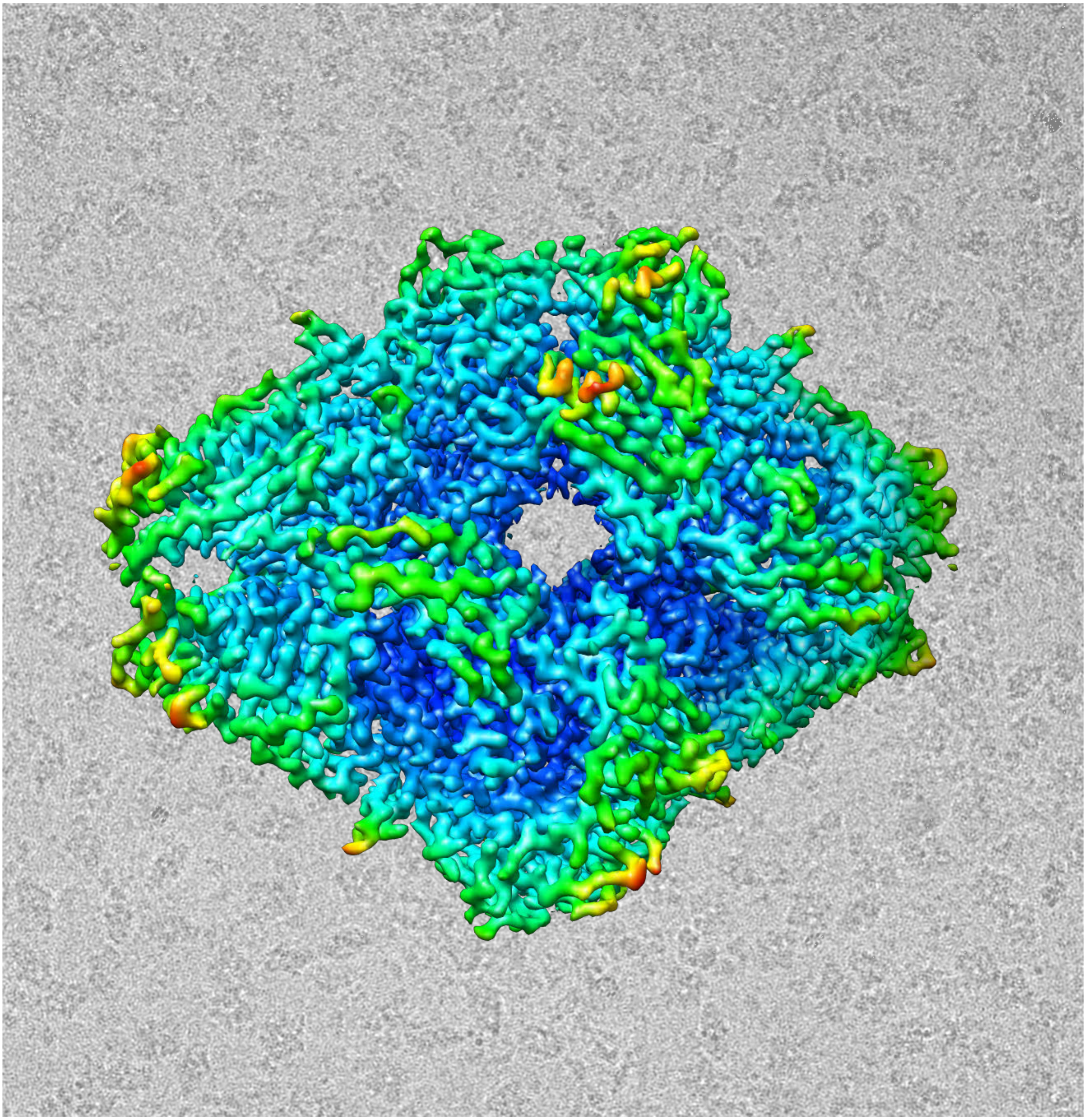
UNIVERSITY OF LEICESTER

Relion 3.1 β–galactosidase tutorial



Last Updated: 2021-03-10

Processing CryoEM Data

Recent innovations in camera technology and processing capabilities have resulted in the so-called ‘resolution revolution’ in cryo-electron microscopy. While we are now able to calculate cryoEM maps at very high resolution, the field is still young. As a result, the level of experience available in most structural biology labs for single-particle cryoEM analysis (SPA) is not yet on-par with that of NMR or Xray crystallography. This tutorial is meant to provide some of that knowledge in a structured way.

This tutorial uses Relion (J Struct Biol, 2012 Dec 180(3). 519-530, github.com/3dem/relion) for almost all processing tasks, and is based very heavily on the Relion tutorial (ftp.mrc-lmb.cam.ac.uk/pub/scheres/relion31_tutorial.pdf) written by Sjors Scheres and co-workers. If you’ve never done SPA before – complete the entire tutorial before moving to other packages. While the details of how each software package works will change, the underlying concepts won’t.

In addition to the Relion tutorial, we have collected much of our SPA experience here. As our experience and expertise evolves, we will hope to capture more information in this document. For example, we now routinely use AI based autopicking via Relion’s External job type. While this is not covered in the tutorial, there is a section at the end describing how to do this. We have also included specific parameters for how to submit jobs to our scheduling system, including differences between the tutorial data and an actual project.

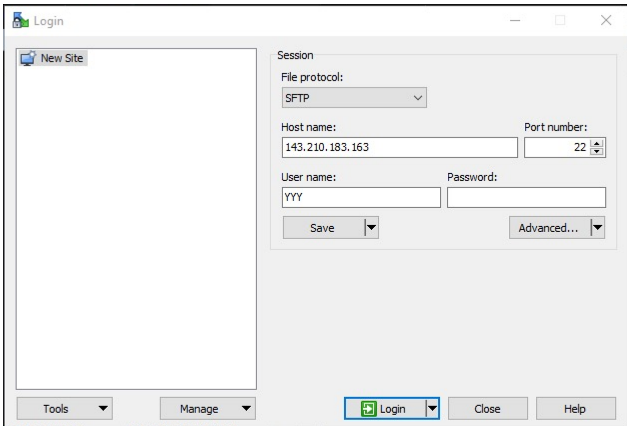
It’s important to emphasize that this is a collaborative production. We have incorporated suggestions and critiques from PI’s, facility managers, Post-docs, Ph.D. students and undergraduate students. If you feel you have something to contribute to this document, please let us know at cryoem@leicester.ac.uk.

Finally: about the style of this tutorial. SPA is done almost entirely on computer clusters that use the Linux operating system. If you don’t know Linux, or how to use the command-line, you can still use this tutorial – but you may find you capabilities greatly expanded by a separate Linux tutorial. Some of the parameters we use are extremely precise. If you have a problem, please check the parameters you’re using very carefully before raising an issue. The tutorial dataset is small, but you should be able to produce a reasonable quality map with it. If you are not able to successfully produce a map better than 4 Å – go back, and try again. You should not attempt to process real data until you can successfully process this data.

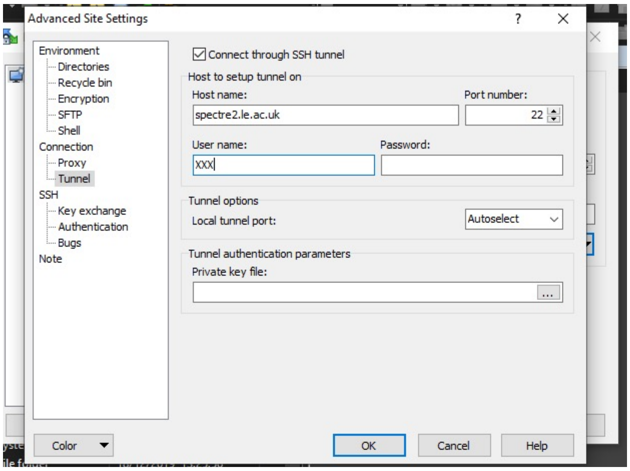
Downloading files from the image processing cluster (Windows)

Some parts of this tutorial require you to download maps to your computer. Here's how to do it if you have a Windows machine.

1. Install WinSCP (from *Program Installer* or winscp.net)
2. Open WinSCP
3. Add a new Site (Session > New Session…)
    1. Make sure protocol is 'SFTP'
    2. Fill in
        - Host name: 143.210.183.163
        - User name: your cluster username
        - Cluster password
    3. Setup Tunnel
        1. Click Advanced
        2. Choose the Tunnel page
        3. Fill in:
            - Host name: spectre2.le.ac.uk
            - User name
            - University password



File protocol: SFTP
Host name: 143.210.183.163
Port number: 22
User name: YYY
Password:



☑ Connect through SSH tunnel
Host to setup tunnel on
Host name: spectre2.le.ac.uk
Port number: 22
User name: XXX
Password:
Tunnel options
Local tunnel port: Autoselect
Tunnel authentication parameters
Private key file:

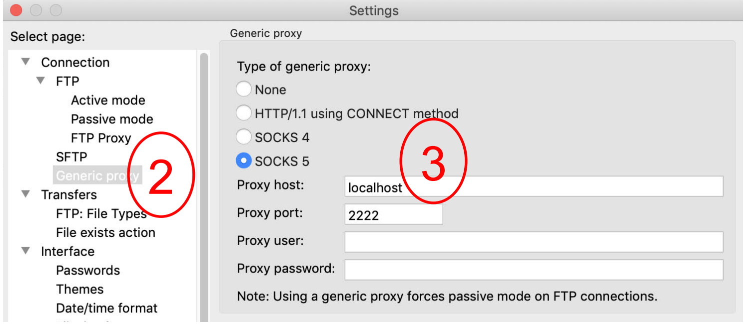
Downloading files from the image processing cluster (Mac)

Some parts of this tutorial require you to download maps to your computer. Here's how to do it if you have a Mac.

1. You will need a 'tunnel' to get the data:
   - Open a terminal window (Applications > Utilities > Terminal), and run:

   ```
   ssh -D 2222 -q -C -N XXX@spectre2.le.ac.uk
   ```

   Replacing XXX with your UoL username. You'll need to enter your UoL password. If the command succeeds, it not generate any errors and appear to hang. This is normal.
2. Download FileZilla Client (filezilla-project.org)
3. Uncompress the download by double-clicking on the .bz2 file
4. Open FileZilla
5. Setup Generic Proxy
   1. Open Settings (Edit > Settings…)
   2. Select *Generic Proxy* page
   3. Fill in
      - Type: Socks 5
      - Host: localhost
      - Port: 2222
   4. Click 'OK'



Generic proxy

Type of generic proxy:

○ None
○ HTTP/1.1 using CONNECT method
○ SOCKS 4
● SOCKS 5

Proxy host: localhost

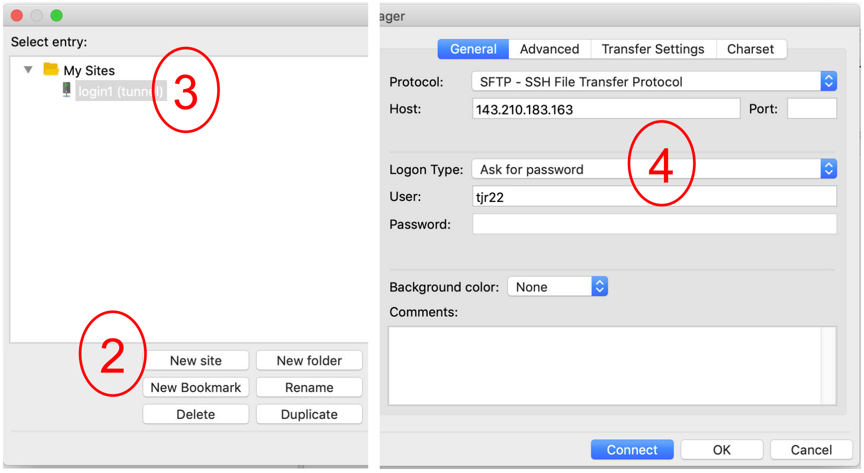
Proxy port: 2222

Proxy user:

Proxy password:

Note: Using a generic proxy forces passive mode on FTP connections.

6. Add a new 'Site'
   1. Open Site Manager (File > Site Manager…)
   2. Click on 'New site'
   3. Change the name
   4. Fill in
      - Protocol: SFTP
      - Host:143.210.183.163
      - User: your Username
   5. Click 'Connect'
   6. Enter your cluster password
   7. Agree to 'Trust Host?'



Protocol: SFTP - SSH File Transfer Protocol

Host: 143.210.183.163

Port:

Logon Type: Ask for password

User: tjr22

Password:

Background color: None

Comments:

Accessing the image processing cluster with NoMachine

NoMachine is the easiest way to access the image processing cluster. While not as powerful as other methods, it's more than enough for this tutorial. NoMachine works with Windows, Mac, and Linux.

- University Windows computer – Install using *Program Installer*
  - NoMachine
  - UCSF Chimera

- Non-Univsersity Windows machine, Mac, Linux
  1. Install NoMachine from:
     www.nomachine.com
  2. Install UCSF Chimera from:
     www.cgl.ucsf.edu/chimera
  3. Download the UoL HPC connection file for NoMachine from:
     uniofleicester.sharepoint.com/sites/Research-Computing/SitePages/nomachine-spectre.aspx
  4. Install the UoL connection into NoMachine by double-clicking on it.

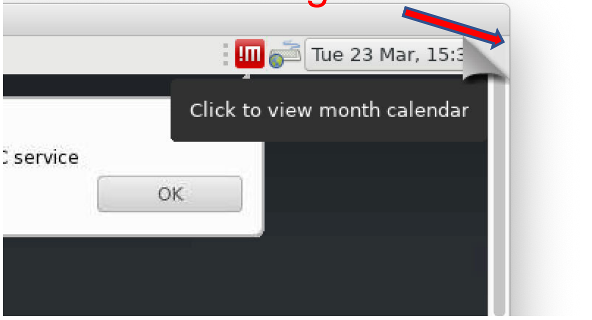
UNIVERSITY OF
LEICESTER

Accessing the image processing cluster with NoMachine

NoMachine is the easiest way to access the image processing cluster. While not as powerful as other methods, it's more than enough for this tutorial. NoMachine works with Windows, Mac, and Linux.

- Logging in to the cluster
  1. Double-click on the UoL HPC connection, then enter your UoL username and password
  2. Once logged in, change your view settings by hovering the mouse over the upper right-hand corner of the window (you may see the window slide across as you move to the corner)

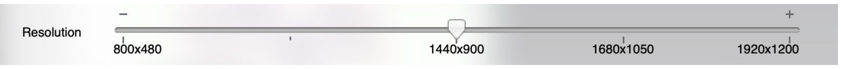
Hover mouse right here



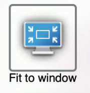
  3. Click on the 'Display' icon

  4. Click on the 'Change settings' icon

  5. Set your Resolution to at least 900 high (e.g. 1440x900)



  6. Click `Done`, then on the 'Fit to window' icon, then `Done`



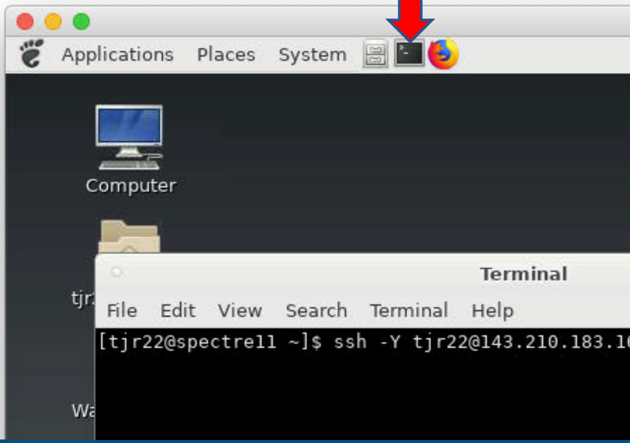
  7. Open a terminal, and log-in to the image processing cluster with:

     ssh –Y XXX@143.210.183.163

     Where XXX is your UoL username.

You'll be prompted for your password. <u>Use your *cluster* password</u>, not your UoL password. If you are asked if you want to trust this host, answer `yes`

Click here to open Terminal



```
[tjr22@spectrell ~]$ ssh -Y tjr22@143.210.183.1
```

Setting up tutorial project

Once logged in (make sure to connect X11), setup the Relion directory structure using the following commands:

1. Make the Relion project top-directory and a directory to contain the micrograph movies (commands are case-sensitive)

```
mkdir relion_tutorial
cd relion_tutorial
mkdir Micrographs
```

2. Symlink (DON'T COPY) the micrograph movies, and the gain reference, into your `Micrographs` directory (include the trailing **.** in your commands.)

```
cd Micrographs
ln -s /net/common/relion_3.1_tutorial/Micrographs/*.tiff .
ln -s /net/common/relion_3.1_tutorial/Micrographs/gain.mrc .
cd ..
```

3. Make sure that you've symlinked the data by listing the directory (you should see several tiff files and the gain reference file listed.)

```
ls Micrographs
```

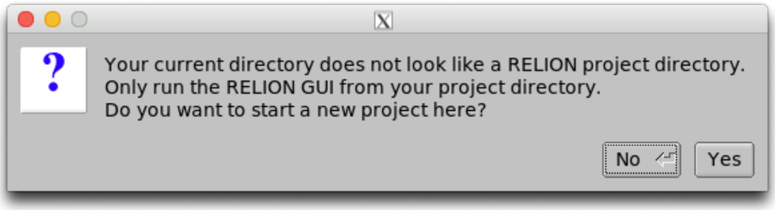
3. Examine the acquisition notes

```
cat /net/common/relion_3.1_tutorial/Micrographs/NOTES
```

4. Start Relion (and return your prompt).

```
relion &
```

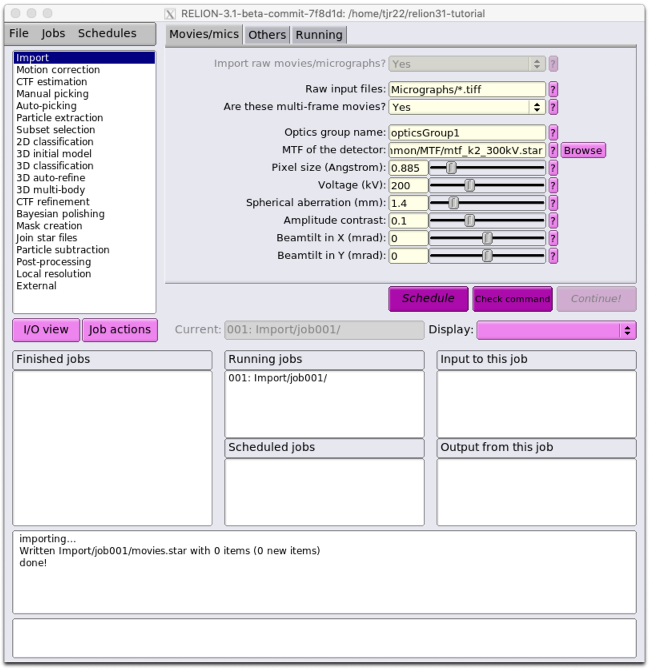
- Relion will ask you if you want to start a new project directory here. Answer 'Yes'.



Your current directory does not look like a RELION project directory.
Only run the RELION GUI from your project directory.
Do you want to start a new project here?

Importing micrographs / micrograph movies

Relion keeps track of what we do, in order to simplify the ‘pipeline’ of data processing for us. The first step is to import our data.



RELION-3.1-beta-commit-7f8d1d: /home/tjr22/relion31-tutorial

Import raw movies/micrographs? Yes
Raw input files: Micrographs/*.tiff
Are these multi-frame movies? Yes
Optics group name: opticsGroup1
MTF of the detector: mon/MTF/mtf_k2_300kV.star
Pixel size (Angstrom): 0.885
Voltage (kV): 200
Spherical aberration (mm): 1.4
Amplitude contrast: 0.1
Beamtilt in X (mrad): 0
Beamtilt in Y (mrad): 0

Current: 001: Import/job001/

Running jobs: 001: Import/job001/

```
importing...
Written Import/job001/movies.star with 0 items (0 new items)
done!
```

To start a new Relion Job, click on the job type in the Available Jobs box.

1. Click on `Import`
2. On Movies/mics tab:

`Import raw movies/micrographs?`
`Yes`

`Raw Input files:`
`Micrographs/*.tiff`

`Are these Multi-frame movies:`
`Yes`

`MTF of the detector:`
`/net/common/MTF/mtf_k2_300kV.star`

`Pixel size (Angstrom):`
(From `NOTES`)

`Voltage (kV):`
(From `NOTES`)

`Spherical aberration (mm):`
(From `NOTES`)

`Amplitude contrast:`
`0.1`

3. Click *Run!* ONCE

You can examine information Relion provides information about many of the options by clicking on the small `?`. Examine the information for `Amplitude contrast` before proceeding.

This job should run very quickly. In the output window you can see how many items were imported (should be 24.) Note that there’s a small idiosyncrasy that sometimes prevents Relion’s GUI from updating a job’s status unless you click on either it or on another job. Click on the import job now and it should move to the Finished jobs box.

