Documentation for STOPGAP 0.7.3

William Wan
Vanderbilt University
01.2024

General Description 4
Overall Workflow 4
- Particle picking 5
- Subtomogram averaging 5
- Classification 5

Installing and running STOPGAP 6
- Requirements 6
- Setting paths 6
- Executables 7

Running STOPGAP 7
- Parsers 7
- Run script 8
- Local Copying 9

Settings Overrides 10
Lists 10
- Motivelists 10
- Wedgelists 12

Subtomogram averaging 14
- Directory structure 15
- Basic parameters 15
- Advanced parameters 16
- Full parameters 17
- Settings 18
- Subtomo modes 19
- Parsing iterations 20
- Job parts and completion 20
- Halfsets 21
- References 22
- Reference lists 22
- Alignment masks 23
- Cross-correlation mask 23
- Fourier Shell Correlations 23
- Angular search 24
- Cone search 24
- Arbitrary Euler angles 26
- Subtomo filtering 27
- Cosine weighting 27
- Score-based weighting 28
- Custom filters 28
- Laplacian operator 29
- Scoring functions 29
- Fast local correlation function 29
- Pearson Correlation 30
- Spectral analysis 31
- Search mode 32
- Subset processing 32
- Simulated annealing 33

Processing packets 34
Subtomogram extraction 34
Folder options 35
File options 35
Extraction parameters 36
Particle extraction 36
Subtomogram formats 37
Template Matching 37
Directory structure 37
Parameters 38
Settings 39
Tomolist 39
Template list 40
Goals of template matching 40
Preparing suitable templates 40
Alignment masks 41
Tomogram masking 41
Angle lists 42
Bandpass filtering 42
Noise flattening 42
Generating motivelists 43
Principle Component Analysis 43
Pre-rotating volumes 45
Multi-scale filtering 45
Performing PCA using the CC-matrix 45
Performing PCA using a covariance matrix 46
Classification by K-means clustering 46
Multi-reference Classification 47
Goals of Classification 47
Number of classes 48
Motivelists for classification 48
Randomly generating references 49
Halfsets 50
Initialization bias and premature convergence 50
Monitoring classification 51
Reproducibility 51

General Description

STOPGAP is a subtomogram averaging package written in MATLAB, and designed for parallelization across CPU clusters. The number of tasks that STOPGAP can perform is increasing over time, but the overall goal is the development of a stable, cohesive package for image processing in subtomogram averaging.

While STOPGAP is meant to be easy to setup and run, it is NOT meant to be simple. Subtomogram averaging problems can be widely variable, and as such it is often important to adjust parameters for specific problems. As such, STOPGAP allows users to adjust many parameters for each task; for example, subtomogram alignment and averaging alone has over 70 adjustable parameters. However, many of these parameters have reasonably defined default values, so the number of parameters that MUST be defined is much lower.

As its name implies, STOPGAP is a rough solution to addressing current processing needs. STOPGAP is not an acronym but is in all caps: it should be shouted.

Overall Workflow

The overall processing workflow for cryo-electron tomography (cryo-ET) and subtomogram averaging includes a number of steps outside the scope of STOPGAP. Steps before STOPGAP include pre-processing of the raw data, initial tomogram alignment, and tomogram reconstruction. We also develop TOMOMAN, a package for managing pre-processing workflow. For the highest resolution possible, data should be collected using the "Hagen" tilt-scheme. Pre-processing should include exposure filtering and reconstruction should be performed with 3D-CTF correction. Post-processing steps after STOPGAP may include local resolution estimation and filtering, and model fitting.

Particle picking

The first task STOPGAP can be used for is particle picking. Particle picking depends on the types of specimen, but can be divided into two main types: discreet particles (e.g. globular cytosolic proteins) and assemblies (e.g. spherical viruses or helical proteins). For discreet particles, we typically use template matching, as these particles are too small or randomly distributed to be efficiently picked manually. However, template matching is substantially more expensive than other approaches. For assemblies, we can use functions in the STOPGAP toolbox to generate starting positions on spheres, helices, or surfaces. This restricts the area of the tomographic volume, and we can also generate rough starting angles from the assembly geometry, restricting angular space; this results in much more efficient particle picking than template matching.

Details on performing template matching are described in its own section.

Subtomogram averaging

After particle picking, STOPGAP is then used for subtomogram alignment and averaging. This is the computational core of STOPGAP, and as such, has the most tunable parameters. However, structures can be determined using a small subset of these options. Subtomogram averaging is described in detail in its own section.

Classification

Classification in subtomogram averaging is typically performed after rough to intermediate alignment. This is particularly important for principle component analysis (PCA) based classification methods , which require the particles to be aligned in order to compare them. Multireference alignment based approaches can be used during initial alignment, though this can have substantial computational costs.

PCA in STOPGAP is largely deprecated, and our general recommendation is to use multireference alignment based approaches.

Installing and running STOPGAP

STOPGAP is compiled MATLAB code, and as such doesn't exactly need installation, but rather needs some support software and correct setting of paths.

Requirements

To run STOPGAP, you need the correct MATLAB or MATLAB MCR installed. The supplied pre-compiled executables are compiled in MATLAB 2020b.

To run STOPGAP in parallel, either locally or on a cluster, you need OpenMPI or SLURM installed, respectively. STOPGAP does not use any of the communication functionality of MPI, but instead each core uses the environmental variables to determine its rank or processor ID. Launching STOPGAP through OpenMPI's mpirun or SLURM's srun will set the proper environmental variables.

Setting paths

In the .tar.gz file, the various bash, executable, and source files for STOPGAP are contained in the exec/ folder.

The lib/ subfolder contains the MATLAB executables and the configuration scripts (stopgap_config_local.sh and stopgap_config_slurm.sh) used to source the MATLAB libraries. The matlabRoot variables in the stopgap config files need to be updated to the appropriate MCR paths for your systems. The stopgap_prepare_mcr.sh is used for generating a temporary MCR folder while the stopgap_prepare_local_temp.sh is used to generate a local temporary directory if local copying is enabled. Both assume that the /tmp/ directory is available as a local temporary directory; this can be modified as needed.

The MATLAB executables require the MCR libraries to be loaded, and as such are not directly executable. The bin/ subfolder in the modules/ folder contains bash scripts used to run the MATLAB executables. These should require no editing.

In order to set the paths for the executables, copy and rename the exec/ folder to a stable location. Set an environmental variable $STOPGAPHOME to this folder.

Executables

STOPGAP executables are typically not run directly, but are instead run through bash scripts contained in the exec/bash/ folder of the .tar.gz file. STOPGAP has three main executable components: the main STOPGAP core, the STOPGAP parser, and the STOPGAP watcher. The functions and usage of these scripts are explained in the “Running STOPGAP” section.

Running STOPGAP

For the most part, STOPGAP is run by first generating a parameter file, then running the parameter file through the STOPGAP core executable. Parameter files are generated using task-specific bash scripts; these scripts are interfaces for the STOPGAP parser, which checks the inputs and generates properly formatted parameter files. The run script is used to generate a submission script for your particular cluster and submit the job to run on a target parameter file. Additionally, if your cluster allows for executables on the submission node, the run file will also launch the STOPGAP watcher, which returns progress information on the job.

Parsers

Currently, STOPGAP supports four tasks:

Filename	Task
stopgap_subtomo_parser.sh	Subtomogram alignment and averaging
stopgap_extract_parser.sh	Subtomogram extraction
stopgap_tm_parser.sh	Template matching
stopgap_pca_parser.sh	Classification by Principle Component Analysis (PCA) (Deprecated)

Parameters for each of these will be described in their own sections.

Parsers output parameter files as .star files. When parameter files with the same name are detected, the parser appends the parameter files. Parameter files have one or more fields that contain information about job completion; STOPGAP will iterate through a parameter file until all jobs are completed.

Run script

The run script is a bash script this is mainly used to generate a submission script and submit the job to the cluster. As such, this is completely dependent on the cluster platform, and is meant more to serve as an example.

The main thing to consider is how the MATLAB executable is run:

```
stopgap_mpi_slurm.sh rootdir paramfilename n_cores
copy_local run_type
```

This needs to be in there in some form, whether you are launching with mpiexec, mpirun, srun or something else.

In the supplied run script (run_stopgap.sh), the script does not actually run the job, but instead generates a "submit_cmd" variable and for SLURM jobs, generates a submission script "submit_stopgap". The script then runs the STOPGAP watcher and passes the submit_cmd as an input variable in single quotes. In this way, the watcher launches, checks the job, and submits to the cluster. If you cannot run executables on the cluster, edit this to simply submit the job. The watcher can also be run on a different computer, as long as the directories are accessible. Job progress from each node is also written into the log files.

