subTOM Documentation

Release 1.1.6

Dustin Reed Morado

Sep 29, 2021

TABLE OF CONTENTS:

1 Installation 3

2 Conventions 5

3 Example Workflow 9

4 Scripts 11

5 subtom_alignment 13

6 subtom_average 19

7 subtom_bandpass 23

8 subtom_cat_motls 25

9 subtom_clean_motl 27

10 subtom_compare_motls 31

11 subtom_even_odd_motl 33

12 subtom_extract_noise 35

13 subtom_extract_subtomograms 39

14 subtom_maskcorrected_fsc 43

15 subtom_preprocess 47

16 subtom_random_subset_motl 53

17 subtom_plot_filter 55

18 subtom_plot_scanned_angles 59

19 subtom_reconstruct 61

20 subtom_renumber_motl 67

21 subtom_rotx_motl 69

22 subtom_scale_motl 71

23 subtom_scale_noisemotl 73

24 subtom_seed_positions 75

25 subtom_shape 77

26 subtom_split_motl_by_row 83

27 subtom_transform_motl 85

28 subtom_unclass_motl 87

29 Functions 89

30 subtom_bandpass 91

31 subtom_cat_motls 93

32 subtom_clean_motl 95

33 subtom_compare_motls 97

34 subtom_even_odd_motl 99

35 subtom_extract_noise 101

36 subtom_extract_subtomograms 103

37 subtom_maskcorrected_FSC 105

38 subtom_parallel_sums 107

39 subtom_plot_filter 109

40 subtom_plot_scanned_angles 111

41 subtom_random_subset_motl 113

42 subtom_renumber_motl 115

43 subtom_rotx_motl 117

44 subtom_scale_motl 119

45 subtom_scan_angles_exact 121

46 subtom_seed_positions 125

47 subtom_shape 127

48 subtom_split_motl_by_row 129

49 subtom_transform_motl 131

50 subtom_weighted_average 135

51 subtom_unclass_motl 137

52 subTOM: General Classification Utilities 139

53 subTOM: Multivariate Statistical Analysis 155

54 subTOM: Multireference 167

55 subTOM: Principal Component Analysis 181

56 subTOM: Wedge-Masked Difference Classification 201

57 subTOM 215

58 Indices and tables 225

SubTOM - *Subvolume processing scripts with the TOM toolbox* is a collection of scripts form a pipeline for subvolume alignment and averaging of electron cryo-tomography data.

CHAPTER

ONE

INSTALLATION

The installation of subTOM is relatively straight-forward. subTOM is currently only built for 64-bit linux computers, with no plans currently to produce builds for Windows or Mac.

1. `git clone /net/dstore2/teraraid/dmorado/software/subTOM`
2. `cd subTOM`
3. `chmod u+x install.sh`
4. `./install.sh <INSTALL_DIR> <MCR_DIR>`

Since subTOM is written in Matlab, having a license to use Matlab is preferable, and makes some tasks simpler, while also allowing for doing your own scripting with the TOM toolbox. However, the effort has been made to make sure that the pipeline can run as a whole using only the Matlab Compiler Runtime, which is freely available software, and can be found here:

https://uk.mathworks.com/products/compiler/matlab-runtime.html

subTOM is currently built against the 2021b/v911 MCR so that is the one that you need to have downloaded and have access to.

At the LMB we have the MCR already installed at `/lmb/home/public/matlab/jbriggs`, which you can use for your installation.

subTOM is also distributed like most software nowadays as a Git repository, so if you do not have Git you can find out how to install and use Git here:

https://git-scm.com/doc

1.1 Step-by-Step Instructions

1. From the directory in which you want to install subTOM clone the repository.

- `git clone /net/dstore2/teraraid/dmorado/software/subTOM`

2. Change into the newly created subTOM directory.

- `cd subTOM`

3. Make the installation script user-executable, so that you can run it.

- `chmod u+x install.sh`

4. Run the install script specifying the installation directory and the directory that contains the MCR installation.

- `./install.sh <INSTALL_DIR> <MCR_DIR>`

- *example* `./install.sh /net/dstore2/teraraid/dmorado/software/subTOM /lmb/home/public/matlab/jbriggs`

1.2 Building

If you have access to the MATLAB compiler you can also build subTOM from the source simply following the steps here, beginning in the subTOM installation directory:

1. `cd src`
2. Edit `subtom_mcc_build.m` and change the top three variables to point correctly at:
   - The subTOM source directory
   - The root folder of the TOM Toolbox
   - The path to the MATLAB Toolbox directory * The Statistics and Machine Learning Toolbox is needed for *subtom_cluster*
3. Run `subtom_mcc_build.m` in MATLAB and it should correctly compile all neccessary functions and place them in the correct locations.
4. If you run into an issue with a MEX-function in TOM toolbox (such as tom_rotate), then you can recompile such a function with the command `mex -R2018a <filename>` and then recompile subTOM.

CHAPTER

TWO

CONVENTIONS

2.1 Preprocessing

Preprocessing is done for dose-fractionated data that comes from detectors that collect movie tilt-images. Preprocessing is done with several programs from various sources.

- **Beam-induced Motion Correction**
  - The command `alignframes` alignframes man page which is a part of the IMOD package is used to do the beam-induced motion correction of the movies. subTOM assumes that the data is collected with SerialEM but the program should support MRC and TIFF format movie-frames collected with other programs as well. However this requires you to have your movies to have the following name-scheme:
    - `<BASENAME>_<FRAME_IDX>_<ANGLE>.<mrc|tif>`
      - Where `<BASENAME>` is your own filename identifier *e.g. TS_01*
      - Where `<FRAME_IDX>` is a three digit identifier of the movie that describes the order the data was collected in *e.g. 000-040*
      - Where `<ANGLE>` is the tilt-angle at which the movie was collected at *e.g. 15.0*
- **Defocus Estimation**
  - The programs CTFFIND4, GCTF, and IMOD's `ctfplotter` ctfplotter man page command can all be used to estimate the defocus in the corrected tilt-series
- **Dose Filtering**
  - The command `alignframes` which is part of the IMOD package is also used to do the filtering of movies based on the accumulated dose of each tilt-image.

2.2 CTF Correction

CTF correction is done in 3D using the program novaCTF, and a run script `run_nova.sh` is included to facilitate performing novaCTF in parallel and on an SGE cluster.

2.3 Particle Picking

Particle picking is done using UCSF Chimera. First users use the built-in Volume Tracer utility to create a Marker Set of points at the center of spherical particles onto which seed positions or a collection of Marker Sets of points along tubular surfaces onto which seed positions. The number of Marker Sets used is not important in picking points on spheres, but in picking points on tubes, each tube should correspond to a single Marker Set. The collection of Marker Sets should be saved to a single file, one per tomogram with the name format:

- `<BASENAME>_<TOMOGRAM_IDX>.cmm`
  - Where `<BASENAME>` is your own filename identifier *e.g. clicker*.
  - Where `<TOMOGRAM_IDX>` is the tomogram number *e.g. 1*.

Motive Lists are then generated for the picked objects using the *PickParticle* plug-in developed in the Briggs' lab by Kun Qu. The format of motive lists is detailed below, and the motive list is assumed to saved to a single file, one per tomogram with the name format:

- `<BASENAME>_<TOMOGRAM_IDX>.em`

2.4 Alignment and Averaging

The alignment parameters for a set of data are stored in a MOTive List or so-called MOTL file, which is a table of 20-fields stored in an EM-format binary data file. Particles are also extracted into subvolumes in EM-format from tomograms which are expected to be in MRC-format with the name:

- `<TOMOGRAM_IDX>.rec`

2.4.1 Orientations

Coordinate System

subTOM uses a right-handed coordinate system where positive rotations are clockwise looking along the directed axis. The orthogonal axes X, Y, Z are with the positive Z-axis pointing out of the screen out at the user.

Image Center

Since Matlab uses array-indices that start from 1, unlike most other programming languages which count from zero, the origin of a subvolume with dimensions, NX, NY, NZ is defined as:

$$O = (\lfloor NX/2 \rfloor + 1, \lfloor NY/2 \rfloor + 1, \lfloor NZ/2 \rfloor + 1)$$

Euler Angle Rotations

Rotations in MOTLs describe the best-found rotation of the reference to the particle in terms of ZXZ Euler angles in degrees. The Euler angle definition in subTOM is:

- The first rotation *Azimuth* or *psi* (ψ) about the Z-axis.
- The second rotation *Zenith* or *theta* (θ) about the new X-axis.
- The final rotaiton *Spin* or *phi* (ϕ) about the final Z-axis.

This is particularly confusing given that phi and psi generally are swapped in other software packages, but is kept for historical reasons from the TOM-toolbox. Therefore care has been taken to use the unambiguous notation azimuth, zenith, and spin in most of the subTOM code and documentation.

Translations

Translations in MOTLs describe the best-found translation of the reference to the particle in pixels with respect to the subvolume origin. This translation occurs after the rotation of the reference about the subvolume origin.

2.5 Motive List Specification

Field	Contents
1	Cross-Correlation Coefficient
2	Marker Set Used from PickParticle
3	Radius of tube/sphere in PickParticle
4	Particle Number (Running count from 1)
5	Tomogram Number (Running count from 1)
6	PickParticle Object Number (Running)
7	Tomogram Number (From Filename)
8	X-coordinate in Tomogram (Integer)
9	Y-coordinate in Tomogram (Integer)
10	Z-coordinate in Tomogram (Integer)
11	X-translation AFTER rotation of Ref.
12	Y-translation AFTER rotation of Ref.
13	Z-translation AFTER rotation of Ref.
14	Not Used (X-shift BEFORE rotation)
15	Not Used (Y-shift BEFORE rotation)
16	Not Used (Z-shift BEFORE rotation)
17	Spin Rotation of Ref. in Degrees
18	Azimuth Rotation of Ref. in Degrees
19	Zenith Rotation of Ref. in Degrees
20	Class Number

2.5.1 Class Number

The class number field acts as a field for classification, but also thresholding. Historically:

- Particles that have class number 1 are always aligned and included in the final average.
- Particles that have class number 2 are always aligned but are not included in the final average.
- Particles that have class number ≤ 0 are not aligned nor included in the final average.

Remaining class numbers 2 can be used in classification to identify homogeneous subsets within a heterogeneous dataset.

CHAPTER

THREE

EXAMPLE WORKFLOW

The best way to learn how to use subTOM is to just start using it. I would suggest downloading the HIV-1 CA-SP1 publicly available dataset EMPIAR-10164 and using the same subset that was used in Turonova et al. An example workflow showing which scripts are used when and where is included here as a sort of guide into what a particular project run of subTOM looks like from start to finish. Scripts are copied from your subTOM installation directory to your project directory and edited from the project directory. Then you make the script executable and run it, again from the project directory.

You can download a PDF version of the workflow: `here`.

CHAPTER

FOUR

SCRIPTS

Scripts are the main point of user-interaction with the processing pipeline. As such care has been taken to make sure that the scripts are well commented and that the user-defined options are well separated from the actual *black-box* level processing further down in the code.

Each script is a simple BASH file that generally calls some piece of code from either IMOD or a subTOM Matlab function. The scripts are meant to be first edited, filling in the necessary information and then executed in the terminal. Matlab is not necessary to run the scripts, just the Matlab Compiler Runtime, which should have been taken care of in the installation.

CHAPTER

FIVE

SUBTOM_ALIGNMENT

The main pipeline process script of subTOM. Iteratively aligns and averages a collection of subvolumes.

This subtomogram alignment script uses five MATLAB compiled scripts below:

- *subtom_scan_angles_exact*
- *subtom_cat_motls*
- *subtom_parallel_sums*
- *subtom_weighted_average*
- *subtom_compare_motls*

5.1 Options

5.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

local_dir Absolute path to the folder on a group share, if the scratch directory is cleaned and deleted regularly this can set a local directory to which the important results will be copied. If this is not needed it can be skipped with the option skip_local_copy below.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

5.1.2 Variables

align_exec Alignment executable

cat_exec Concatenate MOTLs executable

sum_exec Parallel Summing executable

avg_exec Weighted Averaging executable

compare_exec Compare MOTLs executable

5.1.3 Memory Options

mem_free_ali The amount of memory the job requires for alignment. This variable determines whether a number of CPUs will be requested to be dedicated for each job. At 24G, one half of the CPUs on a node will be dedicated for each of the processes (12 CPUs). At 48G, all of the CPUs on the node will be dedicated for each of the processes (24 CPUs).

mem_max_ali The upper bound on the amount of memory the alignment job is allowed to use. If any of the processes request or require more memory than this, the queue will kill the process. This is more of an option for safety of the cluster to prevent the user from crashing the cluster requesting too much memory.

mem_free_avg The amount of memory the job requires for averaging.

mem_max_avg The upper bound on the amount of memory the averaging job is allowed to use.

5.1.4 Other Cluster Options

job_name The job name prefix that will be used for the cluster submission scripts, log files, and error logs for the processing. Be careful that this name is unique because previous submission scripts, logs, and error logs with the same job name prefix will be overwritten in the case of a name collision.

array_max The maximum number of jobs per cluster submission script. Cluster submission scripts work using the array feature common to queuing systems, and this value is the maximum array size used in a script. If the user requests more batches of processing than this value, then the submission scripts will be split into files of up to array_max jobs.

max_jobs The maximum number of jobs for alignment. If the number of batches / exceeds this value the script will immediately quit.

run_local If the user wants to skip the cluster and run the job locally, this value should be set to 1.

skip_local_copy Set this option to 1 to skip the copying of data to local_dir.

5.1.5 Subtomogram Alignment Workflow Options

5.1.6 Parallelization Options

start_iteration The index of the reference to start from : input will be ref_fn_prefix_start_iteration.em and all_motl_fn_prefix_start_iteration.em (define as integer e.g. start_iteration=3)

More on iterations since they're confusing and it is slightly different here than from previous iterations.

The start_iteration is the beginning for the iteration variable used throughout this script. Iteration refers to iteration that is used for subtomogram alignment. So if start_iteration is 1, then subtomogram alignment will work using allmotl_1.em and ref_1.em. The output from alignment will be particle motls for the next iteration. This in the script is avg_iteration variable. The particle motls will be joined to form allmotl_2.em and then the parallel averaging will form ref_2.em and then the loop is done and iteration will become 2 and avg_iteration will become 3.

iterations Number iterations (big loop) to run: final output will be ref_fn_prefix_start_iteration+iterations.em and all_motl_fn_prefix_start_iteration+iterations.em

num_ali_batch The number of batches to split the parallel subtomogram alignment job into.

num_avg_batch The number of batches to split the parallel subtomogram averaging job into.

5.1.7 File Options

all_motl_fn_prefix Relative path and name of the concatenated motivelist of all particles (e.g. allmotl_iter.em , the variable will be written as a string e.g. all_motl_fn_prefix='sub-directory/allmotl')

ref_fn_prefix Relative path and name of the reference volumes (e.g. ref_iter.em , the variable will be written as a string e.g. ref_fn_prefix='sub-directory/ref')

ptcl_fn_prefix Relative path and name of the subtomograms (e.g. part_n.em , the variable will be written as a string e.g. ptcl_fn_prefix='sub-directory/part')

align_mask_fn Relative path and name of the alignment mask Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

cc_mask_fn Relative path and name of the cross-correlation mask this defines the maximum shifts in each direction Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

weight_fn_prefix Relative path and name of the weight file.

weight_sum_fn_prefix Relative path and name of the partial weight files.

5.1.8 Alignment and Averaging Options

tomo_row Which row in the motl file contains the correct tomogram number. Usually row 5 and 7 both correspond to the correct value and can be used interchangeably, but there are instances when 5 contains a sequential ordered value starting from 1, while 7 contains the correct corresponding tomogram.

apply_weight Apply weight to subtomograms (1=yes, 0=no).

apply_mask Apply mask to subtomograms (1=yes, 0=no).

psi_angle_step Angular increment in degrees, applied during the cone-search, i.e. psi and theta (define as real e.g. psi_angle_step=3). Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

psi_angle_shells Number of angular iterations, applied to psi and theta (define as integer e.g. psi_angle_shells=4). Note that in terms of cones this is twice the number of cones sampled. Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

phi_angle_step Angular increment for phi in degrees, (define as real e.g. phi_angle_step=3). Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

phi_angle_shells Number of angular iterations for phi, (define as integer e.g. phi_angle_shells=6). Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

high_pass_fp High pass filter cutoff (in transform units (pixels): calculate as (box_size * pixelsize) / (resolution_real) (define as integer). Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

high_pass_sigma High pass filter falloff sigma (in transform units (pixels): describes a Gaussian sigma for the falloff of the high-pass filter past the cutoff above. Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

low_pass_fp Low pass filter (in transform units (pixels): calculate as (box_size * pixelsize) / (resolution_real) (define as integer). Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

low_pass_sigma Low pass filter falloff sigma (in transform units (pixels): describes a Gaussian sigma for the falloff of the low-pass filter past the cutoff above. Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

nfold Symmetry, if no symmetry nfold=1 (define as integer e.g. nfold=3). Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

threshold Threshold for cross correlation coefficient. Only particles with ccc_new > threshold will be added to new average (define as real e.g. threshold=0.5). These particles will still be aligned at each iteration.

iclass Particles with that number in position 20 of motivelist will be added to new average (define as integer e.g. iclass=1). NOTES: Class 1 is ALWAYS added. Negative classes and class 2 are never added.

5.2 Example

```
scratch_dir="${PWD}"

local_dir=""

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

align_exec="${exec_dir}/alignment/subtom_scan_angles_exact"

cat_exec="${exec_dir}/MOTL/subtom_cat_motls"

sum_exec="${exec_dir}/alignment/subtom_parallel_sums"

avg_exec="${exec_dir}/alignment/subtom_weighted_average"

compare_exec="${exec_dir}/MOTL/subtom_compare_motls"

mem_free_ali=1G

mem_max_ali=64G

mem_free_avg=1G

mem_max_avg=64G

job_name=subTOM

array_max=1000

max_jobs=4000

run_local=0

skip_local_copy=1

start_iteration=1
```

(continues on next page)

(continued from previous page)

```
iterations=3

num_ali_batch=1

num_avg_batch=1

all_motl_fn_prefix="combinedmotl/allmotl"

ref_fn_prefix="ref/ref"

ptcl_fn_prefix="subtomograms/subtomo"

align_mask_fn=("otherinputs/align_mask_1.em" \
               "otherinputs/align_mask_2.em" \
               "otherinputs/align_mask_3.em")

cc_mask_fn=("otherinputs/cc_mask_r10.em" \
            "otherinputs/cc_mask_r05.em")

weight_fn_prefix="otherinputs/ampspec"

weight_sum_fn_prefix="otherinputs/wei"

tomo_row=7

apply_weight=0

apply_mask=1

psi_angle_step=(10 5 2.5)

psi_angle_shells=(4)

phi_angle_step=(20 5)

phi_angle_shells=(6)

high_pass_fp=(1)

high_pass_sigma=(2)

low_pass_fp=(12 15 18)

low_pass_sigma=(3)

nfold=(1 6)

threshold=-1

iclass=0
```

CHAPTER

SIX

SUBTOM_AVERAGE

Calculates the average from a given MOTL file in parallel on the cluster or locally.

This subtomogram averaging script uses five MATLAB compiled scripts below:

- *subtom_parallel_sums*
- *subtom_weighted_average*

6.1 Options

6.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

local_dir Absolute path to the folder on a group share, if the scratch directory is cleaned and deleted regularly this can set a local directory to which the important results will be copied. If this is not needed it can be skipped with the option skip_local_copy below.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

6.1.2 Variables

sum_exec Parallel Summing executable

avg_exec Weighted Averaging executable

6.1.3 Memory Options

mem_free The amount of memory the job requires for alignment. This variable determines whether a number of CPUs will be requested to be dedicated for each job. At 24G, one half of the CPUs on a node will be dedicated for each of the processes (12 CPUs). At 48G, all of the CPUs on the node will be dedicated for each of the processes (24 CPUs).

mem_max The upper bound on the amount of memory the alignment job is allowed to use. If any of the processes request or require more memory than this, the queue will kill the process. This is more of an option for safety of the cluster to prevent the user from crashing the cluster requesting too much memory.

6.1.4 Other Cluster Options

job_name The job name prefix that will be used for the cluster submission scripts, log files, and error logs for the processing. Be careful that this name is unique because previous submission scripts, logs, and error logs with the same job name prefix will be overwritten in the case of a name collision.

array_max The maximum number of jobs per cluster submission script. Cluster submission scripts work using the array feature common to queuing systems, and this value is the maximum array size used in a script. If the user requests more batches of processing than this value, then the submission scripts will be split into files of up to array_max jobs.

max_jobs The maximum number of jobs for alignment. If the number of batches / exceeds this value the script will immediately quit.

run_local If the user wants to skip the cluster and run the job locally, this value should be set to 1.

skip_local_copy Set this option to 1 to skip the copying of data to local_dir.

6.1.5 Subtomogram Averaging Workflow Options

6.1.6 Parallelization Options

iteration The index of the reference to generate : input will be all_motl_fn_prefix_iteration.em (define as integer)

num_avg_batch The number of batches to split the parallel subtomogram averaging job into.

6.1.7 File Options

all_motl_fn_prefix Relative path and name of the concatenated motivelist of all particles (e.g. allmotl_iter.em , the variable will be written as a string e.g. all_motl_fn_prefix='sub-directory/allmotl')

ref_fn_prefix Relative path and name of the reference volumes (e.g. ref_iter.em , the variable will be written as a string e.g. ref_fn_prefix='sub-directory/ref')

ptcl_fn_prefix Relative path and name of the subtomograms (e.g. part_n.em , the variable will be written as a string e.g. ptcl_fn_prefix='sub-directory/part')

weight_fn_prefix Relative path and name of the weight file.

weight_sum_fn_prefix Relative path and name of the partial weight files.

6.1.8 Averaging Options

tomo_row Which row in the motl file contains the correct tomogram number. Usually row 5 and 7 both correspond to the correct value and can be used interchangeably, but there are instances when 5 contains a sequential ordered value starting from 1, while 7 contains the correct corresponding tomogram.

iclass Particles with that number in position 20 of motivelist will be added to new average (define as integer e.g. iclass=1). NOTES: Class 1 is ALWAYS added. Negative classes and class 2 are never added.

6.2 Example

```
scratch_dir="${PWD}"

local_dir=""

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

sum_exec="${exec_dir}/alignment/subtom_parallel_sums"

avg_exec="${exec_dir}/alignment/subtom_weighted_average"

mem_free=1G

mem_max=64G

job_name=subTOM

array_max=1000

max_jobs=4000

run_local=0

skip_local_copy=1

iteration=1

num_avg_batch=1

all_motl_fn_prefix="combinedmotl/allmotl"

ref_fn_prefix="ref/ref"

ptcl_fn_prefix="subtomograms/subtomo"

weight_fn_prefix="otherinputs/ampspec"

weight_sum_fn_prefix="otherinputs/wei"

tomo_row=7

iclass=0
```

CHAPTER

SEVEN

SUBTOM_BANDPASS

Creates and/or applies a bandpass filter to a volume.

This utility script uses one MATLAB compiled script below:

- *subtom_bandpass*

7.1 Options

7.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

7.1.2 Variables

bandpass_exec Bandpass executable

7.1.3 File Options

input_motl_fn Relative path and name of the input volume to build and filter the bandpass against. If you just want to visualize an arbitrary filter you can use subtom_shape to create a template of the correct size and not ask for the filtered output.

filter_fn Relative path and name of the Fourier bandpass filter to write. If you do not want to output the filter volume simply leave this option blank.

output_fn Relative path and name of the filtered volume to write. If you do not want to output the filtered volume simply leave this option blank.

7.1.4 Filter Options

high_pass_fp High pass filter cutoff (in transform units (pixels): calculate as (box_size*pixelsize)/(resolution_real) (define as integer e.g. high_pass_fp=2)

high_pass_sigma High pass filter falloff sigma (in transform units (pixels): describes a Gaussian sigma for the falloff of the high-pass filter past the cutoff above.

low_pass_fp Low pass filter (in transform units (pixels): calculate as (box_size*pixelsize)/(resolution_real) (define as integer e.g. low_pass_fp=7).

low_pass_sigma Low pass filter falloff sigma (in transform units (pixels): describes a Gaussian sigma for the falloff of the low-pass filter past the cutoff above.

7.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

bandpass_exec="${exec_dir}/MOTL/subtom_unclass_motl"

input_fn="ref/ref_1.em"

filter_fn="otherinputs/bandpass_hp2s2_lp15s3.em"

output_fn="ref/ref_hp2s2_lp15s3_1.em"

high_pass_fp=2

high_pass_sigma=2

low_pass_fp=15

low_pass_sigma=3
```

CHAPTER

EIGHT

SUBTOM_CAT_MOTLS

Concatenate motive lists and print on the standard output.

This MOTL manipulation script uses one MATLAB compiled scripts below:

- *subtom_cat_motls*

8.1 Options

8.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

8.1.2 Variables

cat_exec Concatenate MOTLs executable

8.1.3 File Options

input_motl_fns Relative path and filename(s) of the input MOTL files to be concatenated. You can use shell wildcard characters * and ? to specify a given number of files and they will be expanded or you can just list the files one by one.

output_motl_fn Relative path and name of the output MOTL file. If you are not going to write an output file just set this variable to ‘’

output_star_fn Relative path and name of the output STAR file. If you are not going to write an output file just set this variable to ‘’

8.1.4 Concatenate Options

write_motl If you want to write out the concatenated MOTL files set this to 1, however if you just want to print the MOTL contents to the screen, set this to 0.

write_star If you want to write out the concatenated STAR file set this to 1, however if you just want to print the MOTL contents to the screen, set this to 0.

sort_row If you want to have the output MOTL file sorted by a particular field then specify it here. If the given value is not a value between 1-20 then the output MOTL file will be sorted arbitrarily based on the dir command in Matlab.

do_quiet If you just want to write output to files and not print to the screen set this to 1, however if you want to see the output printed to the screen leave this set to 0.

8.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

cat_exec="${exec_dir}/MOTL/subtom_cat_motls"

input_motl_fns=("combinedmotl/allmotl_1_tomo_1.em" \
                "combinedmotl/allmotl_1_tomo_2.em" \
                "combinedmotl/allmotl_1_tomo_3.em" \
                "combinedmotl/allmotl_1_tomo_4.em" \
                "combinedmotl/allmotl_1_tomo_5.em")

output_motl_fn="combinedmotl/allmotl_1.em"

output_star_fn="combinemotl/allmotl_1.star"

write_motl=1

write_star=0

sort_row=4

do_quiet=1
```

CHAPTER

NINE

SUBTOM_CLEAN_MOTL

Cleans a given MOTL file based on distance and/or CC scores.

This MOTL manipulation script uses one MATLAB compiled scripts below:

- *subtom_clean_motl*

9.1 Options

9.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

9.1.2 Variables

clean_motl_exec Clean MOTLs executable

9.1.3 File Options

input_motl_fn Relative path and name of the input MOTL file to be cleaned.

output_motl_fn Relative path and name of the output MOTL file.

output_stats_fn Relative path and name of the optional output cleaning stats CSV file. If you do not want to write out the differences just leave this as "".

The CSV format of the output statistics file for cleaning is a single row with the following columns:

Column	Value
1	Total Initial Number of Particles
2	Number of Particles Removed in Edge Cleaning
3	Number of Particles Removed in Cluster Cleaning
4	Number of Particles Removed in Distance Cleaning
5	Number of Particles Removed in CC Cleaning
6	Total Remaining Number of Particles

9.1.4 Clean Options

tomo_row Which row in the motl file contains the correct tomogram number. Usually row 5 and 7 both correspond to the correct value and can be used interchangeably, but there are instances when 5 contains a sequential ordered value starting from 1, while 7 contains the correct corresponding tomogram.

do_ccclean If the following is set to 1 then the MOTL will be cleaned by CCC either by CCC value or a fraction of the highest CCC values to keep. If it is set to 0 then CCC cleaning will be skipped.

cc_fraction If cleaning by CC then after edge, cluster, and distance cleaning, if any are selected, is completed. The MOTL will be sorted by CCC and then the top fraction as specified here will be kept with the rest discarded. For example if cc_fraction=0.7 the top 70% of the clean data will be kept and the bottom 30% of the cleaned data will be discarded. A value of 1 here means that data is not cleaned by CCC fraction.

cc_cutoff If cc_fraction is 1 and therefore not used then particles with a CCC below this cutoff will be removed from the output MOTL file. Values must be within -1 to 1, with -1 not removing any particles.

do_distance If the following is set to 1 then the MOTL will be cleaned by distance. If it is set to 0 distance cleaning will be skipped.

distance_cutoff Particles that are less than this distance in pixels from another particle will be cleaned with the particle with the highest CCC kept while the others are removed from the output MOTL file.

do_cluster If the following is set to 1 then the MOTL will be cleaned by a clustering criteria that enforces kept particles to exist as clusters. This can be useful when there is no lattice and clusters of particles makes a good indication that a true copy of the reference exists there. If it is set to 0 cluster cleaning will be skipped.

cluster_distance The following determines the radius that defines what is considered a cluster in cluster cleaning.

cluster_size The cluster size specifies how many particles must be found within cluster_distance for the particle to be considered part of a cluster. The particle with the highest CCC in the cluster will be selected as the representative particle for the cluster and the remaining clustered points will be removed.

do_edge If the following is set to 1 then the MOTL will be edge cleaned considering the dimensions of the tomogram in which the particles are contained. If any part of the particle exists outside of the tomogram it will be removed from the MOTL. If it is set to 0 edge cleaning will be skipped.

tomogram_dir Absolute path to the folder where the tomograms are stored. If you are not edge cleaning leave this set to “”.

box_size What is the box size of the particle that will be extracted from the tomogram, which is necessary to specify to be able to edge clean.

write_stats If the following is 1 then the details of how many particles were cleaned in each stage will be written out, if 0 then not.

9.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

clean_motl_exec="${exec_dir}/MOTL/subtom_clean_motl"

input_motl_fn="combinedmotl/allmotl_2.em"
```

(continues on next page)

(continued from previous page)

```
output_motl_fn="combinedmotl/allmotl_cc0.1_dist4_cluster2d10_2.em"

output_stats_fn="combinedmotl/allmotl_cc0.1_dist4_cluster2d10_stats.csv"

tomo_row=7

do_ccclean=1

cc_fraction=1

cc_cutoff=0.1

do_distance=1

distance_cutoff=4

do_cluster=1

cluster_distance=10

cluster_size=2

do_edge=1

tomogram_dir="/net/dstore2/teraraid/dmorado/subTOM_tutorial/data/tomos/bin8"

box_size=36

write_stats=1
```

CHAPTER

TEN

SUBTOM_COMPARE_MOTLS

Compares the translations and rotations between two MOTLS of different iterations of alignment.

This MOTL manipulation script uses one MATLAB compiled scripts below:

- *subtom_compare_motls*

10.1 Options

10.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

10.1.2 Variables

compare_motls_exec Comparison MOTLs executable

10.1.3 File Options

motl_1_fn Relative path and name of the first MOTL file

motl_2_fn Relative path and name of the second MOTL file

output_diffs_fn Relative path and name of the optional output difference CSV file. If you do not want to write out the differences just leave this as "".

The CSV format of the output differences file for comparison is one row per particle in the motive list that has six columns, and a special final line with 22 columns for statistics of differences for the whole motive list. The particle columns are as follows:

Column	Value
1	Particle Index (Motive List row 4)
2	CCC Score for particle in first motive list
3	CCC Score for particle in second motive list
4	Coordinate displacement between motive lists
5	Angular displacement between motive lists
6	Angular displacement ignoring inplane rotations

The special final line columns are as follows:

Column	Value
1	Mean Coordinate displacement between MOTLs
2	Median Coordinate displacement between MOTLs
3	Coordinate displacement standard deviation
4	Maximum Coordinate displacement between MOTLs
5	Mean Angular displacement between MOTLs
6	Median Angular displacement between MOTLs
7	Angular displacement standard deviation
8	Maximum Angular displacement between MOTLs
9	Same as 5 but ignoring inplane rotations
10	Same as 6 but ignoring inplane rotations
11	Same as 7 but ignoring inplane rotations
12	Same as 8 but ignoring inplane rotations
13	Mean CCC score in the first motive list
14	Median CCC score in the first motive list
15	CCC standard deviation in first motive list
16	Minimum CCC score in the first motive list
17	Maximum CCC score in the first motive list
18	Mean CCC score in the second motive list
19	Median CCC score in the second motive list
20	CCC standard deviation in second motive list
21	Minimum CCC score in the second motive list
22	Maximum CCC score in the second motive list

10.1.4 Comparison Options

write_diffs If the following is 1 then the differences will be written out, if 0 then not.

10.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

compare_motls_exec="${exec_dir}/MOTL/subtom_compare_motls"

motl_1_fn="combinedmotl/allmotl_1.em"

motl_2_fn="combinedmotl/allmotl_2.em"

output_diffs_fn="combinedmotl/allmotl_1_2_diffs.csv"

write_diffs=1
```

CHAPTER

ELEVEN

SUBTOM_EVEN_ODD_MOTL

Splits a given MOTL file into even/odd halves for gold-standard refinement.

This MOTL manipulation script uses one MATLAB compiled scripts below:

- *subtom_even_odd_motl*

11.1 Options

11.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

11.1.2 Variables

even_odd_motl_exec Even-Odd split motive list executable

11.1.3 File Options

input_motl_fn Relative path and name of the input MOTL file to be split.

output_motl_fn Relative path and name of the output MOTL file where the even and odd halves are specified by the class number in the 20th row of the motive list. The even half inherits the current class number plus 200 and the odd half inherits the current class numbers plus 100.

even_motl_fn Relative path and name of the output even MOTL file.

odd_motl_fn Relative path and name of the output odd MOTL file.

11.1.4 Even / Odd Options

split_row The following specifies which row of the MOTL will be used to split the data. To simply split into even and odd halves use the particle running ID, which is row 4. To split the halves by tomogram use row 5 or 7, and to split the halves by tube or sphere use row 6.

