# EMAN2.3 Reconstruction Tutorial
## Using the Project Manager

This tutorial was updated in April, 2019. It should not be used with versions of EMAN2 older than EMAN2.3.

---

➡ Boxes like this one will contain additional information and tips. You can complete the tutorial without reading any of these, but they may help you better understand what you are doing.

➡ The main source for EMAN2 documentation is the Wiki at: http://eman2.org . There is also a Google Group for support and discussions: http://groups.google.com/group/eman2 . Anyone can read/search these sites, you only need to join if you want to post.

➡ **GUI Tips:** EMAN2 will work best with a 3-button scroll mouse, though there are alternatives using keyboard modifiers for one button mice or trackpads on Macs.

  • In most EMAN2 windows (2-D images, 3-D volumes, plots, etc.), the middle mouse button will open a control panel for the widget, which is different for each widget type

  • The right mouse button is used for panning in 2-D or 3-D image windows, and can be used to zoom (by shift+dragging), and to reset the zoom (clicking) in plot windows.

  • The scroll-wheel will generally act as a zoom. control-panel for more precise control

  • If you have a one button mouse, holding down the alt/option key combined with a mouse click will serve the same role as a middle-click.

  • In the control panels, and other places in the EMAN2 interface you may encounter 'Value Sliders'. A slider is attached to a text-box with a number in it. Dragging the slider controls the number, and entering a number will change the slider. In addition, the text-box can be used to control the range of the slider and get more precise control. By typing **<value** or **>value** in the text box you can change the limits of the slider.

➡ Text you see in *italics* will generally refer to labels in the GUI, such as buttons to press. Text you see in **bold**, are commands to be typed in. Items like: <param> are parameters you should fill in (without the <>). Items like: [param] are optional parameters (again, fill without the [ ]).

➡ Check your version:  The command e2version.py will tell you exactly what version of EMAN you are using. When reporting bugs or asking questions on the mailing list it is critical to include all of the lines of the output of this program with your question.

➡ **Mac Laptops** - If you have a Retina display (most will nowadays), then you have a lot of control over the resolution of your display. By default, your display will give you an effective resolution of 1440x900 pixels. This isn't enough space to do image processing well. For the workshop, we strongly suggest opening System Preferences-> Display and adjusting it to "More Space" which will be 1920x1200 effective resolution.

➡ **Windows users** - Windows **will** make your life more difficult with EMAN2. If you have no choice but Windows, please remember that programs must be run from the windows command-line, not by clicking on icons. Installation of an 'enhanced command-line' tool for windows, such as *Console 2.x*, will make your life somewhat easier.

➡ **Windows**  likes to aggressively kill "unresponsive" programs. This means if you have asked EMAN2.2 to do something from the GUI, and it hasn't finished yet, simply clicking on an open window while you wait may cause the whole program to be killed. We have no solution for this at present other than "don't do that".

**Why EMAN2?**

- Over 200 general purpose image processing algorithms (filters, masks, transformations,...)
- Simple pipeline for reliable refinements to near-atomic resolution
- Additional pipelines for
    - Subtomogram averaging, with full CTF correction
    - Structure validation (tilt validation, class-average/projection matching, etc.)
    - Conformational and compositional hetereogeneity
- Reads and writes virtually every file format used in CryoEM
- A complete GUI workflow interface, with complete logging of all processing
- Wide range of GUI tools for 2-D plotting, image viewing and manipulation and 3-D rendering
- Complete pipeline for tomography and subnanometer resolution subtomogram averaging

There are now quite a few software packages available in the CryoEM community, and new users in this field are faced with a wide range choices. Larger packages like EMAN, SPIDER, XMIPP and BSOFT compete with the task-specific software like RELION, CISTEM and SIMPLE. Learning how to use the larger packages thoroughly can take a little more time than the simpler packages, however, they also offer a much wider range of analytical tools and image processing methods when you inevitably run into problems with your projects.

For some fraction of projects, you could collect your data on a Krios with a K2 camera, process with RELION and achieve a self-consistent high resolution structure. However, what happens when this straightforward process fails to produce the expected result, or even worse, gives you a structure which turns out to be incorrect due to model bias or other problems? How do you detect the problem and correct it?

This is where packages like EMAN2 shine. Like RELION, EMAN2 provides a simple guided path for single particle refinement to near-atomic resolution. Indeed, for "good" data sets it is easily demonstrable that EMAN2.1 and RELION produce virtually identical structures. However, EMAN2 also provides an extensive set of tools for validating the accuracy of your structure, and investigating what the problem is when, for example, you have a specimen with a large degree of conformational or compositional variability. EMAN2 includes at least 6 different methods for exploring different types of specimen variability, and a range of standard tools for insuring self-consistency between raw data and reconstruction.

**Introduction to the Tutorial**

EMAN2 can be used at many different levels ranging from high-level task-based workflow, to command-line utilities, to writing code in Python or C++. In this tutorial, we will be focusing primarily on the task-focused high-level Project Manager interface. This interface will help you work step-by-step through established techniques such as single particle analysis and subtomogram averaging.

We will be using a Beta Galactosidase data set for this tutorial. This data set is a subset of one of the data sets being used in the 2015 Map Challenge sponsored by the EMDatabank. The original paper published with this data achieved a resolution of 3.2 Å. While the full data set can be processed on a workstation in a day or two, it is a bit much for a laptop in a one day tutorial. So, we will be using a subset of the data, which has been down-sampled for faster processing. Even the reduced data set used here is capable of producing about 4 Å resolution, though the final refinement may still be pushing the capabilities of some laptops.

We strongly recommend going through this tutorial using the provided data set. Once you understand how everything is supposed to work, then you can use your own data or download additional public data sets from sites like http://www.ebi.ac.uk/pdbe/emdb/empiar/. The map challenge (http://challenges.emdatabank.org/?q=2015_map_challenge) is another good source for interesting high resolution test data sets.

There are several important things this tutorial does **not** cover:

1) Movie mode processing
   • This is very data intensive, so we are skipping it for the workshop. The relevant programs are:
   • e2ddd_movie.py and e2ddd_particles.py
2) Particle picking
   • There is (or will be) a separate tutorial on the website covering the new particle picking program, including the neural network-based picker.
3) Tomography
   • A big topic. There is a separate tutorial for this at eman2.org.
4) Detailed Command-line usage
   • I will mention this as I can during the presentations
5) EMAN with Python
   • A fun topic, but not for the whole audience. We have some decent recordings covering this online. Please feel free to email the mailing list/google group any time for help.

Check the Wiki (eman2.org) for other available tutorials, some of which include videos

**Time estimates are provided for most steps in the tutorial below. These estimates assume you are using a single quad-core ~3 Ghz computer.**

➡ **If you are an experienced user (or an impatient one), there is a much shorter 'quick' version of the tutorial without all of the detailed explanations, at the end of this document. The section numbering is the same in both versions, so you can move back and forth as necessary. ie- you can go through the short version, and if you don't understand something, jump back to the detailed version for more information. The detailed version also includes many useful comments and explanations for solving problems with your own data, not just the tutorial set.**

➡ **Note that files named __bispec in EMAN2.2 have been renamed __invar in EMAN2.3. The first time you run e2projectmanager in an EMAN2.2 project, it will (try) to update the naming convention in the appropriate files.**

## Detailed Tutorial

Before getting started, it's a good idea to get a feel for the relative speed of your computer (to set expectations). Run **e2speedtest.py**. This will give you a score telling you how fast a single processor is on your computer. If your machine has 4 cores, you multiply this number by 4 to get a total performance number. Note, however, that some processors have a 'turbo' mode, and if you are using only 1 processor (which is what the test does), it will run faster than if you are using 4 cores at once. This can exaggerate the speedtest score by as much as 30%. My 2018 MacBook Pro scores ~1.2 (per core) on this test without turbo mode.

**1. Prepare your project folder**

The tutorial files are distributed in a .zip file. Unzip the sample data archive in a convenient place. This will give you a folder called workshop_2016 (the data is the same as the EMAN 2.12 tutorial) with a folder called *bgal* inside it. Do not run any EMAN2 programs yet. First, from the command line:

**cd workshop_2016/bgal**

The folder you are now in is called the 'project folder'. You will run virtually all commands from this location. All files associated with this project exist in this folder or subfolders within this folder. EMAN2.3 is designed so all of your data and results are self contained, if you use the GUI in the normal way. The goal is that at the end of the project you will be able to trace any step you performed to produce any of your results without ambiguity. Any EMAN2 commands you run should be run from this folder, not from subfolders or other locations.
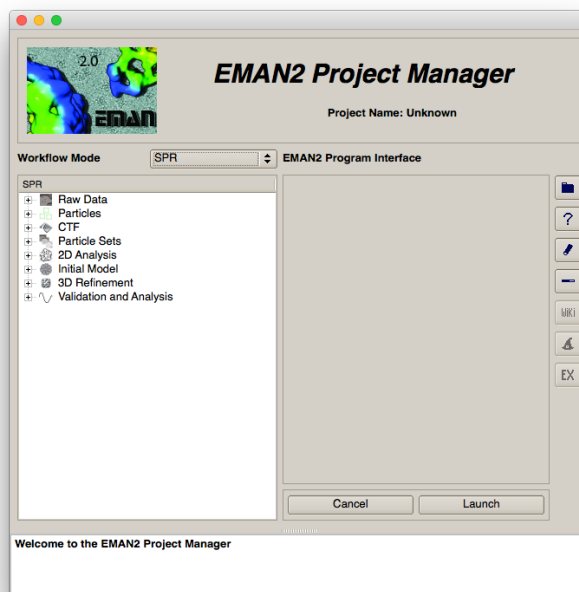
**2. Project Manager**
Type **e2projectmanager.py**. A window should appear that looks like the image above.

- Begin by setting the overall properties of the project, by using the *Project → Edit Project* menu item.
- For the B-gal demo data set, enter the following parameters: mass = 400, Cs = 2.7, voltage = 300, apix=1.275  (these may already be set for you depending on where you got the demo data)
- These parameters are stored in info/project.json, which is a human readable text file. As you progress, many other .json files will appear in this folder, all of which can be read, and (if you are careful) edited with a text editor (NOT a word processor).
- When working on your own project
  - voltage and Cs should be known for your microscope
  - Mass is expressed in kDa, but as resolution increases, this number should actually be less than the true mass since for visualization particles are normally rendered with a lot of vopen space
  - Å/pix (apix) should be a calibrated value for your microscope at the specific mag you used. On typical microscopes this may vary by as much as 5% from the mag reading on the instrument.

## 3. Evaluate images and import data

The next step is to bring the raw data into your project. In this tutorial, we provide you with micrographs and a set of .box files containing pre-selected particle locations. The data set we are using is a small subset of the beta-galactosidase images provided as part of the 2015 Map Challenge. The full data set with full sampling is capable of achieving about 3.2 A resolution.

- ➡ The advantage to the second method is, if you have a significant fraction of 'bad' frames (and most people do), then you can skip particle picking for them. This can save a lot of effort. For this tutorial, there are only 1 or 2 micrographs which are slightly worse than the others, so, you can go through 3b, but you won't find many images to exclude.

- ➡ From this point forward, some of the tasks you complete will take more than a few seconds to run. Open the Task Manager, by selecting the middle icon in the vertical toolbar on the right side of the project manager (         ), and it will show you the progress of EMAN jobs running on the machine.

- ➡ The single particle data you begin with may be either "light" or "dark". That is, the protein particles may appear either lighter or darker than the surrounding solvent in the micrograph. It is CRITICAL for all processing in EMAN that the particle images be "light". If collecting cryoEM data with no stain on a CCD/DDD camera, the contrast in the raw micrographs will be "dark" and should be inverted as the micrographs are imported into the project. The inversion can be done at later stages as well (but should only be done once, if necessary). Many algorithms will fail if this convention is not followed.

- ➡ When working on your own data, you can import already boxed-out particles into EMAN from other software. However, if you have selected particles elsewhere and are starting in EMAN the preferred approach is to import the box coordinates and micrographs, giving you an opportunity to evaluate the micrograph and do micrograph-based in addition to particle-based CTF determination. This is particularly true if you are targeting high resolution. If you are looking at dynamics, or limited to low-intermediate resolutions, it may be fine to simply import the particles and begin at step 5 below.

- ➡ If you have a large number of frames (say >5000), you may give up on the manual approach entirely and decide you have to trust that the frames are all good, and automatic picking and CTF determination are trustworthy. In these projects, there are methods which can be used after initial refinement to weed out bad particles and bad micrographs. These can cost some computational time for the project, but save a lot of human effort.

The downsampled subset we are using will be limited to ~4 A resolution, but will be much faster to process, making it more suitable for a tutorial.

You now have two choices. You can simply trust that all of the data is good, or you can go through the micrographs one at a time and briefly evaluate each to decide whether it is good enough to include. If you opt to simply use all of them (which is fine for the provided tutorial data), you can always return later to learn how to evaluate images.

---

➡ On the Mac, windows may open, but not appear in front of other windows. Click on the Python rocket-ship icon in your dock to bring them to the front if you can't find them.

➡ There are many file formats used in the CryoEM community, including familiar formats like TIFF and PNG, cryo-EM specific formats like IMAGIC and SPIDER, and proprietary formats like DM4 and SER. EMAN2 supports virtually all file formats used in the community (http://blake.bcm.edu/emanwiki/Eman2DataStorage). People often ask "how do I convert file X to work with EMAN2". In most cases, the answer is "you don't have to!" Any EMAN2 program should transparently read any supported file format. Similarly, when specifying output files, simply use the standard file extension and EMAN2 will write to almost any format as well. That said, EMAN2 uses an interdisciplinary format called HDF5 as its native format.

➡ The project manager interface is an expandable tree. Each level of the tree has a form with containing information or parameters, even the items which just appear to be labels for the levels below them.

