Quick Guide to Segmentation and Segment Analysis by Pyto

Vladan Lucic

February 7, 2021

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

$Revision: 1688 $

1 Summary

This guide explains the usage and some basic concepts of the hierarchical connectivity segmentation, as well as segment analysis and classification implemented in Pyto package. These procedures are applied to individual datasets or biological object of interests.

2 Introduction

The segmentation and analysis procedure requires a tomogram of a biological object of interest, such as the presynaptic terminal, synaptic cleft, any other part of a cell, or an isolated/reconstituted system. In case of synapses, one tomogram usually contains one biological object of interest (synapse), but in other cases one tomogram can contain multiple objects of interest.

In addition to a tomogram of an object of interest, a label file that contains boundaries is required. The boundaries can be any distinct cellular regions, such as membranous structures, plasma membrane or organelles. The segmentation procedure finds segments that contact these boundaries based on the density values given in the corresponding greyscale tomogram in the region that surrounds and/or is located between the boundaries (called the segmentation region).

2.1 Terminology

- Dataset: a tomogram and a label file corresponding to one biological object of interest
- Experiment: one dataset together with the results obtained by the analysis of that dataset.

2.2 Segmentation

The segmentation procedure detects structures that contact one or more of the specified boundaries. This procedure is described in Section 6 of this guide.

Segments obtained in this way (termed connectors) are analyzed to obtain a quantitative description of their properties. These include greyscale, morphological and topological properties, as well as the information about their location. Segments may also be classified and separated in appropriate classes. Again, this is done at each dataset (biological object of interest) separately. These tasks are explained sections 8 and 7 of this guide.

The segmentation and the analysis of the segments detected is performed by a single script (segmentation and analysis script). Other scripts can also be used to support this analysis, as is the case for the presynaptic terminal (see the workflows document).

Statistical analysis of segment properties obtained from different datasets, such as between different experimental groups, is described in another document (analysis.pdf).

3 Files and usage

3.1 Directory organization

It is advisable, but not necessary, to organize directories so that each tomogram has a separate directory. Within each of these directories there should be one (or two) where tomogram, boundary segmentation and other files are located and additional directories devoted to pyto procedures: region analysis, segmentation and other. Note that pyto segmentation generates a fair number of files (easily 50, depends on the specified parameters), so a separate directory is recommended.

3.2 Greyscale image files (3D images or tomograms)

These files contain EM density and are used as input files to segmentation. They can be in MRC or EM format, the format is recognized by the extension ('mrc' or 'em'). The information about the data (data type, image size, byte order and array order) is read from the header. For synaptic structures, one tomogram typically contain one synapse, but it can also contain more than one synapse or other biological object or interest.

3.3 Label files

Label (or segmentation) files contain any kind of segments or regions. Segments (elements of the foreground) are identified by positive integers (ids) where each segment is assigned a unique id, while the non-segmented parts (background) should be labeled by 0. Because these files typically correspond to tomograms, any label file should have the same size and positioning as the corresponding tomogram. Boundary files and segmented image files are two basic types of label files.

3.3.1 Boundary files

Boundary files are used as input files for segmentation. A boundary file has to define boundaries, but may also define segmentation regions (region outside the boundaries where segmentation is allowed to occur) as well as other structures.

These files are often generated using manual or semi-automated segmentation in Amira, in which case the label field should be saved as a raw file (and not as an am file). However, other programs can be used to generate boundary files, they only need to conform to the above requirements. If a raw file is used, its properties: size (in voxels), data type, byte order and array order have to be specified. Alternatively, if a boundary file is in EM or MRC format, these values are read form the header.

3.3.2 Segmented image files

These files contain the segments obtained by pyto segmentation. The segments are labeled by positive integers (segment ids). They can be written in EM or MRC formats, depending on the extension specified. Typically, if no segments are found these files are not written. These files are useful to visually check the segmentation procedure.