11.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

even_odd_exec="${exec_dir}/MOTL/subtom_even_odd_motl"

input_motl_fn="combinedmotl/allmotl_1.em"

output_motl_fn="combinedmotl/allmotl_eo_1.em"

even_motl_fn="even/combinedmotl/allmotl_1.em"

odd_motl_fn="odd/combinedmotl/allmotl_1.em"

split_row=4
```

CHAPTER

TWELVE

SUBTOM_EXTRACT_NOISE

This script finds and extracts noise particles from tomograms and generates amplitude spectrum volumes for used in Fourier reweighting of particles in the subtomogram alignment and averaging routines, as a Fourier weight in place of a traditional binary-wedge. Also generates an estimated binary wedge as well from the noise.

It also generates a noise motl file so that the noise positions found in binned tomograms can then be used later on in less or unbinned tomograms and after some positions have been cleaned, which could make it more difficult to pick non-structural noise in the tomogram.

This tomogram extraction script uses one MATLAB compiled scripts below:

- *subtom_extract_noise*

12.1 Options

12.1.1 Directories

tomogram_dir Absolute path to the folder where the tomograms are stored

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

12.1.2 Variables

noise_extract_exe Noise extraction executable

motl_dump_exe MOTL dump executable

12.1.3 Memory Options

mem_free The amount of memory the job requires for alignment. This variable determines whether a number of CPUs will be requested to be dedicated for each job. At 24G, one half of the CPUs on a node will be dedicated for each of the processes (12 CPUs). At 48G, all of the CPUs on the node will be dedicated for each of the processes (24 CPUs).

mem_max The upper bound on the amount of memory the alignment job is allowed to use. If any of the processes request or require more memory than this, the queue will kill the process. This is more of an option for safety of the cluster to prevent the user from crashing the cluster requesting too much memory.

12.1.4 Other Cluster Options

job_name The job name prefix that will be used for the cluster submission scripts, log files, and error logs for the processing. Be careful that this name is unique because previous submission scripts, logs, and error logs with the same job name prefix will be overwritten in the case of a name collision.

run_local If the user wants to skip the cluster and run the job locally, this value should be set to 1.

12.1.5 Noise Extraction Workflow Options

12.1.6 File Options

iteration The iteration of the all particle motive list to extract from : input will be all_motl_fn_prefix_iteration.em (define as integer)

all_motl_fn_prefix Relative path to allmotl file from root folder.

noise_motl_fn_prefix Relative path to noisemotl filename. If the file doesn't exist a new one will be written with the determined noise positions. If a previously existing noise motl exists it will be used instead. If the number of noise particles requested has been increased new particles will be found and added and the file will be updated.

ampspec_fn_prefix Relative path and filename prefix for output amplitude spectrums

binary_fn_prefix Relative path and filename prefix for output binary wedges

12.1.7 Tomogram Options

tomo_row Which row in the motl file contains the correct tomogram number. Usually row 5 and 7 both correspond to the correct value and can be used interchangeably, but there are instances when 5 contains a sequential ordered value starting from 1, while 7 contains the correct corresponding tomogram.

12.1.8 Extraction Options

box_size Size of subtomogram in pixels

just_extract If you already have noise MOTL lists calculated which may contain less than the total number of requested noise, but just want the code to do the extraction then you can set just_extract to 1. Otherwise set it to 0.

ptcl_overlap_factor The amount of overlap to allow between noise particles and subtomograms Numbers less than 0 will allow for larger than a box size spacing between noise and a particle. Numbers greater than 0 will allow for some overlap between noise and a particle. For example 0.5 will allow 50% overlap between the noise and the particle, which can be useful when the box size is much larger than the particle.

noise_overlap_factor The amount of overlap to allow between noise particles Numbers less than 0 will allow for larger than a box size spacing between noise. Numbers greater than 0 will allow for some overlap between noise. For example 0.75 will allow 75% overlap between the noise, which can be useful when there is not much space for enough noise.

num_noise Number of noise particles to extract.

reextract Set reextract to 1 if you want to force the program to re-extract amplitude spectra even if the amplitude spectrum file already exists.

preload_tomogram Set preload_tomogram to 1 if you want to read the whole tomogram into memory before extraction. This is the fastest way to extract particles however the system needs to be able to have the memory to fit the whole tomogram into memory or otherwise it will crash. If it is set to 0, then either the subtomograms can be extracted using a memory-map to the data, or read directly from the file.

use_tom_red Set use_tom_red to 1 if you want to use the AV3/TOM function tom_red to extract particles. This requires that preload_tomogram above is set to 1. This is the original way to extract particles, but it seemed to sometimes produce subtomograms that were incorrectly sized. If it is set to 0 then an inlined window function is used instead.

use_memmap Set use_memmap to 1 to memory-map the tomogram and read subtomograms from this map. This appears to be a little slower than having the tomogram fully in memory without the massive memory footprint. However, it also appears to be slightly unstable and may crash unexpectedly. If it is set to 0 and preload_tomogram is also 0, then subtomograms will be read directly from the tomogram on disk. This also requires much less memory, however it appears to be extremely slow, so this only makes sense for a large number of tomograms being extracted on the cluster.

12.2 Example

```
tomogram_dir="/net/dstore2/teraraid/dmorado/subTOM_tutorial/data/tomos/bin8"

scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

noise_extract_exe="${exec_dir}/alignment/subtom_extract_noise"

motl_dump_exe="${exec_dir}/MOTL/motl_dump"

mem_free="1G"

mem_max="64G"

job_name="subTOM"

run_local=0

iteration=1

all_motl_fn_prefix="combinedmotl/allmotl"

noise_motl_fn_prefix="combinedmotl/noisemotl"

ampspec_fn_prefix="otherinputs/ampspec"

binary_fn_prefix="otherinputs/binary"

tomo_row=7

box_size=128

just_extract=0

ptcl_overlap_factor=0
```

(continues on next page)

(continued from previous page)

```
noise_overlap_factor=0.75

num_noise=1000

reextract=0

preload_tomogram=1

use_tom_red=0

use_memmap=0
```

CHAPTER

THIRTEEN

SUBTOM_EXTRACT_SUBTOMOGRAMS

This script takes an input number of cores, and on each core extract one tomogram at a time as written in a specified row of the all motive list. Parallelization works by writing a start file upon openinig of a tomo, and a completion file. After tomogram extraction, it moves on to the next tomogram that hasn't been started.

This tomogram extraction script uses one MATLAB compiled scripts below:

- *subtom_extract_subtomograms*

13.1 Options

13.1.1 Directories

tomogram_dir Absolute path to the folder where the tomograms are stored

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

13.1.2 Variables

extract_exe Subtomogram extraction executable

motl_dump_exe MOTL dump executable

13.1.3 Memory Options

mem_free The amount of memory the job requires for alignment. This variable determines whether a number of CPUs will be requested to be dedicated for each job. At 24G, one half of the CPUs on a node will be dedicated for each of the processes (12 CPUs). At 48G, all of the CPUs on the node will be dedicated for each of the processes (24 CPUs).

mem_max The upper bound on the amount of memory the alignment job is allowed to use. If any of the processes request or require more memory than this, the queue will kill the process. This is more of an option for safety of the cluster to prevent the user from crashing the cluster requesting too much memory.

13.1.4 Other Cluster Options

job_name The job name prefix that will be used for the cluster submission scripts, log files, and error logs for the processing. Be careful that this name is unique because previous submission scripts, logs, and error logs with the same job name prefix will be overwritten in the case of a name collision.

run_local If the user wants to skip the cluster and run the job locally, this value should be set to 1.

13.1.5 Subtomogram Extraction Workflow Options

13.1.6 File Options

iteration The iteration of the all particle motive list to extract from : input will be all_motl_fn_prefix_iteration.em (define as integer)

all_motl_fn_prefix Relative path to allmotl file from root folder.

subtomo_fn_prefix Relative path and filename for output subtomograms.

stats_fn_prefix Relative path and filename for stats .csv files.

The CSV format of the subtomogram stats is a single file for each tomogram with one line per particle in the tomogram with six columns. The particle columns are as follows:

Column	Value
1	Particle Index (Motive List row 4)
2	Mean value for the subtomogram
3	Maximum value in the subtomogram
4	Minimum value in the subtomogram
5	Standard deviation of values in the subtomogram
6	Variance of values in the subtomogram

13.1.7 Tomogram Options

tomo_row Which row in the motl file contains the correct tomogram number. Usually row 5 and 7 both correspond to the correct value and can be used interchangeably, but there are instances when 5 contains a sequential ordered value starting from 1, while 7 contains the correct corresponding tomogram.

13.1.8 Extraction Options

box_size Size of subtomogram in pixels

subtomo_digits Leading zeros for subtomograms, for AV3, use 1. Other numbers are useful for DYNAMO.

reextract Set reextract to 1 if you want to force the program to re-extract subtomograms even if the stats file and the subtomograms already exist. If the stats file for the tomogram exists and is the correct size the whole tomogram will be skipped. If the subtomogram exists it will also be skipped, unless this option is true.

preload_tomogram Set preload_tomogram to 1 if you want to read the whole tomogram into memory before extraction. This is the fastest way to extract particles however the system needs to be able to have the memory to fit the whole tomogram into memory or otherwise it will crash. If it is set to 0, then either the subtomograms can be extracted using a memory-map to the data, or read directly from the file.

use_tom_red Set use_tom_red to 1 if you want to use the AV3/TOM function tom_red to extract particles. This requires that preload_tomogram above is set to 1. This is the original way to extract particles, but it seemed to sometimes produce subtomograms that were incorrectly sized. If it is set to 0 then an inlined window function is used instead.

use_memmap Set use_memmap to 1 to memory-map the tomogram and read subtomograms from this map. This appears to be a little slower than having the tomogram fully in memory without the massive memory footprint. However, it also appears to be slightly unstable and may crash unexpectedly. If it is set to 0 and preload_tomogram is also 0, then subtomograms will be read directly from the tomogram on disk. This also requires much less memory, however it appears to be extremely slow, so this only makes sense for a large number of tomograms being extracted on the cluster.

13.2 Example

```
tomogram_dir="/net/dstore2/teraraid/dmorado/subTOM_tutorial/data/tomos/bin8"

scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

extract_exe="${exec_dir}/alignment/subtom_extract_subtomograms"

motl_dump_exe="${exec_dir}/MOTL/motl_dump"

mem_free="1G"

mem_max="64G"

job_name="subTOM"

run_local=0

iteration=1

all_motl_fn_prefix="combinedmotl/allmotl"

subtomo_fn_prefix="subtomograms/subtomo"

stats_fn_prefix="subtomograms/stats/tomo"

tomo_row=7

box_size=128

subtomo_digits=1

reextract=0

preload_tomogram=1
```

(continues on next page)

(continued from previous page)

```
use_tom_red=0

use_memmap=0
```

CHAPTER

FOURTEEN

SUBTOM_MASKCORRECTED_FSC

Calculates a *"mask-corrected"* Fourier Shell Correlation between two volumes and generates a final average as well as optionally ad-hoc B-factor sharpened maps.

This script is meant to run on a local workstation with access to an X server in the case when the user wants to display figures. I am unsure if both plotting options are disabled if the graphics display is still required, but if not it could be run remotely on the cluster, but it shouldn't be necessary.

This EM-map analysis script uses just one MATLAB compiled scripts below:

- *subtom_maskcorrected_FSC*

14.1 Options

14.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables.

14.1.2 Variables

fsc_exec Mask-corrected FSC executable.

14.1.3 File Options

ref_a_fn_prefix Relative path and filename prefix of the first half-map.

ref_b_fn_prefix Relative path and filename prefix of the second half-map.

iteration The index of the reference to generate : input will be ref_{a,b}_fn_prefix_iteration.em (define as integer).

fsc_mask_fn Relative path and name of the FSC mask.

filter_a_fn Relative path and name of the Fourier filter volume for the first half-map. If not using the option do_reweight just leave this set to ""

filter_b_fn Relative path and name of the Fourier filter volume for the second half-map. If not using the option do_reweight just leave this set to ""

output_fn_prefix Relative path and prefix for the name of the output maps and figures.

14.1.4 FSC Options

pixelsize Pixelsize of the half-maps in Angstroms.

nfold Symmetry to applied the half-maps before calculating FSC (1 is no symmetry).

rand_threshold The Fourier pixel at which phase-randomization begins is set automatically to the point where the unmasked FSC falls below this threshold.

plot_fsc Plot the FSC curves - 1 = yes, 0 = no

14.1.5 Sharpening Options

do_sharpen Set to 1 to sharpen map or 0 to skip and just calculate the FSC.

b_factor B-Factor to be applied; must be negative or zero.

box_gaussian To remove some of the edge-artifacts associated with map-sharpening the edges of the map can be smoothed with a gaussian. Set to 0 to not smooth the edges, otherwise it must be set to an odd number.

filter_mode There are two mode used for low pass filtering. The first uses an FSC based threshold (mode 1), i.e. after FSC < 0.143, or a pixel-based resolution threhsold (mode 2).

filther_threshold Set the threshold for the low pass filtering described above. Should be less than 1 for FSC based threshold (mode 1), and an integer value for the Fourier pixel-based threshold (mode 2).

plot_sharpen Plot the sharpening curve - 1 = yes, 0 = no.

14.1.6 Reweighting Options

do_reweight Set to 1 to apply the externally calculated Fourier weights filter_A_fn and filter_B_fn to each half-map to reweight the final output map.

14.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

fsc_exec="${exec_dir}/analysis/subtom_maskcorrected_fsc"

ref_a_fn_prefix="even/ref/ref"

ref_b_fn_prefix="odd/ref/ref"

iteration=1

fsc_mask_fn="FSC/fsc_mask.em"

filter_a_fn=""
```

(continues on next page)

(continued from previous page)

```
filter_b_fn=""

output_fn_prefix="FSC/ref"

pixelsize=1

nfold=1

rand_threshold=0.8

plot_fsc=1

do_sharpen=1

b_factor=-150

box_gaussian=3

filter_mode=1

filter_threshold=0.143

plot_sharpen=1

do_reweight=0
```

CHAPTER

FIFTEEN

SUBTOM_PREPROCESS

Aligns dose-fractionated data, sorts and stacks aligned frames, determines the defocus of the tilt-series using CTFFIND4, GCTF, or IMOD CTFPLOTTER and then dose-filters the tilt-series in prepartion for alignment using IMOD/eTomo.

15.1 Options

15.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

frame_dir Relative path to the folder where the dose-fractionated movie frames are located.

15.1.2 Executables

alignframes_exe Absolute path to the IMOD alignframes executable. The directory of this will be used for the other IMOD programs used in the processing. Need version at least above 4.10.29

ctffind_exe Absolute path to the CTFFIND4 executable. Needs version at least above 4.1.13.

gctf_exe Absolute path to the GCTF executable. I wouldn't use it because it rarely works but it seems a version of 1.06 sometimes doesn't crash.

exec_dir Directory for subTOM executables

15.1.3 Memory Options

mem_free The amount of memory the job requires for alignment. This variable determines whether a number of CPUs will be requested to be dedicated for each job. At 24G, one half of the CPUs on a node will be dedicated for each of the processes (12 CPUs). At 48G, all of the CPUs on the node will be dedicated for each of the processes (24 CPUs).

mem_max The upper bound on the amount of memory the alignment job is allowed to use. If any of the processes request or require more memory than this, the queue will kill the process. This is more of an option for safety of the cluster to prevent the user from crashing the cluster requesting too much memory.

15.1.4 Other Cluster Options

job_name The job name prefix that will be used for the cluster submission scripts, log files, and error logs for the processing. Be careful that this name is unique because previous submission scripts, logs, and error logs with the same job name prefix will be overwritten in the case of a name collision.

run_local If the user wants to skip the cluster and run the job locally, this value should be set to 1.

15.1.5 File Options

ts_fmt The format string for the datasets to process. The string XXXIDXXXX will be replaced with the numbers specified between the range start_idx and end_idx.

The raw sum tilt-series will have the name format ts_fmt.st, or ts_fmt.mrc, an extended data file ts_fmt.{mrc,st}.mdoc and could possibly have an associated log ts_fmt.log.

Dose-fractionated movies of tilt-images are assumed to have the name format: ts_fmt_###_*.{mrc,tif} where ### is a running three-digit ID number for the tilt-image and * is the tilt-angle.

start_idx The first tilt-series to operate on.

end_idx The last tilt-series to operate on.

idx_fmt The format string for the tomogram indexes. Likely two or three digit zero padding or maybe just flat integers.

15.1.6 Beam Induced Motion Correction Options

do_aligned If you want to run alignframes to generate the non-dose-weighted tiltseries set this option to 1 and if you want to skip this step set this option to 0.

do_doseweight If you want to run alignframes to generate the dose-weighted tiltseries set this option to 1 and if you want to skip this step set this option to 0.

do_gain_correction Determines whether or not gain-correction needs to be done on the frames. Set to 1 to apply gain-correction during motion-correction, and 0 to skip it. Normally TIFF format frames will be saved with compression and will be unnormalized, and should be gain-corrected. MRC format frames are generally already saved with gain-correction applied during collection, so it can be skipped here.

A good rule of thumb, is if you have a dm4 file in your data you need to do gain-correction, and if you don't see a dm4 file you do not.

gainref_fn The path to the gain-reference file, this will only be used if gain_correction is going to be applied.

defects_fn The path to the defects file, this is saved along with the gain-reference for unnormalized saved frames by SerialEM, and will only be used if gain-correction is going to be applied.

align_bin Binning to apply to the frames when calculating the alignment, if you are using super-resolution you may want to change this to 2. The defaults from IMOD would be 3 for counted data and 6 for super-resolution data. Multiple binnings can be tested and the best one will be used to generate the final sum.

sum_bin Binning to apply to the final sum. This is done using Fourier cropping as in MotionCorr and other similar programs. If you are using super-resolution you probably want to change this to 2, otherwise it should be set to 1.

scale Amount of scaling to apply to summed values before output. The default is 30 however serialEM applies one of 39.3?

filter_radius2 Cutoff Frequency for the lowpass filter used in frame alignment. The unit is absolute spatial frequency which goes from 0 to 0.5 relative to the pixelsize of the input frames (not considering binning applied in alignment). The default from IMOD is 0.06. Multiple radii can be used and the best filter will be selected for the actually used alignment.

filter_sigma2 Falloff for the lowpass filter used in frame alignment. Same units as above. The defaults from IMOD is 0.0086.

shift_limit Limit on distance to search for correlation peak in unbinned pixels. The default from IMOD is 20.

do_refinement If this is set to 1, alignframes will do an iterative refinement of the initially found frame alignment solution. The default in IMOD is to not do this refinement.

refine_iterations The maximum number of refinement iterations to run.

refine_radius2 Cutoff Frequency for the lowpass filter used in refinement. The default in IMOD would be to use the same value used in alignment.

refine_shift_stop The amount of shift at which refinement will stop in unbinned pixels.

truncate_above Movies often contain hot pixels not removed from the pixel-defect mask either from x-rays or other factors and these throw off the later scaling of sums. Traditionally they would be removed in eTomo using the ccderaser command / step, but it has been found to go better to truncate them at the frame-alignment and summing step. To find a reasonable value to truncate above use the command 'clip stats' on several movies to find out where the values start to become outliers, it should be around 5-7 for 10 frame movies of about 3e/A^2 on the K2.

use_gpu If you want to use a GPU set this to 1, but be careful to not use both the cluster and the GPU as this is not supported.

extra_opts If you want to use other options to alignframes specify them here.

15.1.7 CTF Estimation Options

apix The pixel size of the raw movie frames if they exist, or the pixelsize of the "_aligned.st" stack if alignframes and dose-weighting is not being done. The actual pixelsize used in CTF estimation is apix * sum_bin.

do_ctffind4 If this is set to 1, the defocus will be estimated with CTFFIND4.

do_gctf If this is set to 1, the defocus will be estimated with GCTF.

do_ctfplotter If this is set to 1, the defocus will be estimated with CTFPLOTTER.

voltage_kev The accelerating voltage of the microscope in KeV.

cs The spherical aberration of the microscope in mm.

ac The amount of amplitude contrast in the imaging system.

tile_size The size of tile to operate on.

min_res The lowest wavelength in Angstroms to allow in fitting (minimum resolution).

max_res The highest wavelength in Angstroms to allow in fitting (maximum resolution).

min_res_ctfplotter The lowest wavelength in Angstroms to allow in fitting in CTFPLOTTER.

max_res_ctfplotter The highest wavelength in Angstroms to allow in fitting in CTFPLOTTER.

min_def The lowest defocus in Angstroms to scan.

max_def The highest defocus in Angstroms to scan.

def_step The step size in Angstroms to scan defocus.

astigmatism The amount of astigmatism to allow in Angstroms.

tilt_axis_angle The tilt-axis angle of the tilt series. This is only needed if you are estimating the CTF with ctfplotter. You can find this value running the command 'header' on the raw sum tiltseries and looking at the first label (Titles) in the header.

15.1.8 Dose Filtering Options

dose_per_tilt The dose per micrograph in Electrons per square Angstrom.

15.2 Example

```
scratch_dir="${PWD}"

frame_dir="Frames"

alignframes_exe="$(which alignframes)"

ctffind_exe="$(which ctffind)"

gctf_exe="$(which Gctf)"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

mem_free="1G"

mem_max="64G"

job_name="subTOM"

run_local=1

ts_fmt="TS_XXXIDXXXX"

start_idx=1

end_idx=1

idx_fmt="%02d"

do_aligned=1

do_doseweight=1

do_gain_correction=1

gainref_fn="Frames/gainref.dm4"

defects_fn="Frames/defects.txt"

align_bin=1,2,3
```

(continues on next page)

(continued from previous page)

```
sum_bin=1

scale=39.3

filter_radius2=0.167,0.125,0.10,0.06

filter_sigma2=0.0086

shift_limit=20

do_refinement=1

refine_iterations=5

refine_radius2=0.167

refine_shift_stop=0.1

truncate_above=7

use_gpu=0

extra_opts=''

apix=1

do_ctffind4=1

do_gctf=0

do_ctfplotter=1

voltage_kev=300.0

cs=2.7

ac=0.07

tile_size=512

min_res=30.0

max_res=5.0

min_res_ctfplotter=50.0

max_res_ctfplotter=10.0

min_def=10000.0

max_def=60000.0
```

(continues on next page)

(continued from previous page)

```
def_step=100.0

astigmatism=1000.0

tilt_axis_angle=85.3

dose_per_tilt=3.5
```

CHAPTER

SIXTEEN

SUBTOM_RANDOM_SUBSET_MOTL

Draws a random subset from a given MOTL file.

This MOTL manipulation script uses one MATLAB compiled scripts below:

- *subtom_random_subset_motl*

16.1 Options

16.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

16.1.2 Variables

random_subset_motl_exec Random subset motive list executable

16.1.3 File Options

input_motl_fn Relative path and name of the input MOTL file to draw the subset from.

output_motl_fn Relative path and name of the output MOTL file.

16.1.4 Subset Options

subset_size How many particles to be included in the subset.

subset_row The following describes a field in the MOTL to equally distribute particles of the subset amongst. Such that if subset_row was the tomogram row (7), and there were ten tomograms described in the motive list, then the subset of 1000 particles would have 100 particles from each tomogram. If there are more unique values than the subset size then the field is not taken into account.

16.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

random_subset_motl_exec="${exec_dir}/MOTL/subtom_random_subset_motl"

input_motl_fn="combinedmotl/allmotl_1.em"

output_motl_fn="combinedmotl/s5kmotl_1.em"

subset_size=5000

subset_row=7
```

CHAPTER

SEVENTEEN

SUBTOM_PLOT_FILTER

Plots the filter applied to the reference from a user-specified set of band-pass settings. The filter can also be plotted in conjunction with a CTF root square function and a B-factor described exponential decay falloff curve. The plot can also be saved to disk.

This utility script uses one MATLAB compiled scripts below:

- *subtom_plot_filter*

17.1 Options

17.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

17.1.2 Variables

plot_filter_exec Plot filter executable.

17.1.3 File Options

output_fn_prefix Relative path and name prefix of the output plot. If you want to skip this output file leave this set to "".

17.1.4 Plot Filter Options

box_size Size of the volume in pixels. The volume will be a cube with this side length.

pixelsize Pixelsize of the data in Angstroms.

high_pass_fp

High pass filter cutoff (in transform units (pixels): calculate as:

```
high_pass_fp = (box_size * pixelsize) / (high_pass_A)
```

(define as integer e.g. high_pass_fp=2)

high_pass_sigma High pass filter falloff sigma (in transform units (pixels): describes a Gaussian sigma for the falloff of the high-pass filter past the cutoff above.

low_pass_fp

Low pass filter cutoff (in transform units (pixels): calculate as:

```
low_pass_fp = (box_size * pixelsize) / (low_pass_A)
```

(define as integer e.g. low_pass_fp=48)

low_pass_sigma Low pass filter falloff sigma (in transform units (pixels): describes a Gaussian sigma for the falloff of the low-pass filter past the cutoff above.

defocus Defocus to plot along with band-pass filter in Angstroms with underfocus being positive. The graphic will include a line for the CTF root square and how it is attenuated by the band-pass which can be useful for understanding how amplitudes are modified by the filter. If you do not want to use this option just leave it set to "0" or "".

voltage Voltage in keV used for calculating the CTF. If you do not want to plot a CTF function leave this set to "" or "300".

cs Spherical aberration in mm used for calculating the CTF. If you do not want to plot a CTF function leave this set to "" or "0.0".

ac Amplitude contrast as a fraction of contrast (i.e. between 0 and 1) used for calculating the CTF. If you do not want to plot a CTF function leave this set to "" or "1".

phase_shift Phase shift in degrees used for calculating the CTF. If you do not want to plot a CTF function leave this set to "" or "0".

b_factor B-Factor describing the falloff of signal in the data by a multitude of amplitude decay factors. The graphic will include a line for the falloff and how it interacts with both the CTF if one was given and the band-pass filter. If you do not want to use this option just leave it set to "" or "0"

17.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

plot_filter_exec="${exec_dir}/utils/subtom_plot_filter"

output_fn_prefix=""

box_size="192"

pixelsize="1.35"

high_pass_fp="1"

high_pass_sigma="2"

low_pass_fp="48"
```

(continues on next page)

(continued from previous page)

```
low_pass_sigma="3"

defocus="15000"

voltage="300"

cs="2.7"

ac="0.07"

phase_shift="0.0"

b_factor="-130"
```

CHAPTER

EIGHTEEN

SUBTOM_PLOT_SCANNED_ANGLES

Plots the angles searched for a user-specified set of alignment angles. The angles can also be centered about a given initial orientation. The marker of the plot can be adjusted and the plot can also be saved to disk.

This utility script uses one MATLAB compiled scripts below:

- *subtom_plot_scanned_angles*

18.1 Options

18.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

18.1.2 Variables

plot_angles_exec Plot scanned angles executable.

18.1.3 File Options

output_fn_prefix Relative path and name prefix of the output plot. If you want to skip this output file leave this set to "".

18.1.4 Plot Scanned Angles Options

psi_angle_step Angular increment in degrees, applied during the cone-search, i.e. psi and theta (define as real e.g. psi_angle_step=3)

psi_angle_shells Number of angular iterations, applied to psi and theta (define as integer e.g. psi_angle_shells=3)

phi_angle_step Angular increment for phi in degrees, (define as real e.g. phi_angle_step=3)

phi_angle_shells Number of angular iterations for phi, (define as integer e.g. phi_angle_shells=3)

initial_phi Initial first Euler angle rotation around the Z-axis about which the scanned angles are centered. (define as real e.g. initial_phi=45).

initial_psi Initial third Euler angle rotation around the Z-axis about which the scanned angles are centered. (define as real e.g. initial_psi=30).

initial_theta Initial second Euler angle rotation around the X-axis about which the scanned angles are centered. (define as real e.g. initial_theta=135).

angle_fmt If the above angles are specified as degress leave this set to 'degrees', but if the angles above are in radian format set this to 'radians'.

marker_size Set the marker size of the arrows that are drawn for the rotations, reasonable values seem to be around the range of 0.01 to 0.1.

18.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

plot_angles_exec="${exec_dir}/utils/subtom_plot_scanned_angles"

output_fn_prefix=""

psi_angle_step="6"

psi_angle_shells="7"

phi_angle_step="6"

phi_angle_shells="7"

initial_phi="0"

initial_psi="0"

initial_theta="0"

angle_fmt="degrees"

marker_size="0.02"
```

CHAPTER

NINETEEN

SUBTOM_RECONSTRUCT

This is a run script for the reconstruction and possibly also CTF correction processing of electron cryo-tomograhpy data by means of the program novaCTF.

This script is meant to run on a local workstation but can also submit some of the processing to the cluster so that data can be preprocessed in parallel. However, note that the read/write density of operations in novaCTF is extremely large and therefore care should be taken to not overload systems, or be prepared to have a very slow connection to your filesystem.

19.1 Options

19.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

19.1.2 Executables

novactf_exe Absolute path to the novaCTF executable.

newstack_exe Absolute path to the IMOD newstack executable. The directory of this will be used for the other IMOD programs used in the processing.

exec_dir Directory for subTOM executables.

19.1.3 Memory Options

mem_free The amount of memory the job requires for alignment. This variable determines whether a number of CPUs will be requested to be dedicated for each job. At 24G, one half of the CPUs on a node will be dedicated for each of the processes (12 CPUs). At 48G, all of the CPUs on the node will be dedicated for each of the processes (24 CPUs).

mem_max The upper bound on the amount of memory the alignment job is allowed to use. If any of the processes request or require more memory than this, the queue will kill the process. This is more of an option for safety of the cluster to prevent the user from crashing the cluster requesting too much memory.

num_threads Set this value to the number of jobs you want to run in the background before reconstruction. Should be the number of threads on the local system or cluster which for our system is 24 on the cluster and higher on the local systems, but there you should be polite!

19.1.4 Other Cluster Options

job_name The job name prefix that will be used for the cluster submission scripts, log files, and error logs for the processing. Be careful that this name is unique because previous submission scripts, logs, and error logs with the same job name prefix will be overwritten in the case of a name collision.

run_local If the user wants to skip the cluster and run the job locally, this value should be set to 1.

19.1.5 File Options

tomo_fmt The format string for the datasets to process. The string XXXIDXXXX will be replaced with the numbers specified between the range start_idx and end_idx.

tomo_dir_fmt The format string for the directory of datasets to process. The string XXXIDXXXX will be replaced with the numbers specified between the range start_idx and end_idx.

start_idx The first tomogram to operate on.

end_idx The last tomogram to operate on.

idx_fmt The format string for the tomogram indexes. Likely two or three digit zero padding or maybe just flat integers.

19.1.6 General CTF Options

defocus_file Where the defocus list file is located. The string XXXIDXXXX will be replaced with the formatted tomogram index, i.e. XXXIDXXXX_output.txt will be turned into 01_output.txt.

pixel_size The pixel size of the tilt series in nanometers. *Note NANOMETERS!*

amplitude_contrast The amplitude contrast for CTF correction.

cs The spherical aberration of the microscope in mm for CTF correction.

voltage The voltage in KeV of the microscope for CTF correction.

19.1.7 Nova 3D-CTF Options

do_3dctf Set this value to 1 if you want to do 3D-CTF correction during the reconstruction of the tomograms. If this value is set to 0 NovaCTF will still be used but it will generate tomograms largely identical to IMOD's WBP.

correction_type Type of CTF correction to perform.

defocus_file_format File format for the defocus list. Use ctffind4 for CTFFIND4, imod for CTFPLOTTER and gctf for Gctf.

defocus_step The strip size in nanometers to perform CTF correction in novaCTF refer to the paper for more information on this value and sensible defaults.

correct_astigmatism Do you want to correct astigmatism 1 for yes 0 for no.

defocus_shift_file If you want to shift the defocus for some reason away from the center of the mass of the tomogram provide a defocus_shifts file with the shifts. See the paper for more information on this value. If you do not want to use this option leave the value "".

19.1.8 IMOD 2D-CTF Options

do_2dctf Set this value to 1 if you want to do 2D-CTF correction during the reconstruction of the tomograms. As of now if you are doing 2D-CTF correction only "imod" is valid as a value for "defocus_file_format".

defocus_shift If you want to shift the defocus for some reason away from the center of the mass of the tomogram provide the number of pixels to shift here. The sign of the the shift is the same as for SHIFT in IMOD's tilt.com, but depends on the binning of the data, whereas in tilt it is for unbinned data. Refer to the man page for ctfphaseflip for a more detailed description.

defocus_tolerance Defocus tolerance in nanometers, which is one factor that governs the width of the strips. The actual strip width is based on the width of this region and several other factors. Refer to the man page for ctfphaseflip for a more detailed description.

interpolation_width The distance in pixels between the center lines of two consecutive strips. Refer to the man page for ctfphaseflip for a more detailed description.

use_gpu If you want to use a GPU set this to 1, but be careful to not use both the cluster and the GPU as this is not supported.

19.1.9 Radial Filter Options

do_radial Set this value to 1 if you want to radial filter the projections before reconstruction. This corresponds to the W (weighted) in WBP, which is commonly what you want to do, however if you want to only back-project without the weighting set this value to 0.

radial_cutoff The parameters of RADIAL from the tilt manpage in IMOD that describes the radial filter used to weight before back-projection.

radial_falloff The parameters of RADIAL from the tilt manpage in IMOD that describes the radial filter used to weight before back-projection.

19.1.10 IMOD Options

erase_radius The radius in pixels to erase when removing the gold fiducials from the aligned tilt-series stacks. Be careful that the value you give is appropriate for the unbinned aligned stack, which may be different than the value used in eTomo on the binned version.

do_rotate_tomo Set this value to 1 if you want to use trimvol or clip rotx to rotate the tomogram from the PERPENDICULAR XZ generated tomograms to the standard XY PARALLEL orientation. Set this value to 0 if you want to skip this step which greatly speeds up processing and reduces the memory footprint, but at the cost of easy visualization of the tomogram.

do_trimvol Set this value to 1 if you want to use "trimvol -rx" to flip the tomograms to the XY standard orientation from the XZ generated tomograms. Otherwise "clip rotx" will be used since it is much faster.

19.2 Example

```
scratch_dir="${PWD}"

novactf_exe="$(which novaCTF)"

newstack_exe="$(which newstack)"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

mem_free="1G"

mem_max="64G"

num_threads=1

job_name="subTOM"

run_local=0

tomo_fmt="TS_XXXIDXXXX_dose-filt"

tomo_dir_fmt="TS_XXXIDXXXX"

start_idx=1

end_idx=1

idx_fmt="%02d"

defocus_file="ctfplotter/TS_XXXIDXXXX_output.txt"

pixel_size=0.1

amplitude_contrast=0.07

cs=2.7

voltage=300

do_3dctf=1

correction_type="multiplication"

defocus_file_format="imod"

defocus_step=15

correct_astigmatism=1

defocus_shift_file=""
```

(continues on next page)

(continued from previous page)

```
do_2dctf=0

defocus_shift=0

defocus_tolerance=200

interpolation_width=20

use_gpu=0

do_radial=1

radial_cutoff=0.35

radial_falloff=0.035

erase_radius=32

do_rotate_vol=1

do_trimvol=0
```

CHAPTER

TWENTY

SUBTOM_RENUMBER_MOTL

Renumbers the particle indices in a motive list, either sequentially or in a way that preserves particle indices while still making sure there are no duplicates in the list of indices.

This MOTL manipulation script uses one MATLAB compiled scripts below:

- *subtom_renumber_motl*

20.1 Options

20.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

20.1.2 Variables

renumber_motl_exec Renumber motive list executable

20.1.3 File Options

input_motl_fn Relative path and name of the input MOTL file to be renumbered.

output_motl_fn Relative path and name of the output MOTL file.

20.1.4 Renumber Options

sort_row If you want to have the output MOTL file sorted by a particular field before renumbering then specify it here.

do_sequential If the following is 1, particles will be completely renumbered from 1 to the number of particles in the motive list. If it is 0, particles will be renumbered in a way that preserves the original index while still removing any duplicate indices. As a guide you probably want to renumber sequentially after cleaning from initial oversampled coordinates, but do not want to renumber sequentially in other cases.

20.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

renumber_motl_exec="${exec_dir}/MOTL/subtom_renumber_motl"

input_motl_fn="combinedmotl/allmotl_1.em"

output_motl_fn="combinedmotl/allmotl_unique_1.em"

sort_row="4"

do_sequential="0"
```

CHAPTER

TWENTYONE

SUBTOM_ROTX_MOTL

Transforms a given MOTL file so that it matches a tomogram rotated or not rotated by IMOD's 'clip rotx' command.

