Using Relion within SCIPION for single particle data processing

August 2022

Contents

Introduction

Overview

This tutorial aims to familiarize you with cryo-EM single particle analysis (SPA) using Scipion (de la Rosa-Trevín et al., 2016) framework with particular focus on *Relion* (Scheres, 2012) software. We assume you are familiar with Scipion GUI and have some experience with *Relion*. We have re-used many parts of the existing Relion guide, for more details please refer to it first or click on the help button for any parameter. Here we will closely follow that guide and demonstrate the complete workflow starting from raw data up to the final 3 Å structure, highlighting the features of our software framework. After that, you should be able to run Scipion with your own data.

Software requirements

To follow this tutorial you need to have Scipion version 3.x already installed on your system. Also, make sure you have the following plugins and their corresponding binaries installed and configured:

- scipion-em-chimera / chimerax (optional, for visualization only)
- scipion-em-cistem / ctffind4
- scipion-em-motioncorr / motioncor2
- scipion-em-relion / relion 4.x
- scipion-em-xmipp / xmipp

Test data

We will use the same dataset as in *Relion* tutorial: beta-galactosidase collected on a 200 kV JEOL cryo-ARM microscope at Osaka university. The dataset can be downloaded and extracted with the following commands (the full dataset is also available at EMPIAR-10204):

```
wget ftp://ftp.mrc-lmb.cam.ac.uk/pub/scheres/relion30_tutorial_data.tar
tar -xf relion30_tutorial_data.tar
```

After extracting the archive you should have a folder called *relion30_tutorial*. Throughout the tutorial we will refer to this folder as `$DATA/relion30_tutorial`. You also need to download the MTF file for the K2 camera at 200 kV that we will use for post-processing.

1 Preprocessing

1.1 Creating the project and importing raw data

We will start by launching Scipion GUI:

```
> scipion3
```

In the project window click on Create Project button and enter a project name, keeping the default location. Press Create, so that a new project window appears.

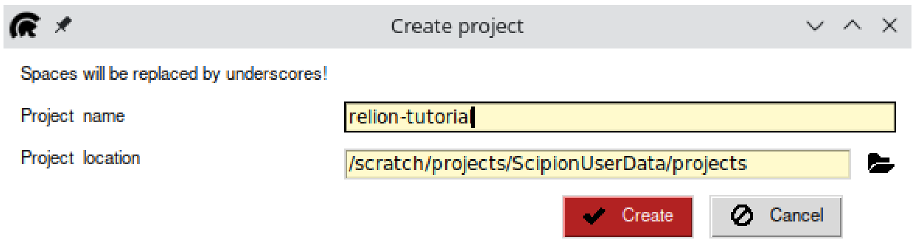

Create project
Spaces will be replaced by underscores!
Project name
relion-tutorial
Project location
/scratch/projects/ScipionUserData/projects
Create
Cancel

Figure 1: Create project dialog.

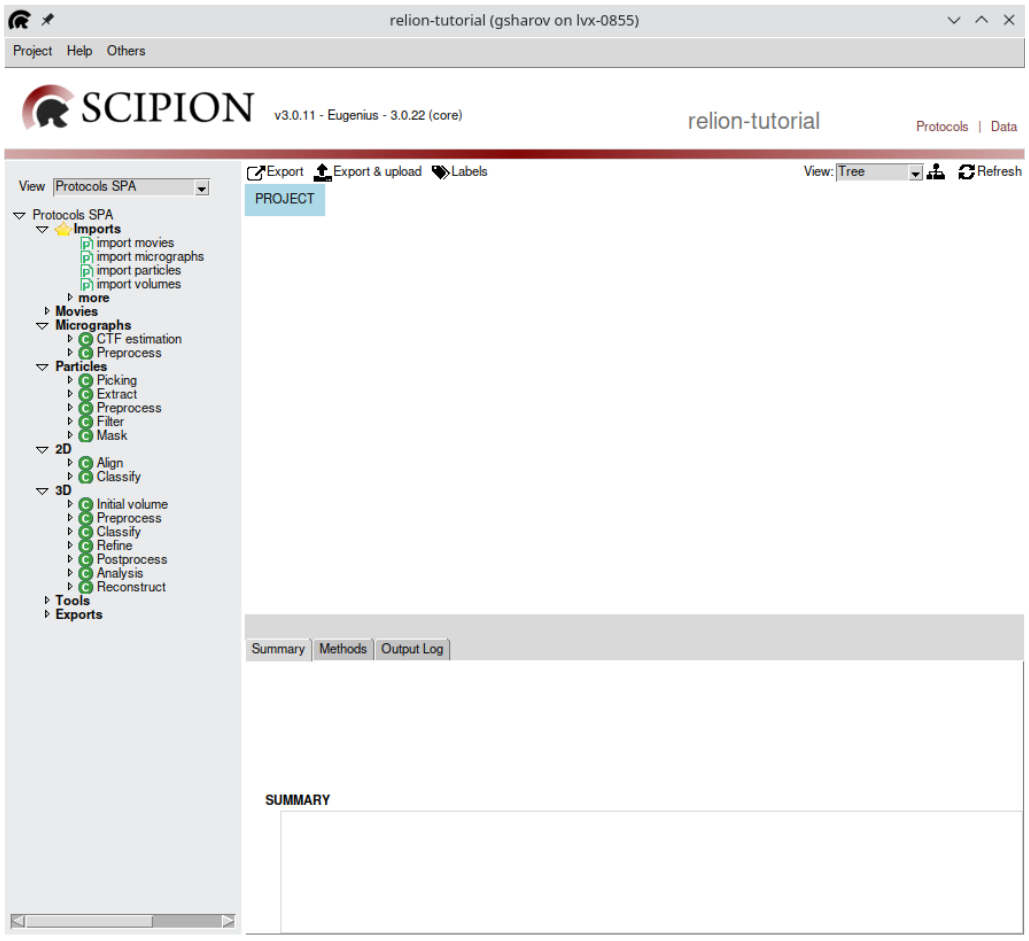


Figure 2: Main project window.

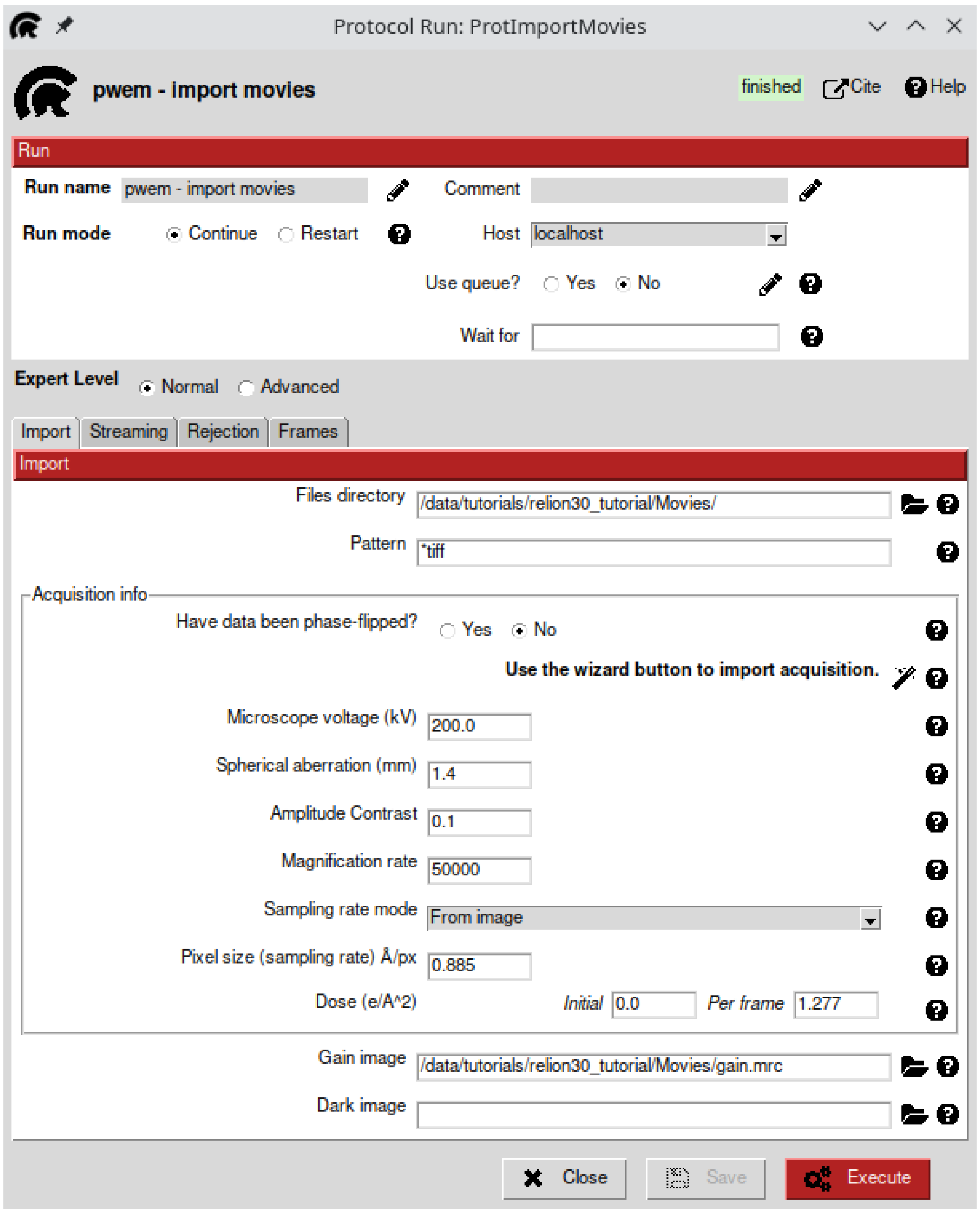
On the left panel select View ⟩ Protocols SPA and double-click on the import movies protocol. On the *Import* tab fill in the parameters as listed below:

- **Files directory**: `$DATA/relion30_tutorial/Movies`
- **Pattern**: *tiff
- **Microscope voltage (kV)**: 200
- **Spherical aberration (mm)**: 1.4
- **Pixel size (Å/px)**: 0.885
- **Dose per frame (e/Å^2)**: 1.277
- **Gain image**: `$DATA/relion30_tutorial/Movies/gain.mrc`

Here we will not import data in streaming (on-the-fly) mode, so just ignore other protocol tabs.

TIP

If you have a defects file from your detector, you will need to input it later, during motion correction. Same applies to the gain image orientation.



Run name: pwem - import movies
Comment:
Run mode: [✓ Continue] Restart
Host: localhost
Use queue? Yes [✓ No]
Wait for:
Expert Level: [✓ Normal] Advanced

Files directory: /data/tutorials/relion30_tutorial/Movies/
Pattern: *tiff

Acquisition info
Have data been phase-flipped? Yes [✓ No]
Use the wizard button to import acquisition.
Microscope voltage (kV): 200.0
Spherical aberration (mm): 1.4
Amplitude Contrast: 0.1
Magnification rate: 50000
Sampling rate mode: From image
Pixel size (sampling rate) Å/px: 0.885
Dose (e/A^2): *Initial* 0.0 *Per frame* 1.277

Gain image: /data/tutorials/relion30_tutorial/Movies/gain.mrc
Dark image:

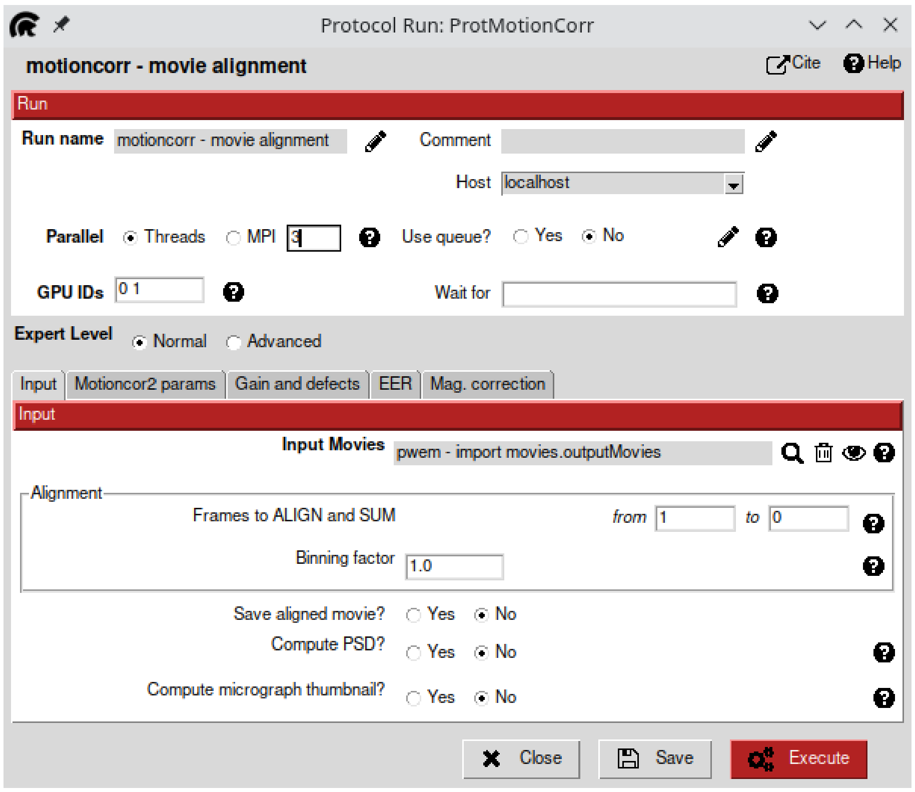
Figure 3: Import movies protocol.

