emClarity

tutorial version 1.1.7

1.0 Contents

\

2.0 How to use this guide

The purpose for this document is to provide the minimal set of instructions needed to begin processing sub-tomogram data using emClarity. Other sources of information include the emClarity wiki and the emClarity mailing list. Of particular interest are a number of tutorial videos which discuss the details, both practical and technical, behind each of the steps you will execute during the tutorial. The first video on that page, "Overview on info related to emClarity"

If at any point you are confused, or something seems to not work as you expect, the videos are a good place to start; feel free to post to the mailing list any questions that you cannot resolve on your own.

Sections 1.0 – 2.2 which detail software and project set-up are needed for all new users. The remainder of section 2, which details tilt-series alignment and getting and formatting the required files, may be skipped initially to get straight into the sub-tomogram workflow, however, in a real project they would of course need to be executed.

2.1 Expected Results

You should be able to get to an ~ 7.0 Å structure by following the tutorial as written.

2.2 Diagnostic information

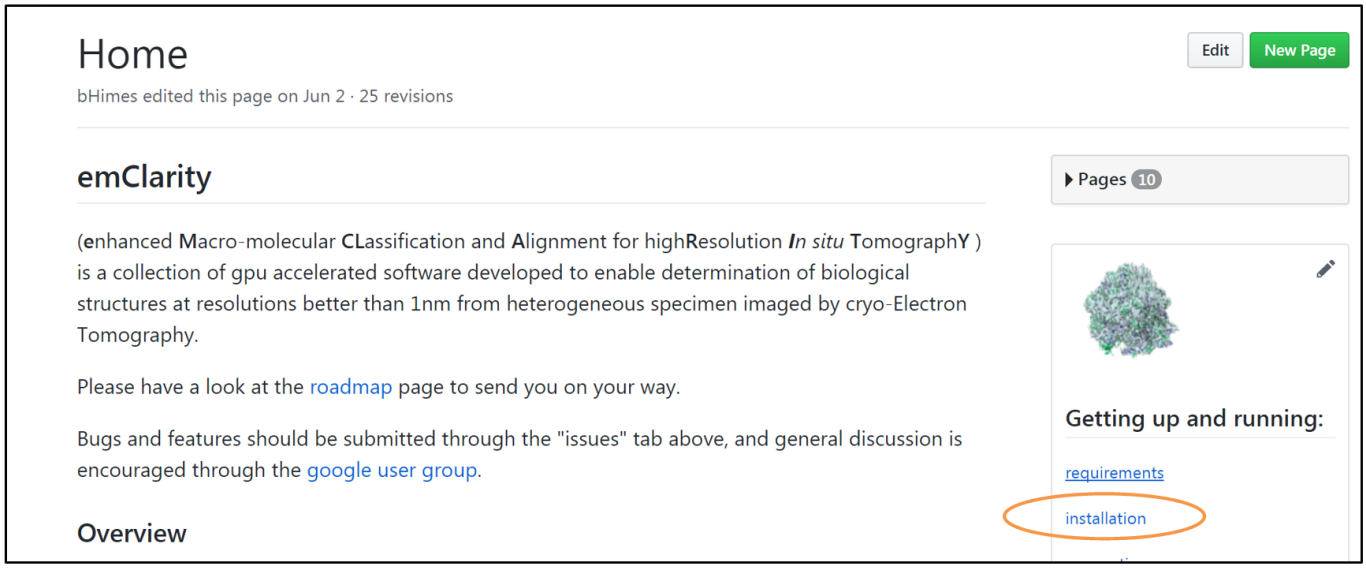
As emClarity is still new and shiny, it will produce quite a bit of diagnostic information to help with trouble shooting. The largest items will be aligned stacks from previous rounds

of tomoCPR in the “aliStacks” directory, as well as reconstructions in the “cache” directory. Anything in the “cache” will be automatically created when needed, so feel free to delete those data when you are finished.

2.3 Setup

2.3.1 Obtaining the dependencies

1) Navigate to the emClarity wiki
2) Click on the installation link on the sidebar to find a section describing the most up to date software and versions required.



Note 1: You must use the version of the matlab MCR as specified.

Note 2: To download the IMOD version, you must right click and “save link as.”

2.3.2 **Installing the dependencies**

Both IMOD and chimera can be installed locally without admin rights

```
>$ ./currentVersionOfIMOD.sh -skip -dir /Path/to

>$ mkdir MyMatlab_MCR ; cd MyMatlab_MCR    # whatever you want to call it

>$ unzip /path/to/mcrdownload.zip

$> ./install -mode silent -agreeToLicense yes -destinationFolder MyMatlab_MCR
```

When the matlab MCR installation is finished, you will be provided with a line to append to your LD_LIBRARY_PATH environmental variable. You may either set this at startup by modifying your ~/.bashrc file, or preferably, by copying this line into a text file called **mcr_bash.sh** saved in the directory where you installed the MCR. The script you will use to run emClarity will source this file on start-up so that the appropriate libraries can be found.

Each of these are well documented, so rather than anticipate every situation, have a go and if you get stuck feel free to ask for help.

2.3.3 **Download and install emClarity**

The source code is kept one level above the wiki. There is a link to download a compressed folder, which should produce the results formally obtained by cloning into the project.

```
~~>$ git clone --depth=1 https://github.com/bHimes/emClarity.git~~      NO LONGER USED

>$ unzip emClarity-1.X.X   # where X is the version you want to install
```

This will create a directory called emClarity which will have a few items in it, which are described in the accompanying README.

Modify the line in the emClarity script to point to the mcr_bash.sh file you created in the installation of the matlab MCR.

Modify also the line to point to the installation directory.

Note 1: The file emClarity is just a shell script that points to the binary which will have a 7 character suffix which is beginning of the hash identifying the particular version (commit) that generated the file. This will also show up in all log files, making trouble shooting more straight forward.

Note 2: If you are running on a distributed computing system, you can use the example "runMatfile.sh" in the docs folder as a template, which essentially creates a run script like emClarity but with additional details relevant to queue submission.

Finally run the following to have emClarity check your installation for you.

```
>$ emClarity check
```

This program will create a log file that will list the locations of imod and chimera as well as available GPUs. emClarity itself has a few checks built in to make sure it can run.