This MOTL manipulation script uses one MATLAB compiled scripts below:

- *subtom_rotx_motl*

21.1 Options

21.1.1 Directories

tomogram_dir Absolute path to the folder where the tomograms used in the INPUT motive list are stored.

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

21.1.2 Variables

rotx_motl_exec Rotx motive list executable

21.1.3 File Options

input_motl_fn Relative path and name of the input MOTL file to be transformed.

output_motl_fn Relative path and name of the output MOTL file.

21.1.4 Rotx Options

tomo_row Row number of allmotl for tomogram numbers.

do_rotx If the following is set to 1 the input MOTL will be transformed in the same way as done by 'clip rotx'. If it is set to 0 the input MOTL will be transformed by the inverse operation (a positive 90 degree rotation about the X-axis).

21.2 Example

```
tomogram_dir="/net/dstore2/teraraid/dmorado/subTOM_tutorial/data/tomos/bin4"

scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

rotx_motl_exec="${exec_dir}/MOTL/subtom_rotx_motl"

input_motl_fn="../bin4/combinedmotl/allmotl_1.em"

output_motl_fn="combinedmotl/allmotl_1.em"

tomo_row="7"

do_rotx="0"
```

CHAPTER

TWENTYTWO

SUBTOM_SCALE_MOTL

Scales a given MOTL file by a given factor. It also resets the shifts in the motive list (rows 11 to 13) to values less than 1 so that with a given scale factor of 1, it can apply the shifts to the tomogram coordinates (rows 8 to 10) so that particles can be reextracted better centered to allow for tighter CC masks to be used in further iterations of alignment.

This MOTL manipulation script uses one MATLAB compiled scripts below:

- *subtom_scale_motl*

22.1 Options

22.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

22.1.2 Variables

scale_motl_exec Scale motive list executable

22.1.3 File Options

input_motl_fn Relative path and name of the input MOTL file to be unbinned.

output_motl_fn Relative path and name of the output MOTL file.

22.1.4 Scaling Options

scale_factor How much to scale up the tomogram coordinate extraction positions (rows 8 through 10 in the MOTL) and the particle shifts (rows 11 through 13).

22.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

scale_motl_exec="${exec_dir}/MOTL/subtom_scale_motl"

input_motl_fn="../bin8/combinedmotl/allmotl_1.em"

output_motl_fn="combinedmotl/allmotl_1.em"

scale_factor=2
```

CHAPTER

TWENTYTHREE

SUBTOM_SCALE_NOISEMOTL

Scales a the individual noise motive lists corresponding to a given MOTL file by a given factor. It first concatenates all the necessary input noise motive lists, then scales the motive list by factor and then finally splits the motive list again by tomogram.

This MOTL manipulation script uses three MATLAB compiled scripts below:

- *subtom_cat_motls*
- *subtom_scale_motl*
- *subtom_split_motl_by_row*

23.1 Options

23.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

23.1.2 Variables

cat_motls_exec Concatenate motive lists executable

scale_motl_exec Scale motive list executable

split_motl_by_row_exec Split MOTL by row executable.

motl_dump_exec MOTL dump executable

23.1.3 File Options

iteration The iteration of the all particle motive list to process from: input will be all_motl_fn_prefix_iteration.em (define as integer e.g. iteration=1)

all_motl_fn_prefix Relative path and prefix to allmotl file from scratch directory.

input_noise_motl_fn_prefix Relative path and prefix to input noisemotls.

output_noise_motl_fn_prefix Relative path and prefix to output noisemotls.

23.1.4 Tomogram Options

tomo_row Row number of allmotl for tomogram numbers.

23.1.5 Scaling Options

scale_factor How much to scale up the tomogram coordinate extraction positions (rows 8 through 10 in the MOTL), e.g. To scale from bin8 to bin4 the factor would be 2.

23.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

cat_motls_exec="${exec_dir}/MOTL/subtom_cat_motls"

scale_motl_exec="${exec_dir}/MOTL/subtom_scale_motl"

split_motl_by_row_exec="${exec_dir}/MOTL/subtom_split_motl_by_row"

motl_dump_exec="${exec_dir}/MOTL/motl_dump"

iteration="1"

all_motl_fn_prefix="combinedmotl/allmotl"

input_noise_motl_fn_prefix="../bin8/combinedmotl/noisemotl"

output_noise_motl_fn_prefix="combinedmotl/noisemotl"

tomo_row="7"

scale_factor="2"
```

CHAPTER

TWENTYFOUR

SUBTOM_SEED_POSITIONS

Takes the motive lists made from clicker files in UCSF Chimera and places a number of points at a given spacing along spherical or tubular surfaces.

This MOTL manipulation script uses one MATLAB compiled scripts below:

- *subtom_seed_positions*

24.1 Options

24.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables.

24.1.2 Variables

seed_pos_exec Seed positions on motive list executable.

24.1.3 File Options

input_motl_fn_prefix Relative path and prefix of the input MOTL files to be seeded. The files are expected to have the format input_motl_fn_prefix_#.em where # is the number corresponding to the tomogram corresponding to the motive list and this value will go into row 7 of the output motive list.

output_motl_fn Relative path and name of the output MOTL file.

24.1.4 Seed Options

spacing The spacing in pixels at which positions will be added to the surface.

do_tubule If this is set to 1 (i.e. evaluates to true in Matlab) then the clicker motive list is assumed to correspond to tubules and points will be added along the tubule-axis. Otherwise the clicker file is assumed to correspond to spheres.

rand_inplane If this is set to 1 (i.e. evaluates to true in Matlab) then the inplane rotation of particles along a tubule will be randomized as opposed to the default which is to place the X-axis orthogonal to the longest tubule axis.

24.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

seed_pos_exec="${exec_dir}/MOTL/subtom_seed_positions"

input_motl_fn_prefix="../startset/clicker"

output_motl_fn="combinedmotl/allmotl_1.em"

spacing=8

do_tubule=0

rand_inplane=0
```

CHAPTER

TWENTYFIVE

SUBTOM_SHAPE

Creates a simple shape of a user-specified type that can be used for masking purposes.

This utility script uses one MATLAB compiled scripts below:

- *subtom_shape*

25.1 Options

25.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

25.1.2 Variables

shape_exec Shape executable

25.1.3 File Options

output_fn Relative path and name of the output volume to write.

ref_fn Relative path and name of a reference to apply the mask to, which can be useful for checking. If you want to skip this check just leave it set to "".

25.1.4 Shape Options

shape

The shape to place into the volume. The available options are:

- sphere,
- sphere_shell
- ellipsoid
- ellipsoid_shell
- cylinder

- tube
- elliptic_cylinder
- elliptic_tube
- cuboid

box_size Size of the volume in pixels. The volume will be a cube with this side length.

radius

For the shapes:

- sphere
- cylinder

Defines the radius of the shape. If you are not creating one of the listed shapes leave this set to "".

inner_radius

For the shapes:

- sphere_shell
- tube

Defines the inner radius of the shape. If you are not creating one of the listed shapes leave this set to "".

outer_radius

For the shapes:

- sphere_shell
- tube

Defines the outer radius of the shape. If you are not creating one of the listed shapes leave this set to "".

radius_x

For the shapes:

- ellipsoid
- elliptic_cylinder

Defines the radius of the shape about the X-axis. If you are not creating one of the listed shapes leave this set to "".

radius_y

For the shapes:

- ellipsoid
- elliptic_cylinder

Defines the radius of the shape about the Y-axis. If you are not creating one of the listed shapes leave this set to "".

radius_z

For the shapes:

- ellipsoid
- elliptic_cylinder

Defines the radius of the shape about the Z-axis. If you are not creating one of the listed shapes leave this set to "".

inner_radius_x

For the shapes:

- ellipsoid_shell
- elliptic_tube

Defines the inner radius of the shape about the X-axis. If you are not creating one of the listed shapes leave this set to "".

inner_radius_y

For the shapes:

- ellipsoid_shell
- elliptic_tube

Defines the inner radius of the shape about the Y-axis. If you are not creating one of the listed shapes leave this set to "".

inner_radius_z

For the shapes:

- ellipsoid_shell
- elliptic_tube

Defines the inner radius of the shape about the Z-axis. If you are not creating one of the listed shapes leave this set to "".

outer_radius_x

For the shapes:

- ellipsoid_shell
- elliptic_tube

Defines the outer radius of the shape about the X-axis. If you are not creating one of the listed shapes leave this set to "".

outer_radius_y

For the shapes:

- ellipsoid_shell
- elliptic_tube

Defines the outer radius of the shape about the Y-axis. If you are not creating one of the listed shapes leave this set to "".

outer_radius_z

For the shapes:

- ellipsoid_shell
- elliptic_tube

Defines the outer radius of the shape about the Z-axis. If you are not creating one of the listed shapes leave this set to "".

length_x

For the shape:

- cuboid

Defines the length of the cuboid about the X-axis. If you are not creating one of the listed shapes leave this set to "".

length_y

For the shape:

- cuboid

Defines the length of the cuboid about the Y-axis. If you are not creating one of the listed shapes leave this set to "".

length_z

For the shape:

- cuboid

Defines the length of the cuboid about the Z-axis. If you are not creating one of the listed shapes leave this set to "".

height

For the shape:

- cylinder
- tube
- elliptic_cylinder
- elliptic_tube

Defines the height of the shape. If you are not creating one of the listed shapes leave this set to "".

center_x For all shapes. Defines the X-coordinate of the center of the shape. The default center is defined as:

```
center_x = floor(box_size / 2) + 1;
```

If you do not want to modify the default value leave this set to "".

center_y For all shapes. Defines the Y-coordinate of the center of the shape. The default center is defined as:

```
center_y = floor(box_size / 2) + 1;
```

If you do not want to modify the default value leave this set to "".

center_z For all shapes. Defines the Z-coordinate of the center of the shape. The default center is defined as:

```
center_z = floor(box_size / 2) + 1;
```

If you do not want to modify the default value leave this set to "".

shift_x For all shapes. Defines a shift along the X-axis after any given rotations. This shift is part of an affine transformation about the given center that is applied to the coordinates before the shape is determined. If you do not want to modify the default value leave this set to "".

shift_y For all shapes. Defines a shift along the Y-axis after any given rotations. This shift is part of an affine transformation about the given center that is applied to the coordinates before the shape is determined. If you do not want to modify the default value leave this set to "".

shift_z For all shapes. Defines a shift along the Z-axis after any given rotations. This shift is part of an affine transformation about the given center that is applied to the coordinates before the shape is determined. If you do not want to modify the default value leave this set to "".

rotate_phi For all shapes. Defines an inplane rotation about the Z-axis. This rotation is part of an affine transformation about the given center that is applied to the coordinates before the shape is determined. If you do not want to modify the default value leave this set to "".

rotate_psi For all shapes. Defines an azimuthal rotation about the Z-axis. This rotation is part of an affine transformation about the given center that is applied to the coordinates before the shape is determined. If you do not want to modify the default value leave this set to "".

rotate_theta For all shapes. Defines a zenithal rotation about the X-axis. This rotation is part of an affine transformation about the given center that is applied to the coordinates before the shape is determined. If you do not want to modify the default value leave this set to "".

sigma For all shapes. Defines the sigma of a Gaussian falloff away from the hard edges of the shape. If you do not want to modify the default value leave this set to "".

25.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

shape_exec="${exec_dir}/utils/subtom_shape"

output_fn="otherinputs/mask.em"

ref_fn="ref/ref_1.em"

shape="sphere"

box_size="128"

radius="32"

inner_radius=""

outer_radius=""

radius_x=""

radius_y=""

radius_z=""

inner_radius_x=""

inner_radius_y=""
```

(continues on next page)

(continued from previous page)

```
inner_radius_z=""

outer_radius_x=""

outer_radius_y=""

outer_radius_z=""

length_x=""

length_y=""

length_z=""

height=""

center_x=""

center_y=""

center_z=""

shift_x=""

shift_y=""

shift_z=""

rotate_phi=""

rotate_psi=""

rotate_theta=""

sigma="3"
```

CHAPTER

TWENTYSIX

SUBTOM_SPLIT_MOTL_BY_ROW

Splits a given MOTL file by unique entries in a given field.

This MOTL manipulation script uses one MATLAB compiled scripts below:

- *subtom_split_motl_by_row*

26.1 Options

26.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

26.1.2 Variables

split_motl_by_row_exec Split MOTL by row executable.

26.1.3 File Options

input_motl_fn Relative path and name of the input MOTL file to be split.

output_motl_fn_prefix Relative path and filename prefix of output MOTL files.

26.1.4 Split Motl Options

split_row Which row to split the input MOTL file by.

26.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

split_motl_by_row_exec="${exec_dir}/MOTL/subtom_split_motl_by_row"

input_motl_fn="combinedmotl/allmotl_1.em"

output_motl_fn_prefix="combinedmotl/allmotl_1_tomo"

split_row=7
```

CHAPTER

TWENTYSEVEN

SUBTOM_TRANSFORM_MOTL

Apply a rotation and a shift to a MOTL file.

This MOTL manipulation script uses one MATLAB compiled scripts below:

- *subtom_transform_motl*

27.1 Options

27.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

27.1.2 Variables

transform_motl_exec Absolute path to transform_motl executable

27.1.3 File Options

input_motl_fn Relative path and name of the input MOTL file to be transformed.

output_motl_fn Relative path and name of the output MOTL file.

27.1.4 Transform Options

shift_x How much to shift the reference along the X-Axis, applied after the rotations described below.

shift_y How much to shift the reference along the Y-Axis, applied after the rotations described below.

shift_z How much to shift the reference along the Z-Axis, applied after the rotations described below.

rotate_phi Hom much to finally rotate the reference in-plane about it's final Z-Axis. (i.e. Spin rotation corresponding to phi).

rotate_psi How much to first rotate the reference about it's initial Z-Axis. (i.e. Azimuthal rotation corresponding to psi).

rotate_theta How much to second rotate the reference about it's intermediate X-Axis. (i.e. Zenithal rotation corresponding to theta).

rand_inplane If this is set to 1 (i.e. evaluates to true in Matlab) then the inplane rotation of particles will be randomized after the application of the given transform.

27.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

transform_motl_exec="${exec_dir}/MOTL/subtom_transform_motl"

input_motl_fn="combinedmotl/allmotl_1.em"

output_motl_fn="combinedmotl/allmotl_transformed_1.em"

shift_x=0.0

shift_y=0.0

shift_z=0.0

rotate_phi=0.0

rotate_psi=0.0

rotate_theta=0.0

rand_inplane=0
```

CHAPTER

TWENTYEIGHT

SUBTOM_UNCLASS_MOTL

Removes the iclass information in the 20th field of a motive list.

This MOTL manipulation script uses one MATLAB compiled scripts below:

- *subtom_unclass_motl*

28.1 Options

28.1.1 Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

28.1.2 Variables

unclass_motl_exec Unclass motive list executable

28.1.3 File Options

input_motl_fn Relative path and name of the input MOTL file to be unclassed.

output_motl_fn Relative path and name of the output MOTL file.

28.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

scale_motl_exec="${exec_dir}/MOTL/subtom_unclass_motl"

input_motl_fn="combinedmotl/allmotl_wmd_5.em"
```

(continues on next page)

(continued from previous page)

```
output_motl_fn="combinedmotl/allmotl_wmd_unclassed_1.em"
```

CHAPTER

TWENTYNINE

FUNCTIONS

Functions are the internals of subTOM which are implemented in Matlab and utilizes the TOM toolbox to do the actual processing of subvolumes alignment and averaging. Each function can be run itself in Matlab, which allows for users to copy and modify these functions in anyway that they see fit.

Care has been taken to make sure the functions have at least upper-level documentation, which can be accessed using the `help` command in Matlab, and is also included here for reference.

CHAPTER

THIRTY

SUBTOM_BANDPASS

Creates and/or applies a bandpass filter to a volume.

```
subtom_bandpass(
    'input_fn', input_fn (''),
    'high_pass_fp', high_pass_fp (0),
    'high_pass_sigma', high_pass_sigma (0),
    'low_pass_fp', low_pass_fp (0),
    'low_pass_sigma', low_pass_sigma (0),
    'filter_fn', filter_fn (''),
    'output_fn', output_fn (''))
```

Simply creates and/or applies a bandpass filter just as would be done during alignment, with the option to write out the Fourier Filter volume as well just for visualization purposes. `input_fn` defines the volume to be filtered, or at minimum the box size used to create the filter volume. The Fourier domain filter created is dependent on the parameters `high_pass_fp`, `high_pass_sigma`, `low_pass_fp`, `low_pass_sigma` which are all in the units of Fourier pixels. If `filter_fn` is a non-empty string then the bandpass filter volume itself is written to the filename given. If `output_fn` is a non-empty string then the bandpass filtered volume is written to the filename given.

30.1 Example

```
subtom_bandpass(...
    'input_fn', 'ref/ref_1.em', ...
    'high_pass_fp', 2, ...
    'high_pass_sigma', 2, ...
    'low_pass_fp', 15, ...
    'low_pass_sigma', 3, ...
    'filter_fn', 'otherinputs/bandpass_hp2s2_lp15s3.em',
    'output_fn', 'ref/ref_hp2s2_lp15s3_1.em')
```

30.2 See Also

- *subtom_scan_angles_exact*
- *subtom_plot_filter*

CHAPTER

THIRTYONE

SUBTOM_CAT_MOTLS

Concatenate motive lists and print on the standard output.

```
subtom_cat_motls(
    'write_motl', write_motl (0),
    'output_motl_fn', output_motl_fn (''),
    'write_star', write_star (0),
    'output_star_fn', output_star_fn (''),
    'sort_row' sort_row (0),
    'do_quiet', do_quiet (0),
    input_motl_fns)
```

Takes the motive lists given in `input_motl_fns`, and concatenates them all together. If `write_motl` evaluates to True as a boolean then the joined motive lists are written out as `ouput_motl_fn`. The function writes the motive list information in STAR format and if `write_star` evaluates to True as a boolean then the joined motive lists are also written out as `output_star_fn`. Since the input motive lists can be in any order and this does not guarantee that the output motive list will have any form of sorting, if `sort_row` is a valid field number the output motive list will be sorted by `sort_row`.

The motive list is also printed to standard ouput. An arbitrary choice has been made to ouput the motive list in STAR format, since it is used in other more well-known EM software packages. If this screen output is not desired set `do_quiet` to evaluate to true as a boolean.

31.1 Example

```
subtom_cat_motls(...
    'write_motl', 1, ...
    'output_motl_fn', 'combinedmotl/allmotl_1_joined.em', ...
    'write_star', 1, ...
    'output_star_fn', 'combinedmotl/allmotl_1_joined.star', ...
    'sort_row', 4, ...
    'do_quiet', 1, ...
    'combinedmotl/allmotl_1_tomo_1.em', ...
    'combinedmotl/allmotl_1_tomo_3.em');
```

31.2 See Also

- *subtom_clean_motl*
- *subtom_compare_motls*
- *subtom_even_odd_motl*
- *subtom_random_subset_motl*
- *subtom_renumber_motl*
- *subtom_rotx_motl*
- *subtom_scale_motl*
- *subtom_seed_positions*
- *subtom_split_motl_by_row*
- *subtom_transform_motl*
- *subtom_unclass_motl*

CHAPTER

THIRTYTWO

SUBTOM_CLEAN_MOTL

Cleans a given MOTL file based on distance and or CC scores.

```
subtom_clean_motl(
    'input_motl_fn', input_motl_fn (''),
    'output_motl_fn', output_motl_fn (''),
    'tomo_row', tomo_row (7),
    'do_ccclean', do_ccclean (0),
    'cc_fraction', cc_fraction (1),
    'cc_cutoff', cc_cutoff (-1),
    'do_distance', do_distance (0),
    'distance_cutoff', distance_cutoff (Inf),
    'do_cluster', do_cluster (0),
    'cluster_distance', cluster_distance (0),
    'cluster_size', cluster_size (1),
    'do_edge', do_edge (0),
    'tomogram_dir', tomogram_dir (''),
    'box_size', box_size (0),
    'write_stats', write_stats (0),
    'output_stats_fn', output_stats_fn (''))
```

Takes the motl given by `input_motl_fn`, and splits it internally by tomogram given by the row `tomo_row` in the MOTL, and then removes particles by one or multiple methods, if `do_ccclean` evaluates to true as a boolean then one of two methods can be applied. Either `cc_cutoff` is specified and particles that have a CCC less than `cc_cutoff` will be discarded. Alternatively `cc_fraction` can be specified as a number between 0 and 1 and that fraction of the data with the highest CCCs will be kept and the rest discarded. If `do_distance` evaluates to true as a boolean then particles that are within `distance_cutoff` pixels of each other will be determined and only the particle with the highest CCC, will be kept. If `do_cluster` evaluates to true as a boolean,then particles must have at least `cluster_size` neighbor particles within `cluster_distance` to be kept after cleaning. Finally if `do_edge` evaluates to true as a boolean then the program will look for a tomogram in `tomogram_dir`, and if a particle of box size `box_size` would extend outside of the tomogram it will be removed.

32.1 Example

```
subtom_clean_motl(...
    'input_motl_fn', 'combinedmotl/allmotl_3.em', ...
    'output_motl_fn', 'combinedmotl/allmotl_3_cc0.1_dist4_c2d10.em', ...
    'tomo_row', 7, ...
    'do_ccclean', 1, ...
    'cc_fraction', 1, ...
    'cc_cutoff', 0.1, ...
    'do_distance', 1, ...
    'distance_cutoff', 4, ...
    'do_cluster', 1, ...
    'cluster_distance', 10, ...
    'cluster_size, 2, ...
    'do_edge', 1, ...
    'tomogram_dir', '../../tomos/bin8', ...
    'box_size', 36, ...
    'write_stats', 1, ...
    'output_stats_fn', 'combinedmotl/allmotl_3_cleaned_stats.csv')
```

32.2 See Also

- *subtom_cat_motls*
- *subtom_compare_motls*
- *subtom_even_odd_motl*
- *subtom_random_subset_motl*
- *subtom_renumber_motl*
- *subtom_rotx_motl*
- *subtom_scale_motl*
- *subtom_seed_positions*
- *subtom_split_motl_by_row*
- *subtom_transform_motl*
- *subtom_unclass_motl*

CHAPTER

THIRTYTHREE

SUBTOM_COMPARE_MOTLS

Compares orientations and shifts between two MOTLs.

```
subtom_compare_motls(
    'motl_1_fn', motl_1_fn (''),
    'motl_2_fn', motl_2_fn (''),
    'write_diffs', write_diffs (0),
    'output_diffs_fn', output_diffs_fn (''))
```

Takes the motls given by `motl_1_fn` and `motl_2_fn` and calculates the differences for both the orientations and coordinates between corresponding particles in each motive list. If `write_diffs` evaluates to true as a boolean, then also a CSV file with the differences in coordinates and orientations to `diffs_output_fn`.

33.1 Example

```
subtom_compare_motls(...
    'motl_1_fn', 'combinedmotl/allmotl_1.em', ...
    'motl_2_fn', 'combinedmotl/allmotl_2.em', ...
    'write_diffs', 1, ...
    'output_diffs_fn', 'combinedmotl/allmotl_1_2_diff.csv')
```

33.2 See Also

- *subtom_cat_motls*
- *subtom_clean_motl*
- *subtom_even_odd_motl*
- *subtom_random_subset_motl*
- *subtom_renumber_motl*
- *subtom_rotx_motl*
- *subtom_scale_motl*
- *subtom_seed_positions*
- *subtom_split_motl_by_row*
- *subtom_transform_motl*

- *subtom_unclass_motl*

CHAPTER

THIRTYFOUR

SUBTOM_EVEN_ODD_MOTL

Split a MOTL file into even odd halves.

```
subtom_even_odd_motl(
    'input_motl_fn', input_motl_fn (''),
    'output_motl_fn', output_motl_fn (''),
    'even_motl_fn', even_motl_fn, (''),
    'odd_motl_fn', odd_motl_fn (''),
    'split_row', split_row (4))
```

Takes the MOTL file specified by `input_motl_fn` and writes out seperate MOTL files with `even_motl_fn` and `odd_motl_fn` where each output file corresponds to roughly half of `input_motl_fn`. The motive list can also write a single motive list file with the half split described using the iclass (20th row of the motive list) where the odd half takes particle's current class number plus 100 and the even half takes the particle's current class number plus 200. The MOTL is split by the values in `split_row`, initially just taking even/odd halves of the unique values in that given row, and then this is slightly adjusted by naively adding to the lesser half until closest to half is found.

34.1 Example

```
subtom_even_odd_motl(...
    'input_motl_fn', 'combinedmotl/allmotl_1.em', ...
    'output_motl_fn', 'combinedmotl/allmotl_eo_1.em', ...
    'even_motl_fn', 'even/combinedmotl/allmotl_1.em', ...
    'odd_motl_fn', 'odd/combinedmotl/allmotl_1.em', ...
    'split_row', 4)
```

34.2 See Also

- *subtom_cat_motls*
- *subtom_clean_motl*
- *subtom_compare_motls*
- *subtom_random_subset_motl*
- *subtom_renumber_motl*
- *subtom_rotx_motl*
- *subtom_scale_motl*

- *subtom_seed_positions*
- *subtom_split_motl_by_row*
- *subtom_transform_motl*
- *subtom_unclass_motl*

CHAPTER

THIRTYFIVE

SUBTOM_EXTRACT_NOISE

Extract noise amplitude spectra on the cluster.

```
subtom_extract_noise(
    'tomogram_dir', tomogram_dir (''),
    'tomo_row', tomo_row (7),
    'ampspec_fn_prefix', ampspec_fn_prefix ('otherinputs/ampspec'),
    'binary_fn_prefix', binary_fn_prefix ('otherinputs/binary'),
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'noise_motl_fn_prefix', noise_motl_fn_prefix ('combinedmotl/noisemotl'),
    'iteration', iteration (1),
    'box_size', box_size (-1),
    'just_extract', just_extract (0),
    'ptcl_overlap_factor', ptcl_overlap_factor (0),
    'noise_overlap_factor', noise_overlap_factor (0),
    'num_noise', num_noise (1000),
    'process_idx', process_idx (1),
    'reextract', reextract (0),
    'preload_tomogram', preload_tomogram (1),
    'use_tom_red', use_tom_red (0),
    'use_memmap', use_memmap (0))
```

Takes the tomograms given in `tomogram_dir` and extracts average amplitude spectra and binary wedges into files with the name formats `ampspec_fn_prefix` _#.em and `binary_fn_prefix` _ #.em, respectively where # corresponds to the tomogram from which it was created.

Tomograms are specified by the field `tomo_row` in motive list `all_motl_fn_prefix` _#.em where # corresponds to `iteration`. and the tomogram that will be processed is selected by `process_idx`. Motive lists with the positions used to generate the amplitude spectrum are written with the name format `noise_motl_fn_prefix` _#.em.

`num_noise` Noise volumes of size `box_size` are first identified that do not clash with the subtomogram positions in the input motive list or other already selected noise volumes. `ptcl_overlap_factor` and `noise_overlap_factor` define how much overlap selected noise volumes can have with subtomograms and other noise volumes respectively with 1 being complete overlap and 0 being complete separation.

If `noise_motl_fn_prefix` _#.em already exists then if `just_extract` evaluates to true as a boolean, then noise volume selection is skipped and the positions in the motive list are extracted and the amplitude spectrum is generated, whether or not the length of the motive list is less than `num_noise`. Otherwise positions will be added to the motive list up to `num_noise`.

If `reextract` evaluates to true as a boolean, than existing amplitude spectra will be overwritten. Otherwise the program will do nothing and exit if the volume already exists. This is for in the case that the processing crashed at some point in execution and then can just be re-run without any alterations.

If `preload_tomogram` evaluates to true as a boolean, then the whole tomogram will be read into memory before extraction begins, otherwise the particles will be read from disk or from a memory-mapped tomogram. If `use_tom_red` evaluates to true as a boolean the old particle extraction code will be used, but this is only for legacy support and is not suggested for use. Finally if `use_memmap` evaluates to true as a boolean then in place of reading each particle from disk a memory-mapped version of the file of will be created to attempt faster access in extraction.

35.1 Example

```
subtom_extract_noise(...
    'tomogram_dir', '../data/tomos/bin8', ...
    'tomo_row', 7, ...
    'ampspec_fn_prefix', 'otherinputs/ampspec', ...
    'binary_fn_prefix', 'otherinputs/binary', ...
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'noise_motl_prefix', 'combinedmotl/noisemotl', ...
    'iteration', 1, ...
    'box_size', 36, ...
    'just_extract', 0, ...
    'ptcl_overlap_factor', 0.0, ...
    'noise_overlap_factor, 0.75, ...
    'num_noise', 1000,
    'process_idx', 1, ...
    'reextract', 1, ...
    'preload_tomogram', 1, ...
    'use_tom_red', 0, ...
    'use_memmap', 0)
```

35.2 See also

- *subtom_extract_subtomograms*
- *subtom_parallel_sums*
- *subtom_scan_angles_exact*
- *subtom_weighted_average*

CHAPTER

THIRTYSIX

SUBTOM_EXTRACT_SUBTOMOGRAMS

Extract subtomograms on the cluster.

```
subtom_extract_subtomograms(
    'tomogram_dir', tomogram_dir (''),
    'tomo_row', tomo_row (7),
    'subtomo_fn_prefix', subtom_fn_prefix ('subtomograms/subtomo'),
    'subtomo_digits', subtomo_digits (1),
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'stats_fn_prefix', stats_fn_prefix ('subtomograms/stats/tomo'),
    'iteration', iteration (1),
    'box_size', box_size (-1),
    'process_idx', process_idx (1),
    'reextract', reextract (0),
    'preload_tomogram', preload_tomogram (1),
    'use_tom_red', use_tom_red (0),
    'use_memmap', use_memmap (0))
```

Takes the tomograms given in `tomogram_dir` and extracts subtomograms specified in `all_motl_fn_prefix` _#.m where # corresponds to `iteration` with size `box_size` into `scratch_dir` with the name formats `subtomo_fn_prefix` _#.em where # corresponds to the entry in field 4 in `all_motl_fn_prefix` _#.em zero-padded to have at least `subtomo_digits` digits.

Tomograms are specified by the field `tomo_row` in motive list `all_motl_fn_prefix` _#.em, and the tomogram that will be processed is selected by `process_idx`. A CSV-format file with the subtomogram ID-number, average, min, max, standard deviation and variance for each subtomogram in the tomogram is also written with the name format `stats_fn_prefix` _#.em where # corresponds to the tomogram from which subtomograms were extracted.

If `reextract` evaluates to true as a boolean, than existing subtomograms will be overwritten. Otherwise the program will do nothing and exit if `stats_fn_prefix` _#.em already exists, or will also skip any subtomogram it is trying to extract that already exists. This is for in the case that the processing crashed at some point in execution and then can just be re-run without any alterations.

If `preload_tomogram` evaluates to true as a boolean, then the whole tomogram will be read into memory before extraction begins, otherwise the particles will be read from disk or from a memory-mapped tomogram. If `use_tom_red` evaluates to true as a boolean the old particle extraction code will be used, but this is only for legacy support and is not suggested for use. Finally if `use_memmap` evaluates to true as a boolean then in place of reading each particle from disk a memory-mapped version of the file of will be created to attempt faster access in extraction.

36.1 Example

```
subtom_extract_subtomograms(...
    'tomogram_dir', '../data/tomos/bin8', ...
    'tomo_row', 7, ...
    'subtomo_fn_prefix', 'subtomograms/subtomo', ...
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'stats_fn_prefix', 'subtomograms/stats/tomo', ...
    'iteration', 1, ...
    'box_size', 36, ...
    'process_idx', 1, ...
    'reextract', 0, ...
    'preload_tomogram', 1, ...
    'use_tom_red', 0, ...
    'use_memmap', 0)
```

36.2 See also

- *subtom_extract_noise*
- *subtom_parallel_sums*
- *subtom_scan_angles_exact*
- *subtom_weighted_average*

CHAPTER

THIRTYSEVEN

SUBTOM_MASKCORRECTED_FSC

Calculates "mask-corrected" FSC and sharpened refs.

```
subtom_maskcorrected_fsc(
    'ref_a_fn_prefix', ref_a_fn_prefix ('even/ref/ref'),
    'ref_b_fn_prefix', ref_b_fn_prefix ('odd/ref/ref'),
    'fsc_mask_fn', fsc_mask_fn ('FSC/fsc_mask.em'),
    'output_fn_prefix', output_fn_prefix ('FSC/ref'),
    'filter_a_fn', filter_a_fn (''),
    'filter_b_fn', filter_b_fn (''),
    'do_reweight', do_reweight (0),
    'do_sharpen', do_sharpen (0),
    'plot_fsc', plot_fsc (0),
    'plot_sharpen', plot_sharpen (0),
    'filter_mode', filter_mode (1),
    'pixelsize', pixelsize (1.0),
    'nfold', nfold (1),
    'filter_threshold', filter_threshold (0.143),
    'rand_threshold', rand_threshold (0.8),
    'b_factor', b_factor (0),
    'box_gaussian', box_gaussian (1),
    'iteration', iteration (1))
```

Takes in two references `ref_a_fn_prefix` _#.em and `ref_b_fn_prefix` _#.em where # corresponds to `iteration` and a FSC mask `fsc_mask_fn` and calculates a "mask-corrected" FSC. This works by randomizing the structure factor phases beyond the point where the unmasked FSC curve falls below a given threshold (by default 0.8) and calculating an additional FSC between these phase randomized maps. This allows for the determination of the extra correlation caused by effects of the mask, which is then subtracted from the normal masked FSC curves. The curve will be saved as a Matlab figure and a PDF file, and if `plot_fsc` is true it will also be displayed.

The script can also output maps with the prefix `output_fn_prefix` that have been sharpened with `b_factor` if `do_sharpen` is turned on. This setting has two threshold settings selected using `filter_mode`, FSC (1) and pixel (2). FSC allows you to use a FSC-value `filter_threshold` as a cutoff for the lowpass filter, while using pixels allows you to use an arbitrary resolution cutoff in `filter_threshold`. The sharpening curve will be saved as a Matlab figure and a pdf file, and if `plot_sharpen` is true it will also be displayed.

Finally this script can also perform and output reweighted maps if `do_reweight` is true, and the pre-calculated Fourier weight volumes `filter_a_fn` and `filter_b_fn`.

37.1 Example

```
subtom_maskcorrected_fsc(...
    'ref_a_fn_prefix', 'even/ref/ref', ...
    'ref_b_fn_prefix', 'odd/ref/ref', ...
    'fsc_mask_fn', 'FSC/fsc_mask.em', ...
    'output_fn_prefix', 'FSC/ref', ...
    'filter_a_fn', '', ...
    'filter_b_fn', '', ...
    'do_reweight', 0, ...
    'do_sharpen', 1, ...
    'plot_fsc', 1, ...
    'plot_sharpen', 1, ...
    'filter_mode', 1, ...
    'pixelsize', 1.35, ...
    'nfold', 6, ...
    'filter_threshold', 0.143, ...
    'rand_threshold', 0.8, ...
    'b_factor', -1500, ...
    'box_gaussian', 3, ...
    'iteration', 1)
```

CHAPTER

THIRTYEIGHT

SUBTOM_PARALLEL_SUMS

Creates raw sums and Fourier weight sums in a batch.

