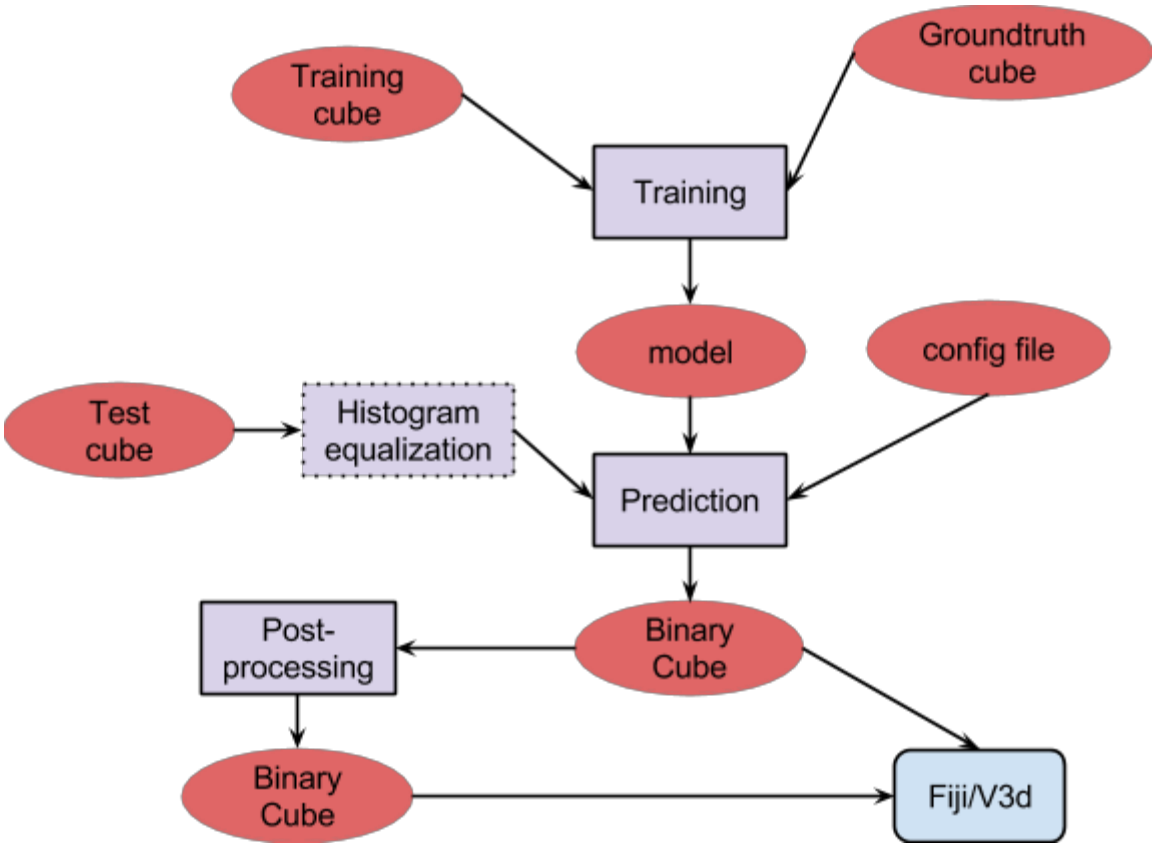
Segmentation of mitochondria in 3d EM datasets

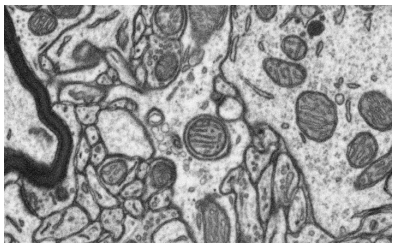
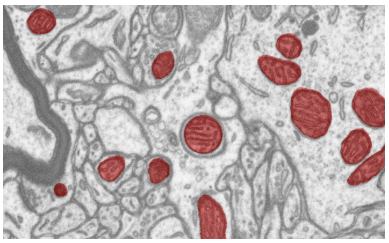
This document provides step-by-step instructions for segmenting mitochondria in 3d EM datasets. The diagram below gives an overview of the segmentation process that consists of two to four steps. First, the system has to be trained to recognize mitochondria. This training step will generate a model file that will be used in the prediction step of the segmentation. The prediction step will then generate a binary TIF cube that can be loaded in Fiji or v3d. Alternatively, one can also post-process the cube to remove small components that are expected to be false detections. Note that if your test cube looks fairly different from the training cube, an additional step (Histogram equalization) might be required.



