# Single-particle processing in RELION-2.1

Sjors H.W. Scheres
(ccpem@jiscmail.ac.uk)

June 9, 2017

**Abstract**

This tutorial provides an introduction to the use of RELION-2.1 for cryo-EM structure determination. This tutorial covers the entire single-particle analysis workflow in RELION-2.1: beam-induced motion-correction, CTF estimation; automated particle picking; particle extraction; 2D class averaging; initial model generation; 3D classification; high-resolution 3D refinement; the processing of movies from direct-electron detectors; and final map sharpening and local-resolution estimation. Carefully going through this tutorial should take less than a day (if you have a suitable GPU or if you follow our precalculated results). After that, you should be able to run RELION on your own data.

If you have any questions about RELION, first check the FAQ on the RELION Wiki and the archives of the CCPEM mailing list. If that doesn't help, subscribe to the CCPEM email list and use the email address above for asking your question. *Please, please, please, do not send a direct email to Sjors, as he can no longer respond to all of those.*

# Contents

# 1 Getting prepared

## 1.1 Recommended reading

The theory behind the refinement procedure in RELION is described in detail in:

- S.H.W. Scheres (2012) "RELION: Implementation of a Bayesian approach to cryo-EM structure determination" *J. Struc. Biol.*, 180, 519-530.
- S.H.W. Scheres (2012) "A Bayesian view on cryo-EM structure determination" *J. Mol. Biol.*, 415, 406-418.

A comprehensive overview of how to use RELION is described in:

- S.H.W. Scheres (2016) "Processing of structurally heterogeneous cryo-EM data in RELION" *Meth. Enzym.*, in press.

## 1.2 Install MPI

Note that you'll need a computing cluster (or a multi-core desktop machine with NVIDIA GPUs) with an MPI (message passing interface) installation. To compile RELION, you'll need a mpi-devel package. The exact flavour (openMPI, MPICH, LAM-MPI, etc) or version will probably not matter much. If you don't have an mpi-devel installation already on your system, we recommend installing openMPI.

## 1.3 Install CUDA

If you have a relatively modern GPU from NVIDIA (with compute capability 3.5+), then you can accelerate your autopicking, classification and refinement jobs considerably. In order to compile RELION with GPU-acceleration support, you'll need to install CUDA. We used CUDA-7.5 to prepare this tutorial. Download it from NVIDIA's website.

## 1.4 Install RELION

RELION is open-source software. Download it for free from the RELION wiki, and follow the installation instructions. If you're not familiar with your job submission system (e.g. Sun Grid Engine, PBS/TORQUE, etc), then ask your system administrator for help in setting up the qsub.csh script as explained in the installation instructions. Note that you will probably want to run so-called hybridly-parallel jobs, i.e. calculations that use both MPI for distributed-memory parallelization AND pthreads for shared-memory parallelization. Your job submission queueing system may require some tweaking to allow this. Again, ask your sysadmin for assistance.

## 1.5 Install motion-correction software

RELION-2.1 provides a wrapper to the UCSF program MOTIONCOR2, which is used for whole-frame micrograph movie-alignment [21]. Download the program from David Agard's page and follow his installation instructions. Alternatively, you may also use UNBLUR [4] from Niko grigorieff's group. Download it from Niko's UNBLUR web site.

## 1.6 Install CTF-estimation software

CTF estimation is not part of RELION. Instead, RELION provides a wrapper to Alexis Rohou and Niko Grigorieff's CTFFIND4 [9]. Please download this from Niko's CTFFIND website and follow his installation instructions. Alternatively, if you have NVIDIA graphics cards (GPUs) in your machine, you may also use Kai Zhang's GCTF [20], which may be downloaded from Kai's website at LMB.

## 1.7 Install RESMAP

Local-resolution estimation may be performed inside RELION's own postprocessing program, or through a wrapper to Alp Kucukelbir's RESMAP [7]. Please download it from Alp's RESMAP website and follow his installation instructions.

## 1.8 Download the test data

This tutorial uses a test data set that is a subset of the micrographs used to calculate various beta-galactosidase reconstructions [15, 2, 19]. To reduce the computational load, these data were 2x down-sampled. The data may be downloaded and unpacked using the following commands:

```
wget ftp://ftp.mrc-lmb.cam.ac.uk/pub/scheres/relion21_tutorial.tar.gz
gunzip relion21_tutorial.tar.gz
tar -xf relion21_tutorial.tar
```

If you are using your own computer system to follow this tutorial, note that in order to run on a cluster, you'll need a `qsub.csh` script that has been configured for your particular queueing system. Ask your system administrator if you don't know whether you have the correct script.

# 2 General description

## 2.1 The RELION GUI

### 2.1.1 A pipeline approach

The GUI serves a central role in it's pipelined approach, details of which have been published in the 2016 Proceedings of the CCP-EM Spring Symposium [3]. We recommend to create a single directory per project, i.e. per structure you want to determine. We call this the project directory. It is important to always launch the relion graphical user-interface (GUI), by typing the command `relion`, from the project directory.

The GUI keeps track of all jobs and how output from one job is used as input for another, thereby forming a workflow or pipeline. Each type of job has its own output directory, e.g. `Class2D/`, and inside these job-type directories, new jobs get consecutive numbers, e.g. `Class2D/job010`. Inside these individual job directories, output names are fixed, e.g. `Class2D/job010/run`. To provide a mechanism to have more meaningful names for jobs, a system of job "aliases" is used, which are implemented as symbolic links to the individual job directories on the filesystem. All info about the pipeline is stored in a file called `default_pipeline.star`, but in normal circumstances the user does not need to look into this file. In case this file gets corrupted, one can copy back a backup of this file from the last executed job directory.

### 2.1.2 The upper half: jobtype-browser and parameter-panel

On the left of the upper half of the GUI is the jobtype-browser: a vertical list of jobtypes, e.g. 2D classification. On the right is a panel with multiple tabs, where parameters to the different types of jobs may be input. On the top left of the GUI are three different menu's, providing a range of functionalities. The Schedule and Run now! buttons can be used to schedule jobs for future execution, or to execute them now. The former is particularly useful in preparing fully automated "pipelines" that can be run iteratively, for example in real-time as data is being collected. See section 2.3 for more details. By clicking in the jobtype-browser on the left-hand side of the GUI, a new job (with a Run now! button) will be loaded in the parameter-panel on the right.

### 2.1.3 The lower half: job-lists and stdout/stderr windows

The lower half of the GUI contains lists of jobs that are still running ( Running jobs ), have already finished ( Finished jobs ), or are scheduled for later execution ( Scheduled jobs ). By clicking jobs in these lists, the parameters of that job will be loaded in the parameter-panel, and the Run now! button will change color and turn into continue now!. Upon clicking the latter, no new output job-directory will be made, but the job will be continued according to the parameters given in the parameter-panel. 2D classification, 3D classifications and 3D auto-refine jobs will need a `_optimiser.star` file to continue from, and will have filenames with the iteration from which they were continued, e.g. `run_ct23`. Other types of jobs may continue from the point until they were executed before, e.g. Motion correction, CTF estimation, Auto-picking and Particle Extraction will continue by running only on those micrographs that weren't done before. The Input to this job and Output from this job lists link jobs together and can be used to browse backwards or forwards in the project history.

At the bottom of the lower half of the GUI, the standard output (stdout) and standard error (stderr) of the selected (finished or running) job will show in black and red text, respectively. The stderr should ideally be empty, any text here is usually worth inspection. These text displays get updated every time you click on a job in the job-lists. Double-clicking on the stdout or stderr displays will open a pop-up window with the entire text for more convenient scrolling.

### 2.1.4 The Display button

The Display: button below the run and schedule buttons serves to visualise the most important input and output files for each job. When a job from the job-lists in the lower half of the GUI is selected, clicking this button will pop-up a menu with all the input and output of this job that can be displayed (for example, particles, micrographs, coordinates, PDF files, etc). A more general functionality to display any (e.g. intermediate) file can be accessed through the `Display` option of the `File` menu on the top left of the GUI.

### 2.1.5 The Job actions button

The Job actions button opens up a little menu with options for the selected (running, finished or scheduled) job. Here, you can access a file called `note.txt` (that is saved in every individual job directory and which may be used to store user comments); you can change the alias of the job; you can mark a job as finished (in case it somehow got stuck); you can make a flowchart of the history of that job (provided LaTeX and the `TikZ` package are installed on your system, also see section 11); or you can delete or clean a job to save disk space (see below).

### 2.1.6 Clean-up to save disk space

Deletion of jobs moves the entire job directory from the project directory into a directory called `Trash/`. You can empty the Trash folder from `File` menu on the top left of the GUI to really free up the space. Until you do so, you can still "undelete" jobs using the corresponding option from the `Jobs` menu on the top left.

To save disk space, you can also "clean" jobs, which will move intermediate files to the Trash folder, e.g. the files written out for all intermediate iterations of refine jobs. There are two cleaning options: `gentle clean` will leave all files intact that could be used as input into another job, while `harsh clean` may also remove those. Evidently, "harsh" cleaning can free up more space, in particular directories with particle stacks or micrographs may become large, e.g. from $\boxed{\text{Motion correction}}$, $\boxed{\text{Particle extraction}}$, $\boxed{\text{Movie refinement}}$ and $\boxed{\text{Particle polishing}}$ job types. One can also clean all directories in the project with a single click using the corresponding options from the `Jobs` menu on the top left of the GUI. You can protect specific directories from "harsh" cleaning by placing a file called `NO_HARSH_CLEAN` inside them, e.g. you may want to protect your final set of polished particles from deletion by executing:

```
touch Polish/job098/NO_HARSH_CLEAN
```

## 2.2 Optimise computations for your setup

### 2.2.1 GPU-acceleration

Dari Kimanius and Bjoern Forsberg from the group of Erik Lindahl (Stockholm) have ported the most computationally expensive parts of RELION for the use of GPUs. Because they used the CUDA-libraries from NVIDIA to do this, GPU-acceleration in RELION only works with NVIDIA cards. These need to be of compute capability 3.5 or higher. Both single and double precision cards will work, so one is not restricted to the expensive double-precision cards, but can use the cheaper gaming cards as well. Details of their implementation can be found in their eLife paper[6].

Two different relion programs have been GPU-accelerated: `relion_autopick` (for $\boxed{\text{Auto-picking}}$) and `relion_refine` (for $\boxed{\text{2D classification}}$, $\boxed{\text{3D classification}}$ and $\boxed{\text{3D auto-refine}}$ jobs). Both the sequential and the MPI-versions of these programs have been accelerated.

### 2.2.2 Disk access

With the much improved speed of image processing provided by the GPU-acceleration, access to the hard disk increasingly becomes a bottle neck. Several options are available on the RELION GUI to optimise disk access for your data set and computer setup. For $\boxed{\text{2D classification}}$, $\boxed{\text{3D classification}}$ and $\boxed{\text{3D auto-refine}}$ one can choose to `use parallel  disc I/O`. When set to `Yes`, all MPI processes will read particles simultaneously from the hard disk. Otherwise, only the master will read images and send them through the network to the slaves. Parallel file systems like gluster of fhgfs are good at parallel disc I/O. NFS may break with many slaves reading in parallel.

One can also set the `number of pooled particles`. Particles are processed in individual batches by MPI slaves. During each batch, a stack of particle images is only opened and closed once to improve disk access times. All particle images of a single batch are read into memory together. The size of these batches is at least one particle per thread used. This parameter controls how many particles are read together in a batch by each thread. If it is set to 3 and one uses 8 threads, batches of 3x8=24 particles will be read together. This may improve performance on systems where disk access, and particularly metadata handling of disk access, is a problem. It has a modest cost of increased RAM usage.

If one has a relatively small data set (and/or a computer with a lot of RAM), then one can `pre-read all particles into RAM` at the beginning of a calculation. This will greatly speed up calculations on systems with relatively slow disk access. However, one should of course be careful with the amount of RAM available. Because particles are read in float-precision, it will take $\frac{N \times boxsize \times boxsize \times 4}{1024 \times 1024 \times 1024}$ Giga-bytes to read N particles into RAM. For 100,000 particles with a 200-pixel boxsize that becomes 15Gb, or 60 Gb for the same number of particles in a 400-pixel boxsize.