```
subtom_parallel_sums(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ref_fn_prefix', ref_fn_prefix ('ref/ref'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'weight_fn_prefix', weight_fn_prefix ('otherinputs/ampspec'),
    'weight_sum_fn_prefix, weight_sum_fn_prefix ('otherinputs/wei'),
    'iteration', iteration (1),
    'tomo_row', tomo_row (7),
    'iclass', iclass (0),
    'num_avg_batch', num_avg_batch (1),
    'process_idx', process_idx (1))
```

Aligns a subset of particles using the rotations and shifts in `all_motl_fn_prefix` _#.em where # corresponds to `iteration` in `num_avg_batch` chunks to make a raw particle sum `ref_fn_prefix` _#_###.em where # corresponds to `iteration` and ### corresponds to `process_idx`. Fourier weight volumes with name prefix `weight_fn_prefix` will also be aligned and summed to make a weight sum `weight_sum_fn_prefix` _#_###.em. `tomo_row` describes which row of the motl file is used to determine the correct tomogram fourier weight file. `iclass` describes which class outside of one is included in the averaging.

38.1 Example

```
subtom_parallel_sums(...
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ref_fn_prefix', 'ref/ref', ...
    'ptcl_fn_prefix', 'subtomograms/subtomo', ...
    'weight_fn_prefix', 'otherinputs/ampspec', ...
    'weight_sum_fn_prefix, 'otherinputs/wei', ...
    'iteration', 1, ...
    'tomo_row', 7, ...
    'iclass', 0, ...
    'num_avg_batch', 1, ...
    'process_idx', 1)
```

38.2 See Also

- *subtom_extract_noise*
- *subtom_extract_subtomograms*
- *subtom_scan_angles_exact*
- *subtom_weighted_average*

CHAPTER

THIRTYNINE

SUBTOM_PLOT_FILTER

Creates a graphic of bandpass filters optionally with CTF.

```
subtom_plot_filter(
    'box_size', box_size (''),
    'pixelsize', pixelsize (1),
    'high_pass_fp', high_pass_fp (0),
    'high_pass_sigma', high_pass_sigma (0),
    'low_pass_fp', low_pass_fp (0),
    'low_pass_sigma', low_pass_sigma (0),
    'defocus', defocus (0),
    'voltage', voltage (300),
    'cs', cs (0),
    'ac', ac ('1.0'),
    'phase_shift', phase_shift (0.0),
    'b_factor', b_factor (0.0),
    'output_fn_prefix', output_fn_prefix (''))
```

Takes in the local alignment filter parameters used in subTOM, `high_pass_fp`, `high_pass_sigma`, `low_pass_fp`, and `low_pass_sigma`; then produces a figure showing the filter that will be applied to the Fourier transform of the reference during alignment. The Fourier pixel frequencies are converted into Angstroms using the given `box_size` and `pixelsize`. A single CTF can also be specified with `defocus`, `voltage`, `cs`, `ac`, `phase_shift`, and the root square of this curve will be plotted in addition to how the band-pass filter affects the amplitude effects of the CTF. Finally a B-factor falloff can also be specified with `b_factor`, and this decay curve will also be plotted and also plotted with the CTF root square, and also the CTF root square and band-pass filter all together, so a cumulative effect of a specific choice of filter parameters at a given defocus and falloff can be observed. If `output_fn_prefix` is not emtpy it is used to save the graphic in MATLAB figure, pdf, and png formatted files.

39.1 Example

```
subtom_plot_filter(...
    'box_size', 192, ...
    'pixelsize', 1.35, ...
    'high_pass_fp', 1, ...
    'high_pass_sigma', 2, ...
    'low_pass_fp', 48, ...
    'low_pass_sigma', 3, ...
    'defocus', 15000, ...
    'voltage', 300, ...
    'cs', 2.7, ...
```

(continues on next page)

(continued from previous page)

```
    'ac', 0.07, ...
    'phase_shift', 0.0, ...
    'b_factor', 0, ...
    'output_fn_prefix', 'alignment_1');
```

39.2 See Also

- *subtom_plot_scanned_angles*
- *subtom_bandpass*
- *subtom_shape*

CHAPTER

FORTY

SUBTOM_PLOT_SCANNED_ANGLES

Creates a graphic of local search rotations.

```
subtom_plot_scanned_angles(
    'psi_angle_step', psi_angle_step (0),
    'psi_angle_shells', psi_angl_shells (0),
    'phi_angle_step', phi_angle_step (0),
    'phi_angle_shells', phi_angle_shells (0),
    'initial_phi', initial_phi (0),
    'initial_psi', initial_psi (0),
    'initial_theta', initial_theta (0),
    'angle_fmt', angle_fmt ('degrees'),
    'marker_size', marker_size (0.1),
    'output_fn_prefix', output_fn_prefix (''))
```

Takes in the local search parameters used in subTOM `psi_angle_step`, `psi_angle_shells`, `phi_angle_step`, and `phi_angle_shells`; then produces a figure showing the angles that will be searched using an arrow marker. The angles are given in either radians or degrees depending on `angle_fmt`. The marker represents the X-axis after rotation and placed on the unit sphere. The initial marker position is at the north pole of the unit sphere. The size of the marker is determined by `marker_size`. The rotations can also be displayed centered on an initial rotation given by `initial_phi`, `initial_psi`, and `initial_theta`. If it is non-empty the figure will be written out in MATLAB figure, PDF and PNG format using the filename prefix `output_fn_prefix`.

40.1 Example

```
subtom_plot_scanned_angles(...
    'psi_angle_step', 6, ...
    'psi_angle_shells', 7, ...
    'phi_angle_step', 6, ...
    'phi_angle_shells', 7, ...
    'initial_phi', 0, ...
    'initial_psi', 0, ...
    'initial_theta', 0, ...
    'angle_fmt', 'degrees', ...
    'marker_size', 0.02, ...
    'output_fn_prefix', 'alignment_1');
```

40.2 See Also

- *subtom_plot_filter*
- *subtom_shape*

CHAPTER

FORTYONE

SUBTOM_RANDOM_SUBSET_MOTL

Generates a random subset of a motive list.

```
subtom_random_subset_motl(
    'input_motl_fn', input_motl_fn (''),
    'output_motl_fn', output_motl_fn (''),
    'subset_size', subset_size (1000),
    'subset_row', subset_row (7))
```

Takes the motive list given by `input_motl_fn`, and generates a random subset of `subset_size` particles where the subset is distributed equally over the motive list field `subset_row`, and then writes the subset motive list out as `output_motl_fn`.

41.1 Example

```
subtom_random_subset_motl(...
    'input_motl_fn', 'combinedmotl/allmotl_2.em', ...
    'output_motl_fn', 'combinedmotl/s5kmotl_2.em', ...
    'subset_size', 5000, ...
    'subset_row', 7)
```

41.2 See Also

- *subtom_cat_motls*
- *subtom_clean_motl*
- *subtom_compare_motls*
- *subtom_even_odd_motl*
- *subtom_renumber_motl*
- *subtom_rotx_motl*
- *subtom_scale_motl*
- *subtom_seed_positions*
- *subtom_split_motl_by_row*
- *subtom_transform_motl*

- *subtom_unclass_motl*

CHAPTER

FORTYTWO

SUBTOM_RENUMBER_MOTL

Renumbers particle indices in a motive list.

```
subtom_renumber_motl(
    'input_motl_fn', input_motl_fn (''),
    'output_motl_fn', output_motl_fn (''),
    'sort_row', sort_row (0),
    'do_sequential', do_sequential (0))
```

Takes the motive list given by `input_motl_fn`, and renumbers the particles in field 4 of the MOTL and writes out the renumbered list to `output_motl_fn`. If `do_sequential` evaluates to true as a boolean then the motive list will just be renumbered from 1 to the number of particles in the MOTL, and the initial particle indices will be lost. If `do_sequential` evaluates to false as a boolean, then particle indices will be kept with any duplicates of the particle index incremented by the largest particle index found in the motive list.

For example if `do_sequential` is 0, and we have 100 particles where the first particle index is 4, and the largest particle index in the motive list is 325. If there are 3 copies of particle index 16 in the motive list, then it will be renumbered so that these 3 copies correspond to particle indices 16, 341, and 666. In this way as long as we keep the original motive list we can trace back the origin of each particle.

42.1 Example

```
subtom_renumber_motl(...
    'input_motl_fn', 'combinedmotl/allmotl_1.em', ...
    'output_motl_fn', 'combinedmotl/allmotl_unique_1.em', ...
    'sort_row', 4, ...
    'do_sequential', 0)
```

42.2 See Also

- *subtom_cat_motls*
- *subtom_clean_motl*
- *subtom_compare_motls*
- *subtom_even_odd_motl*
- *subtom_random_subset_motl*
- *subtom_rotx_motl*

- *subtom_scale_motl*
- *subtom_seed_positions*
- *subtom_split_motl_by_row*
- *subtom_transform_motl*
- *subtom_unclass_motl*

CHAPTER

FORTYTHREE

SUBTOM_ROTX_MOTL

Transforms a motive list to (un)apply a tomogram rotx operation.

```
subtom_rotx_motl(
    'tomogram_dir', tomogram_dir (''),
    'tomo_row', tomo_row (7),
    'input_motl_fn', input_motl_fn (''),
    'output_motl_fn', output_motl_fn (''),
    'do_rotx', do_rotx (0))
```

Takes the motive list given by `input_motl_fn`, and if `do_rotx` evaluates to true as a boolean applies the same transformation as applied by 'clip rotx' in the IMOD package, and else applies the inverse transformation. The resulting motive list is then written out as `output_motl_fn`. The location of the tomograms needs to be given in `tomogram_dir`, as well as the field that specifies which tomogram to use for each particle in `tomo_row`. The size of the tomogram needs to be known to correctly transform the particle center coordinates in fields 8-10 in the motive list.

43.1 Example

```
subtom_rotx_motl(...
    'tomogram_dir', '/net/teraraid/dmorado/sample/date/tomos/bin4', ...
    'tomo_row', 7, ...
    'input_motl_fn', '../bin4/combinedmotl/allmotl_2.em', ...
    'output_motl_fn', 'combinedmotl/allmotl_bin4_rotx_1.em', ...
    'do_rotx', 0)
```

43.2 See Also

- *subtom_cat_motls*
- *subtom_clean_motl*
- *subtom_compare_motls*
- *subtom_even_odd_motl*
- *subtom_random_subset_motl*
- *subtom_renumber_motl*
- *subtom_scale_motl*
- *subtom_seed_positions*

- *subtom_split_motl_by_row*
- *subtom_transform_motl*
- *subtom_unclass_motl*

CHAPTER

FORTYFOUR

SUBTOM_SCALE_MOTL

Scales a given motive list by a given factor.

```
subtom_scale_motl(
    'input_motl_fn', input_motl_fn (''),
    'output_motl_fn', output_motl_fn (''),
    'scale_factor', scale_factor (1))
```

Takes the motive list given by `input_motl_fn`, and scales it by `scale_factor`, and then writes the transformed motive list out as `output_motl_fn`.

44.1 Example

```
subtom_scale_motl(...
    'input_motl_fn', '../bin8/combinedmotl/allmotl_2.em', ...
    'output_motl_fn', 'combinedmotl/allmotl_1.em', ...
    'scale_factor', 2)
```

44.2 See Also

- *subtom_cat_motls*
- *subtom_clean_motl*
- *subtom_compare_motls*
- *subtom_even_odd_motl*
- *subtom_random_subset_motl*
- *subtom_renumber_motl*
- *subtom_rotx_motl*
- *subtom_seed_positions*
- *subtom_split_motl_by_row*
- *subtom_transform_motl*
- *subtom_unclass_motl*

CHAPTER

FORTYFIVE

SUBTOM_SCAN_ANGLES_EXACT

Align a particle batch over local search angles.

```
subtom_scan_angles_exact(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ref_fn_prefix', ref_fn_prefix ('ref/ref'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'weight_fn_prefix', weight_fn_prefix ('otherinputs/ampspec'),
    'align_mask_fn', align_mask_fn ('none'),
    'cc_mask_fn', cc_mask_fn ('noshift'),
    'apply_weight', apply_weight (0),
    'apply_mask', apply_mask (0),
    'psi_angle_step', psi_angle_step (0),
    'psi_angle_shells', psi_angle_shells (0),
    'phi_angle_step', phi_angle_step (0),
    'phi_angle_shells', phi_angle_shells (0),
    'high_pass_fp', high_pass_fp (0),
    'high_pass_sigma', high_pass_sigma (0),
    'low_pass_fp', low_pass_fp (0),
    'low_pass_sigma', low_pass_sigma (0),
    'nfold', nfold (1),
    'threshold', threshold (-1),
    'iteration', iteration (1),
    'tomo_row', tomo_row (7),
    'iclass', iclass (0),
    'num_ali_batch', num_ali_batch (1),
    'process_idx', process_idx (1))
```

Aligns a batch of particles from the collective motive list with the name format `all_motl_fn_prefix` _#.em where # is the number `iteration`. The motive list is split into `num_ali_batch` chunks and the specific chunk to process is specified by `process_idx` . A motive list for the best determined alignment parameters is written out for each batch with the name format `ptct_motl_fn_prefix` _#_#.em where the first # is `iteration` + 1 and the second # is the number `process_idx`.

Particles, with the name format `ptcl_fn_prefx` _#.em where # is the subtomogram ID, are aligned against the reference with the name format `ref_fn_prefix` _#.em where # is `iteration`. Before the comparison is made a number of alterations are made to both the particle and reference:

- If `nfold` is greater than 1 then C#-symmetry is applied along the Z-axis to the reference where # is `nfold`.
- The reference is masked in real space with the mask `align_mask_fn`, and if `apply_mask` evaluates to true as a boolean, then this mask is also applied to the particle. A sphere mask is applied to the particle to reduces the artifacts caused by the box-edges on the comparison. This sphere has a diameter that is 80% the box size and falls of with a sigma that is 15% half the box size.

  - The mask is rotated and shifted with the currently existing alignment parameters for the particle as to best center the mask on the particle density.
  - `apply_mask` can help alignment and suppress alignment to other features when the particle is well-centered or already reasonably well aligned, but if this is not the case there is the risk that a tight alignment will cutoff parts of the particle.

- Both the particle and the reference are bandpass filtered in the Fourier domain defined by `high_pass_fp`, `high_pass_sigma`, `low_pass_fp`, and `low_pass_sigma` which are all in the units of Fourier pixels.
- A Fourier weight volume with the name format `weight_fn_prefix` _#.em where # corresponds to the tomogram from which the particle came from, which is found from the field `tomo_row` in the motive list, is applied to the reference in the Fourier domain, after the reference has been rotated with the currently existing alignment parameters. If `apply_weight` evaluates to true as a boolean, then this weight is also applied to the particle with no rotation. This Fourier weight is designed to compensate for the missing wedge.
  - If a binary wedge is used, then it is reasonable to apply the weight to the particle, however, for more complicated weights, like the average amplitude spectrum, it should not be done.

The local rotations searched during alignment are deteremined by the four parameters `psi_angle_step`, `psi_angle_shells`, `phi_angle_step`, and `phi_angle_shells`. They describe a search where the currently existing alignment parameters for azimuth and zenith are used to define a "pole" to search about in the ceiling of half `psi_angle_shells` cones. The change in zenith between each cone is `psi_angle_step` and the azimuth around the cone is close to the same angle but is adjusted slightly to account for bias near the pole. The final spin angle of the search is done with a change in spin of `phi_angle_step` in `phi_angle_shells` steps. The spin is applied in both clockwise and counter-clockwise fashion.

- The angles phi, and psi here are flipped in their sense of every other package for EM image processing, which is absolutely infuriating and confusing, but maintained for historical reasons, however most descriptions use the words azimuth, zenith, and spin to avoid ambiguity.

Finally after the constrained cross-correlation function is calculated it is masked with `cc_mask_fn` to limit the shifts to inside this volume, and a peak is found and it's location is determined to sub-pixel accuracy using interpolation. The rotations and shifts that gives the highest cross-correlation coefficient are then chosen as the new alignments parameters. Particles with a coefficient lower than `threshold` are placed into class 2 and ignored in later processing, and particles with class `iclass` are the only particles processed.

- If `iclass` is 0 all particles will be considered, and particles above `threshold` will be assigned to iclass of 1 and particles below `threshold` will be assigned to iclass of 2. If `iclass` is 1 or 2 then particles with iclass 0 will be skipped, particles of iclass 1 and 2 will be aligned and particles with scores above `threshold` will be assigned to iclass 1 and particles with scores below `threshold` will be assigned to iclass 2. `iclass` of 2 does not make much sense but is set this way in case of user mistakes or misunderstandings. If `iclass` is greater than 2 then particles with iclass of 1, 2, and `iclass` will be aligned, and particles with a score above `threshold` will maintain their iclass if it is 1 or `iclass`, and particles with a previous iclass of 2 will be upgraded to an iclass of 1. Particles with a score below `threshold` will be assigned to iclass 2.
- The class number is stored in the 20th field of the motive list.

45.1 Example

```
subtom_scan_angles(...
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ref_fn_prefix', 'ref/ref', ...
    'ptcl_fn_prefix', 'subtomograms/subtomo', ...
    'weight_fn_prefix', 'otherinputs/ampspec', ...
    'align_mask_fn', 'otherinputs/align_mask.em', ...
    'cc_mask_fn', 'otherinputs/cc_mask.em', ...
    'apply_weight', 0, ...
    'apply_mask', 1, ...
    'psi_angle_step', 6, ...
    'psi_angle_shells', 8, ...
    'phi_angle_step', 6, ...
    'phi_angle_shells', 8, ...
    'high_pass_fp', 1, ...
    'high_pass_sigma', 2, ...
    'low_pass_fp', 12, ...
    'low_pass_sigma', 3, ...
    'nfold', 6, ...
    'threshold', 0, ...
    'iteration', 1, ...
    'tomo_row', 7, ...
    'iclass', 0, ...
    'num_ali_batch', 1, ...
    'process_idx', 1)
```

45.2 See Also

- *subtom_extract_noise*
- *subtom_extract_subtomograms*
- *subtom_parallel_sums*
- *subtom_weighted_average*

CHAPTER

FORTYSIX

SUBTOM_SEED_POSITIONS

Place particle positions from clicker motive list.

```
subtom_seed_positions(
    'input_motl_fn_prefix', input_motl_fn_prefix ('../startset/clicker'),
    'output_motl_fn', output_motl_fn ('combinedmotl/allmotl_1.em'),
    'spacing', spacing (8),
    'do_tubule', do_tubule (0),
    'rand_inplane', rand_inplane (0))
```

Takes in clicker motive lists from the 'Pick Particle' plugin for Chimera with a name in the format `input_motl_fn_prefix` _#.em, where # should correspond to the tomogram number the clicker corresponds to. This number will be used to fill in the 7th field in the output motive list `output_motl_fn`.

Points are added with roughly a pixel distance `spacing` apart. These points are also set with Euler angles that place them normal to the surface of the sphere or tube on which they lie. Points take the form of a tube is `do_tubule` evaluates to true as a boolean otherwise the clickers are assumed to correspond to spheres. In the case of both the radius is encoded in the 3rd field of the clicker motive and carried over to the output motive list. The second field corresponds to the marker set the clicker file was created from, which is not used in placing spheres but is considered in seeding tubules to delineate between multiple tubules in each tomogram. Finally a running index of tube or sphere is added to the 6th field of the output motive list. If both `do_tubule` and `rand_inplane` evaluate to true as a boolean, then the final Euler angle (phi in AV3 notation, and psi/spin/inplane in other notations) will be randomized as opposed to directed along the tubular axis.

46.1 Example

```
subtom_seed_positions(...
    'input_motl_fn_prefix', '../startset/clicker', ...
    'output_motl_fn', 'combinedmotl/allmotl_1.em', ...
    'spaciing', 4, ...
    'do_tubule', 0, ...
    'rand_inplane', 0)
```

46.2 See Also

- *subtom_cat_motls*
- *subtom_clean_motl*
- *subtom_compare_motls*
- *subtom_even_odd_motl*
- *subtom_random_subset_motl*
- *subtom_renumber_motl*
- *subtom_rotx_motl*
- *subtom_scale_motl*
- *subtom_split_motl_by_row*
- *subtom_transform_motl*
- *subtom_unclass_motl*

CHAPTER

FORTYSEVEN

SUBTOM_SHAPE

Produces a volume of a simple shape for masking.

```
subtom_shape(
    'shape', shape (''),
    'box_size', box_size (''),
    'radius', radius (box_size / 2),
    'outer_radius', outer_radius (box_size / 2),
    'inner_radius', inner_radius (outer_radius - 2),
    'radius_x', radius_x (box_size / 2),
    'radius_y', radius_y (box_size / 2),
    'radius_z', radius_z (box_size / 2),
    'outer_radius_x', outer_radius_x (box_size / 2),
    'outer_radius_y', outer_radius_y (box_size / 2),
    'outer_radius_z', outer_radius_z (box_size / 2),
    'inner_radius_x', inner_radius_x (outer_radius_x - 2),
    'inner_radius_y', inner_radius_y (outer_radius_y - 2),
    'inner_radius_z', inner_radius_z (outer_radius_z - 2),
    'length_x', length_x (box_size),
    'length_y', length_y (box_size),
    'length_z', length_z (box_size),
    'height', height (box_size),
    'center_x', center_x (floor(box_size / 2) + 1),
    'center_y', center_y (floor(box_size / 2) + 1),
    'center_z', center_z (floor(box_size / 2) + 1),
    'shift_x', shift_x (0),
    'shift_y', shift_y (0),
    'shift_z', shift_z (0),
    'rotate_phi', rotate_phi (0),
    'rotate_psi', rotate_psi (0),
    'rotate_theta', rotate_theta (0),
    'sigma', sigma (0),
    'ref_fn', ref_fn (''),
    'output_fn', output_fn (''))
```

Creates a volume of a simple shape, with the volume being a cube of `box_size` length, and writes out the volume as `output_fn`. This volume is generally used for masking. The shape in the volume is defined by `shape` and can be one of several strings, the available shapes are 'sphere', 'sphere_shell', 'ellipsoid', 'ellipsoid_shell', 'cylinder', 'tube', 'elliptic_cylinder', 'elliptic_tube', and 'cuboid'. For each shape there are a number of options available to define the shape.

For each shape an optional gaussian smooth edge can be added by defining `sigma`.

For each shape an optional transform can also be applied to the shape by specifying a shift through the options `shift_x`, `shift_y`, and `shift_z`, and the shapes initial center can be specified by `center_x`, `center_y`, `center_z`. Rotations to the shape are applied through the options `rotate_phi`, `rotate_psi`, and `rotate_theta`. Rotations are done about the center and shifts are applied after any given rotation.

Finally another volume can be given by passing the option `ref_fn` and the shape will be applied to the volume, which can aid in testing how the shape masks the underlying density.

If shape is 'sphere', the shape is defined by `radius`.

If shape is 'sphere_shell', the shape is defined by `inner_radius` and `outer_radius`.

If shape is 'ellipsoid', the shape is defined by `radius_x`, `radius_y`, and `radius_z`.

If shape is 'ellipsoid_shell', the shape is defined by `inner_radius_x`, `inner_radius_y`, `inner_radius_z`, `outer_radius_x`, `outer_radius_y`, and `outer_radius_z`.

If shape is 'cylinder', the shape is defined by `radius` and `height`.

If shape is 'tube', the shape is defined by `inner_radius`, `outer_radius`, and `height`.

If shape is 'elliptic_cylinder', the shape is defined by `radius_x`, `radius_y`, and `height`.

If shape is 'elliptic_tube', the shape is defined by `inner_radius_x`, `inner_radius_y`, `outer_radius_x`, `outer_radius_y`, and `height`.

Finally if shape is 'cuboid', the shape is defined by `length_x`, `length_y`, and `length_z`.

47.1 Example

```
subtom_shape(...
    'shape', 'sphere', ...
    'box_size', 128, ...
    'radius', 32, ...
    'sigma', 3, ...
    'output_fn', 'otherinputs/mask.em');
```

47.2 See Also

- *subtom_plot_filter*
- *subtom_plot_scanned_angles*

CHAPTER

FORTYEIGHT

SUBTOM_SPLIT_MOTL_BY_ROW

Split a MOTL file by a given row.

```
subtom_split_motl_by_row(
    'input_motl_fn', input_motl_fn (''),
    'output_motl_fn_prefix', output_motl_fn_prefix (''),
    'split_row', split_row (7))
```

Takes the MOTL file specified by `input_motl_fn` and writes out a seperate MOTL file with `output_motl_fn_prfx` as the prefix where each output file corresponds to a unique value of the row `split_row` in `input_motl_fn`.

48.1 Example

```
subtom_split_motl_by_row(...
    'input_motl_fn', 'combinedmotl/allmotl_1.em', ...
    'output_motl_fn', 'combinedmotl/allmotl_1_tomo', ...
    'split_row', 7)
```

48.2 See Also

- *subtom_cat_motls*
- *subtom_clean_motl*
- *subtom_compare_motls*
- *subtom_even_odd_motl*
- *subtom_random_subset_motl*
- *subtom_renumber_motl*
- *subtom_rotx_motl*
- *subtom_scale_motl*
- *subtom_seed_positions*
- *subtom_transform_motl*
- *subtom_unclass_motl*

CHAPTER

FORTYNINE

SUBTOM_TRANSFORM_MOTL

Apply a rotation and a shift to a MOTL file.

```
subtom_transform_motl(
    'input_motl_fn', input_motl_fn (''),
    'output_motl_fn', output_motl_fn (''),
    'shift_x', shift_x (0),
    'shift_y', shift_y (0),
    'shift_z', shift_z (0),
    'rotate_phi', rotate_phi (0),
    'rotate_psi', rotate_psi (0),
    'rotate_theta', rotate_theta (0),
    'rand_inplane', rand_inplane (0))
```

Takes the motl given by `input_motl_fn`, and first applies the rotation described by the Euler angles `rotate_phi`, `rotate_psi`, `rotate_theta`, which correspond to an in-plane spin, azimuthal, and zenithal rotation respectively. Then a translation specified by `shift_x`, `shift_y`, `shift_z`, is applied to the existing translation. Finally the resulting transformed motive list is written out as `output_motl_fn`. Keep in mind that the motive list transforms describe the alignment of the reference to each particle, but that the rotation and shift here describe an affine transform of the reference to a new reference. If `rand_inplane` evaluates to true as a boolean, then the final Euler angle (phi in AV3 notation, and psi/spin/inplane in other notations) will be randomized after the given transform.

49.1 Explanation of how the transforms are derived

```
The alignments in the motive list describe the rotation and shift of the
reference to each particle. Since this is a rotation followed by a shift
we can describe this as an affine transform 4x4 matrix as follows:

[ R_1   T_1 ]   [ V ]   [ P ]
[           ] x [   ] = [   ]                          (1)
[  0     1  ]   [ 1 ]   [ 1 ]

Where R_1 is the rotation matrix described by motl(17:19, 1) and T_1 is
the shift column vector described by motl(11:13, 1), and finally V and P
are the coordinates in the reference, and the reference in register with
the particle respectively.

Likewise the rotation and shift we apply to the reference to get a new
updated reference is also an affine transform as follows:
```

(continues on next page)

(continued from previous page)

```
[ R_2   T_2 ]   [ V ]   [  V' ]
[           ] x [   ] = [     ]                    (2)
[  0     1  ]   [ 1 ]   [  1  ]

Where V' is our new reference. Therefore the affine transform we want to
find and place in our updates motive list is:

[ R_?   T_? ]   [  V' ]   [ P ]
[           ] x [     ] = [   ]                    (3)
[  0     1  ]   [  1  ]   [ 1 ]

The most logical path is to go from V' to V and V to P, so we have to
invert the affine transform in (2), and then left multiply it by the
transform in (1). To find the inverse affine transform of (2) we have
that:

R_2 * V + T_2 = V'
R_2 * V = V' - T_2
V = R_2^-1 * (V' - T_2)
V = (R_2^-1 * V') - (R_2^-1 * T_2)

[ R_2^-1   -R_2^-1 * T_2 ]   [  V' ]   [ V ]
[                        ] x [     ] = [   ]         (4)
[    0             1     ]   [  1  ]   [ 1 ]

So we have that:

[ R_1  T_1 ]   [ R_2^-1   -R_2^-1 * T_2 ]   [  V' ]   [ P ]
[          ] x [                        ] x [     ] = [   ]         (5)
[  0    1  ]   [    0             1     ]   [  1  ]   [ 1 ]

[ R_1 * R_2^-1  -R_1 * R_2^-1 * T_2 + T_1 ]   [  V' ] = [ P ]
[                                         ] x [     ] = [   ]       (6)
[     0                      1            ]   [  1  ] = [ 1 ]

And finally:

R_? = R_1 * R_2^-1
T_? = T_1 - R_1 * R_2^-1 * T_2
```

49.2 Example

```
subtom_transform_motl(...
    'input_motl_fn', 'combinedmotl/allmotl_1.em', ...
    'output_motl_fn', 'combinedmotl/allmotl_1_shifted.em', ...
    'shift_x', 5, ...
    'shift_y', 5, ...
    'shift_z', -3, ...
    'rotate_phi', 60, ...
    'rotate_psi', 15, ...
```

(continues on next page)

(continued from previous page)

```
    'rotate_theta', 0.5, ...
    'rand_inplane', 0)
```

49.3 See Also

- *subtom_cat_motls*
- *subtom_clean_motl*
- *subtom_compare_motls*
- *subtom_even_odd_motl*
- *subtom_random_subset_motl*
- *subtom_renumber_motl*
- *subtom_rotx_motl*
- *subtom_scale_motl*
- *subtom_seed_positions*
- *subtom_split_motl_by_row*
- *subtom_unclass_motl*

CHAPTER

FIFTY

SUBTOM_WEIGHTED_AVERAGE

Joins and weights parallel average subsets.

```
subtom_weighted_average(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ref_fn_prefix', ref_fn_prefix ('ref/ref'),
    'weight_sum_fn_prefix', weight_sum_fn_prefix ('otherinputs/wei'),
    'iteration', iteration (1),
    'iclass', iclass (0),
    'num_avg_batch', num_avg_batch (1))
```

Takes the `num_avg_batch` parallel sum subsets with the name prefix `ref_fn_prefix`, the all_motl file with name prefix `motl_fn_prefix` and weight volume subsets with the name prefix `weight_sum_fn_prefix` to generate the final average, which should then be used as the reference for iteration number `iteration`. `iclass` describes which class outside of one is included in the final average and is used to correctly scale the average and weights.

50.1 Example

```
subtom_weighted_average(...
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ref_fn_prefix', './ref/ref', ...
    'weight_sum_fn_prefix', 'otherinputs/wei', ...
    'iteration', 1, ...
    'iclass', 0, ...
    'num_avg_batch', 1)
```

50.2 See Also

- *subtom_extract_noise*
- *subtom_extract_subtomograms*
- *subtom_parallel_sums*
- *subtom_scan_angles_exact*

CHAPTER

FIFTYONE

SUBTOM_UNCLASS_MOTL

Removes the iclass information from a motive list.