Local Copying

Depending on your cluster setup, it may be advantageous to copy files like subtomograms to the node's local storage rather than having every core access the networked storage. Local copying works by first identifying the number of nodes and the number of cores on each node. The first core on each node will then determine the portion of the dataset it will process and copy that locally. Once all files have been copied, the processing will begin.

Local copying is set in the run_stopgap.sh using the copy_local parameter. Setting 0 disables it, 1 enables a temporary copy that will be cleared at the end of the run, and 2 enables a persistent copy. A persistent copy can be useful if the copying is slow or the job contains a lot of data, but this assumes that subsequent jobs will land on the same nodes, so a reservation may be required.

The local directory is determined by the stopgap_prepare_local_temp.sh script and is by default to the "/tmp/" folder with subfolder "stopgap_u{$UID}/". The default temporary folder can be adjusted by changing the "LOCAL_TEMP_ROOT" environmental variable in the stopgap_mpi_slurm.sh and stopgap_watcher.sh scripts. By default, local copying is performed using tar, but can be changed to rsync using the "copy_function" settings override (see below).

For extraction, where you are probably using a small number of cores, each core reads from the network storage but writes subtomos to the local storage. After extraction is complete, the copy_cores copies the subtomos to remote storage.

For alignment and averaging, the motivelist is read in and split between the number of processing cores. Then the copy_core figures out which other cores are on its node, what subtomograms they will processed, and copies those.

Settings Overrides

While parameter files are used to adjust settings for each task, there are often other low level settings that can be also be adjusted. These range from default directory names, volume extensions, and whether to write out raw intermediate files.

Overriding the default settings is performed by adding a plain-text settings file. These can be on a per-task basis by adding a settings file to the processing rootdir, or on a global basis by adding a global_settings.txt file to the exec/lib/ subfolder of the STOPGAP installation.

Settings for each task are described below in the corresponding sections.

Lists

There are a number of different lists used in STOPGAP, with the most used ones being motivelists and wedgelists. The motivelist contains information on each subtomogram including its source tomogram, and it's current orientation parameters; about half this information can be considered constant while the other half changes after each iteration of refinement. Wedgelists contain information about each tomogram; at a minimum they contain the angles of the input tilts, but can contain information about CTF parameters and exposure amounts. All lists in STOPGAP are plain-text, loop-style .star files, though STOPGAP toolbox functions are available to read and write them into more usable formats in MATLAB.

Motivelists

STOPGAP motivelists are iteratively updated, so their names should be formatted be as follows:

```
[motl_name]_[iteration].star
```

STOPGAP motivelists have a data type of "`stopgap_motivelist`" and contain 16 fields.

Field name	**Data Type**	**Description**
motl_idx	int32	Motivelist index; each number identifies a unique entry in the motivelist
tomo_num	int32	Tomogram number
object	int32	Object number
subtomo_num	int32	Subtomogram number
halfset	char	Halfset. Must be A or B.
orig_x	single	Original X coordinate in tomogram.
orig_y	single	Original Y coordinate in tomogram.
orig_z	single	Original Z coordinate in tomogram.
score	single	Score.
x_shift	single	X-shift
y_shift	single	Y-shift
z_shift	single	Z-shift
phi	single	phi angle
psi	single	psi angle
the	single	theta angle
class	int32	class number

The first 8 fields deal with identifying the particle and can generally be considered immutable. However, since the coordinates in the original tomograms are in voxels, they will need to change during rescaling; i.e. when the data is unbinned. The final 8 fields change with each iteration of subtomogram alignment.

NOTE: The shifts and Euler angles are those determined during alignment, and thus describe the reorientation of the reference that matches the subtomogram. The order of operations is to rotate phi-theta-psi around Z-X-Z, and then shift.

Motivelists have three different types: 1, singleref; 2, multiclass; and 3, multi-entry. Singleref motivelists have a single class. Multiclass have different classes, but one entry per class. Multi-entry have n classes with n entries for each subtomogram; theses are exclusively used for multireference alignments and provide storage of local refinement parameters for each reference.

For singleref and multiclass motivelists, each line in the motivelist must have a unique motl_idx. However, each subtomogram does not have to have a single motl_idx. For example, if you are performing an asymmetric refinement of a C2 symmetry particle, you can have two entries for each particle, each with different alignment parameters. Each entry would have a distinct motl_idx, but can share the same subtomo_num.

Motivelists can be read and written in two different ways in MATLAB, using sg_motl_read.m, sg_motl_write.m for read-type 1 and sg_motl_read2.m and sg_motl_write2.m for read-type 2. Read-type 1 formats the motivelist into a struct array that has length equal to the number of entries. Read-type 1 is a 1x1 struct who's fields have lengths equal to the number of entries. Type 1 is generally more intuitive to work with, but because of how MATLAB organizes struct arrays, uses memory many times the actual size of the data. Type 2 is less intuitive but much less memory intensive.

Wedgelists

Wedgelist contain information about each tomogram; in particular, they include information on the tomographic data collection such as the tilt angle of each projection. Since the number of tilts per tomogram can be an arbitrary number, each tomogram has n lines, where n is the number of tilts.

Wedgelist must have the following required fields:

Field Name	**Description**
tomo_num	Tomogram number. Used to match motivelist.
pixelsize	Unbinned pixelsize in Angstroms. Used in tandem with binning factors.
tomo_x	Size of unbinned tomogram in X.
tomo_y	Size of unbinned tomogram in Y.
tomo_z	Size of unbinned tomogram in Z.
tilt_angle	Tilt angle in degrees.

In order to calculate local CTF filters, the following parameters are also required:

Field Name	**Description**
z_shift	Z-shift applied during tomogram reconstruction. Required for proper local defocus calculation.
defocus	Mean defocus of tilt. Supersedes other defocus parameters.
defocus1	Defocus along axis 1
defocus2	Defocus along axis 2
astig_ang	Astigmatism angle
pshift	Phase shift in degrees
voltage	Voltage in KeV
amp_contrast	Amplitude contrast. Must match parameter used for defocus estimation.
cs	Spherical aberration.

Defocus values can be given as a mean defocus or as two defocus values and an astigmatism angle. However, STOPGAP does not currently support astigmatism, so the two defocus values will be averaged to generated a mean defocus value. Z-shift is a parameter used during tomogram reconstruction; it describes the tilting geometry and should be provided to proper local defocus estimation.

If tomograms were generated from exposure filtered data, the exposure filter should also be applied duing STOPGAP processing. This requires an exposure entry in the motivelist that is in units of `e/Å^2`.

Wedgelist can be read and written into MATLAB using `sg_wedgelist_read.m` and `sg_wedgelist_write.m`. By default wedgelist are read in a 'compact' mode, where each tomogram is an entry in a struct array. Reading with 'full' mode provides a struct array with entries the same as those stored in the .star file.

A helpful function for generating wedgelist is sg_wedgelist_add_entry.m which attempts to simplify to generating of a wedgelist. This also serves as an example for writing scripts based around your own data management.

Subtomogram averaging

Subtomogram averaging in STOPGAP consists of 3 steps: alignment, parallel averaging, and final averaging. During alignment, the subtomograms are aligned to a given input reference. After the subtomograms are aligned, new averages are generated using two steps: parallel averaging, which generate partial averages in parallel, and final averaging, which is performed on a single core for each class to compile the partial averages into the final average.

STOPGAP subtomogram averaging can be run in a number of modes set, which is defined by two types and three subtypes. The two types are 'ali' for alignment and 'avg' for averaging; ali modes perform all three averaging steps while avg modes only perform the two averaging steps. The subtypes are 'singleref', 'multiref', 'multiclass' for single reference, multi-reference, and mutli-class. Single reference jobs ignore all class information and align and average one reference. Multi-reference and multi-class both make use of the class data. Multi-reference aligns each subtomogram with all references and assigns classes by best score. Multi-class treats each class separately; classes are only aligned with and averaged into their own classes.