If the data set is too large to pre-read into RAM, but each computing node has a local, fast disk (e.g. a solid-state drive) mounted with the same name, then one can let each MPI slave copy all particles onto the local disk prior to starting the calculations. This is done using the `Copy particles to scratch directory`. If multiple slaves will be executed on the same node, only the first slave will copy the particles. If the local disk is too small to hold the entire data set, those particles that no loner fit on the scratch disk will be read from their original position. A sub-directory called `relion_volatile` will be created inside the specified directory name. For example, if one specifies `/ssd`, then a directory called `/ssd/relion_volatile` will be created. If the `/ssd/relion_volatile` directory already exists, it will be wiped before copying the particles. Then, the program will copy all input particles into a single large stack inside this directory. If the job finishes correctly, the `/ssd/relion_volatile` directory will be deleted again. If the job crashes before finishing, you may want to remove it yourself. The program will create the `/ssd/relion_volatile` directory with writing permissions for everyone. Thereby, one can choose to use `/ssd`, i.e. without a username, as a scratch directory. That way, provided always only a single job is executed by a single user on each computing node, the local disks do not run the risk of filling up with junk when jobs crash and users forget to clean the scratch disk themselves.

Finally, there is an option to `combine iterations through disc`. If set to `Yes`, at the end of every iteration all MPI slaves will write out a large file with their accumulated results. The MPI master will read in all these files, combine them all, and write out a new file with the combined results. All MPI salves will then read in the combined results. This reduces heavy load on the network, but increases load on the disc I/O. This will affect the time it takes between the progress-bar in the expectation step reaching its end (the mouse gets to the cheese) and the start of the ensuing maximisation step. It will depend on your system setup which is most efficient. This option was originally implemented to circumvent bugs on the network cards on our old cluster at LMB. Nowadays, we prefer not to use this option, as it tends to be very slow when refinements reached high resolutions.

## 2.3 On-the-fly processing: running scheduled jobs

The Schedule button can be used to prepare jobs for future execution. Because expected output files of scheduled jobs are already available for input into other jobs from their Browse buttons, it is possible to build a "pipeline" of multiple, consecutive jobs in the Scheduled jobs list. One can execute the scheduled jobs using the `Autorun` menu on the top left of the GUI. The program will ask you how many times you want to run the scheduled jobs. Providing a value larger than one, will result in an iterative execution of all the scheduled jobs. If you choose to do this, there will be another question of how many minutes should at least pass between each repetition of all scheduled jobs. (If the execution of all scheduled jobs takes less than this time, the program will wait with executing the next round of scheduled jobs.) The following jobtypes will only process micrographs that were not finished yet in a previous iteration: Import, Motion correction, CTF estimation, Auto-picking, Particle extraction and Movie refinement. All other jobtypes will restart their calculation from scratch again after each iteration of executing the scheduled jobs.

This setup allows convenient on-the-fly processing of micrographs, as they are being collected. One can set up a script that copies micrographs, or micrograph movies from the microscope computer onto a processing computer (preferably with one or more suitable GPUs for fast processing). (Because copying large files may take some time, and you do not want to import micrographs/movies that have not yet finished copying, this script could copy the files over with a temporary filename, and only once they have arrived entirely could the file then be renamed to their final filename.) A list of scheduled jobs starting with for example an Import of all `Micrographs/*.mrcs` files, could be followed by Motion correction, CTF estimation, Auto-picking (provided you already have suitable templates, otherwise someone would have to manually pick some particles after a few micrographs have been acquired), Particle extraction and then 2D classification. This list of scheduled jobs could be repeated many times, perhaps with at least 30 or 60 minutes between each iteration. The list of imported micrographs will grow ever longer, and each iteration of running the scheduled jobs, Motion correction, CTF estimation, Auto-picking and Particle extraction will only be run on the new micrographs, while the 2D classification will be re-run from scratch with the total number of particles in that iteration. Inspecting 2D class averages already during data collection at the microscope may be useful in order to decide whether the data is worth acquiring in the first place, or whether the orientational distribution is OK or one perhaps needs to collect in regions on the grid with thicker ice. If one has a good 3D reference already, one may even perform 3D classification or 3D auto-refine on-the-fly. This may be useful, for example, to confirm whether a certain cofactor is bound to the complex of interest.

Please note that the subset selection can be scheduled, but the user will be prompted for input, for example by displaying the `_model.star` from a 2D classification to select suitable classes. The execution of all scheduled jobs will not continue until the user saves this selection. Automation of this step may happen in a future release of RELION.

## 2.4 Helical reconstruction

Shaoda He, a PhD-student in the Scheres group, has implemented a workflow for the processing of helical assemblies. This involves additional tabs to the parameter-panels of the Auto-picking, Particle extraction, 2D classification, 3D classification, 3D auto-refine, Particle polishing and Mask create job-types. We do not have a separate tutorial for processing helical assemblies. The general principles remain the same as for single-particle analysis, which is covered in this tutorial. Therefore, users intending to use RELION for helical processing are still encouraged to do this tutorial first. For a detailed description of the helical options, the user is referred to the corresponding pages on the RELION Wiki, or to Shaoda's paper[5].

## 2.5 Sub-tomogram averaging

For sub-tomogram averaging, which was implemented with help from Tanmay Bharat, a former postdoc in the Lowe group at MRC-LMB, the same holds as for helical processing. Many general concepts remain the same as for single-particle analysis, and users intending to perform sub-tomogram averaging in RELION are encouraged to go through this tutorial first. For a detailed description of the sub-tomogram averaging procedures, the user is referred to the corresponding pages on the RELION Wiki, or to Tanmay's paper[1].

# 3 Preprocessing

Although, in principle, RELION can use particles that have been extracted by a different program, this is NOT the recommended procedure. Many programs change the particles themselves, e.g. through phase flipping, band-pass or Wiener filtering, masking etc. All these are sub-optimal for subsequent use in RELION. Moreover, gathering all the required metadata into a correctly formatted RELION-type STAR file may be prone to errors. Because re-extracting your particles in RELION is straightforward and very fast, the procedure outlined below is often a much easier (and better) route into RELION.

Also, several implementations of wrappers around RELION have now been reported (e.g. in EMAN2, SCIPION and APPION). Although we try to be helpful when others write these wrappers, we have absolutely no control over them and do not know whether their final product uses RELION in the best way. Therefore, in case of any doubt regarding results obtained with these wrappers, we would recommend following the procedures outlined in this tutorial.

## 3.1 Getting organised

We recommend to create a single directory per project, i.e. per structure you want to determine. We'll call this the project directory. **It is important to always launch the RELION graphical user-interface (GUI) from the project directory.** Inside the project directory you should make a separate directory to store all your raw micrographs or micrograph movies in MRC format. We like to call this directory `Micrographs/` if all micrographs are in one directory, or for example `Micrographs/15jan16/` and `Micrographs/23jan16/` if they are in different directories (e.g. because the micrographs were collected on different dates). If for some reason you do not want to place your micrographs inside the RELION project directory, then inside the project directory you can also make a symbolic link to the directory where your micrographs are stored.

Single-image micrographs should have a `.mrc` extension, movies should have a `.mrcs` extension. If you collected movies instead of single-image micrographs, then you can use those directly. If you calculated averages of the movies in a different program and you want to use both the average micrographs and the movies, then your movies should be named in a fixed manner with the same rootname as the corresponding average micrograph, followed by an "_" character, a "movie rootname" and a `.mrcs` extension. For example, if the average micrograph is called `mic001.mrc` then the corresponding movie should be called `mic001_movie.mrcs` (where the movie rootname would be `movie`).

When you unpacked the tutorial test data, the Project directory (`betagal/`), and the corresponding `betagal/Micrographs` directory with all the recorded micrograph movies were already created. Go to the project directory, and see how many micrograph movies there are by typing:

```
cd relion21_tutorial/betagal
ls Micrographs/
```

We will start by launching the RELION GUI. As said before, this GUI always needs to be launched from the project directory. To prevent errors with this, the GUI will ask for confirmation the first time you launch it in a new directory. Therefore, the first time you launch the GUI in a new directory, you should not use the "&" character to launch it in the background. Make sure you are inside the project directory, and launch the GUI by typing:

```
relion
```

and answer "y" to set up a new RELION project here.

The first thing to do is to import the set of recorded micrograph movies into the pipeline. Select "Import" from the job-type browser, and fill in the following parameters:

- `Input files:` `Micrographs/*.mrcs`
- `Node type:` `2D micrograph movies`

You may provide a meaningful alias (for example: `movies`) for this job in the white field named `Current job: Give_alias_here`. Clicking the Run now! button will launch the job. A directory called `Import/job001/` will be created, together with a symbolic link to this directory that is called `Import/movies`. A pop-up text editor window with the `note.txt` file contains the actual command executed and allows you to write down any comments, otherwise just close it. Inside the newly created directory a STAR file with all the movies is created. Have a look at it using:

```
less Import/job001/movies.star
```

If you had extracted your particles in a different software package, or you want to transfer particles extracted in RELION-1.4 into RELION-2.x, then instead of going through the Preprocessing steps below, you would use the "Import" job-type to import particles STAR file, 3D references, 3D masks, etc.

## 3.2 Beam-induced motion correction

The Motion correction job-type provides a wrapper to UCSF MOTIONCOR2 [21] for convenient whole-frame movie alignment. On the I/O tab set:

- `Input movies STAR file:` `Import movies/movies.star`

  (Note that the Browse button will only list movie STAR files.)

- `Save aligned movie stacks?` `Yes`

  (We will need these to perform particle polishing later on.)

- `First frame for corrected sum:` `1`
- `Last frame for corrected sum:` `16`

  (These are 16-frame movies, and we'll average over all frames.)

- `Pixel size (A)` `3.54`

  (These are 2x downsampled images with an original pixel size of 1.77A.)

Fill in the Motioncor2 tab as follows:

- `Use MOTIONCOR2?` `Yes`

  (provided you have the required NVIDIA GPU on your computer, otherwise use UNBLUR on the next tab.)

- `MOTIONCOR2 executable:` `/wherever/it/is/motioncor2`

  (use environment variable `$RELION_MOTIONCOR2_EXECUTABLE` to control the default here)

- `Gain-reference image:`

  (This can be used to provide a gain-reference file for on-the-fly gain-reference correction.)

- `Defect file:`

  (This can be used to mask away broken pixels on the detector.)

- `Archive directory:`
- `Number of patches X,Y` `5 5`
- `Group frames:` `1`
- `Binning factor:` `1`

  (we often use 2 for super-resolution movies)

- `Bfactor:` `150`

  (use larger values for super-resolution movies)

- `Which GPUs to use:` `0`

  (MOTIONCORR needs to run on an NVIDIA GPU.)

- `Other MOTIONCORR arguments:`

Fill in the Dose-weight tab as follows:

- `Do dose-weighting?` `Yes`

- `Voltage (kV):` 300
- `Dose per frame (e/A2)` 1
- `Pre-exposure (e/A2)` 0

Executing this program takes approximately 4-5 minutes (but you need an NVIDIA GPU for this to work). Alternatively, you could also use Niko Grigorieff's UNBLUR program [4], which is available throught the Unblur tab. Note that all MOTIONCOR2 settings will be ignored, and that you can only run UNBLUR with its default parameters, so without its advanced options. The unblur wrapper was tested for UNBLUR version 1.0.2. You can look at the estimated beam-induced shifts by selecting the `out: logfile.pdf` from the Display: button below the run buttons, or you can look at the summed micrographs by selecting `out: corrected_micrographs.star`. Depending on the size of your screen, you should probably downscale the micrographs (`Scale: 0.3`) and use `Sigma contrast: 3` and few columns (something like `Number of columns: 3`) for convenient visualisation. Note that you cannot select any micrographs from this display. If you want to exclude micrographs at this point (which we will not do, because they are all fine), you could use the Subset selection job-type.

## 3.3 CTF estimation

Next, we will estimate the CTF parameters for each corrected micrograph. We will the CTF estimation job-type as a wrapper to Kai Zhang's GCTF, for the sake of speed. Note that you can also use Alexis Rohou and Niko Grigorieff's CTFFIND4.1 if you don't have a suitable GPU. (In that case the defaults on the CTFFIND tab should be fine.) CTFFIND4.1 can also use the micrograph movies to calculate better Thon rings. On the I/O tab, use the Browse button to select the `corrected_micrographs.star` file of the Motion correction job. Then fill in the other settings as follows:

On the I/O:

- `Use micrograph without dose-weighting?` Yes

  (These may have better Thon rings than the dose-weighted ones.)
- `Sperical aberration (mm):` 2

  (your microscope manufacturer provides this value)
- `Voltage (kV):` 300
- `Amplitude contrast:` 0.1

  (although amplitude contrast is known to be less, giving values of around 10% has been shown to yield better results for many structures. This may be because of unmodelled low-frequency inelastic scattering.)

- `Magnified pixel size (A):` `3.54`

  (these are 2x binned data, with an original pixel size of 1.77A)

- `Amount of astigmatism (A):` `100`

  (Assuming your scope was reasonably well aligned, this value will be suitable for many data sets.)

