ORION 2
Reference Manual

Computational Biomedicine Lab

Department of Mathematics

University of Houston

CHAPTER 1

Introduction

ORION II is an application for the automatic segmentation and tracing of three-dimensional neuronal images. This document briefly describes its use, as well as the way to train a classifier.

1. System Requirements

This tool was coded in MATLAB R2011b, that requires an x86 processor supporting SSE2 instruction sets, and a recommended minimum of 2GB of RAM. We recommend 1 GB of additional memory for every 10 million voxels on the size of the typical image to be processed.

For a support vector machine engine, we use LIBSVM, that needs to be available in the system.

2. Input

The solids to be used for sample generation, training or pipelining should be given in RAW format, with the respective MHD also stored in the same directory.

3. Ouptut

On this section, we assume the settings as specified in the examples above. Every RAW solid will be accompanied by the corresponding MHD.

For the pipeline script, the outputs for the file `OP1.raw`, would be `OP1_dOP_BIN.raw` (binary solid), `OP1_dOP_PRB.raw` (probability solid), `OP1_dOP_BIN_CL.swc` (centerline file).

For the train script. the output would be the file diademOP.mat, containing a struct called model, containing training specifications (so they can be used also for pipelining), and the classifier.

For the index script, when the Laplacian solids are generated, these will be `OP1_NLAP.raw` and `OP1_PLAP.raw`. Also, the index solids, containing the positive and negative samples, respectively `OP1_P` and `OP1_N`. It also leaves in memory the variables 'p' and 'n' containing, respectively, the number of positive and negative samples. The index

solids can be visualized to verify that its accuracy, using a visualization software, like Avizo.

CHAPTER 2

Index Script

This script allows the user to optimize the parameter for sample selection.

```
53      %%%%%%%%%%%%%%%%%%%
54      % CHANGING OPTIONS %
55      %%%%%%%%%%%%%%%%%%%
56
57      idx.genLap = true;
58      idx.lapFact = [0.25; 0.5; 0.75];
59      idx.boxes = {[300 450 125 250 4 52; ...
60          127 296 203 315 25 60; 26 81 243 361 5 41]};
61      idx.pos = {[1.4 1.1;1 1.1;1.25 1.1]};
62      idx.nameList = {'OP1'};
63      idx.pathList = {'C:\O2\OP1'};
```

`idx.genLap`: This option is a switch that indicates to the software, whether or not Laplacian Solids should be generated. Regardless of whether the boxes or the pos-factors change, as long as the field exp.lapFact remains unchanged, the Laplacian solid may be generated only once, and reused. Index generation may be two orders of magnitud faster if this Laplacian solid is not generated.

`idx.lapFact`: List of Laplacian factors to be used in the selection of samples. Each number needs should be strictly contained between 0 and 1.

`idx.boxes`: Cell of arrays of coordinates of the boxes where the samples are going to be selected. One array for each solid used for training. Each array should be an mx6 matrix, where m is the number of regions from where the samples are to be taken. Each row contains the coordinates on one of such boxes, given in the format xMin, xMax, yMin, yMax, zMin, zMax.

`idx.pos`: Cell of arrays of the percentage parameters for the boxes where the samples are going to be selected. One array for each solid used for training. Each array should be an

mx2 matrix, where m is the number of regions from where the samples are to be taken. Each row contains the percentage parameters on one of such boxes, given in the format positivePercentage, negativePercentage. Each of these paramters should be strictly contained between 0 and 100, but typically it is contained between 0.5 and 2.

`idx.nameList:` Cell containing the file name of each individual solid to be used for training.

`idx.pathList:` Cell containing the path of each individual solid to be used for training.

When the script is executed, the variables 'p' and 'n' are left on memory, and they represent the number of positive and negative samples, and its addition is the total number of samples.