Directory structure

The default directories are as follows:

Directory	Parameter Name	Default Directory
Root	rootdir	---
Temporary	tempdir	temp/
Communication	commdir	comm/
Raw data	rawdir	raw/
References	refdir	ref/
Masks	maskdir	masks/
Lists	listdir	lists/
FSC	fscdir	fsc/
Subtomograms	subtomodir	subtomograms/
Metadata	metadir	meta/
Spectra	specdir	spec/

Basic parameters

Parameter	Description
subtomo_mode	Subtomogram averaging mode. [type]_[subtype] (see above).
startidx	Starting index for job.
iteration	Number of iterations to add.
binning	Binning factor.
motl_name	Root name of motivelist. File names take the format [motl_name]_[iteration].star. Files are in listdir.
ref_name	Root name of references or name of reference list. For file naming conventions, see below.
mask_name	Full name of alignment/FSC mask. File is in maskdir.

ccmask_name	Full name of cross-correlation mask. File is in maskdir.
wedgelist_name	Full name of wedgelist. File is in listdir.
subtomo_name	Root names of subtomograms. File names take the format [subtomo_name]_[subtomo_num].[vol_ext]. Files are in subtomodir.
angincr	Out-of-plane angular increment in degrees.
angiter	Out-of-plane angular iterations.
phi_angincr	In-plane angular increments in degrees.
phi_angiter	In-plane angular iterations.
lp_rad	Low-pass radius in Fourier pixels.
hp_rad	High-pass radius in Fourier pixels.
symmetry	Reference symmetry in Schoenflies notation.
score_thresh	Score threshold for averaging.

Advanced parameters

Parameter	**Description**
search_mode	Algorithm to use for searching angles.
lp_sigma	Low-pass filter dropoff in Fourier pixels.
hp_sigma	High-pass filter dropoff in Fourier pixels.
calc_exp	Calculate exposure filter. (1 = yes, 0 = no)
calc_ctf	Calculate CTF filter. (1 = yes, 0 = no)
cos_weight	Apply cosine weighitng. (1 = yes, 0 = no)
score_weight	Apply score-based weighting. (1 = yes, 0 = no)
apply_laplacian	Apply laplacian function to volumes. (1 = yes, 0 = no)
subset	Percentage of data to use for alignment. To disable, set to 100.
avg_mode	Averaging mode for subset processing. ‘full’ to average full dataset, ‘partial’ to average only aligned subset.

ignore_halfsets	Ignore halfset data in motivelist. (1 = yes, 0 = no)
temperature	Temperature factor for simulated annealing. To disable, set to 0.

Full parameters

Parameter	**Description**
ali_reffilter_name	Alignment reference filter name.
ali_particlefilter_name	Alignment particle filter name.
avg_reffilter_name	Average reference filter name.
avg_particlefilter_name	Average particle filter name.
reffiltertype	Reference filter type.
particlefiltertype	Particle filter type.
specdir	Spectral directory.
ps_name	Power spectrum name. Files are stored in spectral directory.
amp_name	Amplitude spectrum name. Files are stored in spectral directory.
spec_mask	Spectral mask name. File is in mask directory.
search_type	Angular search type.

euler_axes	Axes for arbitrary Euler angle search.
euler_1_incr	Arbitrary Euler angle 1 increment.
euler_1_iter	Arbitrary Euler angle 1 iterations.
euler_2_incr	Arbitrary Euler angle 2 increment.
euler_2_iter	Arbitrary Euler angle 2 iterations.
euler_3_incr	Arbitrary Euler angle 3 increment.
euler_3_iter	Arbitrary Euler angle 3 iterations.
cone_search_type	Cone search type.
scoring_fcn	Scoring function
rot_mode	Rotation interpolation mode (‘linear’, or ‘cubic’)
fthresh	Fourier dynamic range threshold.

Settings

For per-task settings, place them into a file in the rootdir name “subtomo_settings.txt”.

Setting	**Default**	**Description**
wait_time	5	Refresh time for parallelization checks (in seconds).
tempdir	temp	Default temporary directory
commdir	comm	Default communication directory
rawdir	raw	Default raw directory
refdir	ref	Default reference directory
maskdir	masks	Default mask directory
listdir	lists	Default list directory
fscdir	fsc	Default FSC directory
subtomodir	subtomograms	Default subtomogram directory
metadir	meta	Default metadata directory
specdir	spec	Default spectral directory
localtempdir	/tmp/	Default local temporary directory

subtomo_digits	1	Default number of digits in subtomogram number.
vol_ext	.mrc	Default volume extension/format
packets_per_core	5	Default number of packets per core
n_tries	10	Number of tries in reading a file
counter_pct	5	Percentage of job completion for progress output.
calc_ctf	true	Calculate CTF by default.
calc_exp	true	Calcualte exposure filtering by default.
fourier_crop	true	Fourier crop prior to processing
fthresh	800	Default Fourier dynamic range.
fsc_fourier_cutoff	5	Default cutoff for phase-randomized FSC in Fourier pixels.
fsc_n_repeats	5	Default number of repeats for phase-randomized FSC.
fsc_thresh_single	0.5	Default FSC threshold for low-pass filtering without halfsets
fsc_thresh_split	0.143	Default FSC threshold for low-pass filtering with halfsets
write_raw	false	Write out raw intermediate files

Subtomo modes

STOPGAP subtomo can be run in a number of different modes, which is defined by the subtomo_mode parameter. The subtomo_mode has a type and subtype; the mode takes the form [type]_[subtype], e.g. ali_singleref.

Types are ali and avg, which are alignment and average, respectively. Alignment is essentially a full subtomogram averaging run including alignment and the generation of new references, while an averaging run uses a motivelist to generate an average.

Subtypes are singleref, multiref, multiclass for single-reference, multi-reference, and multi-class jobs, respectively. In single-reference jobs, the motivelist class field is ignored, and all subtomograms are aligned with and averaged into single references. In multi-reference jobs, each subtomogram is aligned all references and averaged into the highest-scoring class. In multi-class jobs, each subtomogram is aligned and averaged into it's own class; there is no interaction between classes.

Parsing iterations

When using the STOPGAP subtomo parser, all jobs require a starting index (startidx) and alignment jobs require a number of iterations (iterations).

The starting index denotes what iteration number the parsed jobs will begin at. For a completely new alignment run, indices should start at 1. For continuing a job this should be set at the last iteration index. When aligning, the new files generated at the end of an iteration will be incremented from the starting index. When averaging, the output files will match the starting index.

The number of iterations will tell the parser how many interations you wish to perform with the same parameters. It will then write out the same parameters, but with increasing iteration numbers. The exception to this is when performing simulated annealing (see below). Number of iterations is ignored when parsing an averaging job.

Job parts and completion

The first three fields of the STOPGAP subtomo parameter file are completed_ali, completed_p_avg, and complete_f_avg for completed alignment, completed parallel average, and completed final average. These mark the completion of the three stages of a subtomogram averaging run.

Alignment is the alignment of the individual subtomograms, which are evenly split across all cores. The output of alignment is a new motivelist.

For computational efficiency, averaging is performed in two phases: parallel and final. Parallel averaging makes use of a number of cores to rotate and average individual subtomograms; each core writes out partial average and partial Fourier weighting filter (see below). The number of cores used is the square root of the number of subtomograms; if this is larger than the number of cores, then all cores are used.

Final averaging is performed in parallel for each class. For a single class, one core is used. For multiple classes, the number of cores used is equal to the number of classes; if there are more classes, they are evenly distributed across all cores.

A STOPGAP subtomo job is considered complete only when the complete_f_avg field is completed.

Halfsets

STOPGAP always processes volumes in halfsets. This is because references are figure-of-merit weighted, which requires halfsets for the calculation of FSCs. Halfsets are defined in the halfset field of the motivelist and are treated independently during processing; subtomograms are aligned with and averaged into their own halfsets. As such, it is good practice to maintain equal numbers of subtomograms in each halfset.

There is no requirement to split halfsets. In this case, STOPGAP will automatically randomize the halfsets at each iteration prior to subtomogram averaging, yielding two references which can be used to calculate FSCs. During alignment, these halfsets will be averaged into a single reference.

When the ignore_halfsets parameter in engaged, the two halfset references are averaged prior to alignment. The stored halfset parameters are still used to generate the halfset averages.

References

Reference files are required to perform subtomogram alignment; references must always be given as A and B halfsets. Naming conventions depend one singlref modes, which have no class number, or multiref/multiclass modes.

Singleref name: [refname]_[halfset]_[iteration].[vol_ext].
Multiref/multiclass name: [refname]_[halfset]_[iteration]_[class].[vol_ext].

During averaging, an averaged reference is also generated, without a halfset identifier. This is a figure-of-merit weighted, unsymmetrized, average reference. This is allows for quick visualization of the average, as the halfset averages are not low-pass filtered or figure-of-merit weighted and can be very noisy. This average is not required as an input, and is not used anywhere during processing.