3.0 Obtaining tutorial data

Obtain the tilt-series available from EMPIAR-10045.

Note: an important feature of cryoSTAC is the ability to work with a small data set, and then scale the process. It is best practice to work the whole way through the pipeline with as small a set as possible, partially scaling up to make sure everything holds, and then processing your full data.

The same approach may be taken with the tutorial; it should be possible to obtain a low-resolution but recognizable ribosome with only one or two of the tilt-series, allowing you to quickly work through the full pipeline. This will at least confirm that everything "runs" (producing some output) in your hands and on your gear. Only after this does it make sense to then confirm you are able to produce the correct output, i.e. a ~7 Å map.

3.1 Setting up your working directory

All commands run from emClarity will be run from your project directory, which should be set-up as instructed as emClarity uses specific relative paths to sort and identify important files throughout the pipeline. A map of the file system hierarchy and descriptions related can be found in **appendix F**.

In the example commands, we will call this **projectDir**.

Create a directory where you will keep your raw data as well as perform coarse alignments. We typically name this **rawData**. It is not necessary to copy your data here if you would prefer just to link to it instead.

As IMOD generates many intermediate files, it is nice to make a directory for each tilt-series.

```
>$ mkdir projectDir/rawData

>$ mkdir projectDir/rawData/imodAli05 ; cd projectDir/rawData/
```

3.2 Downloading the tilt-series

Tilts 005 and 008 are mostly flat, so start with either those. It is advisable to use the "Aspera Connect" or alternatively you can pull directly from the ftp server.

Note: the Relion file extension convention ".mrcs" to get the stacks, and not the full tomograms! These should be only a little larger than 1 Gb each.

You should be in your **rawData** directory.

```
>$ for iTilt in 05 ; do wget -b \
ftp://ftp.ebi.ac.uk/pub/databases/empiar/archive/10045/data/ribosomes/Tomograms/${iTilt}/IS002_29101 3_0${iTilt}.mrcs ; done
```

A few comments on the last command:

1) It is not necessary to use a loop here, but some basic unix (bash) shell scripting will be useful to know, so this provides an easy example.
2) wget is a command to download a file from the command line. The -b sends the operation to the background so multiple downloads may proceed at once.
3) The ${iTomo} substitutes each value following the "in" phrase at the beginning at the appropriate index through the for loop.
4) The " \ " just means that the command is continued on the next line, allowing you to hit the enter key without running anything yet.

IMOD expects a different file extension, so we will change to our alignment directory, and create a link to the file but with a different name.

```
>$ cd imodAli05

>$ ln -s ../IS002_29101_3_005.mrcs tilt05.st
```

3.3 Have a look at the raw data.

```
$ imod -bin 4 tilt05.st
```

Note 1: bin decimates (down samples) the data, but only in 2d since IMOD treats this as a stack of images. Using -B 4 would bin in 3d.

Note 2: in the default (zap) window, a left click on the image will play through the stack, allowing you to use motion to assess the quality.

Note 3: While we can fix the tilt-series alignment, the same conditions that cause the automated tracking to fail during data collection, are often those that also preclude high resolution work. If the tracking is very poor, the data are likely not suitable.

The remainder of section 2 can be skipped by copying the expected tilt series alignment results.

```
>$ mkdir projectDir/fixedStacks

>$ cp -r /path/to/emClarity/docs/tiltAlignments/* emClarity_tutorial/fixedStacks

>$ cd emClarity_tutorial/fixed Stacks

>$ nTomo=1; ls ../rawData/*.mrcs | while read iTomo ; do ln -
s ../rawData/${iTomo} tilt${nTomo}.fixed ; nTomo=$(($nTomo + 1)); done
```

3.4 Coarse alignment

While these example tilt series are already aligned, we need to generate a model file that tells emClarity where the gold fiducials are, so they can be removed, and additionally collect a few other files. If you are unfamiliar with this process, or run into any trouble, the tutorial video contains many additional details.

3.4.1 Setup – fixing the header

It is important at this stage to fix the information in the header that is incorrect. The pixel size is not stored correctly for whatever reason. This is something you should always confirm with your own data as well. Running the following from your rawData directory will result in a set of command prompts.

```
>$ alterheader IS002_291013_005.mrcs
```

Enter "cel" which will instruct *alterheader* to alter the cell dimensions, which then also changes the stored pixel size. The formula is to enter the wanted pixel size multiplied by the tilt-series dimensions in pixels. I.e. calculate using the following formula, where NX is the number of pixels in X etc.:

2.17*NX 2.17*NY 2.17 *NZ 90 90 90

It is convenient to set the cel angles to 90 degrees for compatibility with other programs.

Strike enter to return to the menu.

Enter "done" to exit the program

3.4.2 **Running the tilt-series alignment**

From within each imodAliXX directory, the etomo interface may be opened to run a coarse tilt-series alignment.

```
>$ etomo
```

1. Select “build a new tomogram.”
2. Select your data set, single-axis, scan header (gold fiducials are 10nm here) and also change the image rotation angle to **0** . We didn't fix this in the header because it should never be this far off in your own data that is totally unaligned.
3. Select “Create com scripts”

A new screen will pop up that covers the important steps.

4. Pre-processing isn't necessary for this data set, but you should check your own, particularly if you have CCD data.
5. Bin the coarse aligned stack by 4 unchecking “convert to bytes.” It seems a pixel size of 4-8 Å produce good enough alignments.
6. Use all the available fiducial markers. If you have many in your own data, 15-20 spread over the area occupied by your specimen is nice.
7. Run fine alignment, selecting local alignments and robust alignment.
8. Skip tomogram positioning, but do click “create final alignment”
9. Create full aligned stack. This will may be deleted at the end but seems to be necessary to make sure the transforms are properly updated for output.
10. Under the erase beads tab, use erase beads 3d setting thickness to 3000.
11. Select “align and build stack”, “run find beads 3d”, “project model into 2d”

3.4.3 **Gathering the results**

Please pay careful attention to the naming conventions in this section. It is useful to put all of these commands into a simple script.