After checking that all input parameters are correct, press Execute button. The protocol should complete almost immediately as it only creates symbolic links to your raw data and registers metadata in the database. The **Summary** tab should now show that 24 movies have been imported, 24 frames each, with the pixel size of 0.89 Å/px. All movies by default have been assigned a single optics group. Refer to a separate chapter for more information about optics groups.

1.2 Beam-induced motion correction

Aligning individual movie frames is necessary to correct for beam-induced specimen motion and restore high resolution information. In this practical we will use *motioncor2* (Zheng et al., 2017) for movie alignment. Compared to *Relion*'s alternative protocol, *motioncor2* supports execution on GPU and usually runs faster, especially on large movies. There are, however, some downsides: local motion tracks and pixel defects from *motioncor2* cannot be recognized by *Relion*'s bayesian polishing algorithm later in the processing pipeline. Another difference is that *Relion*'s algorithm can save aligned sums of power spectra which can speed up the next processing step - CTF estimation with *CTFFind4*. Additionally, *Relion* can write images in *float16* mode to save disk space, however currently *Xmipp* cannot read this format. A downside of using *Relion* is that the motion correction algorithm can only run on CPU. Altogether, you should decide which protocol to use.

To find *motioncor2* protocol you can either expand the protocols tree on the left panel of the project (check the **Movies** list) or use `ctrl` + `F` to search for `motioncorr - movie alignment`. Select the movie set that you have just imported as input for this protocol. We will not change other default options in other tabs, but you are free to explore them on your own. If you have two GPUs available on your machine, set **GPU IDs** to **"0 1"** and **number of threads** to **3** - this way the protocol will run on every two movies in parallel, each on a separate GPU. The third thread is required to coordinate the jobs. If your machine does not have GPUs, you will have to use `relion - motion correction` protocol. For more details about parallel jobs execution in SCIPION see the chapter at the end of this guide.


Protocol Run: ProtMotionCorr

motioncorr - movie alignment

Run

Run name: motioncorr - movie alignment
Comment:
Host: localhost
Parallel: [✓ Threads] MPI 3
Use queue? Yes [✓ No]
GPU IDs: 0 1
Wait for:

Expert Level: [✓ Normal] Advanced

Input | Motioncorr2 params | Gain and defects | EER | Mag. correction

Input

Input Movies: pwem - import movies.outputMovies

Alignment
Frames to ALIGN and SUM: from 1 to 0
Binning factor: 1.0

Save aligned movie? Yes [✓ No]
Compute PSD? Yes [✓ No]
Compute micrograph thumbnail? Yes [✓ No]

Figure 4: Movie alignment protocol for motioncor2.

After launching the protocol, we can go further and start the CTF estimation. In this case, thanks to the stream processing capability of Scipion , we don't need to wait until the previous protocol finishes to start the next one.

ALTERNATIVES

- relion - motion correction
- xmipp3 - optical alignment
- xmipp3 - flex alignment
- cistem - unblur

If you would like first to check the results of the movie alignment, select the running *motioncor2* protocol and click on Analyze Results to open the viewer. Here you can display the output micrographs set that will show you the global motion plots for each movie or you may choose to plot motion per frame as shown below.

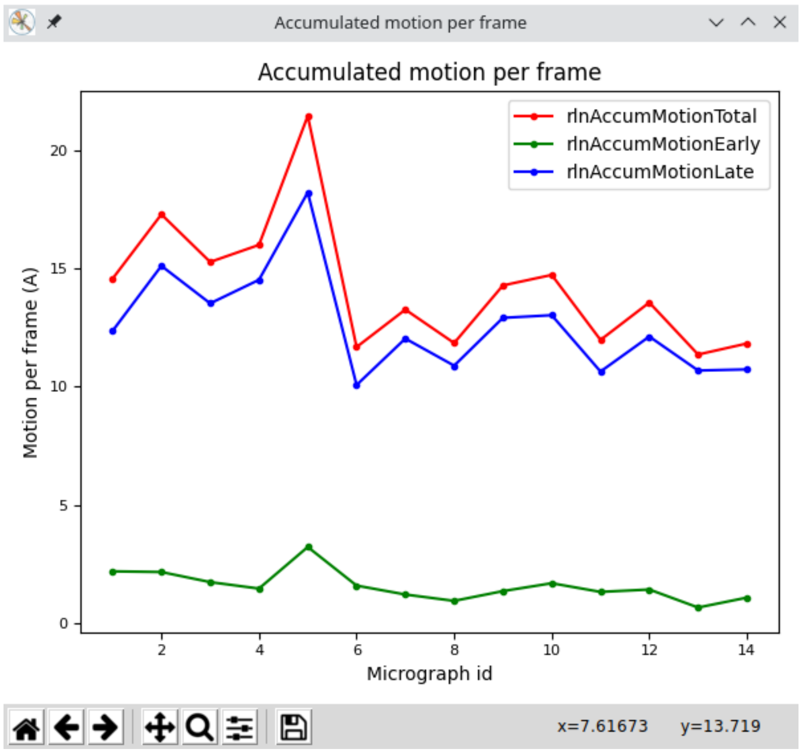
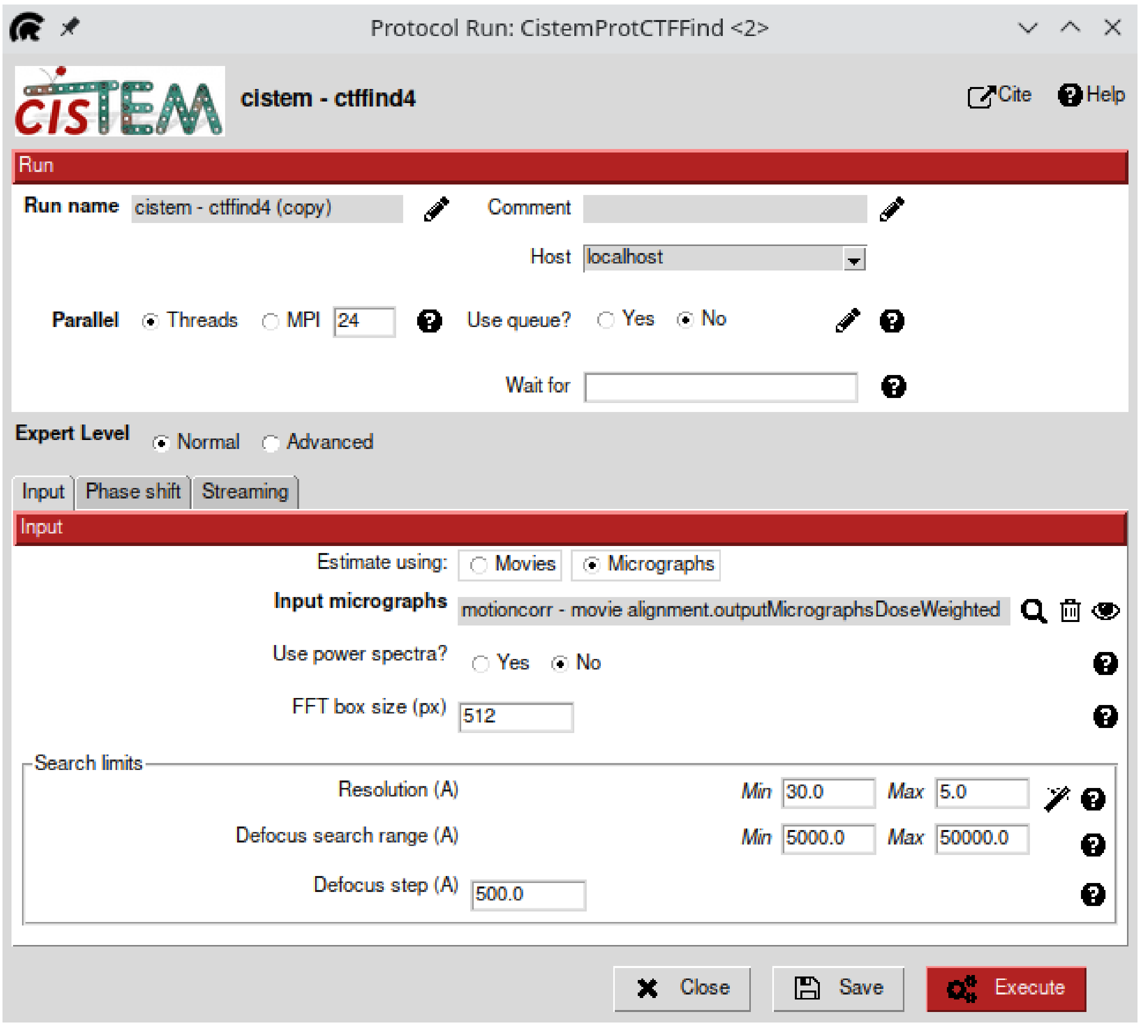


Figure 5: Plot motion per frame for each micrograph.

1.3 CTF estimation

We will now estimate CTF parameters of the aligned micrographs with *CTFFind4* (Rohou and Grigorieff, 2015). By default, the previous motion correction protocol (*Relion* or *motioncor2*) has produced a dose-weighted micrographs set which we will use for CTF determination. Some users prefer to use non-dose-weighted micrographs instead - it is certainly possible, but you would have to run the movie alignment with a specific option that saves such micrographs.

Locate `cistem - ctffind4` protocol and select dose-weighted micrographs from the previous step as the input. If you have used *Relion*'s motion correction, set **Use power spectra? Yes**, otherwise choose **No**. Set number of threads equal to the number of available CPU cores. Other default values are okay for this dataset, but if you want to e.g. adjust the resolution search range for your own data, you can use the wizard shown below (Figure 7). Make sure that your frequency range includes all CTF zeros for the most of micrographs.



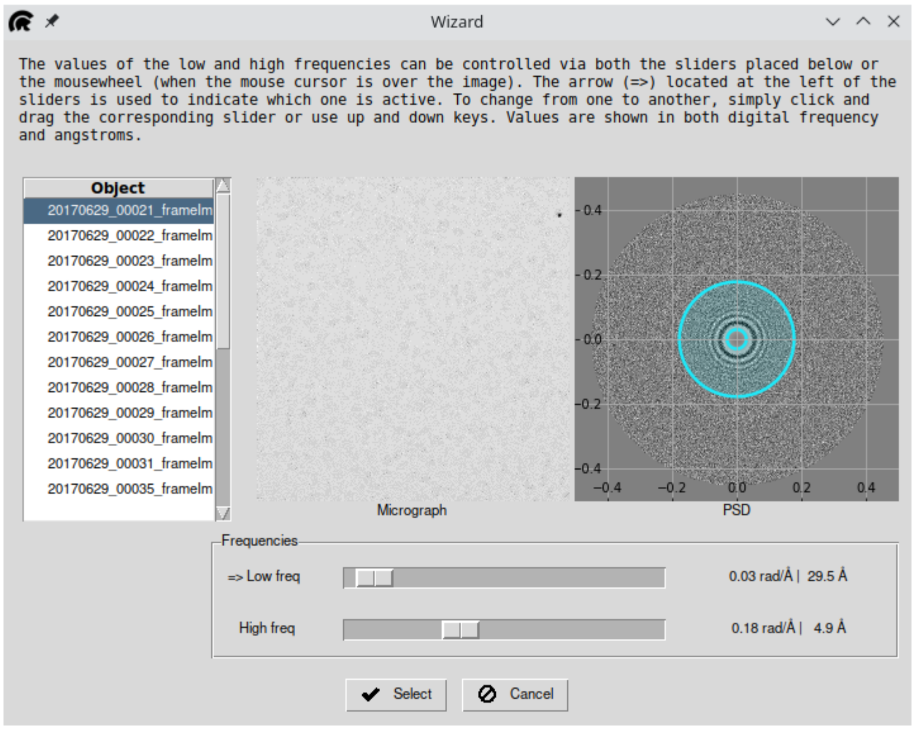
Run name: cistem - ctffind4 (copy)
Comment:
Host: localhost
Parallel: [✓ Threads] MPI 24
Use queue? Yes [✓ No]
Wait for:
Expert Level: [✓ Normal] Advanced

Input

Estimate using: Movies [✓ Micrographs]
Input micrographs: motioncorr - movie alignment.outputMicrographsDoseWeighted
Use power spectra? Yes [✓ No]
FFT box size (px): 512