Reference lists

Instead of assigning a root name to the ref_name, a .star reference list can be given instead. The reference list assigned a list of references and allows for the use of different names, masks, and symmetries for each reference. The formatting of the reference list is:

Field Name	**Description**
ref_name	Root name of reference.
class	Class number.
mask_name	Alignment mask name.
symmetry	Symmetry

Reference lists can be written using the sg_reflist_add_entry.m function and read and written using sg_reflist_read.m and sg_reflist_write.m, respectively.

Alignment masks

Alignment masks are applied to the references prior to scoring to mask in the regions of interest, and such must be the same size. Alignment masks are also used to calculate FSCs and perform figure-of-merit weighting prior to alignment. Technically, hard masks can be used, but this may produce mask artifacts in the alignment, so it is preferable to use a mask with mollified edges.

Cross-correlation mask

The cross-correlation mask is used to restrict the shifts when using cross-correlation based scoring functions like the fast local correlation fuction (FLCF). It is ignored with other scoring functions. The cross-correlation mask is applied to the cross-correlation maps; restricting where peaks appear.

The cross-correlation mask will restrict shifts to vectors within the boundaries of the mask; it is defined from the center of the box (boxsize/2 +1 voxels), and it's angles are with respect to the reference frame. For example, a centered sphere with radius of 5 voxels will allow 5 voxel shifts in all directions. A centered cylinder with radius 3 along the XY-plane and height of 10 along the Z-axis will allow for shift vectors that are up to 3 voxels along the XY-plane and up to 5 voxels along the Z-axis. If this cylinder is shifted up by 2 voxels, it will allow a positive Z-shift of 7 voxels and negative Z-shift of 3 voxels.

Fourier Shell Correlations

STOPGAP subtomo calculates FSCs when loading reference pairs prior to alignment and also when generating averages. STOPGAP calculates mask-corrected FSCs using the alignment mask; by default it uses a randomization cutoff at 5 pixels and 5 repeats.

The FSCs generated prior to alignment are used to figure-of-merit the references before alignment. The FSCs generated during averaging are used to figure-of-merit weight the averaged reference and plots are written out as a .pdf file into the fscdir.

NOTE: This FSC can be used to generally monitor the overall progress of the alignment, but depending on your alignment mask, may not be an accurate assessment of the resolution.

Angular search

The convention for storing orientations in STOPGAP subtomo follows TOM's Euler standard. This is where the angles phi-theta-psi describe rotations about the Z-X-Z axes, respectively. The stored angles describe the rotation from the reference frame to the subtomogram frame.

The default search mode in STOPGAP subtomo is the cone search, which is also used in a number of other packages. However, STOPGAP subtomo also allows the use of arbitrary Euler triples to define search orientations.

The angular search mode is largely independent of stored Euler angles. One exception is the phi angle, which is the first rotation about the Z-axis, or in cone-search terms, the in-plane angle. Changes to this angle can be directly made, resulting in in-plane rotations of the reference.

Cone search

The cone search is defined by two sets of parameters: the cone angular increment and iterations, and the in-plane (phi) angle increment and iterations. The cone angular parameters define what is called the "cone search". Each increment describes a tilt away from the top of the orientational sphere; the search is then swept around the Z-axis, which traces the edge of a cone on the orientational sphere. Iteration of the angular increment produces wider cones.

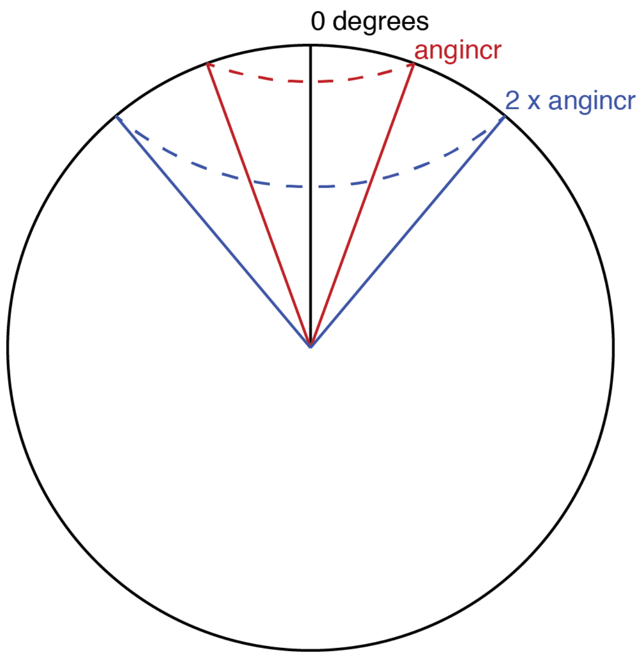
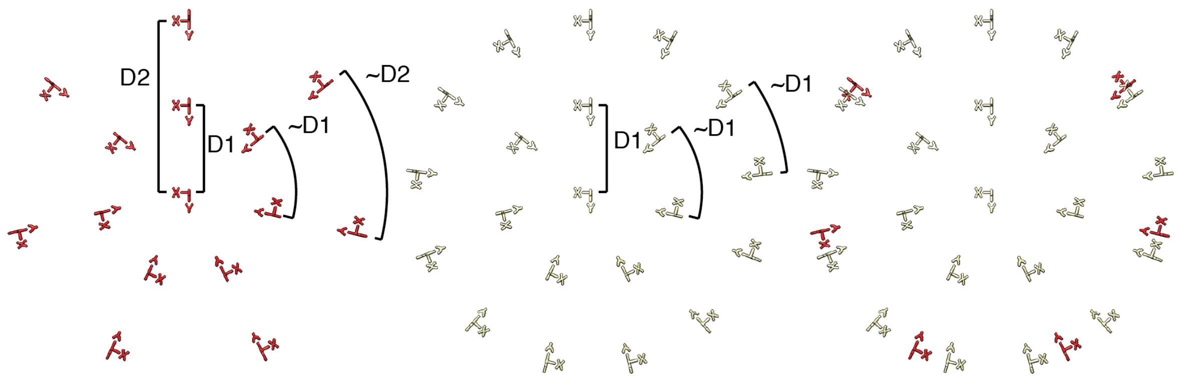


Illustration of the cone search. Black line indicates no tilting, red and blue lines show cones formed from two consecutive tilt iterations.

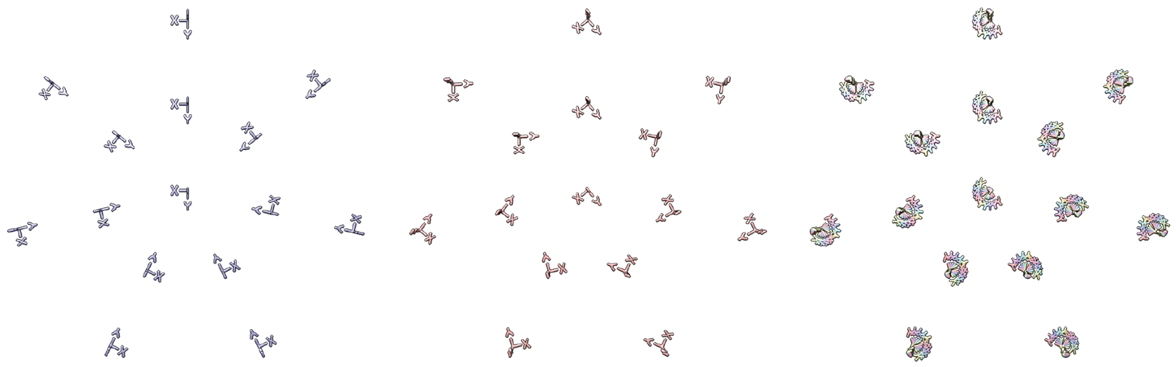
For even sampling in orientational space, the arc-spacing of the tilt is the distance used for the search around the cone edge. This can be done in coarse method, where the arc-distance around the cone edges are equivalent to the total tilt from the top of the sphere (angincr * angiter), or in a complete method where the arc-distances are all related to the angincr parameter.



Angular search positions, viewed from the top. Left: Coarse angular search; arc distances around the cone edge are the same as the total tilt angle. Center: Complete angular search, where only the arc-distance used is related to the *angular increment. Right: overlay of the coarse and complete; the complete method produces more orientations at greater tilt increments.*

In general, we find the coarse search converges within a similar number of iterations to the full search, but with substantially fewer orientations to search. This is particularly true when more angular iterations or smaller angular increments are used.