3.4 Other output files

3.4.1 Result files

These files contain information about segments obtained by pyto segmentation as well as some of the parameters, in the plain text form. They are written even if no segments are generated. These files provide a convenient way for a user to see the results. Each result file has a corresponding segmentation file (except when a segmentation file is not written because is doesn't have any segments) and a corresponding segmentation pickle file.

3.4.2 Segmentation pickle files

These are Python pickle files that contain the segmentation instance (object). Consequently, it contain most if not all parameters, as well as the segmentation results in a compressed, non-human readable form. However, these files are used for further processing and analysis and they constitute something like a distributed object-oriented database.

4 Usage

4.1 Basic usage

The appropriate segmentation script(s) needs to be copied to a pyto segmentation directory and edited. Variables defined in the Parameters section of the segmentation scripts may and should be modified. They all come with a short usage explanation, while this guide is meant to explain concepts behind them. Parameters labeled as experimental, as well as other parts of the script are not recommended to be modified unless the user understands very well what he/she is doing.

The segmentation is run by executing the (modified) segmentation scripts. This can be done from a command line, from a tomogram specific, pyto segmentation directory:

```
> python segmentation_script.py
```

or from IPython (recommended), which is useful for executing only selected parts or debugging, among other things:

```
> ipython
In [1]:  run segmentation_script.py
```

During the run info messages are printed showing important execution steps.

4.2 Advanced usage

The segmentation and analysis procedures can be modified by customizing the segmentation script. Alternatively, these procedures can be performed by using pyto clases directly, within a Python shell. This approach requires a solid understanding of Python, Numpy and Scipy packages as well as pyto.

4.3 Steps

The segmentation and analysis procedure described in this document consists of the following steps:

- Creation and analysis of boundaries (optional, see section 5)
- Segmentation (see section 6)
- Segment analysis (see section 8)

5 Creation and analysis of boundaries

5.1 Making boundary label files

Boundary label files define boundaries, but they may also define other regions such as the segmentation region (region outside the boundaries where segmentation is allowed to occur). They are typically generated by manual segmentation that involves tracing boundary contours in all z-slices, using one of the specialized software packages (such as Amira or Imod), but they can be also generated using automated tools.

Sometimes, not all boundaries can be saved in one file. For example, that can happen because in Amira the number of boundaries that can be saved in one file is limited. In such a case, more than one boundary file is generated for one biological object of interest. It is advisable to combine all these boundary files into one (see script add_segments.py). While multiple boundary files can be used for some of the segmentation scripts, their use is depreciated.

In case of spherical boundaries (such as synaptic vesicles) script discs_to_balls.py can be used to help this procedure. Namely, it is sufficient to manually delineate only the equatorial z-slice of each spherical boundary. Script discs_to_balls.py takes a label file produced in this way as input and generates the boundary labels file where each equatorial outline of a spherical object is replaced by a sphere.

Each boundary should be labeled by a unique id (positive integer) and the user needs to know what these ids are. Note that in general this id can not be inferred form "Material N" shown by Amira. The user needs to determine the ids by visualizing the label field in a program that shows individual pixel values.

Important: Voxels at sides of the boundary label files should not belong to any regions (boundaries or segmentation region), that is they should have 0 values.

5.2 Analysis

Script regions.py provides basic greyscale density statistics (mean, std, min and max) and morphological (volume, surface, surface to volume ratio) analysis of any kind of boundaries and segments. For the analysis of spherical boundaries, such as synaptic vesicles, script vesicles.py provides additional functionality.