This tutorial will give instructions to run a command line executable for each step. The arguments will be passed to each executable through a **configuration file**, which is just a

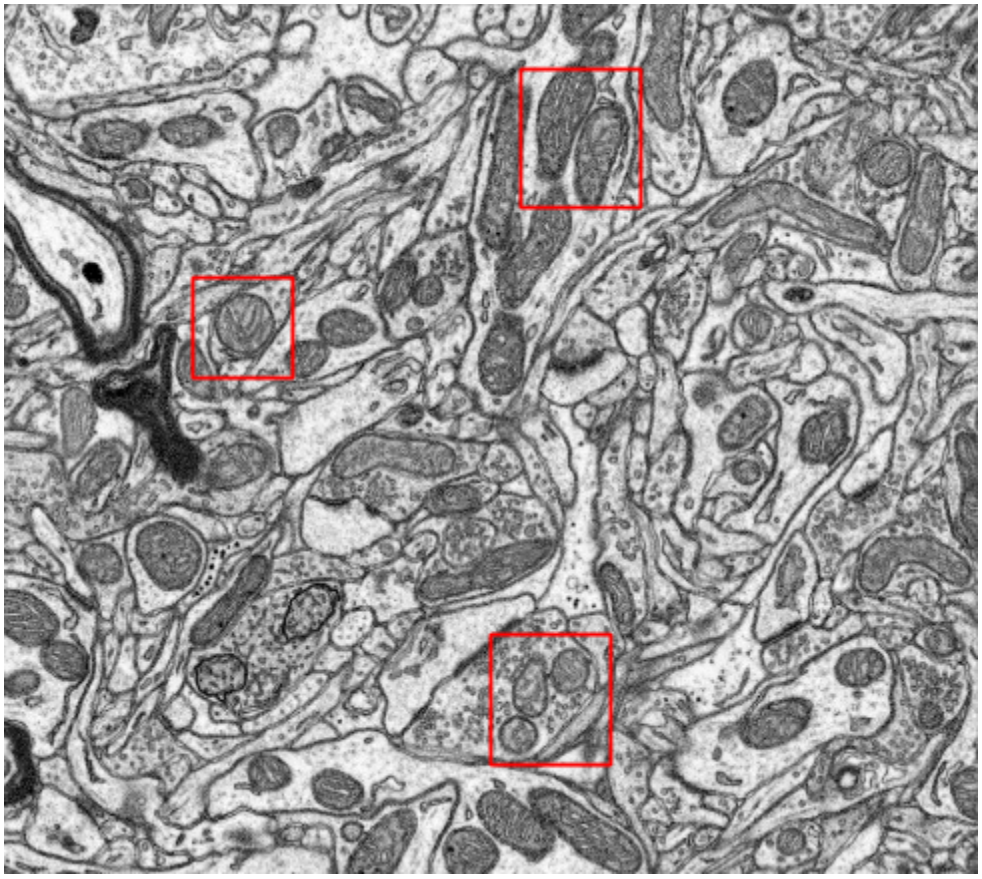
simple text file where each line follows the format *“name=value”*.

Generating ground-truth data for training

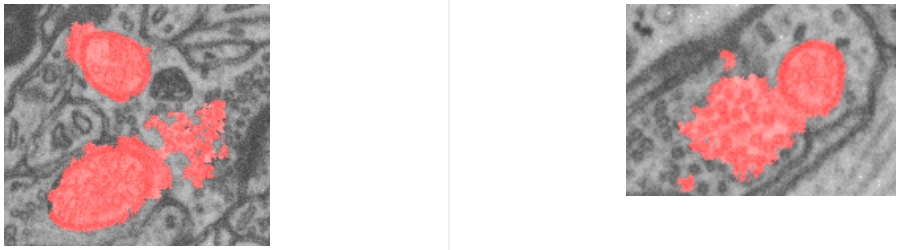
		
Original image	Ground-truth for mitochondria (required for training)	Overlay

Our algorithm requires a ground-truth volume where the instances of the object you want to detect are marked as illustrated on the above figure. The path of this ground-truth volume will have to be specified in the configuration file during training (see *maskTrainingDir* argument described in the training section).