```
subtom_unclass_motl(
    'input_motl_fn', input_motl_fn (''),
    'output_motl_fn', output_motl_fn (''))
```

Takes the motive list given by `input_motl_fn`, and removes all of the iclass information setting all of the particles to 1, this can be useful when moving from classified motive lists to all-particle alignments. Then unclassed motive list is written out as `output_motl_fn`.

51.1 Example

```
subtom_unclass_motl(...
    'input_motl_fn', 'combinedmotl/allmotl_wmd_2.em', ...
    'output_motl_fn', 'combinedmotl/allmotl_wmd_unclassed_2.em')
```

51.2 See Also

- *subtom_cat_motls*
- *subtom_clean_motl*
- *subtom_compare_motls*
- *subtom_even_odd_motl*
- *subtom_random_subset_motl*
- *subtom_renumber_motl*
- *subtom_rotx_motl*
- *subtom_scale_motl*
- *subtom_seed_positions*
- *subtom_split_motl_by_row*
- *subtom_transform_motl*

CHAPTER

FIFTYTWO

SUBTOM: GENERAL CLASSIFICATION UTILITIES

52.1 Classification in subTOM

There are several classification strategies within subTOM. Three are variants of Multivariate Statistical Analysis (MSA), and the final is Multireference alignment. Within all of these there are some functions which are generally applicable and so they are shared between each modus of classification. This includes the clustering of MSA data (`subtom_cluster`), as well as the averaging of data that has been classified (`subtom_parallel_sums_cls` and `subtom_weighted_average_cls`). Since classification, particularly prinicipal component analysis (PCA), is very operation intensive there is also a function to pre-align all extracted particles and write them to disk to speed up later processing steps (`subtom_parallel_prealign`).

52.2 subtom_align_refs

Aligns the class averages from a given MOTL file and then applies the found rotations to class particles and re-averages the classes and all particle average in parallel on the cluster or locally.

This subtomogram alignment and averaging script uses five MATLAB compiled scripts below:

- *subtom_scan_angles_exact_refs*
- *subtom_parallel_sums*
- *subtom_parallel_sums_cls*
- *subtom_weighted_average*
- *subtom_weighted_average_cls*

52.2.1 Options

Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

local_dir Absolute path to the folder on a group share, if the scratch directory is cleaned and deleted regularly this can set a local directory to which the important results will be copied. If this is not needed it can be skipped with the option skip_local_copy below.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

Variables

align_exec Alignment executable

sum_exec Parallel Summing executable

avg_exec Weighted Averaging executable

sum_cls_exec Class Average Parallel Summing executable

avg_cls_exec Class Average Final Averaging executable

motl_dump_exec MOTL dump executable

Memory Options

mem_free The amount of memory the job requires for alignment. This variable determines whether a number of CPUs will be requested to be dedicated for each job. At 24G, one half of the CPUs on a node will be dedicated for each of the processes (12 CPUs). At 48G, all of the CPUs on the node will be dedicated for each of the processes (24 CPUs).

mem_max The upper bound on the amount of memory the alignment job is allowed to use. If any of the processes request or require more memory than this, the queue will kill the process. This is more of an option for safety of the cluster to prevent the user from crashing the cluster requesting too much memory.

Other Cluster Options

job_name The job name prefix that will be used for the cluster submission scripts, log files, and error logs for the processing. Be careful that this name is unique because previous submission scripts, logs, and error logs with the same job name prefix will be overwritten in the case of a name collision.

array_max The maximum number of jobs per cluster submission script. Cluster submission scripts work using the array feature common to queuing systems, and this value is the maximum array size used in a script. If the user requests more batches of processing than this value, then the submission scripts will be split into files of up to array_max jobs.

max_jobs The maximum number of jobs for alignment. If the number of batches / exceeds this value the script will immediately quit.

run_local If the user wants to skip the cluster and run the job locally, this value should be set to 1.

skip_local_copy Set this option to 1 to skip the copying of data to local_dir.

Subtomogram Averaging Workflow Options

Parallelization Options

iteration The index of the reference to generate : input will be all_motl_fn_prefix_iteration.em (define as integer)

num_avg_batch The number of batches to split the parallel subtomogram averaging job into.

File Options

all_motl_fn_prefix Relative path and name of the concatenated motivelist of all particles (e.g. allmotl_iter.em , the variable will be written as a string e.g. all_motl_fn_prefix='sub-directory/allmotl')

output_motl_fn_prefix Relative path and name prefix of the output motivelist of all particles. There will be two versions written out. The first with "_classed_" will retain the iclass values to generate new aligned class averages. The second with "_unclassed_" will set all particles iclass to 1 to generate a cumulative class average.

ref_fn_prefix Relative path and name of the reference volumes (e.g. ref_iter.em , the variable will be written as a string e.g. ref_fn_prefix='sub-directory/ref')

ptcl_fn_prefix Relative path and name of the subtomograms (e.g. part_n.em , the variable will be written as a string e.g. ptcl_fn_prefix='sub-directory/part')

align_mask_fn Relative path and name of the alignment mask. If "none" is given a default spherical mask will be used. (e.g. align_mask_fn='otherinputs/align_mask.em')

cc_mask_fn Relative path and name of the cross-correlation mask this defines the maximum shifts in each direction. If "noshift" is given no shifts are allowed. (e.g. cc_mask_fn='otherinputs/cc_mask_1.em')

weight_fn_prefix Relative path and name of the weight file.

weight_sum_fn_prefix Relative path and name of the partial weight files.

Alignment and Averaging Options

tomo_row Which row in the motl file contains the correct tomogram number. Usually row 5 and 7 both correspond to the correct value and can be used interchangeably, but there are instances when 5 contains a sequential ordered value starting from 1, while 7 contains the correct corresponding tomogram.

ref_class Which class average to align the other class averages against. Because of the AV3 specification for iclass this should be a number that is 3 or above.

apply_mask Apply mask to class averages (1=yes, 0=no)

psi_angle_step Angular increment in degrees, applied during the cone-search, i.e. psi and theta (define as real e.g. psi_angle_step=3)

psi_angle_shells Number of angular iterations, applied to psi and theta (define as integer e.g. psi_angle_shells=3)

phi_angle_step Angular increment for phi in degrees, (define as real e.g. phi_angle_step=3)

phi_angle_shells Number of angular iterations for phi, (define as integer e.g. phi_angle_shells=3)

high_pass_fp High pass filter cutoff (in transform units (pixels): calculate as (box_size*pixelsize)/(resolution_real) (define as integer e.g. high_pass_fp=2)

high_pass_sigma High pass filter falloff sigma (in transform units (pixels): describes a Gaussian sigma for the falloff of the high-pass filter past the cutoff above.

low_pass_fp Low pass filter (in transform units (pixels): calculate as (box_size*pixelsize)/(resolution_real) (define as integer e.g. low_pass_fp=30).

low_pass_sigma Low pass filter falloff sigma (in transform units (pixels): describes a Gaussian sigma for the falloff of the low-pass filter past the cutoff above.

nfold Symmetry, if no symmetry nfold=1 (define as integer e.g. nfold=3)

52.2.2 Example

```
scratch_dir="${PWD}"

local_dir=""

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

sum_exec="${exec_dir}/alignment/subtom_parallel_sums"

avg_exec="${exec_dir}/alignment/subtom_weighted_average"

sum_exec="${exec_dir}/classification/general/subtom_parallel_sums_cls"

avg_exec="${exec_dir}/classification/general/subtom_weighted_average_cls"

mem_free=1G

mem_max=64G

job_name=subTOM

array_max=1000

max_jobs=4000

run_local=0

skip_local_copy=1

iteration=1

num_avg_batch=1

all_motl_fn_prefix="combinedmotl/allmotl"

output_motl_fn_prefix="combinedmotl/allmotl"

ref_fn_prefix="ref/ref"

ptcl_fn_prefix="subtomograms/subtomo"

align_mask_fn="otherinputs/align_mask.em"

cc_mask_fn="otherinputs/cc_mask.em"

weight_fn_prefix="otherinputs/ampspec"

weight_sum_fn_prefix="otherinputs/wei"

tomo_row=7
```

(continues on next page)

(continued from previous page)

```
ref_class=3

apply_mask=0

psi_angle_step=1

psi_angle_step=6

phi_angle_step=1

phi_angle_shells=10

high_pass_fp=1

high_pass_sigma=2

low_pass_fp=10

low_pass_sigma=3

nfold=1
```

52.3 subtom_cluster

Clusters a motive-list using pre-calculated and supplied coefficients and outputs a classified motive-list and class averages.

This subtomogram classification script uses three MATLAB compiled scripts below:

- *subtom_cluster*
- *subtom_parallel_sums_cls*
- *subtom_weighted_average_cls*

52.3.1 Options

Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

local_dir Absolute path to the folder on a group share, if the scratch directory is cleaned and deleted regularly this can set a local directory to which the important results will be copied. If this is not needed it can be skipped with the option skip_local_copy below.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

Variables

cluster_exec Cluster executable.

sum_exec Parallel Summing executable

avg_exec Final Averaging executable

Memory Options

mem_free The amount of memory the job requires. This variable determines whether a number of CPUs will be requested to be dedicated for each job. At 24G, one half of the CPUs on a node will be dedicated for each of the processes (12 CPUs). At 48G, all of the CPUs on the node will be dedicated for each of the processes (24 CPUs).

mem_max The upper bound on the amount of memory the job is allowed to use. If any of the processes request or require more memory than this, the queue will kill the process. This is more of an option for safety of the cluster to prevent the user from crashing the cluster requesting too much memory.

Other Cluster Options

job_name The job name prefix that will be used for the cluster submission scripts, log files, and error logs for the processing. Be careful that this name is unique because previous submission scripts, logs, and error logs with the same job name prefix will be overwritten in the case of a name collision.

array_max The maximum number of jobs per cluster submission script. Cluster submission scripts work using the array feature common to queuing systems, and this value is the maximum array size used in a script. If the user requests more batches of processing than this value, then the submission scripts will be split into files of up to array_max jobs.

max_jobs The maximum number of jobs for alignment. If the number of batches / exceeds this value the script will immediately quit.

run_local If the user wants to skip the cluster and run the job locally, this value should be set to 1.

skip_local_copy Set this option to 1 to skip the copying of data to local_dir.

Parallelization Options

iteration The index of the references to generate : input will be all_motl_fn_prefix_iteration.em (define as integer e.g. iteration=1)

num_avg_batch The number of batches to split the parallel subtomogram averaging job into.

Subtomogram Classification Workflow Options

Coefficient File Options

coeff_all_motl_fn_prefix Relative path and name of the concatenated motivelist to cluster and classify.

coeff_fn_prefix Relative path and name of the coefficients.

Clustering Options

cluster_type The following determines which algorithm will be used to cluster the determined Eigencoefficients. The valid options are K-means clustering, 'kmeans', Hierarchical Ascendent Clustering using a Ward Criterion, 'hac', and a Gaussian Mixture Model, 'gaussmix'.

coeff_idxs Determines which coefficients are used to cluster. The format should be a semicolon-separated list that also supports ranges with a dash (-), for example 1-5;7;15-19 would select the first five coefficients, the seventh and the fifteenth through the nineteenth for classification. If it is left as "all" all coefficients will be used.

num_classes How many classes should the particles be clustered into.

Clustering File Options

cluster_all_motl_fn_prefix Relative path and name of the concatenated motivelist of the output classified particles.

Averaging File Options

ref_fn_prefix Relative path and name prefix of the reference volumes (e.g. ref_iter.em, the variable will be written as a string e.g. ref_fn_prefix='sub-directory/ref')

weight_sum_fn_prefix Relative path and name prefix of the partial weight files.

52.3.2 Example

```
scratch_dir="${PWD}"

local_dir=""

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="XXXINSTALLATION_DIRXXX/bin"

cluster_exec="${exec_dir}/classification/pca/subtom_cluster"

sum_exec="${exec_dir}/classification/pca/subtom_parallel_sums"

avg_exec="${exec_dir}/classification/pca/subtom_weighted_average"

mem_free="1G"

mem_max="64G"

job_name="subTOM"

array_max="1000"

max_jobs="4000"

run_local="0"

skip_local_copy="1"
```

(continues on next page)

(continued from previous page)

```
iteration="1"

num_avg_batch="1"

coeff_all_motl_fn_prefix="combinedmotl/allmotl"

coeff_fn_prefix="class/coeffs"

cluster_type="kmeans"

eig_idxs="all"

num_classes=2

cluster_all_motl_fn_prefix="class/allmotl_class"

ref_fn_prefix="class/ref"

weight_sum_fn_prefix="class/wei"
```

52.4 subtom_multiref_average

Calculates the average from a given MOTL file in parallel on the cluster or locally.

This subtomogram averaging script uses two MATLAB compiled scripts below:

- *subtom_parallel_sums_cls*
- *subtom_weighted_average_cls*

52.4.1 Options

Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

local_dir Absolute path to the folder on a group share, if the scratch directory is cleaned and deleted regularly this can set a local directory to which the important results will be copied. If this is not needed it can be skipped with the option skip_local_copy below.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

Variables

sum_exec Parallel Summing executable

avg_exec Weighted Averaging executable

Memory Options

mem_free The amount of memory the job requires for alignment. This variable determines whether a number of CPUs will be requested to be dedicated for each job. At 24G, one half of the CPUs on a node will be dedicated for each of the processes (12 CPUs). At 48G, all of the CPUs on the node will be dedicated for each of the processes (24 CPUs).

mem_max The upper bound on the amount of memory the alignment job is allowed to use. If any of the processes request or require more memory than this, the queue will kill the process. This is more of an option for safety of the cluster to prevent the user from crashing the cluster requesting too much memory.

Other Cluster Options

job_name The job name prefix that will be used for the cluster submission scripts, log files, and error logs for the processing. Be careful that this name is unique because previous submission scripts, logs, and error logs with the same job name prefix will be overwritten in the case of a name collision.

array_max The maximum number of jobs per cluster submission script. Cluster submission scripts work using the array feature common to queuing systems, and this value is the maximum array size used in a script. If the user requests more batches of processing than this value, then the submission scripts will be split into files of up to array_max jobs.

max_jobs The maximum number of jobs for alignment. If the number of batches / exceeds this value the script will immediately quit.

run_local If the user wants to skip the cluster and run the job locally, this value should be set to 1.

skip_local_copy Set this option to 1 to skip the copying of data to local_dir.

Subtomogram Averaging Workflow Options

Parallelization Options

iteration The index of the reference to generate : input will be all_motl_fn_prefix_iteration.em (define as integer)

num_avg_batch The number of batches to split the parallel subtomogram averaging job into.

File Options

all_motl_fn_prefix Relative path and name of the concatenated motivelist of all particles (e.g. allmotl_iter.em , the variable will be written as a string e.g. all_motl_fn_prefix='sub-directory/allmotl')

ref_fn_prefix Relative path and name of the reference volumes (e.g. ref_iter.em , the variable will be written as a string e.g. ref_fn_prefix='sub-directory/ref')

ptcl_fn_prefix Relative path and name of the subtomograms (e.g. part_n.em , the variable will be written as a string e.g. ptcl_fn_prefix='sub-directory/part')

weight_fn_prefix Relative path and name of the weight file.

weight_sum_fn_prefix Relative path and name of the partial weight files.

Averaging Options

tomo_row Which row in the motl file contains the correct tomogram number. Usually row 5 and 7 both correspond to the correct value and can be used interchangeably, but there are instances when 5 contains a sequential ordered value starting from 1, while 7 contains the correct corresponding tomogram.

52.4.2 Example

```
scratch_dir="${PWD}"

local_dir=""

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

sum_exec="${exec_dir}/classification/general/subtom_parallel_sums_cls"

avg_exec="${exec_dir}/classification/general/subtom_weighted_average_cls"

mem_free=1G

mem_max=64G

job_name=subTOM

array_max=1000

max_jobs=4000

run_local=0

skip_local_copy=1

iteration=1

num_avg_batch=1

all_motl_fn_prefix="combinedmotl/allmotl"

ref_fn_prefix="ref/ref"

ptcl_fn_prefix="subtomograms/subtomo"

weight_fn_prefix="otherinputs/ampspec"

weight_sum_fn_prefix="otherinputs/wei"

tomo_row=7
```

52.5 subtom_cluster

Classifies particles based on given coefficients.

```
subtom_cluster(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'coeff_fn_prefix', coeff_fn_prefix ('class/coeff'),
    'output_motl_fn_prefix', output_motl_fn_prefix ('class/allmotl'),
    'iteration', iteration (1),
    'cluster_type', cluster_type ('kmeans'),
    'eig_idxs', eig_idxs ('all'),
    'num_classes', num_classes ('2'))
```

Takes the motive list given by `all_motl_fn_prefix` and the coefficients specified by `coeff_fn_prefix` for the iteration `iteration` and clusters the data based on the coefficients. Clustering can be done using one of three methods, which are specfied by `cluster_type`. The options are K-Means clustering with 'kmeans', Hierarchical Ascendant Clustering with 'hac' and a Gaussian Mixture Model with 'gaussmix'. A subset of coefficients can be selected and are given as a semicolon-separated string of indices as `coeff_idxs`. The string can also contain ranges delimited by a dash, for example '1;3;5-10'. The data will be clustered into `num_classes` number of clusters and the clustered motive list will be written out to a file given by `output_motl_fn_prefix`.

52.5.1 Example

```
subtom_cluster(...
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'coeff_fn_prefix', 'class/eigcoeff_msa', ...
    'output_motl_fn_prefix', 'class/allmotl_msa', ...
    'iteration', 1, ...
    'cluster_type', 'hac', ...
    'coeff_idxs', '2-5;7;9-20', ...
    'num_classes', '20')
```

52.5.2 See Also

- *subtom_parallel_prealign*
- *subtom_parallel_sums_cls*
- *subtom_scan_angles_exact_refs*
- *subtom_weighted_average_cls*

52.6 subtom_parallel_prealign

Prealigns particles to speed up CC-Matrix calculation.

```
subtom_parallel_prealign(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'prealign_fn_prefix', prealign_fn_prefix ('subtomograms/subtomo'),
    'iteration', iteration (1),
    'num_prealign_batch', num_prealign_batch (1),
    'process_idx', process_idx (1))
```

Prerotates and translates particles into alignment as precalculation on disk to speed up the calculation of the constrained cross-correlation matrix. The alignments are given in the motive list specified by `all_motl_fn_prefix` and `iteration`, and the particles are based on `ptcl_fn_prefix` and # where # is described in row 4 of the motive list. Pre-aligned particles will be written out described by `prealign_fn_prefix`, `iteration` and #. The process is designed to be run in parallel on a cluster. The particles will be processed in `num_prealign_batch` chunks, with the specific chunk being processed described by `process_idx`.

52.6.1 Example

```
subtom_parallel_prealign(
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ptcl_fn_prefix', 'subtomograms/subtomo', ...
    'prealign_fn_prefix', 'subtomograms/subtomo_ali', ...
    'iteration', 1, ...
    'num_prealign_batch', 100, ...
    'process_idx', 1)
```

52.6.2 See Also

- *subtom_cluster*
- *subtom_parallel_sums_cls*
- *subtom_scan_angles_exact_refs*
- *subtom_weighted_average_cls*

52.7 subtom_parallel_sums_cls

Creates raw sums and Fourier weight sums in a batch.

```
subtom_parallel_sums_cls(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ref_fn_prefix', ref_fn_prefix ('ref/ref'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'weight_fn_prefix', weight_fn_prefix ('otherinputs/ampspec'),
    'weight_sum_fn_prefix, weight_sum_fn_prefix ('otherinputs/wei'),
    'iteration', iteration (1),
```

(continues on next page)

(continued from previous page)

```
    'tomo_row', tomo_row (7),
    'num_avg_batch', num_avg_batch (1),
    'process_idx', process_idx (1))
```

Aligns a subset of particles using the rotations and shifts in `all_motl_fn_prefix` _#.em where # corresponds to `iteration` in `num_avg_batch` chunks to make a raw particle sum `ref_fn_prefix` _#_###.em where # corresponds to `iteration` and ### corresponds to `process_idx`. Fourier weight volumes with name prefix `weight_fn_prefix` will also be aligned and summed to make a weight sum `weight_sum_fn_prefix` _#_###.em. `tomo_row` describes which row of the motl file is used to determine the correct tomogram fourier weight file. In this multi-reference version of parallel sums, each unique value of iclass (row 20 in the motive list) will be summed and written out (excluding non-positive values).

52.7.1 Example

```
subtom_parallel_sums_cls(...
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ref_fn_prefix', 'ref/ref', ...
    'ptcl_fn_prefix', 'subtomograms/subtomo', ...
    'weight_fn_prefix', 'otherinputs/ampspec', ...
    'weight_sum_fn_prefix, 'otherinputs/wei', ...
    'iteration', 1, ...
    'tomo_row', 7, ...
    'num_avg_batch', 1, ...
    'process_idx', 1)
```

52.7.2 See Also

- *subtom_cluster*
- *subtom_parallel_prealign*
- *subtom_scan_angles_exact_refs*
- *subtom_weighted_average_cls*

52.8 subtom_scan_angles_exact_refs

Align a particle class averages to a single class average reference.

```
subtom_scan_angles_exact_refs(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ref_fn_prefix', ref_fn_prefix ('ref/ref'),
    'align_mask_fn', align_mask_fn ('none'),
    'cc_mask_fn', cc_mask_fn ('noshift'),
    'apply_mask', apply_mask (0),
    'ref_class', ref_class (3),
    'psi_angle_step', psi_angle_step (0),
    'psi_angle_shells', psi_angle_shells (0),
    'phi_angle_step', phi_angle_step (0),
```

(continues on next page)

(continued from previous page)

```
    'phi_angle_shells', phi_angle_shells (0),
    'high_pass_fp', high_pass_fp (0),
    'high_pass_sigma', high_pass_sigma (0),
    'low_pass_fp', low_pass_fp (0),
    'low_pass_sigma', low_pass_sigma (0),
    'nfold', nfold (1),
    'iteration', iteration (1))
```

Aligns class averages from the collective motive list with the name format `all_motl_fn_prefix` _#.em where # is the number `iteration`. A motive list for the best determined alignment parameters against the class average specified by `ref_class` is written out in two motive lists as given by `output_motl_fn_prefix`. The first with 'classed' keeps the class information to generate new class averages. The second with 'unclassed' discards the class information so a cumulative average can be generated.

Class averages, with the name format `ref_fn_prefx` _class_#_#.em where the first # is the iclass number, and the the second # is `iteration`, are aligned against the reference class average. Before the comparison is made a number of alterations are made to both the class average and reference:

- If `nfold` is greater than 1 then C#-symmetry is applied along the Z-axis to the reference where # is `nfold`.
- The reference is masked in real space with the mask `align_mask_fn`, and if `apply_mask` evaluates to true as a boolean, then this mask is also applied to the class average. A sphere mask is applied to the particle to reduces the artifacts caused by the box-edges on the comparison. This sphere has a diameter that is 80% the box size and falls of with a sigma that is 15% half the box size.
  - `apply_mask` can help alignment and suppress alignment to other features when the particle is well-centered or already reasonably well aligned, but if this is not the case there is the risk that a tight alignment will cutoff parts of the particle.
- Both the particle and the reference are bandpass filtered in the Fourier domain defined by `high_pass_fp`, `high_pass_sigma`, `low_pass_fp`, and `low_pass_sigma` which are all in the units of Fourier pixels.

The local rotations searched during alignment are deteremined by the four parameters `psi_angle_step`, `psi_angle_shells`, `phi_angle_step`, and `phi_angle_shells`. They describe a search where the currently existing alignment parameters for azimuth and zenith are used to define a "pole" to search about in the ceiling of half `psi_angle_shells` cones. The change in zenith between each cone is `psi_angle_step` and the azimuth around the cone is close to the same angle but is adjusted slightly to account for bias near the pole. The final spin angle of the search is done with a change in spin of `phi_angle_step` in `phi_angle_shells` steps. The spin is applied in both clockwise and counter-clockwise fashion.

- The angles phi, and psi here are flipped in their sense of every other package for EM image processing, which is absolutely infuriating and confusing, but maintained for historical reasons, however most descriptions use the words azimuth, zenith, and spin to avoid ambiguity.

Finally after the constrained cross-correlation function is calculated it is masked with `cc_mask_fn` to limit the shifts to inside this volume, and a peak is found and it's location is determined to sub-pixel accuracy using interpolation. The rotations and shifts that gives the highest cross-correlation coefficient are then chosen as the new alignments parameters.

52.8.1 Example

```
subtom_scan_angles(...
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ref_fn_prefix', 'ref/ref', ...
    'align_mask_fn', 'otherinputs/align_mask.em', ...
    'cc_mask_fn', 'otherinputs/cc_mask.em', ...
    'apply_mask', 1, ...
    'ref_class', 3, ...
    'psi_angle_step', 1, ...
    'psi_angle_shells', 8, ...
    'phi_angle_step', 1, ...
    'phi_angle_shells', 8, ...
    'high_pass_fp', 1, ...
    'high_pass_sigma', 2, ...
    'low_pass_fp', 10, ...
    'low_pass_sigma', 3, ...
    'nfold', 1, ...
    'iteration', 1)
```

52.8.2 See Also

- *subtom_cluster*
- *subtom_parallel_prealign*
- *subtom_parallel_sums_cls*
- *subtom_weighted_average_cls*

52.9 subtom_weighted_average_cls

Joins and weights parallel average subsets.

```
subtom_weighted_average_cls(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ref_fn_prefix', ref_fn_prefix ('ref/ref'),
    'weight_sum_fn_prefix', weight_sum_fn_prefix ('otherinputs/wei'),
    'iteration', iteration (1),
    'num_avg_batch', num_avg_batch (1))
```

Takes the `num_avg_batch` parallel sum subsets with the name prefix `ref_fn_prefix`, the all_motl file with name prefix `motl_fn_prefix` and weight volume subsets with the name prefix `weight_sum_fn_prefix` to generate the final average, which should then be used as the reference for iteration number `iteration`.

52.9.1 Example

```
subtom_weighted_average_cls(...
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ref_fn_prefix', './ref/ref', ...
    'weight_sum_fn_prefix', 'otherinputs/wei', ...
    'iteration', 1, ...
    'num_avg_batch', 1)
```

52.9.2 See Also

- *subtom_cluster*
- *subtom_parallel_prealign*
- *subtom_parallel_sums_cls*
- *subtom_scan_angles_exact_refs*

52.10 Indices and tables

- genindex
- modindex
- search

CHAPTER

FIFTYTHREE

SUBTOM: MULTIVARIATE STATISTICAL ANALYSIS

53.1 Multivariate Statistical Analysis Classification

In Multivariate Statistical Analysis (MSA) classification the full set of particles are simplified into a new lower-dimensional representation by means of EigenValue decomposition. Particles projected onto these most variable basis-vectors then can be clustered using a variety of methods.

Within subTOM particles are first compiled into a 2-D Matrix denoted here as the X-Matrix, which holds the aligned, band-pass filtered and masked particle data. To speed up calculation particles can be pre-aligned using the function `subtom_parallel_prealign`. Batches of the X-Matrix are calculated in parallel with `subtom_parallel_xmatrix_msa` and then combined and column-centered with `subtom_join_xmatrix`.

Next the X-Matrix is used to calculated the covariance matrix which is scaled using the so-called 'modulation metric' as described in L. Borland and M. van Heel in J. Opt. Soc. Am. A 1990, which is similar to the Chi-Square metrics used in Correspondance Analysis of ordinal data. This covariance matrix is then decomposed into it's Eigenvectors and Eigenvalues and these are used along with the X-Matrix to determine the Eigenvolumes of the dataset with `subtom_eigenvolumes_msa`.

These volumes are then used to determine the low-rank approximation coefficients in volume space for clustering. A larger particle superset can be projected onto the volumes to speed up classification of large datasets. Coefficients are also calculated in parallel in batches with `subtom_parallel_eigcoeffs_msa` and joined with `subtom_join_eigencoeffs_msa`.

Finally using a user-selected subset of the determined coefficients, the data is clustered either by Hierarchical Ascendant Clustering using a Ward distance criterion, K-Means clustering, or a Gaussian Mixture model with the function `subtom_cluster`. This clustering is then used to generate the final class averages.

53.2 subtom_msa

The main MSA pipeline process script of subTOM.

This subtomogram classification script uses nine MATLAB compiled scripts below:

- *subtom_parallel_prealign*
- *subtom_parallel_xmatrix_msa*
- *subtom_join_xmatrix*
- *subtom_eigenvolumes_msa*
- *subtom_parallel_eigencoeffs_msa*
- *subtom_join_eigencoeffs_msa*

- *subtom_cluster*
- *subtom_parallel_sums_cls*
- *subtom_weighted_average_cls*

53.2.1 Options

Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

local_dir Absolute path to the folder on a group share, if the scratch directory is cleaned and deleted regularly this can set a local directory to which the important results will be copied. If this is not needed it can be skipped with the option skip_local_copy below.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

Variables

cluster_exec Cluster executable.

par_eigcoeff_exec Parallel Eigencoefficient executable.

eigcoeff_exec Final Eigencoefficient executable.

eigvol_exec Eigenvolume Calculation executable.

preali_exec Parallel Subtomogram prealign executable.

par_xmatrix_exec Parallel X-Matrix executable.

xmatrix_exec Final X-Matrix executable.

sum_exec Parallel Summing executable

avg_exec Final Averaging executable

motl_dump_exec MOTL dump executable

Memory Options

mem_free The amount of memory the job requires. This variable determines whether a number of CPUs will be requested to be dedicated for each job. At 24G, one half of the CPUs on a node will be dedicated for each of the processes (12 CPUs). At 48G, all of the CPUs on the node will be dedicated for each of the processes (24 CPUs).

mem_max The upper bound on the amount of memory the job is allowed to use. If any of the processes request or require more memory than this, the queue will kill the process. This is more of an option for safety of the cluster to prevent the user from crashing the cluster requesting too much memory.

Other Cluster Options

job_name The job name prefix that will be used for the cluster submission scripts, log files, and error logs for the processing. Be careful that this name is unique because previous submission scripts, logs, and error logs with the same job name prefix will be overwritten in the case of a name collision.

array_max The maximum number of jobs per cluster submission script. Cluster submission scripts work using the array feature common to queuing systems, and this value is the maximum array size used in a script. If the user requests more batches of processing than this value, then the submission scripts will be split into files of up to array_max jobs.

max_jobs The maximum number of jobs for alignment. If the number of batches / exceeds this value the script will immediately quit.

run_local If the user wants to skip the cluster and run the job locally, this value should be set to 1.

skip_local_copy Set this option to 1 to skip the copying of data to local_dir.

Parallelization Options

iteration The index of the references to generate : input will be all_motl_fn_prefix_iteration.em (define as integer e.g. iteration=1)

num_xmatrix_prealign_batch Number of batches to split the parallel particle prealignment for the X-Matrix calculation into. If you are not doing prealignment you can ignore this option.

num_xmatrix_batch Number of batches to split the parallel X-Matrix calculation job into.

num_eig_coeff_prealign_batch Number of batches to split the parallel particle prealignment for the Eigencoefficients calculations into. If you are not doing prealignment you can ignore this option.

num_eig_coeff_batch Number of batches to split the parallel Eigencoefficient calculation into.

num_avg_batch The number of batches to split the parallel subtomogram averaging job into.

Subtomogram Classification Workflow Options

X-Matrix Options

high_pass_fp High pass filter cutoff (in transform units (pixels): calculate as (box_size * pixelsize) / (resolution_real) (define as integer).

high_pass_sigma High pass filter falloff sigma (in transform units (pixels): describes a Gaussian sigma for the falloff of the high-pass filter past the cutoff above.

low_pass_fp Low pass filter (in transform units (pixels): calculate as (box_size * pixelsize) / (resolution_real) (define as integer).

low_pass_sigma Low pass filter falloff sigma (in transform units (pixels): describes a Gaussian sigma for the falloff of the low-pass filter past the cutoff above.

nfold Symmetry to apply to each pair of particle and reference in X-Matrix calculation, if no symmetry nfold=1 (define as integer e.g. nfold=3).

xmatrix_prealign If you want to pre-align all of the particles to speed up the X-Matrix calculation, set the following to 1, otherwise the particles will be aligned during the computation.

X-Matrix File Options

xmatrix_all_motl_fn_prefix Relative path and name of the concatenated motivelist of all particles (e.g. all-motl_iter.em , the variable will be written as a string e.g. xmatrix_all_motl_fn_prefix='sub-directory/allmotl').

xmatrix_fn_prefix Relative path and name of the X-Matrix.

ptcl_fn_prefix Relative path and name of the subtomograms (e.g. part_n.em , the variable will be written as a string e.g. ptcl_fn_prefix='sub-directory/part').

mask_fn Relative path and name of the classification mask. This should be a binary mask as correlations are done in real-space, and calculations will only be done using voxels passed by the mask, so smaller masks will run faster. If you want to use the default spherical mask set mask_fn to 'none'.

Eigenvolumes Options

num_eigs The number of Eigenvectors and Eigenvalues to calculate.

eigs_iterations The following allows you to adjust the number of iterations to use in the decomposition. If you want to use the default number of iterations leave this set to 'default'.

eigs_tolerance The following allows you to adjust the convergence tolerance of the decomposition calculation. If you want to use the default tolerance leave this set to 'default'.

Eigenvolumes File Options

eig_vec_fn_prefix Relative path and name of the Eigenvectors.

eig_val_fn_prefix Relative path and name of the Eigenvalues.

eig_vol_fn_prefix Relative path and name of the Eigenvolumes.

Eigencoefficient Options

apply_weight If the following is set to 1, the Eigenvolume will have the particles missing-wedge weight applied to it before the correlation is calculated.

tomo_row Which row in the motl file contains the correct tomogram number. Usually row 5 and 7 both correspond to the correct value and can be used interchangeably, but there are instances when 5 contains a sequential ordered value starting from 1, while 7 contains the correct corresponding tomogram.

eig_coeff_prealign If you want to pre-align all of the particles to speed up the Eigencoefficient calculation, set the following to 1, otherwise the particles will be aligned during the computation.

Eigencoefficient File Options

eig_coeff_all_motl_fn_prefix Relative path and name of the concatenated motivelist to project onto the Eigenvolumes. This can be a larger motivelist than the one used to calculate the X-Matrix and Eigenvolumes.

eig_coeff_fn_prefix Relative path and name of the Eigencoefficients.

weight_fn_prefix Relative path and name of the weight file, if you are not applying the weight to the Eigenvolumes this can be left alone.

Clustering Options

cluster_type The following determines which algorithm will be used to cluster the determined Eigencoefficients. The valid options are K-means clustering, 'kmeans', Hierarchical Ascendent Clustering using a Ward Criterion, 'hac', and a Gaussian Mixture Model, 'gaussmix'.

eig_idxs Determines which Eigencoefficients are used to cluster. The format should be a semicolon-separated list that also supports ranges with a dash (-), for example 1-5;7;15-19 would select the first five Eigencoefficients, the seventh and the fifteenth through the nineteenth for classification. If it is left as "all" all coefficients will be used.

num_classes How many classes should the particles be clustered into.

Clustering File Options

cluster_all_motl_fn_prefix Relative path and name of the concatenated motivelist of the output classified particles.

Averaging File Options

ref_fn_prefix Relative path and name prefix of the reference volumes (e.g. ref_iter.em, the variable will be written as a string e.g. ref_fn_prefix='sub-directory/ref')

weight_sum_fn_prefix Relative path and name prefix of the partial weight files.

53.2.2 Example

```
scratch_dir="${PWD}"

local_dir=""

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="XXXINSTALLATION_DIRXXX/bin"

cluster_exec="${exec_dir}/classification/general/subtom_cluster"

par_eigcoeff_exec="${exec_dir}/classification/msa/subtom_parallel_eigencoeffs_msa"

eigcoeff_exec="${exec_dir}/classification/msa/subtom_join_eigencoeffs_msa"

eigvol_exec="${exec_dir}/classification/msa/subtom_eigenvolumes_msa"

preali_exec="${exec_dir}/classification/general/subtom_parallel_prealign"

par_xmatrix_exec="${exec_dir}/classification/msa/subtom_parallel_xmatrix_msa"

xmatrix_exec="${exec_dir}/classification/msa/subtom_join_xmatrix"

sum_exec="${exec_dir}/classification/general/subtom_parallel_sums_cls"

avg_exec="${exec_dir}/classification/general/subtom_weighted_average_cls"

motl_dump_exec="${exec_dir}/MOTL/motl_dump"
```

(continues on next page)

(continued from previous page)

```
mem_free="1G"

mem_max="64G"

job_name="subTOM"

array_max="1000"

max_jobs="4000"

run_local="0"

skip_local_copy="1"

iteration="1"

num_xmatrix_prealign_batch="1"

num_xmatrix_batch="1"

num_eig_coeff_prealign_batch="1"

num_eig_coeff_batch="1"

num_avg_batch="1"

high_pass_fp="1"

high_pass_sigma="2"

low_pass_fp="12"

low_pass_sigma="3"

nfold="1"

xmatrix_prealign=0

xmatrix_all_motl_fn_prefix="combinedmotl/allmotl"

xmatrix_fn_prefix="class/xmatrix_msa"

ptcl_fn_prefix="subtomograms/subtomo"

mask_fn="none"

num_eigs='40'

eigs_iterations='default'

eigs_tolerance='default'
```

(continues on next page)

(continued from previous page)

```
eig_vec_fn_prefix="class/eigvec_msa"

eig_val_fn_prefix="class/eigval_msa"

eig_vol_fn_prefix="class/eigvol_msa"

apply_weight="0"

tomo_row="7"

eig_coeff_prealign="0"

eig_coeff_all_motl_fn_prefix="combinedmotl/allmotl"

eig_coeff_fn_prefix="class/eigcoeff_msa"

weight_fn_prefix="otherinputs/ampspec"

cluster_type="kmeans"

eig_idxs="all"

num_classes=2

cluster_all_motl_fn_prefix="class/allmotl_msa"

ref_fn_prefix="class/ref_msa"

weight_sum_fn_prefix="class/wei_msa"
```

53.3 subtom_eigenvolumes_msa

Computes Eigendecomposition of X-Matrix covariance and projects data on Eigenvectors.