Search limits
Resolution (A): Min 30.0 Max 5.0
Defocus search range (A): Min 5000.0 Max 50000.0
Defocus step (A): 500.0

Figure 6: CTFFind4 protocol.



```
The values of the low and high frequencies can be controlled via both the sliders placed below or
the mousewheel (when the mouse cursor is over the image). The arrow (=>) located at the left of the
sliders is used to indicate which one is active. To change from one to another, simply click and
drag the corresponding slider or use up and down keys. Values are shown in both digital frequency
and angstroms.
```

Object
20170629_00021_frameIm
20170629_00022_frameIm
20170629_00023_frameIm
20170629_00024_frameIm
20170629_00025_frameIm
20170629_00026_frameIm
20170629_00027_frameIm
20170629_00028_frameIm
20170629_00029_frameIm
20170629_00030_frameIm
20170629_00031_frameIm
20170629_00035_frameIm

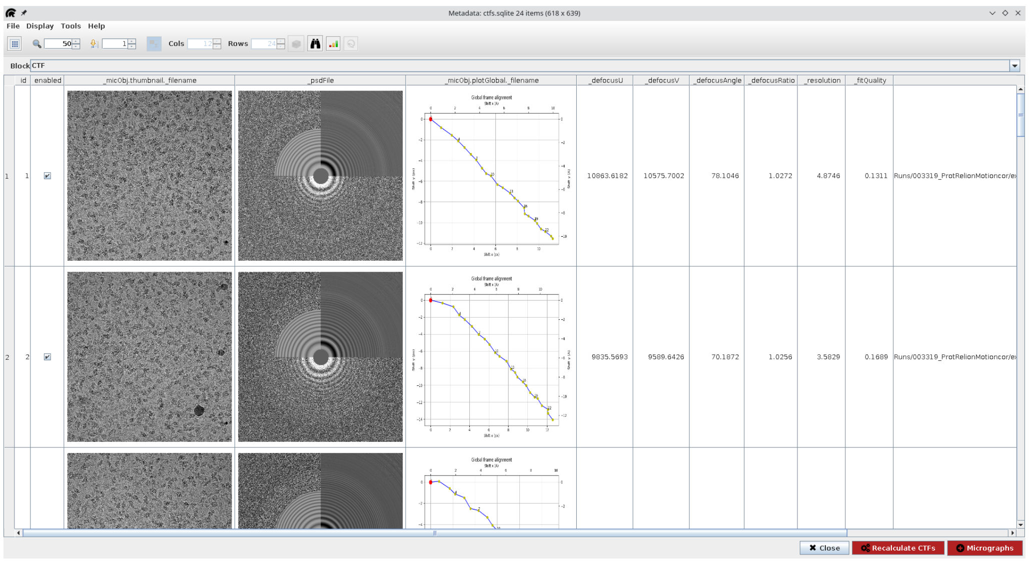
Frequencies
=> Low freq 0.03 rad/Å | 29.5 Å
High freq 0.18 rad/Å | 4.9 Å

Figure 7: A wizard to help to select the frequency range for CTF estimation.

ALTERNATIVES

- gctf - ctf estimation
- xmipp3 - ctf estimation

The output from *CTFFind4* (or any other CTF estimation protocol) can be displayed upon clicking on the Analyze Results button (Figure 8). You can sort the resulting CTFs by defocus, resolution, fit quality etc. If you want to discard some micrographs with bad CTF you may click the right mouse button and select **Disable** for the item that you don't like. You can select multiple items at once using Shift and/or Ctrl. Once you disable all bad images, press on the red Micrographs button to create a new subset of micrographs (along with CTFs) with only enabled items. Luckily, in our tutorial dataset we only have good micrographs, so we don't need to exclude any.



	id	enabled	_micObj.thumbnail._filename	_psdFile	_micObj.plotGlobal._filename	_defocusU	_defocusV	_defocusAngle	_defocusRatio	_resolution	_fitQuality	
1	1	✓				10863.6182	10575.7002	78.1046	1.0272	4.8746	0.1311	Runs/003319_ProtRelionMotioncor/e
2	2	✓				9835.5693	9589.6426	70.1872	1.0256	3.5829	0.1689	Runs/003319_ProtRelionMotioncor/e

Figure 8: Visualization of CTF estimation results.

For a more detailed inspection of the CTF estimation you can right click on any item row and select **CTFFind plot results** (Figure 9). You can also create other kind of plots by yourself, e.g. a histogram to estimate the defocus range of your dataset.

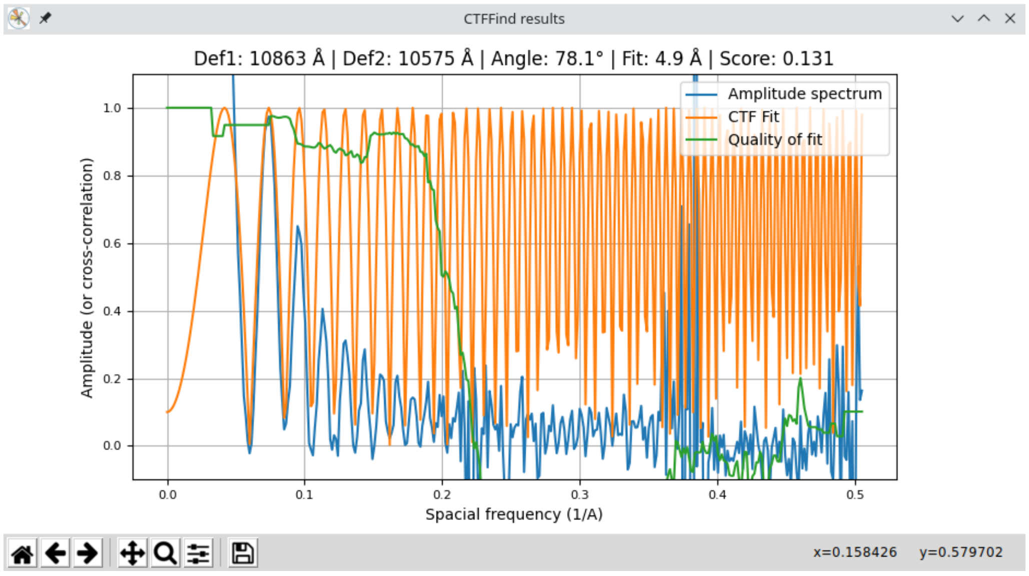


Figure 9: Individual CTF fit plot from CTFFind4.

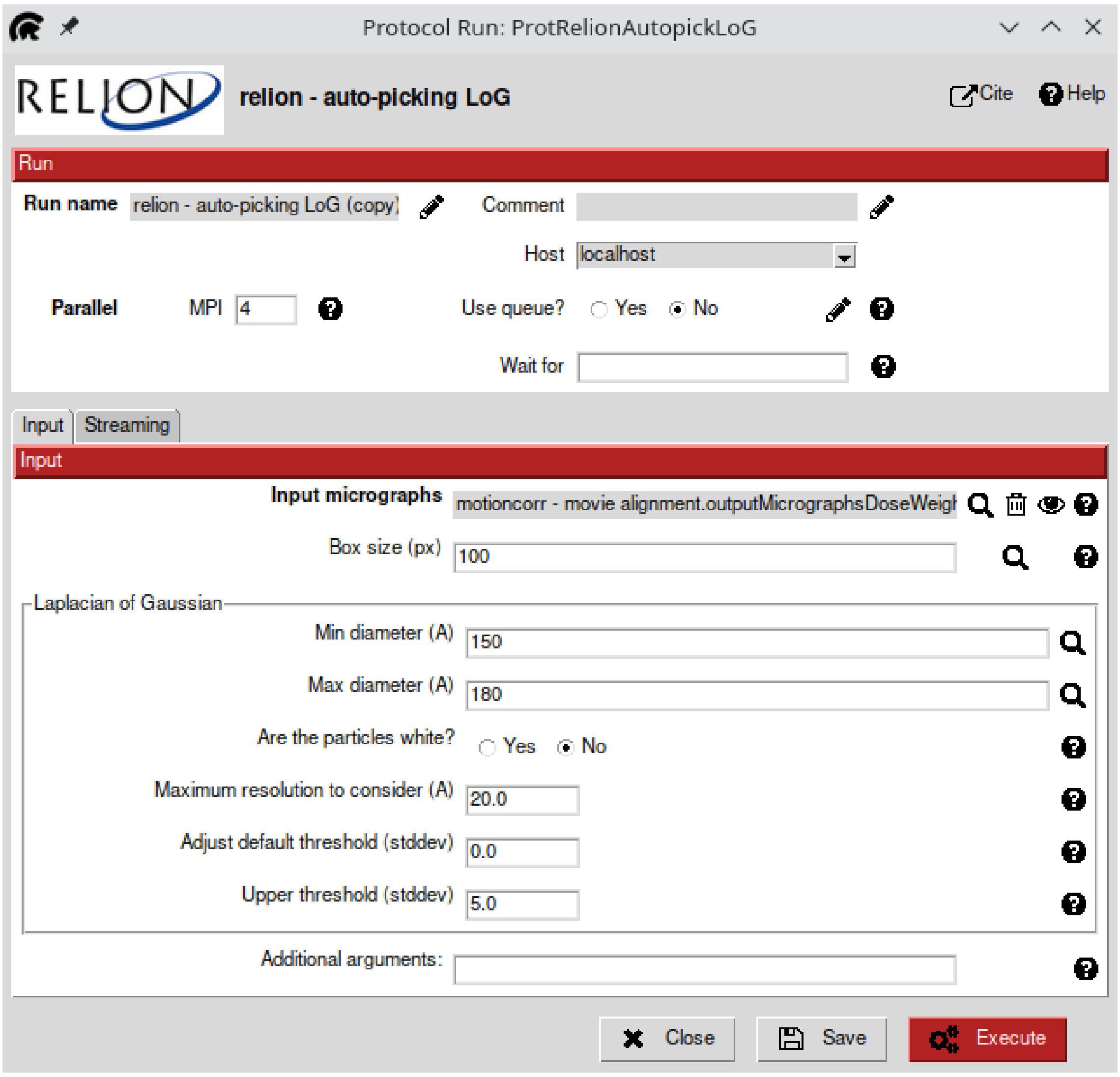
1.4 Particle picking

Manual particle picking can be very tedious and multiple picking tools have been developed that can be more or less convenient depending on the sample and personal preferences. Scipion integrates many different pickers, allowing users to select the one which better fits their needs. It this tutorial we are going to first use *Relion*'s refence-free autopicking method based on the Laplacian-Of-Gaussian filter (relion - auto-picking LoG protocol). This picker is quite fast and easy to use for generating an initial set of particles. Additionally, it does not require CTF information and can be used right after the movie alignment step.

Usually, we advise users to create a small subset of 10-20 micrographs with different defocus values, so that the picking parameters can be quickly optimized for a variety of defocuses. To do this, open the CTF results, sort them by defocus, then randomly select a few micrographs and create a new subset.

Then open the relion - auto-picking LoG protocol and fill in the values on the *Input* tab as described below and shown in Fig. 10. On the *Streaming* tab set **Batch size** e.g. to **8** - this will force *Relion* to pick every 8 micrographs at once. This might be useful when running picking in streaming mode. If you want to pick all items at once, set the value to **0**.

- **Input micrographs**: a small subset of dose-weighted micrographs with different defocuses.
- **Box size (px)**: 100
  This parameter does not affect the protocol run and is only used for visualization of particle picks in the coordinates viewer.
- **Min diameter (Å)**: 150
- **Max diameter (Å)**: 180
- **Upper threshold (stddev)**: 5.0


Protocol Run: ProtRelionAutopickLoG

relion - auto-picking LoG

Run

Run name: relion - auto-picking LoG (copy)
Comment:
Host: localhost
Parallel MPI: 4
Use queue? Yes [✓ No]
Wait for:

Input

Input micrographs: motioncorr - movie alignment.outputMicrographsDoseWeigl
Box size (px): 100

Laplacian of Gaussian
Min diameter (A): 150
Max diameter (A): 180
Are the particles white? Yes [✓ No]
Maximum resolution to consider (A): 20.0
Adjust default threshold (stddev): 0.0
Upper threshold (stddev): 5.0

Additional arguments:

Figure 10: Relion auto-picking LoG protocol.

Once the protocol has finished, you can display output coordinates with Analyze Results, which will open a manual picker interface where you could add or delete particle boxes if you like. You might want to adjust the box size for visualization. To optimize the picking algorithm, it is easier to re-run the protocol with adjusted parameters (e.g. threshold) and display the results again. It should not take very long with only a few input micrographs. Using the parameters suggested above you should get about 5700 picks. Don't worry if some particles have been missed by the algorithm or some contamination was picked, we will optimize the picking later with the reference-based method.