When labelling the ground-truth, we strongly advise to pick a sub-volume with difficult cases (cluster of vesicles, regions containing clutter, weirdly-shaped mitochondria,...). You have to make sure that your training set is representative of the test set, which means that you should include all kind of mitochondria. The red rectangles in the image below contain different groups of mitochondria in terms of their visual appearance (intensity and patterns inside the mitochondria).



An improperly trained algorithm will likely miss some mitochondria whose appearance is too different from the ones included in the training set. Another common mistake is the false detection of clusters of vesicles like the ones shown in the images below. Including these hard examples in your training set will almost surely lead to better results.



Failure case: the algorithm incorrectly segmented clusters of vesicles at proximity of mitochondria.

Training

The training consists of a command line executable that generates a model file, which can then be used for the prediction step.

Mandatory arguments

- *trainingDir*: input volume (TIF cube)
- *maskTrainingDir*: Mask volume (TIF cube or series of png images containing black and white voxels only)

Example

trainingDir=imagesLeft_35-199/
maskTrainingDir=masksLeft_35-199/

Optional arguments

- *outputModel*: Output model file (*default=model.txt*)
- *testDir*: Test input volume (TIF cube)
- *maskTestDir*: Test mask volume (TIF cube containing black and white voxels only)
- *stepForOutputFiles*: output files and corresponding scores are computed every stepForOutputFiles iterations. Reasonable values are 10 for default training algorithm and 100 for stochastic optimization (-w 9 option described below).
- *featureTypes*: Type of features extracted from the image/cube and used to train a model. See list of available features below.
- nMaxIterations: should be around 500-700 iterations.

Example

trainingDir=imagesLeft_200-399/
maskTrainingDir=masksLeft_200-399/
outputModel=model.txt

List of available features

F_HISTOGRAM = 1,
F_COLOR_HISTOGRAM = 2,
F_LOADFROMFILE = 4,

F_GLCM = 8,
F_POSITION = 16,
F_FILTER = 32,
F_GAUSSIAN = 64,
F_SIFT = 128,
F_DFT = 256,
F_BIAS = 512

F_GLCM are Gray-Level Co-occurrence Matrices while F_FILTER consists of the responses of a set of filters, including Gradient magnitude and eigenvalues of Hessian.

The *featureTypes* parameter should set to:
featureTypes=33

Command

Once you have specified the arguments in the configuration file, you can run the command:
`train configuration_file.txt`

All the models files are stored in parameter_vectors0. If you wish to see the score of this parameter file, you can use the sortScore.py python script stored in the “scripts” directory.

How to use the sortScore.py script?
$ ./sortScore.py
config_imagesLeft_35-199 **60**/160 0.75

In the above example, the total number of iterations was 160 but the best parameter file is the one for the 60th iteration. That means you should use the file named parameter_vector0/iteration_60.txt

Prediction

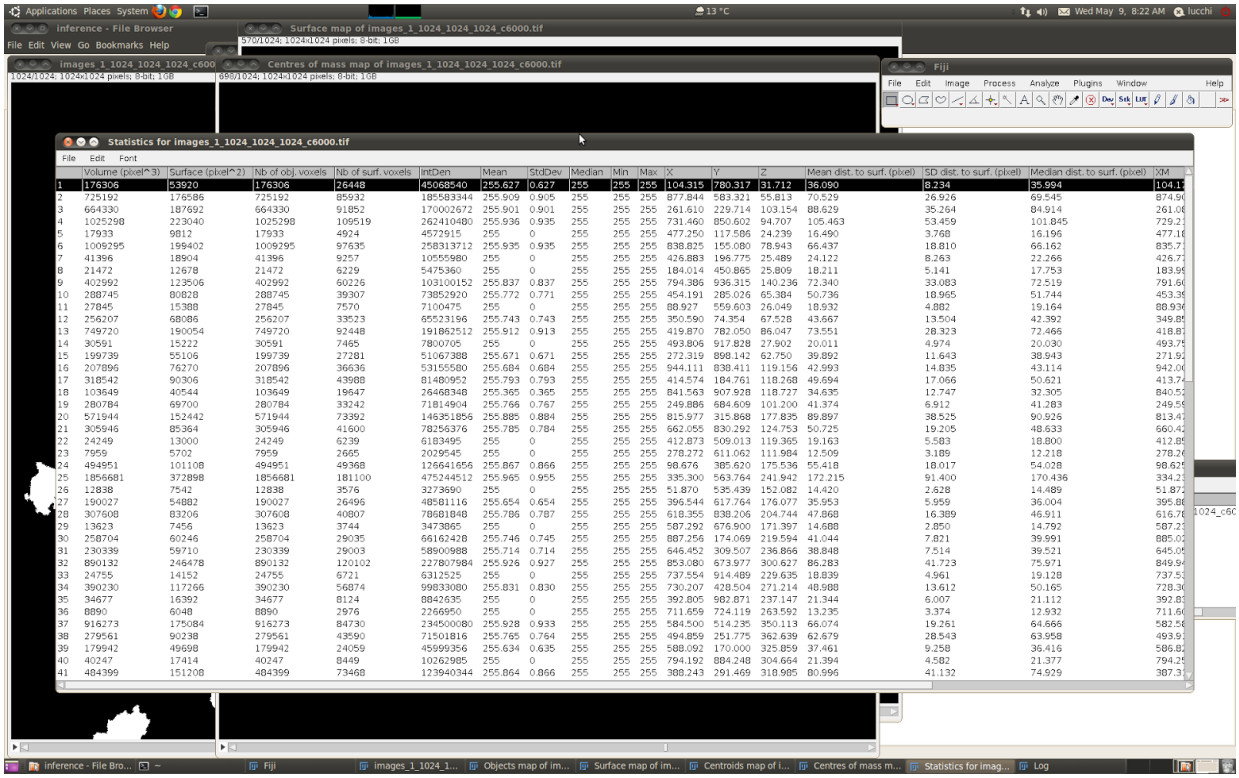
`predict -c configuration_file -w model_file.txt -i image_directory`