```
subtom_eigenvolumes_msa(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'eig_vec_fn_prefix', eig_vec_fn_prefix ('class/eigvec_msa'),
    'eig_val_fn_prefix', eig_val_fn_prefix ('class/eigval_msa'),
    'xmatrix_fn_prefix', xmatrix_fn_prefix ('class/xmatrix_msa'),
    'eig_vol_fn_prefix', eig_vol_fn_prefix ('class/eigvol_msa'),
    'mask_fn', mask_fn ('none'),
    'iteration', iteration (1),
    'num_eigs', num_eigs (40),
    'eigs_iterations', eigs_iterations ('default'),
    'eigs_tolerance', eig_tolerance ('default'))
```

Calculates `num_eigs` weighted projections of particles onto the same number of determined Eigenvectors, by means of a previously calculated X-matrix, named as given by `xmatrix_fn_prefix` and `iteration` to produce Eigenvolumes which can then be used to determine which vectors can best influence classification. The Eigenvectors and Eigenvalues

are also written out as specified by `eig_vec_fn_prefix`, `eig_val_fn_prefix`, and `iteration` The Eigenvolumes are also masked by the file specified by `mask_fn`. The output weighted Eigenvolume will be written out as described by `eig_vol_fn_prefix`, `iteration` and #, where the # is the particular Eigenvolume being written out. Two options `eigs_iterations` and `eigs_tolerance` are also available to tune how eigs is run. If the string 'default' is given for either the default values in eigs will be used.

53.3.1 Example

```
subtom_eigenvolumes_msa(...
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ptcl_fn_prefix', 'subtomograms/subtomo', ...
    'eig_vec_fn_prefix', 'class/eigvec', ...
    'eig_val_fn_prefix', 'class/eigval', ...
    'xmatrix_fn_prefix', 'class/xmatrix', ...
    'eig_vol_fn_prefix', 'class/eigvol', ...
    'mask_fn', 'class/class_mask.em', ...
    'iteration', 1, ...
    'num_eigs', 40, ...
    'eigs_iterations', 'default', ...
    'eigs_tolerance', 'default')
```

53.3.2 See Also

- *subtom_join_eigencoeffs_msa*
- *subtom_join_xmatrix*
- *subtom_parallel_eigencoeffs_msa*
- *subtom_parallel_xmatrix_msa*

53.4 subtom_join_eigencoeffs_msa

Combines Eigencoefficient batches into final matrix.

```
subtom_join_eigencoeffs_msa(
    'eig_coeff_fn_prefix', eig_coeff_fn_prefix ('class/eigcoeff_msa'),
    'iteration', iteration (1),
    'num_coeff_batch', num_coeff_batch (1))
```

Looks for partial chunks of the low-rank approximation coefficients of projected particles with the file name given by `eig_coeff_fn_prefix`, `iteration` and # where # is from 1 to `num_coeff_batch`, and combines them into a final matrix of coefficients written out as described by `eig_coeff_fn_prefix`, and `iteration`.

53.4.1 Example

```
subtom_join_eigencoeffs_msa(...
    'eig_coeff_fn_prefix', 'class/eigcoeff', ...
    'iteration', 1, ...
    'num_coeff_batch', 100)
```

53.4.2 See Also

- *subtom_eigenvolumes_msa*
- *subtom_join_xmatrix*
- *subtom_parallel_eigencoeffs_msa*
- *subtom_parallel_xmatrix_msa*

53.5 subtom_join_xmatrix

Combines chunks of X-Matrix into the final matrix.

```
subtom_join_xmatrix(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'xmatrix_fn_prefix', xmatrix_fn_prefix ('class/xmatrix_msa'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'mask_fn', mask_fn ('none'),
    'iteration', iteration (1),
    'num_xmatrix_batch', num_xmatrix_batch (1))
```

Looks for partial chunks of the X-matrix with the file name given by `xmatrix_fn_prefix`, `iteration`, and # where # is from 1 to `num_xmatrix_batch`, and combines them into a final matrix of coefficients written out as described by `xmatrix_fn_prefix` and `iteration`.

53.5.1 Example

```
subtom_join_xmatrix(
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'xmatrix_fn_prefix', 'class/xmatrix', ...
    'ptcl_fn_prefix', 'subtomograms/subtomo', ...
    'mask_fn', 'otherinputs/classification_mask.em', ...
    'iteration', 1, ...
    'num_xmatrix_batch', 100);
```

53.5.2 See Also

- *subtom_eigenvolumes_msa*
- *subtom_join_eigencoeffs_msa*
- *subtom_parallel_eigencoeffs_msa*
- *subtom_parallel_xmatrix_msa*

53.6 subtom_parallel_eigencoeffs_msa

Computes particle Eigencoefficients

```
subtom_parallel_eigencoeffs_msa(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'xmatrix_fn_prefix', xmatrix_fn_prefix ('class/xmatrix_msa'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'eig_coeff_fn_prefix', eig_coeff_fn_prefix ('class/eigcoeff_msa'),
    'eig_val_fn_prefix', eig_val_fn_prefix ('class/eigval_msa'),
    'eig_vol_fn_prefix', eig_vol_fn_prefix ('class/eigvol_msa'),
    'weight_fn_prefix', weight_fn_prefix ('otherinputs/ampspec'),
    'mask_fn', mask_fn ('none'),
    'high_pass_fp', high_pass_fp (0),
    'high_pass_sigma', high_pass_sigma (0),
    'low_pass_fp', low_pass_fp (0),
    'low_pass_sigma', low_pass_sigma (0),
    'nfold', nfold (1),
    'apply_weight', apply_weight (0),
    'tomo_row', tomo_row (7),
    'iteration', iteration (1),
    'prealigned', prealigned (0),
    'num_coeff_batch', num_coeff_batch (1),
    'process_idx', process_idx (1))
```

Takes a batch subset of particles described by `all_motl_fn_prefix` with filenames given by `ptcl_fn_prefix`, band-pass filters them as described by `high_pass_fp`, `high_pass_sigma`, `low_pass_fp`, and `low_pass_sigma`, and projects them onto the Eigenvolumes specified by `eig_vol_fn_prefix`. This determines a set of coefficients describing a low-rank approximation of the data. A subset of this coefficient matrix is written out based on `eig_coeff_fn_prefix` and `process_idx`, with there being `num_coeff_batch` batches in total.

If `apply_weight` is set to 1 the Eigenvolumes will be reweighted using the correct weight of each particle as described by `weight_fn_prefix` and `tomo_row`, then each particle will be read and projected in a loop. If `prealigned` is set to 1, then it is understood that the particles have been prealigned beforehand and the alignment of the particles can be skipped to save time. `mask_fn` describes the mask used throughout classification and 'none' describes a default spherical mask.

53.6.1 Example

```
subtom_parallel_eigencoeffs_msa(
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'xmatrix_fn_prefix', 'class/xmatrix', ...
    'ptcl_fn_prefix', 'subtomograms/subtomo_ali', ...
    'eig_coeff_fn_prefix', 'class/eigcoeff', ...
    'eig_val_fn_prefix', 'class/eigval', ...
    'eig_vol_fn_prefix', 'class/eigvol', ...
    'weight_fn_prefix', 'otherinputs/ampspec', ...
    'mask_fn', 'otherinputs/classification_mask.em', ...
    'high_pass_fp', 1, ...
    'high_pass_sigma', 2, ...
    'low_pass_fp', 15, ...
    'low_pass_sigma', 3, ...
    'nfold', 1, ...
    'apply_weight', 1, ...
    'tomo_row', 7, ...
    'iteration', 1, ...
    'prealigned', 1, ...
    'num_coeff_batch', 100, ...
    'process_idx', 1)
```

53.6.2 See Also

- *subtom_eigenvolumes_msa*
- *subtom_join_eigencoeffs_msa*
- *subtom_join_xmatrix*
- *subtom_parallel_xmatrix_msa*

53.7 subtom_parallel_xmatrix_msa

Calculates chunks of the X-matrix for processing.

```
subtom_parallel_xmatrix_msa(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'xmatrix_fn_prefix', xmatrix_fn_prefix ('class/xmatrix_msa'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'mask_fn', mask_fn ('none'),
    'high_pass_fp', high_pass_fp (0),
    'high_pass_sigma', high_pass_sigma (0),
    'low_pass_fp', low_pass_fp (0),
    'low_pass_sigma', low_pass_sigma (0),
    'nfold', nfold (1),
    'iteration', iteration (1),
    'prealigned', prealigned (0),
    'num_xmatrix_batch', num_xmatrix_batch (1),
    'process_idx', process_idx (1))
```

Aligns a subset of particles using the rotations and shifts given by `all_motl_fn_prefix` and `iteration`, band-pass filters the particle as described by `high_pass_fp`, `high_pass_sigma`, `low_pass_fp`, and `low_pass_sigma`, optionally symmetrizes the particle with C-fold symmetry `nfold`, and then places these voxels as a 1-D row vector in a data sub-matrix which is historically known as the X-matrix (See Borland, Van Heel 1990 J. Opt. Soc. Am. A). This X-matrix can then be used to speed up the calculation of Eigenvolumes and Eigencoefficients used for classification. The subset of particles compared is specified by the number of particles in the motive list and the number of requested batches specified by `num_xmatrix_batch`, with the specific subset deteremined by `process_idx`. The X-matrix chunk will be written out as specified by `xmatrix_fn_prefix`, `iteration` and `process_idx`. The location of the particles is specified by `ptcl_fn_prefix`. If `prealigned` evaluates to true as a boolean then the particles are assumed to be prealigned, which should increase speed of computation of CC-Matrix calculations. Particles in the X-matrix will be masked by the file given by `mask_fn`. If the string 'none' is used in place of `mask_fn`, a default spherical mask is applied. This mask should be a binary mask and only voxels within the mask are placed into the X-matrix which can greatly speed up computations.

53.7.1 Example

```
subtom_parallel_xmatrix_msa(
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'xmatrix_fn_prefix', 'class/xmatrix', ...
    'ptcl_fn_prefix', 'subtomograms/subtomo_ali', ...
    'mask_fn', 'combinedmotl/classification_mask.em', ...
    'high_pass_fp', 1, ...
    'high_pass_sigma', 2, ...
    'low_pass_fp', 15, ...
    'low_pass_sigma', 3, ...
    'nfold', 1, ...
    'iteration', 1, ...
    'prealigned', 1, ...
    'num_xmatrix_batch', 100, ...
    'process_idx', 1)
```

53.7.2 See Also

- *subtom_eigenvolumes_msa*
- *subtom_join_eigencoeffs_msa*
- *subtom_join_xmatrix*
- *subtom_parallel_eigencoeffs_msa*

53.8 Indices and tables

- genindex
- modindex
- search

CHAPTER

FIFTYFOUR

SUBTOM: MULTIREFERENCE

54.1 Multireference Classification

In multireference classification the full set of particles are compared to a predefined number of reference volumes. Each experimental particle is aligned with respect to all references and is assigned to one of them based on the constrained cross-correlation coefficient as a similarity measure. Averaging over the subsets determined serves to calculate new, improved references for further iterations. The user can then iterate this procedure and stop when no more migration of particles between subsets is observed and the averages within each subset do not change any more.

Within subTOM a function `subtom_rand_class_motl` serves to initialize random classes and initial references if the user does not alread have them, and then the subTOM functions `subtom_scan_angles_exact_multiref`, and `subtom_compare_motls_multiref` have been modified to fit the new modus of multireference classification and alignment.

54.2 subtom_multiref_alignment

The main multireference pipeline process script of subTOM. Iteratively aligns and averages a collection of subvolumes, against multiple references.

This subtomogram alignment script uses five MATLAB compiled scripts below:

- *subtom_scan_angles_exact_multiref*
- *subtom_cat_motls*
- *subtom_parallel_sums_cls*
- *subtom_weighted_average_cls*
- *subtom_compare_motls_multiref*

54.2.1 Options

Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

local_dir Absolute path to the folder on a group share, if the scratch directory is cleaned and deleted regularly this can set a local directory to which the important results will be copied. If this is not needed it can be skipped with the option skip_local_copy below.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

Variables

align_exec Alignment executable

cat_exec Concatenate MOTLs executable

sum_exec Parallel Summing executable

avg_exec Weighted Averaging executable

compare_exec Compare MOTLs executable

motl_dump_exec MOTL dump executable

Memory Options

mem_free_ali The amount of memory the job requires for alignment. This variable determines whether a number of CPUs will be requested to be dedicated for each job. At 24G, one half of the CPUs on a node will be dedicated for each of the processes (12 CPUs). At 48G, all of the CPUs on the node will be dedicated for each of the processes (24 CPUs).

mem_max_ali The upper bound on the amount of memory the alignment job is allowed to use. If any of the processes request or require more memory than this, the queue will kill the process. This is more of an option for safety of the cluster to prevent the user from crashing the cluster requesting too much memory.

mem_free_avg The amount of memory the job requires for averaging.

mem_max_avg The upper bound on the amount of memory the averaging job is allowed to use.

Other Cluster Options

job_name The job name prefix that will be used for the cluster submission scripts, log files, and error logs for the processing. Be careful that this name is unique because previous submission scripts, logs, and error logs with the same job name prefix will be overwritten in the case of a name collision.

array_max The maximum number of jobs per cluster submission script. Cluster submission scripts work using the array feature common to queuing systems, and this value is the maximum array size used in a script. If the user requests more batches of processing than this value, then the submission scripts will be split into files of up to array_max jobs.

max_jobs The maximum number of jobs for alignment. If the number of batches / exceeds this value the script will immediately quit.

run_local If the user wants to skip the cluster and run the job locally, this value should be set to 1.

skip_local_copy Set this option to 1 to skip the copying of data to local_dir.

Subtomogram Alignment Workflow Options

Parallelization Options

start_iteration The index of the reference to start from : input will be ref_fn_prefix_start_iteration.em and all_motl_fn_prefix_start_iteration.em (define as integer e.g. start_iteration=3)

More on iterations since they're confusing and it is slightly different here than from previous iterations.

The start_iteration is the beginning for the iteration variable used throughout this script. Iteration refers to iteration that is used for subtomogram alignment. So if start_iteration is 1, then subtomogram alignment will work using allmotl_1.em and ref_1.em. The output from alignment will be particle motls for the next iteration. This in the script is avg_iteration variable. The particle motls will be joined to form allmotl_2.em and then the parallel averaging will form ref_2.em and then the loop is done and iteration will become 2 and avg_iteration will become 3.

iterations Number iterations (big loop) to run: final output will be ref_fn_prefix_start_iteration+iterations.em and all_motl_fn_prefix_start_iteration+iterations.em

num_ali_batch The number of batches to split the parallel subtomogram alignment job into.

num_avg_batch The number of batches to split the parallel subtomogram averaging job into.

File Options

all_motl_fn_prefix Relative path and name of the concatenated motivelist of all particles (e.g. allmotl_iter.em , the variable will be written as a string e.g. all_motl_fn_prefix='sub-directory/allmotl')

ref_fn_prefix Relative path and name of the reference volumes (e.g. ref_iter.em , the variable will be written as a string e.g. ref_fn_prefix='sub-directory/ref')

ptcl_fn_prefix Relative path and name of the subtomograms (e.g. part_n.em , the variable will be written as a string e.g. ptcl_fn_prefix='sub-directory/part')

align_mask_fn Relative path and name of the alignment mask Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

cc_mask_fn Relative path and name of the cross-correlation mask this defines the maximum shifts in each direction Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

weight_fn_prefix Relative path and name of the weight file.

weight_sum_fn_prefix Relative path and name of the partial weight files.

Alignment and Averaging Options

tomo_row Which row in the motl file contains the correct tomogram number. Usually row 5 and 7 both correspond to the correct value and can be used interchangeably, but there are instances when 5 contains a sequential ordered value starting from 1, while 7 contains the correct corresponding tomogram.

apply_weight Apply weight to subtomograms (1=yes, 0=no).

apply_mask Apply mask to subtomograms (1=yes, 0=no).

keep_class If you want particles to be constrained to their existing class set this to 1, otherwise particles will change to the class of which they align best against.

psi_angle_step Angular increment in degrees, applied during the cone-search, i.e. psi and theta (define as real e.g. psi_angle_step=3). Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

psi_angle_shells Number of angular iterations, applied to psi and theta (define as integer e.g. psi_angle_shells=4). Note that in terms of cones this is twice the number of cones sampled. Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

phi_angle_step Angular increment for phi in degrees, (define as real e.g. phi_angle_step=3). Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

phi_angle_shells Number of angular iterations for phi, (define as integer e.g. phi_angle_shells=6). Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

high_pass_fp High pass filter cutoff (in transform units (pixels): calculate as (box_size * pixelsize) / (resolution_real) (define as integer). Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

high_pass_sigma High pass filter falloff sigma (in transform units (pixels): describes a Gaussian sigma for the falloff of the high-pass filter past the cutoff above. Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

low_pass_fp Low pass filter (in transform units (pixels): calculate as (box_size * pixelsize) / (resolution_real) (define as integer). Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

low_pass_sigma Low pass filter falloff sigma (in transform units (pixels): describes a Gaussian sigma for the falloff of the low-pass filter past the cutoff above. Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

nfold Symmetry, if no symmetry nfold=1 (define as integer e.g. nfold=3). Leave the parentheses and if the number of values is less than the number of iterations the last value will be repeated to the correct length.

threshold Threshold for cross correlation coefficient. Only particles with ccc_new > threshold will be added to new average (define as real e.g. threshold=0.5). These particles will still be aligned at each iteration.

54.2.2 Example

```
scratch_dir="${PWD}"

local_dir=""

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

align_exec="${exec_dir}/classification/multiref/subtom_scan_angles_exact_multiref"

cat_exec="${exec_dir}/MOTL/subtom_cat_motls"

sum_exec="${exec_dir}/classification/general/subtom_parallel_sums_cls"

avg_exec="${exec_dir}/classification/general/subtom_weighted_average_cls"
```

(continues on next page)

(continued from previous page)

```
compare_exec="${exec_dir}/classification/multiref/subtom_compare_motls_multiref"

motl_dump_exec="${exec_dir}/MOTL/motl_dump"

mem_free_ali=1G

mem_max_ali=64G

mem_free_avg=1G

mem_max_avg=64G

job_name=subTOM

array_max=1000

max_jobs=4000

run_local=0

skip_local_copy=1

start_iteration=1

iterations=3

num_ali_batch=1

num_avg_batch=1

all_motl_fn_prefix="combinedmotl/allmotl"

ref_fn_prefix="ref/ref"

ptcl_fn_prefix="subtomograms/subtomo"

align_mask_fn=("otherinputs/align_mask_1.em" \
               "otherinputs/align_mask_2.em" \
               "otherinputs/align_mask_3.em")

cc_mask_fn=("otherinputs/cc_mask_r10.em" \
            "otherinputs/cc_mask_r05.em")

weight_fn_prefix="otherinputs/ampspec"

weight_sum_fn_prefix="otherinputs/wei"

tomo_row=7

apply_weight=0

apply_mask=1
```

(continues on next page)

(continued from previous page)

```
keep_class=0

psi_angle_step=(10 5 2.5)

psi_angle_shells=(4)

phi_angle_step=(20 5)

phi_angle_shells=(6)

high_pass_fp=(1)

high_pass_sigma=(2)

low_pass_fp=(12 15 18)

low_pass_sigma=(3)

nfold=(1 6)

threshold=-1
```

54.3 subtom_multiref_initialize

A startup processing script for multireference classification in subTOM. Splits a given motive list into random equal classes and then generates the average of each class.

This subtomogram averaging script uses three MATLAB compiled scripts below:

- *subtom_rand_class_motl*
- *subtom_parallel_sums_cls*
- *subtom_weighted_average_cls*

54.3.1 Options

Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

local_dir Absolute path to the folder on a group share, if the scratch directory is cleaned and deleted regularly this can set a local directory to which the important results will be copied. If this is not needed it can be skipped with the option skip_local_copy below.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

Variables

sum_exec Parallel Summing executable

avg_exec Weighted Averaging executable

rand_exec Randomize Motive List executable

Memory Options

mem_free The amount of memory the job requires. This variable determines whether a number of CPUs will be requested to be dedicated for each job. At 24G, one half of the CPUs on a node will be dedicated for each of the processes (12 CPUs). At 48G, all of the CPUs on the node will be dedicated for each of the processes (24 CPUs).

mem_max The upper bound on the amount of memory the job is allowed to use. If any of the processes request or require more memory than this, the queue will kill the process. This is more of an option for safety of the cluster to prevent the user from crashing the cluster requesting too much memory.

Other Cluster Options

job_name The job name prefix that will be used for the cluster submission scripts, log files, and error logs for the processing. Be careful that this name is unique because previous submission scripts, logs, and error logs with the same job name prefix will be overwritten in the case of a name collision.

array_max The maximum number of jobs per cluster submission script. Cluster submission scripts work using the array feature common to queuing systems, and this value is the maximum array size used in a script. If the user requests more batches of processing than this value, then the submission scripts will be split into files of up to array_max jobs.

max_jobs The maximum number of jobs for alignment. If the number of batches / exceeds this value the script will immediately quit.

run_local If the user wants to skip the cluster and run the job locally, this value should be set to 1.

skip_local_copy Set this option to 1 to skip the copying of data to local_dir.

Subtomogram Averaging Workflow Options

Parallelization Options

iteration The index of the reference to generate : input will be all_motl_fn_prefix_iteration.em (define as integer e.g. iteration=1)

num_avg_batch The number of batches to split the parallel subtomogram averaging job into.

File Options

all_motl_fn_prefix Relative path and name of the concatenated motivelist of all particles (e.g. allmotl_iter.em , the variable will be written as a string e.g. all_motl_fn_prefix='sub-directory/allmotl')

ref_fn_prefix Relative path and name of the reference volumes (e.g. ref_iter.em , the variable will be written as a string e.g. ref_fn_prefix='sub-directory/ref')

ptcl_fn_prefix Relative path and name of the subtomograms (e.g. part_n.em , the variable will be written as a string e.g. ptcl_fn_prefix='sub-directory/part')

weight_fn_prefix Relative path and name of the weight file.

weight_sum_fn_prefix Relative path and name of the partial weight files.

Averaging Options

tomo_row Which row in the motl file contains the correct tomogram number. Usually row 5 and 7 both correspond to the correct value and can be used interchangeably, but there are instances when 5 contains a sequential ordered value starting from 1, while 7 contains the correct corresponding tomogram.

num_classes The number of classes to split the initial motive list into. The classes will be assigned randomly evenly within the valid particles, (non-negative class values excluding class 2), with the class number starting at 3 to not interfere with the classes 1 and 2 which are reserved for AV3's thresholding process.

54.3.2 Example

```
scratch_dir="${PWD}"

local_dir=""

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

sum_exec="${exec_dir}/classification/general/subtom_parallel_sums_cls"

avg_exec="${exec_dir}/classification/general/subtom_weighted_average_cls"

rand_exec="${exec_dir}/classification/multiref/subtom_rand_class_motl"

mem_free=1G

mem_max=64G

job_name=subTOM

array_max=1000

max_jobs=4000

run_local=0

skip_local_copy=1
```

(continues on next page)

(continued from previous page)

```
iteration=1

num_avg_batch=1

all_motl_fn_prefix="combinedmotl/allmotl"

ref_fn_prefix="ref/ref"

ptcl_fn_prefix="subtomograms/subtomo"

weight_fn_prefix="otherinputs/ampspec"

weight_sum_fn_prefix="otherinputs/wei"

tomo_row=7

num_classes=2
```

54.4 subtom_multiref_rand_class_motl

Randomizes a given number of classes in a motive list.

This MOTL manipulation script uses one MATLAB compiled scripts below:

- *subtom_rand_class_motl*

54.4.1 Options

Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

Variables

rand_exec Randomize Motive List executable.

File Options

input_motl_fn Relative path and name of the input motivelist to be randomized in class.

output_motl_fn Relative path and name of the output motivelist.

Randomize MOTL Options

num_classes The number of classes to split the initial motive list into. The classes will be assigned randomly evenly within the valid particles, (non-negative class values excluding class 2), with the class number starting at 3 to not interfere with the classes 1 and 2 which are reserved for AV3's thresholding process.

54.4.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

rand_exec="${exec_dir}/classification/multiref/subtom_rand_class_motl

input_motl_fn="combinedmotl/allmotl_1.em"

output_motl_fn="combinedmotl/allmotl_multiref_1.em"

num_classes=2
```

54.5 subtom_compare_motls_multiref

Compares orientations, shifts and class changes between two MOTLs.

```
subtom_compare_motls_multiref(
    'motl_1_fn', motl_1_fn (''),
    'motl_2_fn', motl_2_fn (''),
    'write_diffs', write_diffs (0),
    'output_diffs_fn', output_diffs_fn (''))
```

Takes the motls given by `motl_1_fn` and `motl_2_fn` and calculates the differences for both the orientations and coordinates between corresponding particles in each motive list. For multireference alignment the number of particles that have changed class is also determined. If `write_diffs` evaluates to true as a boolean, then also a CSV file with the differences in coordinates and orientations to `diffs_output_fn`.

54.5.1 Example

```
subtom_compare_motls_multiref(...
    'motl_1_fn', 'combinedmotl/allmotl_1.em', ...
    'motl_2_fn', 'combinedmotl/allmotl_2.em', ...
    'write_diffs', 1, ...
    'output_diffs_fn', 'combinedmotl/allmotl_1_2_diff.csv')
```

54.5.2 See Also

- *subtom_rand_class_motl*
- *subtom_scan_angles_exact_multiref*

54.6 subtom_rand_class_motl

Randomizes a given number of classes in a motive list.

```
subtom_rand_class_motl(
    'input_motl_fn', input_motl_fn (''),
    'output_motl_fn', output_motl_fn (''),
    'num_classes', num_classes ('2'))
```

Takes the motive list given by `input_motl_fn`, and splits it into `num_classes` even classes using the 20th row of the motive list, and then writes the transformed motive list out as `output_motl_fn`. The values that go into the 20th row start at 3 and particles that initially have negative or the value 2 in the 20th row are ignored as described in AV3 documentation for the behavior of class numbers.

54.6.1 Example

```
subtom_rand_class_motl(...
    'input_motl_fn', 'combinedmotl/allmotl_1.em', ...
    'output_motl_fn', 'combinedmotl/allmotl_multiref_1.em', ...
    'num_classes', '2')
```

54.6.2 See Also

- *subtom_compare_motls_multiref*
- *subtom_scan_angles_exact_multiref*

54.7 subtom_scan_angles_exact_multiref

Align a particle batch over local search angles.

```
subtom_scan_angles_exact_multiref(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ref_fn_prefix', ref_fn_prefix ('ref/ref'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'weight_fn_prefix', weight_fn_prefix ('otherinputs/ampspec'),
    'align_mask_fn', align_mask_fn ('none'),
    'cc_mask_fn', cc_mask_fn ('noshift'),
    'apply_weight', apply_weight (0),
    'apply_mask', apply_mask (0),
    'keep_class', keep_class (0),
    'psi_angle_step', psi_angle_step (0),
    'psi_angle_shells', psi_angle_shells (0),
    'phi_angle_step', phi_angle_step (0),
    'phi_angle_shells', phi_angle_shells (0),
    'high_pass_fp', high_pass_fp (0),
    'high_pass_sigma', high_pass_sigma (0),
    'low_pass_fp', low_pass_fp (0),
    'low_pass_sigma', low_pass_sigma (0),
    'nfold', nfold (1),
    'threshold', threshold (-1),
    'iteration', iteration (1),
    'tomo_row', tomo_row (7),
    'num_ali_batch', num_ali_batch (1),
    'process_idx', process_idx (1))
```

Aligns a batch of particles from the collective motive list with the name format `all_motl_fn_prefix` _#.em where # is the number `iteration`. The motive list is split into `num_ali_batch` chunks and the specific chunk to process is specified by `process_idx` . A motive list for the best determined alignment parameters is written out for each batch with the name format `ptct_motl_fn_prefix` _#_#.em where the first # is `iteration` + 1 and the second # is the number `process_idx`.

Particles, with the name format `ptcl_fn_prefx` _#.em where # is the subtomogram ID, are aligned against the reference with the name format `ref_fn_prefix` _#.em where # is `iteration`. Before the comparison is made a number of alterations are made to both the particle and reference:

- If `nfold` is greater than 1 then C#-symmetry is applied along the Z-axis to the reference where # is `nfold`.
- The reference is masked in real space with the mask `align_mask_fn`, and if `apply_mask` evaluates to true as a boolean, then this mask is also applied to the particle. A sphere mask is applied to the particle to reduces the artifacts caused by the box-edges on the comparison. This sphere has a diameter that is 80% the box size and falls of with a sigma that is 15% half the box size.
  - The mask is rotated and shifted with the currently existing alignment parameters for the particle as to best center the mask on the particle density.
  - `apply_mask` can help alignment and suppress alignment to other features when the particle is well-centered or already reasonably well aligned, but if this is not the case there is the risk that a tight alignment will cutoff parts of the particle.
- Both the particle and the reference are bandpass filtered in the Fourier domain defined by `high_pass_fp`, `high_pass_sigma`, `low_pass_fp`, and `low_pass_sigma` which are all in the units of Fourier pixels.
- A Fourier weight volume with the name format `weight_fn_prefix` _#.em where # corresponds to the tomogram

from which the particle came from, which is found from the field `tomo_row` in the motive list, is applied to the reference in the Fourier domain, after the reference has been rotated with the currently existing alignment parameters. If `apply_weight` evaluates to true as a boolean, then this weight is also applied to the particle with no rotation. This Fourier weight is designed to compensate for the missing wedge.

  - If a binary wedge is used, then it is reasonable to apply the weight to the particle, however, for more complicated weights, like the average amplitude spectrum, it should not be done.

The local rotations searched during alignment are deteremined by the four parameters `psi_angle_step`, `psi_angle_shells`, `phi_angle_step`, and `phi_angle_shells`. They describe a search where the currently existing alignment parameters for azimuth and zenith are used to define a "pole" to search about in the ceiling of half `psi_angle_shells` cones. The change in zenith between each cone is `psi_angle_step` and the azimuth around the cone is close to the same angle but is adjusted slightly to account for bias near the pole. The final spin angle of the search is done with a change in spin of `phi_angle_step` in `phi_angle_shells` steps. The spin is applied in both clockwise and counter-clockwise fashion.

- The angles phi, and psi here are flipped in their sense of every other package for EM image processing, which is absolutely infuriating and confusing, but maintained for historical reasons, however most descriptions use the words azimuth, zenith, and spin to avoid ambiguity.

Finally after the constrained cross-correlation function is calculated it is masked with `cc_mask_fn` to limit the shifts to inside this volume, and a peak is found and it's location is determined to sub-pixel accuracy using interpolation. The rotations and shifts that gives the highest cross-correlation coefficient are then chosen as the new alignments parameters. Particles with a coefficient lower than `threshold` are placed into class 2 and ignored in later processing, and particles with class `iclass` are the only particles processed.

- If `iclass` is 0 all particles will be considered, and particles above `threshold` will be assigned to iclass of 1 and particles below `threshold` will be assigned to iclass of 2. If `iclass` is 1 or 2 then particles with iclass 0 will be skipped, particles of iclass 1 and 2 will be aligned and particles with scores above `threshold` will be assigned to iclass 1 and particles with scores below `threshold` will be assigned to iclass 2. `iclass` of 2 does not make much sense but is set this way in case of user mistakes or misunderstandings. If `iclass` is greater than 2 then particles with iclass of 1, 2, and `iclass` will be aligned, and particles with a score above `threshold` will maintain their iclass if it is 1 or `iclass`, and particles with a previous iclass of 2 will be upgraded to an iclass of 1. Particles with a score below `threshold` will be assigned to iclass 2.
- The class number is stored in the 20th field of the motive list.

54.7.1 Example

```
subtom_scan_angles_multiref(...
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ref_fn_prefix', 'ref/ref', ...
    'ptcl_fn_prefix', 'subtomograms/subtomo', ...
    'weight_fn_prefix', 'otherinputs/ampspec', ...
    'align_mask_fn', 'otherinputs/align_mask.em', ...
    'cc_mask_fn', 'otherinputs/cc_mask.em', ...
    'apply_weight', 0, ...
    'apply_mask', 1, ...
    'keep_class', 0, ...
    'psi_angle_step', 6, ...
    'psi_angle_shells', 8, ...
    'phi_angle_step', 6, ...
    'phi_angle_shells', 8, ...
    'high_pass_fp', 1, ...
    'high_pass_sigma', 2, ...
```

(continues on next page)

(continued from previous page)

```
    'low_pass_fp', 12, ...
    'low_pass_sigma', 3, ...
    'nfold', 6, ...
    'threshold', 0, ...
    'iteration', 1, ...
    'tomo_row', 7, ...
    'num_ali_batch', 1, ...
    'process_idx', 1)
```

54.7.2 See Also

- *subtom_compare_motls_multiref*
- *subtom_rand_class_motl*

54.8 Indices and tables

- genindex
- modindex
- search

CHAPTER

FIFTYFIVE

SUBTOM: PRINCIPAL COMPONENT ANALYSIS

55.1 Principal Component Analysis Classification

In principal component analysis (PCA) classification the full set of particles are simplified into a new lower-dimensional representation by means of Eigen or Singular Value decomposition methods. Particles projected onto these most variable basis-vectors then can be clustered using a variety of methods.

Within subTOM particles are first compared using Constrained Cross-Correlation taking into account the missing wedge. The pairs used in comparison are pre-calculated with the function `subtom_prepare_ccmatrix`. To speed up calculation particles can be pre-aligned using the function `subtom_parallel_prealign`.

The comparisons are calculated in parallel batches with `subtom_parallel_ccmatrix` and the results are combined with `subtom_join_ccmatrix`. The Cross-Correlation matrix is then decomposed into a user-given number of basis vectors using either Eigenvalue decomposition with `subtom_eigs` or Singular Value decomposition with `subtom_svds`, which the basis vectors and their respective weights.

The particles that were compared against are then projected onto these vectors by first constructing a matrix of the aligned data with `subtom_parallel_xmatrix_pca` and then projected in parallel batches with `subtom_parallel_eigenvolumes` and joined with `subtom_join_eigenvolumes`. These volumes are then used to determine the low-rank approximation coefficients in volume space for clustering. A larger particle superset can be projected onto the volumes to speed up classification of large datasets. Coefficients are also calculated in parallel in batches with `subtom_parallel_eigcoeffs_pca` and joined with `subtom_join_eigencoeffs_pca`.

Finally using a user-selected subset of the determined coefficients, the data is clustered either by Hierarchical Ascendant Clustering using a Ward distance criterion, K-Means clustering, or a Gaussian Mixture model with the function `subtom_cluster`. This clustering is then used to generate the final class averages.

55.2 subtom_pca

The main PCA pipeline process script of subTOM.

This subtomogram classification script uses fourteen MATLAB compiled scripts below:

- *subtom_parallel_prealign*
- *subtom_prepare_ccmatrix*
- *subtom_parallel_ccmatrix*
- *subtom_join_ccmatrix*
- *subtom_eigs*
- *subtom_svds*

- *subtom_parallel_xmatrix_pca*
- *subtom_parallel_eigenvolumes*
- *subtom_join_eigenvolumes*
- *subtom_parallel_eigencoeffs_pca*
- *subtom_join_eigencoeffs_pca*
- *subtom_cluster*
- *subtom_parallel_sums_cls*
- *subtom_weighted_average_cls*

55.2.1 Options

Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

local_dir Absolute path to the folder on a group share, if the scratch directory is cleaned and deleted regularly this can set a local directory to which the important results will be copied. If this is not needed it can be skipped with the option skip_local_copy below.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

Variables

cluster_exec Cluster executable.

eig_exec Eigendecomposition executable.

pre_ccmatrix_exec Prepare CC-Matrix executable.

par_ccmatrix_exec Parallel CC-Matrix executable.

ccmatrix_exec Final CC-Matrix executable.

par_eigcoeff_exec Parallel Eigencoefficient executable.

eigcoeff_exec Final Eigencoefficient executable.

par_eigvol_exec Parallel Eigenvolume executable.

eigvol_exec Final Eigenvolume executable.

preali_exec Parallel Subtomogram prealign executable.

xmatrix_exec Parallel X-Matrix executable.

svds_exec Singular Value Decomposition executable.

sum_exec Parallel Summing executable

avg_exec Final Averaging executable

motl_dump_exec MOTL dump executable

Memory Options

mem_free The amount of memory the job requires. This variable determines whether a number of CPUs will be requested to be dedicated for each job. At 24G, one half of the CPUs on a node will be dedicated for each of the processes (12 CPUs). At 48G, all of the CPUs on the node will be dedicated for each of the processes (24 CPUs).

mem_max The upper bound on the amount of memory the job is allowed to use. If any of the processes request or require more memory than this, the queue will kill the process. This is more of an option for safety of the cluster to prevent the user from crashing the cluster requesting too much memory.