STAR Files

Relion stores metadata in `.star` files. You can look at these files to find out more details about your processing. Let's examine the import star file in the terminal

```
ls
cd Import
ls
cd job001
ls
less movies.star
```

```
# version 30001

data_optics

loop_
_rlnOpticsGroupName #1
_rlnOpticsGroup #2
_rlnMtfFileName #3
_rlnMicrographOriginalPixelSize #4
_rlnVoltage #5
_rlnSphericalAberration #6
_rlnAmplitudeContrast #7
opticsGroup1            1 ../../../net/common/MTF/mtf_k2_300kV.star     0.885000
200.000000     1.400000     0.100000

# version 30001

data_movies

loop_
_rlnMicrographMovieName #1
_rlnOpticsGroup #2
Micrographs/20170629_00021_frameImage.tiff            1
Micrographs/20170629_00022_frameImage.tiff            1
…
```

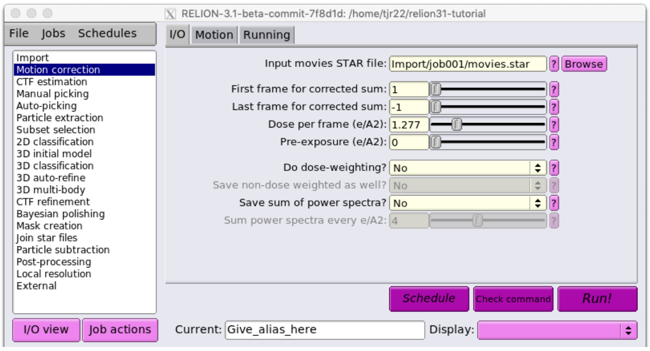
This is a basic STAR file. It has the following structure

- One data block (`data_optics`), called 'optics'
  - One table in that data block (`loop_`), without a name
  - Seven columns in the table, the first of which is called 'rlnOpticsGroupName' (`_rlnOpticsGroupName`)
  - One row in the table.
- A second data block called 'movies'
  - etc,…

Press '`q`' to quit `less`. Type `cd ~/relion_tutorial` to return to the `relion_tutorial` directory within your home directory.

Motion Correction (Relion’s implementation)

In order to reduce the impact of beam-induced motion on our images, we record movies. The first step in processing is to align the frames of those movies and then sum them up into a single image. Here, we’ll use Relion’s implementation of the MotionCor2 algorithm. We don’t normally use dose-weighting.



RELION-3.1-beta-commit-7f8d1d: /home/tjr22/relion31-tutorial

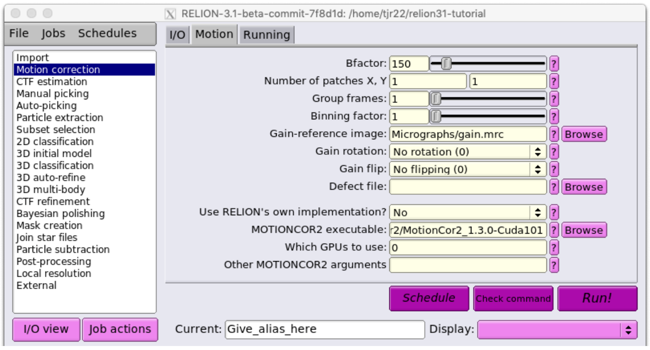
Input movies STAR file: Import/job001/movies.star
First frame for corrected sum: 1
Last frame for corrected sum: -1
Dose per frame (e/A2): 1.277
Pre-exposure (e/A2): 0
Do dose-weighting? No
Save non-dose weighted as well? No
Save sum of power spectra? No
Sum power spectra every e/A2: 4
Current: Give_alias_here
Display:

1. Click on the `Motion Correction` job in the Available jobs box
2. On `I/O` tab:

`Input movies Star file:`
`Import/job001/movies.star`

`Dose per frame:`
(From `NOTES`)

`Do dose-weighting?`
`No`



RELION-3.1-beta-commit-7f8d1d: /home/tjr22/relion31-tutorial

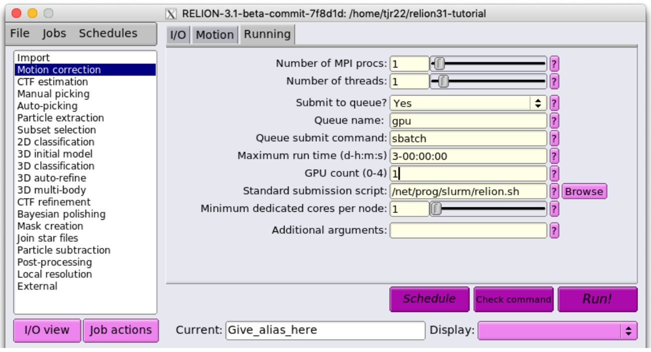
Bfactor: 150
Number of patches X, Y 1 1
Group frames: 1
Binning factor: 1
Gain-reference image: Micrographs/gain.mrc
Gain rotation: No rotation (0)
Gain flip: No flipping (0)
Defect file:
Use RELION's own implementation? No
MOTIONCOR2 executable: r2/MotionCor2_1.3.0-Cuda101
Which GPUs to use: 0
Other MOTIONCOR2 arguments
Current: Give_alias_here
Display:

3. On Motion tab:

`Number of patches X,Y`
`5          5`

`Gain-reference image:`
`Micrographs/gain.mrc`

`Use RELION’s own implementation?`
`Yes`



RELION-3.1-beta-commit-7f8d1d: /home/tjr22/relion31-tutorial

Number of MPI procs: 1
Number of threads: 1
Submit to queue? Yes
Queue name: gpu
Queue submit command: sbatch
Maximum run time (d-h:m:s) 3-00:00:00
GPU count (0-4) 1
Standard submission script: /net/prog/slurm/relion.sh
Minimum dedicated cores per node: 1
Additional arguments:
Current: Give_alias_here
Display:

4. On Running tab:

`Number of MPI processes:`
`1`

`Number of Threads:`
`4` (should be divisor of number of frames)

`Submit to Queue?:`
`Yes`

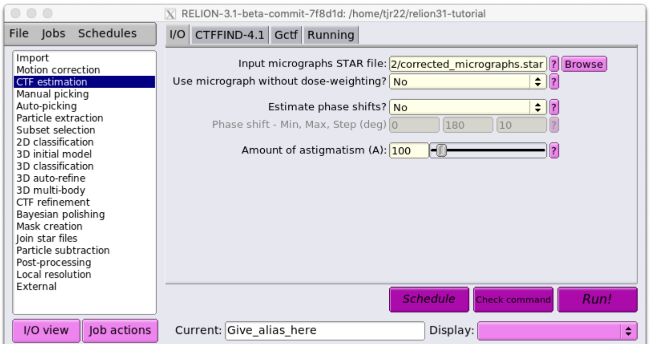
`GPU count:`
`0`

Click *Run!* ONCE

This job should take 4~5 minutes. On real datasets, use 4 MPI, but make sure that MPI × threads ≤ 32 (this is true for all jobs.) To ensure the best performance, make sure you run the job exclusively by changing `sbatch` to `sbatch --exclusive`.

CTF Estimation with CtfFind4

Once we have motion-corrected micrographs, we can estimate the CTF parameters which will be required later on for reconstruction. We prefer CtfFind4 (via the Relion GUI) for this.


RELION-3.1-beta-commit-7f8d1d: /home/tjr22/relion31-tutorial

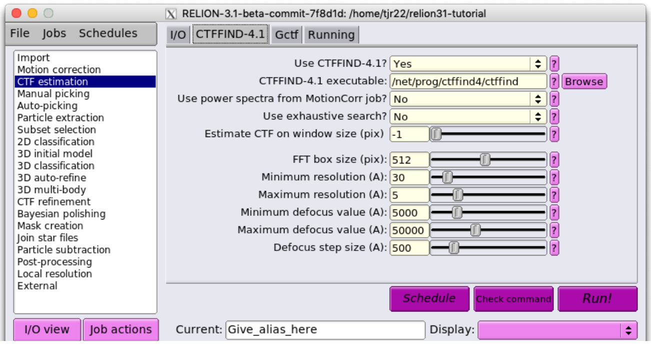
Input micrographs STAR file: 2/corrected_micrographs.star
Use micrograph without dose-weighting? No
Estimate phase shifts? No
Phase shift - Min, Max, Step (deg) 0 180 10
Amount of astigmatism (A): 100
Current: Give_alias_here

1. On I/O tab:

`Input micrographs STAR file:`
`MotionCorr/job002/corrected_micrographs.star`

`Use micrograph without dose weighting:`
`No` (Only say `Yes` if you did dose-weighting and also saved non-dose-weighted images)

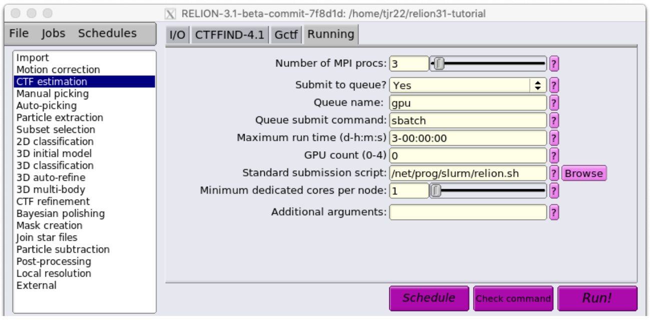
`Estimate phase shifts?`
`No` (Only say `Yes` for phase-plate data)


RELION-3.1-beta-commit-7f8d1d: /home/tjr22/relion31-tutorial

Use CTFFIND-4.1? Yes
CTFFIND-4.1 executable: /net/prog/ctffind4/ctffind
Use power spectra from MotionCorr job? No
Use exhaustive search? No
Estimate CTF on window size (pix) -1
FFT box size (pix): 512
Minimum resolution (A): 30
Maximum resolution (A): 5
Minimum defocus value (A): 5000
Maximum defocus value (A): 50000
Defocus step size (A): 500
Current: Give_alias_here

2. On CTFFIND-4.1 tab:

`Use CTFFIND4.1?`
`Yes`
`FFT box size (pix):`
`512` (Use `1024` or `2048` for datasets that have very weak signals)


RELION-3.1-beta-commit-7f8d1d: /home/tjr22/relion31-tutorial

Number of MPI procs: 3
Submit to queue? Yes
Queue name: gpu
Queue submit command: sbatch
Maximum run time (d-h:m:s) 3-00:00:00
GPU count (0-4) 0
Standard submission script: /net/prog/slurm/relion.sh
Minimum dedicated cores per node: 1
Additional arguments:

3. On Running tab:

`Number of MPI procs:`
`3` (20 for real data)

`Submit to queue?`
`Yes`

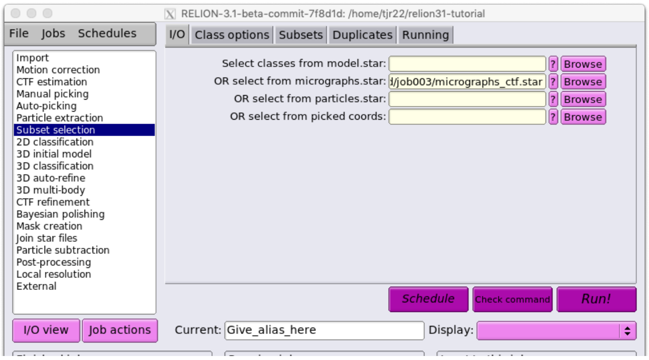
`GPU count (0-4):`
`0`

Click *Run!* ONCE

This job should take 10~30 seconds. On real datasets, use 16 MPI.

Selecting good micrographs via CTF estimation

The first step in finding good particles is to select good micrographs. The easiest way to do this is by eliminating micrographs with obviously bad CTF estimates. We'll use Relion's subset selection job for this task



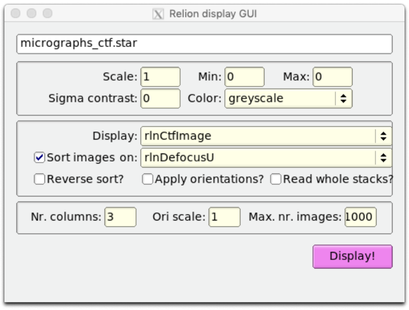
Select classes from model.star:
OR select from micrographs.star: l/job003/micrographs_ctf.star
OR select from particles.star:
OR select from picked coords:
Current: Give_alias_here

1. On I/O tab:

`OR select from micrographs.star:` `CtfFind/job003/micrographs_ctf.star`
(Make sure all the other fields are empty.)

2. Alias the job In `Current:` `all_good_micrographs`

Click ***`Run!`***

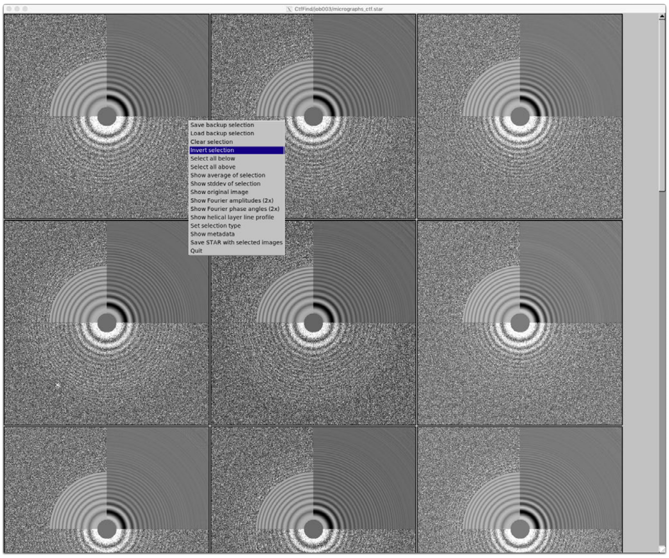


micrographs_ctf.star
Scale: 1 Min: 0 Max: 0
Sigma contrast: 0 Color: greyscale
Display: rlnCtfImage
☑ Sort images on: rlnDefocusU
☐ Reverse sort? ☐ Apply orientations? ☐ Read whole stacks?
Nr. columns: 3 Ori scale: 1 Max. nr. images: 1000

On `Relion Display GUI`:

1. Click the box beside `Sort images on:`
2. Sort on `rlnDefocusU`
3. `Nr. Columns:` `3`
4. Click `Display!`

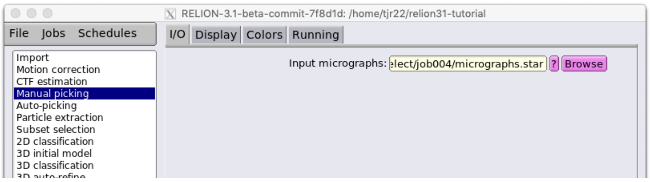
Each image has three parts. The background image is the power spectrum from your micrograph. The upper-left hand quadrant is the circularly averaged power spectrum, after taking into account the fitted astigmatism. The wedge in the upper right-hand quadrant is the fitted power spectrum.



1. Right-click and choose '`Invert selection`', this will select all micrographs.
2. Left click on any micrographs that don't have a good match between the calculated and measured CTF. Pay close attention to the first few and last few micrographs
3. Right-click and select `Save STAR with selected images`.
4. Repeat the selection procedure, sorting on `rlnDefocusU`, `rlnDefocusV`, `rlnCtfFigureOfMerit`, `rlnCtfAstigmatism`, and `rlnCtfMaxResolution`.
    1. Close the CTF display window.
    2. Change the `Sort images on:` selection
    3. `Display!`

Manual picking

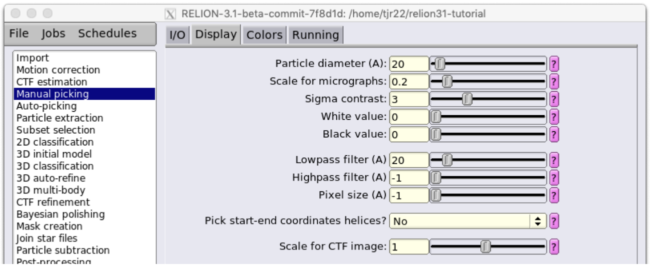
The best way to assess the quality of your data is via manual picking. We can also use the picked particles to generate 2D class averages for reference-based autopicking. Note here that the `Particle diameter` refers only to the size of the circle Relion draws on the micrographs and has nothing to do with the extracted particle box size used.



Input micrographs: ≥lect/job004/micrographs.star

1. On I/O tab:

`Input micrographs:`
`Ctffind/job003/micrographs_ctf.star`

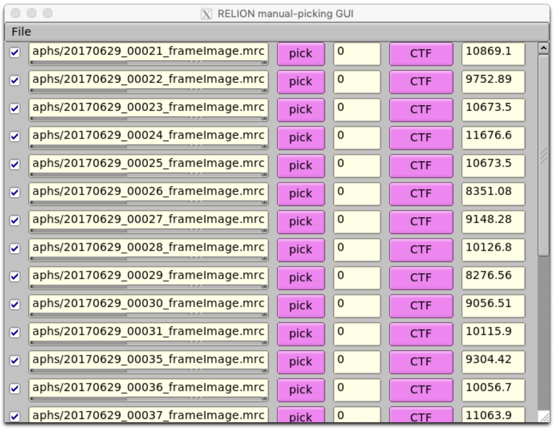


Particle diameter (A): 20
Scale for micrographs: 0.2
Sigma contrast: 3
White value: 0
Black value: 0
Lowpass filter (A) 20
Highpass filter (A) -1
Pixel size (A) -1
Pick start-end coordinates helices? No
Scale for CTF image: 1

2. On Display tab:

`Particle diameter (A):`
`20`

Click *Run!* ONCE. This will open the Manual-picking GUI



☑	aphs/20170629_00021_frameImage.mrc	pick	0	CTF	10869.1
☑	aphs/20170629_00022_frameImage.mrc	pick	0	CTF	9752.89
☑	aphs/20170629_00023_frameImage.mrc	pick	0	CTF	10673.5
☑	aphs/20170629_00024_frameImage.mrc	pick	0	CTF	11676.6
☑	aphs/20170629_00025_frameImage.mrc	pick	0	CTF	10673.5
☑	aphs/20170629_00026_frameImage.mrc	pick	0	CTF	8351.08
☑	aphs/20170629_00027_frameImage.mrc	pick	0	CTF	9148.28
☑	aphs/20170629_00028_frameImage.mrc	pick	0	CTF	10126.8
☑	aphs/20170629_00029_frameImage.mrc	pick	0	CTF	8276.56
☑	aphs/20170629_00030_frameImage.mrc	pick	0	CTF	9056.51
☑	aphs/20170629_00031_frameImage.mrc	pick	0	CTF	10115.9
☑	aphs/20170629_00035_frameImage.mrc	pick	0	CTF	9304.42
☑	aphs/20170629_00036_frameImage.mrc	pick	0	CTF	10056.7
☑	aphs/20170629_00037_frameImage.mrc	pick	0	CTF	11063.9

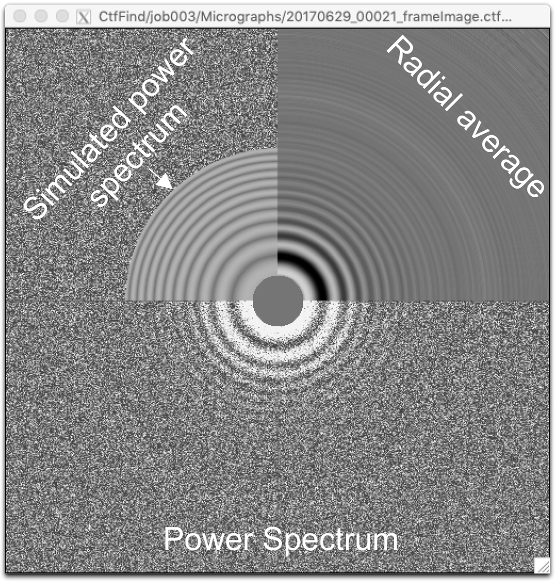
3. On manual-picking GUI:

Click CTF to examine the CTF fit

Click pick to manually pick particles

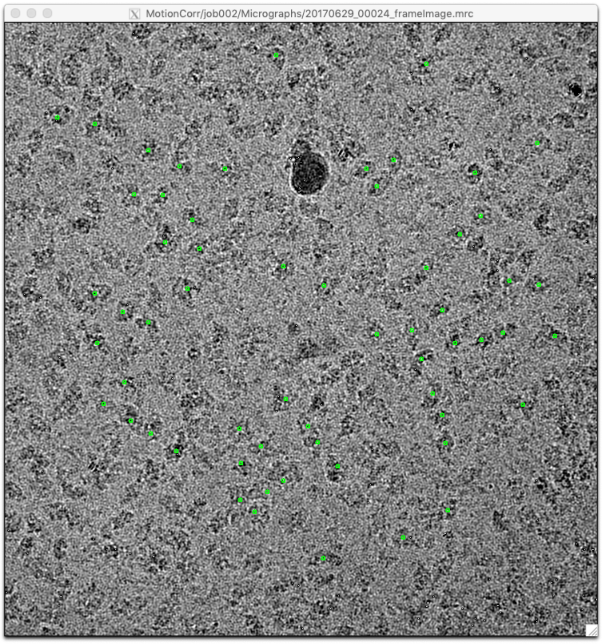
The CTF fit has three parts. The power spectrum of the micrograph, a simulation of the power spectrum based on the fit, and an “astigmatism compensated” radial average of your data.