1) The refined tilt angles

```
>$  mv  tilt1_fid.tlt ../fixedStacks/tilt1.tlt
```

2) The transformation matrix

```
>$  mv  tilt1_fid.xf  ../fixedStacks/tilt1.xf
```

3) The local alignments (if obtained)

```
>$  mv  tilt1local.xf  ../fixedStacks/tilt1.local
```

4) The gold bead model (for erasing later)

```
>$  mv tilt1_erase.fid  ../fixedStacks/tilt1.erase
```

5) The fiducial model needs to be converted to one emClarity can read directly. (This is on the list to automate.)

```
>$ imodtrans -i  ../fixedStacks/tilt1.fixed  ../fixedStacks/tilt1.erase tmp.mod

>$ model2point  -float tmp.mod  ../fixedStacks/tilt1.erase2
```

6) We do not want the final aligned stack that IMOD applied the transformations to (tiltXX.ali) instead there are two alternate choices for the “fixed” data.

   if you didn't do any preprocessing (xray/statistical outlier removal) instead we will link to the raw data.

```
>$ cd  ../fixedStacks  ;  ln  -s  ../rawData/IS002_291013_005.mrcs tilt1.fixed
```

if you DID remove xrays

```
>$  mv tilt1_fixed.st ../fixedStacks/tilt1. fixed
```

7) Repeat this for each of your tilt series, incrementing the name tilt1 tilt2 …tiltN. In practice you can often run multiple alignments concurrently, so in that case you might have named your temporary directory imodAli_1 imodAli_2 …etc.

There is one more file to create which tells emClarity about the dose applied and the data collection scheme. A suggestion for how to create this file is described in the alignment video, but please note that the original video describes the data as being 20˚ → -60˚ , but this should have been +30˚ → -60˚ , 33˚ → +60˚

```
fixedStacks/tilt1.order      ; # a list of tilt angles in the order collected
```

4.0 Obtaining an initial CTF estimation

First set up the microscope parameters in your parameter file, please see the video if the text descriptions are unclear.

For each tilt series run the following command. emClarity will simply send the job to the GPU with the most available memory – **it is advisable to only run one job / GPU at this step. If you are not running on a distributed system with a queue manager, simply open up one shell for each GPU. E.g. for two gpus**:

```
>$  for iTiltSeries in 1 2 3 4 ; do emClarity ctf estimate param0.m tilt${iTilt}; done

>$  for iTiltSeries in 5 6 7 ; do emClarity ctf estimate  param0.m tilt${iTilt}; done
```

When this is finished, review the results. The fit to the power spectrum which is reported in the file “fixedStacks/ctf/tiltXX_psRadial.pdf”

** If the fit seems to be very poor, have a look at the file “fixedStacks/ctf/tiltXX_ccFIT.pdf” which plots the cross-correlation score as a function of defocus. There is often an obvious correct peak (smoothly rising and falling) and the jagged, often close to focus incorrect peak. You can use this to restrict your defocus search range.

An additional directory called “aliStacks” will be created where the tilt-series with the current transformation parameters applied will be placed. After each round of tomoCPR these are re-calculated and the suffix “_ali1_ … _ali2_ …” will be incremented and is = #tomoCPR iterations completed + 1.

Please refer to the relevant section in appendix F for some troubleshooting information should the fit go awry.

5.0 Template-matching

If you have elected to skip the tilt-series alignment and initial CTF estimation by copying the expected results to your fixedStacks directory, you will need to first create all of the alignedStacks. Normally, these are made initially by “emClarity ctf estimate” and in later cycles (after each round of tomoCPR) are updated with “emClarity ctf update.” These both carry out interpolation in Fourier space (shift, rotations and magnification) in order to preserve as much high resolution signal as possible.

Since this is an atypical way to run, for the sake of the tutorial, you can just use Imod’s newstack command. Repeat this for each of your tilt-series.

```
>$ mkdir aliStacks ; cd fixedStacks

>$ newstack -xf tilt05.xf tilt05.fixed ../aliStacks/tilt05_ali1.fixed
```

5.1 Selecting regions in each tilt-series to process

It is often useful to select sub-regions from your tilt-series to be used for subsequent processing; either because there is only specimen in some areas, or alternatively to reduce the final size of reconstructions, which can easily be larger than 100Gb each if a 4k x 4k tilt series is reconstructed.

The following steps are accompanied by a short video demonstrating the process.

Run the script from you docs folder to set things up, either copy to your working directory, add it to your path, or specify the full path. Assuming that you have copied it to your working directory:

```
>$ ./recScript2.sh -1
```

This will create a new directory, “bin10” which will have a bin10 tomogram for reconstructed for each tilt-series present in the fixedStacks directory.

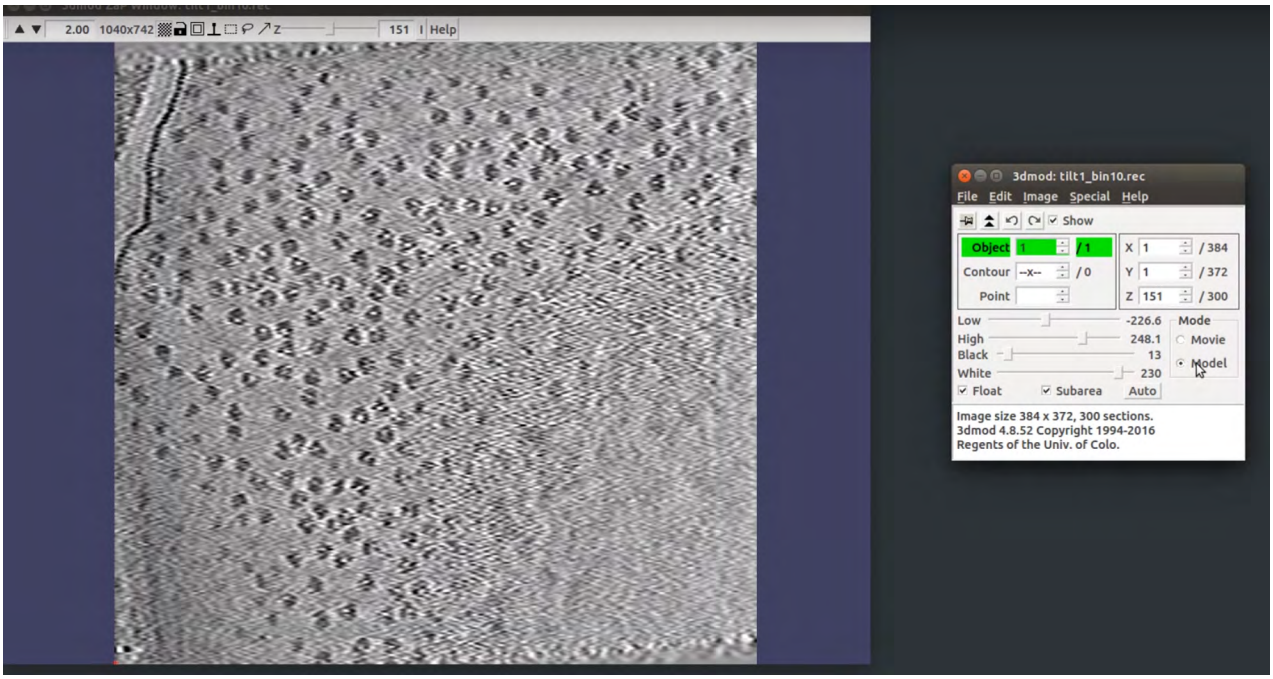
The goal is to define 6 points, xMin/xMax, yMin/Max, and zMin/zMax for each sub-region you wish to study. This is most easily done by creating and IMOD model, where each of these points is represented by its own contour, and selected in this order. You should end up with (number of regions) * 6 contours in your model.