Other Cluster Options

job_name The job name prefix that will be used for the cluster submission scripts, log files, and error logs for the processing. Be careful that this name is unique because previous submission scripts, logs, and error logs with the same job name prefix will be overwritten in the case of a name collision.

array_max The maximum number of jobs per cluster submission script. Cluster submission scripts work using the array feature common to queuing systems, and this value is the maximum array size used in a script. If the user requests more batches of processing than this value, then the submission scripts will be split into files of up to array_max jobs.

max_jobs The maximum number of jobs for alignment. If the number of batches / exceeds this value the script will immediately quit.

run_local If the user wants to skip the cluster and run the job locally, this value should be set to 1.

skip_local_copy Set this option to 1 to skip the copying of data to local_dir.

Parallelization Options

iteration The index of the references to generate : input will be all_motl_fn_prefix_iteration.em (define as integer e.g. iteration=1)

num_ccmatrix_prealign_batch Number of batches to split the parallel particle prealignment for the CC-Matrix calculation into. If you are not doing prealignment you can ignore this option.

num_ccmatrix_batch Number of batches to split the parallel CC-Matrix calculation job into.

num_xmatrix_batch Number of batches to split the parallel X-Matrix calculation job into. This also determines the number of batches the Eigenvolumes calculation will be split into.

num_eig_coeff_prealign_batch Number of batches to split the parallel particle prealignment for the Eigencoefficients calculations into. If you are not doing prealignment you can ignore this option.

num_eig_coeff_batch Number of batches to split the parallel Eigencoefficient calculation into.

num_avg_batch The number of batches to split the parallel subtomogram averaging job into.

Subtomogram Classification Workflow Options

CC-Matrix Options

high_pass_fp High pass filter cutoff (in transform units (pixels): calculate as (box_size * pixelsize) / (resolution_real) (define as integer).

high_pass_sigma High pass filter falloff sigma (in transform units (pixels): describes a Gaussian sigma for the falloff of the high-pass filter past the cutoff above.

low_pass_fp Low pass filter (in transform units (pixels): calculate as (box_size * pixelsize) / (resolution_real) (define as integer).

low_pass_sigma Low pass filter falloff sigma (in transform units (pixels): describes a Gaussian sigma for the falloff of the low-pass filter past the cutoff above.

nfold Symmetry to apply to each pair of particle and reference in CC-Matrix calculation, if no symmetry ccmatrix_nfold=1 (define as integer e.g. ccmatrix_nfold=3)

tomo_row Which row in the motl file contains the correct tomogram number. Usually row 5 and 7 both correspond to the correct value and can be used interchangeably, but there are instances when 5 contains a sequential ordered value starting from 1, while 7 contains the correct corresponding tomogram.

ccmatrix_prealign If you want to pre-align all of the particles to speed up the CC-Matrix calculation, set the following to 1, otherwise the particles will be aligned during the computation.

CC-Matrix File Options

ccmatrix_all_motl_fn_prefix Relative path and name of the concatenated motivelist of all particles (e.g. allmotl_iter.em , the variable will be written as a string e.g. ccmatrix_all_motl_fn_prefix='sub-directory/allmotl').

ptcl_fn_prefix Relative path and name of the subtomograms (e.g. part_n.em , the variable will be written as a string e.g. ptcl_fn_prefix='sub-directory/part').

mask_fn Relative path and name of the classification mask. This should be a binary mask as correlations are done in real-space, and calculations will only be done using voxels passed by the mask, so smaller masks will run faster. If you want to use the default spherical mask set mask_fn to 'none'.

weight_fn_prefix Relative path and name of the weight file.

ccmatrix_fn_prefix Relative path and name of the CC-Matrix.

Eigendecomposition Options

decomp_type The following determines which type of decomposition to perform. If the following is 'eigs', then traditional Eigenvalue decomposition will be calculated and either the largest magnitude or largest algebraic Eigenvalues will be returned, however in the CC-Matrix calculation the Eigenvalues can be negative which can be problematic in later stages of processing, and so 'svds' can also be given and Singular Value Decomposition will calculated instead.

num_eigs The number of Eigenvectors and Eigenvalues (or Left Singular Vectors and Singular Values) to calculate.

eigs_iterations If using 'eigs' the following allows you to adjust the number of iterations to use in the decomposition. If you want to use the default number of iterations leave this set to 'default'.

eigs_tolerance If using 'eigs' the following allows you to adjust the convergence tolerance of the decomposition calculation. If you want to use the default tolerance leave this set to 'default'.

do_algebraic If using 'eigs' the following allows you to calculate the largest algebraic Eigenvalues, which are guaranteed to be positive but not guaranteed to be the largest in magnitude. This is in contrast to the default behavior of calculating the largest magnitude Eigenvalues that are not guaranteed to be non-negative.

svds_iterations If using 'svds' the following allows you to adjust the number of iterations to use in the decomposition. If you want to use the default number of iterations leave this set to 'default'.

svds_tolerance If using 'svds' the following allows you to adjust the convergence tolerance of the decomposition calculation. If you want to use the default tolerance leave this set to 'default'.

Eigendecomposition File Options

eig_vec_fn_prefix Relative path and name of the Eigenvectors (or Left Singular Vectors).

eig_val_fn_prefix Relative path and name of the Eigenvalues (or Singular Values).

X-Matrix File Options

xmatrix_fn_prefix Relative path and name of the X-Matrix.

Eigenvolumes File Options

eig_vol_fn_prefix Relative path and name of the Eigenvolumes.

Eigencoefficient Options

apply_weight If the following is set to 1, the Eigenvolume (or conjugate-space Eigenvector) will have the particles missing-wedge weight applied to it before the Correlation is calculated.

eig_coeff_prealign If you want to pre-align all of the particles to speed up the Eigencoefficient calculation, set the following to 1, otherwise the particles will be aligned during the computation.

Eigencoefficient File Options

eig_coeff_all_motl_fn_prefix Relative path and name of the concatenated motivelist to project onto the Eigenvolumes (conjugate-space Eigenvectors). This can be a larger motivelist than the one used to calculate the CC-Matrix and Eigenvolumes.

eig_coeff_fn_prefix Relative path and name of the Eigencoefficients.

Clustering Options

cluster_type The following determines which algorithm will be used to cluster the determined Eigencoefficients. The valid options are K-means clustering, 'kmeans', Hierarchical Ascendent Clustering using a Ward Criterion, 'hac', and a Gaussian Mixture Model, 'gaussmix'.

eig_idxs Determines which Eigencoefficients are used to cluster. The format should be a semicolon-separated list that also supports ranges with a dash (-), for example 1-5;7;15-19 would select the first five Eigencoefficients, the seventh and the fifteenth through the nineteenth for classification. If it is left as "all" all coefficients will be used.

num_classes How many classes should the particles be clustered into.

Clustering File Options

cluster_all_motl_fn_prefix Relative path and name of the concatenated motivelist of the output classified particles.

Averaging File Options

ref_fn_prefix Relative path and name prefix of the reference volumes (e.g. ref_iter.em, the variable will be written as a string e.g. ref_fn_prefix='sub-directory/ref')

weight_sum_fn_prefix Relative path and name prefix of the partial weight files.

55.2.2 Example

```
scratch_dir="${PWD}"

local_dir=""

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="XXXINSTALLATION_DIRXXX/bin"

cluster_exec="${exec_dir}/classification/general/subtom_cluster"

eigs_exec="${exec_dir}/classification/pca/subtom_eigs"

pre_ccmatrix_exec="${exec_dir}/classification/pca/subtom_prepare_ccmatrix"

par_ccmatrix_exec="${exec_dir}/classification/pca/subtom_parallel_ccmatrix"

ccmatrix_exec="${exec_dir}/classification/pca/subtom_join_ccmatrix"

par_eigcoeff_exec="${exec_dir}/classification/pca/subtom_parallel_eigencoeffs_pca"

eigcoeff_exec="${exec_dir}/classification/pca/subtom_join_eigencoeffs_pca"

par_eigvol_exec="${exec_dir}/classification/pca/subtom_parallel_eigenvolumes"

eigvol_exec="${exec_dir}/classification/pca/subtom_join_eigenvolumes"

preali_exec="${exec_dir}/classification/general/subtom_parallel_prealign"

xmatrix_exec="${exec_dir}/classification/pca/subtom_parallel_xmatrix_pca"

svds_exec="${exec_dir}/classification/pca/subtom_svds"

sum_exec="${exec_dir}/classification/general/subtom_parallel_sums_cls"

avg_exec="${exec_dir}/classification/general/subtom_weighted_average_cls"

motl_dump_exec="${exec_dir}/MOTL/motl_dump"

mem_free="1G"
```

(continues on next page)

(continued from previous page)

```
mem_max="64G"

job_name="subTOM"

array_max="1000"

max_jobs="4000"

run_local="0"

skip_local_copy="1"

iteration="1"

num_ccmatrix_prealign_batch="1"

num_ccmatrix_batch="1"

num_xmatrix_batch="1"

num_eig_coeff_prealign_batch="1"

num_eig_coeff_batch="1"

num_avg_batch="1"

high_pass_fp="0"

high_pass_sigma="2"

low_pass_fp="0"

low_pass_sigma="3"

nfold="1"

tomo_row="7"

ccmatrix_prealign=0

ccmatrix_all_motl_fn_prefix="combinedmotl/allmotl"

ptcl_fn_prefix="subtomograms/subtomo"

mask_fn="none"

weight_fn_prefix="otherinputs/ampspec"

ccmatrix_fn_prefix="class/ccmatrix_pca"

decomp_type='svds'
```

(continues on next page)

(continued from previous page)

```
num_eigs='40'

eigs_iterations='default'

eigs_tolerance='default'

do_algebraic=0

svds_iterations='default'

svds_tolerance='default'

eig_vec_fn_prefix="class/eigvec_pca"

eig_val_fn_prefix="class/eigval_pca"

xmatrix_fn_prefix="class/xmatrix_pca"

eig_vol_fn_prefix="class/eigvol_pca"

apply_weight="0"

eig_coeff_prealign="0"

eig_coeff_all_motl_fn_prefix="combinedmotl/allmotl"

eig_coeff_fn_prefix="class/eigcoeff_pca"

cluster_type="kmeans"

eig_idxs="all"

num_classes=2

cluster_all_motl_fn_prefix="class/allmotl_pca"

ref_fn_prefix="class/ref_pca"

weight_sum_fn_prefix="class/wei_pca"
```

55.3 subtom_eigs

Uses MATLAB eigs to calculate a subset of Eigenvalue/vectors.

```
subtom_eigs(
    'ccmatrix_fn_prefix', ccmatrix_fn_prefix ('class/ccmatrix_pca'),
    'eig_vec_fn_prefix', eig_vec_fn_prefix ('class/eigvec_pca'),
    'eig_val_fn_prefix', eig_val_fn_prefix ('class/eigval_pca'),
    'iteration', iteration (1),
```

(continues on next page)

(continued from previous page)

```
    'num_eigs', num_eigs (40),
    'eigs_iterations', eigs_iterations ('default'),
    'eigs_tolerance', eig_tolerance ('default'),
    'do_algebraic', do_algebraic (0))
```

Uses the MATLAB function eigs to calculate a subset of eigenvalues and eigenvectors given the constrained cross-correlation (covariance) matrix with the filename given by `ccmatrix_fn_prefix` and `iteration`. `num_eigs` of the largest eigenvalues and eigenvectors will be calculated, and will be written out as specified by `eig_val_fn_prefix`, `eig_vec_fn_prefix` and `iteration` respectively. Two options `eigs_iterations` and `eigs_tolerance` are also available to tune how eigs is run. If the string 'default' is given for either the default values in eigs will be used. If `do_algebraic` evaluates to true as a boolean 'la' will be used in place of 'lm' in the call to eigs, this could be a valid option in the case when 'lm' returns negative eigenvalues.

55.3.1 Example

```
subtom_eigs(...
    'ccmatrix_fn_prefix', 'class/ccmatrix', ...
    'eig_vec_fn_prefix', 'class/eigvec', ...
    'eig_val_fn_prefix', 'class/eigval', ...
    'iteration', 1, ...
    'num_eigs', 50, ...
    'eigs_iterations', 'default', ...
    'eigs_tolerance', 'default', ...
    'do_algebraic', 1)
```

55.3.2 See Also

- *subtom_join_ccmatrix*
- *subtom_join_eigencoeffs_pca*
- *subtom_join_eigenvolumes*
- *subtom_parallel_ccmatrix*
- *subtom_parallel_eigencoeffs_pca*
- *subtom_parallel_eigenvolumes*
- *subtom_parallel_xmatrix_pca*
- *subtom_prepare_ccmatrix*
- *subtom_svds*

55.4 subtom_join_ccmatrix

Combines chunks of the Cross-Correlation matrix into the final full matrix.

```
subtom_join_ccmatrix(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ccmatrix_fn_prefix', ccmatrix_fn_prefix ('class/ccmatrix_pca'),
    'iteration', iteration (1),
    'num_ccmatrix_batch', num_ccmatrix_batch (1))
```

Looks for partial chunks of the ccmatrix with the file name given by `ccmatrix_fn_prefix`, `iteration` and # where # is from 1 to `num_ccmatrix_batch`, and then combines these chunks into the final ccmatrix and writes it out to the file specified by `ccmatrix_fn_prefix` and `iteration`

55.4.1 Example

```
subtom_join_ccmatrix(...
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ccmatrix_fn_prefix', 'class/ccmatrix', ...
    'iteration', 1, ...
    'num_ccmatrix_batch', 1)
```

55.4.2 See Also

- *subtom_eigs*
- *subtom_join_eigencoeffs_pca*
- *subtom_join_eigenvolumes*
- *subtom_parallel_ccmatrix*
- *subtom_parallel_eigencoeffs_pca*
- *subtom_parallel_eigenvolumes*
- *subtom_parallel_xmatrix_pca*
- *subtom_prepare_ccmatrix*
- *subtom_svds*

55.5 subtom_join_eigencoeffs_pca

Randomizes a given number of classes in a motive list.

```
subtom_join_eigencoeffs_pca(
    'eig_coeff_fn_prefix', eig_coeff_fn_prefix ('class/eigcoeff_pca'),
    'iteration', iteration (1),
    'num_coeff_batch', num_coeff_batch (1))
```

Looks for partial chunks of the low-rank approximation coefficients of projected particles with the file name given by `eig_coeff_fn_prefix`, `iteration` and # where # is from 1 to `num_coeff_batch`, and combines them into a final matrix of coefficients written out as specified by `eig_coeff_fn_prefix` and `iteration`.

55.5.1 Example

```
subtom_join_eigencoeffs_pca(...
    'eig_coeff_fn_prefix', 'class/eigcoeff', ...
    'iteration', 1, ...
    'num_coeff_batch', 100)
```

55.5.2 See Also

- *subtom_eigs*
- *subtom_join_ccmatrix*
- *subtom_join_eigenvolumes*
- *subtom_parallel_ccmatrix*
- *subtom_parallel_eigencoeffs_pca*
- *subtom_parallel_eigenvolumes*
- *subtom_parallel_xmatrix_pca*
- *subtom_prepare_ccmatrix*
- *subtom_svds*

55.6 subtom_join_eigenvolumes

Computes the final sum of projections of the data onto Eigenvectors.

```
subtom_join_eigenvolumes(
    'eig_vol_fn_prefix', eig_vol_fn_prefix ('class/eigvol_pca'),
    'iteration', iteration (1),
    'num_eigs', num_eigs (40),
    'num_xmatrix_batch', num_xmatrix_batch (1))
```

Calculates the sum of previously calculated Eigenvolume partial sums, (projections onto previously determined Eigen (or left singular) vectors), which can then be used to determine which vectors can best influence classification. The Eigenvolumes are expected to have been split into NUM_XMATRIX_BATCH sums. The output averages will be written out as given by EIG_VOL_FN_PREFIX, ITERATION and #, where the # is the particular Eigenvolume being written out from 1 to NUM_EIGS. For easier viewing a montage of the Eigenvolumes is made along the X, Y, and Z axes, written out as specified by EIG_VOL_FN_PREFIX, (X,Y,Z) and ITERATION.

55.6.1 Example

```
subtom_join_eigenvolumes(...
    'eig_vol_fn_prefix', 'class/eigvol', ...
    'iteration', 1, ...
    'num_eigs', 40, ...
    'num_xmatrix_batch', 100)
```

55.6.2 See Also

- *subtom_eigs*
- *subtom_join_ccmatrix*
- *subtom_join_eigencoeffs_pca*
- *subtom_parallel_ccmatrix*
- *subtom_parallel_eigencoeffs_pca*
- *subtom_parallel_eigenvolumes*
- *subtom_parallel_xmatrix_pca*
- *subtom_prepare_ccmatrix*
- *subtom_svds*

55.7 subtom_parallel_ccmatrix

Calculates pairwise Constrained Cross-Correlation scores of aligned particles.

```
subtom_parallel_ccmatrix(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ccmatrix_fn_prefix', ccmatrix_fn_prefix ('class/ccmatrix_pca'),
    'weight_fn_prefix', weight_fn_prefix ('otherinputs/ampspec'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'mask_fn', mask_fn ('none'),
    'high_pass_fp', high_pass_fp (0),
    'high_pass_sigma', high_pass_sigma (0),
    'low_pass_fp', low_pass_fp (0),
    'low_pass_sigma', low_pass_sigma (0),
    'nfold', nfold (1),
    'iteration', iteration (1),
    'tomo_row', tomo_row (7),
    'prealigned', prealigned (0),
    'num_ccmatrix_batch', num_ccmatrix_batch (1),
    'process_idx', process_idx (1))
```

Aligns a subset of particles using the rotations and shifts in the file given by `all_motl_fn_prefix` and `iteration`. If `prealigned` evaluates to true as boolean, then the particles in `ptcl_fn_prefix` are assumed to be prealigned, which should increase the speed of the processing. The subset of particles compared is specified by the file given by `ccmatrix_fn_prefix`, `iteration`, and `process_idx` appended with '_pairs', and the output list of cross-correlation coefficients will be written out to the file specified by `ccmatrix_fn_prefix`, `iteration`, and `process_idx`. Fourier weight volumes with name prefix `weight_fn_prefix` will also be aligned so that the cross-correlation cofficient can be constrained to only overlapping shared regions of Fourier space. `tomo_row` describes which row of the MOTL file is used to determine the correct tomogram Fourier weight file. The correlation is also constrained by a bandpass filter specified by `high_pass_fp`, `high_pass_sigma`, `low_pass_fp` and `low_pass_sigma`.

55.7.1 Example

```
subtom_parallel_ccmatrix(
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ccmatrix_fn_prefix', 'class/ccmatrix', ...
    'weight_fn_prefix', 'otherinputs/ampspec', ...
    'ptcl_fn_prefix', 'subtomograms/alisubtomo', ...
    'mask_fn', 'otherinputs/classification_mask.em', ...
    'high_pass_fp', 1, ...
    'high_pass_sigma', 2, ...
    'low_pass_fp', 15, ...
    'low_pass_sigma', 3, ...
    'nfold', 1, ...
    'iteration', 1, ...
    'tomo_row', 7, ...
    'prealigned', 1, ...
    'num_ccmatrix_batch', 100, ...
    'process_idx', 1)
```

55.7.2 See Also

- *subtom_eigs*
- *subtom_join_ccmatrix*
- *subtom_join_eigencoeffs_pca*
- *subtom_join_eigenvolumes*
- *subtom_parallel_eigencoeffs_pca*
- *subtom_parallel_eigenvolumes*
- *subtom_parallel_xmatrix_pca*
- *subtom_prepare_ccmatrix*
- *subtom_svds*

55.8 subtom_parallel_eigencoeffs_pca

Computes particle Eigencoefficients

```
subtom_parallel_eigencoeffs_pca(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'eig_coeff_fn_prefix', eig_coeff_fn_prefix ('class/eigcoeff_pca'),
    'eig_val_fn_prefix', eig_val_fn_prefix ('class/eigval_pca'),
    'eig_vol_fn_prefix', eig_vol_fn_prefix ('class/eigvol_pca'),
    'weight_fn_prefix', weight_fn_prefix ('otherinputs/ampspec'),
    'mask_fn', mask_fn ('none'),
    'high_pass_fp', high_pass_fp (0),
    'high_pass_sigma', high_pass_sigma (0),
    'low_pass_fp', low_pass_fp (0),
```

(continues on next page)

(continued from previous page)

```
    'low_pass_sigma', low_pass_sigma (0),
    'nfold', nfold (1),
    'apply_weight', apply_weight (0),
    'tomo_row', tomo_row (7),
    'iteration', iteration (1),
    'prealigned', prealigned (0),
    'num_coeff_batch', num_coeff_batch (1),
    'process_idx', process_idx (1))
```

Takes a batch subset of particles described by `all_motl_fn_prefix` with filenames given by `ptcl_fn_prefix`, band-pass filters them as described by `high_pass_fp`, `high_pass_sigma`, `low_pass_fp`, and `low_pass_sigma`, optionally applies C-symmetry specified by `nfold`, and projects them onto by default the Eigenvolumes specified by `eig_vol_fn_prefix`. This determines a set of coefficients describing a low-rank approximation of the data. A subset of this coefficient matrix is written out based on `eig_coeff_fn_prefix` and `process_idx`, with there being `num_coeff_batch` batches in total.

If `apply_weight` is set to 1 the Eigenvolumes will be reweighted using the correct weight of each particle as described by `weight_fn_prefix` and `tomo_row`, then each particle will be read and projected in a loop. If `prealigned` is set to 1, then it is understood that the particles have been prealigned beforehand and the alignment of the particles can be skipped to save time. `mask_fn` describes the mask used throughout classification and 'none' describes a default spherical mask.

55.8.1 Example

```
subtom_parallel_eigencoeffs_pca(
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ptcl_fn_prefix', 'subtomograms/subtomo_ali', ...
    'eig_coeff_fn_prefix', 'class/eigcoeff', ...
    'eig_val_fn_prefix', 'class/eigval', ...
    'eig_vol_fn_prefix', 'class/eigvol', ...
    'weight_fn_prefix', 'otherinputs/ampspec', ...
    'mask_fn', 'otherinputs/classification_mask.em', ...
    'high_pass_fp', 1, ...
    'high_pass_sigma', 2, ...
    'low_pass_fp', 15, ...
    'low_pass_sigma', 3, ...
    'nfold', 1, ...
    'apply_weight', 1, ...
    'tomo_row', 7, ...
    'iteration', 1, ...
    'prealigned', 1, ...
    'num_coeff_batch', 100, ...
    'process_idx', 1)
```

55.8.2 See Also

- *subtom_eigs*
- *subtom_join_ccmatrix*
- *subtom_join_eigencoeffs_pca*
- *subtom_join_eigenvolumes*
- *subtom_parallel_ccmatrix*
- *subtom_parallel_eigenvolumes*
- *subtom_parallel_xmatrix_pca*
- *subtom_prepare_ccmatrix*
- *subtom_svds*

55.9 subtom_parallel_eigenvolumes

Computes projections of data onto Eigenvectors.

```
subtom_parallel_eigenvolumes(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'eig_vec_fn_prefix', eig_vec_fn_prefix ('class/eigvec_pca'),
    'eig_val_fn_prefix', eig_val_fn_prefix ('class/eigval_pca'),
    'xmatrix_fn_prefix', xmatrix_fn_prefix ('class/xmatrix_pca'),
    'eig_vol_fn_prefix', eig_vol_fn_prefix ('class/eigvol_pca'),
    'mask_fn', mask_fn ('none'),
    'iteration', iteration (1),
    'num_xmatrix_batch', num_xmatrix_batch (1),
    'process_idx', process_idx (1))
```

Calculates the summed projections of particles onto previously determined Eigen (or left singular) vectors, by means of an also previously calculated X-matrix to produce Eigenvolumes which can then be used to determine which vectors can best influence classification. The Eigenvectors are named based on `eig_vec_fn_prefix` and `iteration` and the X-Matrix is named based on `xmatrix_fn_prefix`, `iteration`, and `process_idx`. The Eigenvolumes are also masked by the file specified by `mask_fn`. The Eigenvolumes are split into `num_xmatrix_batch` sums, which is the same number of batches that the X-Matrix was broken into in its computation. `process_idx` is a counter that designates the current batch being determined. The output sum Eigenvolume will be written out as specified by `eig_vol_fn_prefix`, `iteration`, `process_idx` and # where the # is the particular Eigenvolume being written out.

55.9.1 Example

```
subtom_parallel_eigenvolumes(
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ptcl_fn_prefix', 'subtomograms/subtomo', ...
    'eig_vec_fn_prefix', 'class/eigvec', ...
    'eig_val_fn_prefix', 'class/eigval', ...
    'xmatrix_fn_prefix', 'class/xmatrix', ...
    'eig_vol_fn_prefix', 'class/eigvol', ...
```

(continues on next page)

(continued from previous page)

```
    'mask_fn', 'otherinputs/classification_mask.em', ...
    'iteration', 1, ...
    'num_xmatrix_batch', 100, ...
    'process_idx', 1)
```

55.9.2 See Also

- *subtom_eigs*
- *subtom_join_ccmatrix*
- *subtom_join_eigencoeffs_pca*
- *subtom_join_eigenvolumes*
- *subtom_parallel_ccmatrix*
- *subtom_parallel_eigencoeffs_pca*
- *subtom_parallel_xmatrix_pca*
- *subtom_prepare_ccmatrix*
- *subtom_svds*

55.10 subtom_parallel_xmatrix_pca

Calculates chunks of the X-matrix for processing.

```
subtom_parallel_xmatrix_pca(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'xmatrix_fn_prefix', xmatrix_fn_prefix ('class/xmatrix_pca'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'mask_fn', mask_fn ('none'),
    'high_pass_fp', high_pass_fp (0),
    'high_pass_sigma', high_pass_sigma (0),
    'low_pass_fp', low_pass_fp (0),
    'low_pass_sigma', low_pass_sigma (0),
    'nfold', nfold (1),
    'iteration', iteration (1),
    'prealigned', prealigned (0),
    'num_xmatrix_batch', num_xmatrix_batch (1),
    'process_idx', process_idx (1))
```

Aligns a subset of particles using the rotations and shifts given by `all_motl_fn_prefix` and `iteration`, band-pass filters the particle as described by `high_pass_fp`, `high_pass_sigma`, `low_pass_fp`, and `low_pass_sigma`, optionally applies `nfold` C-symmetry, and then places these voxels as a 1-D row vector in a data sub-matrix which is historically known as the X-matrix (See Borland, Van Heel 1990 J. Opt. Soc. Am. A). This X-matrix can then be used to speed up the calculation of Eigenvolumes and Eigencoefficients used for classification. The subset of particles compared is specified by the number of particles in the motive list and the number of requested batches specified by `num_xmatrix_batch`, with the specific subset deteremined by `process_idx`. The X-matrix chunk will be written out as specified by `xmatrix_fn_prefix`, `iteration` and `process_idx`. The location of the particles is specified by `ptcl_fn_prefix`. If `prealigned` evaluates to true as a boolean then the particles are assumed to be prealigned, which should increase speed of computation of CC-Matrix calculations. Particles in the X-matrix will be masked by

the file given by `mask_fn`. If the string 'none' is used in place of `mask_fn`, a default spherical mask is applied. This mask should be a binary mask and only voxels within the mask are placed into the X-matrix which can greatly speed up computations.

55.10.1 Example

```
subtom_parallel_xmatrix_pca(
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'xmatrix_fn_prefix', 'class/xmatrix', ...
    'ptcl_fn_prefix', 'subtomograms/alisubtomo', ...
    'mask_fn', 'combinedmotl/classification_mask.em', ...
    'high_pass_fp', 1, ...
    'high_pass_sigma', 2, ...
    'low_pass_fp', 15, ...
    'low_pass_sigma', 3, ...
    'nfold', 1, ...
    'iteration', 1, ...
    'prealigned', 1, ...
    'num_xmatrix_batch', 100, ...
    'process_idx', 1)
```

55.10.2 See Also

- *subtom_eigs*
- *subtom_join_ccmatrix*
- *subtom_join_eigencoeffs_pca*
- *subtom_join_eigenvolumes*
- *subtom_parallel_ccmatrix*
- *subtom_parallel_eigencoeffs_pca*
- *subtom_parallel_eigenvolumes*
- *subtom_prepare_ccmatrix*
- *subtom_svds*

55.11 subtom_prepare_ccmatrix

Calculates batch of pairwise comparisons of particles.

```
subtom_prepare_ccmatrix(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ccmatrix_fn_prefix', ccmatrix_fn_prefix ('class/ccmatrix_pca'),
    'iteration', iteration (1),
    'num_ccmatrix_batch', num_ccmatrix_batch (1))
```

Figures out the pairwise comparisons to make from the motivelist given by `all_motl_fn_prefix` and `iteration`, and breaks up these comparisons into `num_ccmatrix_batch` batches for parallel computation. Each batch is written

out as an array with the 'reference' particle index and 'target' particle index to an EM-file with the name described by `ccmatrix_fn_prefix`, `iteration`, #, and '_pairs' where the # is the batch index.

55.11.1 Example

```
subtom_prepare_ccmatrix(
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ccmatrix_fn_prefix', 'class/ccmatrix', ...
    'iteration', 1, ...
    'num_ccmatrix_batch', 1000);
```

55.11.2 See Also

- *subtom_eigs*
- *subtom_join_ccmatrix*
- *subtom_join_eigencoeffs_pca*
- *subtom_join_eigenvolumes*
- *subtom_parallel_ccmatrix*
- *subtom_parallel_eigencoeffs_pca*
- *subtom_parallel_eigenvolumes*
- *subtom_parallel_xmatrix_pca*
- *subtom_svds*

55.12 subtom_svds

Uses MATLAB svds to calculate a subset of singular values/vectors.

```
subtom_svds(
    'ccmatrix_fn_prefix', ccmatrix_fn_prefix ('pca/ccmatrix'),
    'eig_vec_fn_prefix', eig_vec_fn_prefix ('pca/eigvec'),
    'eig_val_fn_prefix', eig_val_fn_prefix ('pca/eigval'),
    'iteration', iteration (1),
    'num_svs', num_svs (40),
    'svds_iterations', svds_iterations ('default'),
    'svds_tolerance', svds_tolerance ('default'))
```

Uses the MATLAB function svds to calculate a subset of singular values and vectors given the constrained cross-correlation (covariance) matrix with the filename given by `ccmatrix_fn_prefix` and `iteration`. `num_svs` of the largest singular values and vectors will be calculated, and will be written out based on `eig_val_fn_prefix` and `iteration`; and `eig_vec_fn_prefix` and `iteration` respectively. Two options `svds_iterations` and `svds_tolerance` are also available to tune how svds is run. If the string 'default' is given for either the default values in eigs will be used.

55.12.1 Example

```
subtom_svds(...
    'ccmatrix_fn_prefix', 'pca/ccmatrix', ...
    'eig_vec_fn_prefix', 'pca/eigvec', ...
    'eig_val_fn_prefix', 'pca/eigval', ...
    'iteration', 1, ...
    'num_svs', 50, ...
    'svds_iterations', 'default', ...
    'svds_tolerance', 'default')
```

55.12.2 See Also

- *subtom_eigs*
- *subtom_join_ccmatrix*
- *subtom_join_eigencoeffs_pca*
- *subtom_join_eigenvolumes*
- *subtom_parallel_ccmatrix*
- *subtom_parallel_eigencoeffs_pca*
- *subtom_parallel_eigenvolumes*
- *subtom_parallel_xmatrix_pca*
- *subtom_prepare_ccmatrix*

55.13 Indices and tables

- genindex
- modindex
- search

CHAPTER

FIFTYSIX

SUBTOM: WEDGE-MASKED DIFFERENCE CLASSIFICATION

56.1 Wedge-Masked Difference Classification

In Wedge-Masked Difference (WMD) classification the full set of particles are simplified into a new lower-dimensional representation by means of Singular Value decomposition after attempting to take into account the effects of the missing-wedge. Particles projected onto these most variable basis-vectors then can be clustered using a variety of methods.

Within subTOM, wedge-masked differences (the result of the subtraction of the overall reference, weighted with the particles missing-wedge, and the particle itself) are first compiled into a 2-D Matrix denoted here as the D-Matrix, which holds the aligned, band-pass filtered and masked difference data. To speed up calculation particles can be pre-aligned using the function `subtom_parallel_prealign`. Batches of the D-Matrix are calculated in parallel with `subtom_parallel_dmatrix` and then combined and column-centered with `subtom_join_dmatrix`.

Next the D-Matrix is decomposed by Singular Value decomposition as to skip calculation of the covariance matrix as described in J. Heumann et al. in J. Struct. Biol. 2011. This determines a set of right Singular vectors and Singular values and these are used along with the D-Matrix to determine the Eigenvolumes of the dataset with `subtom_eigenvolumes_wmd`.

These volumes are then used to determine the low-rank approximation coefficients in volume space for clustering. A larger particle superset can be projected onto the volumes to speed up classification of large datasets. Coefficients are also calculated in parallel in batches with `subtom_parallel_coeffs` and joined with `subtom_join_coeffs`.

Finally using a user-selected subset of the determined coefficients, the data is clustered either by Hierarchical Ascendant Clustering using a Ward distance criterion, K-Means clustering, or a Gaussian Mixture model with the function `subtom_cluster`. This clustering is then used to generate the final class averages.

56.2 subtom_wmd

The main WMD pipeline process script of subTOM.

This subtomogram classification script uses nine MATLAB compiled scripts below:

- *subtom_parallel_prealign*
- *subtom_parallel_dmatrix*
- *subtom_join_dmatrix*
- *subtom_eigenvolumes_wmd*
- *subtom_parallel_coeffs*
- *subtom_join_coeffs*
- *subtom_cluster*

- *subtom_parallel_sums_cls*
- *subtom_weighted_average_cls*

56.2.1 Options

Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

local_dir Absolute path to the folder on a group share, if the scratch directory is cleaned and deleted regularly this can set a local directory to which the important results will be copied. If this is not needed it can be skipped with the option skip_local_copy below.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

Variables

cluster_exec Cluster executable.

par_coeff_exec Parallel Coefficient executable.

coeff_exec Final Coefficient executable.

eigvol_exec Eigenvolume Calculation executable.

preali_exec Parallel Subtomogram prealign executable.

par_dmatrix_exec Parallel D-Matrix executable.

dmatrix_exec Final D-Matrix executable.

sum_exec Parallel Summing executable

avg_exec Final Averaging executable

motl_dump_exec MOTL dump executable

Memory Options

mem_free The amount of memory the job requires. This variable determines whether a number of CPUs will be requested to be dedicated for each job. At 24G, one half of the CPUs on a node will be dedicated for each of the processes (12 CPUs). At 48G, all of the CPUs on the node will be dedicated for each of the processes (24 CPUs).

mem_max The upper bound on the amount of memory the job is allowed to use. If any of the processes request or require more memory than this, the queue will kill the process. This is more of an option for safety of the cluster to prevent the user from crashing the cluster requesting too much memory.