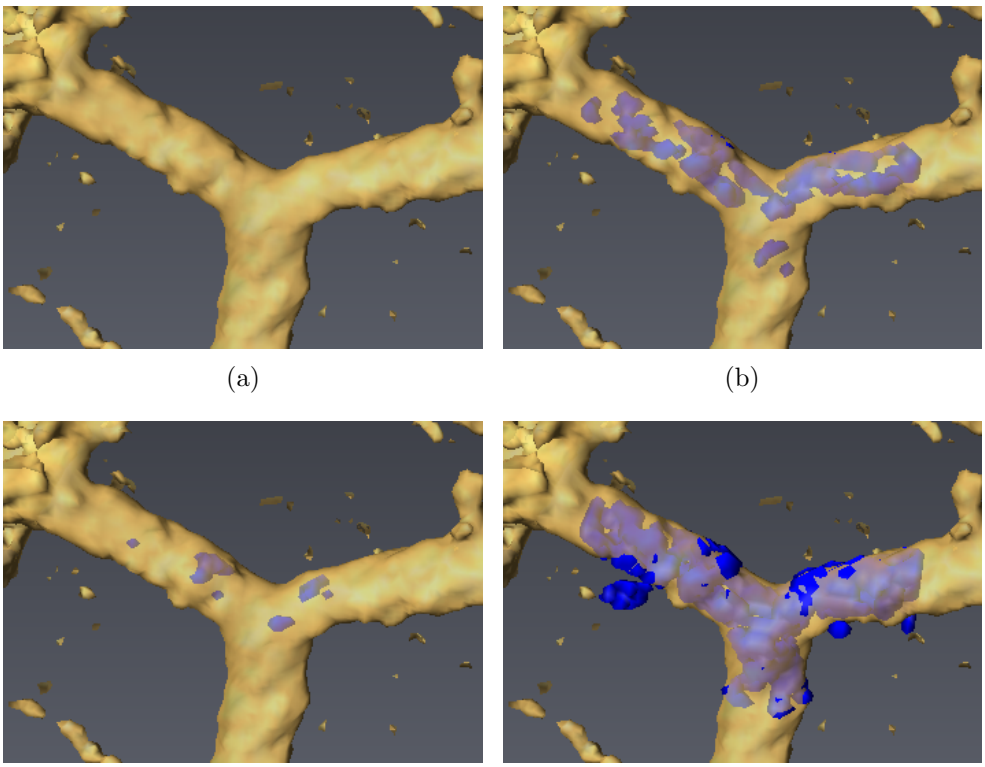
1. Index Generation Strategy

The Index Generation is the only part of the process that is interactive. Once the Laplacian solids have been generated once, change the value of `idx.genLap` to false. It will save execution time.

The coordinates of the boxes should be chosen in such a way that the regions they cover are representative of the structure as a whole.

Once these boxes have been chosen, the parameters (`idx.pos`), have to be chosen in such a way that they genereate as many samples as possible, but minimizing or eliminating false positive or negative samples. In Figure 1, the subfigure 1(c) shows and undersampling for positive samples in a region of the solid. This means, the pos parameter associated (in the first column) is too low. Subfigure 1(d) shows an oversampling, where there are visible false positives, implying a pos parameter too high. Subfigure 1(b) shows what the operator deemed as a satisfactory sampling.

Once these samples have been generated, the settings can be copied into the respective fields in the Train Script.



(a) (b)

(c) (d)

FIGURE 1. Section of a solid with different positive sample settings. (a) Raw solid without samples. (b) Raw solid with seetings deemed appropriate. (c) Raw solid with settings causing undersampling. (d) Raw solid with settings causing oversampling.

CHAPTER 3

Train Script

This script allows the user to set up the options necessary to train a new classifier. Several of the parameters can be first optimized by using idxScript, and later copied to this in particular, exp.lapFact, exp.boxes

```
14      %%%%%%%%%%%%%%%%%%%%
15      % CHANGING OPTIONS %
16      %%%%%%%%%%%%%%%%%%%%
17
18      modelPath ='E:\Model';
19      modelName = 'diademOP';
20      model.suffix = '_dOP';
21      maxNoise = 27;
22      exp.genLap = false;
23      exp.lapFact = [0.25; 0.5; 0.75];
24      exp.boxes = {[300 450 125 250 4 52;...
25          127 296 203 315 25 60; 26 81 243 361 5 41]};
26      exp.pos = {[1.4 1.1;1 1.1;1.25 1.1]};
27      exp.nameList = {'OP1'};
28      exp.pathList = {'C:\O2\OP1'};
29      exp.featType = {'Lap','LP','HP'};
30      exp.featParam = {[.25; .5; .75], [.3; .5; .7],...
31          [.6 .3; .4 .1; .8 .5]};
32      exp.spBySize = true;
33      exp.spByQuan = false;
34      exp.prtSiz = 2500;
35      exp.numPrt = 5;
36      exp.numExp = 1000;
```

`modelPath`: Directory where the classifier is going to be stored.

`modelName`: Name of the file where the classifier is going to be stored.