The following parameters are used for the analysis:

- Boundary ids (variable name `region_ids` in regions.py, or variable_ids in vesicles.py): Defines boundaries (regions or vesicles) that are used to determine contacts between boundaries and segments (connectors) generated by segmentation. These ids should be the same as boundary ids specified in connection script.
- All ids (variable name `all_ids`): All regions (segments) defined in boundary labels file. These have to include boundary ids, as well as all other regions defined in the boundary file.
- If membrane thickness (variable `membrane_thick`) is defined, it is assumed that each region is covered by a membrane (as synaptic vesicles are). In that case, each region is separated into membrane and lumen and the same properties are calculated for membrane and lumen of each segment.
- Variable `distance_id` specifies a region to which distance from each region is calculated. This distance can be calculated as the minimal distance, mean (mean of distances of all voxels of a region to the distance region) or center (from the center of each region), depending on variable `distance_mode`.
- If vesicles.py script is used, the radius and center location is also calculated.

6 Segmentation procedure

Segmentation script connectors.py is suitable for segmentation of connectors between multiple boundaries. A specific example is segmentation of connectors and tethers formed between synaptic vescles and the plasma membrane at the presynaptic terminal, but it can be used in a more general case.

6.1 Connectivity segmentation at a single threshold

We first used a straightforward connectivity-based segmentation method at a single threshold. This method requires specifying a boundary contact condition, that is the number of boundaries that each segment has to contact, and a greyscale threshold value. Any number of boundaries to contact can be specified, but here we focus on segments that directly contact exactly two (2-bound segments) or more than two boundaries (>2-bound segments).

The segmentation consists of three steps, all of them carried in 3D (Algorithm 1, steps 1a-c for a single threshold). First, voxels having greyscale value below (darker then) the given threshold are selected. Each connected cluster of the selected voxels forms a segment, while segments do not contact each other (see Methods). Finally, only those segments are retained that contact the required number of boundaries (the boundaries and the number of boundaries to contact need to be specified).

Examples of this procedure, for three different thresholds are shown on Figure 1A, top row. Clearly, the results depend on the threshold chosen, so the ability to chose the optimal threshold is crucial. However, finding the optimal threshold for noisy images is difficult and it may be highly subjective. Furthermore, in

Algorithm 1 Hierarchical connectivity segmentation and analysis, Step 1, executed for a single threshold shows the single threshold connectivity segmentation procedure. Steps 1-3 constitute the hierarchical connectivity segmentation procedure. Steps 4 and 5 constitute segment analysis and are needed for further processing.

1. Single threshold connectivity segmentation. The following steps are repeated at different thresholds.
   (a) Threshold at a given value, select voxels that are below or equal the threshold.
   (b) Separate selected voxels into mutually disconnected segments.
   (c) Select only the segments that satisfy specified boundary contact condition.
2. Merging of all single threshold segmentations to obtain a hierarchical segment tree.
3. Segment selection (cutting the segment tree) based on their position on the tree.
4. Determination of segment properties.
5. Segment selection or classification based on their properties.

case of variable image background, or for heterogeneous complexes, the optimal threshold value may be impossible to define. As an alternative strategy, it is in some cases possible to define segment properties whose values do not change appreciably within the whole range of acceptable threshold values, thus circumventing the problem of finding the optimal threshold. Nevertheless, the applicability of the single threshold segmentation is limited, thus showing the necessity of using multiple thresholds.

6.2 Hierarchical connectivity segmentation

The hierarchy connectivity segmentation extends the single threshold connectivity segmentation to multiple thresholds. The complete procedure is shown in Algorithm 1, steps 1 (for multiple thresholds), 2 and 3. Namely, the single threshold connectivity segmentation is performed at each of the specified set of threshold values. It is clear that every segment obtained at any threshold (except at the highest one) is entirely contained in a segment obtained at next higher threshold. For example, on Figure 1A top row, segment 2 obtained at the low threshold is contained in segments 5 (middle) and 7 (high threshold), while segment 4 (middle) is contained in segment 7 (high threshold). On the other hand, every segment either completely overlaps with one or more segments obtained at any lower threshold (segments 6 and 5), or it has no overlap with any lower threshold segment (segment 4). Therefore, segments can be arranged as nodes of a hierarchical segments tree, where tree levels correspond to thresholds (Figure 1A bottom row, middle). In this way, the overlapping segments are easily identified as ancestors / descendants of each other.