1. Compare the radial average to the power spectrum. If the bright and dark circles match, that means that the radial average represents our data well.
2. Compare the power spectrum and radial average to the simulated power spectrum. If they match, that means that the CTF parameters describe our data well and particles from the micrograph are likely to be useful for further analysis.

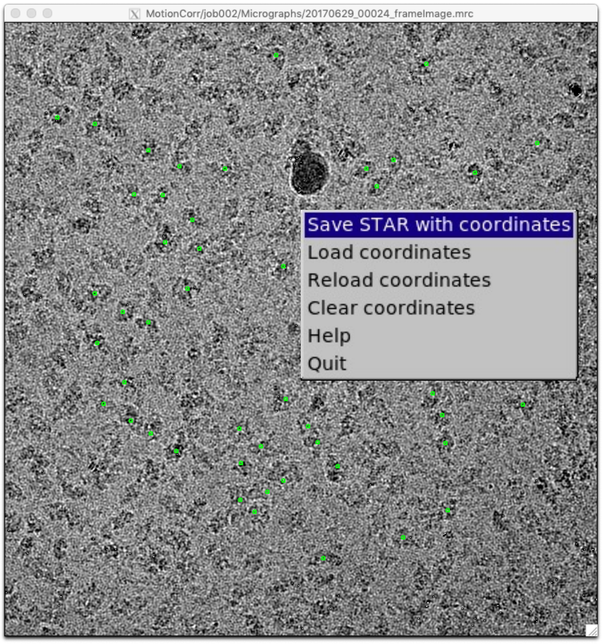


Manual picking

This is the first place in the processing that we can bias our results. If we think we ‘know’ what our particle looks like, we will only pick particles (or noise) that looks like that. Instead, our job is to just pick things we think might be particles. Be as vague as possible!



1. Pick particles with the left mouse button
2. Notice that the circles we get are more like dots. That’s why we set 20 Å particle diameter
3. Middle-click in the center of the marker to remove picked particle

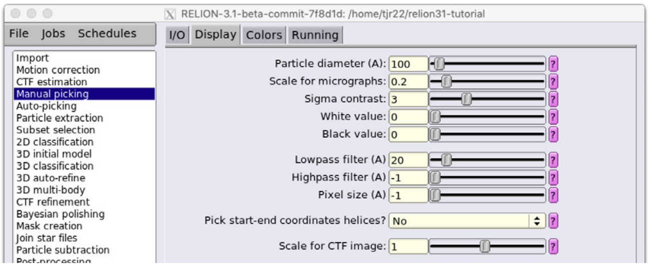


4. Right-click and `Save STAR with coordinates` on every micrograph before you close it, or you’ll lose your picks!
5. Now choose 500-1000 particles from micrographs with relatively high levels of defocus. Target 50-100 particles per micrograph.

Particle sizing

We need an initial guess for the size of our particles. If you have an idea from other sources (MAALS, PDB, etc,…) you can use that as an initial guess. But remember - your data is real, your initial guesses are not. You MUST choose your size based on the data!

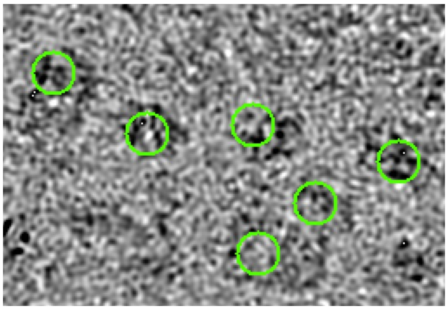
In order to guess the size, we adjust the particle diameter parameter and `Continue!` The manual picking job, checking as we go.



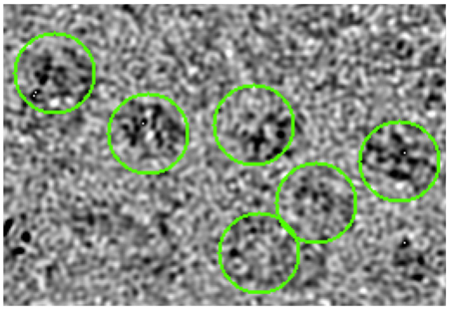
Particle diameter (A): 100
Scale for micrographs: 0.2
Sigma contrast: 3
White value: 0
Black value: 0
Lowpass filter (A) 20
Highpass filter (A) -1
Pixel size (A) -1
Pick start-end coordinates helices? No
Scale for CTF image: 1

1. Click on your `ManualPick/job005`
2. Change the `Particle diameter (A): 100`
3. Click `Continue!`
4. Examine a few micrographs
5. Try changing the scaling to make the micrographs bigger

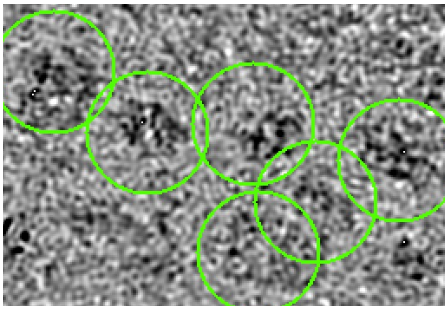
Too Small



Just Right



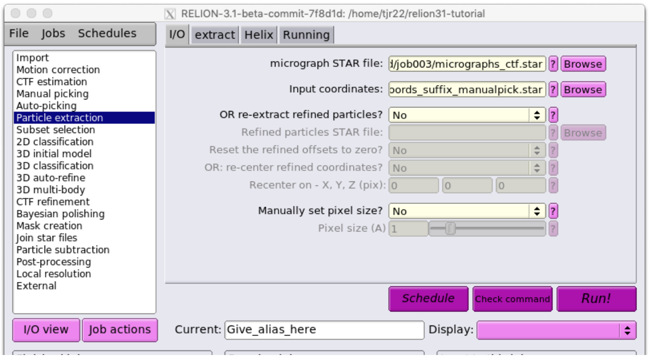
Too Big



Particle extraction

Relion operates on particle images extracted from micrographs. We need to decide how big an image to extract, and by how much we want to reduce the resolution (Å/pixel). If your picking is not exactly correct (and it won't be,) choosing a bigger box size will let Relion optimize the position.

As a guideline (and assuming our protein is approximately spherical,) we want to select a box size twice as large as the largest dimension of the particle. We will then rescale it as small as we can (to increase the calculation speed,) whilst maintaining enough resolution for this stage of processing (3~4 Å/pix.)



micrograph STAR file: d/job003/micrographs_ctf.star
Input coordinates: oords_suffix_manualpick.star
OR re-extract refined particles? No
Refined particles STAR file:
Reset the refined offsets to zero? No
OR: re-center refined coordinates? No
Recenter on - X, Y, Z (pix): 0 0 0
Manually set pixel size? No
Pixel size (A) 1
Current: Give_alias_here
Display:

1. On I/O tab:

`Micrograph STAR file:`
`Select/all_good_micrographs`
`/micrographs_ctf.star`

`Input coordinates?`
`ManualPick/jobXXX/`
`cords_suffix_manualpick.star`

Here is a sample calculations for a 200 Å maximum dimension particle. We use a box size twice as big as our particle, and use at most 4 Å /pixel resolution. Note that Relion requires an even number of pixels:

$$\left(\frac{Particle\ Size\ Guess\ (\mathring{A})}{Pixel\ Size\ (\mathring{A}/pix)}\right)\times 2 = BoxSize$$

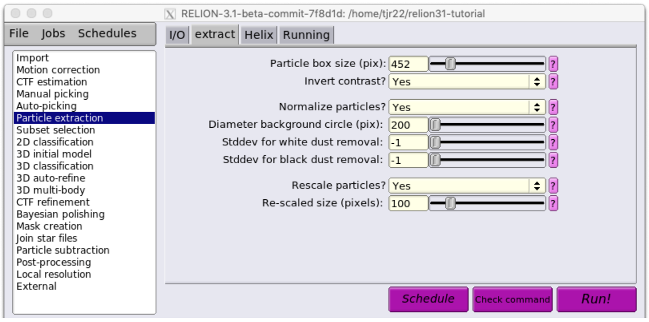
$$\left(\frac{200\ A}{0.885\ \ \mathring{A}/pix}\right)\times 2 = 451.9 \approx 452\ \ pix$$

$$BoxSize \times \left(\frac{Acquired\ Pixel\ Size\ (\mathring{A}/pix)}{Target\ Pixel\ Size\ (\mathring{A}/pix)}\right) = ExtractSize$$

$$452 \times \left(\frac{0.885\ \ \mathring{A}/pix}{4\ \mathring{A}/pix}\right) \approx 100\ pix$$

At this stage, don't exceed 80 pixel boxes - even if it means operating at 10 Å/pixel. Our goal right now is to eliminate junk, which you can still do very easily at very low resolution. If you want to make your boxes smaller, you can go as low as 60 pixels in the first instance.

Particle extraction



RELION-3.1-beta-commit-7f8d1d: /home/tjr22/relion31-tutorial

Particle box size (pix): 452
Invert contrast? Yes
Normalize particles? Yes
Diameter background circle (pix): 200
Stddev for white dust removal: -1
Stddev for black dust removal: -1
Rescale particles? Yes
Re-scaled size (pixels): 100

1. On Extract tab:

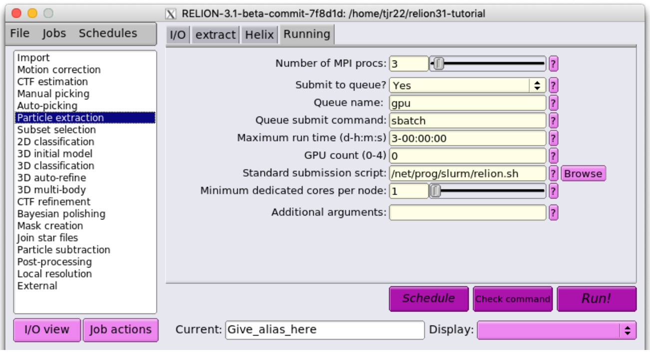
`Particle box size:`
`452`

`Invert contrast?`
`Yes` (`No` for negative stain)

`Normalize particles?`
`Yes`

`Rescale particles?`
`Yes`

`Re-scaled size (pixels):`
`64`



RELION-3.1-beta-commit-7f8d1d: /home/tjr22/relion31-tutorial

Number of MPI procs: 3
Submit to queue? Yes
Queue name: gpu
Queue submit command: sbatch
Maximum run time (d-h:m:s) 3-00:00:00
GPU count (0-4) 0
Standard submission script: /net/prog/slurm/relion.sh
Minimum dedicated cores per node: 1
Additional arguments:
Current: Give_alias_here
Display:

2. On Running tab:

`Number of MPI processes:`
`3` (`16-32` for real data)

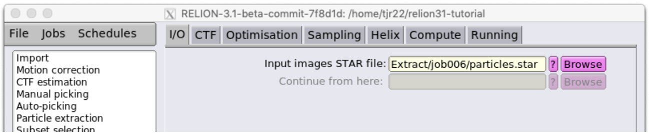
`Submit to queue?`
`Yes`

Click *Run!* ONCE

Note: Relion will produce a warning for every micrograph missing particle location files. This is normal. On our cluster, this job takes ~10 sec.

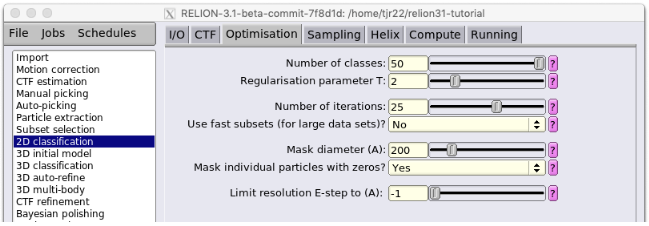
2D classification

For 2D classification, more MPI processes are preferred over more threads because 2D classification requires less memory than 3D classification or refinement. Unlike Motion correction, CTF estimation or Auto-picking, classification and refinement processes need to be coordinated by a master process. Here, we use 1 GPU, 2 MPI processes per GPU and 2 threads per MPI process.

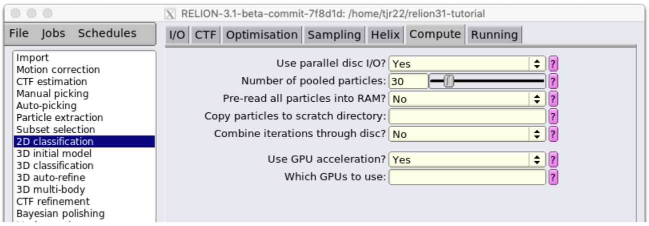


Input images STAR file: Extract/job006/particles.star
Continue from here:

1. On I/O tab:
`Input images STAR file:`
`Extract/job006/particles.star`



Number of classes: 50
Regularisation parameter T: 2
Number of iterations: 25
Use fast subsets (for large data sets)? No
Mask diameter (A): 200
Mask individual particles with zeros? Yes
Limit resolution E-step to (A): -1



Use parallel disc I/O? Yes
Number of pooled particles: 30
Pre-read all particles into RAM? No
Copy particles to scratch directory:
Combine iterations through disc? No
Use GPU acceleration? Yes
Which GPUs to use:

2. On Optimization tab:
`Number of classes:`
`50`

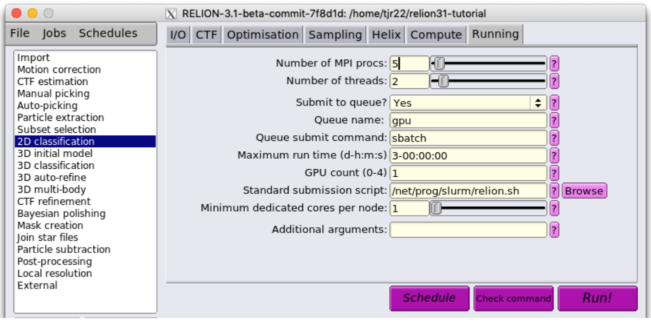
`Mask diameter:`
`250`

`Limit resolution E-step to (A):`
`30`

3. On Compute tab:
`Use parallel disc I/O:`
`Yes`

`Number of pooled particles:`
`30`

`Use GPU acceleration:`
`Yes`



Number of MPI procs: 5
Number of threads: 2
Submit to queue? Yes
Queue name: gpu
Queue submit command: sbatch
Maximum run time (d-h:m:s) 3-00:00:00
GPU count (0-4) 1
Standard submission script: /net/prog/slurm/relion.sh
Minimum dedicated cores per node: 1
Additional arguments:

4. On Running tab:
`Number of MPI procs:`
`3` (see comment below)

`Number of threads:`
`1` (see comment below)

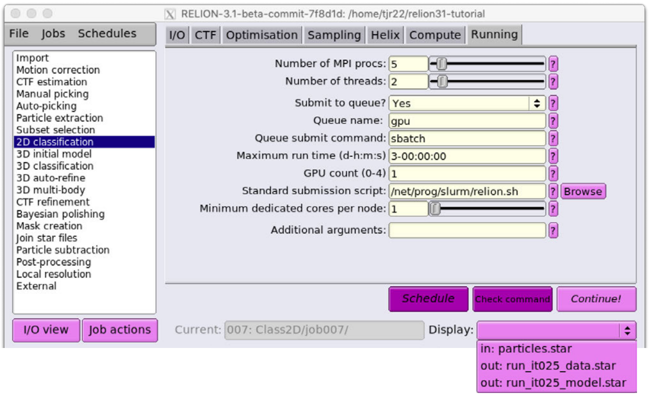
`Submit to queue?`
`Yes`

`GPU count (0-4):`
`1`

Click *Run!* ONCE
On our cluster, this job should take 2~3 mins.

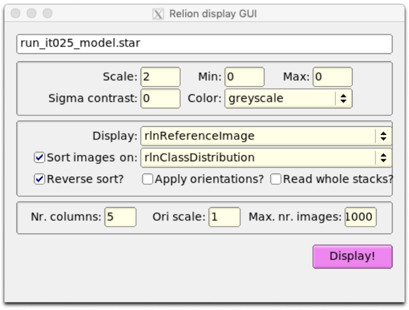
If you encounter an `out of memory` error, you'll need to reduce the number of MPI per GPU. Increase the number of threads to keep (MPI × Threads) ≤ (6 × GPUs). Always keep MPI × Threads ≤ 32.

2D classification analysis



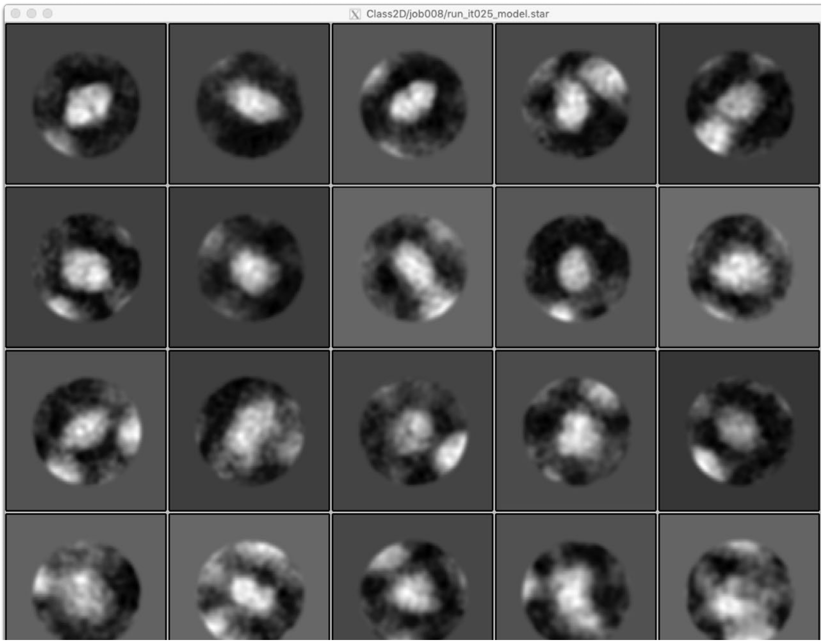
Number of MPI procs: 5
Number of threads: 2
Submit to queue? Yes
Queue name: gpu
Queue submit command: sbatch
Maximum run time (d-h:m:s) 3-00:00:00
GPU count (0-4) 1
Standard submission script: /net/prog/slurm/relion.sh
Minimum dedicated cores per node: 1
Additional arguments:
Current: 007: Class2D/job007/
Display:
in: particles.star
out: run_it025_data.star
out: run_it025_model.star

1. Click on `Display:` drop-down
2. Choose `out: run_it025_model.star`



run_it025_model.star
Scale: 2 Min: 0 Max: 0
Sigma contrast: 0 Color: greyscale
Display: rlnReferenceImage
☑ Sort images on: rlnClassDistribution
☑ Reverse sort? ☐ Apply orientations? ☐ Read whole stacks?
Nr. columns: 5 Ori scale: 1 Max. nr. images: 1000

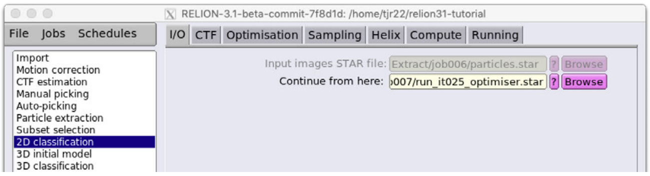
1. Change `Scale` to `2`
2. Sort on `rlnClassDistribution`
3. Reverse sort
4. Set the number of columns to 4
5. Click `Display!`



- Observations
  - Central particles visible
  - Other, nearby particles seen on the edge
  - Some, but not many internal features visible in particles
- Interpretation
  - Mask it too big – alignment being driven by location of adjacent particles
- Next step
  - Reduce mask size

2D classification

Rather than re-doing the 2D classification, we can change some parameters and continue the job. Because `Mask diameter` is one of the parameters we can change and continue the job, we'll do that by selecting the starting point, increasing the number of iterations by 5, and reducing the mask size.