On the Searches tab, you provide general parameters for CTFFIND (see Niko's documentation for their exact meaning).

- `FFT box size (pix):` `512`
- `Minimum resolution (A):` `30`
- `Maximum resolution (A):` `7.1`
- `Minimum defocus cvalue (A):` `5000`
- `Maximum defocus cvalue (A):` `50000`
- `Defocus step size (A):` `500`
- `Amount of astigmatism (A):` `100`
- `Estimate phase shifts` `No`

  (This is useful for phase-plate data only.)

Ignore the CTFFIND-4.1 tab if using GCTF, and vice versa. In this example, as GCTF seems to have trouble working with such low-resolution images as the ones used here, we use CTFFIND4.1, and on the CTFFIND-4.1 tab, we set:

- `Use CTFFIND-4.1?` `Yes`
- `CTFFIND-4.1 executable:` `/wherever/it/is/Gctf`

  (Note that environment variables `$RELION_CTFFIND_EXECUTABLE` and `$RELION_GCTF_EXECUTABLE` are used to control the default values for the GUI entries of the corresponding programs.)

- `Estimate Thon rings from movies` `No`

  (This is not necessary here.)

- `Estimate CTF on window size (pix):` `-1`

  If set to a positive value, the program will crop a central square with this size and estimate CTF on the cropped image only. this is for example useful for scanned films, which may have labels on the side of it.

You can run the program using multiple MPI processes, depending on your machine. Using only a single processor, the job took 2 minutes with CTFFIND-4.1. Once the job finishes there are additional files for each micrograph inside the output `CtfFind/job003/Micrographs` directory: the `.ctf` file contains an image in `MRC` format with the computed power spectrum and the fitted CTF model; the `.log` file contains the output from CTFFIND or GCTF; (only in case of using CTFFIND, the `.com` file contains the script that was used to launch CTFFIND).

You can visualise all the Thon-ring images using the Display button, selecting `out: micrographs_ctf.star`. The zeros between the Thon rings in the experimental images should coincide with the ones in the model. Note that you can sort the display in order of defocus, maximum resolution, figure-of-merit, etc.

If you see CTF models that are not a satisfactory fit to the experimental Thon rings, you can delete the `.log` files for those micrographs, select the `CtfFind/job003` entry from the Finished jobs list, alter the parameters in the parameter-panel, and then re-run the job by clicking the Continue now button. Only those micrographs for which a `.log` file does not exist will be re-processed. You can do this until all CTF models are satisfactory. If this is not possible, or if you decide to discard micrographs because they have unsatisfactory Thon rins, you can use the Subset selection job-type to do this.

## 3.4 Manual particle picking

The next job-type Manual picking may be used to manually select particle coordinates in the (averaged) micrographs. We like to manually select at least several micrographs in order to get familiar with our data. We will use those manually selected particles to calculate reference-free 2D class averages, which will then be used as templates for automated particle picking of the entire data set.

Picking particles manually is a personal experience! If you don't like to pick particles in RELION, we also support coordinate file formats for Jude Short's XIMDISP [17] (with any extension); for XMIPP-2.4 [16] (with any extension); and for Steven Ludtke's E2BOXER.PY [18] (with a `.box` extension). If you use any of these, make sure to save the coordinate files as a text file in the same directory as from where you imported the micrographs (or movies), and with the same micrograph rootname, but a different (suffix+) extension as the micrograph, e.g. `Micrographs/006.box` or `Micrographs/006_pick.star` for micrograph `Micrographs/006.mrc`. You should then use the Import job-type and set `Node type:` to `2D/3D particle coordinates`. Make sure that the `Input Files:` field contains a linux wildcard, followed by the coordinate-file suffix, e.g. for the examples above you HAVE to give `Micrographs/*.box` or `Micrographs/*_pick.star`, respectively.

On the I/O tab of the Manual picking job-type, use the Browse button to select the `micrographs_ctf.star` file that was created in `CtfFind/job003`, ignore the Colors tab, and fill in the Display tab as follows:

- `Particle diameter (A):` `200`

  (This merely controls the diameter of the circle that is displayed on the micrograph.)

- `Scale for micrographs:` `0.5`

  (But this depends on your screen size)

- `Sigma contrast:` `3`

  (Micrographs are often best display with "sigma-contrast", i.e. black will be 3 standard deviation below the mean and white will be 3 standard deviations above the mean. The grey-scale is always linear from black to white. See the DisplayImages entry on the RELION wiki for more details)

- `White value:` `0`

  (Use this to manually set which value will be white. For this to work, `Sigma contrast` should be set to 0)

- `Black value:` `0`

  (Use this to manually set which value will be black. For this to work, `Sigma contrast` should be set to 0)

- `Lowpass filter (A):` `20`

  (Playing with this may help you to see particles better in very noisy micrographs)

- `Highpass filter (A):` `0`

  (This is sometimes useful to remove dark-¿light gradients over the entire micrograph)

- `Pixel size:` `3.54`

  (This is needed to calculate the particle diameter, and the low- and high-pass filters)

- `Scale for CTF image:` `1`

  (This merely controls how large the Thon-ring images will be when you click the `CTF` button for any micrograph)

Run the job by clicking the Run now! button and select a total of approximately 1,000 particles in a few micrographs. Left-mouse click for picking, middle-mouse click for deleting a picked particle, right-mouse click for a pop-up menu in which *you will need to save the coordinates!*. Note that you can always come back to pick more from where you left it (provided you saved the STAR files with the coordinates throught the pop-up menu), by selecting `ManualPick/job004` from the `Finished jobs` and clicking the Continue now button.

## 3.5 Particle extraction

Once you have a coordinate file for every micrograph that you want to pick particles from, you can extract the corresponding particles and gather all required metadata through the Particle extraction job-type. On the corresponding I/O tab, set:

- `micrograph STAR file:` `CtfFind/job003/micrographs_ctf.star`

  (Use the `Browse` button to select this file)

- `Coordinate-file suffix:` `ManualPick/job004/coords_suffix_manualpick.star`

  (Use the `Browse` button to select this file)

- `OR re-extract refined particles?` `No`

  (This is a new option in RELION-2.0. It allows you to use a `_data.star` file from a 2D cassification, 3D classification or 3D auto-refine job for re-extraction of only those particles in the STAR file. This may for example be useful if you had previously down-scaled your particles upon extraction, and after initial classifications you now want to perform refinements with the original-scaled particles. In RELION-1.4 this was cumbersome and required manual editing of the STAR files. In RELION-2.0, this is all done automatically through this option.)

- `Manually set pixel size?` `No`

  (This is only necessary when the input micrograph STAR file does NOT contain CTF information.)

On the extract tab you set the parameters for the actual particle extraction:

- `Particle box size (pix):` `100`

  (This should always be an even number!)

- `Invert contrast?` `Yes`

  (This makes white instead of black particles.)

- `Normalize particles?` `Yes`

  (We always normalize.)

- `Diameter background circle (pix):` `60`

  (Particles will be normalized to a mean value of zero and a standard-deviation of one for all pixels in the background area.The background area is defined as all pixels outside a circle with this given diameter in pixels (before rescaling). When specifying a negative value, a default value of 75% of the Particle box size will be used.)

- `Stddev for white dust removal:` `-1`
- `Stddev for black dust removal:` `-1`

  (We only remove very white or black outlier pixels if we actually see them in the data. In such cases we would use stddev values of 5 or so. In this data set there are no outlier pixels, so we don't correct for them, and leave the default values at -1 (i.e. don't do anything).

- `Rescale particles?` `No`

  (Down-scaling particles will speed up computations. Therefore, we often down-scale particles in the initial stages of processing, in order to speed up the initial classifications of suitable particles. Once our reconstructions get close to the Nyquist frequency, we then re-extract the particles without down-scaling. As explained above, this is much easier in RELION-2.0 than it was in RELION-1.4.)

As we will later on also use the same job-type to extract all autopicked particles, it may be a good idea to give this job an alias like `manualpick`. Ignore the Helix tab, and run using a single MPI processor.

Your particles will be extracted into MRC stacks (which always have an `.mrcs` extension in RELION) in a new directory called `Extract/job005/Micrographs/`. It's always a good idea to quickly check that all has gone OK by visualising your extracted particles selecting `out: particles.star` from the Display: button. Right-mouse clicking in the display window may be used for example to select all particles (`Invert selection`) and calculating the average of all unaligned particles (`Show average of selection`).

## 3.6 Making templates for auto-picking

To calculate templates for the subsequent auto-picking of all micrographs, we will use the 2D classification job-type. On the I/O tab, select the `Extract/job005/particles.star` file (using the Browse button), and on the CTF tab set:

- `Do CTF-correction?` `Yes`

  (We will perform full phase+amplitude correction inside the Bayesian framework)

- `Have data been phase-flipped?` `No`

  (This option is only useful if you pre-processed your data outside RELION)

- `Ignore CTFs until first peak?` `No`

  (This option is occasionally useful, when amplitude correction gives spuriously strong low-resolution components, and all particles get classified together in very few, fuzzy classes.)

On the Optimisation tab, set:

- `Number of classes:` `10`

  (For cryo-EM data we like to use on average at least approximately 100 particles per class. For negative stain one may use fewer, e.g. 20-50 particles per class)

- `Regularisation parameter T:` `2`

  (For the exact definition of T, please refer to [13]. For cryo-EM 2D classification we typically use values of T=2-3, and for 3D classification values of 3-4. For negative stain sometimes slightly lower values are better. In general, if your class averages appear very noisy, then lower T; if your class averages remain too-low resolution, then increase T. The main thing is to be aware of overfitting high-resolution noise.)

- `Number of iterations:` `25`

  (We hardly ever change this)

- `Use subsets for initial updates?` `No`

  (This options performs multiple maximisation steps during the first iteration. This may be useful to speed up 2D classifications of very large data sets, e.g. with hundreds of thousands or millions of particles.)

- `Mask diameter (A):` `200`

  (This mask will be applied to all 2D class averages. It will also be used to remove solvent noise and neighbouring particles in the corner of the particle images. On one hand, you want to keep the diameter small, as too much noisy solvent and neighbouring particles may interfere with alignment. On the other hand, you want to make sure the diameter is larger than the longest dimension of your particles, as you do not want to clip off any signal from the class averages.)

- `Mask individual particles with zeros?` `Yes`

- `Limit resolution E-step to (A):` `-1`

  (If a positive value is given, then no frequencies beyond this value will be included in the alignment. This can also be useful to prevent overfitting. Here we don't really need it, but it could have been set to 10-15A anyway. Difficult classifications, i.e. with very noisy data, often benefit from limiting the resolution.)

On the Sampling tab we hardly ever change the defaults. Five degrees angular sampling is enough for most projects, although some large icosahedral viruses may benefit from finer angular samplings. In that case, one could first run 25 iterations with a sampling of 5 degrees, and then continue that same run (using the Continue now button) for an additional five iteration (by setting

`Number of iterations: 30` on the Optimisation tab) with a sampling of say 2 degrees. For this data set, this is NOT necessary at all. It is useful to note that the same Continue now button may also be used to resume a job that has somehow failed before, in which case one would not change any of the parameters. For continuation of 2D classification, 3D classification, or 3D auto-refine jobs one always needs to specify the `_optimiser.star` file from the iteration from which one continues on the I/O tab.

Ignore the Helix tab, and on the Compute tab, set:

- `Use parallel disc I/O?` `Yes`

  (This way, all MPI slaves will read their own particles from disc. Use this option if you have a fast (parallel?) file system. Note that non-parallel file systems may not be able to handle parallel access from multiple MPI nodes. In such cases one could set this option to No. In that case, only the master MPI node will read in the particles and send them through the network to the MPI slaves.)

- `Number of pooled particles:` `3`

  (Particles are processed in individual batches by MPI slaves. During each batch, a stack of particle images is only opened and closed once to improve disk access times. All particle images of a single batch are read into memory together. The size of these batches is at least one particle per thread used. The nr_pooled_particles parameter controls how many particles are read together for each thread. If it is set to 3 and one uses 8 threads, batches of 3x8=24 particles will be read together. This may improve performance on systems where disk access, and particularly metadata handling of disk access, is a problem. It has a modest cost of increased RAM usage.)

- `Pre-read all particles into RAM?` `Yes`

  (If set to Yes, all particle images will be read into computer memory, which will greatly speed up calculations on systems with slow disk access. However, one should of course be careful with the amount of RAM available. Because particles are read in double-precision, it will take ( N * box_size * box_size * 4 / (1024 * 1024 * 1024) ) Giga-bytes to read N particles into RAM. If parallel disc I/O is set to Yes, then all MPI slaves will read in all particles. If parallel disc I/O is set to No, then only the master reads all particles into RAM and sends those particles through the network to the MPI slaves during the refinement iterations.)