The image_directory argument is the name of the directory that contains the png images in which you want to detect mitochondria.
The model file is the file you obtained during the training step. If you skipped this step, you can try to use one of the precomputed model stored in data/model/
The configuration file is the same file you used for training.
Note that you can get extra information by adding “-v 1” (verbose mode) when calling the program.

Statistics

3d object counter in Fiji

One way to compute statistics is to use the “3d objects counter” tool available in Fiji (Analyze menu). This plugins requires a black and white cube as input (it will run a connected component algortithm so there is no need to do it beforehand). Make sure you uncheck “Exclude objects on edges”. The screenshot below shows the result obtained after running this tool.
Note: It seems that the current implementation of this tool is slow at extracting connected components so please be patient.
Website: http://imagejdocu.tudor.lu/doku.php?id=plugin:analysis:3d_object_counter:start

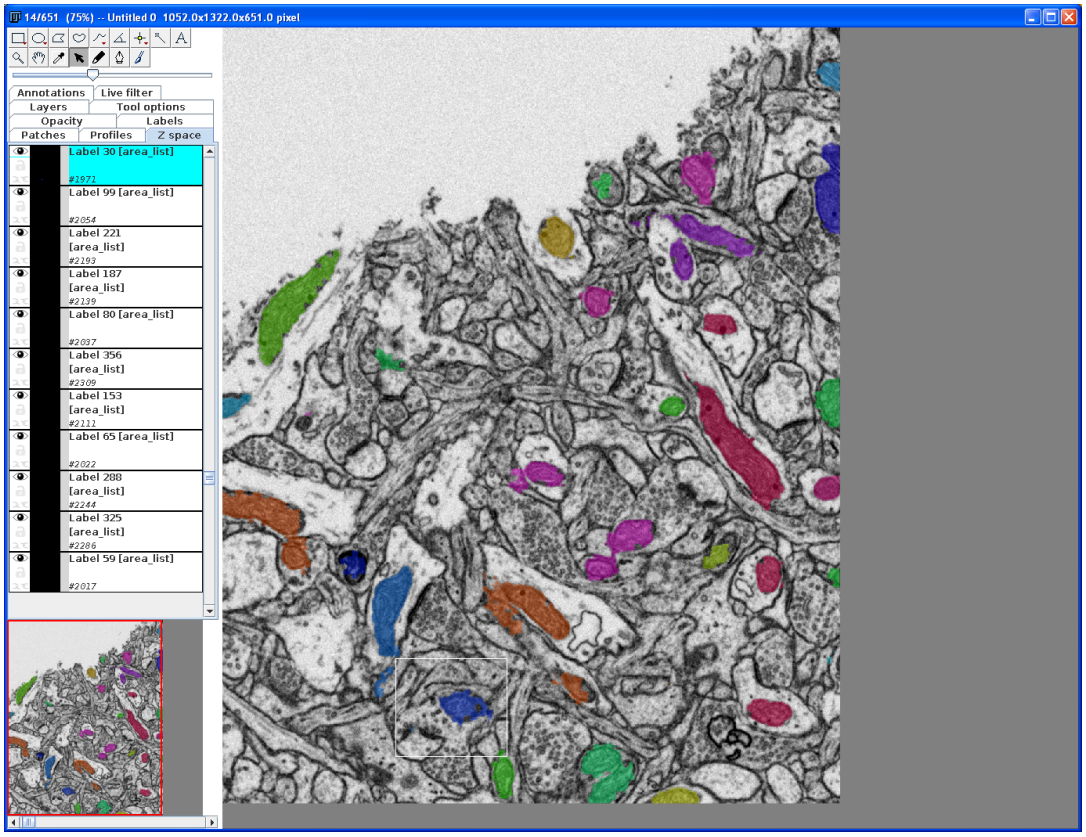


Editing the results

One can use trakEM2 (available in Fiji) to edit the results. This procedure consists of 2 steps described below :

1. Extract connected components
The first step consists in extracting the connected regions from the binary cube created by the prediction step (eventually post-processed). This step can be done with a Fiji plugin available in the menu “Plugins/Process”. You can keep the default settings. Once terminated, save the output volume.
See http://www.longair.net/edinburgh/imagej/find-connected-regions/ for further details.

2. Import results in trakEM
The second step is to create a trakEM project (Click on File -> New ->TrakEM2) and import the original data (Right click -> Import stack). You can then import the cube generated by the connected region tool (Right click -> Import -> Import labels as arealist).
As shown on the image below, you should have a different area list for each mitochondria so you can easily remove the remaining errors (use the “Select and Transform Tool” by clicking on the dark arrow in the menu bar. Then select an object in the image by right click) and eventually refine the segmentation of each mitochondrion.



Split mitochondria

Coming soon...ask us if you need this.

Misc / HowTo / Frequently Asked Questions

1. Can I use this code with 2d images ?
