Frequently Asked Questions

Where are the data files stored?

*cis*TEM sets up a directory and file structure to store results, including aligned movie averages, 2D class averages, particle stacks, alignment parameters (Frealign format) and 3D reconstructions. The following directories are found inside a *cis*TEM project directory:

```
Assets/ClassAverages
Assets/CTF
Assets/Images
Assets/Images/Scaled
Assets/Images/Spectra
Assets/Movies
Assets/Parameters
Assets/ParticlePosition
Assets/ParticleStacks
Assets/Volumes
Assets/Volumes/OrthViews
Scratch/AutoRefine3D
Scratch/ManualRefine3D
Scratch/Refine2DScratch/Startup
```

The `Scratch` directories contain temporary results while *cis*TEM performs computations and their contents is deleted after a job finishes. The project directory also contains the database file with a `.db` extension.

Files in the `Assets` directories follow a name convention that adds one or two numbers to each file name. For example, the 2D class averages inside `Assets/ClassAverages` are named

```
class_averages_0001.mrc
class_averages_0002.mrc
class_averages_0003.mrc
class_averages_0004.mrc
class_averages_0005.mrc…
```

to indicate the class averages resulting from iteration 1 (starting classes), 2, etc. The number in the file name corresponds with the Classification # indicated with the **2D Classify** results. For example, `class_averages_0004.mrc` corresponds with `Classification #4`. Another example is the `Assets/Volumes` directory, which contains 3D reconstructions. For example, if reconstructions were generated using the **Ab Initio 3D** Action, the files are named

```
startup_volume_1_1.mrc
startup_volume_2_1.mrc
startup_volume_2_2.mrc
startup_volume_3_1.mrc
startup_volume_4_1.mrc…
```

and are referred to in the **3D Volumes** Asset panel as

```
Volume From Startup #1 – Class #1Volume From Startup #2 – Class #1
Volume From Startup #2 – Class #2Volume From Startup #3 – Class #1
Volume From Startup #4 – Class #1
```

In this case, the first number indicates the iteration or run of a job, the second indicates the class number. For 3D reconstructions resulting from **Auto Refine** or **Manual Refine**, the files are called

```
volume_1_1.mrc
volume_2_1.mrc
volume_2_2.mrc
volume_3_1.mrcvolume_4_1.mrc
```

with the same conventions for the numbers as above, and they are referred to in the **3D Volumes** Asset panel as

```
Start Params #1 – Class #1
Local #2 (Rnd. 1) – Class #1
Local #2 (Rnd. 1) – Class #2
Local #3 (Rnd. 2) – Class #1Local #4 (Rnd. 3) – Class #1
```

(in this example, local refinement was performed with an imported Refinement Package). The sub-directory `OrthViews` contains files with names matching those in the `Assets/Volumes` directory, but containing central sections and projections in three orthogonal views for diagnostic purposes.

Advanced users are encouraged to look through the different directories and learn where different files and results are located and can be retrieved.

Can I move a project directory?

Project directories can be moved to a different directory. *cis*TEM will attempt to update the paths to all Asset files inside the project directory with the new paths. However, the paths to imported

data files, such as movies, images and particle stacks, are not updated and have to remain accessible at their original location (accessibility could be provided via symbolic link).

Can cisTEM read compressed tif files?

*cis*TEM can read movies that are stored as `tif` files and, optionally, contain LZW compression, a lossless data compression format. Some microscope automation software, such as SerialEM, can store movie data directly as compressed `tif` files. Compression is most efficient when using electron counting detectors and storing movie frames before they are gain-corrected because the uncorrected frames contain integer values that are mostly 0, 1, 2 and 3. *cis*TEM can read uncorrected movies and perform gain correction on the fly with a separately provided gain reference. However, even after gain correction, LZW compression can significantly reduce disk space requirements and accelerate job execution when disk I/O is rate-limiting. For example, a movie of ?-galactosidase (Bartesaghi et al. 2015) containing 38 gain-corrected super-resolution frames collected on the Gatan K2 camera takes about 8.1 GB without compression, but only about 700 MB with LZW compression. Aligning these movies on a 44-core workstation takes about 4.5 hours when working with uncompressed data and only 1.3 hours after compression. Compression and conversion to `tif` files that can be read directly by *cis*TEM can be done using IMOD's (Mastronarde, 1997) mrc2tif tool:

```
mrc2tif –s –c lzw movie_file.mrc movie_file.tif
```