---

### *** Choose only one of the following ***

(for 1-day tutorials/workshops, you only have time for step 3a, don't try to do 3b)

a) Import all of the frames. You will still have the ability later to discard some if they prove to be bad at a later step. (go to step 3a)

b) Evaluate each frame, and only import ones you decide are good (go to step 3b). Note that there were some bugs in EMAN2.2 which would cause problems later if you did this step. Those have been fixed in EMAN2.21.

### 3a. (~5 min) Import only (choice 1)

To bring all of the frames into the project and determine CTF paramters, without manual evaluation, use :

*Raw Data → Import Micrographs & est Defocus*

- Press the *Browse* button, double click on orig_micrographs, then highlight all of the HDF files in that folder.

- Now, select the *edgenorm*, *xraypixel* and *ctfest* checkboxes. Unselect *invert*, because the images provided for the tutorial have already been inverted. *Astigmatism* correction is not necessary for this tutorial project.

- If you set the project parameters correctly in step 2, then the remaining parameters should already have the correct values: apix = 1.275, voltage = 300, Cs = 2.7 and ac = 10. defocusmin and max should be set to 0.6 and 4.0 which is sufficient for all of the data in this project.

- When you press "Launch" it will process each micrograph in-place as specified and store the resulting CTF information in the info/*.json files. It should take only a few minutes to run (this includes CTF fitting).

### 3b. (manual, ~30 min) Evaluate and Import (choice 2)

➡ IF YOU DID STEP 3a, DO NOT ALSO DO STEP 3b. SKIP TO STEP 4

Automatic and manual data acquisition will normally produce a significant fraction (10-50%) of quantitatively "bad" images. While there will also be later opportunities to eliminate bad micrographs and/or particles, these other methods take place <u>after</u> particle picking. Normally, by excluding obviously bad frames at this stage, you can save the substantial effort of particle picking the bad frames. For the tutorial, we are providing particle locations, so you don't gain much benefit in eliminating bad images at this stage. You may still wish to go through this process to learn how micrographs can be evaluated, though.

Including bad data is NOT harmless. It can lead to model bias and reduce the quality of your reconstruction, even if it does not always reduce the measured "resolution". We have tried to streamline the manual evaluation process so it can be done very quickly (~3-5 seconds/frame):

- Raw Data → *Evaluate & Import Micrographs*
- press the *Browse* button
  - This should cause a browser window to open. Double click **orig_micrographs**
  - Select all images in the **orig_micrographs** folder. Press 'OK'
  - Change *box* to 384. (256-1024 is typical, this has nothing to do with particle size)
  - Assuming you set the project settings properly in step 2 above, the remaining parameters should be fine with their default values.
- Press *Launch*.
  - Four windows will appear: Control Panel, Micrograph View, Plot and 2D FFT. You will need to arrange these windows so you can see them all. Make the micrograph window as large as you can, without obscuring the others. Use the mouse-wheel to zoom the micrograph window so you can see the entire micrograph and its pattern of green boxes. Select the first image file in the *Control Panel*.
  - In the *Control Panel* window, change *Ctf Zeroes* to *None*. This will remove the pattern of green rings obscuring the 2-D FFT. If you wish to check the defocus match in 2-D rather than look for drift and other quality indicators, you can re-enable this.
  - Deselect the *invert* checkbox. The micrographs we are using have already been inverted.
  - In the micrograph window, clicking on any green box will toggle it on/off. Note that it may not be obvious that a single 'off' box in the center of 8 other 'on' boxes is actually off, since the lines overlap. The 'on' boxes define the region used for the power spectrum calculation. If there is contamination or some other artifact present in the image, turning 'off' the boxes in these regions will produce a cleaner power spectrum, but this is mainly for visualization, so there is no need to rigorously turn off all contamination boxes.

- You can then go through the images one at a time and decide which ones are appropriate to further process. Most of the provided images are good. This section assumes some familiarity with TEM imaging. If you don't understand these terms, you should read about them before proceeding. Things to consider:
  - Drift: If the image has too much directional falloff in the 2-D power spectrum, you should consider excluding it.
  - Astigmatism: None of the included images should have significant astigmatism, but when processing your own data, you must decide whether to try and correct astigmatism or to remove overly astigmatic images. For very high resolution studies, astigmatism correction is almost always a good idea. For this tutorial it isn't necessary.
  - Particle concentration: If the particle concentration is so high you don't believe you will be able to isolate a lot of particles well separated from others, you may consider excluding it.
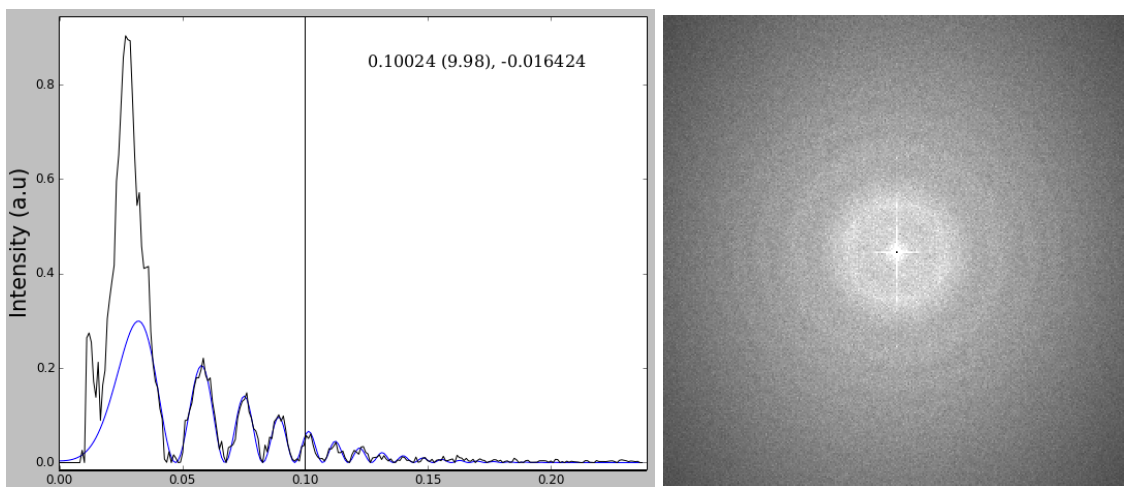


The image above is typical when targeting subnanometer resolution.
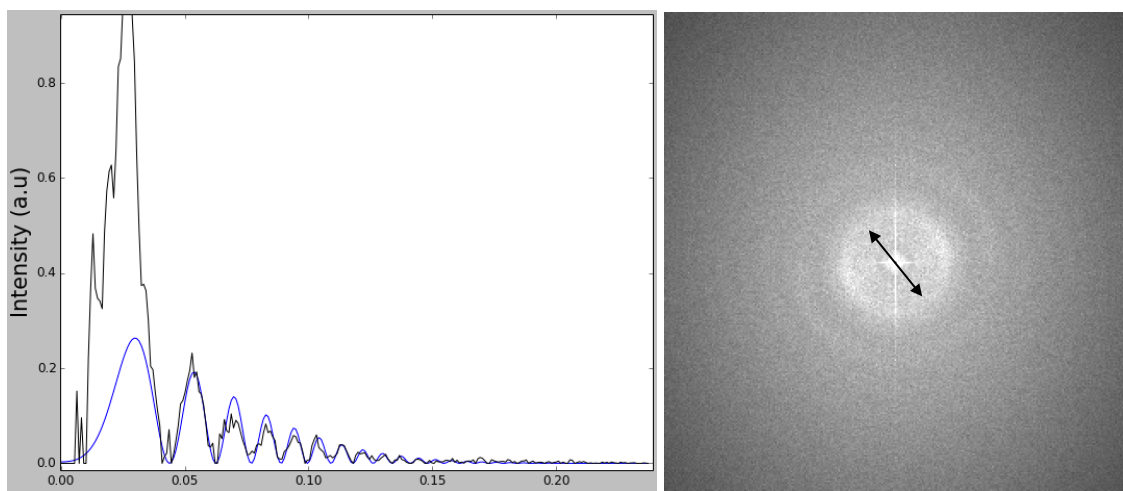


The image above has noticeable drift in the indicated direction (fading rings), but still has good subnanometer resolution signal, so might still be used.
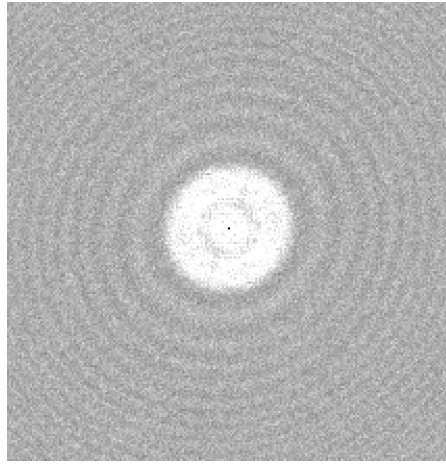
This is an image with excellent resolution, but also astigmatism (non-circular Thon rings). While correctable in most cases, many people just exclude significantly astigmatic images.

Keyboard Shortcuts - To make evaluating and importing images faster, there are a set of keyboard shortcuts you can use. To use these, you must have keyboard focus on the list of micrograph names. To do this, simply click on one of the micrograph names with the mouse. From this point, you can use:

* ∗ up and down arrow - select next/previous image
* ∗ left/right arrow - small changes to defocus
* ∗ i - imports the micrograph into the project
* ∗ u - un-imports a micrograph mistakenly imported
* ∗ 0-9 - sets the quality (before importing)

- If you think you may want to include an image, but it doesn't seem quite as good as the others, you may consider assigning it a lower (or, conversely, higher for particularly good images) quality value. These numbers are qualitative user-assigned values. It can make it easier to select subsets of your data later on, or to locate representative good and bad images within your project. 5 is the default. Larger values normally indicate better images, but there is no set scale.
- For each image you decide to use, press the Import button (harmless to do it more than once). This will copy the image to the micrographs folder, and store the preliminary CTF parameters you've determined. Make sure *invert* is not checked before importing.
- If you accidentally import an image you didn't mean to, you can simply delete the hdf file from the micrographs folder or use the shortcut above. In EMAN2.0 deleting files would have caused major problems. In EMAN2.1, you can manually move files around at pretty much any time.
- **e2evalimage.py <image> ...** (the program you're using now) has many more capabilities you can explore, but for purposes of this tutorial, this is all you need.
- When you have imported all of the good frames, close the *Control Panel* window. This should cause the other windows to close as well.

**4. (~2 min) Extracting/boxing particles from micrographs**

• The data we are using in the workshop came from the first CryoEM map challenge. The particle locations were standardized for this challenge, so, we will use .box files which contain preselected particle locations. These box files include a significant fraction of "particles" which are actually ice contamination. Later in processing we will see a few methods we can use to eliminate most of these bad particles. If you want to learn how to use the new particle picker, there is a short section on it at the end of the tutorial, or follow the tutorial for the new Neural Network based picker:
http://eman2.org/Programs/convnet_pickparticle

• In the Project Manager: *Particles → Import Tools → Import .box or .star files*
  • Use the *Browse* button, and select all of the .box files in the orig_box folder. It is ok to select .box files for micrographs you previously excluded. This will <u>not</u> cause those micrographs to be included in the project unless you import them later.
  • Press *Launch*. **This import process takes only a second or so to complete, and there is no progress display indicating that anything happened.**

• We are now ready to actually extract the images of the particles from the micrographs and save them into particle stack files.

• *Particles → Generate Output*
• Press the Browse button and look at the list of micrographs. You should see a (non zero) number of stored boxes next to each micrograph name. You do not actually need to select these files, we just opened the browser to make sure the box import was successful. If stored boxes shows 0, then something went wrong with step 4, and you should try again or seek assistance.

---

➡ You may have heard the term "stack file". This name derives from the fact that in 'MRC format', sets of 2-D images are stored as a single large image by "stacking" them in 3-D. This is unique to the MRC format. All other formats distinguish between a set of 2-D images and a 3-D volume. To use MRC "stack" files with EMAN2, they must have the .mrcs filename extension. EMAN natively reads and writes virtually all file formats used in CryoEM, but uses the interdisciplinary HDF5 format for all internal processing.

➡ It is critical for good quality reconstructions (there are several reasons) that the box size be 1.5 - 2x the longest dimension of the particle. That is, if a box that just barely contains your particle has a size of 128, you should use a final box size in the 192-256 range. See: http://blake.bcm.edu/emanwiki/EMAN2/BoxSize for information on 'good' sizes to use to optimize processing speed. Picking an appropriate box size at this stage is critical !

➡ On Linux/Mac, you can open the Task Manager( ), and it will let you monitor running jobs to see when they have completed.

➡ When working on your own data, if you have already selected your particles using another program, or prefer to use a different program, it is much preferable to import particle coordinates and micrographs rather than simply import particles using *Particles → Particle Import*.  This will preserve the particle location in the header of each particle, and provide access to the whole micrograph for CTF estimation, etc.

- You should have *write_ptcls* and *allmicrographs* as the only checkboxes selected. By checking allmicrographs, you don't need to use the browser to select which ones to process.
- *box_size* = **288**, *ptclsize doesn't matter.* Then press *Launch*.

- This process should take only a minute. You can use the task manager to see when it's done.


**5. (~10 min) CTF Correction**
We are now beginning the process of correcting for the Contrast Transfer Function (CTF) of the microscope. This process can conceptually be broken into three steps (all three are automatic):
- measuring the CTF parameters of each micrograph
- phase flipping
- amplitude correction

Measuring defocus and astigmatism can be performed on whole micrographs and/or on particle images. If doing astigmatism correction, it is generally better to start with whole micrograph defocus/astigmatism measurement. EMAN has a unique method for measuring additional parameters, such as SSNR (spectral signal to noise ratio) from the particle data, which is used during refinement to make more optimal use of the available data. This step must be performed on particle data, so even if you have estimated defocus on the entire micrographs, as we did above, you still need to perform the second stage of CTF estimation to get these additional parameters.