The cone search is insufficient to describe 3D rotations; an additional in-plane angle is required. This in-plane angle is applied before the cone search, and in the TOM Euler notation (Z-X-Z rotation), this first in-plane angle is the phi angle.


Particle orientations with various in-plane angles. Left: 0 degree in-plane rotation. Center: 40 degree in-plane rotation. Right: Overlap of -40 – 40 degrees in 20 degree steps.

To compose angles, each phi angle produces a set of cone angles, making phi largely independent of the other angles.

Arbitrary Euler angles

Coarse cone angles provide a clever way of efficiently sampling a broad range of orientational space. However, if there are specific constrains on the geometry of your specimen, many cone angles may be impossible. In this case, it may be more efficient to sample around an arbitrary set of Euler axes. In this case, you need to input the three Euler angles you wish to rotate about in the euler_axes parameter. Euler triplets can be any axes, but the second angle must be different from the first; the third can be a repeat of the first. By setting the corresponding angle increments and iterations, the search is essentially performed on a polar grid. It is important to keep in mind that this grid may not be well distributed in orientational space, and is better suited for very particular geometric configurations.

Subtomo filtering

STOPGAP uses a number of different filters for various purposes. Below are the filters specific to STOPGAP subtomo. Other more general filters including bandpass, wedge, CTF, and exposure filtering are described in the main filtering section.

Cosine weighting

The cosine filter weights Fourier slices by the cosine of their tilt angle. This is essentially a non-linear signal dampening of the higher tilts; this is meant to take into account that the higher tilt data is less reliable due to thicker mean-free path. This assumption may not be true if data was collected using cosine-distributed exposure times. This requires no extra parameters other than the tilt angles in the wedgelist.

The setting is the exponent of the cosine weighting. For example, 0 sets no weighitng, 1 sets a weighting equal to cosine of the tilt angle, and a setting of 2 weights it by the square of the cosine. For a 60 degree tilt, a setting of 1 dampens by 0.5 and a setting of 2 dampens by 0.25.

Cosine filtering is performed prior to alignment and during averaging.

Score-based weighting

Score-based weighting applies a b-factor type filter to subtomograms during averaging; the strength of the filter is related to its score, relative to all other subtomograms within the same tomogram. This is relative for each tomogram, as scores can depend on defocus and sample thickness.

The heuristic function STOPGAP uses is:

$$W = \exp(w_f * (S_{max} - S) * f^2)$$

where W is the weighting filter, w_f is a weighitng constant, S_{max} is the highest score in the tomogram, S is the score of a subtomogram, and f is spatial frequency. w_f is calculated so that the pass through of the filter at unbinned Nyquist for the lowest scoring subtomogram is equal to the score-based weighting factor.

Score-based weighting is only performed during averaging and is applied to the halfset references. This weighting can improve the visualization of high-resolution details in the map.

Custom filters

Custom filters generated outside of STOPGAP subtomo can also be used during alignment and averaging. Filters can be applied during alignment or averaging and applied to the reference or filter. For example, a filter to be applied to the reference during alignment is set by the ali_reffilter_name parameter. In addition, the filters can be defined per tomogram or subtomogram; this is defined by the particlefiltertype parameter. For a per-tomogram filter, the final name is [localdir]/[filter_name]_[tomo_num].[vol_ext], while a subtomogram filter is [localdir]/[filter_name]_[subtomo_num].[vol_ext].

Laplacian operator

The Laplacian operator is essentially the second derivative or gradient of a 3D volume. It is often used for edge enhancement, but is also used in molecular docking, which is essentially the same task as subtomogram alignment.

The Laplacian's utility in subtomogram averaging is a bit subtler than edge enhancement. It is effectively a method for normalization, as all values are between-1 and 1. In practice, this normalizes thickness changes through the ice that typically produce fluctuations in low-resolution background. This minimizes the need for high-pass filtering or pre-whitening. When apply_laplacian is engaged, the references are operated on prior to alignment.

NOTE: For final high-resolution refinement (e.g. better than 5 A), the Laplacian operated references tend not to refine as well as high-pass filtered real-space references. At lower resolutions, they typically perform as well or better when high-pass filtering is not tweaked.

Scoring functions

STOPGAP subtomo currently has two difference scoring functions: the fast local correlation function (FLCF) and a Pearson correlation function. In general, the FLCF is probably the best choice, given its relative accuracy and computational efficiency. However, there are some circumstances where the Pearson correlation may give better results. Each scoring function is described in detail below.

Fast local correlation function

The fast local correlation function (FLCF) was first described by Roseman (2003) Ultrmicroscopy 94: 225-236. STOPGAP uses a 3D implementation based off the one in DYNAMO. Briefly, the FLCF performs local normalization with respect to the alignment mask, improving the SNR of the resultant correlation map. Each voxel of correlation map shows the cross correlation (CC) score between the reference and subtomogram with a given shift from the center of the box. The center of a box in STOPGAP is always defined as *floor(boxsize/2)+1*. By finding the voxel with the highest CC, we find the maximum score the corresponding shift. In general, we use a subpixel peak-finding algorithm to interpolate the subpixel shift and CC score. Because shifts are never explicitly used between iterations, shifts are always defined with respect to the extraction positions.

In order to control maximum shifts, we apply a cross-correlation mask (ccmask). The ccmask is rotated by the current search angle and applied to the CC map; everything outside the mask is zero, preventing detection from the peak-finding algorithm. The ccmask should be binary, and the shape is defined with respect to the reference coordinate frame. Simple shapes can be used: spheres represent isotropic shifts, cylinders allow for greater or smaller shifts along one direction. More complicated shapes can be used if you have prior knowledge about spatial distributions. CC masks are always with respect to the original, unshifted extraction positions, so they represent absolute maximum shifts from the extraction position.

Pearson Correlation

The Pearson correlation function calculates a real-space Pearson correlation, and explicitly determines the shifts of each orientation using a maximization routine. The maximization routine does not allow for constraint of shifts, precluding the need for a ccmask. This unconstrained refinement of shifts may be useful if the extraction position is too far off from the true position, but may also make the function more prone to false maxima.

Because the shifts are explicitly determined, the Pearson correlation can be extremely slow. As such, it may be more useful as a function for refining the shifts of previously determined angular orientations. **This function is largely deprecated.**

Spectral analysis

Spectral analysis of the subtomogram and reference can be a useful way of ensuring that the reference is not overfitted, and that the structural information there is present in the raw data. This is particularly useful for averages of assemblies, where characteristic Fourier spectra should be present in the subtomograms prior to averaging the of structure factors.

STOPGAP subtomo can calculate two types of spectra per iteration: the subtomogram-derived power spectrum, and the reference-derived amplitude spectrum. The subtomogram-derived power spectrum is generated in parallel with the reference; after shifting and rotating the subtomogram the amplitudes are calculated and summed. During final averaging, the partial power spectra are summed and weighted averaged. The reference-derived amplitude spectrum is simply the amplitudes of the reference structure. Both are calculated using a real-space mask, set by the specmask_name parameter.

While these two spectra sound the same, there are subtle points that allow the power spectrum to be used as a reality check for the amplitude spectrum. Mainly, the power spectrum is a sum of amplitudes; this is a completely additive operation and which produces no destructive interference. Therefore it's relatively noisy, but all strong features are there because they are strong in a large population of the subtomograms. In the amplitude spectrum, strong features may be artifacts caused by summing poorly aligned structure factors. Iterative refinement can further amplify these features, leading to overalignment or overfitting. So if a feature is present in the amplitude spectrum but not the power spectrum, it is likely an artifact, while features common to both are likely real features.

Search mode

STOPGAP subtomo currently supports two search modes: the normal hill climbing (HC) or “greedy” search mode, and a stochastic hill climbing (SHC) mode. The normal mode scores all search orientations, as defined by the angular parameters, and takes the highest scoring orientation. This is a “greedy” search because the highest scoring orientation is taken during each iteration.

STOPGAP’s implementation of SHC is based on Hans Elmund’s SIMPLE/PRIME software. Rather than searching all orientations, the previous best orientation is scored, and the remaining orientations are scored in random order. As soon as one scores higher than the previous maximum, it is taken and the search moves to the next subtomogram.

Alignment iterations using SHC is generally faster than the normal search, as fewer orientations are sampled, though as the alignments begin to converge, SHC should require progressively more trials to find a better scoring orientation. In taking a better, but suboptimal, score each round, the SHC algorithm may converge more slowly; this can be taken as an advantage as it may avoid local minima. The slower convergence rate, and subsequent higher number of iterations required, may be offset by the speed of each iteration.