- `Copy particles to scratch directory?`

  (This is useful if you don't have enough RAM to pre-read all particles, but you do have a fast (SSD?) scratch disk on your computer.)

- `Combine iterations through disc?` `No`

(This way all MPI nodes combine their data at the end of each iteration through the network. If the network is your main bottle-neck or somehow causing problems, you can set this option to No. In that case, all MPI nodes will write/read their data to disc.)

- `Use GPU acceleration?` `Yes`

  (If you have a suitable GPU, this job will go much faster.)

- `Which GPUs to use:` `0`

  (If you leave this empty, the program will try to figure out which GPUs to use, but you can explicitly tell it which GPU IDs , e.g. 0 or 1, to use. If you use multiple MPI-processors (probably not for this case!), you can run each MPI process on a specified GPU. GPU IDs for different MPI processes are separated by colons, e.g. 0:1:0:1 will run MPI process 0 and 2 on GPU 0, and MPI process 1 and 3 will run on GPU 1.)

On the Running tab, specify the `Number of MPI processors` and the `Number of threads` to use. The total number of requested CPUs, or cores, will be the product of the two values. Threads offer the advantage of more efficient RAM usage, whereas MPI parallelization scales better than threads. Often, for 3D classification and 3D auto-refine jobs you will probably want to use many threads in order to share the available RAM on each (multi-core) computing node. 2D classification is less memory-intensive, so you may not need so many threads. However, the points where communication between MPI processors (the bottle-neck in scalability there) becomes limiting in comparison with running more threads, is different on many different clusters, so you may need to play with these parameters to get optimal performance for your setup.

Because we will run more 2D classification jobs, it may again be a good idea to use a meaningful alias, for example `manualpick`. If you're using GPU-acceleration or a (reasonably sized) cluster, this job should be finished within minutes. Otherwise, on a desktop with only a few cores it may take 15-20 minutes. You can look at the resulting class averages using the Display: button to select `out: run_it025_model.star` from . On the pop-up window, you may want to choose to look at the class averages in a specific order, e.g. based on `rlnClassDistribution` (in reverse order, i.e. from high-to-low instead of the default low-to-high) or on `rlnAccuracyRotations`.

## 3.7 Selecting templates for auto-picking

Selection of suitable class average images is done in the Subset selection job-type. On the I/O tab select the `Class2D/manualpick/run_it025_model.star` file using the Browse button on the line with `Select classes from model.star:`.

On the Class options tab, give:

- `Re-center the class averages?` `Yes`

  (This is a new option in RELION-2.0: it allows automated centering of the 2D class averages. The images are centered based on their center-of-mass, and the calculations for this require that the particles are WHITE (not black). Re-centering is often necessary, as class averages may become non-centred in the 2D classification run, while auto-picking requires centered templates.)

- `Regroup the particles?` `No`

  (This option is useful when there are very few (selected) particles on individual micrographs, in which case the estimation of noise power spectra and scale factors become unstable. By default, the latter are calculated independently per micrograph. This option allows to grouping particles from multiple micrographs together in these calcutaions. RELION will warn you (in classification or auto-refine runs) when your groups become too small.)

An alias like `templates4autopick` may be a good idea. You may again want to order the class averages based on their `rlnClassDistribution`. Select 4-6 class averages that represent different views of your particle. Don't repeat very similar views, and don't include bad class averages. Selection is done by left-clicking on the class averages. You can save your selection of class averages from the right-click pop-up menu using the `Save selected classes` option.

## 3.8 Auto-picking

We will now use the selected 2D class averages as templates in the Auto-picking job-type. However, before we will run the auto-picking on all micrographs, we will need to optimise three of its main parameters on the autopicking tab: the `Picking threshold`, the `Minimum inter-particle distance`, and the `Maximum stddev noise`. This will be done on only a few micrographs in order to save time.

Therefore, we will first need a STAR file with a selection of only 2 or 3 micrographs. There are two ways of doing this in the Subset selection job-type. You can select `ManualPick/job004/coords_suffix_manualpick.star` on the line with `OR select from picked coords:`, and then use the selection boxes on the left to select only 2 micrographs. Note that the menu option `Invert selection` may be handy for this. When you've selected 2-4 suitable micrographs, for example high and low defocus and/or with thin or thick ice, then don't forget to select `Save selection` from the menu. Alternatively, from the I/O tab of the Subset selection job-type, you can select `CtfFind/job003/micrographs_ctf.star` in the line with `OR select from micrographs.star:`. Select your micrographs by left-clicking, and use the right-mouse button to select `Save STAR with selected images` from the pop-up menu. Whatever option you choose, perhaps an alias like `2mics4autopick` to the `Select` job would be meaningful?

Then, on the I/O tab of the Auto-picking job-type, set:

- `Input micrographs for autopick:` `Select/2mics4autopick/micrographs.star`
- `References:` `Select/templates4autopick/class_averages.star`
- `Or use Gaussian blob?` `No`

  (This allows to run autopicking without any references from a 2D classification run.)
- `Pixel size in micrographs (A):` `-1`

  (This will be ignored, as the input STAR file has the pixel-size information from the CtfFind job.)
- `Mask diameter (A):` `200`

  (You can use the same value as you did in the 2D classification job-type.)

On the References tab, set:

- `Lowpass filter references (A):` `20`

  (It is very important to use a low-pass filter that is significantly LOWER than the final resolution you aim to obtain from the data, to keep "Einstein-from-noise" artifacts at bay)
- `Highpass filter (A):` `-1`

  (If you give a positive value, e.g. 200, then the micrograph will be high-pass filtered prior to autopicking. This can help in case of strong grey-scale gradients across the micrograph.)
- `Pixel size in references (A):` `-1`

  (If a negative value is given, the references are assumed to be on the same scale as the input micrographs. If this is not the case, e.g. because you rescaled particles that were used to create the references upon their extraction, then provide a positive value with the correct pixel size in the references here.)
- `Angular sampling (deg):` `5`

  (This value seems to work fine in almost all cases.)
- `References have inverted contrast?` `Yes`

  (Because we have black particles in the micrographs, and the references we will use are white.)
- `Are References CTF corrected?` `Yes`

  (Because we performed 2D class averaging with the CTF correction.)

- `Ignore CTFs until first peak:` `No`

  (Only use this option if you also did so in the 2D classification job that you used to create the references.)

On the autopicking tab, set:

- `Picking threshold:` `0.8`

  (This is the threshold in the FOM maps for the peak-search algorithms. Particles with FOMs below this value will not be picked.)

- `Minimum inter-particle distance (A):` `200`

  (This is the maximum allowed distance between two neighbouring particles. An iterative clustering algorithm will remove particles that are nearer than this distance to each other. Useful values for this parameter are often in the range of 50-60% of the particle diameter.)

- `Maximum stddev noise:` `1.5`

  (This is useful to prevent picking in carbon areas, or areas with big contamination features. Peaks in areas where the background standard deviation in the normalized micrographs is higher than this value will be ignored. Useful values are probably in the range 1.0 to 1.2. Set to -1 to switch off the feature to eliminate peaks due to high background standard deviations.)

- `Write FOM maps?` `Yes`

  (See the explanation below.)

- `Read FOM maps?` `No`

  (See the explanation below.)

- `Shrink factor:` `0`

  (By setting shrink to 0, the autopicking program will downscale the micrographs to the resolution of the lowpass filter on the references. This will go much faster and require less memory, which is convenient for doing this tutorial quickly. Values between 0 and 1 will be the resulting fraction of the micrograph size. Note that this will lead to somewhat less accurate picking than using shrink=1, i.e. no downscaling. A more detailed description of this new parameter is given in the next subsection.)

- `Use GPU acceleration?` `Yes`

  (Only if you have a suitable GPU!)

- `Which GPUs to use:` `0`

  (If you leave this empty, the program will try to figure out which GPUs to use, but you can explicitly tell it which GPU IDs , e.g. 0 or 1, to use.

If you use multiple MPI-processors (not for this case!), you can run each MPI process on a specified GPU. GPU IDs for different MPI processes are separated by colons, e.g. 0:1:0:1 will run MPI process 0 and 2 on GPU 0, and MPI process 1 and 3 will run on GPU 1.)

Ignore the Helix tab, and run using a single MPI processor on the Running tab. Perhaps an alias like `optimise_params` would be meaningful? Using the 2kx2k micrographs in this tutorial, these calculations take about 5 minutes per micrograph on our computer. When using more common 4kx4k micrographs, this is typically approximately 5 minutes *per reference* and per micrograph. When using GPU-acceleration, the job completes in less than 10 seconds, so the writing/reading of FOM maps becomes less relevant.

The expensive part of this calculation is to calculate a probability-based figure-of-merit (related to the cross-correlation coefficient between each rotated reference and all positions in the micrographs. This calculation is followed by a much faster peak-detection algorithm that uses the threshold and minimum distance parameters mentioned above. Because these parameters need to be optimised, the program will write out so-called FOM maps as specified on the References tab. These are two large (micrograph-sized) files per reference. To avoid running into hard disc I/O problems, the autopicking program can only be run sequentially (hence the single MPI processor above) when writing out FOM maps.

Once the FOM maps have been written to disc they can be used to optimise the picking parameters much faster. First, examine the auto-picked particles with the current settings using the `coords_suffix_autopick` option from the Display: button of the job you just ran. Note that the display window will take its parameters (like size and sigma-contrast) from the last Manual picking job you executed. You can actually change those parameters in the Manual picking job-type, and save the settings use the option `Save job settings` from the top left `Jobs` menu. Do this, after you've set on the following on the Colors tab:

- `Blue<>red color particles?` `Yes`
- `MetaDataLabel for color:` `rlnAutopickFigureOfMerit`
- `STAR file with color label:`

  Leave this empty.
- `Blue value:` `1`
- `Red value:` `0`

Executing the job on the Running tab will produce a similar GUI with pick and CTF buttons as before. Open both micrographs from the display window, and decide whether you would like to pick more or less particles (i.e. decrease or increase the threshold) and whether they could be closer together or not (fot setting the minimum inter-particle distance). Note that each particle is colored from red (very low FOM) to blue (very high FOM). You can leave the display windows for both micrographs open, while you proceed with the next step.

Select the `AutoPick/optimise_params` job from the Finished jobs, and change the parameters on the autopicking tab. Also, change:

- `Write FOM maps?` `No`
- `Read FOM maps?` `Yes`

When executing by clicking the Continue now button, this will re-read the previously written FOM maps from disc instead of re-doing all FOM calculations. The subsequent calculation of the new coordinates will then be done in a few seconds. Afterwards, you can right-click in the micrograph display windows and select `Reload coordinates` from the pop-up menu to read in the new set of coordinates. This way you can quickly optimise the two parameters.

Have a play around with all three parameters to see how they change the picking results. Once you know the parameters you want to use for auto-picking of all micrographs, you click the Auto-picking option in the job-type browser on the top left of the GUI to select a job with a Run now! button. On the I/O tab, you replace the input micrographs STAR file with the 2 selected micrographs with the one from the original CTF estimation job (`CtfFind/job003/micrographs_ctf.star`). Leave everything as it was on the References tab, and on the autopicking tab set:

- `Picking threshold:` `0.4`
- `Minimum inter-particle distance (A):` `110`
- `Maximum stddev noise:` `1.1`
- `Write FOM maps?` `No`
- `Read FOM maps?` `No`

This time, the job may be run in parallel (as no FOM maps will be written out). On the Running tab, specify the number of cores to run auto-picking on. The maximum useful number of MPI processors is the number of micrographs in the input STAR file.

Note that there is an important difference in how the Continue now button works depending on whether you read/write FOM maps or not. When you either write or read the FOM maps and you click Continue now, the program will re-pick all input micrographs (typically only a few). However, when you do not read, nor write FOM maps, i.e. in the second job where you'll autopick all micrographs, upon clicking the Continue now button, only those micrographs that were not autopicked yet will be done. This is useful in the iterative running of scheduled jobs, e.g. for on-the-fly data processing during your microscopy session. Also see section 2.3 for more details. If you want to-repick all micrographs with a new set of parameters (instead of doing only unfinished micrographs), then click the Auto-picking entry on the jobtype-browser on the left to get a Run now! button instead, which will then make a new output directory.