ALTERNATIVES

- relion - auto-picking (reference-based)
- sphire - cryolo picking
- topaz - topaz picking
- xmipp3 - manual and automatic
- eman2 - boxer (manual and automatic)
- gautomatch - auto-picking
- ...many others

1.5 Extracting particles

Once we have a set of coordinates, we can proceed to `relion - particle extraction` protocol. This protocol will crop, normalize and downscale the picked particles to speed up initial processing.

On the *Input* tab, specify:

- **Input coordinates**: set of coordinates from the previous picking protocol.
- **CTF estimation**: set of CTFs from CTFFind4 protocol or a subset of those.
- **Particle box size (pix)**: 256
- **Rescale particles?**: Yes
- **Re-scaled size (px)**: 64
  This is the final downscaled box size. 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.

On the *Preprocess* tab, select:

- **Invert contrast?**: Yes
- **Normalize particles?**: Yes
- **Diameter background circle before scaling (px)**: 200

On the *Streaming* tab set **Batch size** to **0**. Select a few (4-8) MPI processes and execute the protocol. Once it has finished, display the output particles set and confirm that the box size is not too tight around the protein area, which is important for proper noise estimation.

ALTERNATIVES

- xmipp3 - extract particles

2 2D classification and reference-based picking

2.1 Reference-free 2D class averaging

The reference-free 2D classification in *Relion* is a great tool to throw away bad particles. Although it is recommended to try to only pick good particles in the previous step, it can be very laborious if you have a large dataset. Most of the time there are still particles in the data 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 a good way of cleaning up your data.

In general, you can run `relion - 2D classification` to either generate the 2D template averages for picking (using a smaller set of particles) or to classify the whole dataset and clean it up before going to 3D refinement. The latter case is more computationally intensive and it is recommended to use a cluster or a multi-core computer, or even better a GPU if possible. Let's first run a quick classification to generate templates for subsequent auto-picking. Open the `relion - 2D classification` protocol and fill in the parameters as shown below:

On the *Input* tab:

- **Input particles**: set of downscaled particles extracted in the previous protocol.
  It is always recommended to use particles downsampled to 3-4 Å/px for 2D classification to speed-up the computations. If your data were of a good quality you should be able to see secondary structure elements in your final class averages.

- **Particle mask diameter (Å)**: 200
  This should roughly match the mask diameter used during extraction (which was in pixels).

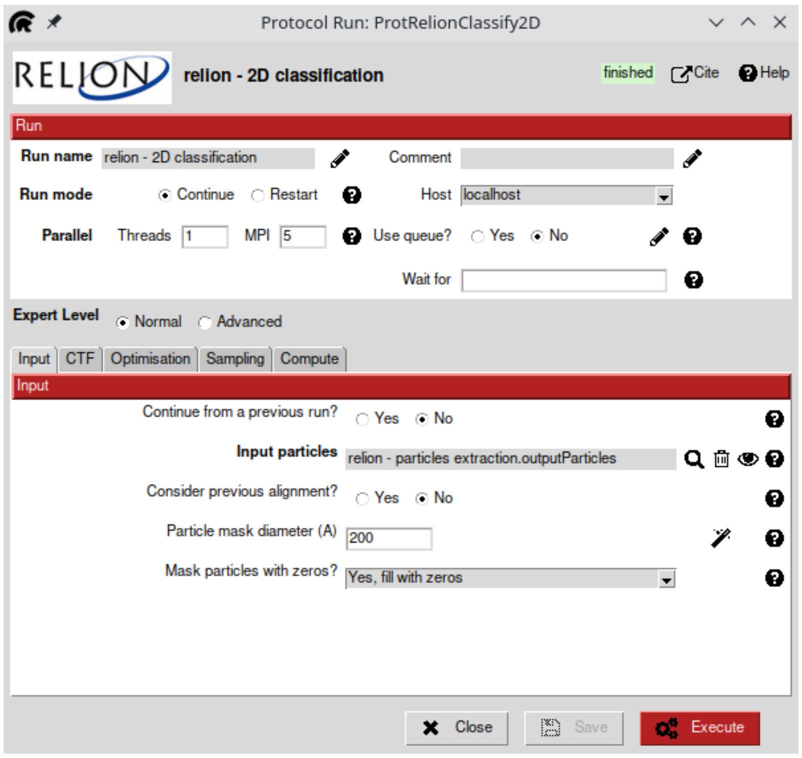
Leave *CTF* tab options by default and on the *Optimisation* tab select:

- **Number of classes**: 20
  That should be enough for roughly 6000 particles that we have. For cryo-EM data *Relion* needs on average at least 100 particles per class. For negative stain one may use fewer, e.g. 20-50 particles per class.

- **Use VDAM algorithm**: No
  There's no real advantage in using it for such a small set.

- **Number of iterations**: 20

If you are using GPUs, on the *Compute* tab make sure to set **Use GPU acceleration? Yes**. On our machine we have 2 GPUs, so we select 5 MPIs and 4 threads, leaving **Which GPUs to use** empty. This way we will have two MPI processes running on each GPU.


Protocol Run: ProtRelionClassify2D

relion - 2D classification — finished

Run
Run name: relion - 2D classification
Comment:
Run mode: [✓ Continue] Restart
Host: localhost
Parallel: Threads 1, MPI 5
Use queue? Yes [✓ No]
Wait for:

Expert Level: [✓ Normal] Advanced

Input
Continue from a previous run? Yes [✓ No]
Input particles: relion - particles extraction.outputParticles
Consider previous alignment? Yes [✓ No]
Particle mask diameter (A): 200
Mask particles with zeros?: Yes, fill with zeros

Figure 11: Relion 2D classification protocol.

2.2 Analyzing 2D results and creating subsets

It is possible to analyze the results while the classification job is still running. The viewer form shown in Figure 12 allows to access different visualization options. By default, the last iteration will be displayed, but you can select any previous one(s).

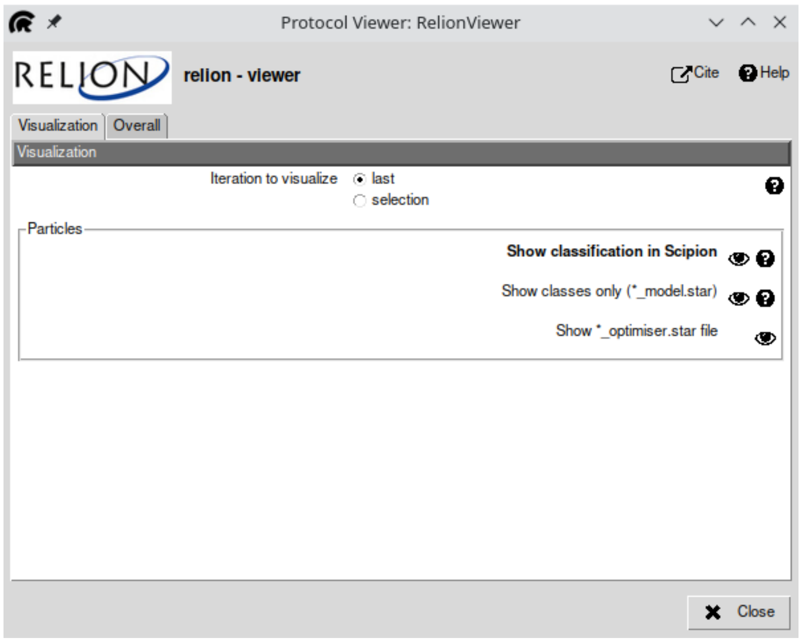

Protocol Viewer: RelionViewer
RELION
relion - viewer
Cite
Help
Visualization
Overall
Visualization
Iteration to visualize
last
selection
Particles
Show classification in Scipion
Show classes only (*_model.star)
Show *_optimiser.star file
Close

Figure 12: Relion viewer form for visualizing classification results.

The most commonly used options are:

- **Show classification in Scipion**: this option will display the 2D classes of the last found iteration. This may take a few minutes depending on the dataset size, the reason being SCIPION generating a set of classes and checking the class assignment for each particle. By default, the class averages are shown in *gallery* mode and sorted by descending number of particles in each class. You can right click on any class and select **Open images** to see the particles assigned to it. If you switch to the *table* mode, you can see the number of particles in each class as well as accuracy of translations and rotations. This viewer allows you to create subsets of particles, classes or averages.

- **Show classes only (*_model.star)**: this option will display the same 2D classes by opening the latest *Relion*'s **_model.star* file directly. This is much quicker and is usually used to watch the classification progress. The downside here is that you cannot create any subsets.

In principle, the viewer is very flexible and allows you to display any star file produced by *Relion*, sort it by any column and create plots you like. Feel free to take your time and play with the viewer interface.

Once the classification has completed, display the results and select a few distinct 2D class averages for template picking. You don't need to select all good-looking 2D classes at this step. Click on Averages to save a new subset.

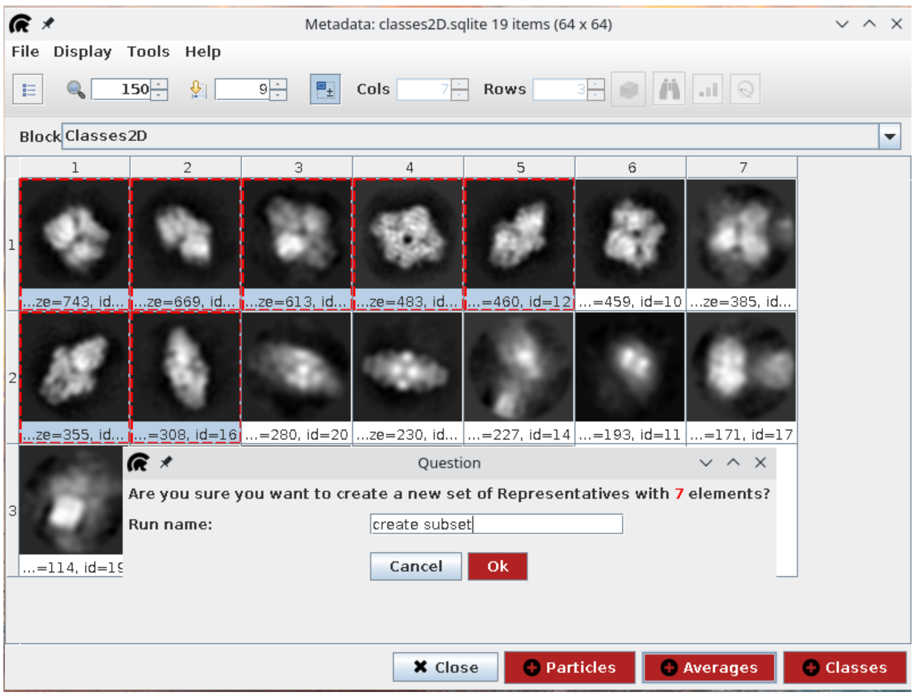

Metadata: classes2D.sqlite 19 items (64 x 64)
File Display Tools Help
Cols Rows
Block Classes2D
Question
Are you sure you want to create a new set of Representatives with 7 elements?
Run name: create subset
Cancel
Ok
Close
Particles
Averages
Classes

Figure 13: Creating a subset of averages. Relion classes are displayed as Scipion classification.

2.3 Reference-based picking and another 2D classification round

Now we will auto-pick particles from our micrographs using the templates generated in 2D classification. Find relion - auto-picking protocol and fill in the options as below:

On the *Input* tab:

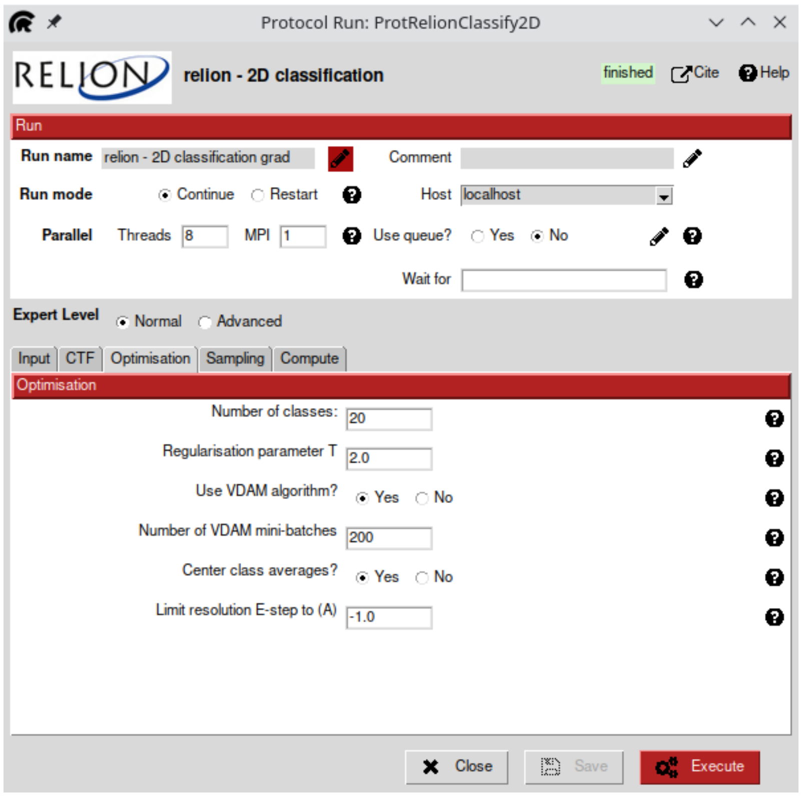
- **Input micrographs**: the full set of dose-weighted micrographs from the motioncorr protocol (or a subset of those with good CTFs).
- **CTF estimation**: set of CTFs from CTFFind4 protocol.