References

Mastronarde, D.N., 1997. Dual-axis tomography: an approach with alignment methods that preserve resolution. J. Struct. Biol. 120, 343–352. doi:10.1006/jsbi.1997.3919

Bartesaghi, A., Merk, A., Banerjee, S., Matthies, D., Wu, X., Milne, J. L., Subramaniam, S., 2015. 2.2 Å resolution cryo-EM structure of ?-galactosidase in complex with a cell-permeant inhibitor. Science 348, 1147-1151. doi:10.1126/science.aab1576

How do I run cisTEM using multiple computers?

by Craig Yoshioka

Intro

cisTEM has been designed to run using worker processes that communicate to the GUI through an intermediate job management process that is launched on job initiation. Communication between these entities is done using TCP network sockets, and so all the computers participating in processing need to be able to find and communicate with each other for everything to work.

Networking

Firewall

The computer running the cisTEM GUI will open network ports, as needed, in the range (3000-5000) to listen for connecting job managers. The job managers, in turn, will open ports to listen for worker processes that they spawn. It is thus simplest to configure Firewalls to allow incoming and outgoing connections between all the computers involved.

Multiple Network Interfaces

In a complicated cluster environment it is likely that there are multiple network interfaces available on different nodes. By default, cisTEM will open a port on every network interface it finds, and workers will work through the list of given IP addresses until they find one that connects back to the job controller. This process can be costly in a complicated network environment so in these cases it is best to manually specify the IP addresses to be used by the GUI and job controller. This can greatly speed up the ability of cisTEM to "wire" itself together, and in certain extreme circumstances prevent failures due to timeouts. It also gives the benefit of allowing one to specify the "best" network fabric to use in cases where there are redundant ones.

System Resources

Due to the 100s of connections a job controller might make, it could be possible to exhaust system resources limits, especially on busy shared systems, such as head nodes. Sysadmins should be aware of this possibility and increase what limits they reasonably can (https://easyengine.io/tutorials/linux/increase-open-files-limit), but users should also be aware of the possibility of excessive load, and adapt to reduce resource use on busy systems. Further instructions on how this can be accomplished are given below for specific Job Schedulers.

Job Schedulers

Slurm

There are several ways to use cisTEM with Slurm, each with its advantages and disadvantages. It might also be the case that some Slurm cluster configurations allow more access than others, and thus open up more interesting ways in which to run cisTEM, but here we will work with the assumption that users do not have direct SSH access to Slurm processing nodes, and will likely start by running the GUI on the head node.

Slurm 1 - Simple Config

Start cisTEM GUI on the head node. If your cluster head node has many network interfaces (which is likely), than it is recommended to specify the one that is most appropriate for communicating with the processing nodes and enter that address for **GUI Address** and **Controller Address** in the Settings panel. Then modify the **Command** for each run profile to look like:

```
sbatch -n 1 -c 1 --mem #{GB wanted}G -p #{partition name} --wrap="$command"
```

Swap in appropriate values for `#{}`. Also think about raising the **Launch Delay** (to 100ms or more) to lessen the load on the job scheduler when trying to start 100+ worker processes. The above should get you running (or pretty far), but please look further at the documentation for sbatch, particularly the `-t` and `--gres` options depending on your cluster configuration.

