaining to one project have the same variables.
5. Alternatively, it might be easier to generate catalog files by a batch procedure (see the presynaptic example)

Note: Catalog files are used for both generation of structure specific pickles and for statistical analysis.

2.6.2 Set parameters in the presynaptic analysis script (part 1/2)

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.

1. Copy pyto/scripts/presynaptic_stats.py script to the work directory (typically analysis/work/work.py) and set the following variables
2. `categories:` names of all experimental conditions (property treatment) that should be considered.
3. `identifiers:`identifiers of all experiments (property identifier) that should be considered.
4. `reference`: dictionary with each key category is statistically compared with the corresponding value category, so all experimental groups have to be among keys and values are typically experimental groups like 'control', 'untreated', or 'WT'.

5. Set `catalog_directory` and `catalog_pattern` variables to define the catalog files (default values are fine for the recommended directory organization)

6. Names of properties (catalog variables) need to be modified only if non-default values were used to make catalog files. The variable names are `tethers_name`, `connectors_name`, `sv_name`, `sv_membrane_name`, `sv_lumen_name`, `layers_name`, and `clusters_name`.

7. Names of the structure specific pickles generated by this task are defined by variables: `sv_pkl`, `tethers_pkl`, `connectors_pkl`, `layers_pkl` and `cluster_pkl`. Usually, the default values are retained.

8. Other variables are not relevant for this task, so the default values should be kept. However, some of them are modified at a later point (see below)

Note: Presynaptic analysis script is used for both generation of structure specific pickles and for statistical analysis.

2.6.3 Generate structure specific pickles

To generate the structure specific pickles, run presynaptic_stats.py script from the Python prompt (such as IPython or Jupyter notebook), in the same directory where the presynaptic_stats.py script is located:

```
>>> import presynaptic_stats.py
>>> presynaptic_stats.main(individual=True, save=True)
```

Each structure specific pickle obtained in this way contains the segment analysis data from all datasets. These pickles are specific for the following structures:

- vesicles: vesicles are defined in the boundary labels file
- tethers: subset of all segments that contact one vesicle and the active zone membrane
- connectors: subset of all segments that interconnect two vesicles
- layers: regions containing cytoplasm and vesicles at specific distances (parallel to) the active zone membrane
- clusters: all three clustering as defined previously

The data contained in the structure specific pickles is also preprocessed. That is, in addition to properties calculated by the segmentation and analysis of the individual datasets, other properties are also calculated. These include:

- properties that are in pixel units (length, various distances, surface, volume) are converted to nanometers
- properties that combine results from different analyses (for example, number of connectors per vesicle)
- some vesicles can be excluded based on their size, tethers and connectors contacting these vesicles are also excluded

The time needed to generate the structure specific pickles depends on the (individual analysis pickle) file access and transfer time.

2.6.4 Set parameters in the presynaptic analysis script (part 2/2)

Some of the variables in the presynaptic analysis script (work.py) that were not modified to generate structure specific pickles (part 1), need to be modified to do the statistical analysis:

1. `reference`: This variable was already set (part 1, above), strictly speaking it is only needed to perform statistical analysis.

2. `vesicle_radius_bins`: The middle bin sets the range of radii that define synaptic vesicles.

3. `distance_bins`: Define the zones within the presynaptic terminal based on the distance to the active zone membrane.

4. Other variables in section parameters: These are convenience shortcuts, may be changed if needed.

5. All dict-type variables where keys are experimental group names (`color`, `category_label`, ...) need to be set to the current experimental groups.

6. Other variables in Printing and Plotting sections concern formatting and plotting of results, they can be changed if needed.

2.6.5 Perform statistical analysis

The same presynaptic_stats.py script that was used in the previous step to make the structure specific pickles should be used for this step. The structure specific pickles are loaded and the data is further processed. This mainly separates vesicles and the associated connectors and tethers in different groups, depending on the vesicle distance to the active zone membrane and the existence of connectors and tethers. After that statistical comparison of different properties between the experimental groups can be explored.

All this can be separated into the following steps:

- One may modify (remove some elements of) variables `categories`, `identifiers` and `reference`, as well as the printing and plotting parameters
- printing and plotting related variables are self explanatory, the default values are usually appropriate, except that the default categories need to be replaced by the actual ones
- Initialize from the Python prompt (such as IPython or Jupyter notebook):

```
>>> import presynaptic_stats.py
>>> from presynaptic_stats import *
>>> presynaptic_stats.main()
```

- Execute predefined functions that calculate statistics between experimental groups and display the results. These are located at the end of presynaptic_stats.py script in sections SV distribution and below .from the same Python environment as the one used for the previous step

It is often convenient to use Jupyter for the previous steps, because the results are then saved in the Jupyter notebook and can be inspected at a later time without re-executing the statistical commands.

3 Single membrane attached complexes

3.1 Purpose

To segment and analyze complexes (biological material) bound to a boundary.

A boundary can be a lipid membrane, part of an organelle or an arbitrary surface created by the user. Generally, a boundary together with the attached complexes constitute one experiment (dataset), so one tomogram can contain several experiments. It is also possible to consider multiple boundaries and attached complexes as one experiment.