Luckily, this process is fully automatic, with no human intervention at all. In previous versions of EMAN2, it was mostly automatic, but there were still several manual steps to go through. We now have *e2ctf_auto.py* which combines all of these steps into a single rapid automatic process. We can double check the results to insure there were no failures after running the automated process. Failures will be dramatically reduced if you have first done the automatic whole-micrograph defocus estimation.

➡ If you are working with your own data, and happen to be working in negative stain, we strongly encourage you to perform CTF correction, even if you do not believe it will gain you any resolution. There is more to CTF correction than simply achieving higher resolutions.

> http://eman2.org/NegativeStain

➡ This is a good point at which to explain the file naming convention for raw data within the project. The only thing you need at this point is a folder called *particles*. Inside this folder there should be one image file for the particles from each micrograph:

- Each file <u>must</u> be named *micrographname.hdf* or *micrographname_ptcls.hdf*

- The *micrographname* above must match the files in the *micrographs* folder if present.

- While EMAN supports most file formats, within the project, you <u>must</u> use HDF format, because other formats do not support arbitrary header information. When importing particles from some other software, they are converted to HDF.

- During the following steps, we will be creating modified versions, or 'variants' of the particles (phase flipping CTF correction, downsampling, filtering, etc.) The naming convention used for these versions is a __ (double underscore) separator. eg - *abc123_ptcls.hdf* might become *abc123__ctf_flip.hdf, or abc123__ctf_flip_lp12.hdf.* You are free to make any such variants you like as long as they include exactly the same particles as the parent _ptcls files.

➡ SSNR = Spectral Signal to Noise Ratio. This is a measure of data quality as a function of resolution, and computing this is one of EMAN2's unique features. A Signal to Noise Ratio of 1.0 means there is an equal amount of signal and noise. SSNR is high at lowe resolution, and low at high resolution.

➡ SSNR is additive if proper weighting is applied when averaging particles together. Meaning, if the SSNR were 0.05 at 10Å resolution, it should take roughly 20 particles in that orientation to achieve a minimal average SSNR of 1. In theory you could use this method to achieve arbitrary resolutions, but in practice SSNR values below ~0.02 (50 times more noise than signal) are difficult to recover, regardless of the number of particles.

➡ Do not become too distracted by B-factors you may see. These are not important in EMAN2 processing. Generally, if you find some reasonable consensus B-factor for your data, you can just fill that number into the fixedbfactor box and not worry about it.

➡ When working with your own data, do NOT import "CTF corrected particles" from other software. When programs say they have "CTF corrected particles", it generally means they have been phase-flipped, or phase flipped and filtered with some sort of per-particle Wiener filter. This unknown processing will prevent EMAN from doing optimal reconstructions. When importing particles they must always be the original raw particle images. It is often possible to import defocus/astig values along with the particles if you want to insure both programs use the same values (*e2import.py, e2ctffind3util.py, e2emx.py*).

➡ Along the same lines, EMAN2 has *e2reliontoeman.py* which will take particle stacks and a STAR file from Relion and convert it automatically into an EMAN2 project for comparative refinement. There are also programs to go from EMAN2 to other software (*e2refinetofrealign.py, e2refinetorelion2d.py, e2refinetorelion3d.py, e2emx.py*).

➥ If you don't know how many cores your machine has, now is the time to research it. Please be aware that there is a difference between "cores" and "threads". Current generation Intel processors have a feature called hyperthreading, which may give them, say, "4 cores and 8 threads". Normally the number you should use is the number of cores. On newer machines you may get a 10-20% boost by using roughly the number of cores*1.2, but you should not use the number of cores x2.

➥ If you decide to perform astigmatism correction at some point (do not try and do this the first time through the tutorial), you should first determine astigmatism on the whole frames, and only then do astigmatism fitting in this program.

➥ It is also possible to manually run the *CTF → generate output* program and make your own processed versions of the particles with specific parameters you select.
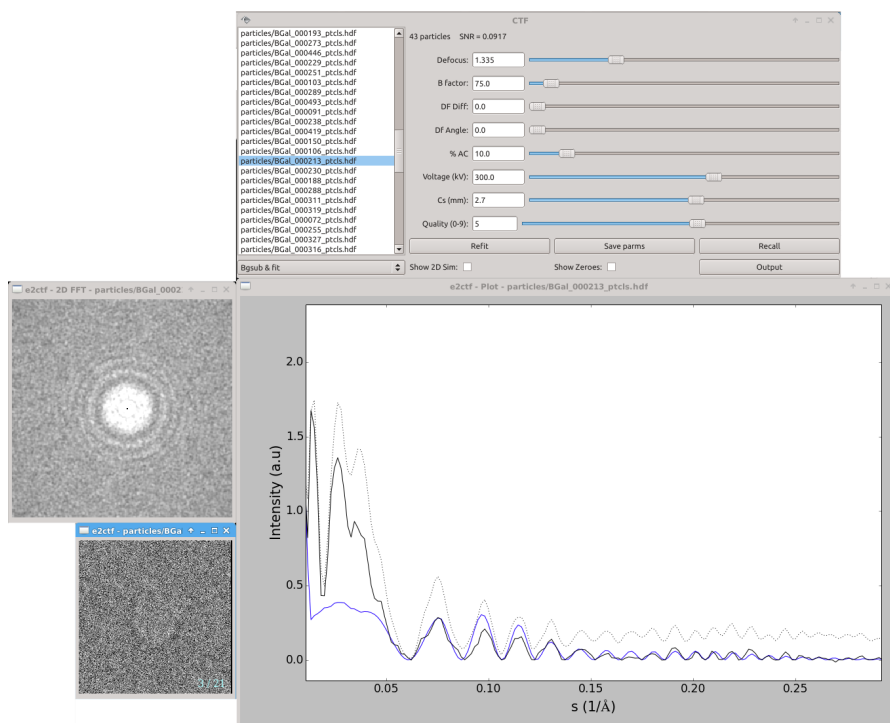
**Automated CTF processing**

- Select *CTF → CTF Autoprocess*
  - check the *hires* box. This gives the automatic processing a hint that the data may extend to near-atomic resolution. If you collected negative stain data, or were using a 20 year old 100 keV scope, you should select *lores*. The tutorial set we are using was collected on a Krios with a K2 direct detector, and easily extends to 4 Å resolution.
  - The Image Parameters are: *apix=1.275, Voltage=300 and Cs=2.7*. If you entered the project parameters correctly above, these should be pre-filled for you.
  - While not absolutely necessary, setting constbfactor to a reasonable value, say 80.0 can improve self-consistency of the fitting results. It shouldn't really impact the final reconstruction.
  - The only other option we need is "threads" which tells the program how many processors to use on the local computer. You should know or find out this number.
- Once the parameters are set, **Launch**. As usual, you can open the progress meter to see how it's doing. This entire process, including generating various types of phase-flipped particles, should take about 5 minutes to complete.
- In addition to fitting the CTF, this program automatically generates several processed output stacks for each micrograph, depending on the resolution target. For high resolution work, these will include CTF phase flipped: _lp12, _lp5 and _fullres.
  - The _lp12 particles have been heavily downsampled, and low-pass filtered to ~14 Å resolution. These particles are suitable for initial model generation, eliminating very bad particles and quick refinements.
  - The _lp5 particles have been less heavily downsampled and low-pass filtered to ~5 Å resolution, and are suitable for quick, intermediate resolution refinements.
  - The _fullres particles are suitable for final refinements to high resolution.
  - Finally, _invar particles, which can help improve 2-D class-averaging, and speed refinements, but are never used directly as particles. These are bispectral rotational/ translational invariants, not particle images themselves.

**5a. (manual, ~10 min) Visual CTF**

*This step is not necessary for the tutorial data, since the autofitting should work more or less perfectly. It is here to teach concepts. You may consider skipping it for now, and coming back to it while you are waiting for a refinement to finish running, or somesuch.*

- When automatic processing is done, it is advisable to check the results to make sure there were no fitting errors. To open the CTF fitting interface, select: *CTF→Visual CTF*
  - Make sure *allparticles* and *sortdefocus* are selected (the other parameters should be automatically set), then press *Launch.* You will get 4 windows (shown below).
  - When automatic CTF fitting does fail, it almost always fails dramatically, producing defocus values at either the low or high extreme. Sometimes this is due to specifying too narrow a search range when autofitting, but there are certain types of particle artifacts which can also confuse the routine. So, at a bare minimum, in this step we want to double-check a few of the closest to focus and a few of the furthest from focus images.
- This analysis is based on particles rather than tiled regions from the micrograph, as used in the earlier micrograph assessment step. Basing the analysis on particles allows us to accurately estimate the SSNR and structure factor (next step) from the particle data. While it is possible to estimate defocus and astigmatism from random boxed out regions of the micrograph, the method used here relies on the fact that each box contains a particle reasonably close to the center of the box. If too many "bad" particles are present, the SSNR and structure factor estimates will be impacted.
- The control panel window (titled CTF) allows you to select which image to work on, and adjust the various parameters for that image interactively. The particles window shows the average of the first 20 particles (without alignment) as well as the first 20 particles individually (use up and down arrow in that window). The 2-D FFT window shows the average 2-D power spectrum for the particles in the current image, optionally (if Show 2D Sim is set) the simulated 2-D curve, and optionally (if Show Zeroes is set) the first several zeroes as blue rings.
- Finally, the plot window shows several curves. In "Bgsub & Fit" mode, the black curve is the background subtracted, rotationally averaged power spectrum. The dotted black curve is the same thing, but using a different background subtraction scheme which exaggerates the CTF oscillations. The blue curve is the fit based on the parameters shown in the control panel. We do not yet have a structure factor curve, so, while ideally the blue and black curves would match perfectly, at this point we can expect significant deviations at low resolution. Don't bother trying to 'fix' this by adjusting sliders, it is normal

for this point in the process. Even later on, there is no need for a perfect fit in amplitudes as long as the zeroes are in the right places.

- While there are many interesting things we can do with this program, at this point we mainly need to insure that the defocus values are correct. Again, the automatic fitting is quite good, but with some specimens, or with particularly low quality data, it will make occasional errors. Generally if it's wrong, it is significantly wrong. If you find an image with a significantly incorrect defocus, adjust it so it's approximately correct, and hit the *refit* button (you can use it several times if necessary). This will be much easier if you zoom in on plot window as shown above. Typically a vertical range of ~0-1 and a horizontal range of ~0-0.25 will work well.
- If this doesn't solve the problem, you can fit manually, then press the *save parms* button instead.
- This project is small enough, you may take the time to look at all of the images. If you have 1000 images, and the automatic fitting seems to work fairly well, you may just check the first dozen closest to focus and last dozen far from focus images. If all of those are fit correctly, odds are good that the rest are too.
- If you didn't screen and eliminate bad micrographs before boxing, this can be a good time to try and identify any bad images in the data set. You can do this by looking for artifacts like uncorrected drift (directional falloff in the 2-D power spectrum) or strong astigmatism (asymmetry in the 2-D power spectrum), or overall poor high resolution signal. This particular subset of the full B-gal challenge data set is almost entirely 'good' images. If you find a micrograph you wish to exclude later, set it's quality to a lower value (the default is 5), and you can use this later to exclude it.
- Again, the primary purpose of this process is to correct any gross defocus errors. If you don't find any, you can just proceed to step 6. If you DID correct any defocus values, before going to step 6, rerun the automatic process in step 5. This will respect any defocus adjustments you made, and regenerate all of the output files accordingly.

---

➡ New windows will be randomly placed on the screen the first time you run a program within a project. Once you have positioned them once, if you run the same program again, they will normally follow your rearrangement.

➡ In the plot window, to zoom in on a region of the plot, drag with the right mouse button. To rescale, click the right mouse button without moving the mouse. Left-mouse drag will show cross-hairs. For the X-axis it will show both the value and the reciprocal of the value (resolution). As usual, middle-click will open the control-panel.

➡ % Amplitude Contrast can only be determined experimentally through some rather tricky experiments, and frankly, slightly different values will not have a strong impact in most reconstructions. Values of 5-15% generally work best for cryo data. However, for negative stain data, you will likely want to use a much larger value than the default 10% (~40-80% will generally work better).

➡ When screening manually, there are some keyboard shortcuts available to speed the process. You must select one of the micrograph names in the list to move the keyboard focus on that widget for this to work: up and down arrow select different images; left and right arrow make small adjustments to defocus; pressing a number 1-9 will set the quality for the current image to that value;  'b' will set the B-factor to 100; 's' will save the current parameters; and 'r' will restore the currently saved parameters.

**6. Building sets**

*This step is provided for reference only. The new automatic CTF fitting program we used in the last step automatically builds a set containing all particles for us. You may want to make other sets later though, after marking bad particles, for example.*

Particle sets (.lst files) are text files which reference the image files which actually contain particles. They are analogous to the STAR files used in Relion and Bsoft, but have more functionality. Once created, you can treat a .lst file in EMAN2 as if it were an actual image file with any program. This allows you to experiment with doing reconstructions with different subsets of your particle data without having many copies of the actual images.
  - *Particle sets→ build particle sets*
    - check the *allparticles* and *excludebad* checkboxes
    - enter "all" int the *setname* box (you can use whatever name you like)
    - If you gave any micrographs a lower quality value when manually assessing them, you could also enter 5 in the *minqual* box, which will exclude any images with the quality set below 5.
    - There are other options for including only particles within a given range of defocus or with other specific parameters. We don't need these for this tutorial, but can be quite useful in other projects.
    - *Launch*


**7. (~20 min) Generating reference free (unsupervised) class averages (2D refinement)**
There are two purposes to this step. First, we need to generate a few good class-averages we can use to make an initial 3-D map for refinement. Second, we can optionally identify some fraction of bad particles and remove them from our data set.  For the first purpose 20-30 class-averages would be sufficient, but if we want to identify bad particles, more averages gives us more precision, and the number of classes has little impact on how long it will take to run.

The standard 2D class-averaging program is called e2refine2d.py and has changed only modestly in the 20 years since the program of a similar name was created in EMAN1. In August 2017, we added a new 2D class averaging program which solves this problem using some new mathematical methods, which are both faster, as well as appearing to produce better class-averages, but this program will only be available if you are using a snapshot version of EMAN2.2 from late August.