One benefit of this approach is that it will make fairly effective utilization of cluster resources since each worker is scheduled individually. It also makes it trivial to change the number of worker processes wanted in cisTEM. The downside is that if the cluster is very busy, then workers may get queued or delayed unpredictably, which can make cisTEM sad under heavy cluster load. Your Slurm cluster might have partitions that are specially configured to reduce latency for starting jobs during high loads (at the cost of longer run times)... if so, think about using one of these partitions for cisTEM.

Slurm 2 - Offload Job Manager

f system resource utilization on the head node is a concern (open file sockets, etc.), then it is possible to launch the cisTEM_job_manager on a processing node by changing the **Manager Command** field to:

```
srun -p #{partition name} $command
```

For this to work, though, one has to reset **Controller Address** back to Automatic (since you won't know the IP address of the assigned node ahead of time).

Slurm 3 - Single Allocations

If submitting individual worker jobs to Slurm becomes a problem (load, queuing, etc.), then it is possible to configure cisTEM to make a single allocation for each job. To best understand the rest of this section, it is good to know that Slurm allocates resources and manages processes in two distinct steps. Many of the Slurm command-line tools combine both actions, but this is primarily for convenience. When a command like `srun` or `sbatch` is used, the user specifies both the number of processes wanted and the amount of resources needed. Slurm will then wait until it can allocate the requested resources, at which point it will run as many copies of the command the user requested, distributing the processes over the allocated resources. Alternatively, allocation and process spawning can be done in two distinct steps using `salloc`. When using `salloc`, one requests the resources, and once the allocation has been made, it will create an environment that causes `srun` to spawn processes in the current allocation. So to recap: when outside of an allocation `srun` both allocates resources and spawn processes, but from within an existing allocation, it just spawns processes using the currently allocated resources.

To get this to work change **Manager Command** to:

```
salloc -N #{nodes wanted} -c #{cores per node} -p #{partition to use} srun -N 1
```

and change **Command** to:

```
srun -N 1 -n 1 -c 1 $command
```

the **No. Copies** should equal: `#{nodes wanted} * #{cores per node} - 1`

To give a brief explanation using a concrete example:

Manager Command:

```
salloc -N 8 -c 44 -p mpi srun -N 1 -n 1 -c 1 $command
```

Command:

```
srun -N 1 -n 1 -c 1 $command
```

No. Copies: 351

Launch Delay: 50

What happens is that the GUI requests a new allocation for 8 nodes with 44 cores each (total 352 cores) and when allocated, immediately run cisTEM_job_control using a single process and core in that allocation. Since cisTEM_job_control is now running in an existing allocation, it then spawn 351 copies of the worker process in the current allocation when it calls `srun`. This approach will prevent unexpected delays in executing worker jobs, but requires one to "do the math" to allocate the appropriate # of CPU cores wanted. Also, for some steps in cisTEM, such as merging, it can also be wasteful since only 2 CPU cores (out of the 351 allocated) will be in use.

Filesystems

NFS

cisTEM runs fairly reliably from NFS, and this might be a preferred filesystem for running cisTEM unless I/O contention becomes a major bottleneck. The reason NFS becomes rarer in larger cluster configurations is because a single server just cannot scale effectively to cover many users running many jobs across many different processing nodes. That said, it is likely your cluster will have some NFS options available, and as long as the load on the NFS server is reasonable, this is probably the easiest way to get cisTEM running across multiple nodes.

Lustre

cisTEM can be made to run very well from a distributed filesystem like Lustre with the exception of the SQLite database file that it uses for Projects. This does pose a problem since the default

behavior when creating a project in cisTEM is to create a new directory for the Project with the database file inside and directories for `Assets` and `Scratch`. Optimally, it is preferable to have `Assets` and `Scratch` on Lustre (as well as any raw or imported data), but the database file to an NFS server, or local to the GUI. What we would advise for the moment is to close cisTEM after creating a new Project, copy the database file to an NFS server, and then create a symbolic link from the new location to the old.