On the *References* tab:

- **References**: 2D
- **Input references**: the subset of averages generated above

On the *Autopicking* tab only change **Picking threshold to 0.05**. On the *Streaming* tab set **Batch size to 0** so that all input micrographs will be picked together in one protocol step. With 1 MPI process and a single GPU the job takes only a few seconds. Now you should have around 8000 particles picked.

Repeat the particle extraction protocol with the new set of coordinates, leaving all parameters the same. Once the extraction is completed, re-run the 2D classification, but now use **VDAM algorithm with 200 mini-batches**. This algorithm does not support MPIs, so select multiple threads for this protocol instead.


Protocol Run: ProtRelionClassify2D

relion - 2D classification — finished — Cite — Help

Run

Run name	relion - 2D classification grad	Comment	
Run mode	[✓ Continue] Restart	Host	localhost
Parallel	Threads 8 MPI 1	Use queue?	Yes [✓ No]
		Wait for	

Expert Level [✓ Normal] Advanced

Input | CTF | Optimisation | Sampling | Compute

Optimisation

Number of classes:	20
Regularisation parameter T	2.0
Use VDAM algorithm?	[✓ Yes] No
Number of VDAM mini-batches	200
Center class averages?	[✓ Yes] No
Limit resolution E-step to (A)	-1.0

Figure 14: Relion 2D classification with VDAM algorithm.

EXERCISE

You may select particles from the best 2D classes manually again or try the new `relion - 2D class ranker` protocol that can automatically detect good classes. You can play with the **auto-selection threshold** parameter to find the optimum value. This protocol does not generate a set of particles but a set of classes, so you will need to generate a particle set from its output yourself.

3 *De novo* 3D model generation

Relion implements a gradient-driven algorithm to generate a 3D initial model *de novo* from the 2D particles. Provided you have a reasonable distribution of viewing directions, and your data were good enough to yield detailed class averages in 2D classification, this algorithm is very likely to produce a suitable low-resolution model that can subsequently be used for 3D classification or 3D auto-refinement.

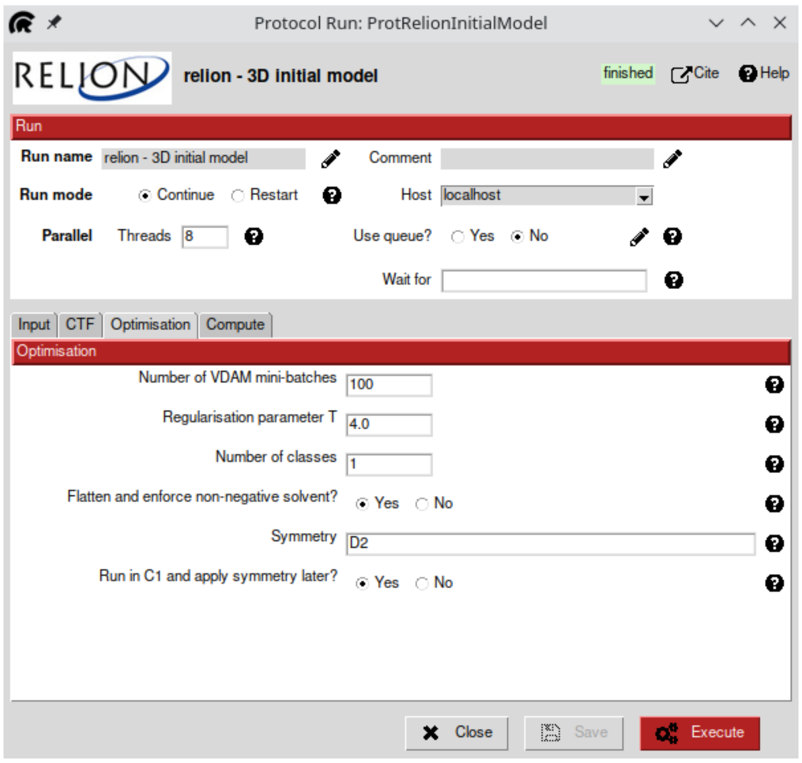
Find `relion - 3D initial model` protocol and fill in its parameters:

On the *Input* tab:

- **Input particles**: select the subset of good particles after 2D classification.
- **Particle mask diameter (Å)**: 200

On the *Optimisation* tab:

- **Number of VDAM mini-batches**: 100
- **Symmetry**: D2.
  The actual refinement will be run in C1 and afterwards the volume will be aligned to the identified symmetry axes and symmetrized.



Run name: relion - 3D initial model
Comment:
Run mode: [✓ Continue] Restart
Host: localhost
Parallel Threads: 8
Use queue? Yes [✓ No]
Wait for:

Optimisation
Number of VDAM mini-batches: 100
Regularisation parameter T: 4.0
Number of classes: 1
Flatten and enforce non-negative solvent? [✓ Yes] No
Symmetry: D2
Run in C1 and apply symmetry later? [✓ Yes] No

Figure 15: Relion 3D initial model protocol.

Once the job is completed, display the resulting symmetrized volume by slices or in *ChimeraX* (if you have the corresponding plugin installed). You should probably confirm that the symmetry point group was correct and that the symmetry axes were identified correctly. As always, the viewer allows you to visualize many other parameters which we leave for you to explore.

4 3D classification and refinement

4.1 Unsupervised 3D classification

Relion provides a powerful 3D multi-reference refinement procedure that is very useful to analyze the heterogeneity in your dataset. The corresponding protocol in

Scipion can be found in: 3D ❯ Classify ❯ relion - 3D classification. We will use mostly its default parameters, except for the following ones:

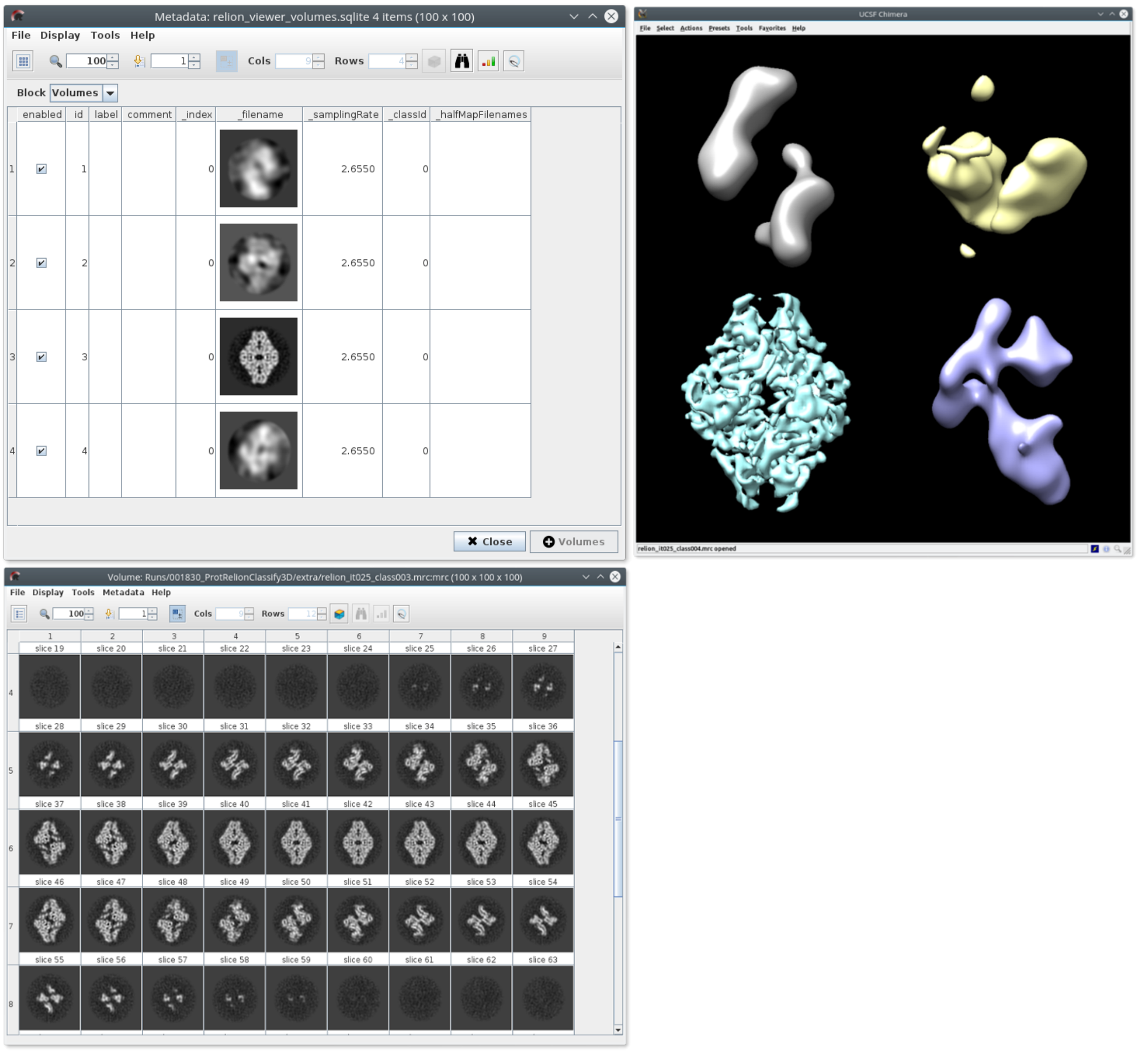
On the *Input tab*:

- **Input particles**: a subset of particles used for 3D initial model protocol. You can also use the output particles set from that protocol, however we will ignore any input alignment information here, so there's not much difference.
- **Particle mask diameter (Å)**: 200
  Same as for 2D classification.

On the *Reference 3D map* tab:

- **Input volume(s)**: The output symmetrized volume from the initial model protocol.
- **Is initial 3D map on absolute greyscale?** Yes
- **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 with no symmetry can get separated from proper ones, and the symmetry can be verified in the reconstructed maps.
- **Initial low-pass filter (Å)**: 50
  One should NOT use high-resolution starting models as they may introduce bias into the refinement process.

On the *Optimisation* tab set **Number of classes to 4**. Press Execute to launch the protocol. Once it has finished, analyze the results. In our case we can see that one of the classes contains most of the particles (Figure 16). You may also check the plots for 3D angular distribution or see how the particle distribution among the 3D classes changes over the course of classification (Figure 17).


Metadata: relion_viewer_volumes.sqlite 4 items (100 x 100)
File Display Tools Help
Cols Rows
Block Volumes
Close
Volumes
UCSF Chimera
relion_it025_class004.mrc opened
Volume: Runs/001830_ProtRelionClassify3D/extra/relion_it025_class003.mrc:mrc (100 x 100 x 100)
File Display Tools Metadata Help

	enabled	id	label	comment	_index	_filename	_samplingRate	_classId	_halfMapFilenames
1	✓	1			0		2.6550	0	
2	✓	2			0		2.6550	0	
3	✓	3			0		2.6550	0	
4	✓	4			0		2.6550	0	

Figure 16: Resulting 3D volumes after Relion classification (top left) displayed as slices (bottom) and in UCSF Chimera (top right).

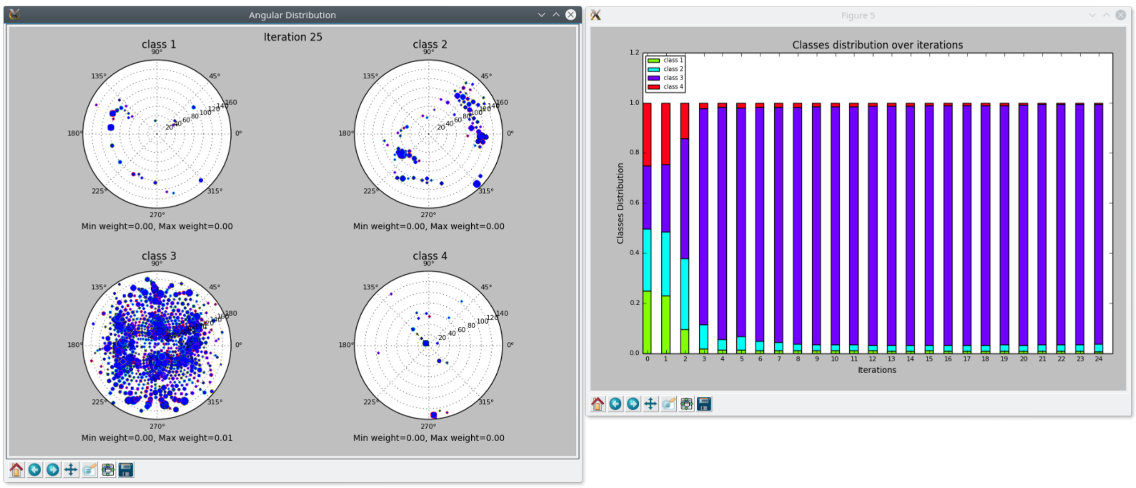


Figure 17: Left: 3D angular distribution of each 3D volume. Right: The overall particle distribution among the classes.