`model.suffix`: Suffix to be included as a field in the model package. When the pipeline script is run, this suffix will be added to the files segmented.

`maxNoise:` The maximum size of a disconnected particle in the solid, that will be considered as noise.

`exp.genLap:` This option is a switch that indicates to the software, whether or not Laplacian Solids should be generated. Regardless of whether the boxes or the pos-factors change, as long as the field exp.lapFact remains unchanged, the Laplacian solid may be generated only once, and reused. Index generation may be two orders of magnitud faster if this Laplacian solid is not generated.

`exp.lapFact:` List of Laplacian factors to be used in the selection of samples. Each number needs should be strictly contained between 0 and 1.

`exp.boxes:` Cell of arrays of coordinates of the boxes where the samples are going to be selected. One array for each solid used for training. Each array should be an mx6 matrix, where m is the number of regions from where the samples are to be taken. Each row contains the coordinates on one of such boxes, given in the format xMin, xMax, yMin, yMax, zMin, zMax.

`exp.pos:` Cell of arrays of the percentage parameters for the boxes where the samples are going to be selected. One array for each solid used for training. Each array should be an mx2 matrix, where m is the number of regions from where the samples are to be taken. Each row contains the percentage parameters on one of such boxes, given in the format positivePercentage, negativePercentage. Each of these paramters should be strictly contained between 0 and 100, but typically it is contained between 0.5 and 2.

`exp.nameList:` Cell containing the file name of each individual solid to be used for training.

`exp.pathList:` Cell containing the path of each individual solid to be used for training.

`exp.featType:` Cell containing the specifications of the types of features to be used for training. There are three types of features implemented: 'Lap' (Laplacian), 'LP' for Low Pass, 'HP' for High Pass.

`exp.featParam:` Cell containing the parameter arrays for each of the feature type to be used. These parameters are strictly contained between 0 and 1. For Laplacian and Low Pass features, the array should be a one column vector with as many entries as features of that type are to be produced, and a High Pass feature should be a two column matrix, with as many

rows as features of the type are to be produced. The second element on every row should be strictly smaller than the first.

`exp.spBySize:` Boolean that specifies if the sample set should be partitioned on pieces of a specific size. At least one of the booleans `exp.spBySize` and `exp.spByQuan` should be true.

`exp.spByQuan:` Boolean that specifies if the sample set should be partitioned on a specific number of pieces. At least one of the booleans `exp.spBySize` and `exp.spByQuan` should be true.

`exp.prtSiz:` If `exp.spBySize = true`, `exp.prtSiz`, then represents the size of each partition of the sample set. If both `exp.spBySize` and `exp.spByQuan` are true, then the product of `exp.prtSiz` and `exp.numPrt` shuold be less than or equal to the number of samples. We recommend `exp.prtSiz` to be as large as possible, allowing at least a four fold validation.

`exp.numPrt:` If `exp.spByQuan = true`, `exp.numPrt`, then represents the number of partitions of the sample set. If both exp.spBySize and exp.spByQuan are true, then the product of exp.prtSiz and exp.numPrt shuold be less than or equal to the number of samples. We recommend exp.numPrt to be between four and ten.

`exp.numExp:` Number of experiments to be ran to find the optimal parameters for the SVM classifier.

CHAPTER 4

Pipeline Script

If an SVM classifier is already available, the file 'pipelineScript.m' can be configured to process one or multiple files. At this point, it is assumed that a classifier has been trained. Six options are necessary for execution:

```
10      %%%%%%%%%%%%%%%%%%%%
11      % CHANGING OPTIONS %
12      %%%%%%%%%%%%%%%%%%%%
13      prd.nameList = {'OP1','OP2','OP3'};
14      prd.pathList = {'C:\O2\OP1','C:\O2\OP2','C:\O2\OP3'};
15      prd.modName = 'diademOP';
16      prd.modPath = 'E:\Model';
17      zSmearFactor = 1;
18      clSuffix = '_CL';
19
```

`prd.nameList`: Cell containing the file name of each individual solid to be process.

`prd.PathList`: Cell containing the path of each individual solid to be process.

`prd.modName`: Name of the file where the classifier is stored.

`prd.modPath`: Directory where the classifier is stored.

`zSmearFactor`: If the zSmear Factor is known to the user, it can be entered here. If it is not known, we recommend to use 1 as a default.

`clSuffix`: A sufix to be added to the centerline output file. (It may be an empty string, but it has to be specified.)