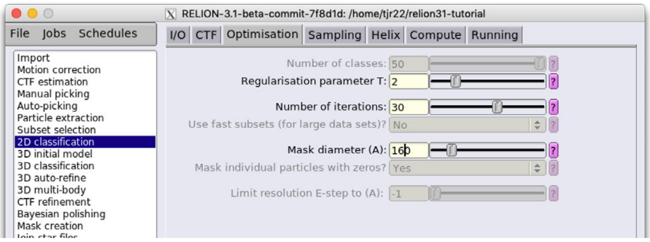


RELION-3.1-beta-commit-7f8d1d: /home/tjr22/relion31-tutorial

Input images STAR file: Extract/job006/particles.star

Continue from here: ›007/run_it025_optimiser.star

1. Click on the previous `2D classification` job in the `completed jobs` box to load the job
2. On I/O tab:
`Continue from here:`
(Select last iteration)



RELION-3.1-beta-commit-7f8d1d: /home/tjr22/relion31-tutorial

Number of classes: 50

Regularisation parameter T: 2

Number of iterations: 30

Use fast subsets (for large data sets)? No

Mask diameter (A): 160

Mask individual particles with zeros? Yes

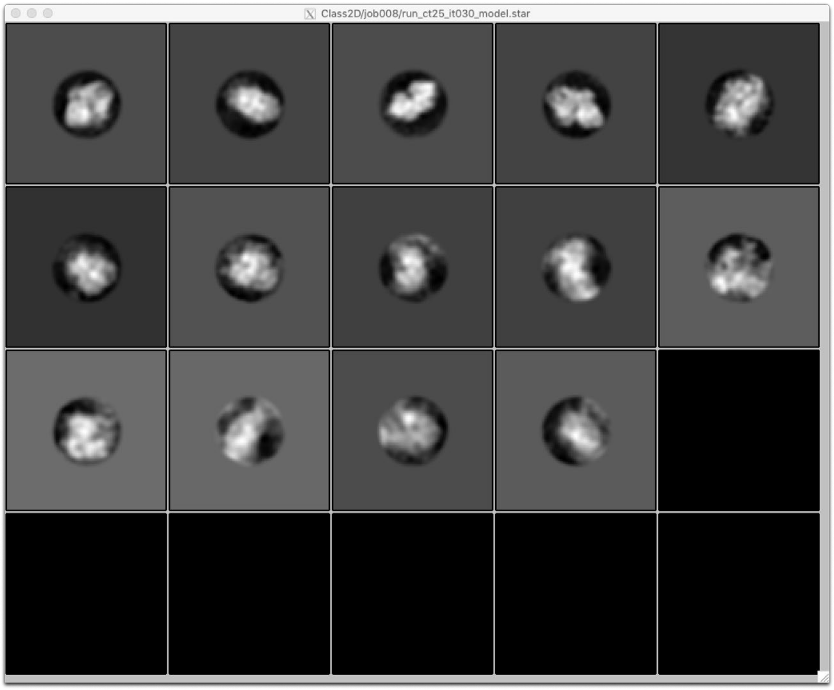
Limit resolution E-step to (A): -1

3. On Optimisation tab:
`Number of iterations:`
`30`

`Mask Diameter:`
`160`

Click *Continue!* ONCE

Once finished, `Display:`
`out: run_ct_25_it030_model.star`



Class2D/job008/run_ct25_it030_model.star

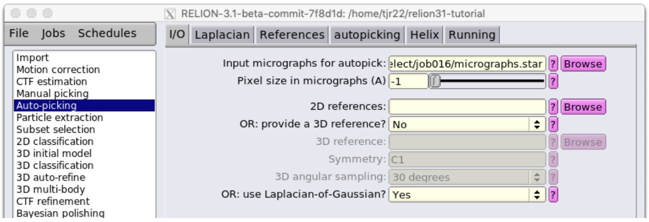
- Observations
    - Good classes
        - Single particle within mask
        - Particles entirely within mask
        - Internal features visible
        - Very dark background
    - Bad classes
        - 'Fuzzy'
        - Density extends to mask
        - etc,…
    - Empty classes
        - Relion tends to collapse 2D classes together. If this is a known issue. You can try turning on '`Ignore CTFs until first peak`' on CTF tab.
- Interpretation
    - Good classes sufficient for reference-based particle picking
- Next step
    - Reference-based picking

LoG picking

As an alternative to manual picking, we can use a generalized picker. These are less accurate than reference-based picking, but faster than manual picking. However, they do require accurate estimates of particle size to be effective, and the best way to do that is via manual picking and 2D classification.

Relion implements Laplacian-of-Gaussian (LoG) picking, which basically means it picks oval blobs within a set size range. Artificial-intelligence based pickers are more powerful, (and are available as External jobs now and likely to be included in Relion in the future,) but will use the same GUI approach.

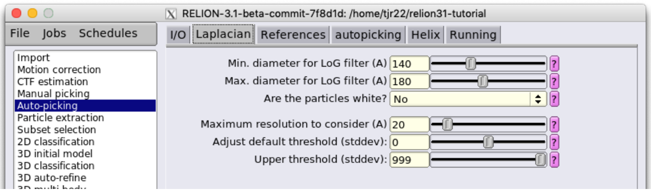
The first step is to select 3-5 micrographs across the defocus range (using the `OR select from micrographs.star` option in the `Subset selection` job. Select from your CTF estimation job, display `rlnCtfImage`, and sort by `rlnDefocusU`. Give the job a sensible alias, like `3_micrographs`.)



Input micrographs for autopick: elect/job016/micrographs.star
Pixel size in micrographs (A) -1
2D references:
OR: provide a 3D reference? No
3D reference:
Symmetry: C1
3D angular sampling: 30 degrees
OR: use Laplacian-of-Gaussian? Yes

1. On I/O tab:
`Input micrographs for autopick:`
`Select/3_micrographs/micrographs.star`

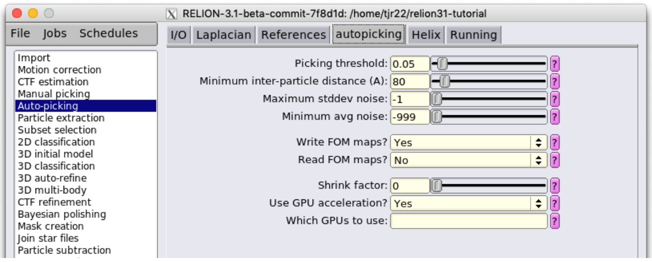
`OR use Laplacian-of Gaussians?`
`Yes`



Min. diameter for LoG filter (A) 140
Max. diameter for LoG filter (A) 180
Are the particles white? No
Maximum resolution to consider (A) 20
Adjust default threshold (stddev): 0
Upper threshold (stddev): 999

2. On Laplacian tab:
`Min. diameter for LoG filter (A):`
`140` (a bit smaller that the minimum size of your particle)

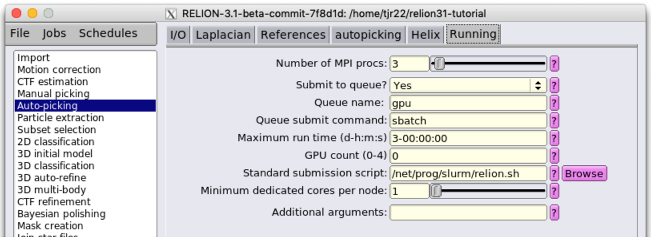
`Max. diameter for LoG filter (A):`
`180` (a bit bigger that the maximum size of your particle)



Picking threshold: 0.05
Minimum inter-particle distance (A): 80
Maximum stddev noise: -1
Minimum avg noise: -999
Write FOM maps? Yes
Read FOM maps? No
Shrink factor: 0
Use GPU acceleration? Yes
Which GPUs to use:

3. On autopicking tab:
`Write FOM maps?`
`Yes`

`Use GPU acceleration?`
`No`



Number of MPI procs: 3
Submit to queue? Yes
Queue name: gpu
Queue submit command: sbatch
Maximum run time (d-h:m:s) 3-00:00:00
GPU count (0-4) 0
Standard submission script: /net/prog/slurm/relion.sh
Minimum dedicated cores per node: 1
Additional arguments:

4. On Running tab:
`Number of MPI:`
`5` (or number of micrographs, whichever is fewer)

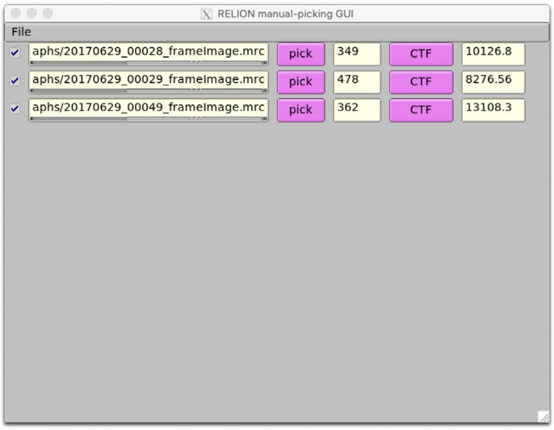
`Submit to queue?`
`Yes`

Click *Run!* ONCE

LoG picking

If you turn on `Write FOM maps`, Relion will write-out maps containing the Figure of Merit for each pixel (as generation of these maps is the most computationally expensive step, this is helpful for optimization of picking parameters.) If you then `Continue!` the job by reading the FOM maps, Relion with **replace** the particles in the previous round of picking with those chosen with the new parameters. Otherwise `Continue!` will ignore already picked micrographs and only pick new micrographs with the updated parameters.

We can now optimize our picking parameters.

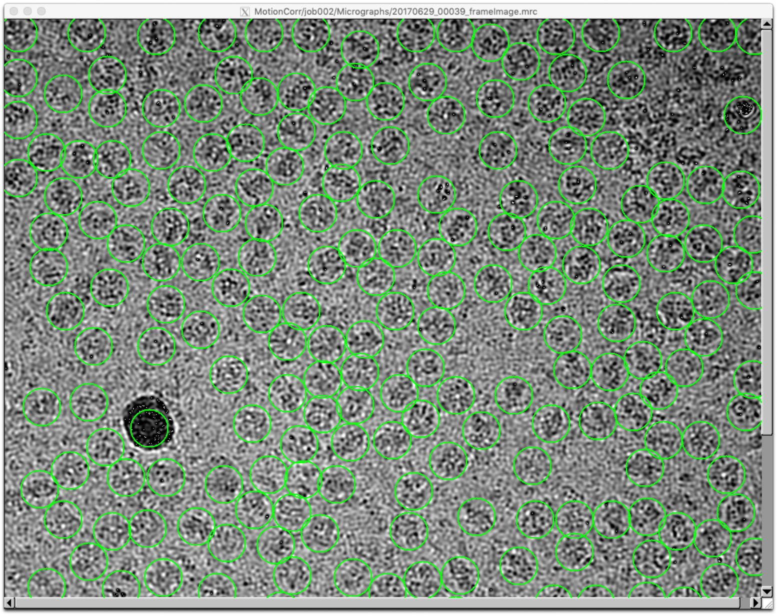


☑	aphs/20170629_00028_frameImage.mrc	pick	349	CTF	10126.8
☑	aphs/20170629_00029_frameImage.mrc	pick	478	CTF	8276.56
☑	aphs/20170629_00049_frameImage.mrc	pick	362	CTF	13108.3

Click on the finished job

1. Display the results in the same was as for the 2D classification, choosing `out: cords_suffix_autopick.star`
2. Display all micrographs by clicking on `pick`

Note: Relion takes the display parameters from the last `Manual picking` job. If you need to change something, select `Manual picking`, change the parameters you want, then choose the `Jobs` menu at the top and click `Save job Settings`

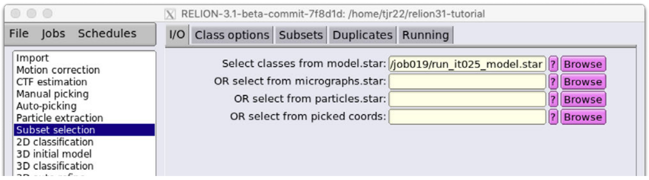


- Observations
    - Most (but not all,) of the particles have been picked
    - Some contamination picked
- Interpretation
    - Good enough for initial LoG pick. If not, see the section on reference-based picking to adjust the `Default threshold`.
- Next step
    - Extract particles and make 2D classes for references

    Please do this now, using a 320 Å box size (you need to convert to pixels) and 64 pixel re-scaled size.

Subset selection (Classes)

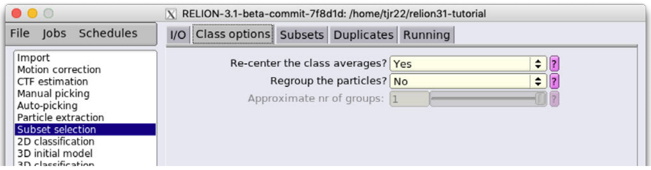
We now need to select classes to be our references for auto-picking our full dataset. Select the fewest classes (3~5) that have the most varied viewpoints. In the next step, they will be down-sampled to 20 Å, so resolution isn't important, shape is.



Select classes from model.star: /job019/run_it025_model.star
OR select from micrographs.star:
OR select from particles.star:
OR select from picked coords:

1. On I/O tab:

`Select classes from model.star:`
`Class2D/job###/run_it025_model.star`
(Make sure all the other fields are empty.)

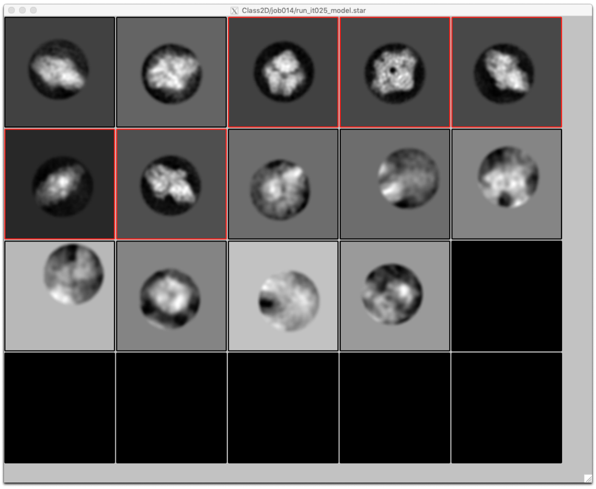


Re-center the class averages? Yes
Regroup the particles? No
Approximate nr of groups: 1

2. On Class options tab:
`Re-center the class averages?`
`Yes`

Click ***`Run!`***

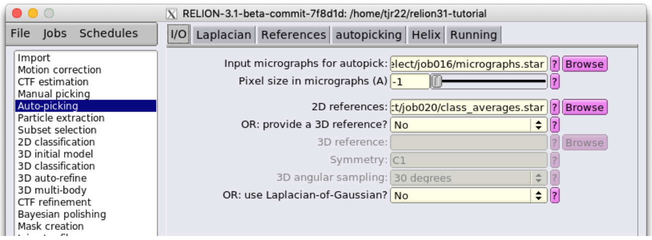
Display as for 2D classification



1. Select 3-5 classes:
    - Must look like protein projections
    - Try to get the most varied views
    - Don't just use the highest ranked
2. Right-click and `Save selected classes`
3. Close all display windows

Reference-based picking

The *de facto* standard way to pick your full dataset is via reference-based template matching. Now that we have good references (which are not always so easy to get,) we can use Relion's reference-based auto-picker and our 3-5 micrographs to tune the picking parameters.

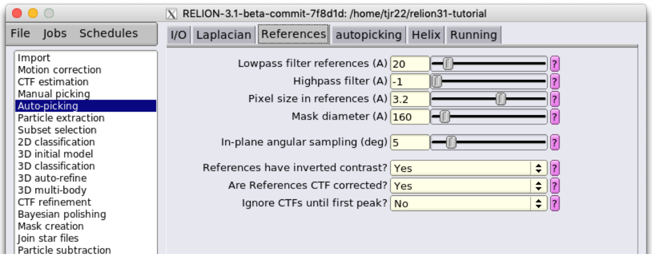


Input micrographs for autopick: elect/job016/micrographs.star
Pixel size in micrographs (A) -1
2D references: t/job020/class_averages.star
OR: provide a 3D reference? No
3D reference:
Symmetry: C1
3D angular sampling: 30 degrees
OR: use Laplacian-of-Gaussian? No

1. On I/O tab:
`Input micrographs for autopick:`
`select/job###/micrographs.star`

`2D References:`
`Select/job###/class_averages.star`

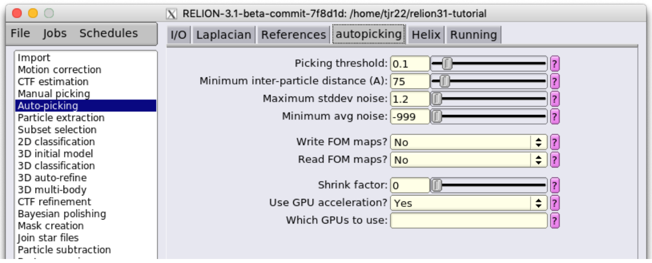
`OR use Laplacian-of Gaussians?`
`No`



Lowpass filter references (A) 20
Highpass filter (A) -1
Pixel size in references (A) 3.2
Mask diameter (A) 160
In-plane angular sampling (deg) 5
References have inverted contrast? Yes
Are References CTF corrected? Yes
Ignore CTFs until first peak? No

2. On References tab:
`Pixel size in references (A):`
(From extract job `run.out` file)

`Mask Diameter (A):`
`160`



Picking threshold: 0.1
Minimum inter-particle distance (A): 75
Maximum stddev noise: 1.2
Minimum avg noise: -999
Write FOM maps? No
Read FOM maps? No
Shrink factor: 0
Use GPU acceleration? Yes
Which GPUs to use:

3. On autopicking tab:
`Picking threshold`
`0.1`

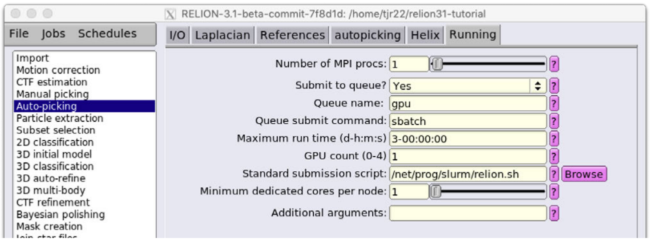
`Minimum inter-particle distance (A):`
(half the estimated particle size)

`Maximum stddev noise:`
`-1`

`Write FOM maps?`
`Yes`

`Shrink factor:`
`0`

`Use GPU acceleration?`
`Yes`



Number of MPI procs: 1
Submit to queue? Yes
Queue name: gpu
Queue submit command: sbatch
Maximum run time (d-h:m:s) 3-00:00:00
GPU count (0-4) 1
Standard submission script: /net/prog/slurm/relion.sh
Minimum dedicated cores per node: 1
Additional arguments:

4. On Running tab:
`Number of MPI:`
`1` (1 MPI/GPU)

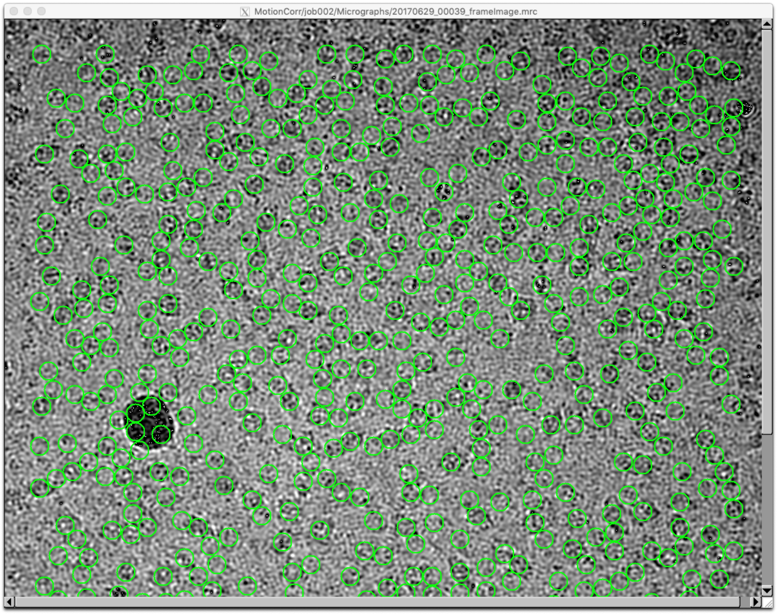
`Submit to queue?`
`Yes`

`GPU count (0-4):`
`1`

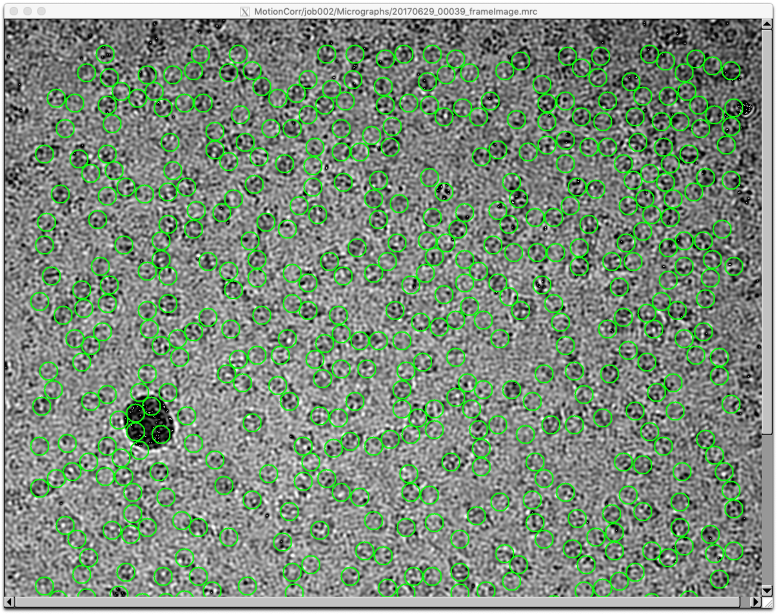
Click *Run!* ONCE

Reference-based picking

Click on the finished job, and display the results in the same was as for the LoG pick.