Segments obtained at multiple thresholds are presented in a compact form, comprising of a hierarchical segments tree (Figure 1A bottom middle) and one image where each voxel is labeled by the lowest threshold segment that contains the voxel (Figure 1A bottom left). However, segments provided in this way do not constitute a proper segmentation because segments from different thresholds overlap, resulting in voxels belonging to multiple segments.

In order to obtain a proper segmentation, the segment tree needs to be cut, that is a non-overlapping subset of segments is selected so that the selected segments do not have ancestor - descendant relation. There are different ways to do that:

- In case an optimal threshold can be found, only the segments identified at that threshold are retained. This can be achieved by simply assigning only one threshold value (the results of this classification are the same as the segmentations shown in Figure 1A top row).
- When the main goals is to detect individual segments, and it is not important that their shape exactly matches the biological material, retained are only the segments that appear for the first time in the hierarchy (thresholds are ordered in the increasing fashion), also called new or leafs (Figure 1A bottom right).

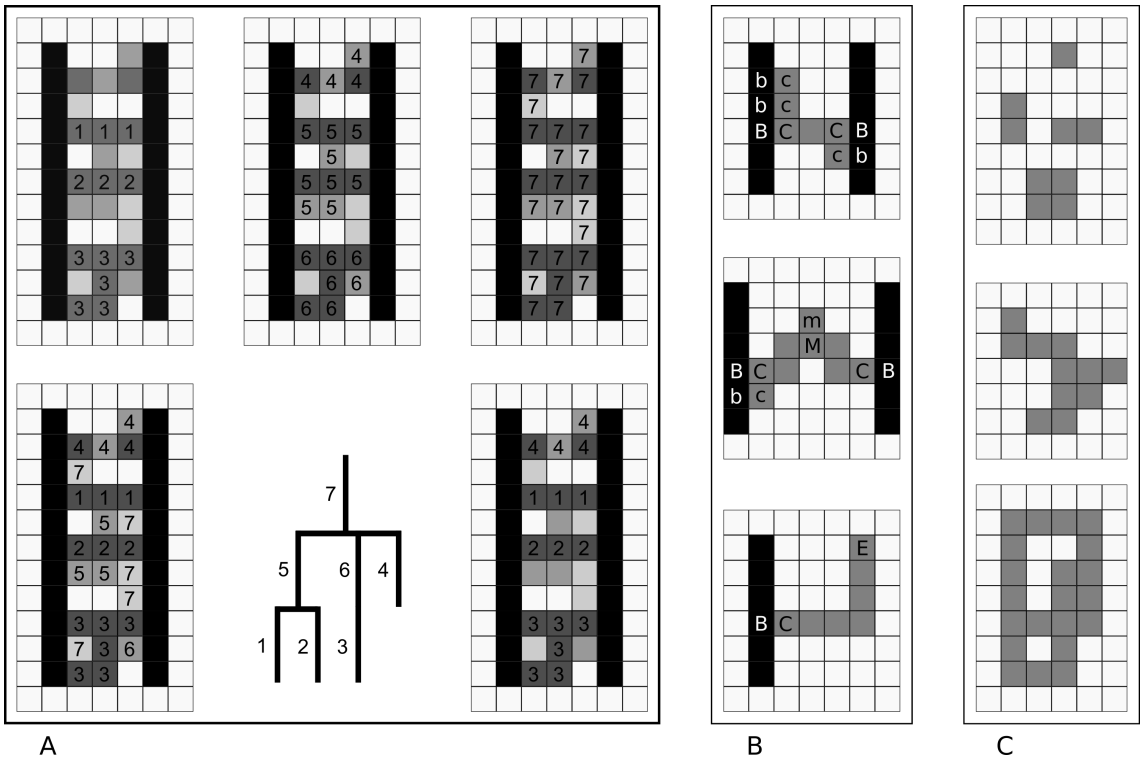

Figure 1: Segmentation and analysis methods. A Single threshold segmentations of a 2D image at low, middle and high thresholds (top row left, middle and right). Pixels available for segmentation are shown in gray, while black denotes boundaries. Pixels comprising the resulting segments are indicated by numbers, different numbers denote different segments. The hierarchical connectivity segmentation: image containing all threshold segmentations in the compact form (bottom, left), the corresponding hierarchical tree (bottom, middle) and the proper hierarchical connectivity segmentation obtained by selecting leaf segments (bottom, right). Black pixels represent boundaries and the segmentation region consists of all other pixels (light to dark grey). B Segment length determination for segments contacting two boundaries (2-bound segments) in straight (top) and mid line modes (middle) and for 1-connected segments. In all cases, boundary contact pixels are labeled by b and B, segment contact pixels by c and C, and central (mid) segment pixels by m and M. Points actually used to calculate length (obtained by length minimization over all contact pixels) are labeled by capital letters. Length of 2-bound segments is calculated as the Euclidean distance between B- and C-labeled pixels for B2B and C2C contact modes, respectively in straight line mode, and as $\overline{CM}+\overline{MC}$ in C2C mid line mode. Length of 1-bound segments is calculated as the distance $\overline{CE}$ in C contact mode and as $\overline{BE}$ in B contact mode. C Topology in 2D. Basic 0-simplex (single pixel), 1-simplices (line segments) and 2-simplex (surface element) (top). Segments having Euler characteristic of 1 and -1 and the number of independent loops of 0 and 2 (middle and bottom, respectively). In all cases boundaries are shown in black and segments in various shades of grey.