NOTE: SHC is a useful method for refining orientations but should NOT be used in determining particle positions. For example, if you are aligning points on an oversample surface to determine the particle positions, SHC can lead to convergence of incorrect structures and positions. However, when particle positions have been determined and shifts properly restrained, SHC shows fairly consistent convergence towards higher resolution structures.

Subset processing

STOPGAP subtomo can process random subsets of the data at each iteration. Reasons to process subsets of the data include speeding up alignment at intermediate resolutions and to slow convergence of alignment. These behaviors are controlled by two parameters: subset which indicates the subset to process in percent, and avg_mode which indicates whether to generate averages from the entire motivelist or only from the processed subset.

When refining a large dataset, intermediate resolution structures can be determined from subsets of the data; e.g. for a 100,000 subtomogram dataset, 10,000 subtomograms may be sufficient up to 10 Å resolution. In this case, substantial computational time can be saved by aligning only 10,000 subtomograms until the final alignment steps. In this type of approach, a good strategy would be to use a small subset of particles and perform partial averages. In this way, each new reference is generated from completely optimized parameters, and convergence should be rapid. Keep in mind, in such an approach your rotational search needs to stay relatively large as during each iteration a substantial proportion of the subset was not aligned at the previous iteration, meaning the stored orientation may be far from optimal.

If your refinement is prone to false minima, slowing the convergence of the alignment may help the global optimization. In this case, refining subsets of the data and generating full averages can slow convergence. In this approach, the average structure changes more slowly as only a subset of the averaged particles change at each iteration. The randomization of the subset also slows convergence, and when combined with SHC and a large angular search, can probe a substantial amount of orientation space.

Simulated annealing

Another method for slowing convergence and probing orientation space is to use simulated annealing. In STOPGAP, simulated annealing is a performed as a special type of SHC. The previous optimal angle is scored and the remaining search angles are scored in random order, the first better scoring angle is immediately accepted. Poorer scores are randomly accepted with a probability determined by the temperature factor. The temperature factor is a percentage probability; e.g. when the temperature factor is 10, a lower score is accepted with a 10% probability.

To set up simulated annealing, set a temperature factor and desired number of iterations in the parser. The parser will then generate a set of iterations with a linearly descending annealing schedule. For example, when temperature=10 and iterations=10, 10 iterations are generated with temperature factors of 10,9,8,...,1. Other annealing schedules can be manually implemented.

Processing packets

Alignment is performed as a set of "packets", in order to minimize issues related to some cores being significantly slower than others. Rather than splitting the motivelist evenly across a number of cores, it is split into packets. Packets are determined by the packets_per_core setting, which has a default of 5. For example, a 10 core job with 5 packets per core will split the motivelist into 50 parts. Each core will process an initial 5 assigned to it, then it will look for unprocessed packets and start on them. In this way, the total alignment time is not directly constrained to the slowest node. If you find that there are some extremely slow cores, you can increase the packets_per_core setting.

If you are using copy_local, the motivelist is first split between nodes in proportion to the number of cores assigned to each node. From there, they are then split into packets. This is to minimize the number of subtomograms that each node needs to copy.

Subtomogram extraction

Parameters for subtomogram extraction are generated using the stopgap_extract_parser.sh. The required files for subtomogram extraction are a motivelist and the source tomograms. The motivelist contains the position of each subtomogram; these are matched to the tomograms with the tomo_num field.

Tomograms can be supplied using the tomolist_name or tomodir parameter. The tomolist_name is an input plain-text file that contains the tomo_num and the full path and filename to the tomogram. This follows updates to TOMOMAN, where tomo_num values are not pulled from the filenames during acquisition, but are instead arbitrarily assigned. In doing this, there's no clean way to parse a tomogram name from the STOPGAP motivelist, so this new tomolist is required. Newer novaCTF scripts will generate tomograms using names based off the arbitrary raw data names. The sg_extract_make_tomolist function takes care of conversions from TOMOMAN to STOPGAP. Otherwise, the tomodir parameter defines a folder that contains tomograms named [tomo_num].rec.

Folder options

Directory	**Parameter Name**	**Default Directory**
Root	rootdir	---
Temporary	tempdir	temp/
Communication	commdir	comm/
Lists	listdir	lists/
Metadata	metadir	meta/
Subtomograms	subtomodir	subtomograms/

The rootdir is a full path. Other directories are subdirectories of the rootdir.

File options

Parameter	**Description**
motl_name	Full name of motivelist. File is within listdir.
wedgelist_name	Full name of wedgelist. File is within listdir

tomolist_name	Full name of tomolist. File is within listdir
tomodir	Full path to tomogram directory

Extraction parameters

Parameter	**Description**
subtomo_name	Root name of subtomograms. Files are within subtomodir. File name convention is [subtomo_name]_[subtomo_num].[vol_ext]/
boxsize	Edge size of subtomogram. Subtomograms are extracted as cubes.
pixelsize	Pixelsize in Angstroms.
output_format	Output format of subtomograms.

Particle extraction

For subtomogram extraction, the motivelist is read, and each core reads in a tomogram. Subtomogram information is extracted from the tomogram and written out in the desired format (see below). If the particle center places the edges of the subtomogram outside the tomogram, the missing areas are filled with zeros. The extracted information is normalized before writing out; normalization consists of setting they grey values mean to zero and standard deviation to one. When a core is finished with extracting the subtomograms, it will then begin processing the next unprocessed tomogram; when all tomograms are extracted the job ends. Subtomograms are named [subtomo_name]_[subtomo_num].[vol_ext], with subtomo_num defined in the motivelist, and vol_ext as the file extension of the format chosen; by default this is .mrc format.

A "stats" file is written by default to the metadir subfolder. This file contains the mean and standard values, which can be different than the target normalized values.

Subtomogram formats

By default, the volume formats in STOPGAP are .mrc files, though this can be overridden. The output format is defined by the output_format parameter. Three types of .mrc format are supported: 8-bit integer (mode 0), 16-bit integer (mode 1), and 32-bit float (mode 2), as well as .em format. For integer modes, the grey values are scaled after normalization such that the minimum and maximum values fill the data range. The default is 8-bit .mrc; for locally scaled data this appears to provide sufficient grey value range for high-resolution structure determination. The different types are for minimizing the size of the subtomogram files; all files are converted to single precision during subtomogram alignment and averaging.

Template Matching

Template matching is an approach to particle picking that searches for a template within a tomogram. The template is rotated through a series of search angles; at each angle the template is correlated with the tomogram. At each angle, the voxels are compared to a cumulative correlation map (scoring map); the highest valued voxels are stored in the scoring map. Orientations of the maximum values are also stored in a separate orientation map. Following scoring, the user must decide the score threshold for particle picking.

Template matching in STOPGAP makes use of the same missing-wedge model as STOPGAP subtomo; as such, optimal performance is obtained when exposure-filtering and CTF parameters are included.

Directory structure

The default directories are as follows:

Directory	Parameter Name	Default Directory
Temporary	tempdir	temp/

Communcation	commdir	comm/
Metadata	metadir	meta/
Lists	listdir	lists/
Templates	tmpldir	tmpl/
Masks	maskdir	masks/
Output Maps	mapdir	maps/

Parameters

Parameter	**Description**
tomolist_name	Full name of tomolist. File is in listdir.
wedgelist_name	Full name of wedgelist. File in listdir.
tlist_name	Full name of template list. File in listdir.
smap_name	Root name of scoring map. Output name will be [smap_name]_[tomo_num].[vol_ext]. File is in mapdir.
omap_name	Root name of orientation map. Output name will be [omap_name]_[tomo_num].[vol_ext]. File is in mapdir.
tmap_name	Root name of template map. Output name will be [tmap_name]_[tomo_num].[vol_ext]. File is in mapdir.
binning	Binning factor of dataset.
lp_rad	Low-pass radius in Fourier pixels.
lp_sigma	Low-pass dropoff in Fourier pixels.
hp_rad	High-pass radius in Fourier pixels.
hp_sigma	High-pass dropoff in Fourier pixels.
calc_exp	Calculate exposure filters. (1 = yes, 0 = no)
calc_ctf	Calculate CTF filters. (1 = yes, 0 = no)
apply_laplacian	Apply laplacian transform prior to scoring. (1 = yes, 0 = no)
scoring_fcn	Scoring function. (currently only ‘flcf’ supported)
noise_corr	Perform noise correlation/flattening. (1 = yes, 0 = no)

Settings

For per-task settings, place them into a file in the rootdir name “tm_settings.txt”.