- Observations
  - All particles (except those on the edge) picked
  - Ice contamination picked
  - Some noise picked
- Interpretation
  - Good starting point for optimization
- Next steps
  1. Turn **off** `Write FOM maps` and turn **on** `Read FOM Maps`
  2. Turn **off** GPU acceleration and queue submission
  3. Increase Picking threshold
  4. Click `Continue!`
  5. Right-click on micrographs and reload coordinates
  6. Repeat steps 3-5 until all particles picked and minimal noise picked



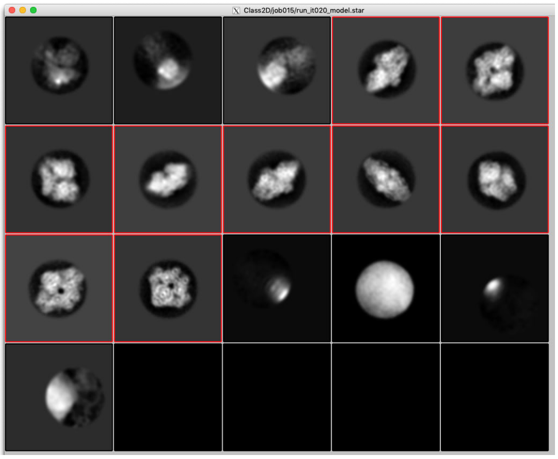
- Observations
  - All particles (except those on the edge) picked
  - Ice contamination picked
  - Minimal noise picked
- Interpretation
  - Need to optimize background filters
- Next steps
  1. Repeat 3-6 above, varying `Maximum stddev noise` (start at 1.0)
  2. Repeat 3-6 above varying `Manimum average noise` (start at -0.5)

Once you have optimized picking parameters, click on the Auto-picking job in the available jobs box to create a new job with the same parameters. Change the input micrographs to your `all_good_micrographs` job, turn off `Read FOM Maps`, turn on GPU acceleration, and pick your full dataset (if you get fewer than 10k particles, adjust your settings to increase the number of picks.) For real data, increase the number of MPI procs and GPUs to 2-4.

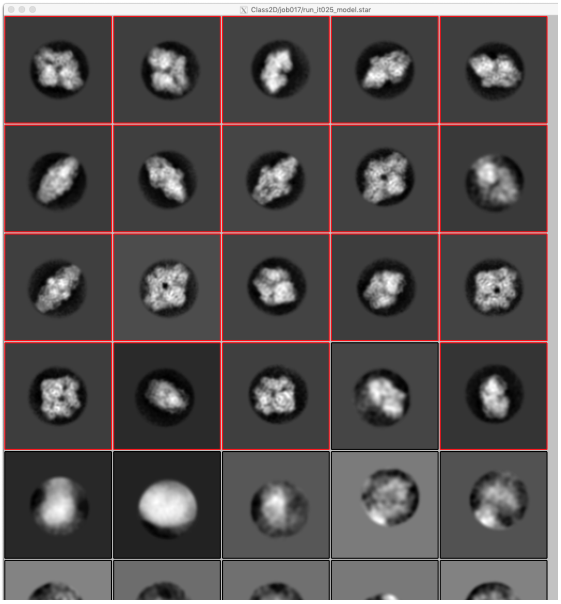
Particle set filtration via 2D classification

Now that we have finally done a full pick of our data, we can use 2D classification to filter out junk particles. Note that at this stage, our philosophy should be <u>to remove junk particles, NOT select good classes</u>. (note the numbers below are for example – yours may differ.)

15267 particles



7168 particles



6064 particles

1. `Extract` your particles with the same box size as before, and 64 pixel re-scalled size (~30s for 3 MPI processes).
2. Run `2D classification` with 50 classes, 20 iterations and `Use fast subsets` (~10 mins).
3. Do `Subset selection`, turning on re-grouping particles in the `Classes` tab (50 groups.) Select anything that doesn't look awful.
4. Run `2D classification` on the particles from step 3 with 200 classes, 25 iterations. (~20 mins). With a "real" dataset, you only use fast-subsets when you have more than ~50k particles, so turn it off at this stage.
5. Re-select as above.
6. Repeat `2D` & `Subset selection` loop until no-more junk is evident. (5~20 mins per job). Here, one more iteration should be enough. In a full dataset you may need several more cycles.

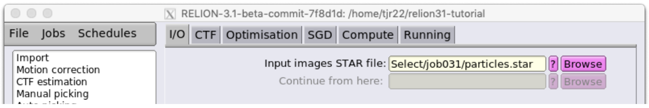
Note that you can select a class, right-click and `show metadata` to find out how many particles are in each class. Classes with fewer than 100-200 particles will almost always look bad (fuzzy, blurred, low-resolution, etc,...) but may still be worth keeping.

If, at this stage, your classes are collapsing from hundreds to a dozen or so, try turning on `Ignore CTFs until first peak?` on the CTF tab. The output classes will look odd, but should separate junk better (this should be considered a "last resort," though.)

Initial model generation

In order to do 3D classification, we require an initial 3D model. Although there are many ways to do this, we'll only see one here: *ab-initio* model generation using *stochastic gradient descent*.

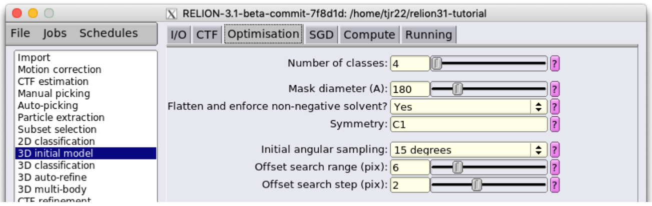
Note: you may be tempted to generate your initial model from (possibly hypothetical) PDB/atomic coordinate data. This is, by far, the easiest way to impose model bias on your data – and is strongly discouraged. We do not provide instructions on how to do this.



Input images STAR file: Select/job031/particles.star

Continue from here:

1. On I/O tab:
`Input images STAR file:`
`Select/job###/particles.star`



Number of classes: 4

Mask diameter (A): 180

Flatten and enforce non-negative solvent? Yes

Symmetry: C1

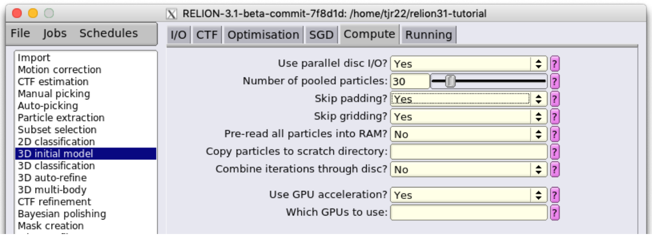
Initial angular sampling: 15 degrees

Offset search range (pix): 6

Offset search step (pix): 2

2. On Optimization tab:
`Number of classes:`
`4`

`Mask diameter:`
`180` (A bit bigger than the mask for the 2D)



Use parallel disc I/O? Yes

Number of pooled particles: 30

Skip padding? Yes

Skip gridding? Yes

Pre-read all particles into RAM? No

Copy particles to scratch directory:

Combine iterations through disc? No

Use GPU acceleration? Yes

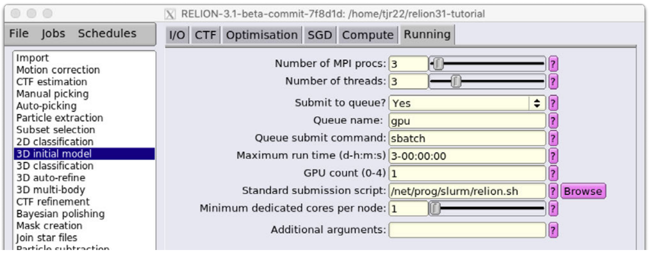
Which GPUs to use:

3. On Compute tab:
`Use parallel disc I/O?:`
`Yes`

`Number of pooled particles:`
`30`
`Skip padding?`
`Yes`

`Skip gridding?`
`Yes`

`Use GPU acceleration?`
`Yes`



Number of MPI procs: 3

Number of threads: 3

Submit to queue? Yes

Queue name: gpu

Queue submit command: sbatch

Maximum run time (d-h:m:s) 3-00:00:00

GPU count (0-4) 1

Standard submission script: /net/prog/slurm/relion.sh

Minimum dedicated cores per node: 1

Additional arguments:

4. On Running tab:
`Number of MPI procs:`
`3` (see comment below)

`Number of threads:`
`2` (see comment below)

`Submit to queue?`
`Yes`

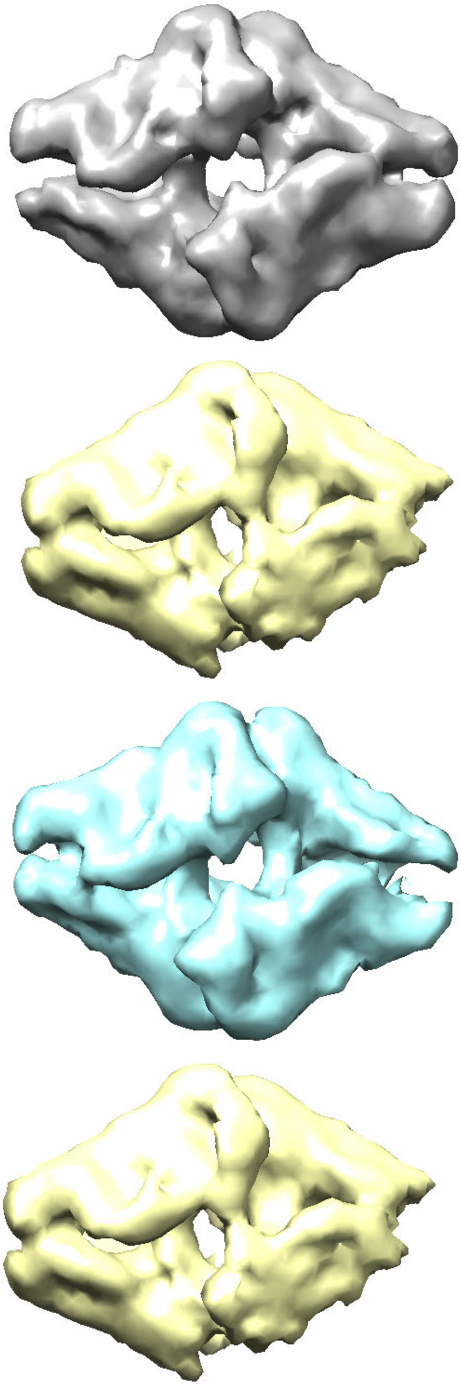
`GPU count (0-4):`
`1`

Click *Run!* ONCE. This should take 20~60 mins.

3D processing (classification and refinement) requires more GPU memory than 2D processing, so we can use fewer MPI processes per GPU. For classification jobs, we can generally use 2 MPI/GPU (plus one master process.) Just like in 2D, though, if you encounter an `out of memory` error, you'll need to reduce the number of MPI per GPU. Increase the number of threads to keep (MPI × Threads) ≤ (10 × GPUs). Always keep MPI × Threads ≤ 32.

Initial model generation

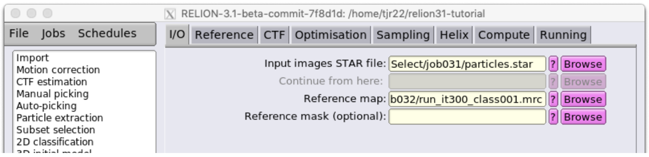
We will now assess the 4 initial models we generated using UCSF Chimera. Choosing the correct starting model is both important, and something of an art. In general, look for models that look like they may actually contain your 2D classes. They should also ‘look’ like protein, and have most of their density in the middle of the box, rather than as a shell on the outside of the box.



1. Download all four initial models from your processing directory to your local machine, and open UCSF Chimera
2. Once Chimera loads, open the 4 models by `File > Open`, then browse to wherever you saved them, select all 4 `run_it300_class00#.mrc` files, then click `open`.
3. On the volume viewer window, do `Features` > `Data Display Options`, then change the `Maximum number of histograms shown` to `4`
4. On the volume viewer window toolbar, click on `data`, then click on the dataset missing from the histograms
5. On the UCSF Chimera window toolbar, click `Tools` > `Structure Comparison` > `Tile structures`. Then click `OK` on the `Tile Structures` window
6. On the UCSF Chimera window toolbar, click `Favorites` > `Command line`. In the Command line that appears at the bottom of the window, type in `set independent` and press enter.
7. Of the four Initial maps shown here, all four are reasonable, but the grey and light blue both look much better. are suitable and can be used. The grey map will be used going forward.

3D classification

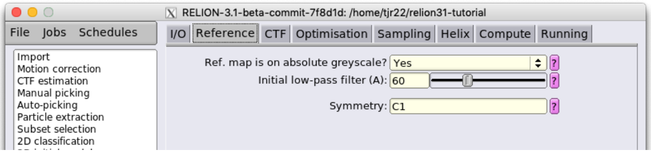
Now that we have an initial model, we can finally start choosing ‘good’ particles, rather than eliminating junk. We do this via 3D classification.



Input images STAR file: Select/job031/particles.star
Continue from here:
Reference map: b032/run_it300_class001.mrc
Reference mask (optional):

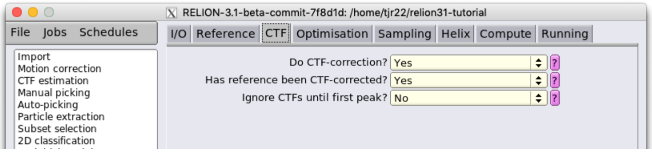
1. On I/O tab:
`Input images STAR file:`
(last select job)

`Reference Map:`
(your favorite 3D initial model)



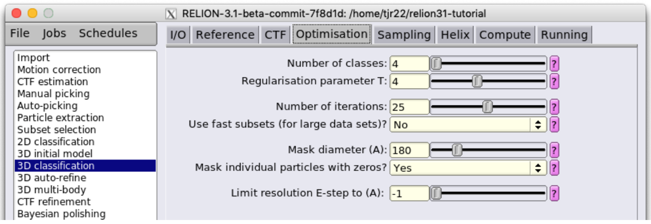
Ref. map is on absolute greyscale? Yes
Initial low-pass filter (A): 60
Symmetry: C1

2. On Reference tab:
`Ref. map is on absolute greyscale?`
`Yes` (only for maps directly from Relion refinements, `No` for resized maps, etc,… If you’re not sure, choose `No`)



Do CTF-correction? Yes
Has reference been CTF-corrected? Yes
Ignore CTFs until first peak? No

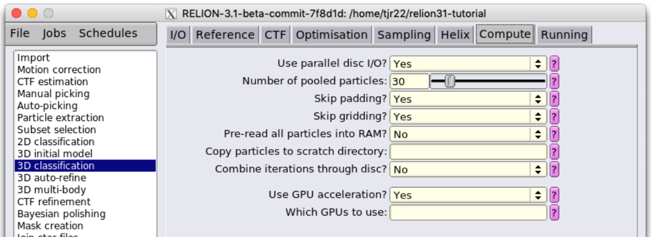
3. On CTF tab:
`Has reference been CTF corrected?`
`1. Yes`



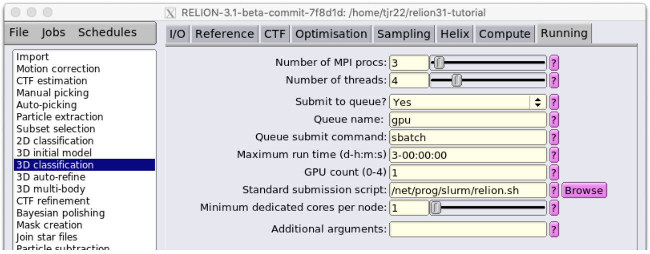
Number of classes: 4
Regularisation parameter T: 4
Number of iterations: 25
Use fast subsets (for large data sets)? No
Mask diameter (A): 180
Mask individual particles with zeros? Yes
Limit resolution E-step to (A): -1

4. On Optimization tab:
`Number of Classes:`
`4`

`Mask Diameter:`
`180`



Use parallel disc I/O? Yes
Number of pooled particles: 30
Skip padding? Yes
Skip gridding? Yes
Pre-read all particles into RAM? No
Copy particles to scratch directory:
Combine iterations through disc? No
Use GPU acceleration? Yes
Which GPUs to use:



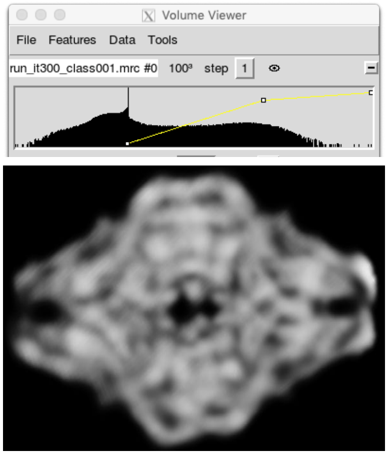
Number of MPI procs: 3
Number of threads: 4
Submit to queue? Yes
Queue name: gpu
Queue submit command: sbatch
Maximum run time (d-h:m:s) 3-00:00:00
GPU count (0-4) 1
Standard submission script: /net/prog/slurm/relion.sh
Minimum dedicated cores per node: 1
Additional arguments:

5. Compute and Running tabs:
Same as for `3D initial model`

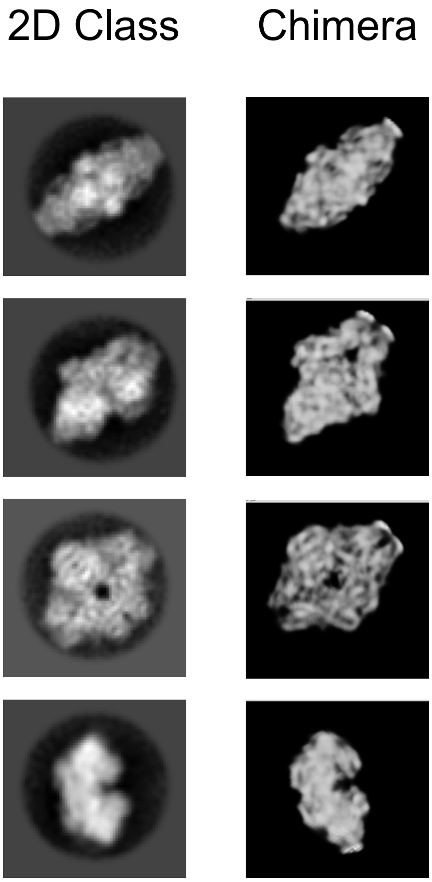
Click *Run!* ONCE
This job should take 15~20 mins.

Interlude: 2D classes vs 3D model

If you’re not sure if your model may match the 2D classes, you can change the view in chimera and try to find the right perspective. Here’s how class001 matches up



1. In Chimera, select File > open and select the initial model.
2. On the Chimera window select Tools > Viewing Controls > Camera and change projection to `orthographic`.
3. Use the scroll wheel to zoom-out so you can see the entire map.
4. On the Volume Viewer, change the Style from ‘surface’ to ‘solid’
5. Move the middle control point on the histogram to match the figure (left).



6. On the Relion window, go to your last 2D classification job, and `Display:` `out: run_it025_model.star`
7. Set the `Scale:` to `2`, `Sort images on:` `rnlClassDistribution`, and `Reverse sort`.
8. Click `Display!`
9. Now you can use the left mouse button in Chimera to rotate the model around, comparing it to your 2D Classes from Relion

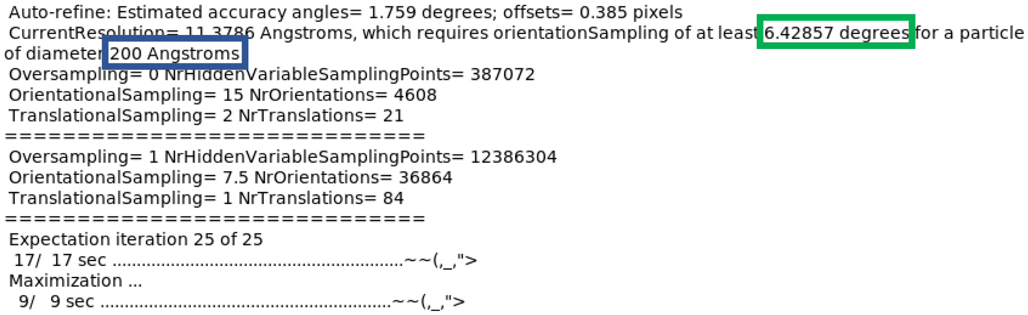
3D classification

To monitor our 3D classification, we look in three different places

1. Relion's output window:

Double-clicking on the window brings up the contents of the run.out file (no live-refresh, though)

Here we can see how the run time is progressing, the highest resolution map, the best angular accuracy achieved, and best offset accuracy achieved. Here's the output for the end of the run:



```
 Auto-refine: Estimated accuracy angles= 1.759 degrees; offsets= 0.385 pixels
 CurrentResolution= 11.3786 Angstroms, which requires orientationSampling of at least 6.42857 degrees for a particle
of diameter 200 Angstroms
 Oversampling= 0 NrHiddenVariableSamplingPoints= 387072
 OrientationalSampling= 15 NrOrientations= 4608
 TranslationalSampling= 2 NrTranslations= 21
==============================
 Oversampling= 1 NrHiddenVariableSamplingPoints= 12386304
 OrientationalSampling= 7.5 NrOrientations= 36864
 TranslationalSampling= 1 NrTranslations= 84
==============================
 Expectation iteration 25 of 25
  17/  17 sec ............................................................~~(,_,">
 Maximization ...
   9/   9 sec ............................................................~~(,_,">
```

Note that we've reached an alignment resolution (not model resolution) of 11.4 A. If we want higher resolution, we either need to use a smaller mask diameter (blue box) or finer orientational sampling (green box) – and hope our data support it.