If you don't see the following option, you are using an outdated version of EMAN2. Check your e2version.py results and make sure you have at least EMAN2.21:
  - Select *2D Analysis→Bispectrum-based Class Averaging*
    - *Input* = **sets/all__ctf_flip_lp5.lst**
    - *Ncls* = **64**, nbasisfp = **8**
    - *classalign* = **rotate_translate_tree**, *params* = **flip=1**   (this should be the default)
    - *parallel* = **thread:4**  (or how ever many cores you have)
    - *classaverager* = **ctf.weight.autofilt**
    - The defaults should be fine for the other options, *Launch*


It is interesting to note that with the new bispectrum-based class averager, you may actually prefer the class-averages produced in the very first iteration (classes_00.hdf) vs those produced after 1 iterative round (classes_01.hdf). With the old e2refine2d.py, this definitely would not be the case. In the

➡ Unchecking normproj will tend to do a better job in separating junk from real particles, whereas keeping normproj will tend to produce more different views (of lower quality).

➡ If working with your own data, you should target a minimum of ~20 particles per class (on average) and a maximum of ~500 particles per class. If trying to process a data set with 1 million particles, you may not want to use this method to identify bad particles, and simply use this step on a ~10,000 particle subset to make class-averages for building an initial model. Unsupervised class-averages in EMAN are not used for high-resolution refinement, so bad particle identification is the only reason you would need to run this on your complete particle set. Note that the alternative strategy discussed below for identifying bad particles is largely automatic, and works very well, so you may not want to use this method for finding bad particles at all.

➡ For most projects, this is also an ideal point at which to look for structural heterogeneity in your data. If you see several class averages apparently in near-identical orientations, but with subtly different internal features, this may be a sign that your particle is moving in solution. There is an additional tutorial discussing approaches used to address this issue. Aside from ice contamination, the tutorial data set is highly homogeneous.

subsequent steps it is fine to use any of the class-average output files you like. Ostensibly later iterations will have more self-consistent classes, but for making initial models, it really makes little difference.

## 8. (optional, ~10 min) Eliminate some of the bad particles

_This step is optional_ - There is an additional fully automated strategy for eliminating bad particles, which we will use later in the tutorial. This step can be useful if your particle picker included many bad particles consisting of ice contamination or other non-particle artifacts.

Once class-averaging has finished, we need to take a critical look at the averages and see if we can identify any classes containing predominantly bad particles. The BGal data set used for the tutorial contains a large fraction of ice contamination "particles" which need to be eliminated at some point. To mark particles as bad, we will use e2evalparticles.py.

- 2D Analysis → Mark bad particles by class

- There are no options, just press Launch

- This will cause (initially) only one window to open:

- The window is divided into 3 sections:

  - The upper section shows a list of all class averages generated so far in the current project, both reference-free and (if you have run a 3-D refinement) reference based

  - The middle section permits you to modify which class-averages are currently selected for operation

  - The bottom section allows you to perform various operations on the particles associated with the selected class-averages. The only option we will use right now in this section is _Mark as Bad_.
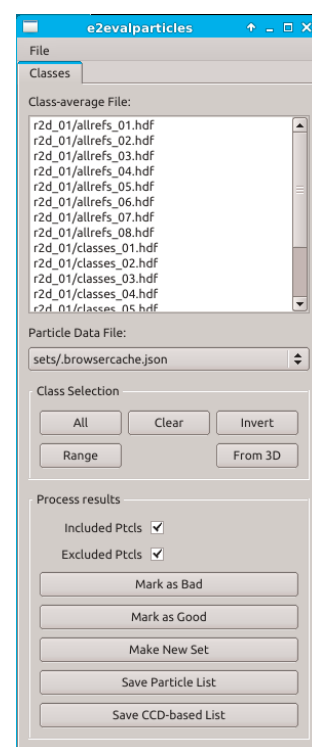
See: http://blake.bcm.edu/emanwiki/EMAN2/Programs/e2evalparticles for other uses.

- At present, we need to look at the results of our 2-D reference free classification. In the top section, select *r2db_01/classes_00.hdf*. While we could look at *classes_00.hdf* instead, the *allrefs* files contain the same images, but sorted and aligned to make it easier to compare the different averages visually.

- This will cause 3 additional windows to open. Two of these windows, titled: *Included Particles* and *Excluded Particles*, will be empty initially. The third will show the class-averages from the selected file. Single click on one of the class-averages, which will cause the windows to be populated, then position windows for convenient viewing. It would be good to see all of the class-averages at once (maybe with some scaling). The *Excluded Particles* window can be smaller.

- Note: There is a random element to the 2-D class-averaging process, so everyone will see a somewhat different set of class-averages.

- Single-click on any of the class-averages. After a short delay, you will see particles appear in the Included and Excluded particles windows. These correspond to the particles associated with this class-average. The Included particles were used (after alignment) to produce the actual average you selected. The Excluded particles are particles which were members of the class, but looked little enough like the final average that they were excluded from the averaging process.

- We now need to identify any class-averages which consist predominantly of "bad" particles, yet do this without eliminating too many "good" particles. Below you will see my class-averages, and the particles associated with average #9.
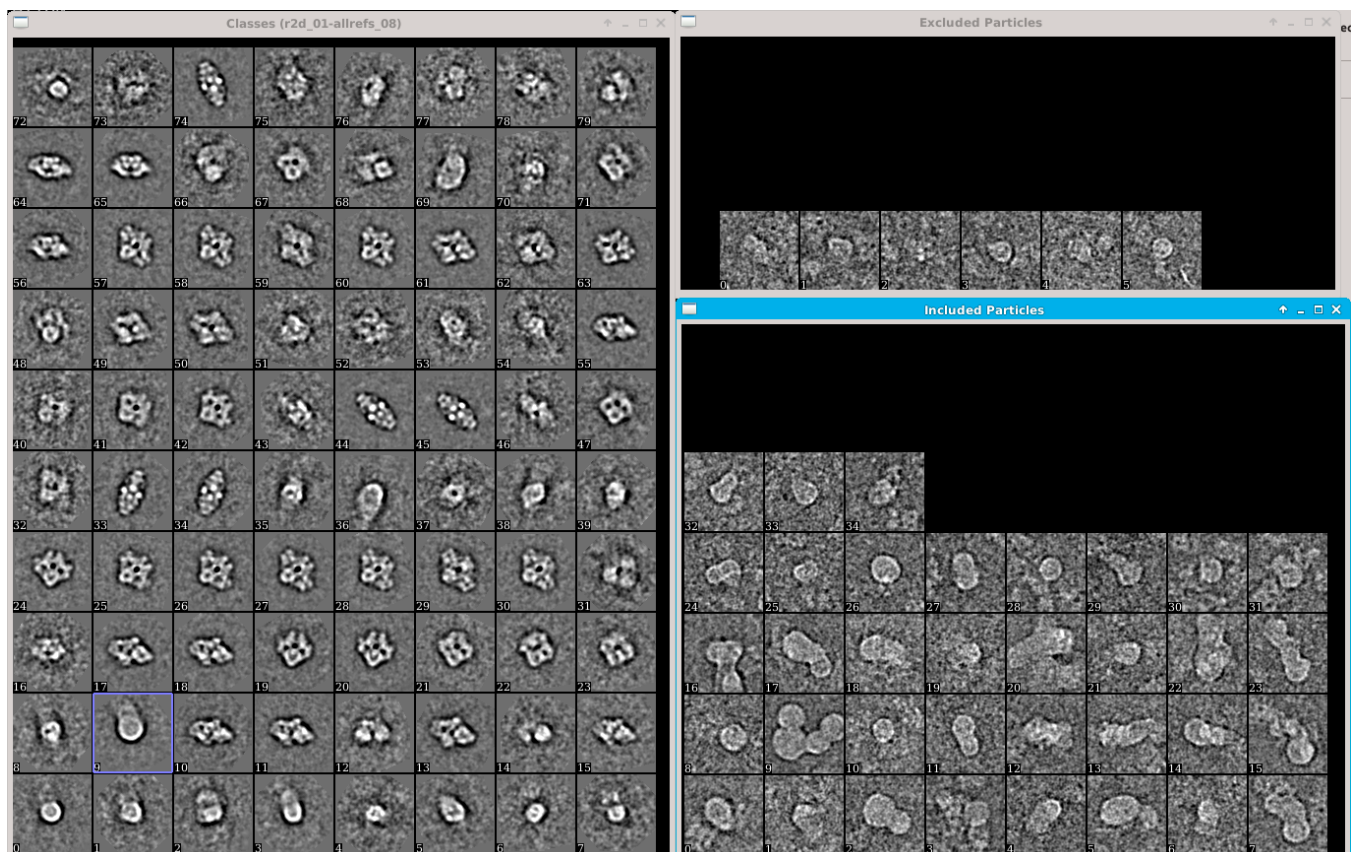
➡ This process requires judgement. While we would like to eliminate all of the bad particles, it is also possible to incorrectly identify some particular particle shape as bad particles, when it is really just an unusual orientation of the particle with a projection you weren't expecting. If you eliminate all (or most) of the particles in some view like this, you will wind up with a significantly worse reconstruction with an anisotropic resolution.

➡ It is often difficult for people to understand why we can't get rid of bad particles easily and automatically. Bad particles come in several different types. The type we are looking at above is quite obvious, and can be largely excluded automatically during processing. The far more difficult sort of bad particles to identify are those which are pure (or nearly pure) noise. The one clear characteristic of noise images is that they don't look like anything, even each other. That means they will tend to be randomly distributed through any set of class-averages. Even worse, the good particles with the best high resolution signal are often quite close to focus. Since this decreases low resolution contrast, eliminating low contrast particles has the side-effect of eliminating the very best data along with the junk.

➡ Generally speaking one bad particle like the ones shown above will do far more damage to a reconstruction than one good particle helps the reconstruction. In fact, particles like this can easily do more damage than 5 good particles would help the structure. So, if you have to sacrifice a fraction of your good particles to get rid of really bad particles, this is usually a worthwhile trade to make.

• Hopefully it is obvious that these "particles" (lower right window in figure) are actually just ice contamination. In most cases we won't have such a clean separation and the particles will consist of some bad and some good particles. We would now like to mark these particles as "bad" so they aren't used during our 3-D reconstruction. Now double-click on any class-averages you think are mostly bad particles, you will see a small blue mark appear in the corner of each. You cannot click on individual bad particles, only class-averages.

• Note that it is <u>not</u> possible to mark individual particles as bad using this interface. If you want to manually mark particles as bad, there is a section in the Appendix which explains how to do this. For now, mark all of the class-averages you wish to exclude with the little blue mark. It will not be the end of the world if you leave a few bad particles in the set. The goal is to eliminate 80-90% of the bad particles, to make our overall population healthier.

• Once you are satisfied with your blue marks, press the *Mark as Bad* button, and confirm that you want to do this in the dialog that appears. This will add all of the bad particles from all of the marked class-averages to an internal list of bad particles maintained for each micrograph. The next time you use the "build sets" interface, it will automatically exclude these particles from new sets you create. It is safe to repeat this process. Marking new particles as bad adds to the list, it doesn't remove existing bad particles from the list.

• You can exit the program (close the windows) when you're done.

### 9. (manual, ~5 min) Selecting Good Class-averages

There are several strategies you can use to produce initial models. One strategy (stochastic gradient descent) can actually work with raw particle sets, and doesn't even require class-

averages. The strategy outlined here uses the traditional initial model generator in EMAN2. Feel free to experiment with the others.

You will get the best results, generally, if you use all of the class averages, perhaps excluding some obviously bad ones. There are two general strategies for manually selecting a subset of images in a stack file and writing that subset to a new file. Either you can mark bad images for deletion, then save to a new file, or you can mark good particles, and save the marked good particles to a new file. With this strategy, our intent is to only mark a few of the 64 class-averages as bad, so we will use the first strategy. The second strategy is described in a box for your reference, but you don't need to use this method now.

- Browse to **r2db_01** (or whatever the highest number r2db folder you have is)
- Double click on **classes_sort_00.hdf** (the variant without _sort_ is also ok)
- Middle-click on the window showing the class-averages to open the control panel
- In the control panel press the *Del* button. This will put the left mouse button in delete mode.
- Click on any class-averages you are pretty certain are bad, ie - not a projection of B-gal, but something like ice contamination, or just a noisy average of junk. This will cover the offending class-average with a colored box, marking it for deletion. Note that this does not actually change the file you are viewing. This is just a mark on the display for now.
- Once you have marked all of the bad class-averages, press the Save button in the upper right corner of the control-panel, and enter "good_classes.hdf". The name doesn't matter, as long as you use the same name in the next step.

---

Sometimes you will need to extract a few specific images from a stack. Rather than using the above approach and deleting everything you want to exclude, if you only want a few images, it is more useful to mark the ones you wish to keep instead:

- Open the the stack in a multi image display window as above.
- Middle-click on the window showing the class-averages to bring up the control panel.
- In the control panel press the *Sets* button under the histogram plot, then also select the *Sets* tab (next to *Main*).
- Press the *New* button and type **good** in the box that appears. You should now have a green (or blue) colored set with this title. Click on the colored word *good* to make it active.
- Now single-click on each image you wish to extract. You will see a colored mark appear. Click a second time if you change your mind and wish to exclude the image.
- When done, press the *Save* button in the lower right corner of the control panel (not the one in the upper right). This will save the selected images in the highlighted category to a specified file.

---

**10. (~10 - 30 min) Making an initial model**

There is a lot of controversy in the cryoEM community on this point. Some people feel that initial model generation is the most critical step in refinement, and that you need to use difficult and time-consuming experimental methods to get an initial model before you can proceed. We have shown that the simple approach used in EMAN2 can give a completely reliable initial model with no additional experiments required in the vast majority of cases.