Similarly to the 2D classification analysis, select and save a particles subset from the best 3D class. Additionally, save the 3D volume for this class (this will produce a set of volumes with 1 item) - we will use it as a reference map.

4.2 High-resolution 3D refinement

Once a subset of sufficient homogeneity has been selected, one may use the 3D auto-refine procedure 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 and minimize overfitting. Combined with a procedure to estimate the accuracy of the angular assignments, 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 datasets.

4.2.1 Re-extracting particles with less down-scaling

Before executing the 3D refinement protocol we should first re-extract the particles at a smaller pixel size, so that we can potentially go to higher resolution. To do this, first use `pwem - extract coordinates` protocol, using as input the resulting particles from the best class of the 3D classification job. Set the option **Apply particles shifts? to Yes** to re-center the coordinates using the shifts estimated during 3D classification. After obtaining a new set of coordinates, extract the particles again. This time use the **box size of 360 px and re-scale it to 256 px**. This will limit our maximum achievable resolution (Nyquist frequency) to 2.5 Å (the pixel size will be 360*0.885/256=1.244 Å/px), which is probably enough for such a small dataset.

4.2.2 Running 3D auto-refine

Since we have changed the pixel size of our particles, we need to re-scale our 3D reference map to the 256-pixel box size as well. This can be done with `relion - crop/resize volumes` protocol using the following options:

- **Input volumes**: set of volumes (1 item) produced above
- **Rescale volumes?**: Yes
- **New sampling rate (Å/px)**: 1.244
  We have calculated this above. You can also find this number on the Summary tab of the previous particle extraction protocol.
- **Rescale volumes to a new box?**: Yes

- **New box size (px)**: 256

Once you have extracted the new set of particles and re-scaled the reference map to the same pixel size, you are ready to go for 3D refinement. Execute the protocol 3D ≫ Refine ≫ relion - 3D auto-refine with the following parameters:

On the *Input* tab:

- **Input particles**: new set of particles (256 px box size).
- **Particle mask diameter (Å)**: 200

On the *Reference 3D map* tab:

- **Input volume(s)**: the re-scaled volume of the best 3D class.
- **Is initial 3D map on absolute greyscale?** No, because we have re-scaled the reference volume and the grey values no longer match the particle data.
- **Symmetry**: D2
  We now aim for high-resolution refinement, so imposing symmetry will effectively quadruple the number of particles.
- **Initial low-pass filter (Å)**: 50
  We typically start auto-refinements from low-pass filtered maps 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.

Parameters on the *CTF* and *Auto-sampling* tabs remain default. Note that the initial angular sampling on the latter tab will only be used in the first few iterations, from there on the algorithm will automatically decrease the angular sampling rates until convergence.

As the MPI nodes are divided between one master (who does nothing else than bossing the others around) and two sets of workers who do all the work on the two half-sets, it is most efficient to use an odd number of MPI processors, and the minimum number of MPI processes for 3D auto-refine jobs is 3. 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 dataset you may want to run with as many threads as you have cores on your cluster nodes. On our machine we used 3 MPIs and 4 threads

Once the protocol is running, you can follow its progress on the **Summary** tab which will show the resolution for the current iteration. You may also click on Analyse Results and inspect the viewer for this protocol.

4.3 Mask creation and 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** and provided a mask for the refinement). 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 is called *post-processing*. First however, we'll need to make a mask to define where the protein ends and the solvent region starts.

4.3.1 Making a 3D mask

Locate the protocol `relion - create 3D mask` and provide the following parameters:

- **Input volume**: use the output volume from the 3D refinement job.
- **Lowpass filter map by (Å)**: 15.0
- **Initial binarisation threshold**: 0.005
  This should be a threshold at which rendering of the low-pass filtered map in ChimeraX shows absolutely no noisy spots outside the protein area. Move the threshold up and down to find a suitable spot. Often good values for the initial threshold are around 0.01-0.04.
- **Extend binary mask by (px)**: 3
  Use this to make your initial binary mask less tight.
- **Add a soft-edge (px):** 6
  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.

4.3.2 Postprocessing

Now find `relion - post-processing` protocol and input the parameters below:
On the *Input* tab:

- **Select a previous refinement protocol**: use the last 3D refinement run.
- **Solvent mask**: select the mask created in the step above.
- **Calibrated pixel size (Å)**: 0.0
  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. 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 *Sharpening* tab:

- **MTF of the detector**: select **mtf_k2_200kV.star** file that you have downloaded before
- **Original detetor pixel size (Å)**: 0.885
  This is the original pixel size in the raw (non-super-resolution!) movies.

Leave the other options as default and run the protocol. Open the results and display the masked (and sharpened) volume. Check FSC curves and the Guinier plots for this structure. Make sure that the FSC of the phase-randomized map (the red curve) is more-or-less zero at the estimated resolution of the post-processed map. If it is not, then your mask is too sharp or has too many details. In that case create a stronger low-pass filter and/or a wider and softer mask in the step above and repeat the post-processing. Check the **Summary** tab, you should have reached about 3.2 Å resolution at this step.

5 CTF and aberration refinement

Next, we'll use the CTF refinement to estimate the asymmetrical and symmetrical aberrations in the dataset, to check whether there is any anisotropic magnification and to re-estimate per-particle defocus values. Running this job can lead to further improvements in resolution at a relatively minor computational cost, but it all depends on how flat your ice was (for per-particle defocus estimates), and how well you had aligned your microscope (for the aberrations). Let's start with the higher-order aberrations, to see whether this data suffered from beam tilt or trefoil (which are asymmetric aberrations), or from tetrafoil or an error in spherical aberration (which are symmetric aberrations).

5.1 Higher-order aberrations

Find and open `relion - ctf refinement` protocol. On the *Input* tab select:

- **Input particles**: select the output set of particles from the 3D auto-refinement.
- **Input Postprocess**: the corresponding post-processing protocol.

On the *Fit* tab:

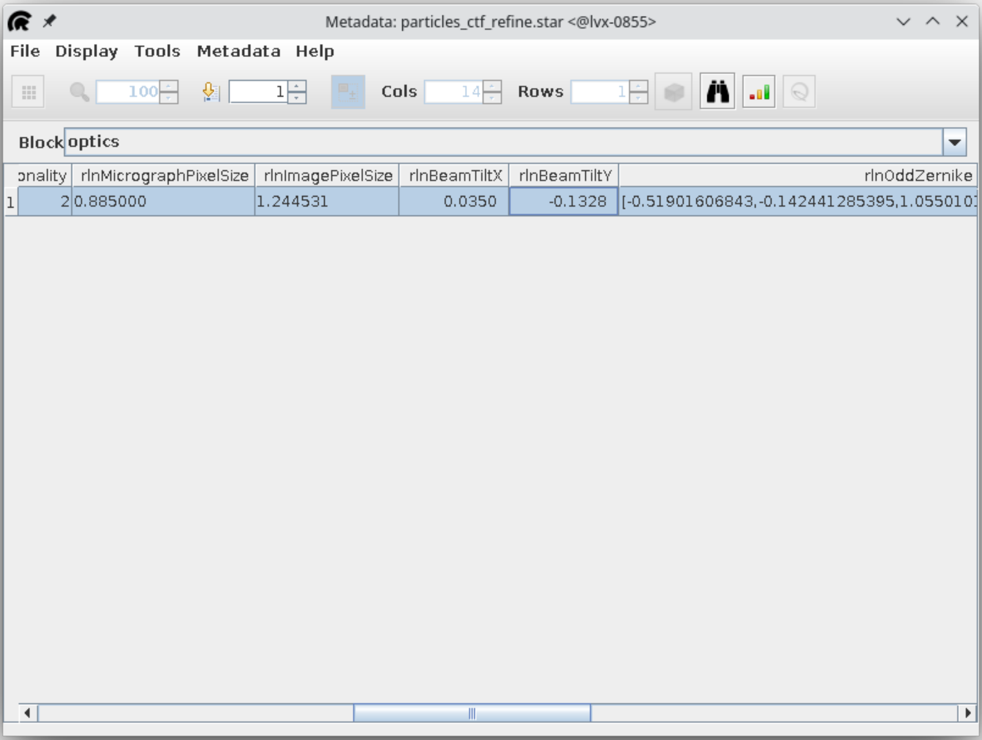
- **Estimate beam tilt?**: Yes
  In this example we're first looking for beam tilt, and do the anisotropic magnification and the per-particle defocus later. In general, one would try to first estimate the source of the largest errors.
- **Also estimate trefoil?**: Yes

- **Estimate 4th order aberrations?** Yes

  This is done mostly for illustrative purposes here. One would not expect a big improvement at the current resolution of 3 Å.

This program is only implemented on the CPU. Using 1 MPI and 8 threads, on our computer, this job has finished in less than a minute.

If you now open the results and display beam tilt / trefoil plots you'll see that this data actually suffered from some beam tilt: one side of the asymmetrical aberration image is blue, whereas the other side is red. You can find the values in the optics table of the output STAR file (click Browse for the CTF refinement protocol and find the *extra/particles_ctf_refine.star* file in the protocol folder):



Block optics

	onality	rlnMicrographPixelSize	rlnImagePixelSize	rlnBeamTiltX	rlnBeamTiltY	rlnOddZernike
1	2	0.885000	1.244531	0.0350	-0.1328	[-0.51901606843,-0.142441285395,1.055010

Figure 18: Open the *extra/particles_ctf_refine.star* file with SCIPION and display the *optics* table.

There was also a small error in the spherical aberration (see the 4-th order plot), as the symmetrical aberration image shows a significant, circularly symmetric difference (the image is blue at higher spatial frequencies, i.e. away from the center of the image). Importantly, for both the asymmetric and the symmetric aberrations, the model seems to capture the aberrations well.

If the data had suffered from trefoil, then the asymmetric aberration plot would have shown 3-fold symmetric blue/red deviations. If the data had suffered from tetrafoil, then the symmetric aberration plot would have shown 4-fold symmetric blue/red deviations. Examples of those are shown in the extended data fig. 8 of (Falcon et al., 2019).

5.2 Anisotropic magnification

Next, let's see whether these data suffer from anisotropic magnification. Create a copy of the previous protocol and replace the following values:

- **Input particles**: select the output set of particles from the previous CTF refinement.
- **Estimate anisotropic magnification**: Yes

Again, analyse the results by opening the viewer. There seem to be some blue-red trends, but the actual anisotropy is very small, as assessed from the *rlnMagMat??* values of the (2x2) transformation matrix in the optics table of the output STAR file.

5.3 Per-particle defocus values

Lastly, let's re-estimate the defocus values for each particle. Again, use the output from the previous job as input for this one:

- **Input particles**: select the output set of particles from the previous CTF refinement.
- **Perform CTF parameter fitting?**: Yes
- **Fit defocus?** per-particle
- **Fit astigmatism?** per-micrograph
- **Estimate beam tilt?**: No
- **Estimate 4th order aberrations?** No

The defocus estimation viewer has two plots: defocus stdev for the whole dataset (left) and per-particle defocus for each micrograph (right). Can you spot micrographs with a tilted ice layer?

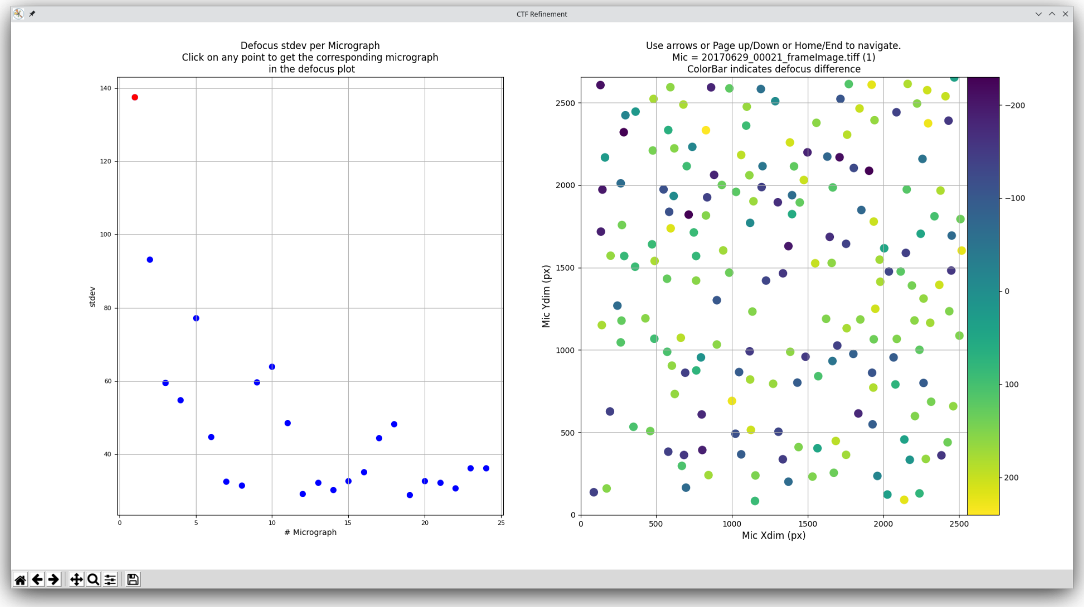


Figure 19: Defocus refinement plots.