Add alternate version for virus particles.

```
>$ cd bin10

>$ imod -S tilt1_bin10.rec
```

Each model point must be on a new contour. Either press the “n” key each time or open the edit → object → type dialog and set the maximum value of points per contour to 1.



Finally, save the model with the same name as the reconstruction, where the postfix has been changed from “rec” to “mod” in this case tilt1_bin1.mod.

You will do this for each of your 2 or 7 tilt series.

After all are completed, we need to convert these models into an input for emClarity, again using the recScript2.sh.

** It is helpful to keep a list of ~ how many particle you expect based on this visualization. Of course you can incorporate other a prior information here as well.

```
>$ cd ../      # back to your project directory

>$ for iTilt in 1 2 3 4 5 6 7 ; do recScript2.sh tilt${iTilt}; done
```

This will create a directory called “recon” with files in it needed for emClarity. Those with the postfix “.coords” are the final barrier into the sub-tomogram workflow. If you decide after template matching to scrap a particular tilt-series, just delete this file prior to initializing the project meta data.

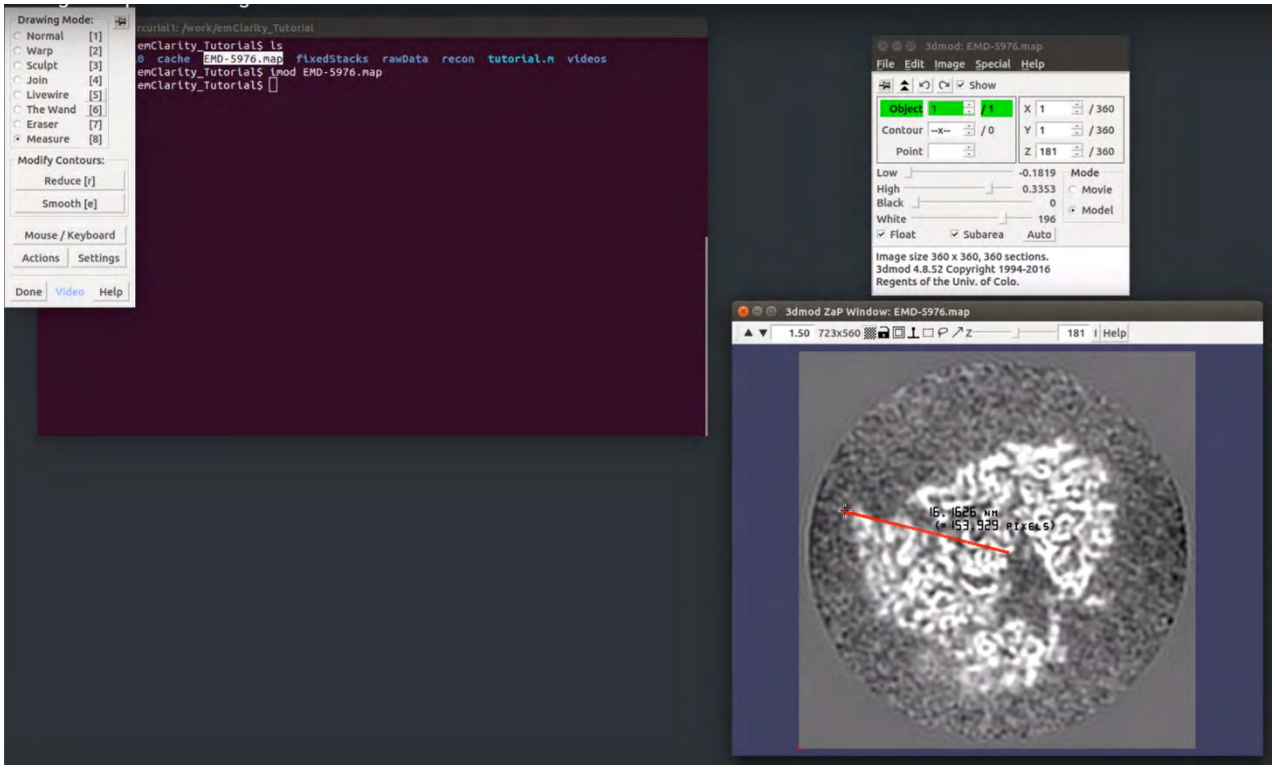
5.2 Choosing and preparing a reference

Download a suitable yeast 80s map to use as a reference, EMD-5976 for example. We assume we have some low resolution structure (< 40A) available.

Determine the particle radius in each dimension, this easily done in imod, selecting the Special menu, and then “drawing tools” as demonstrated in this video.

You will also need to rescale the map to make to match the data. Pixel sizes are in Angstrom. **Please note that the reference should not be binned, this will be done by emClarity internally *after* the wedge mask is applied for each rotated reference, which is substantially more accurate**.

```
>$ emClarity rescale EMD-5976.map riboReScale.mrc 1.05 2.17 GPU
```



5.3 Selecting template matching parameters

Select a threshold that is about 10% > than the expected number of particles. It is okay if this is not very accurate, as the results can be interactively edited in 3d and later through classification. It is best practice to clean up the data as much as is reasonable prior to relying on classification though.