2. Per-iteration model.star file

On the command line, typing

`less Class3D/jobXXX/run_it025_model.star` will show the details of iteration 25.

Here is the most relevant section of that file:

```
data_model_classes

loop_
_rlnReferenceImage #1
_rlnClassDistribution #2
_rlnAccuracyRotations #3
_rlnAccuracyTranslations #4
_rlnEstimatedResolution #5
_rlnOverallFourierCompleteness #6
Class3D/job029/run_it025_class001.mrc     0.925191     1.759000     0.385000     11.378571     0.999540
Class3D/job029/run_it025_class002.mrc     0.010207     5.000000     1.246000     35.400000     0.826843
Class3D/job029/run_it025_class003.mrc     0.017992     3.595000     0.853000     26.550000     0.938008
Class3D/job029/run_it025_class004.mrc     0.046610     3.640000     0.890000     24.507692     0.988701
```

What we can see is that a single class has 92.5% of the particles, with an accuracy of 1.759 degrees and 0.358 pixels. The other classes are mostly empty, and any particles in those classes are most likely junk.

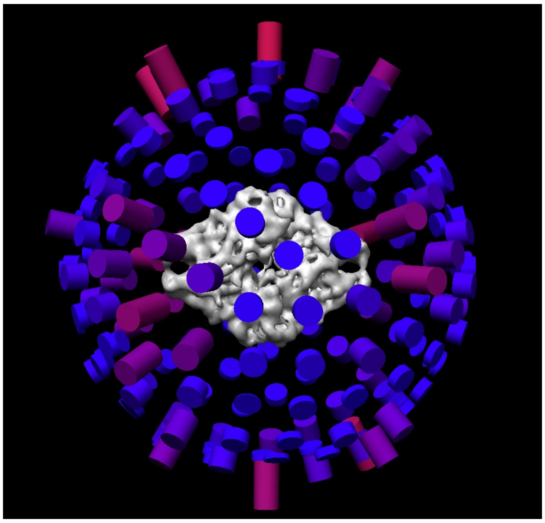
3. We can also view the classes in UCSF Chimera (*you should do that now*) Along with information from the model.star file, you should be able to choose which class(es) to progress to refinement.

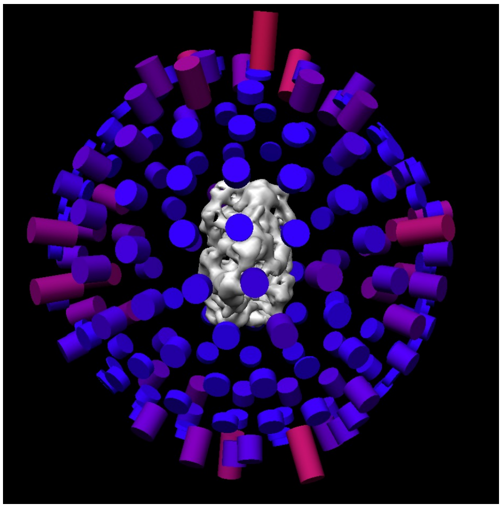
Note: This dataset typically doesn't require much 3D classification, but several rounds of Class3D/Select jobs are typically required.

Detecting preferential orientation

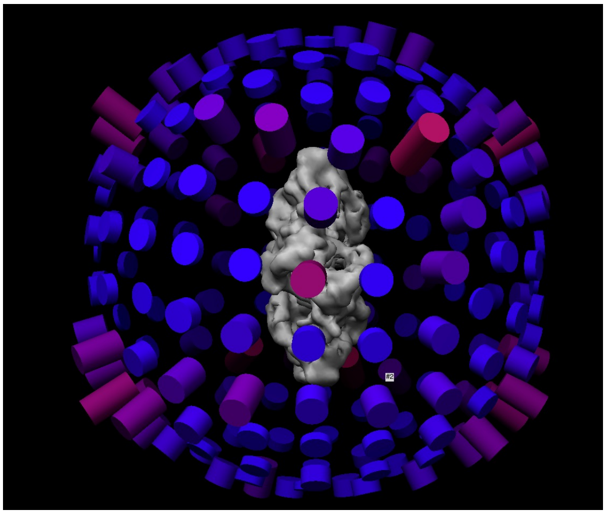
Preferential orientation, where the particles orient themselves on the grid, leads to anisotropic (not the same in all orientations,) resolution, and can make analysis of your map very difficult. Here's how to detect it.

1. Load a single model in chimera (here, we load class001.)
2. Using the File > load on the main Chimera window, open up the corresponding `XXX_angdist.bild` file





The `.bild` files get rendered as columns around our model. The size and color of the column have the same meaning: the number of particles in that view. As you can see from this, there isn't a completely uniform distribution of views, however there are views from a number of different perspectives. This gives us confidence that there are enough views present and our data do not show preferential orientation.

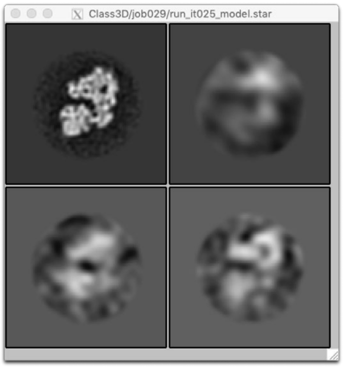


Subsets in 3D

Selecting particles is slightly different between 2D and 3D.

1. We don’t want to sort the classes (we choose which one(s) we want by inspecting the classes in Chimera.)
2. We need to ensure that we don’t have any duplicate particles, which will interfere with ‘gold-standard’ resolution estimates later on

6064 particles



5504 particles

1. Perform subset selection as you would for 2D
Ensure that you don’t sort the classes. Your results may look different – include all good classes.

2. Remove duplicates via another Subset selection job.
On the I/O tab, choose the previous Select job in `OR select from particles.star`
On the Class options tab, turn off `re-group particles`
On the duplicates, turn on `OR: remove duplicates?`

Click `Run!`

Note: depending on the picking, it’s possible that no particles are removed.

Re-extraction

Thus-far, the particles in this tutorial were extracted at 4.98 Å /pix, which gives a Nyquist resolution of 9.96 Å. Qualitatively, the 3D classifications appear to be giving maps at ~10Å, so we now want to reduce our Å/pix to continue. We'll start with a modest increase in resolution to 3.54Å/pix.

Classification and refinement jobs also improve the center position of our particles substantially. As a result, we no-longer choose our box size based on imprecise picking positions. Instead, we now choose the box size to capture the delocalized information due to the CTF of our microscope. Rosenthal and Henderson (2003) show what **maximum** useful size box we should use:

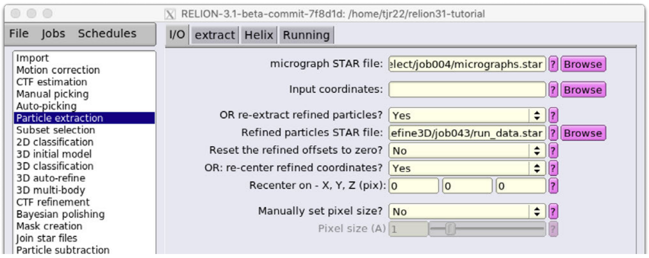
$$BoxSize\ in\ \mathring{A} = \frac{Maxmimum\ Defocus\ in\ \mathring{A}}{Constant \times Target\ Resolution\ in\ \mathring{A}} + Particle\ Size\ in\ \mathring{A}$$

Rado Danev's guidelines for the value of *Constant* (25 for 300 keV, 20 for 200 keV, and 12.5 for 100 keV) are convenient here. (Get the defocus range by displaying the log file from the `CTF estimation` job):

$$BoxSize\ in\ \mathring{A} = \frac{13\,000\mathring{A}}{20 \times 7.08\mathring{A}} + 160\mathring{A}$$

$$BoxSize\ in\ \mathring{A} \approx 252\mathring{A}$$

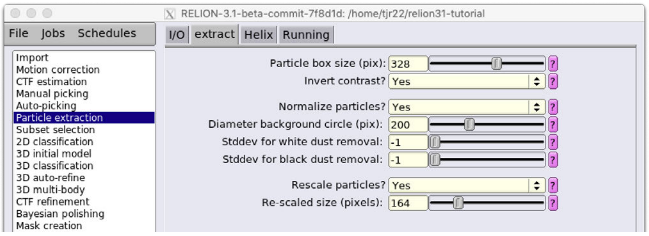
It's worth noting that <u>larger boxes introduce more noise and higher defocus images have lower signal-to-noise at higher resolution</u>, so this number should be considered an absolute maximum, and in our experience box sizes lower than this maximum will perform better once you start getting to Nyquist values below ~5Å. The minimum recommended box size is about 1.5 × particle size; we'll use a box size of 240Å.



micrograph STAR file: elect/job004/micrographs.star
Input coordinates:
OR re-extract refined particles? Yes
Refined particles STAR file: efine3D/job043/run_data.star
Reset the refined offsets to zero? No
OR: re-center refined coordinates? Yes
Recenter on - X, Y, Z (pix): 0 0 0
Manually set pixel size? No
Pixel size (A) 1

1. On I/O tab:

`Micrograph STAR file:`
(Same as before)
`Input coordinates:`
(Empty)
`OR re-extract refined particles?`
`Yes`
`Refined particles STAR file:`
(Select `particles.star` file)
`OR: re-center refined coordinates?`
`Yes`



Particle box size (pix): 328
Invert contrast? Yes
Normalize particles? Yes
Diameter background circle (pix): 200
Stddev for white dust removal: -1
Stddev for black dust removal: -1
Rescale particles? Yes
Re-scaled size (pixels): 164

2. On Extract tab:
`Particle box size (pix):`
`272`
`Particle box size:`
`68`

New extract parameters require new maps

We need maps to match our particle sizes. Relion 3.1 does allow us to mix box sizes and resolutions across datasets and initial models. However, it will use the box size and resolution of the first image/model is encounters, which for 3D classification or refinement is the initial model.

There are two ways to get higher resolution maps. The first way is to convert our current map using using `relion_image_handler`.

1. In the terminal window, find the Class3D output file:

```
ls Class3D/job###/run_class001.mrc
```

2. Now call `relion_image_handler`, giving it the input (`--i`) and output (`--o`) file names, as well as the current (`--angpix`) and new resolution (`--rescale_angpix`) and new box size (`--new_box`) from the `Extract` job (note that the slashes at the ends of the line are for convenience. You can leave them out if your screen is wide enough). Here, I use an old pixel size of 3.186Å, a new pixel size of 3.54Å, and a new box size of 68 pixels. Make sure to use your own values here!

```
relion_image_handler \
--i Refine3D/jobXXX/run_class001.mrc \
--o Refine3D/jobXXX/run_class001_68box_3.54angpix.mrc \
--angpix 3.186 --rescale_angpix 3.54 --new_box 68
```

If you examine the particles `datablock` in the `particles.star` file from the extract job, you'll notice it has fields for `rnlAnglePsi`, `rnlAngleRot`, `rnlAngleTilt`, `rnlOriginXAngst` and `rnlOriginYAngst`, as well as `rlnDefocusU`, `rnlDefocusV` and `rnlDefocusAngle` – the three alignment angles, two shifts, and three defocus values required to do a reconstruction. So the other way to generate a new map is to do a reconstruction directly from the particles. We do this using `relion_reconstruct` (making sure to include the `--ctf` flag to apply ctf correction.)

```
relion_reconstruct --ctf \
--i Extract/job027/particles.star \
--o Extract/job027/map.mrc
```

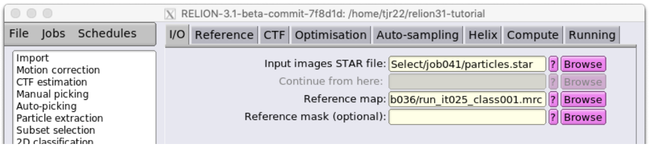
As a matter of routine, you should re-do the last classification job you did before re-extraction. Often – the higher resolution will allow Relion to separate more junk. Please repeat the Class3D job using the particles and map from this extract job (when choosing the 3D map, you may have to go up two levels by clicking on the `../` twice to get to the actual Extract job), then do a Subset selection job before moving on.

3D auto-refinement

This is the first time we get to see how much resolution we may be able to get out of our data. The auto-refinement algorithm in Relion will progressively increase the precision of the alignment while avoiding overfitting. Unlike classification, refinement doesn't have a fixed number of iterations – it proceeds until it converges.

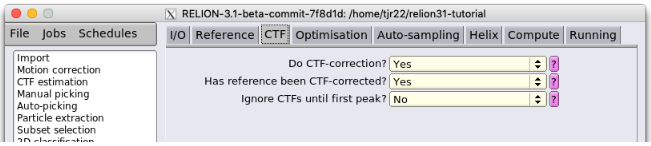
This is also the first time we're likely to run out of memory. If you see a `ERROR: CudaCustomAllocator out of memory` error, this means that the GPU has run out of RAM, and you have to change back to one MPI per GPU instead of the 2 MPI per GPU we've been using so-far.

Note: Our particles were extracted at 3.54 Å /pix, so our maximum resolution is 7.08 Å.



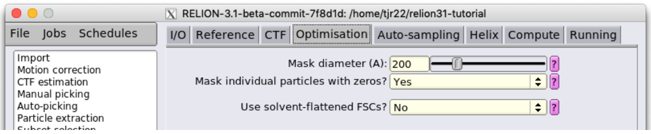
Input images STAR file: Select/job041/particles.star
Continue from here:
Reference map: b036/run_it025_class001.mrc
Reference mask (optional):

1. On I/O tab:
`Input imges STAR file:`
(3D select job)
`Reference Map:`
(your 3D class)



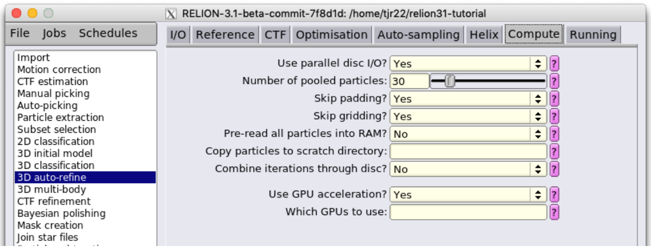
Do CTF-correction? Yes
Has reference been CTF-corrected? Yes
Ignore CTFs until first peak? No

2. On Reference tab:
`Ref. map is on absolute greyscale?`
`Yes` (only for maps directly from Relion refinements, `No` for resized maps, etc,…)



Mask diameter (A): 200
Mask individual particles with zeros? Yes
Use solvent-flattened FSCs? No

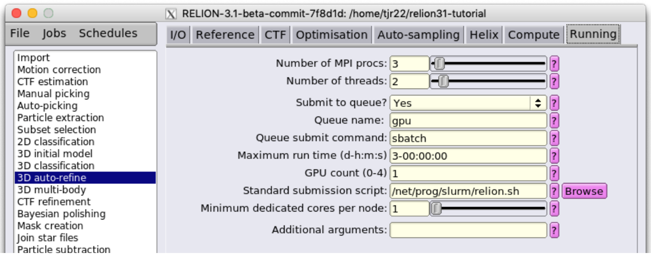
3. On CTF tab:
`Has reference been CTF corrected?`
`1. Yes`



Use parallel disc I/O? Yes
Number of pooled particles: 30
Skip padding? Yes
Skip gridding? Yes
Pre-read all particles into RAM? No
Copy particles to scratch directory:
Combine iterations through disc? No
Use GPU acceleration? Yes
Which GPUs to use:

4. On Optimization tab:
`Mask Diameter:`
`180`

5. On Compute tab: Same as Class3D



Number of MPI procs: 3
Number of threads: 2
Submit to queue? Yes
Queue name: gpu
Queue submit command: sbatch
Maximum run time (d-h:m:s) 3-00:00:00
GPU count (0-4) 1
Standard submission script: /net/prog/slurm/relion.sh
Minimum dedicated cores per node: 1
Additional arguments:

6. On Running tab: Same as Class3D
Note: The description of running parameters for 3D classification applies here as well.

Click *Run!* ONCE
This job should take 2-5 mins.

When the job finishes, it will report the unmasked resolution (i.e. the resolution of the entire volume. To find out how well we really did, we need to post-process our map (which means we need a mask.)

Mask generation

Generation of masks can be done many ways. The most common is via UCSF Chimera. Here, we generate a solvent mask from our auto-refined map to use in post-processing.

Mask generation proceeds in 4 steps.

1. The map is low-pass filtered to remove any high-resolution features
2. It is then binarised by setting all values below a threshold to zero, and all those above to one
3. The map is expanded by some number of pixels
4. A [0 - π/2] cosine-shaped density is added to soften the edge of the map

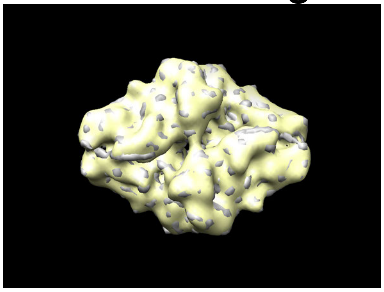
In order to properly determine the threshold for binarization, it is helpful to first low-pass filter your map before downloading it and loading in into UCSF Chimera:

```
relion_image_handler \
--i Refine3D/job###/run_class001.mrc \
--o Refine3D/job###/run_class001_15Alowpass.mrc \
--lowpass 15
```

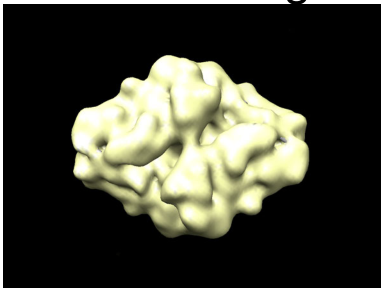
In order to properly determine the threshold for binarization, it is helpful to first low-pass filter your map before loading in into UCSF Chimera.

Now load `run_class001.mrc` and `run_class001_15Alowpass.mrc` into Chimera. Set the step sizes for both maps to 1. Use the unfiltered map to see your density and set the unfiltered map level as high as possible before seeing ‘dust’ (noise) come up.

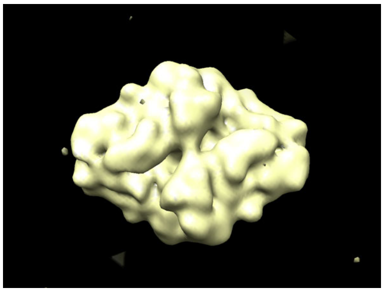
0.05: Too high

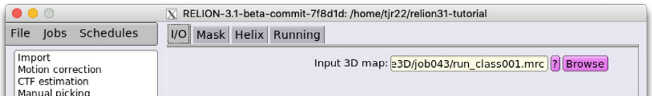


0.02: Just Right



0.015: Too low

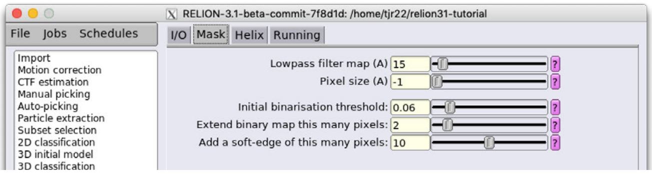




Input 3D map: e3D/job043/run_class001.mrc

1. On I/O tab:

`Input 3D map:`
`Refine3D/job###/run_class001.mrc`



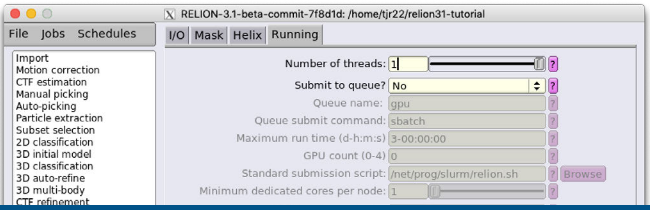
Lowpass filter map (A): 15
Pixel size (A): -1
Initial binarisation threshold: 0.06
Extend binary map this many pixels: 2
Add a soft-edge of this many pixels: 10

2. On Mask tab:

`Initial binarization threshold:`
(Just Right value)

`Extend binary map this many pixels:`
`2`

`Add a soft edge of this many pixels:`
`5-10`



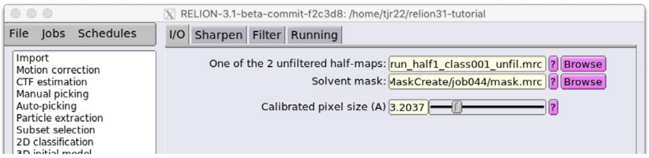
Number of threads: 1
Submit to queue? No
Queue name: gpu
Queue submit command: sbatch
Maximum run time (d-h:m:s) 3-00:00:00
GPU count (0-4) 0
Standard submission script: /net/prog/slurm/relion.sh
Minimum dedicated cores per node: 1

3. On Running tab:

`Number of threads:`
`1`
(Or you can submit to the cluster using up to 16 threads

Post-processing

Relion can auto-sharpen our map, and report the resolution within the solvent mask we just generated, using the Post-process job. Note that this only generates reliable sharpening if your resolution is significantly better than 10 Å (which, at 6.8 Å, ours is already.)