- Alternatively, segments can be chosen at the highest threshold level before they merge for the first time with another segment. These are called new branch tops.

This step can be considered as a segment classification, as is therefore combined with other classifications (see 7).

The cutting procedure needs to be chosen based on the biological objectives, so that the final segments accurately represent structures of interest. For example, if the goal is to detect individual complexes that link synaptic vesicles, the cutting should be done by retaining leaf segments. This procedure ensures that all detected segments are taken into account, a valid segmentation is obtained and that segments are selected irrespective of the local background level (Figure 1A bottom right).

In order to obtain similar but generally thicker segments, segments may be selected at the highest threshold before they merge with other segments (branch top segments) (in this way segment 6 of Figure 1A would be chosen instead of segment 3). In addition, cutting schemes based on the threshold values may be used to eliminate weaker segments, while biological constraints often dictate which and how many boundaries a particular type of segments needs to contact.

In summary, the hierarchical connectivity segmentation consist of the connectivity segmentation at multiple thresholds, the establishment of a hierarchical representation of segments and a segment selection (steps 1-3 in Algorithm 1). This procedure combines information from different greyscale levels to yield a proper segmentation that avoids the oversegmentation problem and does not depend strongly on the local background.

6.3 Segmentation parameters

- Threshold (variable `threshold`)is specified as a single number for the single threshold segmentation. For the hierarchical segmentation a list of threshold values is specified and the segmentation is repeated for each of the thresholds. The usual way to set the threshold range is to start with a very low threshold at which no segments are detected, and choose as the maximum value a threshold that is somewhat above what is perceived to be the highest greyscale value that still shows biological material. The threshold step is then chosen so that not many new thresholds are chosen at each higher level and that there are at least 20 different threshold levels. The larger the number of thresholds, the longer the execution time is.