EXERCISE

It is probably a good idea to re-run `3D auto-refine` and `post-processing` at this stage, so we can confirm that the new particle set actually gives better results. After another post-processing job, the resolution should improve to below 3 Å.

6 Bayesian polishing

Relion implements a Bayesian approach to per-particle, reference-based beam-induced motion correction. This approach aims to optimise a regularised likelihood, which allows us to associate with each hypothetical set of particle trajectories a prior likelihood that favors spatially coherent and temporally smooth motion without imposing any hard constraints. The smoothness prior term requires three parameters that describe the statistics of the observed motion. To estimate the prior that yields the best motion tracks for this particular dataset, one can first run the program in "training mode". Once the estimates have been obtained, one can then run the program again to fit tracks for the motion of all particles in the dataset and to produce adequately weighted averages of the aligned movie frames.

To save time, we will not run the training and proceed straight to polishing. But first, we need to run an extra protocol `relion - assign optics groups`. It can be used in many different cases described later, but here we need it to tell Scipion which gain reference was used during motion correction and whether its orientation has been changed or not. Open the protocol and fill in the parameters as below, then execute. The protocol will create an updated set of movies.

- **Input set**: use output movies set from the motion correction protocol.
- **Gain reference**: `$DATA/relion30_tutorial/Movies/gain.mrc`

Now create a relion - bayesian polishing protocol and fill in the values:
On the *Input* tab:

- **Input ALIGNED movies**: the movies set from the assign optics groups protocol.
- **Input particles**: the particles set from the last CTF refinement.
- **Input Postprocess**: the last post-processing protocol.

On the *Train or Polish* tab:

- **Operation**: Perform particle polishing
- **Sigma for velocity (Å/dose)**: 0.42
- **Sigma for divergence (Å)**: 1600
- **Sigma for acceleration (Å/dose)**: 2.61

Polishing algorithm runs only on CPU and may require a lot of memory (RAM). However, for our small dataset it took about 5 minutes to complete this job on our machine with 1 MPI and 24 threads. Using the results viewer we can now plot the scale-factors or B-factors used for the radiation-damage weighting.

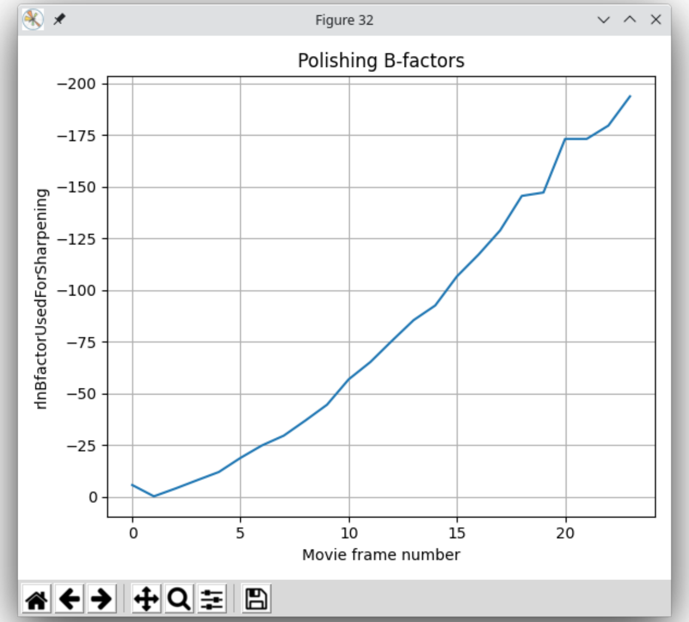


Figure 20: Per-frame B-factors plot.

EXERCISE

After polishing, the signal-to-noise ratio in the particles has improved, and one should submit a new 3D auto-refine job followed by post-processing job with this new set of ”shiny” particles. We leave this exercise to the user. In the end you should be able to get a map at overall resolution of 2.8 Å. Not bad for 3 GB of data, right?

7 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. *Relion* offers a post-processing-like procedure with a soft spherical mask that is moved around the entire map to estimate the local resolution.

Locate `relion - local resolution` protocol. It looks very similar to the post-processing job. On the *Input* tab select:

- **Select a previous refinement protocol**: choose a completed 3D auto-refine protocol
- **User-provided soft mask**: provide the same mask as for post-processing
- **Calibrated pixel size (Å)**: 1.244
- **Provide B-factor**: use the value from the last post-processing run
- **MTF of the detector**: select **mtf_k2_200kV.star** file that you have downloaded before
- **Original detetor pixel size (Å)**: 0.885

Run the protocol with 8 MPIs. Once completed, open the results and display the map in *ChimeraX*, colored by local resolution, using the resolution range of 2.5-4 Å.

ALTERNATIVES

- bsoft - blocres
- fsc3d - estimate resolution
- resmap - local resolution
- xmipp3 - local deepres
- xmipp3 - local monores

8 Advanced topics

8.1 Optics groups

A concept of optics groups has been originally implemented in *Relion* 3.1 to allow combining datasets with different aberration corrections. Nowadays, optics groups allow to keep most of information related to the data acquisition (pixel size, voltage etc.) separate from the particle metadata and also allow users to combine datasets with different box sizes or pixel sizes. SCIPION implementation of optics groups consists of a single string field (**opticsGroupInfo**) inside an *Acquisition* object that is populated during any import protocol run and associated with a set of items. This string contains the same optics table you would normally see in a STAR file. By default, the string is kept empty since in SCIPION voltage, pixel size and other parameters are kept in separate metadata fields. The *Relion* plugin in SCIPION takes care of creating optics table for STAR files automatically and usually users don't have to worry about it.

TIP

At the moment, SCIPION does not support processing of mixed data with different box/pixel sizes.

To allow support for multiple optics groups we have created `relion - assign optics groups` protocol that can be used to add extra information for movies, micrographs or particles sets at any point of the processing workflow. One can assign parameters either for a single optics group (MTF, beam tilt, gain reference, defects file) or multiple groups using an extra STAR file. The STAR file should contain:

- *data_optics* table with values for each group
- *data_micrographs* table with two columns:
  - *rlnMicrographName* that matches the same name in the input set
  - *rlnOpticsGroup* number that matches the group number in the optics table

The most common use cases for this protocol are:

- Right after the import step.
  For example, you are importing data from different microscope sessions or you already have the information about beam tilt groups and want to set this from the very beginning of your workflow. In this case you should run `assign optics groups` protocol on a joined set of items, providing an input STAR file with groups information. Remember, when you use `pwem - join sets` to join

sets with different optics the information will be lost because that protocol does not handle optics groups.

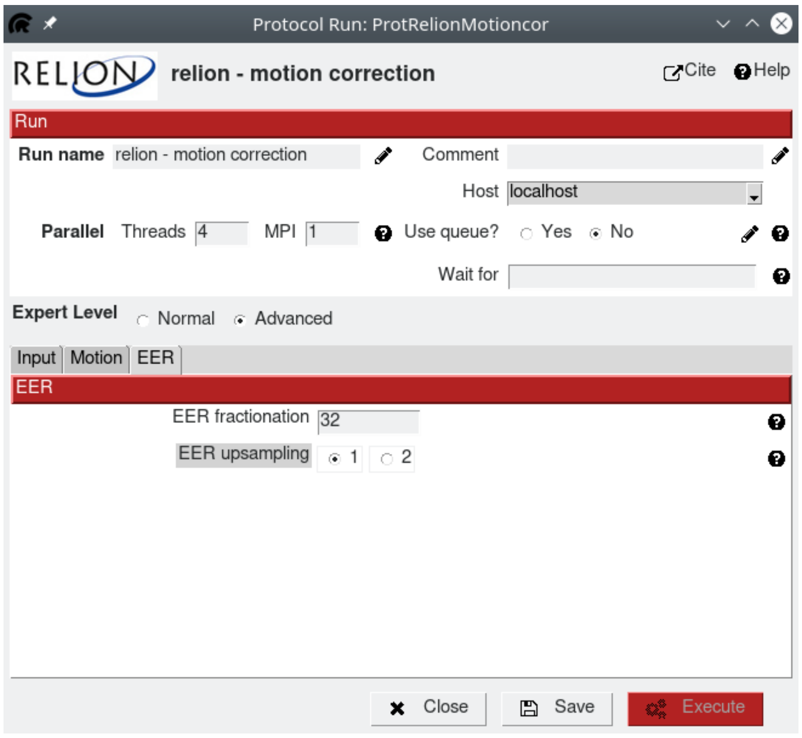
- Before running CTF refinement.
  If you have processed all of your data with a single optics group up to this point and want to separate the groups in order to have a more accurate refinement, now is a good time to assign optics to your particles.
- Before running bayesian polishing.
  As we did in the tutorial above, you need to assign optics to the output movies set from the motion correction protocol, unless your data has been gain-corrected before it was imported. This step is important for SCIPION to deduce gain image orientation, because it does not know if you e.g. have flipped your gain reference during motion correction.

8.2 EER data processing

EER (electron-event representation) is a new movie format for Falcon 4 Thermo Fischer Scientific cameras. Conventional representations of cryo-EM movies store pixel intensities for each exposure fraction. In contrast, in EER each electron-detection event is recorded as a tuple of position and time (x, y, time), indicating where and when the electron was detected on the sensor (Guo et al., 2020).

When importing EER movies into a SCIPION project, you should specify the dose per single EER frame during import step. For example, if your EER movies have 1000 frames each and your total dose was 30 e/Å^2, you should input 30/1000=0.03 as your dose per frame. Pixel size specified during `import movies` should be the physical pixel size unless you plan to render EER on 8K grid.

Relion can render any EER movie into a 4K x 4K or 8K x 8K movie with ”normal” frames using the upsampling parameter. You also need to tell it how many EER frames will be fractionated / joined together to create a standard movie. Both parameters are set in the `motion correction` protocol. The real advantage of *Relion* is on-the-fly frame rendering during motion correction, so the rendered movies are never saved to the disk. In the figure below we show the `relion - motion correction` protocol, where we group every 32 EER frames and render them on a 4K grid (upsampling = 1).


Protocol Run: ProtRelionMotioncor

relion - motion correction

Run

Run name: relion - motion correction
Comment:
Host: localhost
Parallel: Threads 4, MPI 1
Use queue? Yes [✓ No]
Wait for:

Expert Level: Normal [✓ Advanced]

Input | Motion | EER

EER

EER fractionation: 32
EER upsampling: [✓ 1] 2

Figure 21: EER options for Relion motion correction protocol.

The rule of thumb is to use the group size that gives you 0.5-1.25 e/Å^2/frame. In the example above we fractionate our EER movie into 1000/32=31 frames, so each new frame will have 30/31=0.97 e/Å^2 . Rendering on 8K grid (upsampling = 2) is not recommended for *Relion* because this can create artifacts around detector's defect lines.

TIP

If you want to change the rendering mode (4K or 8K) and/or the fractionation during bayesian polishing, this is not yet supported by Scipion . We hope to implement this feature soon!

You can see more details about EER format in the Relion documentation.

ALTERNATIVES

- motioncorr - movie alignment

8.3 Ewald sphere correction

Ewald sphere curvature is known to limit the resolution of large particles, particularly in the sub-2Å range. *Relion* implements a correction algorithm that is integrated into the 3D reconstruction routine (Russo and Henderson, 2018). The common way of doing the Ewald sphere correction within Scipion is:

1. Run relion - reconstruct protocol for *half1* and *half2* subsets of particles, providing extra arguments: **--ewald --mask_diameter ???**
2. Execute post-processing protocol from the command line, as you cannot provide two half-maps as inputs for relion - post-processing protocol yet
3. Repeat the two steps above, adding an extra argument **--reverse_curvature** and see which final map gives a better result

8.4 Local filtering during alignment with SIDESPLITTER

SIDESPLITTER (Ramlaul et al., 2020) is an algorithm that performs real-space filtering of both half-maps during 3D refinement according to signal-to-noise ratio (SNR). This helps to prevent overfitting in the regions with low SNR. The algorithm is supported by *Relion* reconstruction program. In order to use it you need to have the sidesplitter plugin installed and SIDESPLITTER_HOME correctly set. After that, adding **--external_reconstruct** to **Additional arguments** field in relion - 3D auto-refine or relion - 3D multi-body protocol is enough to activate it.