Setting	**Default**	**Description**
wait_time	5	Refresh time for parallelization checks (in seconds).
counter_pct	5	Percentage of job completion for progress output.
tempdir	temp	Default temporary directory
commdir	comm	Default communication directory
tmpldir	tmpl	Default template directory
maskdir	masks	Default mask directory
mapdir	maps	Default map directory
listdir	lists	Default list directory
rawdir	raw	Default raw directory
metadir	meta	Default metadata directory
vol_ext	.mrc	Default volume extension/format
fourier_crop	true	Fourier crop prior to processing
write_raw	false	Write out raw intermediate files

Tomolist

The tomolist is a plain-text file that contains a list of tomograms to process. On each line is a full path to a MRC-formatted tomogram (.rec) and optionally a mask, which will be applied to the scoring map. The tomograms need to be named for their tomogram numbers, e.g. [tomo_num].rec.

The tomolist is used duing job parsing; the parser will pull the relevant information and store it in the parameter file.

Template list

The template list is a .star file that contains information on each template to be matched; when there are multiple entries they are matched simultaneously, resulting in the output of a tmap.

Template lists contain the full name of the template and mask files, the template symmetry, and the angle list to use.

Template lists can be generated using sg_tm_template_list_add_entry.m. Running this multiple times will append existing lists.

Goals of template matching

Before going into procedures, it is important to define “good” template matching. Ideally, the final scoring map should show an autocorrelation function (more precisely, a type of rotational autocorrelation) at the position of each particle. The central peak of the autocorrelation should be substantially higher than all other peaks, allowing for clean thresholding of the scores. While background noise is to be expected, it should amorphous and should not resemble features in the tomogram (e.g. membranes and organelles).

Preparing suitable templates

A suitable template for matching is one that is the same structure as or a close homolog of the target structure. The template must have the same voxel size as the tomogram to be matched. The template can be generated from a PDB file (e.g. using molmap in UCSF Chimera) or using previously determined EM maps, but the best templates are generated from the structures determined from the dataset. In particular, structures generated in STOPGAP subtomo using STOPGAP’s filtering are best suited for template matching as these are all generated using the same filtering framework.

Since it is unlikely that you have a STOPGAP structure prior to starting your project, an outside template can be used for initial template matching. The structure determined from the initial match can be used for subsequent matching. This can be done iteratively, as higher signal-to-noise templates will produce better matching. It is probably a good idea to start with a small subset of your dataset, e.g. one or two tomograms, and determining a low resolution structure (e.g. 30 – 50 Å) before matching the full dataset.

Alignment masks

Like with subtomogram alignment, STOPGAP template matching uses the FLCF. In general, tighter, shaped masks will allow for better local normalization, producing better scoring maps during template matching. However, it is important that you don't cut the density with the mask; if you wish to only match a part of your template, then segment your template. It is also important to mollify your mask; the dropoff should be at least 3 or 4 voxels.

Tomogram masking

Tomograms can be masked to suppress spurious correlations. These types of correlations can come from dark contamination on the surface of the ice; this can be minimized by defining a boundary mask at the edge of the ice. Edges within the tomogram caused by the edge of the tilt image in the aligned stack can be masked out with an edge mask. A useful tool for defining masks is sg_tm_create_boundary_mask.m.

Masking tomogram also reduces computation time, as generally STOPGAP only calculates correlations within the boundaries of the tomogram mask.

Angle lists

Angle lists define the search angles to use in Euler angles. A useful function for generating angle lists is sg_tm_generate_anglist.m. Symmetry allows for smaller parts of orientation space to be searched, allowing for fewer computations. The main parameter to adjust in angular increment, i.e. the angular distance between orientations. Finer increments should produce better dynamic range in the scoring maps, but at the cost of more computations. Realistically, going finer than 12 degrees, at least when filtered to lower resolutions (<20 A), is unnecessary.

Bandpass filtering

Bandpass filtering is important here, as it is with all steps of subtomogram averaging. The higher the resolution used, the more quickly the correlation drops with respect to angular error; therefore higher resolution matches also require finer sampling, but may give better precision. However, the more resolution used, the more model bias is imposed; this may require you to validate your structure through means other than determining the structures to higher resolutions. Typically, I would recommend using a relatively low resolution to perform matching.

Noise flattening

A special feature of STOPGAP template matching is noise correlation or noise flattening. This uses the same idea used for mask-corrected FSC calculations: a noise template is generated from the template by randomizing its phases. The noise template is then matched along with the regular template, resulting is a noise score map. This is then used to flatten the background of the scoring map. In terms of CC value distributions, the scoring maps are roughly normally distributed and noise flattening simply shifts this distribution. However, a substantial number of voxels are shifted below zero; we can then flatten the map by removing all negative correlations. This makes the true peaks relatively stronger and helps in determining the threshold parameters.

Generating motivelists

Motivelists can be generated from scoring maps using sg_tm_generate_motl.m. First the maps are thresholded to a given CC value; the CC value is generally similar within the dataset. One way to estimate the cutoff is the plot the sorted CC values; this results in a roughly logarithmic curve. The approximate threshold region is within the linear region after the inflection point. Another method can visualize the score map as an isosurface and find the threshold that only shows peaks.

After thresholding, particle positions are picked from the highest remaining peaks using a distance threshold, which should be set to the radius of the particle. True matches should have a peak resembling the autocorrelation peak; as such they should have a certain size distribution that can be defined with the cluster_size parameter. Random salt-and-pepper noise can be removed with the minimum cluster size threshold while large peaks, such as from gold fiducials, can be removed with the maximum threshold.

Principle Component Analysis

One method for classification is principle component analysis (PCA), where the principle components of the dataset are computed as eigenvectors, and each subtomogram is given a set of eigenvalues for each eigenvector. This is somewhat analogous to performing a Fourier transform, but rather than sines and cosines, the basis functions are eigenvectors. Eigenvectors are ordered by the largest range of eigenvalues; e.g. eigenvector 1 has the widest range of eigenvalues. Technically, PCA only delivers the eigenvectors and eigenvalues, not classes; classification is performed by clustering the eigenvalues. In STOPGAP, the user first selects the eigenvectors to classify with, which are then used in a k-means clustering algorithm.

PCA-based classification attempts to separate classes by trying to identify the variable parts of the dataset; this is distinct from multi-reference based classification, which classifies by similarity to a set of pre-generated references.

STOPGAP allows for two different ways of calculating PCA: the first is by calculating a CC-matrix between all pairs of subtomograms (see doi: 10.1016/j.jsb.2007.07.006), similar to the PCA algorithms in AV3 and DYNAMO. PCA is then performed on the CC-matrix to yield eigenfactors that provide the weight of each subtomogram on each eigenvector; these are used to generate the eigenvectors, when are then used to calculate the final eigenvalues.

The second method is to generate a covariance matrix from the subtomogram data, which simply consists of 3D image data projected on to rows of the covariance matrix. Singular value decomposition (SVD) is performed on the covariance matrix to directly yield the eigenvectors and eigenvalues; this generally requires substantially less computation that CC-matrix based approaches. This is similar to the approaches in PEET and emClarity (see doi: 10.1016/j.jsb.2011.05.011 and 10.1038/s41592-018-0167-z).

The two methods have their own advantages and disadvantages. The calculation of the CC-matrix can be quite expensive, but the missing wedge is explicitly considered in the pairwise constrained cross-correlation. However, the amount of common Fourier space between two subtomograms can be quite variable, which can result in poor normalization of the real-space correlations. This can indirectly result in particle orientations becoming a significant feature in the CC-matrix. However, the covariance-based approach does not take into account the missing wedge at all. Typically, it is performed using difference data (i.e. the difference between the subtomogram and the reference), which when combined with local masking may produce acceptable results.

Each method can be performed with the subtomogram EM-density data, or with difference data in the form of “wedge-masked differences” (WMDs) or “amplitude-weighted phase-differences” (AWPDs).

Pre-rotating volumes

An initial step prior to PCA calculation is to pre-rotate volumes. Subtomograms are rotated into their aligned positions while matching filters are also generated. These filters are primarily used in CC-matrix based PCA, as they are required to calculate pairwise correlations, generate weighted eigenvectors, and for eigenvalue calculation.