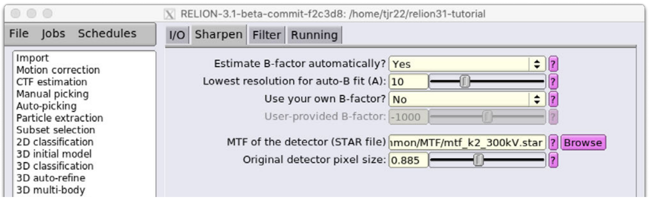


One of the 2 unfiltered half-maps: run_half1_class001_unfil.mrc
Solvent mask: MaskCreate/job044/mask.mrc
Calibrated pixel size (A) 3.2037

1. On I/O tab:

`One of the 2 unfiltered half-maps:`
`Refine3D/job###/run_half1_class001_unfil.mrc`

`Solvent mask:`
`MaskCreate/job###/mask.mrc`



Estimate B-factor automatically? Yes
Lowest resolution for auto-B fit (A): 10
Use your own B-factor? No
User-provided B-factor: -1000
MTF of the detector (STAR file) mon/MTF/mtf_k2_300kV.star
Original detector pixel size: 0.885

2. On Sharpen tab:

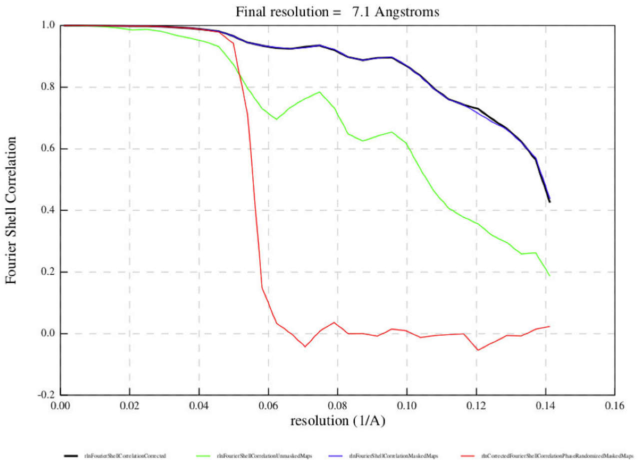
`MTF of the detector (STAR file)`
`/net/common/MTF/mtf_k2_300kV.star`

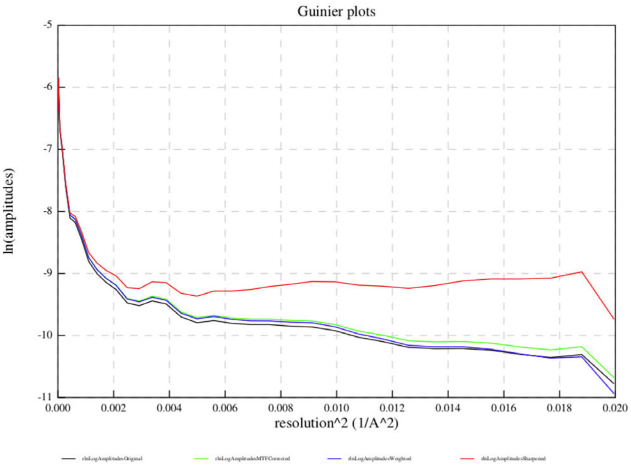
`Original detector pixel size:`
`0.885`

You can choose to submit to the queue or not.
Click *Run!* ONCE

The output of this job has lots of information. Most notably, the b-factor Relion applies for sharpening is -229, and the calculated final resolution is 7.08 Å, which is exactly Nyquist.

As with many other Relion jobs, Post-processing generates a PDF output file. In this case, various FSC (resolution) and Guinier (sharpening) curves are plotted. *Please* `Display` `out: logfile.pdf` *now.*





The most important line in the FSC plot is the black line, show the FSC for the masked map after correction for masking effects. Here, we can see that the FSC doesn't reach zero, indicating that our data will probably support higher resolutions than the current Nyquist resolution.

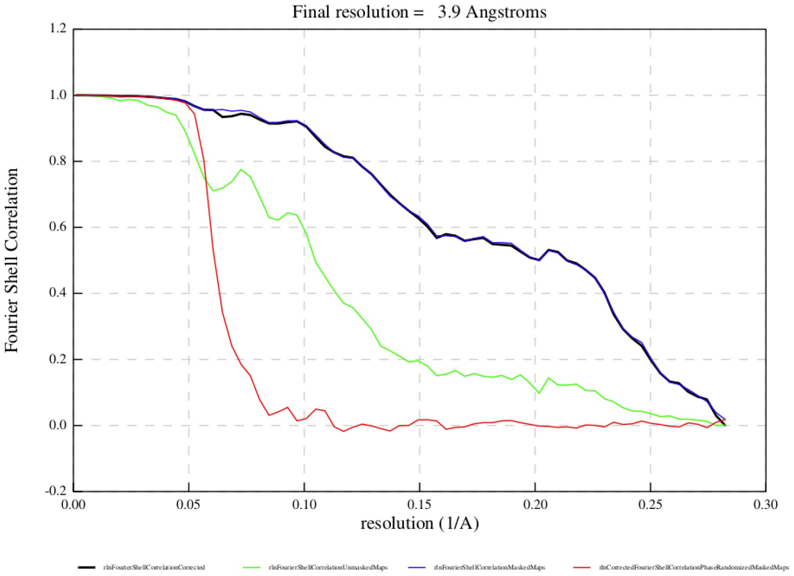
The most important line in the Guinier plot is the red line. Ideally we want it to remain flat at higher resolutions, before falling off at the highest resolution. Here, you can see how there's a small rise at the higher resolutions, indicating very slightly over sharpening (this degree of over-sharpening is not significant.)

Next steps

From the output of the latest 3D auto-refine and Post-process jobs, we can see that the map is at (or very close to) the Nyquist resolution, and the FSC didn't fall to zero. We should therefore be able to extract more resolution from our data.

1. Re-extract the particles from the `run_data.star` file from the Refine3D job. Remember that Relion requires even box sizes, so increase the `Particle box size` to 280 and the `Re-scaled size (pixels)` to 140 pixels.
2. Reconstruct a map from these particles.
3. Skip the Class3D job for this tutorial. Run a new Refine3D job with the newly extracted particles and map. (Note how much longer this job takes, even with only ~5k particles!)
4. Make a mask and Post-process the new map.

This time, the b-factor is -59 and the map went to 3.9Å. The log file shows that the FSC went to zero – so going to a finer Å/pix will not deliver more resolution (and may make it worse as more noise than signal will be introduced).



If you examine the penultimate iteration in the Refine3D job, you should notice that the estimated resolution before the two half-maps were combined was 8~9Å, which then dropped to 4~5Å once the maps were combined. This is strongly suggestive that signal-to-noise is currently limiting our resolution, rather than alignment quality.

So it seems clear that we need to either remove more bad particles (Using 2D or 3D classification, with or without `image alignment` (look in the `Sampling` tab); or we need to add more good particles.

In a real dataset, the next step would be to filter out more bad particles, but we'll skip that now. DO NOT skip this step on real data - filtering out bad particles is <u>much more powerful</u> than adding new good particles!

There are two ways of adding good particles: Collecting more data or use symmetry.

Using symmetry

The main problem with symmetry is that is assumes your particle is symmetric. Any non-symmetry in your particles will reduce the resolution of your map. Applying symmetry will also cause you to lose any biologically relevant asymmetry information. Although using symmetry often gets you a prettier map, it may also be the wrong thing to do. You have been warned!

There are two steps required to use symmetry (see note below before actually doing these steps):

1. Align your map to the appropriate symmetry axis (axes).
    1. In the terminal, call `relion_align_summetry`, giving it the input (`--i`) and output (`--o`) file names, as well as the symmetry operator (`--sym`), in this case, D2.

        ```
        relion_align_symmetry \
        --i Refine3D/job###/run_class001.mrc \
        --o Refine3D/job###/run_class001_D2-aligned.mrc \
        --sym D2
        ```

    2. Although strictly unnecessary, you can then apply the symmetry to your map using `relion_image_handler`:

        ```
        relion_image_handler \
        --i Refine3D/job###/run_class001_D2-aligned.mrc \
        --o Refine3D/job###/run_class001_D2-aligned_D2-applied.mrc \
        --sym D2
        ```

    It may be interesting to apply symmetry to you unaligned map as well. *Please do that now and examine the results in Chimera.*

2. Repeat your Refinements with the new symmetry-aligned starting map and forcing symmetry in 3D auto-refine (`Symmetry` in the Reference tab of the 3D auto-refine job)

The tutorial data would now go to Nyquist resolution (3.54Å.) Normally, you would apply symmetry, check the resolution of the auto-refinement, and then re-extract the particles and re-scale the maps. For expediency, please do the re-extraction, rescaling, symmetrical refinement and post-processing now (use a 280 pixel box and down-scale particles to 200 pixels.)

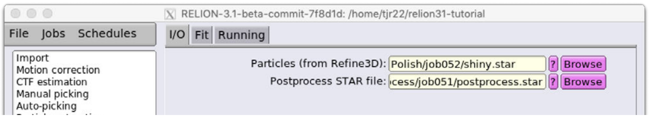
You should now have a map at 3.3~3.4Å resolution (well above the new Nyquist resolution of ~2.5Å. In a real dataset, now would be a good time to remove more bad particles. For this tutorial, we will proceed directly to refinement of the CTF parameters and per-particle motion correction. Because it's generally quicker, we start with a basic CTF refinement.

CTF refinement

CTF refinement works by comparing your particles to your map – so the better the map, the better CTF parameter estimates you can generate – which should get you a better map. As a result, you may need to do 2 or more cycles of CTF refinement + 3D auto-refine.

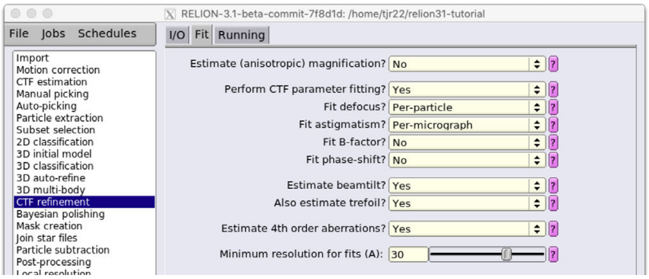
At low resolutions (worse that ~4.5 A,) only the defocus and large values of astigmatism have a significant impact on your map. Once you start to go to higher resolutions, other optical aberrations of the microscope can become limiting. Relion has the ability to estimate these additional aberrations using the CTF refinement job. Subsequent rounds of refinement will then take these estimates into account.

The first step is to refine the defocus estimates for all of our particles. In addition, we can estimate the beamtilt and trefoil (3rd order) aberrations. Other aberration estimations are possible, but we don't recommend using them at this resolution.



Particles (from Refine3D): Polish/job052/shiny.star
Postprocess STAR file: cess/job051/postprocess.star

1. On I/O tab:
`Particles (from Refine3D):`
(use the `run_data.star` file from your refine job)
`Postprocess STAR file:`
(matching post-process job from refine job)



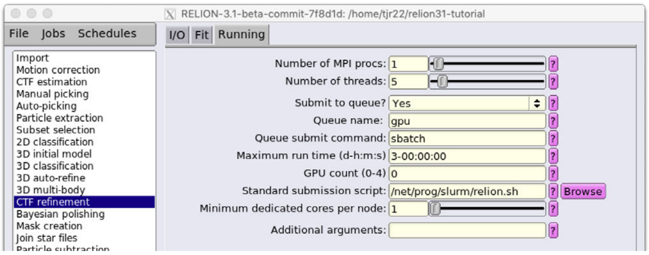
Estimate (anisotropic) magnification? No
Perform CTF parameter fitting? Yes
Fit defocus? Per-particle
Fit astigmatism? Per-micrograph
Fit B-factor? No
Fit phase-shift? No
Estimate beamtilt? Yes
Also estimate trefoil? Yes
Estimate 4th order aberrations? Yes
Minimum resolution for fits (A): 30

2. On Fit tab:
`Estimate (anisotropic magnification?`
`No`
`Perform CTF parameter fitting?`
`Yes`
Fit Defocus?
Per-particle
Fit astigmatism?
Per-micrograph
Estimate beamtilt?
Yes
Also estimate trefoil?
Yes

3. On Running tab:
`Number of MPI procs:`
`1` (4 for real data)

`Number of Threads:`
`5`

`Submit to queue?`
`Yes`



Number of MPI procs: 1
Number of threads: 5
Submit to queue? Yes
Queue name: gpu
Queue submit command: sbatch
Maximum run time (d-h:m:s) 3-00:00:00
GPU count (0-4) 0
Standard submission script: /net/prog/slurm/relion.sh
Minimum dedicated cores per node: 1
Additional arguments:

Click *`Run!`* ONCE. This should take 1~2 mins.

If your map is low-resolution, especially noisy, or your particles are especially small, fitting the beamtilt and trefoil may actually decrease the resolution of your reconstruction. Because polishing and CTF refinement are iterative, it is often best to fit just the defocus and astigmatism on the first iteration and only fit the others once the map no-longer improves. As phase-shift only makes sense for phase-plate data, and even then only plays a major role at low resolutions, so it's common not to fit the phase-shift at all.

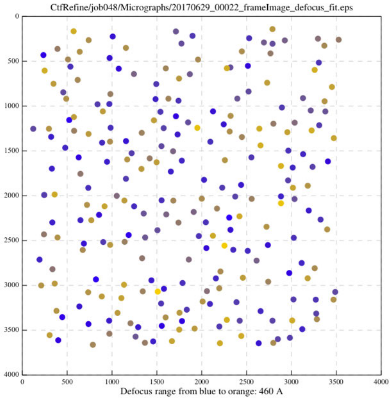
Now repeat the 3D auto-refine and post-process jobs to see if there's any improvement (make sure to use the CtfRefine job as the input particles!)

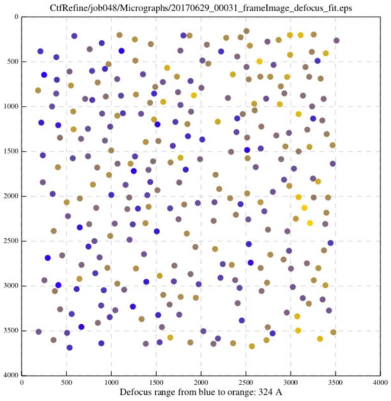
CTF refinement (output)

The logfile from the CTF refinement job contains a wealth of information.

Defocus refinement

There is an even distribution from Blue to Yellow in most micrographs, indicating that the particles are distributed across the thickness of the ice. Micrographs with particles localized to the ice/water interface show strongly blue and strongly yellow particles with nothing between.

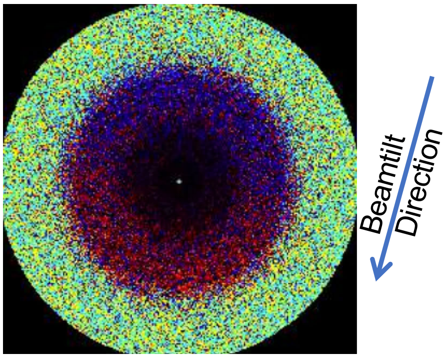




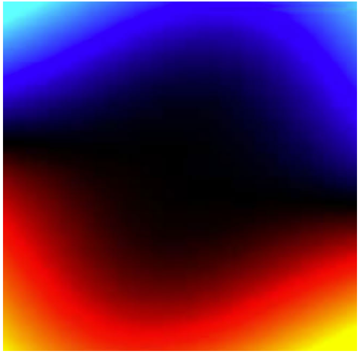
Several micrographs were tilted. Note that the particles on, for example micrograph 31, go from mostly blue to mostly gold

Asymmetrical Aberrations

Beam tilt is evident from the lower scale asymmetric plot (top left plot) as a blue-red transition across the plot. No trefoil aberrations are evident (they would be triangular blue-red shapes.)



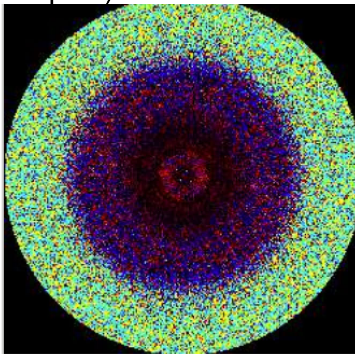
The fitted Zernike polynomials (below) used for correction seem to match the measured aberrations (above).



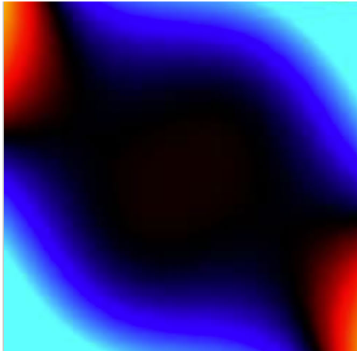
The fitted beam tilt is 0.03 x -1.17 mRad.
(found in the `particles_ctf_refine.star` file.)

Symmetrical Aberrations

A small Cs error is evident from the lower scale symmetric plot (top left plot) as a blue ring at higher resolutions (inner to outer.) No tetrafoil aberrations are evident (they would be square blue-red shapes.)

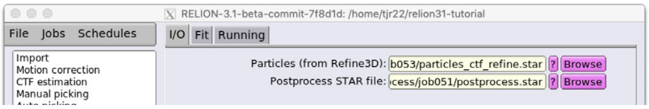


The fitted Zernike polynomials (below) used for correction seem to match the measured aberrations (above).



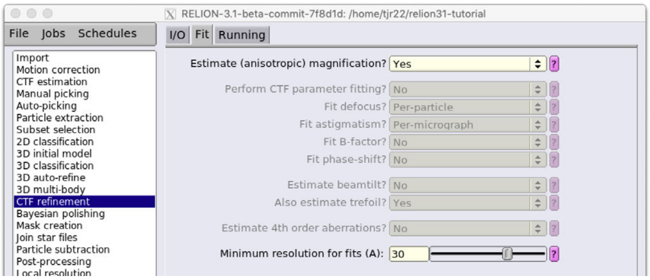
Anisotropic Magnification (CTF refinement)

We can also estimate the anisotropic magnification using the CTF refinement job. Combining the other estimation tasks with this one is not possible, so we simply take the output of the previous job and use it as input here. (We actually run this later in the tutorial.)



Particles (from Refine3D): b053/particles_ctf_refine.star
Postprocess STAR file: ıcess/job051/postprocess.star

1. On I/O tab:
`Particles (from Refine3D):`
(use the file from your previous ctf refinement job)
`Postprocess STAR file:`
(same as for previous polishing job)



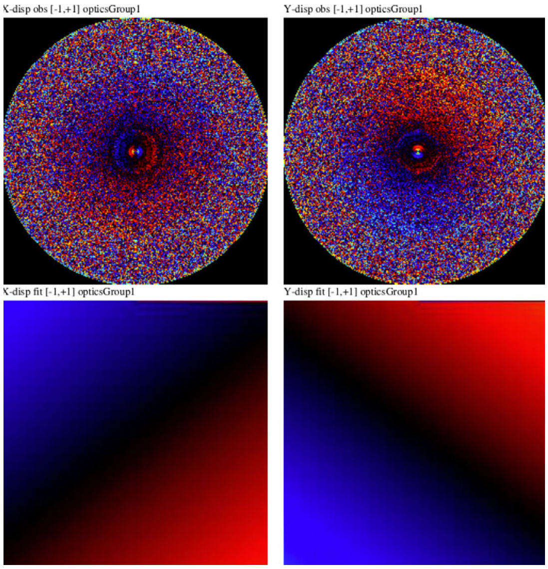
Estimate (anisotropic) magnification? Yes
Perform CTF parameter fitting? No
Fit defocus? Per-particle
Fit astigmatism? Per-micrograph
Fit B-factor? No
Fit phase-shift? No
Estimate beamtilt? No
Also estimate trefoil? Yes
Estimate 4th order aberrations? No
Minimum resolution for fits (A): 30

2. On Fit tab:
Turn everything off, then:

`Estimate (anisotropic magnification?`
`Yes`

Click `Run!` ONCE. This should take 1~2 mins.