- Boundaries are specified by a boundary label file (`labels_file_name`) and boundary ids (`boundary_ids`)

- A segmentation region (step 1) is a region where the segments are made. It is formed by:

  - Specifying id of the segmentation region (`conn_region`), provided that this region is defined in the boundary labels file.
  - Alternatively, If the segmentation region is not specified in the boundary labels file, the segmentation region id should be set to 0, so that the entire background is used for segmentation.
  - In any case, segmentation does not proceed over any other labels defined in boundary ids and all ids variables
  - Segmentation region defined by one of the previous rules can be restricted by specifying the maximal distance to the boundaries and the manner in which these regions surrounding different boundaries are combined

- A connected object (step 2) is defined based on the segment connectivity structuring element (`struct_el_connectivity`)

- A contact between a segment and a boundary (step 3) is defined based on the segment-boundary structuring element (`contact_struct_el_connectivity`)

- Contact requirements specify a number of different boundaries that a segment is required to contact in order to be retained (step 3). It can be exact (eg. 1, 2) or given as $\geq$1, $\geq$2, ... Number of contacted boundaries states how many different boundaries a segment contacts.

- A segment that contacts one boundary can have more than one contact with that boundary. Contacts are considered separate if the voxels that directly contact the boundary form more than one connected object according to contact counting structuring element (`count_struct_el_connectivity`). This is used to calculate the number of contacts a given segment and a given boundary, which is different from the number of different boundaries that a given segment contacts.

6.4 Important note about boundary and other ids

Incorrect id specification accounts for a majority of problems (exceptions raised) encountered while running the segmentation. In case of error, please check the ids carefully first.

Ids corresponding to regions (segments) of the boundary labels file need to be specified by the following variables in segmentation scripts:

- Boundary ids (variable name `boundary_ids`): Defines boundaries that are used to determine contacts between boundaries and segments generated by this segmentation. These ids should be the same as region_ids (`vesicle_ids`) specified in regions (or vesicles) analysis script.
- All ids (variable name `all_ids`): All regions (segments) defined in boundary labels file. These have to include boundary ids, as well as all other regions defined in the boundary file. Regions (segments) of the boundary file that are not specified here are discarded, that is they are set to 0 for segmentation purposes. Consequently, they may be used for the segmentation region if segmentation region id is 0.
- Segmentation region id (variable name `conn_region`): Id of the region that is segmented. It can be specified as a separate region in the boundary file, or set to 0 in which case the whole background may be segmented (including all regions not specified in the all ids).

7 Segment classification

Classification of segments can be performed immediately after segmentation, or after the analysis. If hierarchical segmentation is performed, the first classification actually extracts non-overlapping segments and makes a flat segmentation (see section6.2). Other classifications are typically done after, but they can also precede extracting a flat segmentation.

In general, all classes generated by one classification are classified separately by a subsequent classification. Further classifications can be chosen from the following options:

- Volume: segments are classified by specified volume bins, one class if generated for each bin
- Contacted boundaries: segments contacting specified boundaries are separated from those that do not contact them. Classes are made separately from the selected and from the remaining segments.
- Number of contacted boundaries: segments are classified according to the number of contacted boundaries.

As a result of classification, one or more separate classes of segments are generated. Segmentation image, results and segmentation pickle files are saved for each class.

8 Analysis of individual segments

Greyscale, morphological and other properties can be chosen to be calculated for each segment. These can be done immediately after segmentation and/or after classification. Properties that are used for classification have to be calculated after segmentation. On the other hand, time consuming calculations might be better to be performed after classification because a smaller number of segments is present at this stage.

This analysis is performed for each class obtained from the classification step. The properties calculated are written to the result files and saved to the segmentation pickle file. Current analysis includes calculation of the following properties.

8.1 Segments and contacts

Every segment created by the segmentation procedures presented here consists of mutually interconnected voxels. That is, each two voxels of a segment are connected via a sequence of neighboring (directly contacting) voxels belonging to the same segment, and no voxel from one segment can be connected to a voxel of any other segment. The choice of structuring element determines what constitutes a neighborhood (direct contact) of a voxel. Here, two (3D) voxels are considered to be in direct contact (neighborhood) if they share a face, that is a voxel (regarded as a cube) can have 6 neighbors. We note that in 2D this corresponds to pixels sharing an edge, or having 4 neighbors. Other choices of structuring elements in 3D are also possible, namely those that define neighboring voxels as those sharing edges (18 neighbors) and sharing vertices (26 neighbors). These are implemented in Pyto (except for topological analysis, see below), but we will not consider them further.