- Despite all of the caveats and descriptions, the actual initial model generation program is quite easy to use. Just select: *Initial Model→Make Initial Model*
- *Input* = **good_classes.hdf**, *sym* = **D2**, *iterations* = **12**, *tries* = **12**,
- *randorient* **unchecked,** *shrink* = **3** (if there are problems you can try shrink=2)
- Since we are under time constraints at the workshop, I suggest starting with 8 or 12 *tries*, then if you don't get a good one, run another 12. If you have a little more time, or more than 4 cores, you might want to start with 20 or 30 to make it more likely you will get a good result on your first try.
- *parallel*= **thread:4** (again, replace 4 with the actual number of cores on your machine)
- Use the default values for the other options, and *Launch*

---

➡ This is an updated version of the program in EMAN2.3. Rather than purely starting with random blobs, the first 7 tries will start with predefined geometries based on the specified symmetry. Any additional starting models beyond 7 tries will be randomized. If you check *randorient*, starting models are generated from the class-averages in random orientations. With the new version we suggest leaving randorient unchecked unless you have difficulties.

➡ For most structures, there are a number of 'local minima' in the energy space. What that means is, there are a number of incorrect structures which can still appear to agree fairly well (but not as well as the correct structure) with the input data. So, some fraction of the answers you get out will be bad starting models. Such bad starting models are usually quite obvious. The severity of this problem varies considerably with the shape of the molecule and the amount of orientation coverage you have. Interestingly, particles like ribosomes, generally viewed as 'difficult' have virtually no local minima, and will produce a usable starting model almost all of the time.

➡ If your best initial model comes out without a pretty clear handedness, you may want to try again.

➡ Heterogeneity is a big potential issue in most projects. If you have a particle that is highly heterogeneous, the EMAN initial model strategy will produce a model, but it will not be unique (since there isn't one). Single particle tomography may offer the best solution towards understanding the heterogeneity in your specimen. Once you understand the heterogeneity, there is a separate tutorial on the Wiki for dealing with such cases.

➡ If your particles have a single, strongly preferred orientation, especially if this is combined with a low symmetry, mathematically there may not be enough information to produce an unambiguous starting model. However, it is also important to note that in this situation, even if you get a good starting model, refinement will also tend to degrade rather than improve the model. To perform a proper 3-D reconstruction, you must have a reasonable number of particles in orientations covering at least one *great circle* around the unit sphere.

➡ If you do have a difficult structure, single particle tomography is our recommended solution ( http://eman2.org/e2TomoSmall )

For high symmetry objects like virus particles, there is another program e2initialmodel_hisym.py which does a better job on high symmetry objects. This program is best run with just a few (3-5) class-averages, and with heavy downsampling.
There is also an experimental new initial model generator in examples/ called initmodel_sgd.py. This program uses stochastic methods so is best run with a fairly large number of class averages (or even raw particles). In many cases it performs exceptionally well with the provided default parameters, but not always.

You are also absolutely free to use initial models generated in any other software you like. As long as the A/pix value in the header of the initial model is set to the correct value, you can use it directly with e2refine_easy as an initial model, and it will automatically rescale/clip as necessary to make it work. You can double-check the A/pix value in the header using either e2iminfo.py -H or with the file browser. If the header value is wrong, you can use e2proc3d.py with the —apix option to correct it.

- When the program is done, open the browser (🗁). You will see a new folder called *initial_models*. The program saves all of the initial models it generated, not just the best one. For each trial, 4 files will be produced in the initial_models folder. We will look at three of them in the tutorial: *model_NN_MM, model_NN_MM_proj* and *model_NN_MM_aptcl*. The first time you run e2initialmodel, NN will be 00, and MM will indicate the number of the attempt. If you need to run the program again, new results will have NN = 01.

- model_00_01.hdf ostensibly contains the best 3-D refined initial model, but looking at the 3-D map is not the best way to detect whether it is bad or good. Instead, start by looking at model_00_01_aptcl.hdf. The image below shows two possible results, the left is good, and the right is bad:

- Each of these files contains adjacent pairs of images. The first image is one of the class-averages you provided and the second image is a projection of the initial model it created, and so on in alternating fashion. If you have a good initial model, the class-averages and projections should agree with each other very well (aside from noise).
  - In the left (good) case above, you will see that this is the case every class is a good qualitative match to the corresponding projection, with the possible exception of 0-1.
  - In the right (bad) case, while several of the pairs agree well, such as 38-39, many of them do not. In a good initial model, agreement should be good for ALL of the class averages.
- If one or more do not match, there are 3 possibilities: 1) a bad initial model, 2) a bad class-average, 3) heterogeneity in the particle population. In the left result, the first image (and perhaps the third) is likely just not a very good class average, as other than this, agreement is excellent. Having seen the left case, we know that the right case is just a (random) bad initial model.
- You should also look at model_00_01_proj.hdf. This file contains a sequential set of projections of the final model from all orientations. The goal here is to make sure that all of the projections look "good". ie - like projections of a rational protein. Most bad initial models will be quite distorted looking in some orientations.
- If your best model doesn't agree well, you should try running the program again. If you persistently fail to get completely self-consistent results, the conclusion would be that you have some structural heterogeneity in your particles. That situation is covered by the other tutorial on the wiki. This B-gal data set should not have significant heterogeneity other than the particles which are actually ice contamination.
- Once you identify a starting model with a good _aptcl file, go ahead and open the corresponding model_NN_MM.hdf file in 3-D and have a look at it. This will be the starting model we use for refinement. Note that the <u>handedness of this model will be arbitrary</u>. There is a 50/50 chance that it is correct. We deal with that after refinement.

**11. (~1 min) Building sets - again**
*Note - if you skipped step 8 above do this step anyway. In addition to excluding bad particles, we are also making a reduced set for faster refinement.*

Above we went through some manual effort to identify bad particles (unless you skipped that step). However, so far all we have done is put a mark on the bad particles. At present they are all still included in the sets/*.lst files which we use for our refinements. This is done intentionally, so you can retain the original set of particles, in case you want to go back and check what effect eliminating the bad particles had.  We now need to process the data without these bad particles, so we need to build new sets.

- *Particle sets→ build particle sets*
  - check the *allparticles* and *excludebad* checkboxes
  - enter **all-bad1** int the *setname* box, this name indicates to me that I have made one pass at removing bad particles, but otherwise the set includes everything. If I made another pass and excluded even more bad particles, I would call it **all-bad2**, but this naming convention is arbitrary.
  - *Launch*

We'll make one more set containing less of the data for our initial refinement (for speed)

- *Particle sets→ build particle sets*
  - Uncheck the *allparticles* box
  - Press the *Browse* button. Click on the *snr-hi* column to sort, and select the first 20-25 images (with the highest *snr-hi* values), then click *OK*.
  - enter **best-bad1** in the *setname* box.
  - *Launch*

**12. (~25 min) 3D Refinement**

We are now ready for our first quick refinement, to turn our initial model into something with a few more features. If the initial model is decent, it might be possible to jump straight to a 4 Å map at this point, but it is usually a good idea to take a couple of steps on the way there to make sure things are coming out as expected. As we push resolution using the traditional refinement algorithm, the compute time required increases very rapidly:

- 1 CPU-h : Quick (~15 Å resolution) refinement using ~1000 _lp12 downsampled particles
- 2 CPU-h : Quick (~15 Å resolution) refinement using all ~5000 _lp12 particles
- 12 CPU-h : (~6 Å resolution) refinement using all of the _lp5 particles
- 70 CPU-h : (~4.0 Å resolution) refinement using all of the __fullres particles
- 168 CPU-h : (~3.6 Å resolution) refinement using all of the __fullres particles

*Note - times above are CPU-hours, so to get actual estimated wall-clock time, divide by the number of cores on your computer. This estimate becomes less accurate for large numbers of cores (like 128 or 256), but is fine for a desktop machine.*

However, it is worth noting that the "--invar" option now available for refinement can decrease these times quite dramatically. While it does a very good job for preliminary refinements, at this point in its development we don't yet recommend it for the final stage refinement when pushing resolution. For the earlier rounds of 3D refinement, checking the 'invar' checkbox is generally quite beneficial.

The 15 minute refinement results are usually too coarse for accurate bad particle removal, but it should show if there is some fundamental problem in the processing leading up to this point, so we will begin with that, then move to the 3-hour (if 'invar' is not used) refinement to identify bad particles, and finally finish up with the 70 hour refinement, which will run overnight. We won't have time for the 168 hour refinement at the workshop. Note also that if you have access to a linux cluster, using a very modest 128 cores, you could get your 3.6 Å map in under 2 hours.

The program we will use for our first refinement is e2refine_easy. As its name implies, it is easy to use, with very few required parameters. It has an advanced system of heuristics to automatically select the best options for your refinement. These are <u>not</u> just default values. The program considers the information it has about your project combined with the target resolution and other parameters you give it to select what it believes will be the best options.

Its choices, and some of the reasoning for them, are all documented in a report file generated during the refinement. If something goes wrong, you can then override <u>any</u> of these options from the command-line as required.

Now do:
- *3D refinement → Single map refinement (e2refine_easy).*

➡ scaling is not perfectly linear with number of processors, and is worse for small projects like this one than it is for large projects. If you have 100,000 particles, you can get roughly linear scaling for a larger number of cores.

➡ e2refine_easy computes a 'gold standard' resolution as part of the refinement process, for each iteration of the refinement algorithm, and uses this information with to near-optimally CTF correct and filter the final 3-D maps. The 'gold standard' does not eliminate noise bias, but it does prevent noise bias from entering the resolution calculation, which in turn influences how the map is filtered.

➡ You'll note that the apix (Å/pixel) parameter is set to 0, not its actual value. This prompts the program to read the value from the header of the particles being used for the reconstruction. That way if the data is downsampled (as it is in this case) the *apix* is always properly adjusted. In addition, *e2refine_easy* will automatically rescale and resize the starting model based on the *apix* values it finds in the two files.

➡ The *speed* parameter is the primary control the user has over the refinement process (unless you want to override all of the detailed options). A speed of 1 strongly weights the heuristics towards a high quality reconstruction at optimal resolution, but pays little attention to how long it will take. A speed of 7 (the maximum) will run very quickly, but has only barely sufficient sampling for the target resolution. 5, the default should give a very solid reconstruction reasonably efficiently. Normal usage would be to use 5 for most refinements, then once you've done the very best you can with 5, try something in the 1-3 range to see if it improves your map quality or resolution.

➡ Mousing over individual parameters will give help on each, though on some OS versions the default font coloring makes these difficult to read. If you want more information on each option, run e2refine_easy.py --help from the command-line, and it will give full details on all of the options.

➡ The project manager simply uses the parameters you enter to run a command-line program for you. You could achieve exactly the same results by running the resulting command yourself. If you click on the *command* tab when entering parameters for a program it will show you the exact command that will be launched by the GUI. You can add additional options to the end of this command before pressing *Launch*. Just be warned that if you do this and switch back to the *GUI* tab and additions you made which are not shown in the property list will be lost.

➡ Clearly we won't be running any jobs on Linux clusters at the workshop, but if you continue to do single particle analysis, eventually it is almost inevitable that you will need to do this. The final refinement below with speed=1 can take ~150 CPU-h to finish. With the full data set that has 10x as many particles and even finer sampling, we'd be talking hundreds of CPU-h. Even if you had a powerful 24 core workstation, this is still more than a day to finish the job. On a cluster you could bring the time down to a couple of hours. Note that all of this extra data and computation would be to improve from ~3.6 A we can achieve with the tutorial data to the published ~3.2 A resolution with the full data. There is fairly extensive documentation on the topic in the Wiki (http://blake.bcm.edu/emanwiki/EMAN2/Parallel) where you can find many of the necessary details. If you have problems, don't be shy about asking!
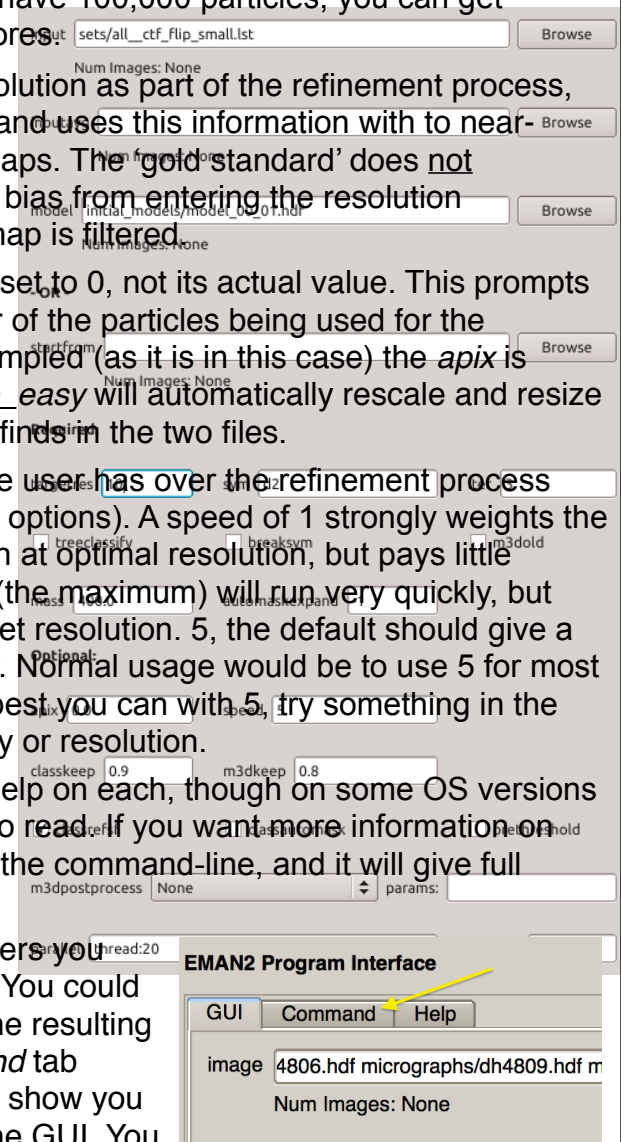
- *input →* **sets/best-bad1__ctf_flip_lp12.lst**
- *inputavg →* leave empty
- *model →* select your good starting model (model_00_01.hdf or similar)
- *targetres→***15**, *sym→***d2**, *iter→***2**, *mass→***400**, *speed→*5**,** *apix→***0,** *ampcorrect→***auto,** *tophat→*(empty text box)
- *parallel→***thread:4**  and *threads→***4** (or appropriate numbers)
- *Launch*
- e2refine_easy creates an HTML (web page) report as it runs explaining the various decisions it makes and the parameters it used, as well as summarizing the results. Browse into the refine_01/report folder and select index.html. You can press the "Firefox" button (or just browse to the file in your web-browser if that doesn't work on your machine).
- The report is updated continuously, including resolution and convergence plots. You will need to reload the index.html page to display the new content.
- With ~1000 particles on the quad-core desktop computers, this should take about 15 minutes to complete.