The main parts of this procedure are:

- Segmentation and analysis of individual datasets
- Statistical analysis of multiple datasets

3.2 File organization

3.2.1 Segmentation and analysis of individual datasets

- Each tomogram should have a separate directory with experiment subdirectories. Typically, all tomogram directories are located within one directory (for example called 'tomo')
- Each tomogram should have a directory where the tomogram and the boundary label file(s) are located ('3d', 'visualization', or similar). There can be one boundary file for all experiments of a tomogram, or one boundary file for each experiment
- Individual experiment should have clear, meaningful names (such as such as 'ctrl_1', 'ctrl_2', 'snc_tko_1', 'kcl_5'). These names should be also used for the unique experiment identifiers for statistical analysis.
- Each should have the following subdirectories:
  - common: contains tomo_info.py
  - One directory for each task: regions, layers and connectors

3.2.2 Statistical analysis of multiple datasets

- Statistical analysis directory (called 'analysis') should be next to tomo. It should contain the following subdirectories, these should be created after the segmentation and analysis of individual tomograms:
  - catalogs: catalog files, one for each tomo / synapse, their names should be tomo identifiers
  - work: statistical analysis between experimental groups

3.2.3 Common

- All script files used here are located in pyto/scripts directory. They need to be copied to their final destination and edited, as explained below.

3.3 Prerequisites

3.3.1 Tomogram

Contains one or more biological objects of interest, should be in mrc or em format.

3.3.2 Boundary label file

This is a label file that defines all labels (also called material or region). It is an image file with integer datatype where the positioning and size of exactly corresponds to the tomogram. It can be in mrc, em or raw format (no header, only data array).

This file has to contain distinct labels for each of the following:

- one or more boundaries
- region where complexes are located (segmentation region), one for each boundary

The segmentation region has to make a direct contact with the boundary, because the segmentation starts from the segmentation region voxels that directly contact the boundary. Also, the segmentation region should exclude the space where complexes of interest can not be located (because the space is occupied by another organelle, for example).

It is important that each boundary and each segmentation region have a unique label id and that no other voxels are labeled by the same vesicle label id. Other organelles and similar structures may also be labeled if needed for other purposes.

3.4 Segmentation and analysis of individual datasets (synapses)

3.4.1 Set common parameters

In order to avoid entering same parameters in multiple scripts, the common parameters are entered in common/tomo_info.py.

- `image_file_name`: tomogram file name
- `labels_file_name`: boundary label file
- `labels_data_type`: data type of the boundary label file (should be set to None for mrc and em formats)
- `all_ids`: ids of all labels in the label file that should be considered
- `boundary_ids`: id of the boundary
- `vesicle_ids = []`
- `segmentation_region`: id of the segmentation region

3.4.2 Regions script

This script (found in pyto/scripts/regions.py) calculates the basic greyscale statistics and morphological characteristics of the structures defined in the boundary labels file (includes boundaries and the segmentation region). Strictly speaking it is not necessary, but it is recommended to run this script, because the results are useful to set the segmentation threshold range, and for troubleshooting.

The results are saved in:

- _regions.dat: human readable file containing all results
- .pkl: pickle file containing all results

It is recommended to visually inspect the results for obvious problems.

3.4.3 Vesicles script

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

3.4.4 Connectors script

This is the main segmentation and segment analysis script (found in pyto/scripts/connectors.py). The following parameters have to be defined:

1. Setting the threshold range (`threshold`) depends on the application, here is a procedure that might work:
   (a) Initial assignment: The lowest threshold can be set so that there is no or just a few segments detected. The upper limit can be set to the value that roughly separates complexes of interest and the noise (empty cytoplasm) in the tomogram, using the results of the regions script (in particular the segmentation region greyscale statistics) might help. There should be around 20 thresholds between the lower and the upper limit.
   (b) The segmentations obtained by running the script with the initial threshold range should be checked at each threshold level to determine the optimal threshold.
   (c) The working threshold range should include few levels below and above the optimal threshold.
   (d) The range can be adjusted by trial and error, that is by visual inspection of segments detected by this script. It is important to chose the threshold range in the same manner for all tomograms.
2. Boundary contact conditions: `n_boundary = 1`, `count_mode = 'exact'`

3. Classification: keep classification 1, comment out all other. If some segments are too large, the volume classification can be imposed, or the upper threshold limit should be reduced.
4. `length_contact_mode = 'b-max'`
5. `length_line_mode = 'straight'`
6. Set `distance_id`: label id of the boundary

The values of other parameters can be left at the default values. Depending on the application, changing some might improve the results.

The execution of this script creates the following output files:

- *_thr-*.dat, *_thr-*.pkl, *_thr-*.mrc: Segmentation results in a human readable (.dat) and pickle (.pkl) formats, as well as the segmetnation image (.mrc)
- Files not containing 'thr' nor 'new' (.dat and .pkl): Results from segmentations at all thresholds together
- *_new*: Final segmentation

It is recommended to visually inspect the segmentation images as well as the results (.dat) files to check for obvious problems.

3.4.5 Layers

Greyscale analysis of thin layers parallel to the boundary, on the segmentation region (optional).

- boundary_id_1:
- boundarry_id_2 = None
- num_layers: number of 1-pixel thick layers formed
- segments_file: name of the final segmentation pickle file generated by the connectors script.

The execution of this script creates the following output files:

- .dat: human readable file containing results
- .mrc: layers image file

It is recommended to visually inspect the layers image as well as the results for obvious problems.

3.5 Statistical analysis of multiple datasets

- Like for the presynaptic workflow

4 Troubleshooting

- Problems with boundary labels have been the most common reason for errors so far. These include incorrect creation of boundaries, such as having more than one vesicle labeled by the same label id, or some additional voxels being labeled by an id used for a vesicle. Entering wrong label ids in scripts also falls in this category.
- If program crashes, please check whether warning messages were generated at any point before the crash.