Once the job finishes, examine the resulting logfile. Similar to the aberrations in the previous step, we have measured X and Y distortion in the top plots, and fitted corrections in the bottom plots.



Although it is clear from the logfile that the CTF refinement job can detect anisotropic magnification in the data, the output matrix looks (something) like this:

$$\begin{bmatrix} 1.001341 & 0.001665 \\ 0.001609 & 0.997878 \end{bmatrix}$$

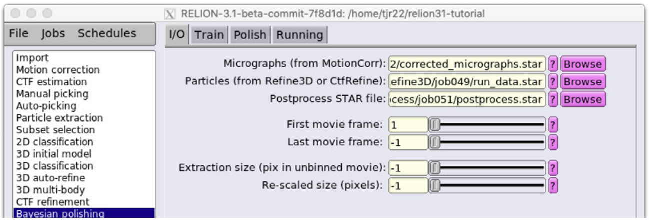
With anisotropic magnification on the top left and bottom right, and skew on the top right and bottom left. This indicates that there's only ~0.3% anisotropic magnification or skew, which is insignificant at this particle size.

Bayesian polishing

It is convenient to think of Bayesian polishing as a re-extraction directly from the original micrograph movies. Unlike standard motion correction, polishing fits map projection to each frame to estimate the movement (as well as the per-frame resolution.) So, just like CTF refinement, the better the map, the better the polishing – which should give you a better map.

While it requires you to tune several parameters for your particular dataset for **best** performance, you can do at least one round of polishing using the default parameters before you have to do the fitting (which takes a long time!)

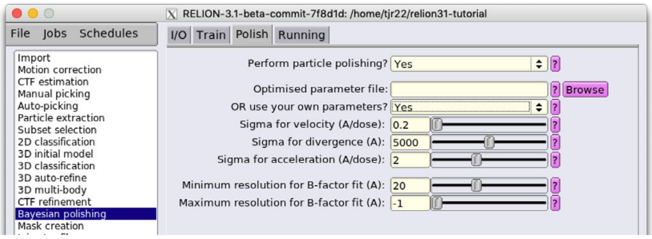
Determining precise extraction parameters is an art. For now, we'll aim for 3Å target resolution – suggesting we should use a 426 pixel box down-sampled to 252 pixels. But see the next slide on how this is done in real projects.



Micrographs (from MotionCorr): 2/corrected_micrographs.star
Particles (from Refine3D or CtfRefine): efine3D/job049/run_data.star
Postprocess STAR file: cess/job051/postprocess.star
First movie frame: 1
Last movie frame: -1
Extraction size (pix in unbinned movie): -1
Re-scaled size (pixels): -1

1. On I/O tab:
(Fill in appropriate values)
`Extraction size (pix in unbinned movie):`
`426`
`Re-scaled size (pixels):`
`252`

2. On Train tab:
`Train optimal parameters?`
`No`

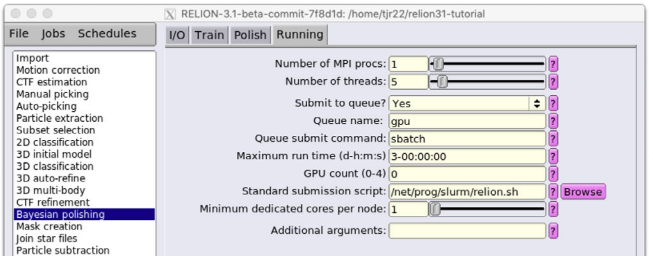


Perform particle polishing? Yes
Optimised parameter file:
OR use your own parameters? Yes
Sigma for velocity (A/dose): 0.2
Sigma for divergence (A): 5000
Sigma for acceleration (A/dose): 2
Minimum resolution for B-factor fit (A): 20
Maximum resolution for B-factor fit (A): -1

3. On Polish tab:
`Perform particle polishing?`
`Yes`

`OR use your own parameters?`
`Yes`

`Submit to queue?`
`Yes`



Number of MPI procs: 1
Number of threads: 5
Submit to queue? Yes
Queue name: gpu
Queue submit command: sbatch
Maximum run time (d-h:m:s) 3-00:00:00
GPU count (0-4) 0
Standard submission script: /net/prog/slurm/relion.sh
Minimum dedicated cores per node: 1
Additional arguments:

4. On Running tab:
`Number of MPI procs:`
`1` (4 for real data, or fewer if you get a memory error)

`Number of Threads:`
`5`

`Submit to queue?`
`Yes`

For real data:
`Queue submit command:`
`sbatch --exclusive`

Click *`Run!`* ONCE. This should take ~10 mins.

Now repeat the 3D auto-refine and post-process jobs to see if there's any improvement (make sure to reconstruct a new map and align it to D2, and use the Polishing job `shiny.mrc` file as the input particles!)

Now that we're trying to squeeze the best out of our data, we want to set `Mask individual particles with zeros?` to `No` in the 3D auto-refine `Optimization` tab and set the `Skip Padding?` option to `No` in the `Compute` tab.

Choosing your box size

This tutorial has so-far presented a fairly complex, math-heavy way of choosing your box size. In reality, most people just choose a box size twice the size of their particle. For example, for have a 160 Å particle at 1.5 Å/pix, a typical final box size would be ~214 pixels, with most users choosing 200, 220, or sometimes 256 pixels.

Ultimately, the box size you choose for your final, highest-resolution, map(s) is a compromise between two factors:

- Bigger boxes incorporate more delocalization information
- Bigger boxes incorporate more noise.

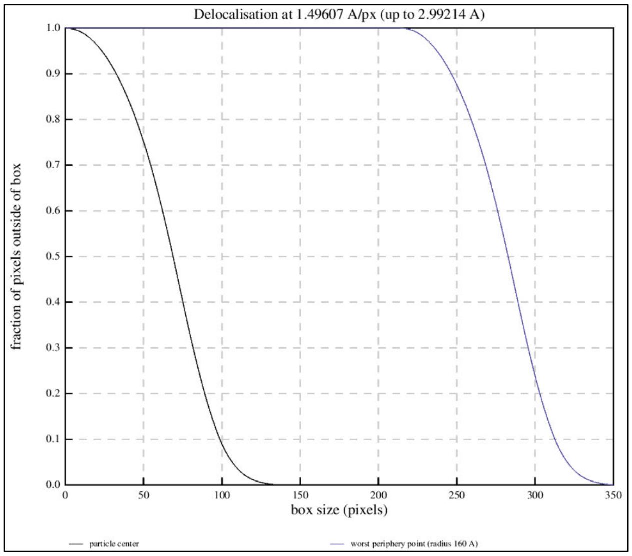
… so the only real option is to try several box sizes out and see which gives the best map (or, if you're looking for a quick proxy, the least-negative fitted sharpening b-factor.) Relion does, however, provide one hint as to how big our box sizes should be.

1. Plot the delocalization for some particular particle set, including the radius of your particle (in this case 160Å)

```
relion_plot_delocalisation \
--i Polish/job###/shiny.star \
--o Polish/job###/deloc \
--rad 160
```

2. Open the plot using evince (or download to view with your pdf viewer of choise)

```
evince Polish/job###/deloc.eps
```



The black line shows how many pixels of information you exclude for the center of your particle at some particular box size. The blue line is a bit more complicated. It shows the 'worst case'. You can think of it like this: for a spherical particle, how many of the particle images have all the delocalized information captured before you get to the nearest edge.

So, for our box size of 252 pixels, we have all of the information from the center of our particles, but only ~15% of the images have all of the highest resolution information from the edge of the particle. This may sound bad, but recall that the delocalization is circular, so most of the highest-resolution information will still be captured in most of the particle images, along with virtually all of the lower-resolution information.

The take-home message from the plot is this: There is virtually no information for boxes over 300 pixels, and none at all for boxes greater than 350.

Next steps

After post-processing, the map is at 3.09Å with a fitted b-factor sharpening of -22.71.

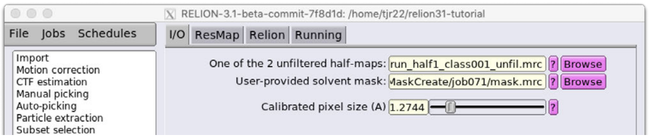
Now we repeat the CtfRefine – Polish – 3D auto-refine cycle until we get no improvement.

1. CtfRefine
    - Turn on 4th order aberration estimation – normally we wouldn't do this until we were below 3Å.
2. CtfRefine (anisotropic magnification)
    - Use particles from previous CtfRefine job
    - Turn on `Estimate (anisotropic) magnification`. This will turn off all the other estimations.
3. Run a 3D auto-refine job (optional)
4. Run a Bayesian Polishing job using default settings. For a 2.6Å target resolution, we use a box size of 464 pixels down-sampled to 316 pixels.
5. Run a 3D auto-refine job and post-process. You should now have a map at <3.0Å and fitted b-factor sharpening of > -20 (i.e. -19)

- Repeat the above cycle, using the new map.
    - Train the Polishing job (this takes a long time.)
        - Normally, you would use 5-15K particles for training. The more particles, the better the parameter estimates – but you may run out of memory. If this happens, reduce the number of particles and try again. If this still doesn't work, set --sbs in the `Additional arguments` field in the `Running` tab (but this will make training even slower!)
        - In this case, because we only have 4-5k particles, use 3000 particles.
        - On the Running tab, set:
            - `Queue submit command` to `sbatch --exclusive -C skylake` to ensure you use the new nodes with more memory and you reserve all the memory on the node.
            - MPI to 1 and Threads to 16.
    - Use the trained parameters by pointing the `Optimised parameter file:` to your training job output.
- Once you've done the post-processing, repeat the Refine3D and post-processing jobs:
    - Reference map: `run_class001.mrc` from previous Refine3D job
    - Reference mask: soft mask derived from the reference map
- Repeat the above, turning on `solvent-flattened FSCs` on the `Optimization` tab of the Refine3D job.
- The resolution (sharpening b-factor) obtained in writing this tutorial, after post-processing, were:
    - Refine3D: 2.85Å (-19.84)
    - Refine3D with masking: 2.85Å (-18.25)
    - Refine3D with masking and solvent-flattened FSCs: 2.85Å (-18.37)
- People often want their final map to be a 'full' resolution map. If you try this, you'll find a box size of 464 pixels is too big for the GPU memory. Instead, repeat the final Polish, Refine and Post-processing jobs with a box size of pix (and no down-sampling).

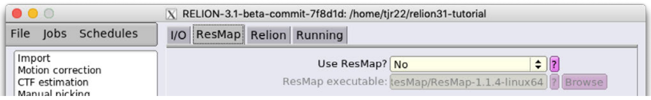
Local Resolution

Once you have obtained a density map, it can be useful to display the local resolution estimate to get a better feel for your results. Here, we use Relion's built-in local resolution estimator for the calculation, then UCSF Chimera for display.



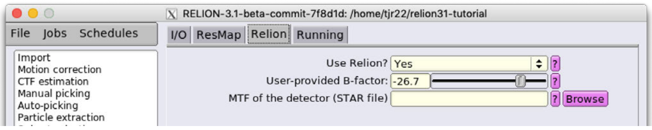
One of the 2 unfiltered half-maps: run_half1_class001_unfil.mrc
User-provided solvent mask: MaskCreate/job071/mask.mrc
Calibrated pixel size (A) 1.2744

1. On I/O tab:
`One of the 2 unfiltered half-maps:`
`Refine3D/job###/run_data.star`
`Calibrated pixel size:`
(from `Extract/Polishing` job)



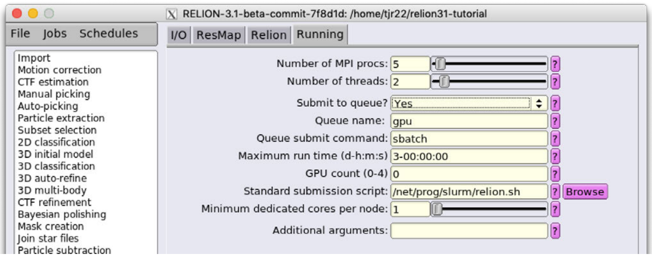
Use ResMap? No
ResMap executable: esMap/ResMap-1.1.4-linux64

2. On ResMap tab:
`Use ResMap?`
`No`



Use Relion? Yes
User-provided B-factor: -26.7
MTF of the detector (STAR file)

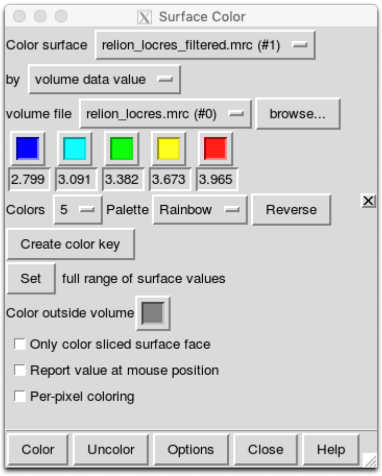
3. On Relion tab:
`Use Relion?`
`Yes`
`User-provided B-factor:`
(from `Post-process` job)
`MTF of the Detector (STAR file)`
`/net/common/MTF/mtf_k2_300kV.star`



Number of MPI procs: 5
Number of threads: 2
Submit to queue? Yes
Queue name: gpu
Queue submit command: sbatch
Maximum run time (d-h:m:s) 3-00:00:00
GPU count (0-4) 0
Standard submission script: /net/prog/slurm/relion.sh
Minimum dedicated cores per node: 1
Additional arguments:

3. On Running tab:
`Number of MPI procs:`
`4` (8 for real data)
`Number of Threads:`
`2`
`Submit to queue?`
`Yes`

Click ***Run!*** ONCE. This should take ~45 mins.



Color surface relion_locres_filtered.mrc (#1)
by volume data value
volume file relion_locres.mrc (#0)
2.799 3.091 3.382 3.673 3.965
Colors 5 Palette Rainbow
Set full range of surface values
Color outside volume
☐ Only color sliced surface face
☐ Report value at mouse position
☐ Per-pixel coloring

4. In UCSF Chimera:
    4. Open `relion_locres_filtered.mrc` and `relion_locres.mrc` files from your `LocalRes/job###` directory
    5. Hide the `relion_locres.mrc` map
    6. On the `Volume Viewer`, click on `Tools > Surface Color`.
    7. Color surface relion_locres_filtered.mrc by volume data value
    8. Set volume file relion_locres.mrc
    9. Click Options
    10. Click Reverse
    11. Click Color
    12. (Optional) Change the histogram contour and click Set (full range of surface values)

Cleanup

Relion projects can take up lots of space. In most cases, most of this space can be reclaimed once the processing is done. Relion includes a `Gentle clean` option to remove the intermediate steps from jobs (we have already used this for removing FOM maps in picking.) This should be run once you're happy that you won't need, for example, iteration 17 from a 25 iteration 2D classification. You can also gently clean an entire project once you've decided that a project has finished.

Relion also includes a Harsh Clean option, which will remove nearly everything apart from the star files that record what you did. You should use this once you've decided a project isn't worth doing anything with, but you still need to keep it, i.e. to match up to your lab notebook, etc,…

1. Empty Trash
   1. On Relion menu bar, click on `File`
   2. Click on `Empty trash`

2. Gently clean all jobs
   1. On Relion menu bar, click on `Jobs`
   2. Click on `Gently clean all jobs`

Appendix: External jobs

Relion has the ability to run external jobs and include the results into it’s pipeline. So that software written by various groups can be used, we have written a small Python package called relion-external.

The list of currently available packages is:

- Topaz
  - Picking using pre-trained or your own model
  - Training
  - Denoising
- crYOLO
  - Picking using pre-trained or your own model
  - Training
  - Denoising
- Janni
  - Denoising

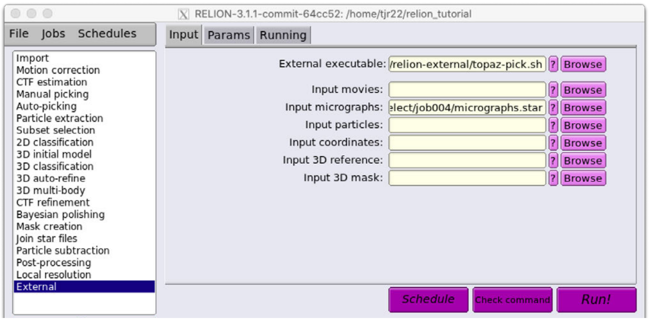
External: Topaz picking

Although it is essential that you examine your data (which is why we recommend at least a quick manual-picking strategy,) AI based autopickers perform very well on most datasets. We have integrated Topaz (https://github.com/tbepler/topaz) into Relion via the External job type.

Not all of the features of Topaz are currently available. If you require a feature we haven't yet implements, please let us know. To see what features are available, go to the command line and run:

```
/net/prog/relion-external/topaz-pick.sh -h
```

Note that the Relion external job will add "`--`" to any option label. For example, if you want to specify the model path, use "`model`" as the label in the `Params` tab, not "`--model`". Here we demonstrate how to setup a default run.



RELION-3.1.1-commit-64cc52: /home/tjr22/relion_tutorial

External executable: /relion-external/topaz-pick.sh

Input movies:

Input micrographs: elect/job004/micrographs.star

Input particles:

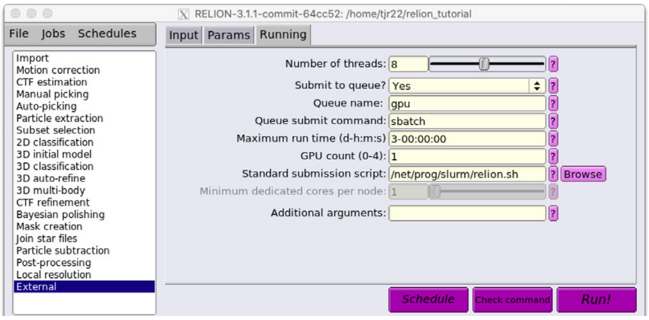
Input coordinates:

Input 3D reference:

Input 3D mask:

1. On I/O tab:
`External executable:`
`/net/prog/relion-external/topaz-pick.sh`

`Input Micrographs`
`Select/job###/micrographs.star`



RELION-3.1.1-commit-64cc52: /home/tjr22/relion_tutorial

Number of threads: 8

Submit to queue? Yes

Queue name: gpu

Queue submit command: sbatch

Maximum run time (d-h:m:s) 3-00:00:00

GPU count (0-4): 1

Standard submission script: /net/prog/slurm/relion.sh

Minimum dedicated cores per node: 1

Additional arguments:

2. On Running tab:
`Min. Number of threads:`
`8` (use up to 8 threads per GPU)

`GPU count:`
`1` (use 4 for a real dataset)

Click on the finished job, and display the results in the same was as for any other picking job. You can adjust the default threshold between `-6` and `6` (the default is 0). Higher values mean fewer picks. If you add a `threshold` in the `Params` tab and continue the job, relion-external will use the already-generated Topaz scores rather than re-running Topaz from scratch.

Note that Topaz generates scores across batches of micrographs - so if you have very poor picking it's possible your problem is not in the micrograph you're looking at but another in the batch. For example: users sometimes accidentally include the gain-reference images output by EPU in their projects. If one of these gain-reference images is included in a batch by Topaz, very poor picking often results.

Appendix: 3D classification with multiple initial maps

If you ask Relion to run 3D classification with 4 classes, by default it will make four copies of your initial map, randomly assign each particle to one of the classes, and use the resulting reconstructions as starting points for further classification. The small variations this procedure produce are often more than enough for subsequent iterations to amplify into the classes we want.

If, however, you're concerned that a starting map is too biased towards one particular structure, you can run a 3DAutoRefine job on the whole data set to get a consensus map. That map can then be used as a starting template for further classification.

In rare cases, though, we may want each class to be 'seeded' with very different initial maps (see, for example, Morris et. al, NSMB 2019 26(10), 890-8). We can do this using an appropriately constructed `.star` file.

For example, if we had four maps in a directory called e.g. `relion_project_dir/my_maps/map1.mrc` (where `relion_project_dir/` is our relion project directory,) we could create a star file `relion_project_dir/my_maps/initial_models_for_jobXXX.star`:

```
data_
loop_
_rlnReferenceImage
my_maps/map1.mrc
my_maps/map2.mrc
my_maps/map3.mrc
my_maps/map4.mrc
```

And then put this star file in the `Reference Map:` field of the IO tab of a 3D Classification job. Note that if you provide 4 maps and ask for more or fewer than 4 classes, Relion will use 4 classes, but subsequent metadata handling will get confused – so make sure that if you specify the same number of classes on the `Optimization` tab as entries in your `.star` file.