During pre-rotation, volumes are also symmetrized; this can partially fill the missing wedge of the subtomograms and yield better PCA results using either method.
Pre-rotation is performed by parsing the 'rot_vol' task in the PCA parser.

Multi-scale filtering

STOPGAP allows for generating eigenvectors using multiple bandpass filters in order to generate eigenvectors that reflect that variance at different resolution regimes. This approach is based off that implemented in emClarity. The bandpass filters used during PCA are defined in a filter list, which is generated using the sg_pca_append_filter_list.m function.

Performing PCA using the CC-matrix

The CC-matrix first needs to be calculated by calculating the pairwise constrained cross correlation between each pair of subtomograms. Each subtomogram is filtered with the other subtomogram's missing wedge, so that only common information is compared in real-space.

The CC-matrix itself is symmetric about the diagonal and equal to one on the diagonal. As such, each pair of subtomograms only needs to be calculated once, and the self-pairs do not need to be calculated. CC-matrix calculation is performed by parsing the calc_ccmat task in the PCA-parser.

After the CC-matrix is calculated, PCA is performed in several steps. First, eigenfactors for each eigenvector are calculated from the CC-matrix. These are then used to generate eigenvolumes as weighted averages of the subtomograms. The subtomograms are then correlated against the eigenvolumes to generate their eigenvalues. These steps are all performed by parsing the 'calc_pca_cccmat' task in the PCA parser.

Performing PCA using a covariance matrix

The first task is to generate the covariance matrix; this is performed using the 'calc_covar' task in the PCA-parser. Rows of the covariance matrix are simply the masked-in voxels from each subtomogram.

From there, the PCA is performed locally in MATLAB using the sg_pca_covar_eigen function. This will perform SVD on the covariance matrix and write out the eigenvectors and eigenvalues. Running SVD on a ~160000x3600 covariance matrix takes about 10 minutes.

Parameters are taken from the PCA parameter file; as such the correct line index must be passed as the param_idx variable. The first data line in the .star file is index 1.

Classification by K-means clustering

The ultimate results of PCA are the eigenvalues for each eigenvector. From there, one will typically classify the subtomograms using k-means clustering. This is performed locally using the sg_pca_kmeans_cluster.m. Again, most parameters are parsed from the PCA parameter file, but you must supply which eigenvectors to classify with and how many classes you want.

The eigenvectors are supplied as pair values, with the first column containing the filter-list index, and the second containing the eigenvector from that list. To help visualize the eigenvalue distributions of each eigenvector, eigenvalues files can be plotted as histograms using the sg_pca_plot_eigenvalue_hist.m

Based on your inputs, a new motivelist is written with the given suffix.

Multi-reference Classification

Multi-reference classification is a classification is an approach that is based on aligning the subtomogram against multiple references. Ideally, each reference represents a distinct class, and each subtomogram will correlate highest with its correct class. Algorithmically, this is built into subtomogram averaging function of STOPGAP, and classification is run by supplying multiple references and using either a multiclass or multi-entry motivelist. However, there are many caveats to using mult-reference based approaches. This section details some of these caveats and provides some general strategies for reproducible classification.

Goals of Classification

Classification can be a bit of a nebulous term; you may be trying to remove incorrectly picked subtomograms (i.e. particles that do not contain any structures), separate distinct molecules, or separate the same molecules in different conformations or binding states. Each use case has slightly different requirements.

Removing incorrectly picked particles is probably the more challenging problem for multi-reference classification. Multi-reference approaches rely on the scoring functions; if you can't remove bad particles purely from scores, multiple references may not help. This is because the bad particles probably consist of heterogeneous data like carbon edges; they are unlikely to converge onto a distinct class. In such cases, PCA might be a better approach.

For classifying distinct molecules, it is best if you can supply multiple pre-determined references. Even low-resolution references are likely helpful, so it may be best to manually pick particles to establish these initial references. However, if this is not possible, multi-reference classification can still work with randomly generated references, though it might also be worth using PCA.

Classifying conformations or binding states is probably the most challenging classification problem. This is because the differences between classes are subtler than in other cases. It is also likely that you won't be able to see differences by eye, and are unable to pick and determine independent structures. In such cases starting references will have to be randomly generated, which carries it's own set of challenges. In our experience, PCA is unlikely to be useful in these cases.

Number of classes

One of the first problems is to decide a number of classes. If the number of classes is not known, you will have to guess the number of classes or use more classes than you need. STOPGAP allows classes to empty during classification so assigning more classes than there are may converge onto the correct number of classes. Multiple classes that converge to the same structure can also be determined and combined later. However, since each subtomogram is only assigned to one class at each iteration, too many classes can result in too few subtomograms per class and unstable classification.

Motivelists for classification

For classification, two types of motivelists can be used: multiclass or multi-entry motivelists. Multi-entry motivelists are for when you think it's useful to keep multiple independent alignments. This is probably most useful when you are trying to classify different molecules. For classifying multiple conformations or binding states, a multiclass motivelist might be better, as it might prevent angular divergence of the class alignments.

NOTE: STOPGAP knows to read classes based on the class numbers in the motivelist. Regardless of how the references are generated, the initial motivelist used for multireference alignment must contain the proper number of classes. For example, classification with 5 classes numbered 1 to 5 must either use a multiclass motivelist with classes 1 to 5 assigned to some number of tomograms, or a 5 entry motivelist with classes 1 to 5 assigned.

Randomly generating references

When outside starting references are not available, references can be generated from random subsets of subtomograms. One method is to evenly split the dataset into the number of desired classes; this can be performed using `sg_motl_apply_random_classes`. However, this can cause bias, as each subtomogram may prefer the reference that contains them.

A better method may be to generate references with a specified number of randomly selected subtomograms; in this approach subtomograms can be appear in several references or no references, reducing initialization bias. The number of subtomograms should be high enough to ensure sufficient angular sampling. This can be performed using `sg_motl_intiailize_random_subset_classes`. NOTE: when using this method, you will need to use the motivelist generated using `sg_motl_apply_random_classes` for averaging the starting references, but a motivelist generated using `sg_motl_intiailize_random_subset_classes` for alignment.

Halfsets

When using randomly generated references, ignore_halfsets must be enabled. If it is not, the random A and B references will not be of the same class, preventing convergence and resulting in poor resolutions. Since halfsets will be disabled, it is wise to limit the resolution used in alignment to prevent overalignment.

Initialization bias and premature convergence

Two problems with multi-reference classification are initialization bias and premature convergence. Initialization bias can occur when Initialization bias is the tendency for subtomograms to never change class during alignment. While there is the probability that some subset may randomly be assigned to a correct class at the start, typically the number of subtomograms that don't change classes is much higher than likely. Another related problem premature convergence, i.e. when classification reaches convergence within a short number of iterations.

While initialization bias can be reduced though reference generation as described above, an additional method for preventing initialization bias is to use several rounds of simulated annealing. By allowing for sub-optimal class assignments, it can be assured that all subtomograms change class at least once. It is important to note that convergence will not happen during simulated annealing, and more alignment iterations must be performed with no simulated annealing.

To prevent premature convergence, it is best to perform multi-reference alignment using SHC. SHC will slow the rate of convergence and allow subtomograms to jump between classes during initial iterations. As the references improve, subtomograms should still converge into distinct classes; if they do not, it is a sign of poor classification and your strategy should be re-evaluated.

Monitoring classification

There are several scripts in the STOPGAP toolbox for evaluating your classification. The number of class changes for each subtomogram can be plotted as a histogram-using `sg_motl_plot_class_changes`; this provides a way to assess for initialization bias. The status of the classification can be assessed with to plots. The first is to monitor the number of subtomograms changing class at each iteration; this can be plotted using `sg_motl_plot_class_convergence`. It's unlikely that this number will be zero, but it should stabilize to a small portion of the dataset (< 1%). If the number settles to a substantial part of the dataset, the classification is unstable. The second is to monitor the class occupancies (i.e. number of subtomograms in each class); this can be plotted with `sg_motl_plot_class_occupancies`. These should jump around during simulated annealing and the first few iterations of SHC, but should eventually plateau at stable values. If the class occupancies do not stabilize, your classification is unstable and you may need to reduce the number of classes or revise your method of generating starting references.

Reproducibility

STOPGAP attempts to minimize bias in multi-reference classification by providing methods that introduce some level of stochasticity such as simulated annealing and SHC. However, these methods do not ensure unbiased or correct classification. It is best practice to perform the same classification protocol multiple times: reproducible results despite using stochastic methods are a good indicator of successful and correct classification.