BeeGFS

We have only started testing cisTEM on BeeGFS, but in most respects it should have the same caveats as Lustre with slightly different tradeoffs. For example, in our limited testing, BeeGFS has done better than Lustre at dealing with files like cisTEM's SQLite database, but is still not ideally suited to this sort of file I/O. Nevertheless, so far we have not had to copy database files off of BeeGFS to get acceptable performance. Furthermore, BeeGFS can be configured so that users can easily create their own on-the-fly distributed filesystems using disks on processing nodes, such as SSDs (https://www.beegfs.io/wiki/BeeOND). This provides many of the benefits of using local SSD scratch storage while also providing a single filesystem across multiple processing nodes. The only drawback is that data must be staged into and out of the filesystem. Thus it is currently only useful in cases where a user has long standing exclusive access to a number of machines.

How do I start 3D classification?

For 3D classification, it is recommended in most cases to first refine with a single class. This saves time since only one class is refined, and it ensures that the classes that emerge out of classification will be all be roughly aligned with each other, making it easier to recognize structural differences between the classes. However, if particles in a dataset vary significantly in size and structure, it may be better to refine with several classes from the beginning, skipping the refinement with a single class.

3D classification starts with the creation of a Refinement Package that specifies the number of desired classes. Follow the steps described in Refinement Package Assets and create a Refinement Package either using another Refinement Package as a template, or by creating one based on Particle Positions or 2D class averages. The number of 3D classes has to be specified in the field “Number of classes for 3D refinement” when running the Refinement Package wizard.

When using a Refinement Package as a template that has more than one class (from an earlier 3D classification), the wizard solicits a number of additional questions that allows the user to select which of the existing classes (particles and alignment parameters) will be used, and how to assign these classes to the classes in the new Refinement Package. A particle is assigned to the class for which it has the highest occupancy. Classes can be excluded from the new Refinement Package by answering No to “Carry over all particles?” The corresponding particles will be excluded from the newly created stack and subsequent refinement. All classes that are carried forward can then be assigned to the classes in the new Refinement Package. This step also allows the combination of several classes into a single new class. Therefore, using the wizard, all possible combinations of class selections and removals are possible.

To start refinement and classification with the new Refinement Package, use the **Auto Refine** or **Manual Refine**

Action and select the new Refinement Package as the active input for the refinement. Active references for each of the classes in a Refinement Package can be selected or changed in the **Assets** panel, as well as in the **Manual Refine** Actions panel. When running **Auto Refine**, the starting references will always be generated from the parameters after randomizing the particle occupancies.

How do I define focused masks?

A focused mask can be used in the **Manual Refine** panel to focus classification on sub-regions of a 3D map. A focus mask is defined as a sphere specified by radius and x,y,z coordinates of the sphere center, all given in Å. The x,y,z coordinates are measured from the first voxel in the volume. The sphere is then projected along the direction of the view of each particle to define a circular region that is used to evaluate differences between the reference/class average and the particle. In other words, the focus mask excludes regions that do not contain relevant signal for the classification. A radius of 30 to 40 Å is usuall appropriate but smaller or larger values can also be tried to optimize results.

One way to determine the correct x,y,z coordinates is to load the 3D map into UCSF Chimera and place a sphere onto the area of interest:

1. Open the 3D map in Chimera. The map should be assigned model #0.
2. Display the command line by selecting **Tools > General Controls > Command Line**.
3. Execute `vop threshold #0 maximum -1000 setMaximum 1.0` (this assumes that the minimum voxel value in the 3D map is larger than 1000). This will create a new model #1.
4. Execute `shape sphere radius 30 color red mesh true` on the **Command Line**. This will generate a sphere with a 30 Å radius, numbered model #2 (change the radius as needed).
5. Open **Tools > General Controls > Model Panel** and deactivate models #0 and #1. Then move the sphere (model #2) into model #1 until it is fully contained in that volume.
6. Execute `mask #1 #2` on the **Command Line**. This will generate a new model #3 of a solid sphere with a 30 Å radius.
7. Close models #1 and #2 in **Model Panel** and make sure model #0 is visible but deactive.
8. Select model #3 (the solid sphere) in **Volume Viewer** and change its color to red (or something easily distiguishable from the original 3D map) and make it transparent.
9. Move model #3 to the desired location of the focus mask in the 3D map.
10. Execute `vop resample #3 onGrid #0` on the **Command Line**. This will generate a copy of the solid sphere as a new model #1, now resampled in the same coordinate system as the original 3D map.
11. Execute `measure center #1` on the **Command Line**. This will display the x,y,z coordinates of the mask in pixel coordinates below the command line. These coordinates have to be converted to Å by multiplying them with the pixel size of the 3D map before they can be used in *cis*TEM's **Manual Refine** panel.