Yes, the only thing you have to do is to edit your config file and set the following variables as specified below :
slice3d=0
msrc=0
voc=1

2. How do I load my own features ?
You have to add the following options in your configuration file:
featureTypes=4
feature_file=DIRECTORY_NAME/featureFile_scaled.txt
featureSizePerFile=146

The feature type *4* means that you want to load features from a file. The *featureSizePerFile* is the dimensionality of the feature. The *feature_file* parameter specifies the path of a text file that contains the path of the features to be loaded. It is a text file with the following format :
DIRECTORY_NAME
histograms
eigenvalues_of_hessian

The first line is the name of the directory that contains the features while the following line**s** specify the names of the file that contain the features to be loaded. Three different types of files are supported and will be detected according to the file extension:

1. ".tif" : TIF files (standard single channel TIF cubes).
2. ".txt" : Text file (libsvm file format)
3. ".bin" : Binary file (Carlos file format)

The format for binary file is the following:

* <unsigned int numRows>
* <unsigned int numCols>
* <numRows * numCols float values (32-bit float data)>
* The data is in column-major format, and each row belongs to a given supervoxel.

3. The segmentation results are not very smooth.
Try to increase the value of min_percent_to_assign_label. Reasonable values should be between 0.2 and 0.5.

4. I edited the config file on Windows and the program seems to be reading the wrong configuration parameters. Windows adds return characters at the end of each line. You can check this by using cat:
```
cat -v input.txt
```

You can remove the return characters by running the following command:
```
awk '{ sub("\r$", ""); print }' input.txt > output.txt
```

5. Get a plot showing the behavior of the algorithm, i.e. evolution of the training and test scores...
```
python plotAll.py -d INPUT_DIR
```

References

[1] Exploiting Enclosing Membranes and Contextual Cues for Mitochondria Segmentation, MICCAI 2014
A. Lucchi, C. Becker, P.M. Neila, P. Fua

[2] Supervoxel-Based Segmentation of Mitochondria in EM Image Stacks with Learned Shape Features, IEEE Transactions on Medical Imaging, Vol. 30, Nr. 11, October 2011
A. Lucchi, K.Smith, R. Achanta, G. Knott, P. Fua.

[3] Learning for Structured Prediction Using Stochastic Subgradient Descent with Working Sets, CVPR, 2013
A. Lucchi, Y. Li, P. Fua.