Other Cluster Options

job_name The job name prefix that will be used for the cluster submission scripts, log files, and error logs for the processing. Be careful that this name is unique because previous submission scripts, logs, and error logs with the same job name prefix will be overwritten in the case of a name collision.

array_max The maximum number of jobs per cluster submission script. Cluster submission scripts work using the array feature common to queuing systems, and this value is the maximum array size used in a script. If the user requests more batches of processing than this value, then the submission scripts will be split into files of up to array_max jobs.

max_jobs The maximum number of jobs for alignment. If the number of batches / exceeds this value the script will immediately quit.

run_local If the user wants to skip the cluster and run the job locally, this value should be set to 1.

skip_local_copy Set this option to 1 to skip the copying of data to local_dir.

Parallelization Options

iteration The index of the references to generate : input will be all_motl_fn_prefix_iteration.em (define as integer e.g. iteration=1)

num_dmatrix_prealign_batch Number of batches to split the parallel particle prealignment for the D-Matrix calculation into. If you are not doing prealignment you can ignore this option.

num_dmatrix_batch Number of batches to split the parallel D-Matrix calculation job into.

num_coeff_prealign_batch Number of batches to split the parallel particle prealignment for the coefficients calculations into. If you are not doing prealignment you can ignore this option.

num_coeff_batch Number of batches to split the parallel coefficient calculation into.

num_avg_batch The number of batches to split the parallel subtomogram averaging job into.

Subtomogram Classification Workflow Options

D-Matrix Options

high_pass_fp High pass filter cutoff (in transform units (pixels): calculate as (box_size * pixelsize) / (resolution_real) (define as integer).

high_pass_sigma High pass filter falloff sigma (in transform units (pixels): describes a Gaussian sigma for the falloff of the high-pass filter past the cutoff above.

low_pass_fp Low pass filter (in transform units (pixels): calculate as (box_size * pixelsize) / (resolution_real) (define as integer).

low_pass_sigma Low pass filter falloff sigma (in transform units (pixels): describes a Gaussian sigma for the falloff of the low-pass filter past the cutoff above.

nfold Symmetry to apply to each pair of particle and reference in D-Matrix calculation, if no symmetry nfold=1 (define as integer e.g. nfold=3).

tomo_row Which row in the motl file contains the correct tomogram number. Usually row 5 and 7 both correspond to the correct value and can be used interchangeably, but there are instances when 5 contains a sequential ordered value starting from 1, while 7 contains the correct corresponding tomogram.

dmatrix_prealign If you want to pre-align all of the particles to speed up the D-Matrix calculation, set the following to 1, otherwise the particles will be aligned during the computation.

D-Matrix File Options

dmatrix_all_motl_fn_prefix Relative path and name of the concatenated motivelist of all particles (e.g. all-motl_iter.em , the variable will be written as a string e.g. dmatrix_all_motl_fn_prefix='sub-directory/allmotl').

dmatrix_fn_prefix Relative path and name of the D-Matrix.

ptcl_fn_prefix Relative path and name of the subtomograms (e.g. part_n.em , the variable will be written as a string e.g. ptcl_fn_prefix='sub-directory/part').

dmatrix_ref_fn_prefix Relative path and name prefix of the reference volume used for calculating the wedge-masked differences (e.g. ref_iter.em, the variable will be written as a string e.g. dmatrix_ref_fn_prefix='sub-directory/ref')

weight_fn_prefix Relative path and name of the weight file, used for calculating the wedge-masked differences.

mask_fn Relative path and name of the classification mask. This should be a binary mask as correlations are done in real-space, and calculations will only be done using voxels passed by the mask, so smaller masks will run faster. If you want to use the default spherical mask set mask_fn to 'none'.

Eigenvolume Options

num_svs The number of right Singular Vectors and Singular Values to calculate.

svds_iterations The following allows you to adjust the number of iterations to use in the decomposition. If you want to use the default number of iterations leave this set to 'default'.

svds_tolerance The following allows you to adjust the convergence tolerance of the decomposition calculation. If you want to use the default tolerance leave this set to 'default'.

Eigenvolumes File Options

eig_val_fn_prefix Relative path and name of the Eigenvalues.

eig_vol_fn_prefix Relative path and name of the Eigenvolumes.

variance_fn_prefix Relative path and name prefix of the calculated variance map.

Coefficient Options

coeff_prealign If you want to pre-align all of the particles to speed up the coefficient calculation, set the following to 1, otherwise the particles will be aligned during the computation.

Eigencoefficient File Options

coeff_all_motl_fn_prefix Relative path and name of the concatenated motivelist to project onto the Eigenvolumes. This can be a larger motivelist than the one used to calculate the D-Matrix and Eigenvolumes.

coeff_fn_prefix Relative path and name of the coefficients.

Clustering Options

cluster_type The following determines which algorithm will be used to cluster the determined Eigencoefficients. The valid options are K-means clustering, 'kmeans', Hierarchical Ascendent Clustering using a Ward Criterion, 'hac', and a Gaussian Mixture Model, 'gaussmix'.

coeff_idxs Determines which coefficients are used to cluster. The format should be a semicolon-separated list that also supports ranges with a dash (-), for example 1-5;7;15-19 would select the first five coefficients, the seventh and the fifteenth through the nineteenth for classification. If it is left as "all" all coefficients will be used.

num_classes How many classes should the particles be clustered into.

Clustering File Options

cluster_all_motl_fn_prefix Relative path and name of the concatenated motivelist of the output classified particles.

Averaging File Options

ref_fn_prefix Relative path and name prefix of the reference volumes (e.g. ref_iter.em, the variable will be written as a string e.g. ref_fn_prefix='sub-directory/ref')

weight_sum_fn_prefix Relative path and name prefix of the partial weight files.

56.2.2 Example

```
scratch_dir="${PWD}"

local_dir=""

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="XXXINSTALLATION_DIRXXX/bin"

cluster_exec="${exec_dir}/classification/general/subtom_cluster"

par_eigcoeff_exec="${exec_dir}/classification/wmd/subtom_parallel_coeffs"

eigcoeff_exec="${exec_dir}/classification/wmd/subtom_join_coeffs"

eigvol_exec="${exec_dir}/classification/wmd/subtom_eigenvolumes_wmd"

preali_exec="${exec_dir}/classification/general/subtom_parallel_prealign"

par_xmatrix_exec="${exec_dir}/classification/wmd/subtom_parallel_dmatrix"

xmatrix_exec="${exec_dir}/classification/wmd/subtom_join_dmatrix"

sum_exec="${exec_dir}/classification/general/subtom_parallel_sums_cls"

avg_exec="${exec_dir}/classification/general/subtom_weighted_average_cls"

motl_dump_exec="${exec_dir}/MOTL/motl_dump"
```

(continues on next page)

(continued from previous page)

```
mem_free="1G"

mem_max="64G"

job_name="subTOM"

array_max="1000"

max_jobs="4000"

run_local="0"

skip_local_copy="1"

iteration="1"

num_dmatrix_prealign_batch="1"

num_dmatrix_batch="1"

num_coeff_prealign_batch="1"

num_coeff_batch="1"

num_avg_batch="1"

high_pass_fp="1"

high_pass_sigma="2"

low_pass_fp="12"

low_pass_sigma="3"

nfold="1"

tomo_row="7"

dmatrix_prealign=0

dmatrix_all_motl_fn_prefix="combinedmotl/allmotl"

dmatrix_fn_prefix="class/xmatrix_wmd"

ptcl_fn_prefix="subtomograms/subtomo"

dmatrix_ref_fn_prefix="ref/ref"

weight_fn_prefix="otherinputs/ampspec"

mask_fn="none"
```

(continues on next page)

(continued from previous page)

```
num_svs='40'

svds_iterations='default'

svds_tolerance='default'

eig_val_fn_prefix="class/eigval_wmd"

eig_vol_fn_prefix="class/eigvol_wmd"

coeff_prealign="0"

coeff_all_motl_fn_prefix="combinedmotl/allmotl"

coeff_fn_prefix="class/coeff_wmd"

cluster_type="kmeans"

coeff_idxs="all"

num_classes=2

cluster_all_motl_fn_prefix="class/allmotl_wmd"

ref_fn_prefix="class/ref_wmd"

weight_sum_fn_prefix="class/wei_wmd"
```

56.3 subtom_eigenvolumes_wmd

Computes Singular Value Decomposition of D-Matrix and projects data on right Singular Vectors.

```
subtom_eigenvolumes_wmd(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'dmatrix_fn_prefix', dmatrix_fn_prefix ('class/dmatrix_wmd'),
    'eig_val_fn_prefix', eig_val_fn_prefix ('class/eigval_wmd'),
    'eig_vol_fn_prefix', eig_vol_fn_prefix ('class/eigvol_wmd'),
    'variance_fn_prefix', variance_fn_prefix ('class/variance_wmd'),
    'mask_fn', mask_fn ('none'),
    'iteration', iteration (1),
    'num_svs', num_svs (40),
    'svds_iterations', svds_iterations ('default'),
    'svds_tolerance', svds_tolerance ('default'))
```

Calculates `num_svs` weighted projections of wedge-masked differences onto the same number of determined Right-Singular Vectors, by means of the Singular Value Decomposition of a previously calculated D-matrix, named as given by `dmatrix_fn_prefix` and `iteration` to produce Eigenvolumes which can then be used to determine which vectors can best influence classification. The Eigenvolumes are also masked by the file specified by `mask_fn`. The output weighted Eigenvolume will be written out as specified by `eig_vol_fn_prefix`, `iteration` and #, where the # is

the particular Eigenvolume being written out. The calculated Eigenvalues which correspond to the square of the singular vectors are also written oun as given by `eig_val_fn_prefix` and `iteration`, and the variance map of the data is written out as determined by `variance_fn_prefix` and `iteration`. Two options `svds_iterations` and `svds_tolerance` are also available to tune how svds is run. If the string 'default' is given for either the default values in svds will be used.

56.3.1 Example

```
subtom_eigenvolumes_wmd(...
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ptcl_fn_prefix', 'subtomograms/subtomo', ...
    'dmatrix_fn_prefix', 'class/dmatrix', ...
    'eig_val_fn_prefix', 'class/eigval', ...
    'eig_vol_fn_prefix', 'class/eigvol', ...
    'variance_fn_prefix', 'class/variance', ...
    'mask_fn', 'class/class_mask.em', ...
    'iteration', 1, ...
    'num_svs', 40, ...
    'svds_iterations', 'default', ...
    'svds_tolerance', 'default')
```

56.3.2 See Also

- *subtom_join_coeffs*
- *subtom_join_dmatrix*
- *subtom_parallel_coeffs*
- *subtom_parallel_dmatrix*

56.4 subtom_join_coeffs

Combines coefficient batches into the final matrix.

```
subtom_join_coeffs(
    'coeff_fn_prefix', coeff_fn_prefix ('class/coeff_wmd'),
    'iteration', iteration (1),
    'num_coeff_batch', num_coeff_batch (1))
```

Looks for partial chunks of the low-rank approximation coefficients of projected particles with the file name given by `coeff_fn_prefix`, `iteration` and # where # is from 1 to `num_coeff_batch`, and combines them into a final matrix of coefficients written out as described by `coeff_fn_prefix`, and `iteration`.

56.4.1 Example

```
subtom_join_coeffs(...
    'coeff_fn_prefix', 'class/coeff', ...
    'iteration', 1, ...
    'num_coeff_batch', 100)
```

56.4.2 See Also

- *subtom_eigenvolumes_wmd*
- *subtom_join_dmatrix*
- *subtom_parallel_coeffs*
- *subtom_parallel_dmatrix*

56.5 subtom_join_dmatrix

Combines chunks of D-Matrix into the final matrix.

```
subtom_join_dmatrix(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'dmatrix_fn_prefix', dmatrix_fn_prefix ('class/dmatrix_wmd'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'mask_fn', mask_fn ('none'),
    'iteration', iteration (1),
    'num_dmatrix_batch', num_dmatrix_batch (1))
```

Looks for partial chunks of the D-matrix with the file name given by `dmatrix_fn_prefix`, `iteration`, and # where # is from 1 to `num_dmatrix_batch`, and combines them into a final matrix of differences written out as described by `dmatrix_fn_prefix` and `iteration`.

56.5.1 Example

```
subtom_join_dmatrix(
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'dmatrix_fn_prefix', 'class/dmatrix', ...
    'ptcl_fn_prefix', 'subtomograms/subtomo', ...
    'mask_fn', 'otherinputs/classification_mask.em', ...
    'iteration', 1, ...
    'num_dmatrix_batch', 100);
```

56.5.2 See Also

- *subtom_eigenvolumes_wmd*
- *subtom_join_coeffs*
- *subtom_parallel_coeffs*
- *subtom_parallel_dmatrix*

56.6 subtom_parallel_coeffs

Computes wedge-masked difference coefficients.

```
subtom_parallel_coeffs(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'dmatrix_fn_prefix', dmatrix_fn_prefix ('class/dmatrix_wmd'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'ref_fn_prefix', ref_fn_prefix ('ref/ref'),
    'coeff_fn_prefix', coeff_fn_prefix ('class/coeff_wmd'),
    'eig_val_fn_prefix', eig_val_fn_prefix ('class/eigval_wmd'),
    'eig_vol_fn_prefix', eig_vol_fn_prefix ('class/eigvol_wmd'),
    'weight_fn_prefix', weight_fn_prefix ('otherinputs/ampspec'),
    'mask_fn', mask_fn ('none'),
    'high_pass_fp', high_pass_fp (0),
    'high_pass_sigma', high_pass_sigma (0),
    'low_pass_fp', low_pass_fp (0),
    'low_pass_sigma', low_pass_sigma (0),
    'nfold', nfold (1),
    'prealigned', prealigned (0),
    'iteration', iteration (1),
    'tomo_row', tomo_row (7),
    'num_coeff_batch', num_coeff_batch (1),
    'process_idx', process_idx (1))
```

Takes a batch subset of particles described by `all_motl_fn_prefix` with filenames given by `ptcl_fn_prefix`, band-pass filters them as described by `high_pass_fp`, `high_pass_sigma`, `low_pass_fp`, and `low_pass_sigma`, optionally applies `nfold` C-symmetry, and projects them onto the Eigenvolumes specified by `eig_vol_fn_prefix`. This determines a set of coefficients describing a low-rank approximation of the data. A subset of this coefficient matrix is written out based on `coeff_fn_prefix` and `process_idx`, with there being `num_coeff_batch` batches in total.

If `apply_weight` is set to 1 the Eigenvolumes will be reweighted using the correct weight of each particle as described by `weight_fn_prefix` and `tomo_row`, then each particle will be read and projected in a loop. If `prealigned` is set to 1, then it is understood that the particles have been prealigned beforehand and the alignment of the particles can be skipped to save time. `mask_fn` describes the mask used throughout classification and 'none' describes a default spherical mask.

56.6.1 Example

```
subtom_parallel_coeffs(
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'dmatrix_fn_prefix', 'class/dmatrix', ...
    'ptcl_fn_prefix', 'subtomograms/subtomo_ali', ...
    'ref_fn_prefix', 'ref/ref', ...
    'coeff_fn_prefix', 'class/coeff', ...
    'eig_val_fn_prefix', 'class/eigval', ...
    'eig_vol_fn_prefix', 'class/eigvol', ...
    'weight_fn_prefix', 'otherinputs/ampspec', ...
    'mask_fn', 'otherinputs/classification_mask.em', ...
    'high_pass_fp', 1, ...
    'high_pass_sigma', 2, ...
    'low_pass_fp', 15, ...
    'low_pass_sigma', 3, ...
    'nfold', 1, ...
    'prealigned', 1, ...
    'iteration', 1, ...
    'tomo_row', 7, ...
    'num_coeff_batch', 100, ...
    'process_idx', 1)
```

56.6.2 See Also

- *subtom_eigenvolumes_wmd*
- *subtom_join_coeffs*
- *subtom_join_dmatrix*
- *subtom_parallel_dmatrix*

56.7 subtom_parallel_dmatrix

Calculates chunks of the D-matrix for processing.

```
subtom_parallel_dmatrix(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'dmatrix_fn_prefix', dmatrix_fn_prefix ('class/dmatrix_wmd'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'ref_fn_prefix', ref_fn_prefix ('ref/ref'),
    'weight_fn_prefix', weight_fn_prefix ('otherinputs/ampspec'),
    'mask_fn', mask_fn ('none'),
    'high_pass_fp', high_pass_fp (0),
    'high_pass_sigma', high_pass_sigma (0),
    'low_pass_fp', low_pass_fp (0),
    'low_pass_sigma', low_pass_sigma (0),
    'nfold', nfold (1),
    'iteration', iteration (1),
    'tomo_row', tomo_row (7),
```

(continues on next page)

(continued from previous page)

```
    'prealigned', prealigned (0),
    'num_dmatrix_batch', num_dmatrix_batch (1),
    'process_idx', process_idx (1))
```

Aligns a subset of particles using the rotations and shifts in the file given by `all_motl_fn_prefix` and `iteration` and then subtracts the particle from the reference specified by `ref_fn_prefix` and `iteration` and places these voxels of the difference as a 1-D row vector in a data sub-matrix which is denoted as the D-matrix (See Heumann, et al. 2011 J. Struct. Biol.). The particle and reference are also filtered by a bandpass filter specified by `high_pass_fp`, `high_pass_sigma`, `low_pass_fp` and `low_pass_sigma`, and optionally symmetrized with `nfold` C-symmetry, before subtracted.The reference is masked in Fourier space using the weight specified by `weight_fn_prefix` and `tomo_row`. The subset of particles compared is specified by the number of particles in the motive list and the number of requested batches specified by `num_dmatrix_batch`, with the specific subset deteremined by `process_idx`. The D-matrix chunk will be written out as given by `dmatrix_fn_prefix`, `iteration`, and `process_idx`. The location of the particles is specified by `ptcl_fn_prefix`. If `prealigned` evaluates to true as a boolean then the particles are assumed to be prealigned, which should increase speed of computation of D-Matrix calculations. Particles in the D-matrix will be masked by the file given by `mask_fn`. If the string 'none' is used in place of `mask_fn`, a default spherical mask is applied. This mask should be a binary mask and only voxels within the mask are placed into the D-matrix which can greatly speed up computations.

56.7.1 Example

```
subtom_parallel_dmatrix(
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'dmatrix_fn_prefix', 'class/dmatrix', ...
    'ptcl_fn_prefix', 'subtomograms/subtomo_ali', ...
    'ref_fn_prefix', 'ref/ref', ...
    'weight_fn_prefix', 'otherinputs/ampspec', ...
    'mask_fn', 'combinedmotl/classification_mask.em', ...
    'high_pass_fp', 1, ...
    'high_pass_sigma', 2, ...
    'low_pass_fp', 15, ...
    'low_pass_sigma', 3, ...
    'nfold', 1, ...
    'iteration', 1, ...
    'prealigned', 1, ...
    'num_dmatrix_batch', 100, ...
    'process_idx', 1)
```

56.7.2 See Also

- *subtom_eigenvolumes_wmd*
- *subtom_join_coeffs*
- *subtom_join_dmatrix*
- *subtom_parallel_coeffs*

56.8 Indices and tables

- genindex
- modindex
- search

CHAPTER

FIFTYSEVEN

SUBTOM

SubTOM - *Subvolume processing scripts with the TOM toolbox* is a collection of scripts form a pipeline for subvolume alignment and averaging of electron cryo-tomography data.

57.1 B-Factor by Subsets

To estimate the B-factor in maps of low to intermediate resolution, Guinier plot analysis is unsuitable because the structure factor dominates the appearance and slope of the amplitude decay at resolutions up to 10 Angstroms.

Therefore another way to estimate the B-factor is to look at how the Resolution decays over smaller and smaller subsets of the particles that form each half-map. A linear function is fit to the reciprocal square of resolutions determined by Gold-standard FSCs against the natural log of asymmetric units.

The method is described in detail in Rosenthal, Henderson 2003, and here the averaging and analysis functions `subtom_maskcorrected_fsc_bfactor`, `subtom_parallel_sums_bfactor`, and `subtom_weighted_average_bfactor` have been slightly modified from their non-bfactor counterparts to first generate the average of successively smaller subsets, with each subset being roughly half of the subset before it until the subset would be less than 128 particles. Then the resolution of each subset is determined and the B-factor is estimated and this estimate B-factor can then be used to sharpen the final post-processed map.

57.2 subtom_b_factor_by_subsets

Estimates the B-Factor by determine the resolution of subsets of particles as described in Rosenthal, Henderson 2003.

This subtomogram averaging analysis script uses three MATLAB compiled scripts below:

- *subtom_maskcorrected_FSC_bfactor*
- *subtom_parallel_sums_bfactor*
- *subtom_weighted_average_bfactor*

57.2.1 Options

Directories

scratch_dir Absolute path to the folder with the input to be processed. Other paths are relative to this one.

mcr_cache_dir Absolute path to MCR directory for the processing.

exec_dir Directory for executables

Variables

sum_exec Parallel Summing executable

avg_exec Weighted Averaging executable

fsc_exec FSC executable

Memory Options

mem_free The amount of memory the job requires for alignment. This variable determines whether a number of CPUs will be requested to be dedicated for each job. At 24G, one half of the CPUs on a node will be dedicated for each of the processes (12 CPUs). At 48G, all of the CPUs on the node will be dedicated for each of the processes (24 CPUs).

mem_max The upper bound on the amount of memory the alignment job is allowed to use. If any of the processes request or require more memory than this, the queue will kill the process. This is more of an option for safety of the cluster to prevent the user from crashing the cluster requesting too much memory.

Other Cluster Options

job_name The job name prefix that will be used for the cluster submission scripts, log files, and error logs for the processing. Be careful that this name is unique because previous submission scripts, logs, and error logs with the same job name prefix will be overwritten in the case of a name collision.

array_max The maximum number of jobs per cluster submission script. Cluster submission scripts work using the array feature common to queuing systems, and this value is the maximum array size used in a script. If the user requests more batches of processing than this value, then the submission scripts will be split into files of up to array_max jobs.

max_jobs The maximum number of jobs for alignment. If the number of batches / exceeds this value the script will immediately quit.

run_local If the user wants to skip the cluster and run the job locally, this value should be set to 1.

Subtomogram Averaging Workflow Options

Parallelization Options

iteration The index of the reference to generate : input will be all_motl_fn_prefix_iteration.em (define as integer)

num_avg_batch The number of batches to split the parallel subtomogram averaging job into.

File Options

all_motl_a_fn_prefix Relative path and name prefix of the concatenated motivelist of all particles in the first half-map.

all_motl_b_fn_prefix Relative path and name prefix of the concatenated motivelist of all particles in the second half-map.

ref_a_fn_prefix Relative path and name prefix of the reference volumes of the first half-map.

ref_b_fn_prefix Relative path and name prefix of the reference volumes of the second half-map.

ptcl_a_fn_prefix Relative path and name prefix of the subtomograms that comprise the first half-map.

ptcl_b_fn_prefix Relative path and name prefix of the subtomograms that comprise the second half-map.

weight_a_fn_prefix Relative path and name prefix of the weight files for the first half-map.

weight_b_fn_prefix Relative path and name prefix of the weight files for the second half-map.

weight_sum_a_fn_prefix Relative path and name prefix of the partial weight files of the first half-map.

weight_sum_b_fn_prefix Relative path and name prefix of the partial weight files of the second half-map.

output_fn_prefix Relative path and prefix for the name of the output maps and figures.

Averaging Options

tomo_row Which row in the motl file contains the correct tomogram number. Usually row 5 and 7 both correspond to the correct value and can be used interchangeably, but there are instances when 5 contains a sequential ordered value starting from 1, while 7 contains the correct corresponding tomogram.

iclass Particles with that number in position 20 of motivelist will be added to new average (define as integer e.g. iclass=1). NOTES: Class 1 is ALWAYS added. Negative classes and class 2 are never added.

Mask Corrected FSC Workflow Options

File Options

fsc_mask_fn Relative or absolute path and name of the FSC mask.

filter_a_fn Relative or absolute path and name of the Fourier filter volume for the first half-map. If not using the option do_reweight just leave this set to “”

filter_b_fn Relative or absolute path and name of the Fourier filter volume for the second half-map. If not using the option do_reweight just leave this set to “”

FSC Options

pixelsize Pixelsize of the half-maps in Angstroms

nfold Symmetry to applied the half-maps before calculating FSC (1 is no symmetry)

rand_threshold The Fourier pixel at which phase-randomization begins is set automatically to the point where the unmasked FSC falls below this threshold.

plot_fsc Plot the FSC curves - 1 = yes, 0 = no

Sharpening Options

do_sharpen Set to 1 to sharpen map or 0 to skip and just calculate the FSC

box_gaussian To remove some of the edge-artifacts associated with map-sharpening the edges of the map can be smoothed with a gaussian. Set to 0 to not smooth the edges, otherwise it must be set to an odd number. If an even number is given one will be added to the value to make it odd.

filter_mode There are two mode used for low pass filtering. The first uses an FSC based threshold (mode 1), i.e. after FSC < 0.143, or a pixel-based resolution threhsold (mode 2).

filter_threshold Set the threshold for the low pass filtering described above. Should be less than 1 for FSC based threshold (mode 1), and an integer value for the Fourier pixel-based threshold (mode 2).

plot_sharpen Plot the sharpening curve - 1 = yes, 0 = no

Reweighting Options

do_reweight Set to 1 to apply the externally calculated Fourier weights filter_A_fn and filter_B_fn to each half-map to reweight the final output map.

57.2.2 Example

```
scratch_dir="${PWD}"

mcr_cache_dir="${scratch_dir}/mcr"

exec_dir="/net/dstore2/teraraid/dmorado/software/subTOM/bin"

sum_exec="${exec_dir}/alignment/subtom_parallel_sums_bfactor"

avg_exec="${exec_dir}/alignment/subtom_weighted_average_bfactor"

fsc_exec="${exec_dir}/analysis/b_factor_by_subsets/subtom_maskcorrected_fsc_bfactor"

mem_free="1G"

mem_max="64G"

job_name="subTOM"

array_max="1000"

max_jobs="4000"

run_local="0"

iteration="1"

num_avg_batch="1"

all_motl_a_fn_prefix="even/combinedmotl/allmotl"
```

(continues on next page)

(continued from previous page)

```
all_motl_b_fn_prefix="odd/combinedmotl/allmotl"

ref_a_fn_prefix="FSC/ref_a"

ref_b_fn_prefix="FSC/ref_b"

ptcl_a_fn_prefix="subtomograms/subtomo"

ptcl_b_fn_prefix="subtomograms/subtomo"

weight_a_fn_prefix="otherinputs/ampspec"

weight_b_fn_prefix="otherinputs/ampspec"

weight_sum_a_fn_prefix="FSC/wei_a"

weight_sum_b_fn_prefix="FSC/wei_b"

output_fn_prefix="FSC/ref_auto_b"

tomo_row="7"

iclass="0"

fsc_mask_fn="FSC/fsc_mask.em"

filter_a_fn=""

filter_b_fn=""

pixelsize=1

nfold=1

rand_threshold=0.8

plot_fsc=1

do_sharpen=1

box_gaussian=1

filter_mode=1

filter_threshold=0.143

plot_sharpen=1

do_reweight=0
```

57.3 subtom_maskcorrected_FSC_bfactor

Calculates "mask-corrected" FSC and sharpened refs.

```
subtom_maskcorrected_fsc_bfactor(
    'ref_a_fn_prefix', ref_a_fn_prefix ('even/ref/ref'),
    'ref_b_fn_prefix', ref_b_fn_prefix ('odd/ref/ref'),
    'motl_a_fn_prefix', motl_a_fn_prefix ('even/combinedmotl/allmotl'),
    'motl_b_fn_prefix', motl_b_fn_prefix ('odd/combinedmotl/allmotl'),
    'fsc_mask_fn', fsc_mask_fn ('FSC/fsc_mask.em'),
    'output_fn_prefix', output_fn_prefix ('FSC/ref'),
    'filter_a_fn', filter_a_fn (''),
    'filter_b_fn', filter_b_fn (''),
    'do_reweight', do_reweight (0),
    'do_sharpen', do_sharpen (0),
    'plot_fsc', plot_fsc (0),
    'plot_sharpen', plot_sharpen (0),
    'filter_mode', filter_mode (1),
    'pixelsize', pixelsize (1.0),
    'nfold', nfold (1),
    'filter_threshold', filter_threshold (0.143),
    'rand_threshold', rand_threshold (0.8),
    'box_gaussian', box_gaussian (1),
    'iclass', iclass (0),
    'iteration', iteration (1))
```

Takes in two references `ref_a_fn_prefix` _#.em and `ref_b_fn_prefix` _#.em where # corresponds to `iteration` and a FSC mask `fsc_mask_fn` and calculates a "mask-corrected" FSC. This works by randomizing the structure factor phases beyond the point where the unmasked FSC curve falls below a given threshold (by default 0.8) and calculating an additional FSC between these phase randomized maps. This allows for the determination of the extra correlation caused by effects of the mask, which is then subtracted from the normal masked FSC curves. The curve will be saved as a Matlab figure and a PDF file, and if `plot_fsc` is true it will also be displayed.

The script can also output maps with the prefix `output_fn_prefix` that have been sharpened with `b_factor` if `do_sharpen` is turned on. This setting has two threshold settings selected using `filter_mode`, FSC (1) and pixel (2). FSC allows you to use a FSC-value `filter_threshold` as a cutoff for the lowpass filter, while using pixels allows you to use an arbitrary resolution cutoff in `filter_threshold`. The sharpening curve will be saved as a Matlab figure and a pdf file, and if `plot_sharpen` is true it will also be displayed.

This function estimates the B-factor to apply versus applying an ad-hoc B-factor by fitting a curve to the drop in resolution as the number of particles decreases. This is detailed in Rosenthal and Henderson, 2003, doi:10.1016/j.jmb.2003.07.013

Finally this script can also perform and output reweighted maps if `do_reweight` is true, and the pre-calculated Fourier weight volumes `filter_a_fn` and `filter_b_fn`.

57.3.1 Example

```
subtom_maskcorrected_fsc_bfactor(...
    'ref_a_fn_prefix', 'even/ref/ref', ...
    'ref_b_fn_prefix', 'odd/ref/ref', ...
    'motl_a_fn_prefix', 'even/combinedmotl', ...
    'motl_b_fn_prefix', 'odd/combinedmotl', ...
    'fsc_mask_fn', 'FSC/fsc_mask.em', ...
    'output_fn_prefix', 'FSC/ref', ...
    'filter_a_fn', '', ...
    'filter_b_fn', '', ...
    'do_reweight', 0, ...
    'do_sharpen', 1, ...
    'plot_fsc', 1, ...
    'plot_sharpen', 1, ...
    'filter_mode', 1, ...
    'pixelsize', 1.35, ...
    'nfold', 6, ...
    'filter_threshold', 0.143, ...
    'rand_threshold', 0.8, ...
    'box_gaussian', 3, ...
    'iclass', 0, ...
    'iteration', 1)
```

57.3.2 See Also

- *subtom_parallel_sums_bfactor*
- *subtom_weighted_average_bfactor*

57.4 subtom_parallel_sums_bfactor

Subsets version of parallel sums for finding B-factor.

```
subtom_parallel_sums_bfactor(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ref_fn_prefix', ref_fn_prefix ('ref/ref'),
    'ptcl_fn_prefix', ptcl_fn_prefix ('subtomograms/subtomo'),
    'weight_fn_prefix', weight_fn_prefix ('otherinputs/ampspec'),
    'weight_sum_fn_prefix, weight_sum_fn_prefix ('otherinputs/wei'),
    'iteration', iteration (1),
    'tomo_row', tomo_row (7),
    'iclass', iclass (0),
    'num_avg_batch', num_avg_batch (1),
    'process_idx', process_idx (1))
```

Aligns a subset of particles using the rotations and shifts in `all_motl_fn_prefix` _#.em where # corresponds to `iteration` in `num_avg_batch` chunks to make a raw particle sum `ref_fn_prefix` _#_###.em where # corresponds to `iteration` and ### corresponds to `process_idx`. Fourier weight volumes with name prefix `weight_fn_prefix` will also be aligned and summed to make a weight sum `weight_sum_fn_prefix` _#_###.em. `tomo_row` describes

which row of the motl file is used to determine the correct tomogram fourier weight file. `iclass` describes which class outside of one is included in the averaging.

The difference between this function and the other version of subtom_parallel_sums is that this function creates a number of subsets of the particle and weight sum subsets, so that smaller and smaller populations of data are summed, and these subsets can then be used to estimate the B-Factor of the structure.

57.4.1 Example

```
subtom_parallel_sums_bfactor(...
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ref_fn_prefix', 'ref/ref', ...
    'ptcl_fn_prefix', 'subtomograms/subtomo', ...
    'weight_fn_prefix', 'otherinputs/ampspec', ...
    'weight_sum_fn_prefix, 'otherinputs/wei', ...
    'iteration', 1, ...
    'tomo_row', 7, ...
    'iclass', 0, ...
    'num_avg_batch', 1, ...
    'process_idx', 1)
```

57.4.2 See Also

- *subtom_maskcorrected_FSC_bfactor*
- *subtom_weighted_average_bfactor*

57.5 subtom_weighted_average_bfactor

Joins and weights subsets of average subsets.

```
subtom_weighted_average_bfactor(
    'all_motl_fn_prefix', all_motl_fn_prefix ('combinedmotl/allmotl'),
    'ref_fn_prefix', ref_fn_prefix ('ref/ref'),
    'weight_sum_fn_prefix', weight_sum_fn_prefix ('otherinputs/wei'),
    'iteration', iteration (1),
    'iclass', iclass (0),
    'num_avg_batch', num_avg_batch (1))
```

Takes the `num_avg_batch` parallel sum subsets with the name prefix `ref_fn_prefix`, the all_motl file with name prefix `motl_fn_prefix` and weight volume subsets with the name prefix `weight_sum_fn_prefix` to generate the final average, which should then be used as the reference for iteration number `iteration`. `iclass` describes which class outside of one is included in the final average and is used to correctly scale the average and weights.

The difference between this function and the other version of subtom_weighted_average is that this function expects there to be a number of subsets of the average subsets, so that smaller and smaller populations of data are averaged, and these subsets can then be used to estimate the B-Factor of the structure.

57.5.1 Example

```
subtom_weighted_average_bfactor(...
    'all_motl_fn_prefix', 'combinedmotl/allmotl', ...
    'ref_fn_prefix', './ref/ref', ...
    'weight_sum_fn_prefix', 'otherinputs/wei', ...
    'iteration', 1, ...
    'iclass', 0, ...
    'num_avg_batch', 1)
```

57.5.2 See Also

- *subtom_maskcorrected_FSC_bfactor*
- *subtom_parallel_sums_bfactor*

57.6 Indices and tables

- genindex
- modindex
- search

CHAPTER

FIFTYEIGHT

INDICES AND TABLES

- genindex
- modindex
- search