What are the columns of the Frealign X parameter file?

Frealign X particle alignment parameter files contain 17 columns, one more than older Frealign parameter files. The additional column (column #12) contains phase plate phase shifts in radians. The number in the first column identifies the position of the particle the parameters correspond to (duplicates are allowed but are usually not useful). Below are the first 10 lines of a Frealign X parameter file (one line per particle):

```
C           PSI   THETA     PHI       SHX       SHY     MAG  FILM
DF1      DF2  ANGAST  PSHIFT     OCC      LogP      SIGMA   SCORE
CHANGE      1  193.24  110.62  136.55     -8.22     -2.80       0    -
1   4999.6   4890.8  -34.41    0.60  100.00      -210    12.8305
39.40    0.00      2  147.89   94.08  128.93      2.33     -1.88
0    -1   4999.6   4890.8  -34.41    0.60  100.00      -177
16.8191   33.91    0.00      3  261.75  224.60  316.39     -2.89
5.92       0    -1   4999.6   4890.8  -34.41    0.60  100.00      -
251    14.0160   37.30    0.00      4  180.99   74.08  179.94
8.70     -2.83       0    -1   4999.6   4890.8  -34.41    0.60
100.00      -373    24.3739   29.53    0.00      5  196.89   33.78
43.99     -4.74      7.15       0    -1   4999.6   4890.8  -34.41
0.60  100.00      -259    13.1877   39.52    0.00      6  240.10
306.33  194.91     -4.37      7.50       0    -1   4999.6   4890.8  -
34.41    0.60  100.00      -119    11.1007   52.38    0.00      7
70.12   35.30  -14.79      0.02     10.70       0    -1   4999.6
4890.8  -34.41    0.60  100.00      -403    14.9790   20.93
0.00      8   45.42  124.52  290.82     -1.55      2.82       0    -1
4999.6   4890.8  -34.41    0.60  100.00      -152    16.1655   42.96
0.00      9   91.11   93.68    1.03     11.12     -0.53       0    -1
4999.6   4890.8  -34.41    0.60  100.00      -345    30.6854   23.24
0.00
     10  344.02    0.07   16.02    -21.78      3.83       0    -1   4999.6   48
```

Can I terminate the GUI without killing the connected running processes?

Unfortnately, this is not possible at the moment. The GUI is an integral part of most of the processing steps implemented in *cis*TEM. This means that some of the calculations are done by the GUI itself, and results generated by other processes spawned by the GUI will be sent back to the GUI to be recorded and further analyzed. This means that the GUI must be kept running at all times. Terminating the GUI will also terminate all connected processes.

Some users prefer to run cisTEM remotely, launching a new job while at work and then checking on results later from home. This can be accomplished by installing and running remote desktop software such as x2go (open-source, multi-platform). The software creates a virtual desktop on a server, typically a workstation at the workplace that a user can connect to using a client running on a laptop or computer at home. The server sends a compressed video stream of the desktop to the client that allows the user to run software as if they were sitting in front of the workstation. Sessions can be suspended at any time, including when running GUI-driven software such as *cis*

TEM.

How do I run cisTEM using cloud computing?

Michael Cianfrocco explains on his Cloud computing tools for cryo-EM web page how to set up a server on Amazon Web Services to run *cis*TEM. Please follow his instructions.

Table of Contents