Below are the tips from **Takanori Nakane** on how to run *SIDESPLITTER* successfully:

- Start at relatively high resolution (10-15 Å)
- Use a very soft mask. A mask that cuts through a micelle is not good, even when it is soft.
- **Use solvent-flattened FSCs?**: Yes
- **Mask particles with zeros?**: No, fill with random noise.

With *SIDESPLITTER*, *Relion* is sometimes too pessimistic on angular accuracy and 3D refinement stops at coarse sampling. If so, run another 3D refinement with local search (angular sampling 1.8 deg, local search from 1.8 deg), possibly with **--minimum_angular_sampling 1.0**. If the refinement becomes very slow with **Mask particles with zeros?: No**, use **--maxsig 3000** as an extra argument.

ALTERNATIVES

- xmipp3 - deepEMhancer can also be used with **--external_reconstruct** (Ramírez-Aportela et al., 2022)

8.5 3D multi-body analysis

Relion offers yet another algorithm to deal with heterogeneous complexes: multi-body refinement. Briefly, the map gets split into two or more independently moving rigid bodies which are analyzed and refined separately. Upon convergence, principal component analysis may be used to characterize the most dominant motions in the complex. The detailed protocol for running multi-body refinement can be found here. The corresponding SCIPION protocol GUI looks very similar, so you could use the guide above for `relion - 3D multi-body` protocol as well.

8.6 Other useful Relion protocols

In this section we briefly describe some of the *Relion* protocols available in SCIPION that were not mentioned in the tutorial but might still be useful in certain cases.

1) `relion - subtract projection` protocol.

When dealing with heterogeneity, besides masked 3D auto-refinements and focused 3D classifications, one can consider using partial signal subtraction to remove part of the signal from particle images (Scheres, 2016). This allows to remove the inconsistency in comparing experimental projections with projections from masked references. The protocol usually starts from a consensus 3D refinement (of a whole structure) and applies a mask around the part of the map that one would like to keep. Optionally, subtracted particles can be re-centered on the mask or on arbitrary 3D coordinate. Re-centering can be particularly useful to make sure the center of the remaining part of the map matches the origin of rotations of the volume box. However, it's important to note that once the subtracted particles have been re-centered and later re-aligned, the new alignment cannot be applied to the original particles anymore.

2) A group of export protocols. These protocols can export results into a self-contained folder, allowing users to continue image processing outside SCIPION . Export of sets of coordinates, CTFs (micrographs with CTF information) and particles is possible.

3) `relion - clean project` protocol. Many *Relion* jobs create a lot of intermediate files which can be deleted to save the disk space. This protocol finds all finished *Relion* protocols in a SCIPION project and moves their intermediate files to a Trash folder. For iterations-based protocols it keeps the files only for the last iteration. The folder structure is preserved, so the data can be easily restored if necessary.

Moreover, the plugin for *Relion* includes several protocols which do not have their own GUI in the *Relion* interface. Such examples are the centering of class averages, movie compression, gain estimation for movie compression, volume symmetrization and a few others.

8.7 Parallel Relion jobs execution in Scipion

Many *Relion* programs implement a hybrid parallelisation scheme with MPIs for distributed-memory parallelisation and threads for shared-memory parallelisation. MPI is typically used to run on distinct computing nodes of a cluster (i.e.

they cannot share memory), while the threads make full use of the shared memory of modern multi-core CPUs. Moreover, some *Relion* programs have been GPU-accelerated. Below we provide recommendations for running different *Relion* job types, which have originally been written by **Takanori Nakane** for MRC LMB computing cluster. Remember that you should always consult your IT department first on the proper use of cluster resources and on how to submit jobs.

8.7.1 GPU jobs

The jobs listed below have been GPU-accelerated. To enable GPU computing, go to the *Compute* tab of a protocol and set **Use GPU acceleration? Yes**. If **Which GPUs to use** is left empty, the job itself will try to allocate available GPU resources (this might depend on your cluster configuration and submission template). You can override the default allocation by providing a list of GPUs (0,1,2,3, etc) to use. MPI-processes are separated by "**:**" and threads - by "**,**". For example, submitting a job with 4 MPIs and 3 threads with **Which GPUs to use "2:2:1,3"** would mean that worker 1 runs 3 threads on GPU2, worker 2 runs 3 threads on GPU2, worker 3 distributes 3 threads as evenly as possible across GPU1 and GPU3. This syntax might look complicated and it may be a better choice to leave processes allocation to the cluster queue manager, e.g. SLURM.

Let's now consider a GPU cluster node or a GPU workstation that has 32 CPU hyper-threaded cores, 4 GPUs and 128 GB of RAM.

- `relion - 3D classification`: 5 MPI x 8 threads. Odd number of MPIs is necessary to have a separate MPI master process. Since that process only coordinates the workers and does not perform actual computations, this is effectively (5-1)*8=32 cores. If you have not provided anything in **Which GPUs to use** field, each MPI worker will run on a separate GPU. Be aware, if you try to run more than 1 MPI process per GPU you may ran out of GPU memory. This is more relevant for 3D jobs.
- `relion - 3D auto-refine`: same as above
- `relion - 3D multi-body`: same as above
- `relion - 3D initial model`: the new gradient refinement algorithm does not support MPIs yet, so you can only use threads.
- `relion - 2D classification`: this job runs faster with 2 MPIs per GPU, so we recommend 9 MPI x 4 threads. If you are using **VDAM** algorithm, it does not support MPIs yet, so stick to using only threads.
- `relion - auto-picking (reference-based)`: this job does not use threads, so you can use e.g. 4 MPIs, one for each GPU.

TIP

Note that running a *Relion* job on 8 GPUs is inefficient and not recommended due to various overheads.

8.7.2 CPU jobs

Now let's consider a CPU cluster node with 112 hyper-threaded cores and 700 Gb of RAM. You should specify number of MPIs and number of threads to make the most of the allocated 112 cores. The best balance between MPI processes and threads depends on the job type and your dataset (box size, movie frames, etc) but we'd recommend:

- `relion - 3D classification`: 9 MPI x 14 threads. Odd number of MPIs is necessary to have a separate MPI master process. Since that process only coordinates the workers and does not perform actual computations, this is effectively (9-1)*14=112 cores. If you want to allocate two full nodes, just increase number of MPIs to 17: (17-1)*14=224. Remember, *Relion* does not scale very well over too many cores and becomes inefficient. Also, a very large number of MPIs will mean many processes will need to combine their results at the end of each iteration.
- `relion - 3D auto-refine`: same as above
- `relion - 3D multi-body`: same as above
- `relion - 2D classification`: same as above
- `relion - motion correction`: 14 MPI x 8 threads. Each MPI process will work on a different movie and will need at least $frame\_width * height * (frames + 2 + threads\_per\_mpi) * 4$ bytes of RAM. It is also recommended to have number of frames divisible by number of threads (e.g. for 40 frames use 1, 2, 4, 5, 8, 10, 20 or 40 threads).
- `relion - ctf refinement`: same as above, 14 MPI x 8 threads.
- `relion - bayesian polishing`: same as above. You might need to reduce number of MPIs if the frame recombination step runs out of RAM.
- `relion - particle extraction`: this job does not use threads, so you can use e.g. 28 MPIs.
- `relion - auto-picking LoG`: same as above

We also recommend you to read about optimizing your computations by using a scratch disk (preferably SSD). You can find all computation-related options on the *Compute* tab of any *Relion* protocol.

8.7.3 Non-Relion protocols parallelization

While *Relion* software handles parallelization through it's own MPI/threads interface, other protocols might not implement such an interface. In this case, Scipion can handle jobs execution by itself. For example, `motioncorr - movie alignment` protocol used in this tutorial can only run on one or more GPUs. Due to large memory requirements, motioncor2 developers strongly recommend to run only a single process per GPU. Scipion provides a way to allocate multiple GPUs while keeping this constraint. For example:

- If one sets Number of threads to 3 and GPU IDs to "0 1", then thread 1 will be the master, while thread 2 will use GPU 0 and thread 3 - GPU 1.
- If one chooses 5 threads and GPU IDs = "0 1", then two threads will run on GPU 0 and two threads on GPU 1 (which is not recommended for motioncor2).
- With 3 threads and GPU IDs set to "0 1 2 3" thread 2 will use GPUs 0 and 1, and thread 3 - GPUs 2 and 3. In this case a single movie will be processed by multiple GPUs.

The same parallelization scheme as above applies to `gctf - ctf estimation` protocol, which also supports multi-GPU execution.

8.7.4 Streaming (on-the-fly) processing

The general description of the streaming concept in Scipion is provided here. Many Scipion protocols now support streaming, from the import step up to 2D classification. The protocols for CTF estimation, particle extraction and auto-picking in most Scipion plugins also have a *Streaming* tab where a user can provide a **batch size**. By default (batch size = 1), all input protocol items will be processed one by one, which is common for on-the-fly scenarios. However, users can specify a larger number of items that will be then processed "in batches", which can sometimes decrease the computational load and speed up processing. This is particularly helpful for CTF estimation or particle picking protocols, where it's better to run it on several micrographs at once to minimize overhead. Setting this value to 0 is used for non-streaming cases when all input items are processed together.

TIP

CTTFind4 cannot process more than one input micrograph at once, so it will run on the micrographs one by one, despite the batch size parameter value. However, the batch size will group items into a single protocol step, which still minimizes the overhead.

Motion correction protocols do not support batch processing, so the items are always processed one by one. For instance, if you execute `motioncorr - movie alignment` with **GPU IDs set to "0 1 2 3"**, four movies will be processed in parallel, each on a separate GPU.

9 Wrapping up

That's it! If you have completed the tutorial and read and understood the guide up to this point, you should be ready for some serious SPA data processing using Scipion framework! If you found any mistakes in this guide or have more questions, get in touch with us. Below are the resources for further reading:

- Official Relion tutorial for SPA
- Our paper on using Relion within Scipion (Sharov et al., 2021). In this paper we go through another dataset using a more complex workflow
- Other Scipion tutorials for SPA

References

de la Rosa-Trevín, J., Quintana, A., del Cano, L., Zaldívar, A., Foche, I., Gutiérrez, J., Gómez-Blanco, J., Burguet-Castell, J., Cuenca-Alba, J., Abrishami, V., Vargas, J., Otón, J., Sharov, G., Vilas, J., Navas, J., Conesa, P., Kazemi, M., Marabini, R., Sorzano, C., and Carazo, J. (2016). Scipion: A software framework toward integration, reproducibility and validation in 3d electron microscopy. *J. Struc. Biol.*, 195(1):93 – 99.

Falcon, B., Zivanov, J., Zhang, W., Murzin, A. G., Garringer, H. J., Vidal, R., Crowther, R. A., Newell, K. L., Ghetti, B., Goedert, M., and Scheres, S. H. W. (2019). Novel tau filament fold in chronic traumatic encephalopathy encloses hydrophobic molecules. *Nature*, 568(7752):420–423.

Guo, H., Franken, E., Deng, Y., Benlekbir, S., Singla Lezcano, G., Janssen, B., Yu, L., Ripstein, Z. A., Tan, Y. Z., and Rubinstein, J. L. (2020). Electron-event representation data enable efficient cryoEM file storage with full preservation of spatial and temporal resolution. *IUCrJ*, 7(5):860–869.

Ramírez-Aportela, E., Carazo, J. M., and Sorzano, C. O. S. (2022). Higher resolution in cryo-EM by the combination of macromolecular prior knowledge and image-processing tools. *IUCrJ*, 9(5).

Ramlaul, K., Palmer, C. M., Nakane, T., and Aylett, C. H. (2020). Mitigating local over-fitting during single particle reconstruction with sidesplitter. *Journal of Structural Biology*, 211(2):107545.

Rohou, A. and Grigorieff, N. (2015). CTFFIND4: Fast and accurate defocus estimation from electron micrographs. *J. Struc. Biol.*, 192(2):216–221.

Russo, C. J. and Henderson, R. (2018). Ewald sphere correction using a single side-band image processing algorithm. *Ultramicroscopy*, 187:26–33.

Scheres, S. (2016). Chapter six - processing of structurally heterogeneous cryo-em data in relion. *Methods in Enzymology*, 579:125–157.

Scheres, S. H. W. (2012). RELION: Implementation of a bayesian approach to cryo-EM structure determination. *J. Struc. Biol.*, 180(3):519 – 530.

Sharov, G., Morado, D. R., Carroni, M., and de la Rosa-Trevín, J. M. (2021). Using *RELION* software within the *Scipion* framework. *Acta Crystallographica Section D*, 77(4):403–410.

Zheng, S., Palovcak, E., Armache, J.-P., Verba, K., Cheng, Y., and Agard, D. (2017). Motioncor2: anisotropic correction of beam-induced motion for improved cryo-electron microscopy. *Nature methods*, 14(4):331–332.