You can again check the results by clicking the `coords_suffix_autopick` option from the Display: button. Some people like to manually go over all micrographs to remove false positives. For example, carbon edges or high-contrast artifacts on the micrographs are often mistaken for particles. You can do this for each micrograph using the pop-up window from the Display: button. Remove particles using the middle-mouse button; you can hold it down to remove many false positives in larger areas. Remember to save the new coordinates using the right-mouse click pop-up menu!

Once you're happy with the overall results, in order to save disc space you may want to delete the FOM-maps that were written out in the first step. You can use the `Gentle clean` option from the Job actions button to do that conveniently.

Once you are happy with your entire set of coordinates, you will need to rerun a Particle extraction job, keeping everything as before, but change the `Input coordinates` for the newly generated, autopick ones. This will generate your initial single-particle data set that will be used for further refinements below. Perhaps an alias like `allmics_autopicked` would be meaning ful?

### 3.8.1 The shrink parameter

To enable faster processing, RELION implements a filtering option of the micrographs through the command-line argument `--shrink`. The simplest way to use it is to simply use `--shrink 0`. But it can be used with much more control. If this is desired, it works in the following way:

The default value for `--shrink` is 1.0, which has no effect and does not filter the micrographs at all. This is identical behaviour to versions prior to the RELION-2.0 autopicker.

- `--shrink` $val = 0$ results in a micrograph lowpassed to `--lowpass`, the same as that of the reference templates. This is recommended use for single-particle analysis (but not for helical picking).
- `--shrink` $0 < val \leq 1$ results in a size $= val \cdot micrograph\_size$, i.e. $val$ is a scaling factor applied to the micrographs original size.
- `--shrink` $val > 1$ results in a size $= val - (val \mod 2)$, i.e. the smallest even integer value lower than val. This is then a way to give the micrographs a specific (even) integer size.

If the new size is smaller than `--lowpass`, this results in a *non-fatal* warning, since this limits resolution beyond the requested (or default) resolution. Because the RELION autopicker does many FFTs, the size of micrographs is now also automatically adjusted to avoid major pitfalls of FFT computation. A note of this and how it can be overridden is displayed in the initial output of each autopicking the user performs.

## 3.9 Particle sorting

RELION also has a functionality to quickly sort particles based on the difference between each particle and their aligned, CTF-corrected reference. The sorting program is accessible from the Particle sorting job-type and can be run whenever references are available for each particle, i.e. after Auto-picking, 2D classification, 3D classification or 3D auto-refine. To demonstrate its use, we will now run it after the auto-picking.

On the I/O tab, set:

- `Input particles to be sorted:` `Extract/allmics_autopicked/particles.star`

  (The file that was generated by the extraction program after the autopicking. In case of 2D classification, 3D classification or 3D auto-refine jobs, the `_data.star` from the last iteration should be given.)

- `Are these from an Extract job?` `Yes`

  (Set this to No, when the input particles come from a 2D classification, 3D classification or 3D auto-refine job.)

- `Autopicking references:` `Select/templates4autopick/class_averages.star`

  (Provide the references that were used for the autopicking job that gave the coordinates for the extracted particles.)

On the CTF tab, set:

- `Are References CTF corrected?` `Yes`
- `Ignore CTFs until first peak?` `No`

This program runs very quickly (in minutes) so does not need to be run with many MPI processors. Perhaps you could use `after_autopick` as an alias? This job will produce a file called `particles_sort.star` in the output directory, which will be the same as the input file, but with an additional column called `rlnParticleSelectZScore`. You can then select this file in the Subset selection job-type, on the line that says `OR select from particles.star`. In the pop-up display window, you can then visualise the particles based on the new column. There will be a strong tendency to have "good" particles at the top and "bad" particles at the bottom of the display. One could scroll up from the bottom and decide a given point from which below all particles need to be discarded and use the right-mouse pop-up menu to select `Select all above` and/or (de-)select particles on a one-by-one basis. Don't forget to save the output STAR file using the right-mouse pop-up menu.

# 4 Reference-free 2D class averaging

We almost always use reference-free 2D class averaging to throw away bad particles. Although we often try to only include good particles for the particle extraction step in the previous section (for example by manually supervising the auto-picking results, and by sorting the extracted particles), most of the times there are still particles in the data set that do not belong there. Because they do not average well together, they often go to relatively small classes that yield ugly 2D class averages. Throwing those away then becomes an efficient way of cleaning up your data.

## 4.1 Running the job

Most options will remain the same as explained when we were generating templates for the auto-picking in the previous section, but on the I/O tab of the 2D classification job-type, set:

- `Input images STAR file:` `Select/after_sorting/particles.star`

Because we now have approximately 10,000 particles, we can use many more classes. Therefore, on the Optimisation tab, set

- `Number of classes:` `100`

This job is much more expensive than the previous one (the algorithm scales linearly with the number of particles and the number of classes), so run it in parallel on your cluster. If you have one or GPUs, you can still run this job within half an hour or so. You could use an alias like `after_sorting`.

After the job has finished, we can launch a Subset selection job, with the `_model.star` file from this run as input. An alias like `class2d_aftersort` may be meaningful. Now select all nice-looking classes by clicking on them (and/or using the right-mouse pop-up menu option `Select all classes above`). At this point, if you would have used a low threshold in the auto-picking procedure, you should be very wary of "Einstein-from-noise" classes, which look like low-resolution ghosts of the templates used to pick them, on which high-resolution noise may have accumulated. Avoid those in the selection. After all good classes have been selected use the right-mouse pop-up menu option to save the selection.

Note that this procedure of selecting good classes may be repeated several times. Also, note that after the 2D classification one could re-run the sorting algorithm to identify remaining outliers in the data.

## 4.2 Analysing the results in more detail

*If you are in a hurry to get through this tutorial, you can skip this sub-section. It contains more detailed information for the interested reader.*

For every iteration of 2D or 3D classification RELION performs, it writes out a set of files. For the last iteration of our 2D class averaging calculation these are:

- `Class2D/after_sorting/run_it025_classes.mrcs` is the MRC stack with the resulting class averages. These are the images that will be displayed in the RELION GUI when you select the `_model.star` file from the Display: button on the main GUI. Note that RELION performs full CTF correction (if selected on the GUI), so your class averages are probably white on a black background. If the data is good, often they are very much like projections of a low-pass filtered atomic model. The quality of your 2D class averages are a very good indication of how good your 3D map will become. We like to see internal structure within projections of protein domains, and the solvent area around you particles should ideally be flat. Radially extending streaks in the solvent region are a typical sign of overfitting. If this happens, you could try to limit the resolution in the E-step of the 2D classification algorithm.

- `Class2D/after_sorting/run_it025_model.star` contains the model parameters that are refined besides the actual class averages (i.e. the distribution of the images over the classes, the spherical average of the signal-to-noise ratios in the reconstructed structures, the noise spectra of all groups, etc. Have a look at this file using the `less` command. In particular, check the distribution of particles over each class in the table `data_model_classes`. If you compare this with the class averages themselves, you will see that particles with few classes are low-resolution, while classes with many particles are high-resolution. This is an important feature of the Bayesian approach, as averaging over fewer particles will naturally lead to lower signal-to-noise ratios in the average. The estimated spectral signal-to-noise ratios for each class are stored in the `data_model_class_N` tables, where N is the number of each class. Likewise, the estimated noise spectra for each group are stored in the tables called `data_model_group_N`. The table `data_model_groups` stores a refined intensity scale-factor for each group: groups with values higher than one have a stronger signal than the average, relatively low-signal groups have values lower than one. These values are often correlated with the defocus, but also depend on accumulated contamination and ice thickness.

- `Class2D/after_sorting/run_it025_data.star` contains all metadata related to the individual particles. Besides the information in the input `particles.star` file, there is now additional information about the optimal orientations, the optimal class assignment, the contribution to the log-likelihood, etc. Note that this file can be used again as input for a new refinement, as the STAR file format remains the same.

- `Class2D/after_sorting/run_it025_optimiser.star` contains some general information about the refinement process that is necessary for restarting an unfinished run. For example, if you think the process did not converge yet after 25 iterations (you could compare the class averages from iterations 24 and 25 to assess that), you could select this job in the Finished jobs panel, and on the I/O tab select this file for `Continue from here`, and then set `Number of iterations: 40` on the Optimisation tab. The job will then restart at iteration 26 and run until iteration 40. You might also choose to use a finer angular or translational sampling rate on the Sampling tab. Another useful feature of the optimiser.star files is that it's first line contains a comment with the exact command line argument that was given to this run.

- `Class2D/after_sorting/run_it025_sampling.star` contains information about the employed sampling rates. This file is also necessary for restarting.

## 4.3 Making groups

*If you are in a hurry to get through this tutorial, you can skip this sub-section. It contains more detailed information for the interested reader.*

RELION groups particles together to do two things: estimate their average noise power spectrum and estimate a single-number intensity scale factor that describes differences in overall signal-to-noise ratios between different parts of the data, e.g. due to ice thickness, defocus or contamination.

The default behaviour is to treat all particles from each micrograph as a separate group. This behaviour is fine if you have many particles per micrograph, but when you are using a high magnification, your sample is very diluted, or your final selection contains only a few particles per micrograph, then the estimation of the intensity scale factor (and the noise spectra) may become unstable. We generally recommend to have at least 10-20 particles in each group, but do note that initial numbers of particles per group may become much smaller after 2D and 3D classification.

In cases with few particles per micrograph we recommend to group particles from multiple micrographs together. For this purpose, the GUI implements a convenient functionality in the Subset selection job-type: when selecting a `_model.star` file on the I/O tab, one can use `Regroup particles? Yes` and `Approximate nr of groups: 5` on the Class options tab to re-group all particles into 5 groups. (The actual number may vary somewhat from the input value, hence the "Approximate" on the input field.) This way, complicated grouping procedures in previous releases of RELION may be avoided. As the micrographs in this tutorial do contain sufficient particles, we will not use this procedure now.

Please note that the groups in RELION are very different from defocus groups that are sometimes used in other programs. RELION will always use per-particle (anisotropic) CTF correction, irrespective of the groups used.

# 5 De novo 3D model generation

RELION uses a Stochastic Gradient Descent (SGD) algorithm to generate a 3D initial model *de novo* from the 2D particles. This implementation is very similar to the one described for the cryoSPARC program [8]. Provided you have a relatively even distribution of viewing directions, and your data were good enough to yield detailed clss averages in 2D classification, this algorithm is very likely to yield a suitable, low-resolution model that can subsequently be used for 3D classification or 3D auto-refinement.

## 5.1 Selecting a suitable particle subset

First, run a Subset selection job, and use `Class2D/after_sorting/run_it025_model.star` at the option to `Select classes from model.star`. We used a `for_inimodel` alias. In the resulting pop-up Display window, (reverse) sort the classes on `rlnClassDistribution`, and use the option at the bottom to set `Max nr selected parts per class` to `200`. This will randomly select 200 particles from each class that is selected. This will result in a more even angular distribution of particles in the resulting particles STAR file. For the subsequent SGD calculations, it is typically enough to use several to ten thousand of particles. Using more might sometimes lead to worse results. Probably the most important thing to consider when selecting particles for SGD calculations, is to include as many different views as possible. We selected 16 different 2D classes, which thus resulted in `16*200=3200` particles.

## 5.2 Running the job

After selection of your 2D classes, select the `Select/for_inimodel/particles.star` file on the I/O tab of the 3D initial model jobtype. Leave the details on the CTF and Sampling tabs, and fill in the SGD tab as follows:

- `Mask diameter (A)` `200`

  (The same as before).

- `Symmetry` `C1`

  (If you don't know what the symmetry is, it is probably best to start with a C1 reconstruction.)

- `Number of iterations:` `3`

  (Make the number of iterations times the number of particles in the input STAR file approximately 10,000.)

- `SGD subset size:` `200`

(We hardly ever change this.)

- `Write-out frequency subsets` 10

  (This will write out intermediate reconstructions during each iteration, allowing you to monitor progress.)

- `Limit resolution SGD to (A)` 20

  (As the SGD is not regularised, it is important to limit the maximum resolution during refinement. Often values around 20-30 A work well. More difficult and very large structures may benefit from even stronger limits.)

- `SGD increased noise variance half-life:` -1

  (When set to a positive value, the initial estimates of the noise variance will internally be multiplied by 8, and then be gradually reduced, having 50% after this many particles have been processed. By default, this option is switched off by setting this value to a negative number. In some difficult cases, switching this option on helps. In such cases, values around 1000 have found to be useful.)

On the Compute tab, optimise things for your system. You may well be able to pre-read the few thousand particles into RAM. GPU acceleration will also yield speedups, though multiple maximisation steps during each iteration will slow things down compared to standard 2D or 3D refinements or classifications. Using 4 GPU cards, this run took approximately 2-3 minutes per iteration on our system.