For particles like the ribosome, there are no real constraints on the orientation, so searching a full grid in 12 or 15 degree increments is required.

This would be specified as [180,15,180,15] for +/- 180 in 15 degree steps (polar angle, which also determines the azimuthal angle) and +/- 180 degrees in 12 degree steps in the planar angle.

If you have a particle with C6 symmetry (with the symmetry axis corresponding to the Z-axis) you might search a more limited range [180,15,36,9]

The sampling rate should be chosen to give a running pixel size between 8-12 Angstroms. Generally speaking, a larger particle can be downsampled more.

5.4 Running the job

Similar to ctf estimation, each call to the template search program should be run on its own gpu.

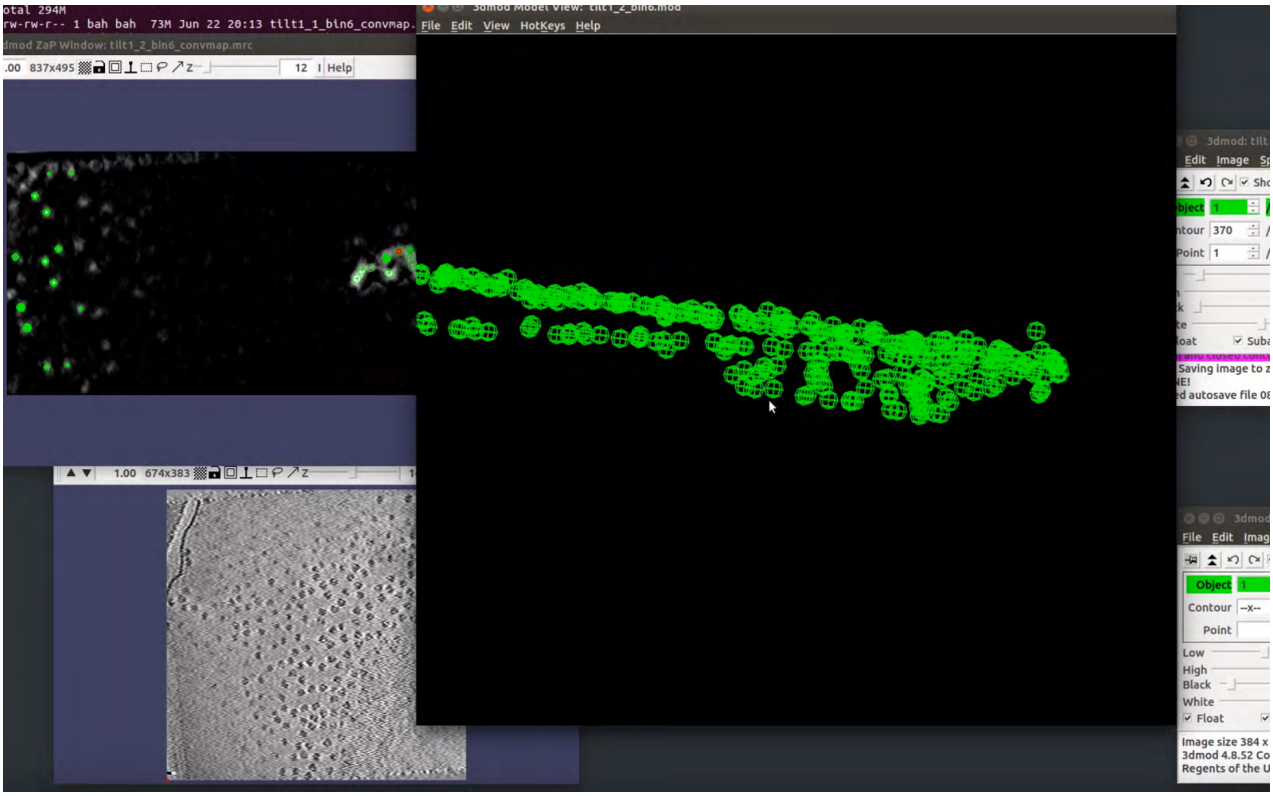
```
>$  for iTilt in 1 2 3 4 5 6 7;  do for jTomo in 1 2 3 4 ; do emClarity templateSearch
param0.m tilt${i} ${j} riboReScale.mrc 1 1 ; done ; done
```

Where the final two parameters are the symmetry and gpuIDX, the number of tomograms will match whatever you have selected from each given tilt series.

Note: If you have symmetry, C6 for example, enter a 6 for the second to last parameter here. This will randomize the euler angles to any one of the symmetry related positions, which helps to reduce wedge bias.

5.5 Analyzing and editing the results in 3d

The primary goal here is to remove false positives due to strong homogeneous features like carbon edges, membranes, or residual gold beads. In addition to the bin10 tomogram, which helps to visualize the sample, the template matching produces a “cumulative correlation” map which can be opened alongside a 3d model of the best peaks. Taken together, these tools make editing the results pretty effortless as demonstrated in the [tutorial video].



6.0 Initial run

The final steps to getting up and running are described in this tutorial video.

6.1 Initialize the subTomoMeta

This step creates the binary database that tracks the information for you project. It will look in your “convmap” directory for any model files and also in your “recon” directory to import the appropriate dimensions for each area you include in your reconstructions.

```
>$ emClarity init param0.m
```

You will also need to create ctf corrected tomograms. If you decide to work at the same binning as during template matching, you MUST delete the existing tomograms in your cache directory.

```
>$ emClarity ctf 3d param0.m
```

6.2 Create an initial average

Every cycle of alignment and averaging begins by creating an average for each half-set of the data. For cycle 0 (zero) the alignment parameters are coming from template matching.

```
>$ emClarity avg param0.m 0 RawAlignment
```

This will output a directory called **initialTomoAvgs** which saves a sub-tomogram average from each individual reconstructed area. It is useful to scan through these and make sure no larger errors were made earlier that were missed in the template matching step. This is only produced for cycle 0.

For all other cycles, the following output is created, where the numbers are replaced with the cycle number. There is an “EVE” for every “ODD” I’ll list.