Connectivity is central for our segmentation approach. A segment is said to contact a boundary if a segment voxel is in the direct contact with a boundary voxel. The segment contact region consists of all segment voxels that directly contact the boundary according to the specified structuring element (labeled by 'c' and 'C' on Figure 1B), while the boundary contact region consists of all boundary points that directly contact the segment (labeled by 'b' and 'B' on Figure 1B). Again, we chose the 6-neighbors structuring element to define these contacts, but other choices are possible. However, contact points do not need to be mutually connected (for example, contact between segment 3 and the left boundary on Fig. 1A). Consequently, multiple contact points between a segment and a boundary can exist. Counting contacts requires setting another structuring element, which we again chose to have 6 neighbors.

8.2 Greyscale, morphology and localization

The following properties are determined for all segments:

- Greyscale density: mean, std, min and max.
- Morphology: volume, surface (in voxels) and surface to volume ratio.
- Boundary distance (only for segments that contact exactly two boundaries): Minimal distance between the two boundaries that the segment contacts.
- Distance between a segment and a specified region: This region has to be specified in the boundary labels file and its label id has to be given. The distance can be calculated from the center of the segment, as a mean over all segment voxels, or it can be the minimal distance from a segment voxel to the region.

8.3 Length

Segment length is defined for segments that contact one and two boundaries (1-bound and 2-bound segments, respectively). The length is estimated in respect to the boundaries contacted, so the segment length determination requires finding contact regions.

Various length calculation modes are available. In case of 2-bound segments, the line-like length (line mode 'straight') can be calculated as the shortest distance between two voxels belonging to different segment (contact mode 'C2C') or boundary (contact mode 'B2B') contact regions (Figure 1B). Similarly, contact modes 'B2C' and 'C2B' are also defined. The choice of the contact mode depends on the biological nature of the segments. We note that among the contact modes, 'B2B' mode typically yields the largest and 'C2C' the smallest length. In cases when edge-to-edge length is required, 'B2C' and 'C2B' modes are expected to produce better estimates than the other two contact modes.

Alternatively, the segment midpoint can be used for the length calculation, thus taking into account the curvature of the segment. Specifically, this calculation is performed by minimizing the sum of distances between one (segment or boundary) contact region and a central region on the segment and between the central point and the other contact region (line mode 'mid') (Figure 1B). The minimization is performed over all contact and central region voxels, where the central regions consists of voxels that have approximately same distances to contact regions on both boundaries.

The length of 1-bound segments is calculated between a contact region and the most distant segment voxel. Namely, the distance is minimized over the contact region and maximized over the segment voxels. Similarly, 'C' and 'B' contact modes are defined.

In all cases the Euclidean distance between voxels is calculated.

8.4 Topology

Euler characteristic of a segment is a topological invariant that can be determined by counting (basic) simplices. These simplices have to be specified in all relevant dimensions (from 0 to the dimensionality of the segment), so in our case there are 0-, 1-, 2- and 3-simplices. We defined them as follows: 0-simplex is a point or single voxel, 1-simplices are line segments or two neighboring voxels oriented along principal coordinate axes (2 in 2D, 3 in 3D), 2-simplices are surface elements or square patterns of size 2x2 (1 in 2D, 3 in 3D) and 3-simplex is a volume element or a cubic pattern of size 2x2x2 (none in 2D, 1 in 3D) (Figure 1C shows simplices in 2D, the extension to 3D is straightforward).

If s_n denotes number of n-simplices belonging to a N-dimensional segment, its Euler characteristic (χ) is calculated as:

$$\chi = \sum_{i=0}^{N} (-1)^i s_i$$

For example, counting the number of basic 0-, 1- and 2-simplices for segments shown in Figure 1C middle and bottom, gives (11, 11, 1) and (21, 24, 2) and the Euler characteristic of 1 and -1, respectively.