## 5.3 Analysing the results

Look at the output volume with a 3D viewer like UCSF Chimera. If you recognise additional point group symmetry, then you have 2 options. Firstly, you could rotate the molecule (for example in UCSF Chimera, using the `resample onGrid` option) to orient the symmetry axes according to the conventions in RELION. Secondly, you may re-run the SGD algorithm with the correct point-group symmetry. Perhaps surprisingly, it may actually be more difficult to find a good solution when imposing symmetry, and this is also the case with this data set. (A better betagal data set does allow SGD refinement in D2...) In our precalculated results, we have included also a run with alias `symD2`, which has converged to a suboptimal solution. One way to distinguish suboptimal solutions from the correct one is to project the resulting map into many different directions, and to see whether the projections match the 2D class averages calculated before. This can be done by going as follows:

```
cd InitialModel/symC1
mkdir Projections
relion_project --i run_it003_class001.mrc --o Projections/proj --nr_uniform 50
```

```
relion_display --i Projections/proj.star
```

Comparing the projections from the `symC1` and the `symD2` runs with the 2D class averages inside `Class2D/after_sorting`, it is clear that the C1 model is better. Therefore, we used UCSF Chimera to have it's smmetry axes (approximately) parallel to the XYZ axes. The procedure we used for that was to read in `InitialModel/symC1/run_it003_class001.mrc`; use the `duplicate` option under the `File` menu of the `Volume viewer`. Then to deactivate map #1 on the `Model Panel`; rotate map #0 to have it's symmetry axis more or less parallel to the XYZ axes; and then to use in the `Command Line` entry field: `run_it003_class001.mrc`; and save this model in the `InitialModel/symC1/` directory with the name `inimodel.mrc`.

One can check the result of the alignment procedure by imposing D2 symmetry on the resulting map:

```
relion_image_handler --i inimodel.mrc --o inimodel_symD2.mrc --sym D2
```

The output map should be similar to the input map.

# 6 Unsupervised 3D classification

All data sets are heterogeneous! The question is how much you are willing to tolerate. RELION's 3D multi-reference refinement procedure provides a powerful unsupervised 3D classification approach.

## 6.1 Running the job

Unsupervised 3D classifcation may be run from the 3D classification job-type. On the I/O tab set:

- `Input images STAR file:` `Select/class2d_aftersort/particles.star`
- `Reference map:` `InitialModel/symC1/inimodel_symD2.mrc`

  (Note that this map does not appear in the Browse button as it is not part of the pipeline. You can either type it's name into the entry field, or first import the map using the Import jobtype. Also note that, because we wil be running in symmetry C1, we could have also chosen to use the non-symmetric `inimodel.mrc`.)

- `Reference mask (optional):`

  (Leave this empty. This is the place where we for example provided large/small-subunit masks for our focussed ribosome refinements. If left empty, a spherical mask with the particle diameter given on the Optimisation tab will be used. This introduces the least bias into the classification.)

On the Reference tab set:

- `Ref. map is on absolute greyscale:` `Yes`

  (Given that this map was reconstructed from this data set, it is already on the correct greyscale. Any map that is not reconstructed from the same data in RELION should probably be considered as not being on the correct greyscale.)

- `Initial low-pass filter (A):` `50`

  (One should NOT use high-resolution starting models as they may introduce bias into the refinement process. As also explained in [12], one should filter the initial map as much as one can. For ribosome we often use 70Å, for smaller particles we typically use values of 40-60Å.)

- `Symmetry:` `C1`

  (Although we know that this sample has D2 symmetry, it is often a good idea to perform an initial classification without any symmetry, so bad particles can get separated from proper ones, and the symmetry can be verified in the reconstructed maps.)

On the CTF tab set:

- `Do CTF correction?` `Yes`
- `Has reference been CTF-corrected?` `Yes`

  (As this model was made using CTF-correction in the SGD.)
- `Have data been phase flipped?` `No`
- `Ignore CTFs until first peak?` `No`

  (Only use this option if you also did so in the 2D classification job that you used to create the references.)

On the Optimisation tab set:

- `Number of classes:` `4`

  (Using more classes will divide the data set into more subsets, potentially describing more variability. The computational costs scales linearly with the number of classes, both in terms of CPU time and required computer memory.)
- `Number of iterations:` `25`

  (We typically do not change this.)
- `Regularisation parameter T:` `4`

  For the exact definition of T, please refer to [13]. For cryo-EM 2D classification we typically use values of T=1-2, and for 3D classification values of 2-4. For negative stain sometimes slightly lower values are better. In general, if your class averages appear noisy, then lower T; if your class averages remain too-low resolution, then increase T. The main thing isto be aware of overfitting high-resolution noise. We happened to use a value of 2 in our pre-calculated results. Probably a value of 4 would have worked equally well...
- `Mask diameter (A):` `200`

  (Just use the same value as we did before in the 2D classification job-type.)
- `Mask individual particles with zeros?` `Yes`
- `Limit resolution E-step to (A):` `-1`

  (If a positive value is given, then no frequencies beyond this value will be included in the alignment. This can also be useful to prevent overfitting. Here we don't really need it, but it could have been set to 10-15A anyway.)

On the Sampling tab one usually does not need to change anything (only for large and highly symmetric particles, like icosahedral viruses, does one typically use a 3.7 degree angular sampling at this point). Ignore the Helix tab, and fill in the Compute tab like you did for the previous 2D-classification. Again, on the

Running tab, one may specify the `Number of MPI processors` and `threads` to use. As explained for the 2D classification job-type, 3D classification takes more memory than 2D classification, so often more threads are used. However, in this case the images are rather small and RAM-shortage may not be such a big issue. Perhaps you could use an alias like `first_exhaustive`, to indicate this is our first 3D classification and it uses exhaustive angular searches?

When analysing the refined reconstructions, it is extremely useful to also look at them in slices, not only as a thresholded map in for example UCSF Chimera. In the slices view you will get a much better impression of unresolved heterogeneity, which will show up as fuzzy or streaked regions in the slices. Slices also give a good impression of the flatness of the solvent region. Use the Display: button and select any of the reconstructions from the last iteration to open a slices-view in RELION.

When looking at your rendered maps in 3D, e.g. using UCSF Chimera, it is often a good idea to fit them all into the best one, as maps may rotate slightly during refinement. In Chimera, we use the `Tools -> Volume Data -> Fit in Map` tool for that. For looking at multiple maps alongside each other, we also like the `Tools -> Structure Comparison -> Tile Structures` tool, combined with the `independent` center-of-rotation method on the `Viewing` window.

As was the case for the 2D classification, one can again use the Subset selection to select a subset of the particles assigned to one or more classes. On the I/O tab select the `_model.star` file from the last iteration. The resulting display window will show central slices through the 4 refined models. Select the best classes, and save the corresponding particles using the right-mouse pop-up menu. Use an alias like `class3d_first_exhaustive`.

## 6.2 Analysing the results in more detail

*Again, if you are in a hurry to get through this tutorial, you can skip this subsection. It contains more detailed information for the interested reader.*

The output files are basically the same as for the 2D classification run (we're actually using the same code for 2D and 3D refinements). The only difference is that the map for each class is saved as a separate MRC map, e.g. `run_it025_class00?.mrc`, as opposed to the single MRC stack with 2D class averages that was output before.

As before, smaller classes will be low-pass filtered more strongly than large classes, and the spectral signal-to-noise ratios are stored in the `data_model_class_N` tables (with $N = 1, \dots, K$) of the `_model.star` files. Perhaps now is a good time to introduce two handy scripts that are useful to extract any type of data from STAR files. Try typing:

```
relion_star_printtable Class3D/first_exhaustive/run_it025_model.star
  data_model_class_1 rlnResolution rlnSsnrMap
```

It will print the two columns with the resolution (`rlnResolution`) and the spectral signal-to-noise ratio (`rlnSsnrMap`) from table `data_model_class_1` to the screen. You could redirect this to a file for subsequent plotting in your favourite program. Alternatively, if `gnuplot` is installed on your system, you may type:

```
relion_star_plottable Class3D/first_exhaustive/run_it025_model.star
  data_model_class_1 rlnResolution rlnSsnrMap
```

To check whether your run had converged, (as mentioned above) you could also monitor:

```
grep _rlnChangesOptimalClasses Class3D/first_exhaustive/run_it???_optimiser.star
```

As you may appreciate by now: the STAR files are a very convenient way of handling many different types of input and output data. Linux shell commands like `grep` and `awk`, possibly combined into scripts like `relion_star_printtable`, provide you with a flexible and powerful way to analyze your results.

# 7 High-resolution 3D refinement

Once a subset of sufficient homogeneity has been selected, one may use the 3D auto-refine procedure in RELION to refine this subset to high resolution in a fully automated manner. This procedure employs the so-called *gold-standard* way to calculate Fourier Shell Correlation (FSC) from independently refined half-reconstructions in order to estimate resolution, so that self-enhancing over-fitting may be avoided [15]. Combined with a novel procedure to estimate the accuracy of the angular assignments [14], it automatically determines when a refinement has converged. Thereby, this procedure requires very little user input, i.e. it remains objective, and has been observed to yield excellent maps for many data sets. Another advantage is that one typically only needs to run it once, as there are hardly any parameters to optimize.

## 7.1 Running the job

Select the 3D auto-refine job-type on the RELION GUI. The input parameters on the GUI largely remain the same as for the 3D classification run type, although some of them are no longer available. The orientational sampling rates on the Sampling tab will only be used in the first few iterations, from there on the algorithm will automatically increase the angular sampling rates until convergence. Therefore, for all refinements with less than octahedral or icosahedral symmetry, we typically use the default angular sampling of 7.5 degrees, and local searches from a sampling of 1.8 degrees. Only for higher symmetry refinements, we use 3.7 degrees sampling and perform local searches from 0.9 degrees.

As of version 2.0, RELION has an option called `Use solvent-flattened FSCs` on the Optimisation tab. If set to `Yes`, the auto-refinement will use a solvent-correction on the gold-standard FSC curve, very much like the one used in the Post-processing job-type. Therefore, for this to work, one has to also provide a mask on the I/O tab. This option is particularly useful when the protein occupies a relatively small volume inside the particle box, e.g. with very elongated molecules or when one focusses refinement on a small part using a mask. In such cases, non-masked gold-standard FSCs would under-estimate the resolution of the refinement. For our refinement now, leave this option at `No` (as we're not providing a mask anyway).

Just select the `Select/class3d_first_exhaustive/particles.star` as the input STAR file, and we could use one of the refined maps from the 3D classification run (e.g. `Class3D/first_exhaustive/run_it025_class002.mrc`) as initial model. Alternatively, we could use the same `ininmodel_symD2.mrc` model we generated before; it would probably make little difference. Again, it is best to low-pass filter the initial map strongly to prevent bias towards high-frequency components in the map, and to maintain the "gold-standard" of completely independent refinements at resolutions higher than the initial one. Let's again use 50Å. Now that we see very clearly that the molecule is symmetric, let's impose D2 symmetry in the refinement (so make sure that whatever reference model you are using has it's symmetry axes according to the RELION conventions!).

As the MPI nodes are divided between one master (who does nothing else than bossing the others around) and two sets of slaves who do all the work on the two half-sets, it is most efficient to use an odd number of MPI processors. Memory requirements may increase significantly at the final iteration, as all frequencies until Nyquist will be taken into account, so for larger sized boxes than the ones in this test data set you may want to run with as many threads as you have cores on your cluster nodes. Perhaps an alias like `after_first_class3d` would be meaningful?

## 7.2 Analysing the results

Also the output files are largely the same as for the 3D classification run. However, at every iteration the program writes out two `run_it0??_half?_model.star` and two `run_it0??_half?_class001.mrc` files: one for each independently refined half of the data. Only upon convergence a single `run_model.star` and `run_class001.mrc` file will be written out (without `_it0??` in their names). Because in the last iteration the two independent half-reconstructions are joined together, the resolution will typically improve significantly in the last iteration. Because the program will use all data out to Nyquist frequency, this iteration also requires more memory and CPU.