- cycle000_projectName_class0_Raw_ODD_NoWgt.mrc
    - The “raw” subtomogram average, with possible quality weighting (a B-factor based on current score, which is only applied from cycle001 onward)
- cycle000_projectName_class0_Raw_ODD_Wgt.mrc
    - The sum of the 3D-sampling funcitons, used in the Wiener filter and also in the sub-tomogram alignment step.
- cycle000_projectName_class0_REF_ODD.mrc
    - This is the wiener filtered map, which is also low-pass filtered according to the “gold-standard” fsc, and will be the reference for this cycle of alignment.
- FSC/ cycle000_projectName_class0_Raw_1-fscFull_GLD.pdb
    - Plot of the fsc calculated with a tight-mask, phase-randomized corrected tight mask, and a soft-mask with solvent correction. The same file with .txt extension has the same columns, with the first being 1/Å.
- FSC/ cycle000_projectName_class0_Raw-1-shapeMask_1.mrc
    - Soft shape based mask for the solvent corrected FSC calculatoin
- FSC/ cycle000_projectName_class0_Raw-1-paritcleVolEst_1.mrc
    - Tight mask used for the phase randomized corrected fsc calculation (just for visualization, not used internally) and also for estimating the particles volume.

Note: *The “class0” part is related to the multi-reference alignment, which is currently disabled. I may re-introduce it in the future, so for now, this may simply be ignored.*

6.3 Alignment

An example shell script with accompanying parameter files can be found in your emClarity/docs/exampleRunScripts. This simple approach should get you to an ~7 Å map. Further improvements require a more detailed approach.

The command to run is simply:

```
>$  emClarity  alignRaw param0.m  0
```

A directory called **alignResume/cycle000_projectName** will be created, where a text file for the alignments found in each tilt-series is printed out. These are used in case a run crashes or you wish to stop and then restart.

Note:

1) *If you want to re-run an alignment with new parameters, you will need to delete these text files, otherwise they will be read in and not duplicated.*
2) *emClarity currently is set up to run on a single node (computer) at a time. One hack to run on two nodes, is to specify the alignment cycle number as negative. This will run a copy of alignRaw in reverse, writing the alignments to the same directory. This will not save the alignments in the subTomoMeta data, only the forward process will do so!*

6.4 tomoCPR

tomoCPR is used to refine the tilt-series geometry, and typically is not used every iteration of sub-tomogram alignment, as this would be overkill. I tend to use it each time at the end of alignment at a given binning, before stepping down to the next binning.

Prior to tomoCPR, I remove any sub-tomograms that may have drifted onto an overlapping position. This command selects the higher scoring of the two and deletes the others.

```
>$  emClarity removeDuplicates  param3.m 3
```

You can then run tomoCPR

```
>$  emClarity  tomoCPR  param3.m  3
```

If you have run emClarity before, you'll note that the molecular mass is no-longer passed on the command line. It is instead in your parameter file, **"particleMass"** and is still specified in megaDaltons. This should be accurate +/- ~50% of your best estimate for the protein/DNA mass inside the region you define by **"particleRadius."**

After completing tomoCPR, you'll need to update your stacks and also make new reconstructions.

```
>$  emClarity  ctf  update  param4.m
```

If you have run emClarity before, you'll note that the "-1" is no longer taken on the command line, as this is the default operation.

```
>$  emClarity  ctf  3d  param4.m
```

If you have run emClarity before, you'll note that the previous cycle no longer needs to be specified on the command line.

7.0 Classification

7.1 PCA

7.1.1 Selecting regions to for analysis in real space

You may want to focus your classification on a particular region of your particle. To do so, simply change the “Cls_mType” to either ‘sphere’, ‘cylinder’, or ‘rectangle,’ and change the “Cls_mRadius” to specify the desired radius for the mask in each dimension (in angstroms.)

Note: that a soft shape-based mask will also be used in addition to this geometric mask.

```
>$  emClarity  pca  param4.m  4 0
```

While the particle extraction and missing-wedge compensation is done on the GPU, this singular value decomposition is done on CPUs. The parameter “nCpuCores” specifies how many cores to use, up to the total number visible to matlab.

This step can be memory intensive as a matrix holding all of the voxels from each sub-tomogram to be analyzed, must be held in the main (host) memory. This is reduced by using a smaller area.

Data are periodically pulled off the GPU and put into host memory for this same reason. The number of sub-tomograms kept at any on time on the GPU for this step is specified by the “PcaGpuPull” option. This is defaulted to 1200, which seems to work well, but if you run into OOM problems, reducing this may help.

7.1.2 **Running on all or some of the data**

For very large data sets (tens of thousands) the eigenvectors describing the important variation can be obtained with a subset of the data, which saves a lot of computation. The command is the same as above, but with the parameter “Pca_randomSubset=0” now changed to “Pca_randomSubset=XXXX” where XXXX is the number to include. It seems 25% or at least 3-4000 is a good way to go.

You must still project each sub-tomogram down along these eigenvectors, for the full data set. This is done by running a second call to the pca program, but with a one as the final argument

```
>$ emClarity pca param4.m 4 1
```

7.1.3 **Analyzing eigenimages**

Before moving on to clustering, you must choose the feature vectors you want to use. You can think of these as a reduced representation of the most interesting components of your structure(s).

For each resolution you have specified in the parameter “pcaScaleSpace” you will have the following three files. XX corresponds to the “Pca_maxEigs” which is the maximum number of eigenvectors to solve in the decomposition. The N corresponds to the resolution at position N in your “pcaScaleSpace” vector in the parameter file.

- cycle004_projectName_vairanceMapXX-STD-N.mrc
    - These may be opened on top of your averages from this cycle, and should highlight regions where there is significant variability across the data set. If these maps show little blobs everywhere, there is either no significant variability (just picking up noise) or there is still substantial wedge bias at this resolution.
- cycle004_projectName_eigenImageXX-STD-mont-N.mrc

    - This is a 3D montage of the eigenvectors, organized from 1 at the lower left moving across the bottom row incrementing by one. These also correspond to the fraction of variance explained, with eigenvector #1 explaining the greatest portion.
- cycle004_projectName_eigenImageXX-SUM-STD-mont-N.mrc
    - The eigenvectors can be difficult to interpret, so it is useful to add them to the average map, which highlights the difference that is being explained. This is usually what I look at to select which eigenvectors (feature vectors) I want to use in subsequent clustering.

