Lesson 10: Resampling

PEET alignments are often performed in a multi-resolution fashion, starting with binned data, for example, and then proceeding to unbinned data for higher resolution. Let's explore how to use imodtrans to transform a PEET model created for use with one tomogram to apply to another tomogram with a different voxel size. PEET's requirements are more stringent than 3dmod in this regard. Just because a model displays correctly in 3dmod on a given volume does not guarantee that it will work correctly with PEET. In the BPV alignment exercise, we used a model created on a 2X binned, NAD-filtered volume. Let's generate a new version of this model, which will work with the original, unbinned tomogram in PEET.

1) `cd $WORKSHOP_HOME/PEET_Labs/BPV_-3`

2) `3dmod bpv_bin2.rec bpv_bin2.rec.nad bpv_bin2.mod`

3) Press **Auto** and check **Float** and **Subarea** on the 3dmod info window to automatically adjust the brightness and contrast as the images change. Toggle between the raw tomogram and the filtered (nad) tomogram using the **4th D** arrows on the ZaP window. While the model lines up properly on both volumes, the virus particles are difficult to see against the background noise in the unfiltered volume. That's why many people use a heavily NAD-filtered volume for initial modeling. Close 3dmod.

4) `imodinfo bpv_bin2.mod`
   Notice the presence of the **Model to Image index cords**: section and, specifically, that SCALE is 15.2 for each of X, Y, and Z. Run `header bpv_bin2.rec` and you will see that this matches the voxel

size in angstroms of the binned volume. Because this model was created using 3dmod on an identically binned volume, the appropriate information has been stored automatically in the model header. Even without this information, (*e.g.* if a model was created using point2model), PEET would function correctly as long as the model coordinates are entered in voxels.

5) `model2point -float bpv_bin2.mod bpv_bin2.txt`
This saves a text version of the particle locations for later comparison. The float option cause coordinates to be save as floating point, rather than rounding to the nearest integer.

6) `imodtrans -i bpv.rec bpv_bin2.mod bpv.mod`
This creates a new model, suitable for use with the original, unbinned volume. Note that if *bpv_bin2.mod* lacked the Model to Image Index cords information, we would first need to create a model containing
that information, *e.g.* by running:

```
imodtrans -i bpv_bin2.rec bpv_bin2.mod temp.mod
mv temp.mod bpv_bin2.mod
imodtrans -i bpv.rec bpv_bin2.mod bpv.mod
```

7) Run `imodinfo bpv.mod` and note that the new SCALE reflects the correct 7.6 angstrom voxel size for the unbinned volume.

8) `model2point -float bpv.mod bpv.txt`

9) Run `head bpv.txt bpv_bin2.txt` and compare the coordinates of the first few particles. Notice that the coordinates for the original, unbinned volume are twice those of the 2X binned volumes, as you'd expect. This illustrates how to transform models from one scaling to another while always satisfying a key PEET requirement: Particle coordinates must reflect positions within the volume in voxels, with the origin at the lower left corner.

BPV ALIGNMENT WITH UNBINNED DATA

We've already examined a symmetrized alignment of BPV using 2X binned data. Now let's apply those results to generate a final, symmetrized average using the original, unbinned data. Because there will now be 60 entries in the volume table, this case is considerably more complex than the previous one, although the ideas are exactly the same. To avoid having to manually modify the shifts in 60 motive lists, we'll first use createAlignedModel to generate aligned models and motive lists. Then we'll transform the aligned models for use with the original, unbinned tomogram. Finally, we'll use the transformed models and the aligned motive lists to perform the final, unbinned alignment. Because a number of commands are required, I've created a script named *prepareForUnbinnedAlignment.sh* in the symmetrized, binned alignment directory to handle the necessary steps. Let's take a look at it.

10) `cd PEET/icosahedral`

11) `gedit prepareForUnbinnedAlignment.sh`

Read over this shell script and its comments. In addition to things we have discussed before, one new feature is illustrated. After generating the aligned motive lists, we'd like to make them visible in the directory we'll use for the final alignment under different names—ones that we can easily expand using templates. While this could also be done by renaming the files while copying or moving them, in this case, we've chosen to just make a "link" back to the original files. The net effect will be the same either way. NOTE: `ln <src> <dst>` creates a "hard link" and will work on any of Linux, OS X, or Windows with Cygwin. Symbolic links, created by `ln -s <src> <dst>`, are often preferred on Linux or OS X, but will not work reliably on Windows. Exit gedit.

12) `./prepareForUnbinnedAlignment.sh 2>&1 \`
`tee prepareForUnbinnedAlignment.log`
where "\" should be followed immediately by **Enter**. This will execute the commands in the *prepareForUnbinnedAlignment.sh* script. It will run rather quickly, completing in less than a minute.

13) `cd ../unbinned`

14) `etomo *.epe`
As in previous exercises, examine the parameter settings, making sure you understand them. Then press **Open averages in 3dmod** to explore the resulting averages which we've precomputed for you. As a point of reference, this unbinned alignment required about 20 hours using 20-30 cores with 2.4 – 3.3 GHz clock rates. Close Etomo and any 3dmod windows.