Note that the automated increase in angular sampling is an important aspect of the auto-refine procedure. It is based on signal-to-noise considerations that are explained in [14], to estimate the accuracy of the angular and translational assignments. The program will not use finer angular and translational sampling rates than it deems necessary (because it would not improve the results). The estimated accuracies and employed sampling rates, together with current resolution estimates are all stored in the `_optimiser.star` and `_model.star` files, but may also be extracted from the stdout file. For example, try:

```
grep Auto Refine3D/after_first_class3d/run.out
```

# 8 Mask creation & Postprocessing

After performing a 3D auto-refinement, the map needs to be sharpened. Also, the gold-standard FSC curves inside the auto-refine procedures only use unmasked maps (unless you've used the option `Use solvent-flattened FSCs`). This means that the actual resolution is under-estimated during the actual refinement, because noise in the solvent region will lower the FSC curve. RELION's procedure for B-factor sharpening and calculating masked FSC curves [2] is called "post-processing". First however, we'll need to make a mask to define where the protein ends and the solvent region starts. This is done using the Mask Creation job-type.

## 8.1 Making a mask

On the I/O tab, select the output map from the finished 3D auto-refine job: `Refine3D/after_first_class3d/run_class001.mrc`. On the Mask tab set:

- `Lowpass filter map (A):` 15

  (A 15Å low-pass filter seems to be a good choice for smooth solvent masks.)

- `Pixel size (A):` 3.54

  (This value will only be used to calculate the low-pass filter.)

- `Initial binarisation threshold:` 0.02

  (This should be a threshold at which rendering of the low-pass filtered map in for example Chimera shows absolutely no noisy spots outside the protein area. Move the threshold up and down to find a suitable spot. Remember that one can use the command-line program called `relion_image_handler` with the options `--lowpass 15 --angpix 3.54` to get a low-pass filtered version of an input map. Often good values for the initial threshold are around 0.01-0.04.)

- `Extend binary map this many pixels:` 2

  (The threshold above is used to generate a black-and-white mask. The white volume in this map will be grown this many pixels in all directions. Use this to make your initial binary mask less tight.)

- `Add a soft-edge of this many pixels:` 3

  (This will put a cosine-shaped soft edge on your mask. This is important, as the correction procedure that measures the effect of the mask on the FSC curve may be quite sensitive to too sharp masks. As the mask generation is relatively quick, we often play with the mask parameters to get the best resolution estimate.)

Ignore the Helix tab and use an alias like `first3dref_th002_ext2_edg3`. You can look at slices through the resulting mask using the Display: button, or you can load the mask into UCSF Chimera. The latter may be a good idea, together with the map from the auto-refine procedure, to confirm that the masks encapsulates the entire structure, but does not leave a lot of solvent inside the mask.

## 8.2 Postprocessing

Now select the Post-processing job-type, and on the I/O tab, set:

- `One of the 2 unfiltered half-maps:` `Refine3D/after_first_class3d/run_half1_class001_unfil.mrc`
- `Solvent mask:` `MaskCreate/first3dref_th002_ext2_edg3/mask.mrc`
- `Calibrated pixel size (A):` `3.54`

  (Sometimes you find out when you start building a model that what you thought was the correct pixel size, in fact was off by several percent. Inside RELION, everything up until this point was still consistent. so you do not need to re-refine your map and/or re-classify your data. All you need to do is provide the correct pixel size here for your correct map and final resolution estimation.)

On the Sharpen tab, set:

- `MTF of the detector (STAR file):` `mtf_falcon2_300kV.star`

  (The manufacturer may provide you with a suitable MTF curve for your own detector, which you will need to store in the same format as the `mtf_falcon_300kV_down4.star` file that is stored in your project directory. You can also download a few commonly curves from the Analyse results entry on the RELION wiki.)

- `Estimate B-factor automatically:` `Yes`

  (This procedure is based on the [10] paper and will therefore need the final resolution to extend significantly beyond 10 Å. If your own map does not reach that resolution, you may want to use your own "ad-hoc" B-factor instead.)

- `Lowest resolution for auto-B fit (A):` `10`

  (This is usually not changed.)

- `Use your own B-factor?` `No`

On the Filter tab, set:

- `Skip FSC-weighting?` `No`

  (This option is sometimes useful to analyse regions of the map in which the resolution extends *beyond* the overall resolution of the map. This is not the case now.)

Run the job (no need for a cluster, as this job will run very quickly) and use an alias like `first3dref`. Using the Display button, you can display slizes through the postprocessed map and a PDF with the FSC curves and the Guinier plots for this structure. You can also open the `PostProcess/first3dref/postprocess.mrc` map in Chimera, where you will see that it is much easier to see where all the alpha-helices are then in the converged map of the 3D auto-refine procedure. The resolution estimate is based on the phase-randomization procedure as published previously [2]. Make sure that the FSC of the phase-randomized maps (the red curve) is more-or-less zero at the estimated resolution of the postprocessed map. If it is not, then your mask is too sharp or has too many details. In that case use a stronger low-pass filter and/or a wider and more softer mask in the Mask creation step above, and repeat the postprocessing. Actually, you may want to make the mask you just made a bit softer, so perhaps try an edge of 5 pixels as well? If you use the Continue now button for that, the previous results will be overwritten. Click on the job-type browser on the left to generate a Run now! button, which will create a new output directory.

## 8.3 Correcting handedness

Careful inspection of the map may indicate that the handedness is incorrect. Remember that it is impossible to determine absolute handedness from a data set without tilting the microscopy stage. The SGD algorithm in the 3D initial model jobtype therefore has a 50% chance of being in the opposite hand. In our precalculated results, this is indeed the case. One may flip the handedness of the two unfiltered half-maps as follows:

```
cd Refine3D/after_first_class3d/
relion_image_handler --i run_half1_class001_unfil.mrc \
  --o run_fliphand_half1_class001_unfil.mrc --flipX
relion_image_handler --i run_half2_class001_unfil.mrc \
  --o run_fliphand_half2_class001_unfil.mrc --flipX
cd ../../
```

And one may then re-run the Post-processing job, giving these maps as input.

## 9 Local-resolution estimation

The estimated resolution from the post-processing program is a global estimate. However, a single number cannot describe the variations in resolution that are often observed in reconstructions of macromolecular complexes. Alp Kucukelbir and Hemant Tagare wrote a nifty program to estimate the variation in resolution throughout the map [7]. RELION implements a wrapper to this program through the Local resolution job-type. Alternatively, one can choose to run a post-processing-like procedure with a soft spherical mask that is moved around the entire map. In the example below, we use the latter.

On the I/O tab set:

- `One of the two unfiltered half-maps:` `Refine3D/after_first_class3d/run_half1_class001_unfil.mrc`
- `User-provided solvent mask:` `MaskCreate/first3dref_th002_ext2_edg3/mask.mrc`
- `Calibrated pixel size:` `3.54`

  (Sometimes you find out when you start building a model that what you thought was the correct pixel size, in fact was off by several percent. Inside RELION, everything up until this point was still consistent. so you do not need to re-refine your map and/or re-classify your data. All you need to do is provide the correct pixel size here for your correct map and final resolution estimation.)

On the ResMap tab set `Use ResMap?` to `No`; on the Relion tab set:

- `Use Relion?` `Yes`
- `User-provided B-factor:` `-250`

  (This value will be used to also calculate a locally-filtered and sharpened map. Probably you want to use a value close to the one determined automatically during the Post-processing job.)

- `MTF of the detector (STAR file):` `mtf_falcon2_300kV.star`

  (The same as for the Post-processing job.)

Use an alias like `first3dref`. The RELION procedure is not very fast. It took approximately 3 minutes for this small case. The output is a file called `LocalRes/first3dref/relion_locres.mrc` that may be used in UCSF Chimera to color the `Postprocess/first3dref/postprocess.mrc` map according to local resolution. This is done using `Tools -> Volume data -> Surface color`, and then select `by volume data value` and browse to the `resmap` file. Unique to the RELION option is the output of a locally-filtered (and sharpened map), which may be useful to describe the overall variations in map quality in a single map. This map is saved as `LocalRes/first3dref/relion_locres_filtered.mrc` and can be visualised directly in UCSF Chimera (and optionally also colored by local resolution as before).

In this particular case, because the data were downsampled and because beta-galactosidase isn't a very flexible molecule, the local-resolution variation isn't very exciting to look at. But it's good to know that anyway!

# 10 Movie-processing & particle polishing

## 10.1 Getting organised

If you performed the beam-induced motion correction through the RELION pipeline, as explained in section 3.2, then you're already all set to go. If you performed motion correction outside the RELION workflow, then you'll need to position all your (corrected) movies in the same directory as the micrographs you imported in the beginning (e.g. the `Micrographs/` directory in this tutorial). Inside this directory, make sure that your movies are called with the same rootname as the averaged micrograph from which you picked your particles before; MINUS its `.mrc` extension; PLUS an underscore; PLUS a common `movie rootname` ("movie" in this case) PLUS a `.mrcs` extension. For example, the movie for `mic001.mrc` should be called `mic001_movie.mrcs`. Import these movies using the Import job-type and select the `2D micrograph movies` option.

Also, it is assumed later on in the movie-particle normalisation that each of the original micrographs was the average of its corresponding movie. If this is somehow not the case for your own data, then you could have used the program `relion_image_handler` to calculate average micrographs from any given number of movie frames, before starting the entire image processing procedure.

## 10.2 Movie refinement

The Movie refinement job-type combines two processes in a single job: extraction of movie-frame particles and refinement of those against a 3D model. On the I/O tab, set:

- `Input movies STAR file:`
  `MotionCorr/job002/corrected_micrograph_movies.star`
- `Rootname of movies files:` `movie`

  (Because these movies filenames are ending with `_movie.mrcs`.)
- `Continue 3D auto-refine from:`
  `Refine3D/after_first_class3d/run_it020_optimiser.star`

  (Only the particles that were included in this 3D auto-refine run will be extracted from the movies.)
- `Process micrographs in batches of:` `-1`

  (This option allows to process micrograph movies in smaller batches, as we've seen computational instabilities when processing large movie data sets. As this test data set only contains 15 movies, processing in batches is not necessary, and hence a value of -1 is used. For large data sets, we often process micrographs in batches of 50-100.)

On the extract tab, set:

- `First movie frame to extract:` `1`
- `Last movie frame to extract:` `16`
- `Average every so many frames:` `1`

  (Adjust this value to have a dose of at least approximately 1 electron/Å$^2$ in the averaged frames, so use a value higher than 1 if you have a dose of less than 0.5-1 electrons/Å$^2$ in each individual movie frame. )

- `Maximum number of MPI nodes:` `8`

  (The first step of this job, the extraction of movie particles, is very intensive in disk I/O. Therefore, it may not be efficient to use a large number of MPI processes for this part of the job. Regardless of the number of MPI processes provided on the `Running` tab, the program will not use more processes than the number provided here. On our cluster, 8 seems a good value.)

- `Particle box suze (pix):` `100`

  (Make sure this is the same size as you gave in the Particle extraction job-type.)

- `Invert contrast?` `Yes`

  (Use the same as before in the Particle extraction job-type.)

- `Rescale particles?` `No`

  (Use the same as before in the Particle extraction job-type.)

On the normalise tab, set:

- `Normalize movie-particles?` `Yes`
- `Diameter background circle (pix):` `60`

  (Use the same as before in the Particle extraction job-type.)

- `Stddev for white dust removal:` `-1`
- `Stddev for black dust removal:` `-1`

On the refine tab, set:

- `Running average window:` `7`

  (The optimal value depends on the size of the particle, the dose rate, etc. Basically, the main question is how accurately can you still align this particle with the accumulated dose in the running average. For large particles like ribosomes, a dose of 3-5 electrons/Å$^2$ in the running average will be enough. For smaller particles, e.g. for beta-galactosidase, you may want to increase this to 7-9 electrons/Å$^2$ in the running average.)

- `Stddev on the translations (pix):` `1`

  (This will restrain how far a movie-frame can move away from the average position.)

- `Also include rotational searches?` `No`

  (Including the rotations becomes computationally much more expensive. And the particle polishing procedure below will disregard any beam-induced rotations anyway. Only for very large objects may including the beam-induced rotations (and skipping the polishing step) be useful. Still, even for ribosomes we now prefer the polishing procedure outlined below.)

As mentioned above: this job actually performs 2 functions: it will first extract the movie particles, and will then align them with respect to the 3D reference. The first step is done with a restricted number of MPI processes, but you may need to run the second part with many more, and this is what you should select on the Running tab. The amount of memory required is approximately the same as for the 3D auto-refinement, so you need to adjust the number of threads accordingly. Perhaps you could use an alias like `first3dref` again?

The only useful output file from this job will be a new movie-frame particle STAR file with the refined offsets for all movie-particles. It will be called called `MovieRefine/first3dref/run_avg7_off1_data.star`. The reason why the value of the running average width and the stddev of the translations is reflected in the output name is that one often tries different values of these parameters, and one would not want to re-extract all movie-particles for each setting. In order not to repeat the movie-extraction step every time, one can thereby use the Continue now button so that one will just get another output file with the corresponding filename in the same output directory.

## 10.3 Particle polishing

The particle polishing algorithm takes the (possibly very noisy) movement tracks from the `MovieRefine/first3dref/run_avg7_off1_data.star` and fits straight lines through it. In order to further reduce the noise in these tracks, it also exploits the observation that neighbouring particles on a micrograph often move in similar directions. It does so by evaluating the tracks of multiple neighbouring particle together, where the contribution of each particle to its neighbours tracks is determined by a Gaussian function of the inter-particle distance. The standard-deviation of this Gaussian (in pixels) is determined by the user. In a second step, the polishing algorithm estimates a B-factor and a linear intensity factor to describe the resolution-dependent power of the signal in each individual movie frame. This way, resolution-dependent radiation damage can be modelled by performing a weighted average of all aligned movie frames for each particle. The resulting particles are called “polished” or “shiny”, as they have increased signal-to-noise ratios.

On the Particle polishing job-type, on the I/O tab set:

- `Input STAR file with aligned movies:` `MovieRefine/first3dref/run_avg7_off1_data.star`
- `Mask for the reconstruction:` `MaskCreate/first3dref_th002_ext2_edg3/mask.mrc`

  (Use the same one that gave good results in the post-processing.)

On the Movement tab set:

- `Linear fit particle movements?` `Yes`
- `Stddev on particle distance (pix):` `300`

  (The larger the particle, the less noisy the movement tracks from the movie-refinement become and therefore the smaller value can be used here. For ribosomes we often use values like 100 pixels. Smaller values allow to describe ore complicated movement patterns.

On the Damage tab set:

- `Perform B-factor weighting?` `Yes`
- `Highres-limit per-frame maps (A):` `6`

  (This option is merely to save computational resources. As individual-frame reconstructions tyopically do not extend out to Nyquist, you can give this value here to limit the frequencies taken into account when performing the single-frame reconstructions.)

- `Lowres-limit B-factor estimation (A):` `20`

  (The algorithm will estimate relative Guinier-plots between individual-frame reconstructions and a reconstruction from all movie-frames averaged together. These plots may be used to fit a straight line, and we have seen that often lower frequencies may be included in these plots than in the original Guinier-plots as proposed by [10].)

- `Average frames B-factor estimation:` `1`

  (One should always try to use a value of 1 first. If somehow this gives reconstructions that do not get high-enough resolutions to estimate reliable B-factors from, then one could average together the movie frames from 3 or seven 5 movie frames using this option. Only odd values are allowed.)

- `Symmetry:` `D2`

On the normalise tab, set:

- `Diameter background circle (pix):` `60`

  (Use the same as before in the Particle extraction job-type.)

- `Stddev for white dust removal:` `-1`
- `Stddev for black dust removal:` `-1`

Ignore the Helix tab, use an alias like `first3dref`, and run the job. Note that particle polishing has NOT been GPU-accelerated, so you'll probably want to run this on a cluster. The maximum useful number of MPI processors for this job is the number of movie frames in the data times two. If your particle boxes are relatively large, then the memory requirements may be significant, although typically not larger than during the last iteration of the 3D auto-refine runs.

The program will perform the linear fitting of the motion tracks, the single-frame reconstructions to estimate B-factors and scale-factors for the radiation-damage weighting, and it will create the new shiny particles. The estimated B-factors and scale-factors, together with the Guinier plots (and their fits with straight lines) that were used to calculate these factors, and the movement tracks and their linear fits are all in a large PDF file called `Polish/first3dref/logfile.pdf`, which you can open from the Display: button.

We often observe relatively big negative B-factors for the first 1-2 frames where the sample moves a lot, and then also in the later frames where radiation damage sets in. The movies from this tutorial were recorded with 1 electron/Å$^2$ per movie-frame, so the total dose of 16 electrons/Å$^2$ isn't very high, and radiation damage only plays a minor role. Often the intercept of the fitted line with the Y-axis of the Guinier-plot (i.e. the linear intensity scale factor) decreases with dose, although also here the first few frames may be relatively bad.

If somehow the estimated B-factors or scale factors look weird, then one could examine the linear fits through the Guinier plots. Sometimes, the Guinier plots cannot be fitted well with a straight line over the entire frequency range that was input on the Damage tab. It may then be worth re-running the job with different parameters. Also, if single-frame reconstructions do not extend beyond the 20 Å frequency as defined above, then no line can be fitted and the program will not work (it will give NaN for the estimated B-factors). In such cases, one may choose to skip the B-factor weighting, or to average multiple frames using the `Average frames B-factor estimation: 1` on the same tab. It is probably also not a bad idea to visualise the STAR file with the output polished particle (`shiny.star`) from the Display: button to make sure everything has gone all right.

If you're somehow not yet happy with the polishing results, you can use the Continue run button to repeat part of the calculations. When you click it, the program will look for the following files, and if they are there it will skip that step of the calculation and move to the next step:

- `frame0??_half?_class001_unfil.mrc` (If any of these maps is missing, it will be re-calculated. If they are already there, then their reconstruction calculation will be skipped. If you've changed the option `Average frames B-factor estimation`, then you should re-calculate all reconstructions. In that case, it's better to start a new job instead of continuing an old one, or you'd need to remove all existing frame reconstructions first.)

- `bfactors.star` (If this file is missing, the program will read in all existing frame reconstructions and determine the B-factors and scale-factors from linear fits through their Guinier plots. Therefore, when you've for example changed the resolution range of the fits on the Damage tab, but not the `Average frames B-factor estimation`, it's very useful to remove the existing `bfactors.star` file, so the new B-factors and scale factors will be calculated, but you don't need to reconstruct all the maps anymore. This will go much faster.

- `shiny.star` (If this file is missing, the existing `bfactors.star` file will be read and the shiny particles will be re-weighted according to these parameters. This can be very useful, for example if you decided to fit smooth curves through the previously determined B-factor and scale-factor plots, and you had replaced the existing `bfactors.star` file with the newly fitted one. In this case, the program will only re-make the shiny particles.)

## 10.4 Re-refine and classify the polished particles

The polished particles in `shiny.star` have increased signal-to-noise ratios compared to the original ones. This may be useful in further classifications. Please see our latest chapter in Methods in Enzymology [11] for an in-depth description of different classification strategies at this point. As this chapter is behind a paywall, you can also download an early draft of it by using:

```
wget ftp://ftp.mrc-lmb.cam.ac.uk/pub/scheres/relionreview2016.pdf
```

At the very least we ALWAYS use the polished particles to perform a final 3D auto-refine run, as the reconstruction from the polishing itself only positions each particle in its most-likely orientation instead of performing the probability-weighted angular assignment that yields higher resolution maps. Therefore, repeat the 3D auto-refine run with the `shiny.star` as input. After this refinement, don't forget to re-do the post-processing and the local-resolution estimation.

In this test case, the reconstruction already went to the Nyquist frequency of the (down-sampled) data before movie-refinement and particle polishing. Therefore, the latter had a negligible impact on the map. We hope this demonstration was still useful in teaching you how to do the movie-processing.

## 11 Wrapping up

Do you wonder how you got to your final reconstruction? Select the last job you performed from the Finished jobs list and try the `Make flowchart` option from the Job actions button. You'll need LaTeX and the `TikZ` package on your system in order for this to work. The first flowchart shows you the path how you got to this end. Note that flowcharts longer than 10 steps will be cut into pieces. There may be branches in your work flow. Therefore, following the flowchart of your last job, there will also be flowcharts for each branch. You can click on the blue boxes to get to the corresponding position in the PDB file. On the last page will be a flowchart without the exact job names, which may be useful for publication purposes.

That's it! Hopefully you enjoyed this tutorial and found it useful. If you have any questions about RELION, please first check the FAQ on the RELION Wiki and the CCPEM mailing list. If that doesn't help, use the CCPEM list for asking your question. *Please, please, please, do not send a direct email to Sjors, as he can no longer respond to all of those.*

If RELION turns out to be useful in your research, please do cite our papers and tell your colleagues about it.

## References

[1] Tanmay A. M. Bharat, Christopher J. Russo, Jan Lwe, Lori A. Passmore, and Sjors H. W. Scheres. Advances in Single-Particle Electron Cryomicroscopy Structure Determination applied to Sub-tomogram Averaging. *Structure (London, England: 1993)*, 23(9):1743–1753, September 2015.

[2] Shaoxia Chen, Greg McMullan, Abdul R. Faruqi, Garib N. Murshudov, Judith M. Short, Sjors H. W. Scheres, and Richard Henderson. High-resolution noise substitution to measure overfitting and validate resolution in 3d structure determination by single particle electron cryomicroscopy. *Ultramicroscopy*, 135:24–35, December 2013.

[3] Rafael Fernandez-Leiro and Sjors H. W. Scheres. A pipeline approach to single-particle processing in RELION. *Acta Crystallographica. Section D, Structural Biology*, 73(Pt 6):496–502, June 2017.

[4] Timothy Grant and Nikolaus Grigorieff. Measuring the optimal exposure for single particle cryo-EM using a 2.6  reconstruction of rotavirus VP6. *eLife*, 4:e06980, 2015.

[5] Shaoda He and Sjors H. W. Scheres. Helical reconstruction in RELION. *Journal of Structural Biology*, in press, 2017.

[6] Dari Kimanius, Bjrn O Forsberg, Sjors HW Scheres, and Erik Lindahl. Accelerated cryo-EM structure determination with parallelisation using GPUs in RELION-2. *eLife*, 5, November 2016.

[7] Alp Kucukelbir, Fred J Sigworth, and Hemant D Tagare. Quantifying the local resolution of cryo-EM density maps. *Nature methods*, 11(1):63–65, January 2014.

[8] Ali Punjani, John L. Rubinstein, David J. Fleet, and Marcus A. Brubaker. cryoSPARC: algorithms for rapid unsupervised cryo-EM structure determination. *Nature Methods*, 14(3):290–296, March 2017.

[9] Alexis Rohou and Nikolaus Grigorieff. CTFFIND4: Fast and accurate defocus estimation from electron micrographs. *Journal of Structural Biology*, 192(2):216–221, November 2015.

[10] Peter B Rosenthal and Richard Henderson. Optimal determination of particle orientation, absolute hand, and contrast loss in single-particle electron cryomicroscopy. *Journal of Molecular Biology*, 333(4):721–745, October 2003.

[11] S. H. W. Scheres. Processing of Structurally Heterogeneous Cryo-EM Data in RELION. *Methods in Enzymology*, 579:125–157, 2016.

[12] Sjors H W Scheres. Classification of Structural Heterogeneity by Maximum-Likelihood Methods. In *Cryo-EM, Part B: 3-D Reconstruction*, volume 482 of *Methods in Enzymology*, pages 295–320. Academic Press, 2010.

[13] Sjors H W Scheres. A Bayesian view on cryo-EM structure determination. *Journal of Molecular Biology*, 415(2):406–418, January 2012.

[14] Sjors H W Scheres. RELION: Implementation of a Bayesian approach to cryo-EM structure determination. *Journal of Structural Biology*, 180(3):519–530, December 2012.

[15] Sjors H W Scheres and Shaoxia Chen. Prevention of overfitting in cryo-EM structure determination. *Nature methods*, 9(9):853–854, September 2012.

[16] Sjors H W Scheres, R. Nunez-Ramirez, C. O. S Sorzano, J. M Carazo, and R. Marabini. Image processing for electron microscopy single-particle analysis using XMIPP. *Nature Protocols*, 3(6):977–90, 2008.

[17] J M Smith. Ximdisp–A visualization tool to aid structure determination from electron microscope images. *Journal of structural biology*, 125(2-3):223–228, May 1999.

[18] Guang Tang, Liwei Peng, Philip R Baldwin, Deepinder S Mann, Wen Jiang, Ian Rees, and Steven J Ludtke. EMAN2: an extensible image processing suite for electron microscopy. *Journal of Structural Biology*, 157(1):38–46, January 2007.

[19] Kutti R Vinothkumar, Greg McMullan, and Richard Henderson. Molecular Mechanism of Antibody-Mediated Activation of -galactosidase. *Structure (London, England: 1993)*, 22(4):621–627, April 2014.

[20] Kai Zhang. Gctf: Real-time CTF determination and correction. *Journal of Structural Biology*, 193(1):1–12, January 2016.

[21] Shawn Q. Zheng, Eugene Palovcak, Jean-Paul Armache, Kliment A. Verba, Yifan Cheng, and David A. Agard. MotionCor2: anisotropic correction of beam-induced motion for improved cryo-electron microscopy. *Nature Methods*, 14(4):331–332, April 2017.