7.1.4 **Analyzing variance maps**

For now, refer to figure 6 in the preprint, or figure 3 in the Nature Methods paper.

7.1.5 **Selecting features for clustering**

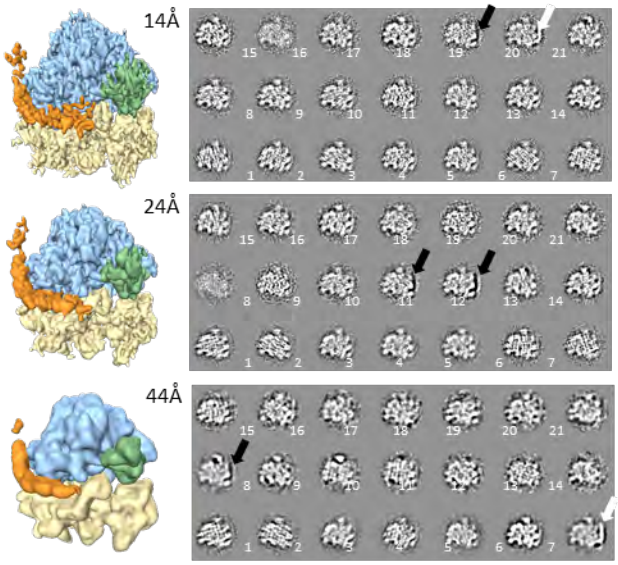
It seems that usually 6-18 per resolution is plenty to obtain good separation.

You can select as many (or few) from any resolution as you want, but the number of entries in each row of the “Pca_coeffs” vector in your parameter file must be constant. Just put a zero in place. Say you want # 4-6 from your highest resolution, 7-11 in your middle, and only number 1 from your lowest resolution.

Pca_coeffs=[4, 5, 6, 0 ; 7, 8, 9 ,10, 11 ; 1, 0, 0, 0, 0]

You may find that the missing-wedge compensation is not perfect, which results in “streaky” looking eigenvector-sums (third listing in 6.1.3) This is most probable at the highest resolutions, and these eigenvectors should be ignored. (see #1,2 at the low resolution in the figure below.)

Once you have specified these features, it is time to move on to clustering.



7.2 Cluster

7.2.1 Select algorithm

I have elected to leave this to default to kMeans as it seems to do quite well. HAC and SOMs are also useful, but do not seem to do any better, and so simplicity is preferred.

7.2.2 Set number of classes

It is hard to know ahead of time how many different classes you will have. In emClarity, you may specify multiple class occupancies, and then create those averages and decide which is best.

For example, run classification with 2, 4, or 8 classes each you would set: “Pca_clusters=[2,4,8]” in your parameter file.

7.2.3 **Run and analyze results**

```
>$ emClarity cluster param4.m 4
```

If you would like to see the class populations, each are printed into a text file in your project directory called projectedName_cycle004_ClassIDX.txt.

To create a montage of your class averages for the class of 8, you would then set:

“Cls_className=8”

“Cls_classes_odd=[1:8,1.*ones*(1,8)]” or equivalently

“Cls_classes_odd=[1,2,3,4,5,6,7,8 ; 1,1,1,1,1,1,1,1]”

Each entry in the second row is a CX symmetry, which may not be the same for each class, for example if you have a hexameric receptor lattice with 3 and 6 fold symmetric centers defined by a binding partner.

You would then run (notice the difference at the end)

```
>$ emClarity avg param4.m 4 Cluster_cls
```

This can be done for each of your classes specified in the “Pca_clusters” vector.

7.2.4 **Selecting classes to use as references**

If you find there are three good classes and one that is junk, and you want to remove this from further consideration, you would open your class montage in imod. Change to model mode (strike the “m” key) and then create a new model point for each class you want to delete.

Save this model as something like

“cycle004_odd.mod”

Then run

```
>$ model2point cycle004_odd.mod  cycle004_odd.txt
```

Subsequently, you will use the emClarity “geometry” command, which has many functions to modify the binary subTomoMeta database, to remove each member of the selected classes.

```
>$ emClarity geometry param4.m 4 ClassAlignment RemoveClasses cycle004_odd.txt STD
```

7.2.5 **Skipping class average alignment and rawAlignment**

Finally, you will need to “skip” the currently disabled class average alignment, and update the subTomoMeta so you may proceed to the next cycle of averaging/alignment.

```
>$ emClarity skip param4.m 4
```

Turn off classification in your next parameter file, and proceed. “flgClassify=0”

```
>$ emClarity avg param5.m 5 RawAligment
```

Notes:

- In addition to skipping the class average alignment, where class averages are aligned together, I’ve also currently disabled the multi-reference alignment approach. While I think there may be benefits, I think it is generally easier to split your project for further refinement as follows. Lets say you have two major classes you’d like to keep, like a rotated and un-rotated 40s subunit of the ribosome. You would

- first perform the steps in the two preceding sections removing all classes that aren't your rotated class.
- Copy your subtomoMeta to a new file (you'll change "subTomoMeta=projectName_rotated" in your new parameter file for this "sub-project"

```
>$ cp projectName.mat projectName_rotated.mat
```

- Now you reset the original project to where you were at prior to removing these classes

```
>$ emClarity geometry param4.m 4 ClassAlignment RemoveClasses undo STD
```

- Go back and remove now all but the non-rotated classes and proceed, now naming something like projectName_nonRotated.mat

- You'll be able to run these in the same directory, as all output includes the current projects name.

I am fully aware that this seems like a very-roundabout way to do things, and it is! I apologize for the complexity, but it is there in order not to break many currently "hidden" capabilities in emClarity, which we are in effect side-stepping with this procedure.

Appendix F: File system hierarchy

This section is intended to describe the layout of the project directory that you will establish to use with emClarity. As the software matures, I hope that some of the specific naming conventions and path requirements may be relaxed, but for now many of the programs look along the relative paths and for specific naming conventions described presently.

8.0 The project directory

This can be named anything you would like – in this document, I've called it "projectDir", but something a bit more descriptive is probably useful. Each subheading **3.X** is on the level directly below projectDir/3.X

8.1 rawData

Naming convention: free, emClarity does not directly read from this directory.

Sub-directories: **imodAliXX**, directories used to keep the results from the coarse imod alignment.