**Refinement 2:**
- The initial refinement should not take very long to run. As soon as it finishes, do a quick check on **refine_01/threed_02.hdf** with the browser to make sure it  looks reasonable (ie - something like a low resolution B-Gal). You may also take a look at the report/index.html to make sure your convergence and resolution curves are doing what you would expect (the web page explains this, at least somewhat).
- If there are no problems, go ahead and start the second refinement running. Make the following changes to the last run:
  - *input →* **sets/all-bad1__ctf_flip_lp5.lst**
  - *model →* **refine_01/threed_02.hdf**
  - *tophat →* **global**    (type this word in the box)
  - *targetres→***6**, *iter→***4**, *speed→***5,**
  - then **Launch**

- The tophat option above is new in EMAN2.2. When you run the postprocessing program in Relion, it changes the Wiener filter it normally uses during refinement into a sharp cutoff filter (also known as a "tophat function". This filter makes fine details stand out more strongly (like side-chains and the pitch of alpha helices), but also produces some high resolution artifacts. This filter is the reason why in the past people often thought Relion maps looked "better" than maps of the same resolution from a lot of other software. EMAN now provides the "tophat" option in e2refine_easy to do basically the same thing.
- There are three accepted values for "tophat" in e2refine_easy. " " (empty text box), "global" and "local".
  - *tophat*=(empty), the tophat option will not be used, and the final filter will be just like EMAN2.12. This is probably a good idea for structures worse than ~10 Å resolution.
  - *tophat*=**global** does basically the same thing Relion does in post-processing, and sharpens your map to make it look prettier (but watch the artifacts).
  - *tophat*=**local** is a new option unique (I believe) to EMAN2. This option will compute a local resolution map for your refined structure, then use this resolution map to apply a local filter to each region of the structure. This means that regions which may be dynamic will be filtered to be blurrier in the final map, and regions which are very stable will be sharpened appropriately. For some structures, this can have a dramatic positive effect, and actually refine stable regions to a higher resolution, by iteratively downweighting the inconsistent regions. It is still experimental, but it has proven very useful on many structures containing flexible domains.
- Note that EMAN is not using the popular ResMap program to compute local resolution, but a "local FSC". The results can be visualized in a similar way, but the property being computed is significantly different. The program e2fsc.py can be used to compute these ResMap-like volumes, given an unfiltered even and odd input map, and can also apply local filtration. It is also possible to run ResMap from within EMAN for comparative purposes.

- Now that you have a real refinement running, let's go over what all of these options are and what they mean (you don't have to remember all of this):
  - *input* - this is simply the set (.lst file) containing the phase-flipped particles to be refined.
  - *inputavg* - if specified, this permits you to use a different stack of particle images for reconstruction and alignment. input is used for alignment, inputavg is used for the final class-averages and 3-D reconstruction. This is used in cases such as direct-detector movie mode processing when you want to use high-dose particles to determine orientation, but low dose (minimal radiation damage) particles for the final map.
  - *model* - Perhaps an unfortunate historical term. We refer to the 3-D map used to seed the iterative refinement as the initial model. Model is also often used to describe the PDB model derived from a high resolution map. That is not the meaning here. This file must be a 3-D map file with a valid A/pix value in the header. You do not need to resize/rescale the map. This is handled automatically.
  - *startfrom* - this can be used <u>instead</u> of all 3 parameters above. If you have already completed a refinement run, and wish to continue without changing the input particle set, but perhaps wish to slightly change other parameters, specify the name of an existing refine_XX folder, and it will pick up where that refinement left off. It is not possible to change the input particle stack with this option, as that would violate the assumptions for the 'gold standard' method.

- *targetres* - This is the resolution (in Å) you are trying to achieve in this particular refinement run. If you select a much higher resolution than you can actually achieve, it will make the refinement take much longer, and in extreme cases may even make the refinement quality somewhat worse (not typical). If you set the resolution much lower than the provided data can achieve, it will not sample sufficiently, and you may get some odd resolution curves with apparently exaggerated resolution. If your refinement ever achieves a resolution better than your targetres, you MUST rerun the refinement with a higher resolution target to have a valid result.
- *sym* - the enforced symmetry of your structure. Your initial model must be in a symmetry-aligned orientation matching the specified symmetry here. See: http://blake.bcm.edu/emanwiki/EMAN2/Symmetry
- *iter* - Number of refinement iterations to run. Linearly related to time required for the job to run. With a decent starting model, most EMAN2 jobs will converge after ~3 iterations, and we often run 4 just to make sure we have achieved pseudoconvergence. If you are starting with a very poor starting model, such as trying to refine a ribosome from a featureless ellipsoid, you may need to run 10-12 iterations to converge to something sensible, much like the initial model generator.
- *tophat* - explained in the box on the previous page. Important new option!
- *ampcorrect* - There are different strategies for CTF amplitude correction. For maps with resolution worse than 6-7 Å, we use the 1-D structure factor we automatically fit when doing CTF fitting above. For very high resolution maps, however, a B-factor correction+Wiener filter strategy (this is similar to what Relion and Frealign typically do) works a bit better. "auto" is usually the best option to select here.
- *treeclassify* - don't use this at present. It is an experimental new option.
- *breaksym* - A powerful option for studying pseudosymmetric objects, such as icosahedral viruses with portal complexes or multimeric proteins with highly homologous subunits. This will determine the orientation of each particle first with the stated symmetry. It will then search that orientation in each of the N asymmetric units for the best match. Converges much more quickly if a symmetry broken starting model is provided, but after enough iterations you can even start with a symmetrized starting map. This is discussed in more depth in the separate heterogeneity tutorial.
- *m3dold* - Not normally used. Reverts the 3-D reconstruction to an older version of the code. If you experience any strange artifacts in your 3-D reconstruction you could try turning this on. If it helps (it shouldn't), please report the situation to sludtke@bcm.edu.
- *mass* - in kDa. However, that isn't really what this parameter means. The goal is to set 1.0 as a reasonable isosurface threshold for the map. Historically this has been done by assuming a mean protein density of 1.35, and using the mass to compute a volume which should be enclosed by an isosurface. However, if you were to visualize a 4 Å resolution map using this method, you would see a solid density in space. An isosurface where the backbone and sidechains are visible is mostly open space, and has perhaps 1/2 the volume you would normally assume. So in a project targeting high resolution this mass value should normally be significantly less than the true mass of the target molecule.
- *automaskexpand* - A 'solvent flattening' procedure is a standard part of most single particle refinement procedures. That is, a mask is constructed just outside what is believed to be the protein density, the mask is softened with a gaussian falloff to the mean solvent level. This improves both refinements as well as resolution values by eliminating pure noise. However, if you have a ligand with weak fractional association, or a dynamic component on the exterior of the molecule it can create a weak density just outside the

main molecule, which may get sliced off by this automatic masking. This parameter allows you to specify an additional number of 1 voxel shells to expand the mask beyond the automatically determined size.

- *classkeep* - the fraction of particles in each orientation class which should be included in the class-average. When making class-averages, the particles are combined, then each is compared against the just-created average. If this value is, for example 0.9, then the 10% of the particles that look least like the average will be excluded from the class-average.
- *m3dkeep* - similar to *classkeep*, but the fraction of class-averages to include when building the 3-D reconstruction. Classes are already automatically weighted based on number of particles or SSNR, but if you have some class-averages in underpopulated orientations which are really simply bad, this allows you to exclude them from the reconstruction.
- *classrefsf* - causes the class-averages to be filtered to have a similar structure factor as the projections. This doesn't actually influence the information content in the class-average, but makes them easier to visually compare with the projections.
- *classautomask* - an experimental option for performing 2-D masking of the class-averages during iterative class-averaging.
- *prethreshold* - An experimental option for applying a threshold to the 3-D map before using it as a reference for the next round of refinement.
- *m3dpostprocess* - allows you to specify a manual processor to be applied to the 3-D map at the end of each iteration. Normally not necessary as the maps are automatically filtered.

## 13. (~5 min) Eliminate bad particles
*This is a new addition to EMAN 2.2. It is an alternative/supplement to the bad particle identification strategy above in step 8.*

You will need to wait for the step 12 refinement to finish before doing this step.

Can we eliminate bad particles by simply comparing each particle to the corresponding projection or class average, and throw out the ones which agree poorly?  Yes and no. This idea is, in fact, already used during refinement to eliminate bad particle outliers within each class. Unfortunately, when used aggressively, this method also eliminates the very best high resolution particles (close to focus).

What we are doing in this step is very similar. Instead of computing a single similarity metric between each particle and each projection, which has poor separation, the comparison is made over 4 resolution ranges, and then we combine 2-4 of these values to make the decision about which particles to keep. We are looking for particles which are consistently good at multiple resolutions

- *Validation and Analysis* → *Eval particle Qual*
  - *refine_XX*→your highest numbered refine folder
  - both checkboxes should be selected. includeprojs is for visualization only, and should not normally be used when sets with very large numbers of particles are considered, but for the tutorial it will be useful.
  - **Launch**
- It should take <5 m for this to run. You can monitor it as usual with the process monitor.
- When this program is finished, it will have created four new sets:
- sets/pf02_good_all__ctf_flip_lp5.lst
- sets/pf02_bad_all__ctf_flip_lp5.lst
- sets/pf02_good_all__ctf_flip_invar.lst
- sets/pf02_bad_all__ctf_flip_invar.lst
- This will be the program's attempt to automatically split your particles into good and bad particle subsets. You can simply trust this result, or you can go through the more detailed instructions below to manually assess your data.

You can skip the below section if you like and move on to step 14. It explains how the automatic separation above works, and gives you the opportunity to assess whether the automatic separation is likely to be good, and possibly to do a better job:

- You will also have two new files: **ptclfsc_XX.txt** and **ptclfsc_XX_projections.hdf**. The .hdf file is not useful on its own, it is referenced when visualizing the .txt file.
- Open the browser and double-click on **ptclfsc_XX.txt**.  This will open a window containing a scatter plot like this:
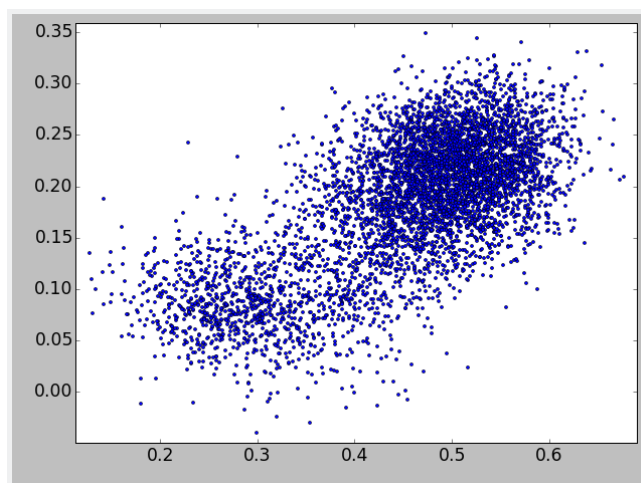
- Each point in this plot represents one particle in the data set. The X axis of this plot is the particle agreement (larger better) from 30-100 Å and the Y axis is the agreement from 15-30 Å. Note the characteristic 2 lobed distribution.
- If you left-drag with the mouse over these data points, in addition to identifying the points in the plot, it will also open 2 image display windows showing the particle and the corresponding (aligned) map projection.
- For the workshop data, you should see a distribution very much like the image above. If you don't see something like this, with a pretty clear separation, you may not have a good structure. In this situation, the most likely problem is that you started with a bad initial model, but there may be a few other possibilities as well. Best to ask an expert if it isn't obvious.
- If working with your own data, and the distribution appears horizontal rather than sloped, or if there really isn't any separation between the "lobes", this generally indicates very noisy data, either due to the particles being very small, or the ice being significantly too thick. If the particles are >200kDa and you see this problem, you may need to investigate your grid preparation. If it is a small particle, it may simply not be possible to accurately distinguish between "good" and "bad" particles.
- You will see the particles appear very noisy when doing this. If you open the control-panel for the noisy particle display, then go to the *Filt* tab and check all 3 checkboxes you see there, the particle will be filtered for display and it should be much easier to visually assess the difference between the particle and projection. This interactive visualization is not required to extract the good subset of the particles, but allows you to look at a few of the worst particles to better understand what about them makes them so bad.
- By default the plot widget displays the first 2 columns present in a text file, but this file actually contains 8 columns:
  - The first 4 columns of this file are the quality of the particle over different resolution ranges: 30-100A, 15-30A, 8-15A and 4-8A.
  - alt, az Euler angles in degrees
  - class number
  - defocus in um
  - There is also a comment after # on each line identifying the particle associated with each line in the file.
- If you middle-click on the plot, as usual, a control panel will appear. In the control panel you will see two widgets labelled: *X Col* and *Y col*. These control which columns are used for the X and Y axes respectively (numbering starts with 0). Try changing *Y Col* to 2. You should see a slightly different plot, but with a similar shape, implying we are able to discriminate individual particle quality at 8-15 Å as well.
- Now set *Y Col* to **3**. You should see that the distribution now is roughly symmetric about Y=0, with only a slight positive bias. This says that our information at 4-8 Å is not strong enough to act as a good discriminator of particle quality. This is typical for even the best data sets. For some data sets even the 8-15 Å band will not be useable. Set *Y Col* back to **1**.
- We now wish to separate the "good" particles from this distribution.
  - Press the *Classification* button in the control panel, causing a new window to appear.
  - In the K-means section of the new window, *Nseg*→**2** and *Axes*→**0,1,2**
  - Check the *Eq Wt Axes* box, then press *K-means*
- You should see 2 new plots appear in the control-panel list. Since the points are at exactly the same location as the original points, we can't see them in the plot yet. Uncheck the original curve (blue), and you should now see the green and red subgroups of the data emerge. It should show a fairly clear separation between the groups.

- Remember, the points in the upper right of the plot are the "good" particles. We now wish to extract these to a new **sets/XXX.lst** file. To do this, in the plot control panel, click on whichever set (red or green) is in the upper right.
- With this one subset selected, in the *Classification* window, enter **goodqual** in the Prefix text box, then press the adjacent *New Sets* button. A window should appear saying "New sets created: sets/goodqual_X.txt"
- You can now close the plot window. In the browser, you should now see that **sets/goodqual_0.lst** now contains a subset of the original particles which should have excluded about 20% of the whole data set.
- While this certainly did not do a perfect job of eliminating bad particles, it certainly made a major step in that direction. You can repeat this process after running an additional refinement with this subset, and should be able to get rid of a few more bad particles, but not nearly as many as in this initial attempt.


**14. (20+ h, depending on options and ambition) Final refinement**

If everything went well, you should now have a refinement which has achieved at least ~6-8 Å resolution. If the resolution curve has gone beyond Nyquist due to the high quality of the data, then you can't completely trust the resolution value (ie - the resolution value can only be trusted to a bit past the target resolution you specify), and you should run another refinement with better sampled data.

You now need to decide how ambitious you want to be, based on the amount of time and the computer you have available (and how much you care about the final resolution of this learning exercise). Refer again to the list of anticipated timings at the beginning of step 12 above to consider how many CPU-hours you think you can afford, and adjust appropriately.

So far we have been using the _lp5 data sets. These particles have been downsampled and low pass filtered so they are only useful to ~5 Å resolution, and at 5 Å will have only marginal detail. Switching to the full resolution data will make the refinement take significantly (probably ~5x) longer to run, but is the only way to get to 4 (or 3.6 Å).

In each of these cases, you will be running a new e2refine_easy run.

**Option 1 (fastest, worst resolution)**
If you want to use the _lp5 data and just make sure you get the best resolution you can from that data:

- You will be using the "good" set you prepared in step 13 above as the *input* for this new refinement. If you are just using the automatically generated results, this file will be **sets/pf02_good_all__ctf_flip_lp5.lst** . If you followed the manual instructions, you will have named this file yourself. If the refinement folder wasn't **refine_02**, the **02** may also be different. If you are using the actual 2.22 release, it is possible that the name will be a little different, but you should be able to figure out the correct file.
- *targetres* → **4.5,** *speed* → **5**
- *model* →**refine_xx/threed_yy.hdf**   (fill in xx and yy with the highest numbers available)
- *iter* →**3**

## Option 2 (bit slower, ~4 Å resolution)
Use the full resolution data, but with less aggressive options:

Before we can start the refinement, we have to deal with an issue with the
**pf02_good_all__ctf_flip_lp5.lst** file we just created. That file was derived from a refinement
made with _lp5 data, and we now wish to work with the _fullres data. Unlike the other sets we
have built, the method above doesn't produce a set for all of the different file types for us, so,
exit the project manager, and from the command-line (you may have to change "02" to another
number depending on how many refinements you have run, and the name may be a bit
different depending on the exact EMAN2 version you have):
- **cp sets/pf02_good_all__ctf_flip_lp5.lst sets/pf02_good_all__ctf_flip_fullres.lst**
- **e2proclst.py s sets/pf02_good_all__ctf_flip_fullres.lst --retype ctf_flip_fullres** (this will
  retain all of the same particle numbers in the .lst file but change the particle filenames to be
  __ctf_flip_fullres within this file)

- input → sets/ptclfsc_XX_good__ctf_flip_fullres.lst
- *targetres* → **4,** *speed* → **5**
- *model* →**refine_xx/threed_yy.hdf**   (fill in xx and yy with the highest numbers available)
- *iter* →**4**    (may be ok with only 3)

## Option 3 (slowest, ~3.6 Å resolution)
Use the full resolution data, targeting the best possible structure. Note that this would
appropriate to run on a workstation with many cores, or even a linux cluster. The wiki has
instructions for using EMAN2 on clusters:

Before we can start the refinement, we have to deal with an issue with the
**pf02_good_all__ctf_flip_lp5.lst** file we just created. That file was derived from a refinement
made with _lp5 data, and we now wish to work with the _fullres data. Unlike the other sets we
have built, the method above doesn't produce a set for all of the different file types for us, so,
exit the project manager, and from the command-line (you may have to change "02" to another
number depending on how many refinements you have run, and the name may be a bit
different depending on the exact EMAN2 version you have):
- **cp sets/pf02_good_all__ctf_flip_lp5.lst sets/pf02_good_all__ctf_flip_fullres.lst**
- **e2proclst.py s sets/pf02_good_all__ctf_flip_fullres.lst --retype ctf_flip_fullres** (this will
  retain all of the same particle numbers in the .lst file but change the particle filenames to be
  __ctf_flip_fullres within this file)

- input → sets/ptclfsc_XX_good__ctf_flip_fullres.lst
- *targetres* → **3.2,** *speed* → **1**
- *model* →**refine_xx/threed_yy.hdf**   (fill in xx and yy with the highest numbers available)
- *iter* →**4**

**When you've picked one of the options above and filled in the parameters, go ahead and
launch the full refinement job.**

➡ Note that setting startfrom to an existing refine_xx folder is NOT equivalent to setting
    *model* to the final map from the same refine_xx folder. If you specify a single input model, it
    will be phase-randomized to relatively low resolution as starting models for the even/odd

half refinements (that is, you take a step backwards if you do this). If you use startfrom, the existing even/odd references are used, and the refinement can progress naturally, with no model bias, <u>but</u> you are not able to change the input data set.

## 15. Looking at your results

First, look at the refinement report in refine_XX/report/index.html. You can open this file in a web browser directly, or you can use the EMAN2 browser with the 'info' button (which won't show plots) or press 'Firefox' if you have that browser installed.

If your resolution curves are not "healthy". That is, they have strange peaks, don't fall all the way to zero, or have other artifacts, then some sort of bias still managed to creep its way into your project. If this happens, I'd suggest asking me about it. Sometimes you will see this in the tightly masked curve, but the curve with the normal mask should be fine. This simply indicates that the automatic tight mask was a little too aggressive, and you should ignore that curve. For the tutorial project, with a speed=5 refinement, the curve should look something like:
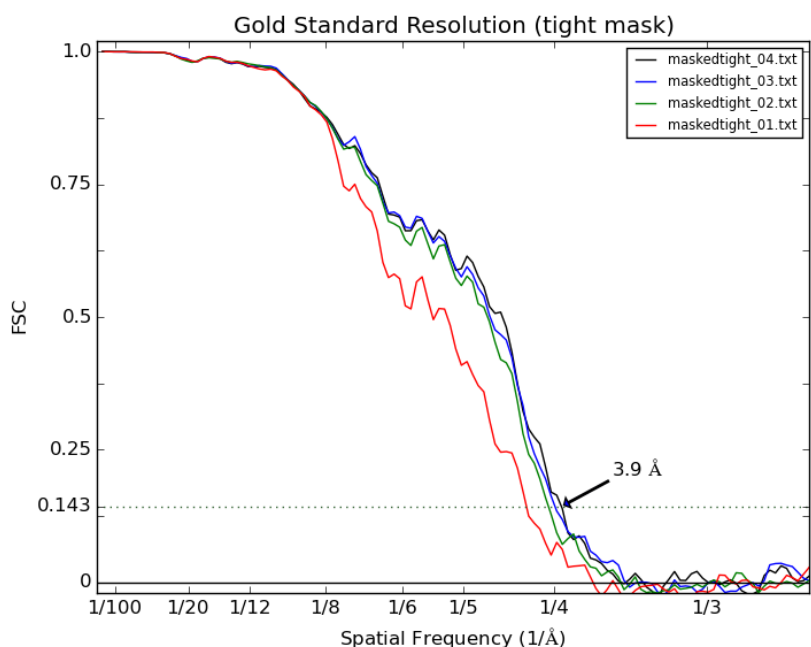


Next, look at the final refined map (refine_03/threed_XX.hdf).
 • You can open the map in the EMAN browser and look at it directly
 • You can load it into Chimera, and also open PDB 3j7h.
    • If the PDB vs map alignment is off, you can use Chimera's "fit in map" tool
    • If you still have a poor match, the handedness of the map is probably inverted. There is no way to determine handedness in a CryoEM experiment without doing a tilt experiment, so the handedness (regardless of software) has a 50/50 chance of being wrong. If you need to fix it, there are 2 easy ways to do it in EMAN2
       • Use the EMAN2 browser, and browse to the map, then press "filtertool".
          • configure a processor: *xform* in the left box, *flip* in the right box. Enter "z" in the axis box. Check the checkbox near *xform*.
          • Use the menu to *save processed map*.
       • Alternatively, from the command-line, use a command  like:
          • **e2proc3d.py refine_03/threed_04.hdf processed_map.hdf --process xform.flip:axis=z**
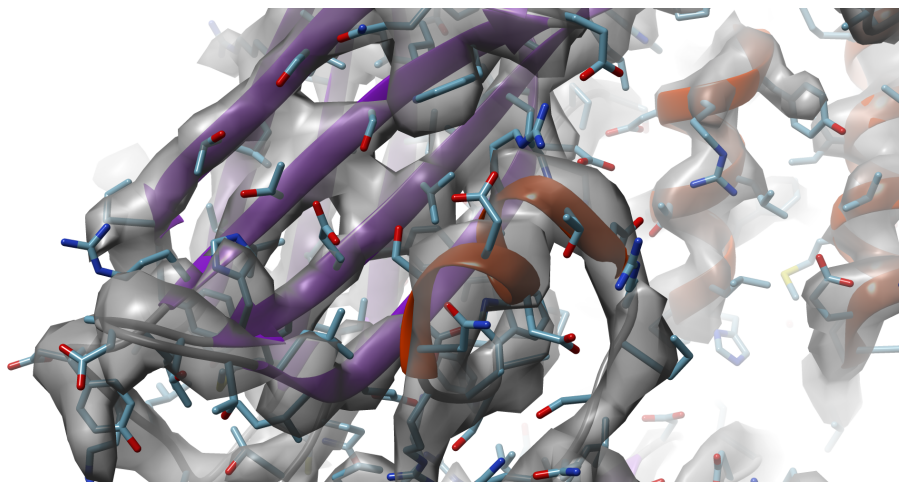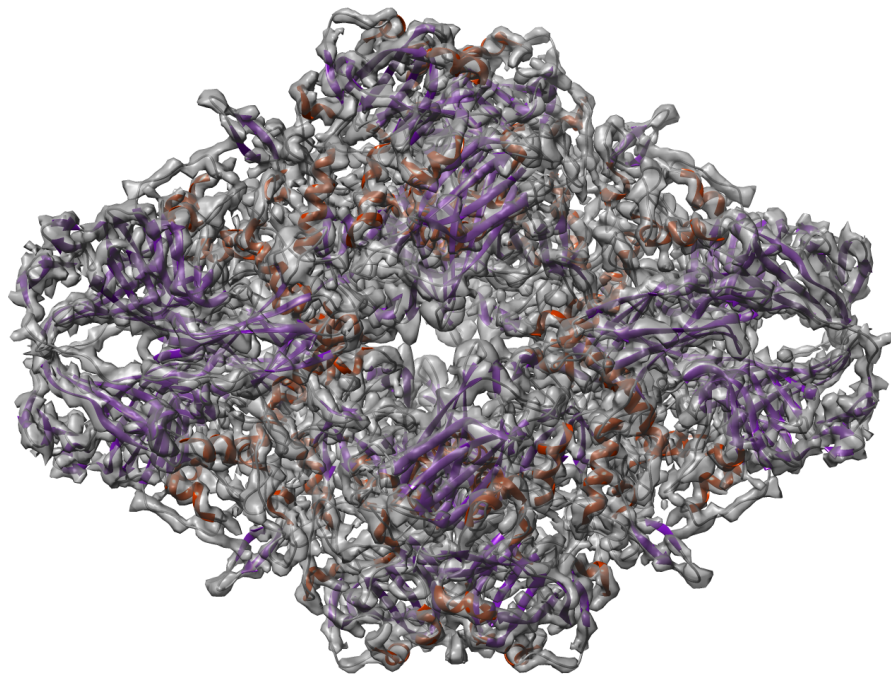
• Now you can open processed_map in Chimera, and you should be able to dock 3j7h

At this resolution, beta strands should be reasonably separated, alpha helices should exhibit pitch, and larger sidechains should be visible. You should see something like:

➡ If you refine exactly the same data set with Relion or Frealign, you will almost certainly find that the resolution curves do not overlap perfectly. With very good data like this you should expect to get nearly identical results. The cause for the disagreement in the FSC curves is generally differences in masking. Relion in particular uses a very aggressive masking strategy, which sometimes pushes the 'gold standard' resolution beyond where (in my opinion) it has any right to be. So, if comparing between packages it is critical that you use the same masks on both structures.

Particularly if you observe any anisotropy in the resolution (if the map looks smeared or blurry in some particular direction), it is a good idea to take a look at the orientation distribution of

your map. If you run *Validation and Analysis → Run e2eulerxplor*, it will give you a window showing the distribution of particle orientations within one asymmetric unit. As usual, middle-clicking on the orientation display window will give a control panel allowing you to select which iteration of which refinement to look at. The height of each cylinder indicates the relative number of particles in that orientation. Clicking on a cylinder will show the projection and class-average from that orientation. You will find that this particular specimen does have a fairly significant preferred orientation, which has likely produced some resolution anisotropy. There are ways to evaluate this in EMAN2, but the results are difficult to visualize and relate back to the map, so we won't discuss that in this tutorial.

The next assessment which is good to perform is to compare projections of the final map to reference-based class-averages, reference-free class averages, and perhaps even individual particles. There are two primary tools you can use to do this comparison: e2ptclvsmap.py and e2classvsproj.py. These programs can be used to generate figures for papers demonstrating good agreement between data and reconstruction.

For example, if we wish to compare our original reference free class averages to the projections from our final map. First we rescale the final map to match the downsampled data we used for class-averaging:
- **e2proc3d.py refine_04/threed_04.hdf rescaled.hdf --process math.fft.resample:n=4.5633**

Then we run the comparison:
- **e2classvsproj.py r2d_01/allrefs_06.hdf rescaled.hdf clsvsproj.hdf --ang=7.5 --sym=d2**

The result, clsvsproj.hdf, will look something like:
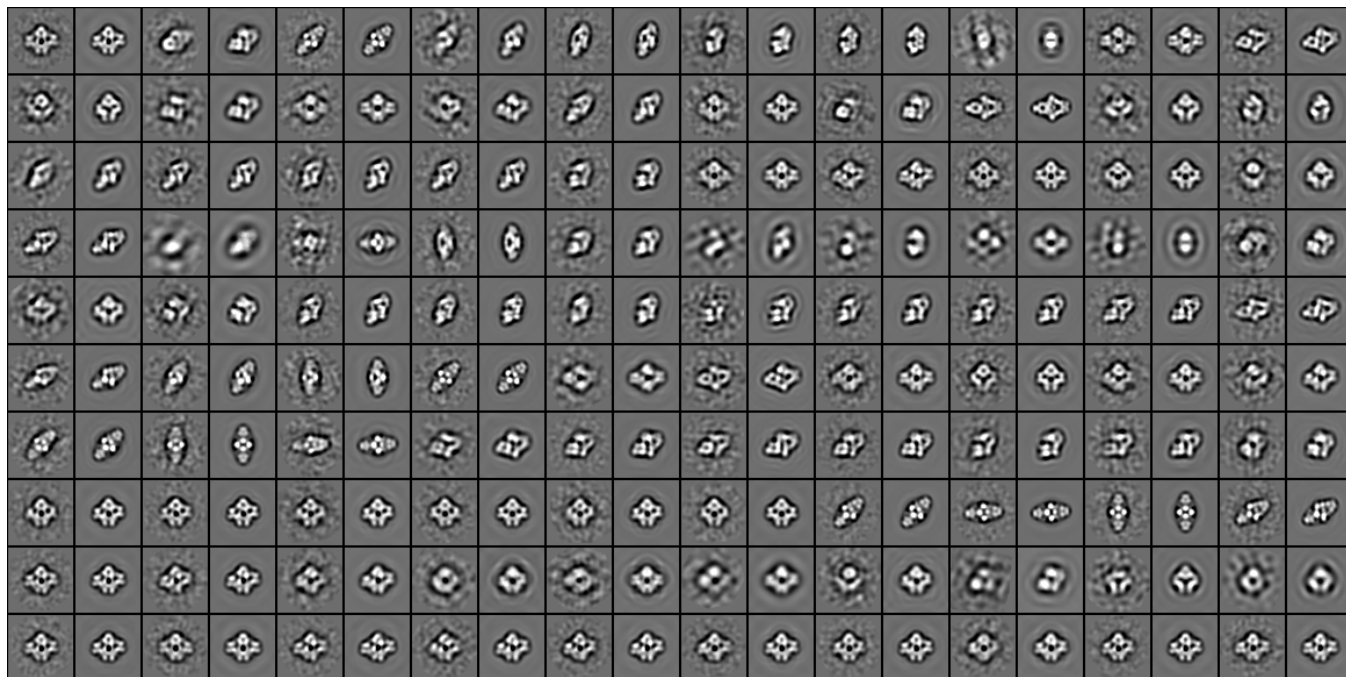


The images alternate, with first, a class-average and second, the corresponding projection. It is important to note that e2classvsproj.py also filters the projections to match the apparent filtration of the class-averages. Ideally, every reference-free class-average should have a corresponding matching projection. The exception would be class-averages which are believed

to be artifacts, such as ice contamination. If there are any class-averages which appear "good", which do not have a matching projection, you should be very concerned. This is an indication of an inconsistency between the data and the map. There could be several possible causes:

- The class-average could represent some other species of particle present in the boxed out particle set. In that case, particles associated with such classes should be removed.
- The 3-D structure is incorrect, and is excluding a view which should be present in the map. This could result if a very bad starting model were used for refinement. However, in this case, it should not have been possible to achieve a high resolution refinement.
- The particle may have some compositional or conformational variability in solution. This issue will be considered in a separate tutorial.

In the example above you'll see that there is a surprisingly good match even for the class-averages which we identified as consisting primarily of ice contamination. These class-averages are unsupervised, so the shapes of these blurry averages are not due to any sort of bias by a 3-D map. This implies that some coarse features of the particle are still managing to emerge even with a strong random ice-blob component. Still, the key factor to note is that ALL of the class-averages showing detailed features have a very nicely matching projection of the map, so we have very good self-consistency.

If you did not use the **tophat=local** option discussed above, you may wish to consider running: **e2fsc.py refine_XX/threed_even_unmasked.hdf refine_XX/threed_odd_unmasked.hdf** This will compute a local FSC map showing how the resolution varies across the map. This will have lower sampling than the original input maps, but can be visualized in chimera as a surface color. Alas this isn't a very useful thing to do on this particular data set because it is quite homogeneous with little local motion. This program can also be used to perform local filtration (tophat=local), and the parameters for the resolution estimate are adjustable at the command-line (**e2fsc.py --help**).

While we will not perform it in this tutorial, it can also be useful to run ResMap ([http://resmap.sourceforge.net/](http://resmap.sourceforge.net/)) on your refinement results. Note that ResMap works best when provided with unmasked/unfiltered volumes. While it is impossible in any CryoEM reconstruction to provide truly "unfiltered" reconstructions, because CTF correction necessarily involves filtration, you can use the same threed_even_unmasked.hdf (and odd) you used above. These are the unmasked volumes corresponding to the last successful iteration in the refinement. These are the two files you should provide to Resmap. While Resmap can work with a single map, its estimates are far better if you give it the unmasked even/odd maps from a "gold standard" pair. This "resolution" estimate is mathematically quite different than that computed by e2fsc.py.

Once you have optimized your reconstruction as described in this tutorial, there are other factors you can consider:

- Movie mode alignment - The provided data for this tutorial includes already aligned/averaged/downsampled direct detector movies. This is both due to data size and processing limits. The alignment was NOT done with the UCSF software used in many groups, but with EMAN. In our testing, EMAN produces extremely good movie alignments, quite rapidly, and does not require a GPU to do them. This is done with **e2ddd_movie.py**.

- astigmatism - the astigmatism present in the tutorial data really isn't big enough to bother correcting it. If you do correct astigmatism (with your own data), the best approach is to determine defocus/astigmatism for the entire micrographs. e2ctf_auto will then respect these values in favor of the particle based values. The whole micrograph estimation can be done in EMAN as we did in this tutorial, or if you prefer, you can use CTFFIND3 or other tools, then import the results.
- magnification anisotropy - some microscopes have a significant amount of magnification anisotropy, and it may vary with time. This is a completely different effect than astigmatism. Anecdotally this seems to occur only on FEI microscopes and is typically in the 1-2% range. It is a significant effect for large virus particles, where it may limit the resolution out as far as 5-6 A, but for small particles, the effect is often negligible to ~2 A. One approach for measuring this effect is to put an amorphous gold specimen in the microscope and assess the circularity of the gold ring. However, you can also estimate this directly from the particles if the effect is large enough to have an impact on them. To do this, run **e2evalrefine.py** with the **--anisotropy** option. This will measure the anisotropy from the particles in relation to a completed high resolution refinement. If you find a significant anisotropy, this can be corrected on the raw particle data with **e2proc2d.py** with the **--anisotropic** option. We do not suggest making the corrections at the micrograph level due to long-range Moire patterns.

**Overview/Quickstart**

This is a highly abbreviated form of the full single particle reconstruction tutorial. The numbering follows the detailed tutorial above, so you can switch back and forth. While this quick tutorial will get you through the process, you may not learn much on the way. It is a good starting point for those who already have EMAN2 experience and are just trying to learn what recent changes have been made.

IF YOU ENCOUNTER ANY UNEXPECTED BEHAVIOR HERE, PLEASE CHECK THE FULL TUTORIAL ABOVE BEFORE ASKING FOR HELP!

1. **cd workshop_2016/bgal** (assumes you have already unzipped the data)
2. **e2projectmanager.py**
   a. *Project → edit project* (on the menu bar, not the task list)
      i. *Mass* = **400**, *Cs* = **2.7**, *voltage* = **300**, *apix* =**1.275**
3. There are 2 options for importing/evaluating the data. See above for details. Here we show only the automatic form:
   a. (~5 min) *Raw Data → Import Micrographs & est Defocus*
      i. *Browse*, then select all HDF files in the **orig_micrographs** folder (press OK)
      ii. check *edgenorm*, *xraypixel* and *ctfest* checkboxes, and uncheck *invert* and *astigmatism*.
      iii. Other parameters should be correct and match 2.a.i above. *Launch*
4. (~2 min) Import box locations in to project, and extract particle images
   a. *Particles → Import Tools → Import .box or .star files*
   b. *Browse* button, and select all of the .box files in the **orig_box** folder
   c. *Launch*. Finishes almost instantly.
   d. *Particles → Generate Output*
   e. Press the Browse button and select all of the micrographs
   f. box_size → **288**
   g. *Launch*. Only takes a minute or two. Use Process Monitor to watch progress (middle icon on the right)
5. (~5 min) CTF Fitting and preprocessing of particles
   a. CTF → CTF Autoprocess
   b. check the *hires* box.
   c. Image parameters should be prefilled. *threads* → **4** (# cores your computer has)
   d. *Launch*
   e. See section 5a above if you wish to interactively browse results

6. Building sets

    a. Nothing to actually do here. Already done for you in step 5.

7. (~20 min) Generating reference free (unsupervised) class averages (2D refinement)

    *a. 2D Analysis → Bispectrum-based Class Averaging*

    b. *Input* → **sets/all__ctf_flip_lp5.lst**

    c. *Ncls* → **64**,

    d. *classalign* = **rotate_translate_tree**, *params* = **flip=1**

    e. *parallel* → **thread:4**  (NOT thread<u>s</u>:N, replace 4 with the # of cores on your machine)

    f.  The defaults should be fine for the other options, *Launch*

8. (manual, ~10 min) Eliminate bad particles by identifying bad classes
   (skip this step for 1 day workshops)
    a.  Instructions are extensive, please read the detailed section 8 above.

9. (manual, ~5 min) *Select good class-averages*
    a.  Display **r2db_01/classes_00.hdf** in a tiled display using the browser
    b.  Open the control-panel and use the "sets" interface to identify 15-20 good class-averages.
    c.  Save these good averages as **good_classes.hdf**

10. (~2 min) *Initial Model→Make Initial Model*
    a.  *Input* = ***good_classes.hdf***
    b.  *Sym* = **D2**, *tries* = **12,** *iterations* = **12**
    c.  *parallel*:  as above
    d.  *randorient* → **checked**
    e.  *Launch*
    f.  When initial model generator is done, open browser and look in "initial_models"
    g.  If you don't know how to evaluate the results, please read the full tutorial section above for step 10. You may need to rerun the program if the Monte-carlo didn't produce a good result on the first try.

11. (~1 min) Building Sets Again (better to read the section above)
    a. *Particle sets→ build particle sets*
    b. select *allparticles* and *excludebad*
    c. Enter **all-bad1** as the *setname*
    d. *Launch*
    e. *Particle sets→ build particle sets*
    f.  de*select allparticles*
    g. press *Browse* and select the best 20-25 images (with the highest SSNR-hi)
    h. Enter **best-bad1** as the *setname*
    i.  *Launch*

12. Initial refinements

    a. (~25 min) *3D refinement → Single map refinement (e2refine_easy)*
      i.   input → **sets/best-bad1__ctf_flip_lp12.lst**
      ii.  model → select your good starting model (model_00_01.hdf usually)
      iii. targetres → **15**, sym → **d2**, iter → **2**, mass → **400**, speed → **5**, apix → **0**
      iv.  *ampcorrect*→**auto**, *tophat*→(empty text box)

     v.   invar → **checked** (optional, much faster but slightly reduced quality)

    vi.  fill in *parallel* as above. Also fill in the number of cores in the *threads* box

    *vii. Launch*

  b.  Check results quickly. If ok, run the longer refinement to ~7 Å

  c.  (~4 hours) *3D refinement → Single map refinement (e2refine_easy)*
      i.   change *model* → **refine_01/threed_02.hdf**
     ii.  *input* → **sets/all-bad1__ctf_flip_lp5.lst**
    iii.  *tophat* → **global** (type this word in the box)
    iv.  *targetres→**6**, iter→**4**, speed→**5***
     v.   invar → Not recommended for final refinement
    vi.  *Launch* again. This one will take a bit longer

13. (manual ~5 minutes) Eliminate bad particles

  a. *Validation and Analysis → Eval particle Qual*

  b. *refine_XX* to best current refine_ folder

  c. Check both boxes for tutorial data, may disable *includeprojs* for large projects

  *d. Launch* (Should take ~5 min)

  *e.* Will create two new sets/* files. (see detailed instructions for more)

14. (~15-30 h) Final refinement, *3D refinement → run e2refine_easy*

There are several choices, please read section 14 above.

15. Evaluate results!