The Euler characteristic can be also expressed in terms of ranks of homology group, which are also topologically invariant:

$$\chi = \sum_{k=0}^{N-1} (-1)^k b_k$$

The rank of k-dimensional homology group (b_k) can be understood as the number of independent k-dimensional cycles. Specifically, b_0 is the number of disconnected objects (1 for both segments of Figure 1C), b_1 is the number of independent closed loops (0 and 2) and b_2 is the number of (2D) holes (0 for 2D segments), thus yielding values 1 and -1 for the Euler characteristic.

The number of simplices of all dimensions, as well as b_0 and b_2 (in 3D) can be obtained using standard morphological image-processing routines. Consequently, the number of independent closed loops (b_1), which represents the (topological) complexity of a segment, can be calculated using the above two equations.

The calculation of these topological properties is implemented in Pyto for 2D and 3D segments. Also note that these calculations require that neighboring voxels are defined as those that share a face (3D) or an edge (2D).

8.5 Clustering

Connectivity clustering was defined to generate clusters formed by interlinked boundaries and segments, that is boundaries (segments) belonging to the same cluster can be linked by a path composed of directly connected segments and boundaries. In order to compare a connectivity clustering of boundaries or segments with a hierarchical clustering of the same objects, the hierarchical clustering was cut so that the number of clusters is equal to that of the connectivity clustering. Similarity between a connectivity and a hierarchical clustering was estimated using three clustering similarity measures. Rand's method takes all pairs of objects and divides the number of pairs that belong to the same cluster in both clusterings (N_{11}) together with the number of pairs that are in different clusters (N_{00}) with the total number of pairs:

$$R = \frac{N_{00} + N_{11}}{n(n-1)/2}$$

where n is the number of objects. Fowlkes - Mallows similarity is related to probability that a pair that is in the same cluster in one clustering if it is in the same cluster in the other:

$$B = \frac{N_{11}}{\sqrt{\sum n_k(n_k-1)/2 \sum n_{k'}(n_{k'}-1)/2}}$$

where n_k and $n_{k'}$ are the number of objects in the k-th cluster of the first and the second clustering, respectively. We note that here comparison between two clusterings (after the trees were cut) is needed, while the original formulation allows comparing full hierarchical trees at multiple clustering levels. Finally, Variation of information is based on entropy and information associated with clustering:

$$VI = -\sum_{k} P(k)ln(P(k)) - \sum_{k'} P(k')ln(P(k')) - \sum_{k,k'} P(k,k')ln\frac{P(k,k')}{P(k)P(k')}$$

where $P(k)$ and $P(k')$ are probabilities that an object is in cluster k in the first and in cluster k' in the second clustering respectively, and $P(k,k')$ is the probability that and object is both in cluster k in the first and in cluster k' in the second clustering. Rand's and Fowlkes - Mallows coefficients increase with increasing similarity, while variation of information decreases.

8.6 Analysis of other structures

Some of the analysis described above (greyscale, morphology, localization) can be also applied to other structures. These include:

- Boundaries defined in the boundary labels files (as described in Section 5.2).
- Organelles and other structures defined in label files.
- Specific regions in respect to a boundary, such as thin layers parallel to a boundary or between two boundaries (termed layers), or concentric regions located between two roughly parallel boundaries (columns)

9 Applications

Practical details about of the segmentation and analysis procedure are available for the following specific applications:

- Presynaptic terminal (see workflows.pdf)
- Synaptic cleft (work in progress)

10 Citing

Please consider citing us if you use segmentation and analysis in Pyto:

Lučić V, Fernández-Busnadiego R, Laugks U and Baumeister W, 2016. Hierarchical detection and analysis of macromolecular complexes in cryo-electron tomograms using Pyto software. J Struct Biol. 196(3):503-514. doi: 10.1016/j.jsb.2016.10.004.

Thank you.