Files:

Tilt-series with frames aligned. Any naming convention is okay.

8.2 fixedStacks

Naming convention: strict, emClarity looks for files on this path.

Sub-directories: **ctf**

Files:

- **tiltXX.fixed**, if the raw data have had the statistical outliers removed, this can just be a pointer (link) to the appropriate raw data renamed. A simple numbering scheme is advised, the prefix is flexible, but no underscores or other special characters. If the raw data are un-modified, then this would be the cleaned up copy of that data.
- **tiltXX.tlt,** The tilt angles listed one per line corresponding to the same positions in the tiltXX.fixed stack.
- **tiltXX.order,** the tilt angles listed one per line corresponding to the order of data collection. emClarity uses this to apply the appropriate dose weighting. Only the angles are needed here, the dose is calculated internally.
- **tiltXX.xf,** the imod transform file with six entries per line, one line per tilt. The first four correspond to a transformation matrix describing in-plane rotation, scaling (magnification), and skew. The last two columns are X and Y shifts.
- **tiltXX.local,** optional assuming there are enough gold-beads to get an initial local alignment; if not, this file need not be present. Local alignments will still be calculated during tomoCPR.

Note: All the information in the files 2-4 are not used following the initialization of the project as they are incorporated into the tiltXX_aliN_ctf.tlt files described in the next

section. The local alignments, if present, are used until the first round of tomoCPR is run.

8.2.1 **ctf**

Naming convention: automatic, this directory is created by emClarity. It is updated throughout the sub-tomogram workflow.

Sub-directories: non-specified.

Files:

- **tiltXX_aliN_ctf.tlt,** then “N” is equal to the number of completed tomoCPR runs + 1. So before any tomoCPR has been run N=1. The initial copy is generated by “emClarity ctf estimate” and is read into the subTomoMeta binary by “emClarity init”.
    a. Columns
        1. Tilt-series index, this file is arranged with the highest tilt angles first which is a hangover from an early convention of mine that isn’t strictly necessary, and will probably be changed at some point.
        2. X-shift
        3. Y-shift
        4. Tilt angle
        5. ?
        6. ?
        7. Xform_1
        8. Xform_2
        9. Xform_3
        10. Xform_4
        11. Cumulative electron dose, this is calculated by “emClarity ctf estimate” based on the total dose, assumed to increase steadily

with time, unless the total dose is negative in the parameter file, then a 1/cos scheme is considered.

12. Astigmatism in meters. 2x + average defocus = major axis
13. Astigmatism angle in radians. CTFFIND4 convention.
14. Scale factor – not used presently
15. Average defocus in meters.
16. Pixel size in meters.
17. Spherical aberration in meters.
18. Electron wavelength in meters.
19. Fraction of amplitude contrast.
20. Tilt-series X dimension in pixels
21. Tilt-series Y dimension in pixels
22. Tilt-series Z dimension in pixels

- **tiltXX_psRadial.pdf**, This is a basic plot of the measured (blue) and fit defocus (green) for the initial ctf estimate. This value is used for every tilt until “emClarity ctf refine” is run.

Note 1 that the amplitudes are scaled relatively and don’t carry any absolute meaning. Additionally, the background is still estimated by fitting knots to the expected zeros for the ctf which can produce non-sensical results if an inaccurate defocus is found.

Note 2 the fitting is generally robust, but at times fails, and this most often can be remedied by restricting the search range specified. Fortunately, this range is usually easy to determine using the next file.

- tiltXX_ccFIT.pdf, This is a plot of the cross-correlation score found at each tested defocus. If the fit in the radial plot looks poor, this file will usually reveal a lower but much “cleaner” peak around which the search range may be restricted, and ctf estimate run again.aliStacks
- ?

8.3 aliStacks

Naming convention: strict, emClarity looks for files on this path. Created by emClarity during “emClarity ctf estimate”. Updated during “emClarity ctf update”.

Sub-directories: **none**

Files:

- **tiltXX_aliN.fixed**, these are stacks resampled in the Fourier domain and must be updated after each round of tomoCPR. N = number tomoCPR complete + 1.

8.4 cache

Naming convention: strict, emClarity looks for files on this path. Created automatically by emClarity if needed.

Sub-directories: **none**

Files:

- **tiltXX_aliN_binX.fixed** – binned stacks when sampling rate > 1.
- **tiltXX_aliN_binX.rec** – tomograms.
    a. emClarity templateSearch automatically reconstructs non-ctf corrected tomograms, which are deleted at the end of the run.
    b. emClarity ctf 3d creates persistent corrected tomos. Note these are automatically deleted by emClarity ctf update so that multiple rounds of tomoCPR can be run at the same sampling rate if desired.

Note: if a reconstruction is present at the current binning, emClarity ctf 3d will skip its reconstruction, hence the automatic deletion. This also means that if you abort a run, you may need to manually delete a partially complete reconstruction. That is, IMOD writes the header first, and places slices into the file as they are back projected, leaving a corrupt file if a run is pre-maturely exited.

8.5 recon

Naming convention: strict, emClarity looks for files on this path. Created by "recScript2.sh" shell script.

Sub-directories: **none**

Files:

- **tiltXX_M_recon.sh** – relates initial reconstruction parameters for each sub-region selected in the bin10 step. When running emClarity templateSearch, M corresponds to the tomogram number.
- **tiltXX_recon.coords** – this holds the information for each reconstructed subregion in a given tilt-series. This is the file read into the subTomoMeta and is used whenever a tomogram is made.

Note: this file may be renamed or deleted prior to running emClarity init in order to remove an entire tilt-series from further analysis.

convmap

Naming convention: strict, emClarity looks for files on this path. emClarity creates a directory convmap_wedgeTypeX_binX which allows multiple template matching results to be retained at the same time. This may be renamed or linked to.

Sub-directories: **none**

Files:

- **tiltXX_M_binX_convmap.mrc**
- **tiltXX_M_binX_angles.mrc**
- **tiltXX_M_binX.mod**
- **tiltXX_M_binX.pos**
- **tiltXX_M_binX.csv**

8.6 bin10

Naming convention: free, emClarity does not directly use this directory, instead the supplied script, recScript2.sh which you are free to modify creates and uses this to aide you in marking out subregions to reconstruct.

Sub-directories: **none**

Files: