emClarity tutorial

April 8, 2020

Contents

List of Figures

List of Tables

1 How to use this guide

1.1 Run the jobs

The main purpose for this tutorial is to provide the minimal set of instructions needed to begin processing sub-tomogram data using emClarity. We will not introduce all of the methods that emClarity has at its disposal, but we will do our best to at least let you know what is and isn't possible to do. If at any point you are confused, or something seems to not work as you expect, you might find more information on the wiki; feel free to search the mailing list archive, or post new questions to the community forum, hosted on google groups, should you have any questions you cannot resolve on your own.

Tip: *To display every procedure available, run* `emClarity help` *from the command line.*

1.2 Algorithms

emClarity source code is available online, and we encourage you to go through the code to look at the algorithms we are using. Unfortunately, this can be challenging work because emClarity is currently expanding; it is likely that some big part of the code will change. In this context, this tutorial can also help you stay up to date on what emClarity is doing behind the scenes.

Section **15** contains the algorithms for each section presented in this tutorial. Please keep in mind that these are simplified descriptions of what emClarity is actually doing, as we often don't mention the details that were implemented to make the code more efficient.

1.3 Installation and Requirements

This tutorial will not talk about the installation process nor the software and hardware requirements. If you are looking for this, you can find more information on the wiki.

1.4 Parameter files

emClarity is currently using a parameter file to manage inputs. You can find an example here.

2 Workflow

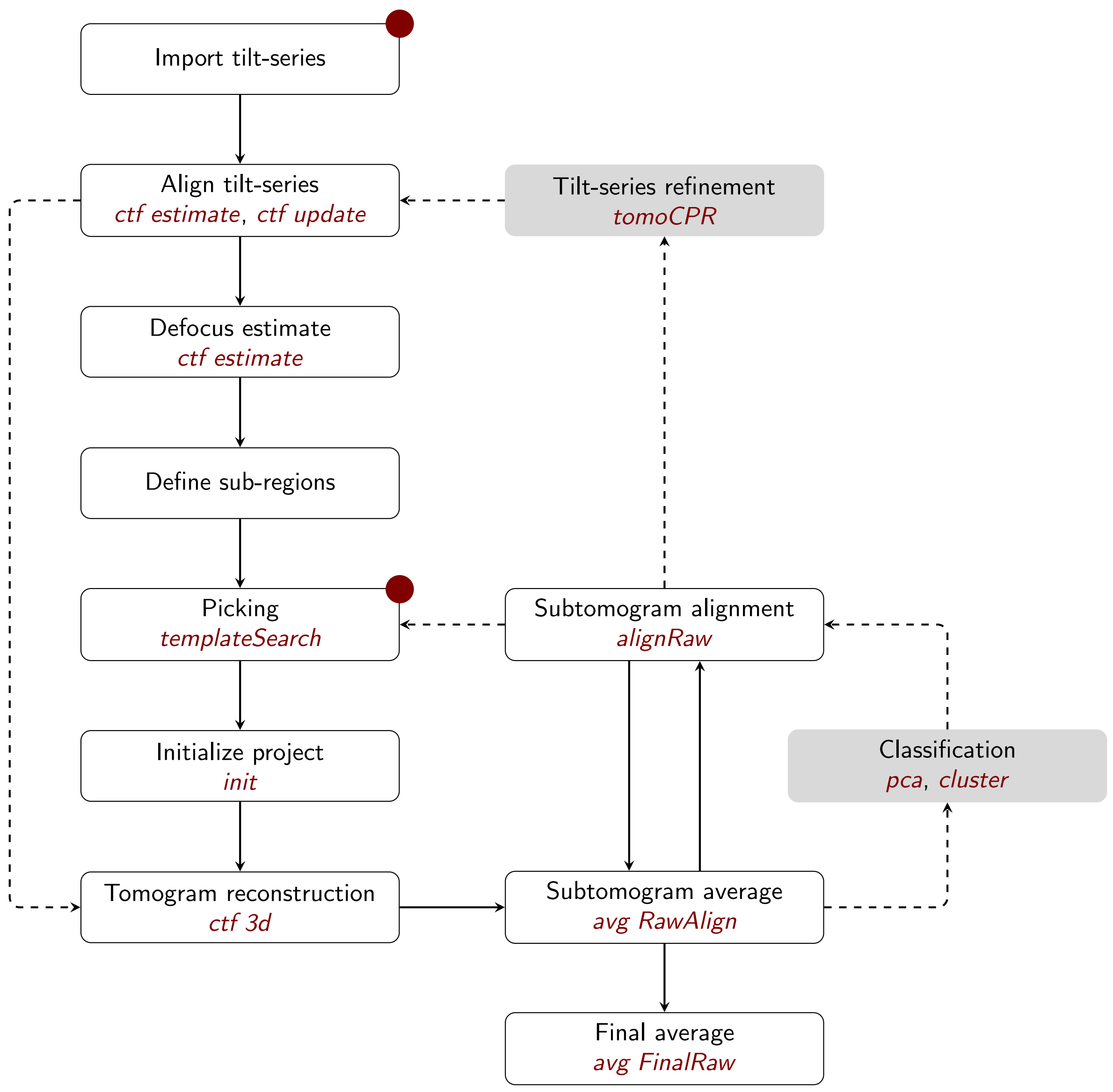

Figure 1: emClarity workflow. In most cases, the aligned tilt-series are the only data you need to start with, as the (sub)tomograms will be generated at later stages. The classification and tilt-series refinement steps are optional. The red markers indicates the possibility to import data from other software (see section **4.3** and **7** for more details).

3 The project directory

emClarity needs to be run from the "project directory", referred to here as `<projectDir>`. Every output will be saved in this directory. As we go along, we'll present in more detail each sub-directory and their content.

- `<projectDir>`:
  - **name**: anything you like.
  - **description**: contain every emClarity input directories (listed below) and every outputs. You must run emClarity from this directory.
  - **created by**: you.
- `<projectDir>/fixedStacks`:
  - **name**: fixed.
  - **description**: contain the raw (*not* aligned) tilt-series (`*.fixed`) and the initial tilt-series alignment files (`*.xf`, `*.tlt` and optionally `*.local` and `*.erase`). See section 4.3 for more details.
  - **created by**: you.
- `<projectDir>/fixedStacks/ctf`:
  - **name**: fixed.
  - **description**: created by `ctf estimate` (section 5) and updated after tilt-series refinement by `ctf update` (section 12). It contains the radial averages (`*_psRadial1.pdf`) and stretched power spectrum (`*_PS2.mrc`) computed by `ctf estimate`, as well as some metadata (`*_ctf.tlt`) used throughout the entire workflow.
  - **created by**: emClarity.
- `<projectDir>/aliStacks`:
  - **name**: fixed.
  - **description**: created by `ctf estimate` (section 5) and updated after tilt-series refinement by `ctf update` (section 12). It contains the aligned, dose-weighted tilt-series. These stacks are mostly used by `ctf 3d` to compute the tomograms at different binning.
  - **created by**: emClarity.
- `<projectDir>/cache`:
  - **name**: fixed.
  - **description**: created automatically by emClarity if needed. Store any stack or reconstruction for the current binning. If a reconstruction is present at the current binning, `ctf 3d` will skip its reconstruction. This also means that if a reconstruction is aborted, you may need to manually delete a partially complete reconstruction. That is, tilt 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.

  - **created by**: emClarity.

- `<projectDir>/convmap`:
  - **name**: fixed.
  - **description**: When creating a project with `init` (see section 8), emClarity will look in this directory to grab outputs from `templateSearch` (see section 7). If you pick your particles with emClarity, the content of this directory is generated by `templateSearch`.
  - **created by**: you.

- `<projectDir>/recon`:
  - **name**: fixed.
  - **description**: Created during section 6. It holds the information for each reconstructed sub-region in a given tilt-series. The `*_recon.coords` files are read into the metadata created by `init` and is used whenever a tomogram is made.
  - **created by**: emClarity.

- `<projectDir>/<binX>`:
  - **name**: anything you like.
  - **description**: emClarity does not directly use this directory, but it is used by `recScript2.sh` to define the sub-regions (see section 6 for more details).
  - **created by**: you.

- `<projectDir>/FSC`:
  - **name**: fixed.
  - **description**:
  - **created by**: emClarity.

4 Get your data ready

4.1 Download tutorial

In this tutorial, we will use the 12 tilt-series of the fast-incremental single-exposure (FISE) data deposited on EMPIAR-10304 from [1]. To download the motion-corrected tilt-series, it is advisable to use the "Aspera Connect" option or alternatively you can pull directly from the ftp server with the following command:

```
#!bin/bash
for nb in $(seq 1 12); do
  wget -b ftp://ftp.ebi.ac.uk/\
  pub/databases/empiar/archive/10304/data/tilt${nb}.mrc
done
```

– **Sample**:	70S ribosomes, 10nm colloidal gold fiducials.
– **Tilt-scheme**:	Dose-symmetric starting from 0°, 3°increment, ±60°, 120e/Å^2 total exposure.
– **Instruments**:	Krios with single-tilt axis holder equipped with a Gatan K3 direct electron detector. 2.1Å/pix. ~176° tilt axis rotation.

Table 1: Tutorial data-set

You should be able to get a ~5.9Å map from this tutorial.

Note 1: *For this tutorial, we don't necessarily recommend to align the 12 tilt-series, as it can be quite redundant, especially if you are familiar with this step. For beginners, we do recommend to at least try aligning one tilt-series with ETomo. In any case, you can find the alignment files here.*

4.2 Tilt-series alignment

This tutorial is not about tilt-series alignment, but we do recommend using the fiducial alignment procedure from the ETomo pipeline (*autofidseed*, *tiltalign*, etc.). One powerful option of this pipeline is to be able to solve for a series of local alignments using subsets of fiducial points, which can then be used by emClarity (via the IMOD *tilt* program) to reconstruct the tomograms.

During the alignments, you should keep an eye at the ratio of known measurements (fiducials) and unknown parameters (shifts, rotations, tilts and magnifications) you try to solve for. If this ratio is too low, it means the program is likely to accommodate random errors to solve for the model you are asking for. In face of such low ratio, you can simplify the alignment model during the refinement by grouping and/or fixing the variables (see *restrictalign* "OrderOfRestrictions" for more details).

You are free to use another software, but it is likely to require more efforts to integrate your data into emClarity (see the next **4.3** section). We do not plan to support imports from tomography software other than IMOD in the near future.

Note 2: *If you align the tutorial data with ETomo, you may have to change the extension from .mrc to .st (IMOD naming convention for stack of images) depending on the version* *installed on your machine. Moreover, you will have to specify the initial tilt angles using the ETomo setup interface, or using a rawtlt file (see an example here), or by adding the tilts to the extended header.*

Note 3: *The IMOD batchruntomo interface encapsulates the ETomo pipeline and runs the operations required to align the tilt-series. To deal with low ratio, batchruntomo is applying successive restrictions to the alignment. The default behavior is to group the rotations, then group the magnifications, then fix the tilt angles, then solve for only one rotation and then fix the magnification.*

Tip: *You might have noticed than some images from the tutorial data set are (mostly) blank. These images can be ignored during alignment (ETomo: "view to skip" option). We don't recommend to remove these images from the stack as it is possible to remove them with emClarity (see section **5**) while accounting for the cumulative electron dose.*

4.3 Transfer to emClarity

My tilt-series alignments are finally done, so how to transfer them to emClarity now? Please pay careful attention to the naming conventions in this section, as there are used throughout the pipeline.

For each tilt-series, emClarity needs:

- `<prefix>.fixed`: the raw (*not* aligned) tilt-series. These should *not* be dose-weighted nor phase flipped. They correspond to the original `tilt1.mrc` to `tilt12.mrc` stacks available in EMPIAR. If you did any preprocessing (X-ray removal, etc.), you should use these.

  Note 4: *While it will produce substandard results, if you have no option but to use images which are already exposure filtered, please add the following to your paramter file:* `applyExposureFilter=0`

- `<prefix>.xf`: the file with alignment transforms to apply to the `<prefix>.fixed` stacks. This file should contain one line per view, each with a linear transformation specified by six numbers. The first 4 numbers is the 2x2 rotation matrix (in-plane rotation, scaling/magnification) and the last 2 numbers are the X and Y shifts (in un-binned pixels). See section **5** for more details.

  If you did the alignment with eTomo, it corresponds to "OutputTransformFile" from *tiltalign* which is set by default to `<prefix>_fid.xf` (fiducial alignment) or `<prefix>.xf` (fiducialess alignment).

- `<prefix>.tlt`: the file with the solved tilt angles. One line per view, angles in degrees.

  If you did the alignment with eTomo, it corresponds to the OutputTiltFile from *tiltalign*, which is by default `<prefix>_fid.tlt` (fiducial alignment) or `<prefix>.tlt` (fiducialess alignment).

- **(optional)** `<prefix>.local`: the file of local alignments. This file is similar to the `<prefix>.xf` file, but contains one transformation per view and per patch, plus an additional header per patch. See the *tiltalign* documentation for more details.

  If you did the alignment with eTomo, it corresponds to the "OutputLocalFile" from *tiltalign*, which is by default `<prefix>local.xf`.

- **(optional)** `<prefix>.erase`: the file with the coordinates (in pixel) of the fiducial beads to erase before ctf estimation. These coordinates must correspond to the aligned stack. One line should contain the x, y and z (view) coordinates of the beads you wish to remove. Alternatively, you can remove the beads from the raw `<prefix>.fixed` before importing them to emClarity.

  If you did the alignment with eTomo, it corresponds to the `<prefix>_erase.fid` file.

These files should be copied to `<projectDir>/fixedStacks` (see section 3).

Tip: *You don't necessarily need to copy the tilt-series to the* `fixedStacks` *directory; use soft links:* `ln -s <...>/<prefix>.mrc <...>/fixedStacks/<prefix>.fixed`

Tip: *It is best practice to work the whole way through the workflow with the smallest data-set as possible and once you checked that everything holds, then process your full data. The same approach may be taken with this tutorial; it should be possible to obtain a low-resolution but recognizable 70S ribosome with only two or three of the tilt-series. This will confirm that everything "runs" (producing some output) in your hands and on your gear. Only after this it does make sense to then confirm you are able to produce the correct output, i.e. a 7Å map.*

5 Defocus estimate

5.1 Objectives

There are two main objectives. First, create the aligned dose-weighted stacks. These stacks will be then used to compute the tomograms at later stages. Second, estimate the defocus of each view of the stack (two defoci and the astigmatism angle, per view).

5.2 Parameters

Table 2: `ctf estimate` parameters. Your parameter file should have the following parameters.

Microscope settings	
– `VOLTAGE`*	Accelerating voltage of the microscope in Volts (e.g. 300e3).
– `Cs`*	Spherical aberration of the microscope in meters (e.g. 2.7e-6).
– `AMPCONT`*	Percent amplitude contrast (e.g. 0.09).
– `PIXEL_SIZE`*	Pixel size in meter per pixel (e.g. 1.8e-10). Must match the header of the stacks in `fixedStacks/*.fixed`.
Fiducials	
– `beadDiameter`	Diameter of the beads to erase, in meter (e.g. 10e-9). This parameter is used if fiducial beads need to be erase, thus only for the stacks that have a `fixedStacks/*.erase` file.
Tilt-scheme	
– `CUM_e_DOSE`*	Total exposure in e/Å^2.
– `doseAtMinTilt`*	The exposure each view receive (should be about `CUM_e_DOSE` / nb of views), in e/Å^2.
– `oneOverCosineDose`*	Whether or not it is a Saxton scheme (dose increase as 1/cos(α), α being the tilt angle); this will scale `doseAtMinTilt` according to the tilt angle (e.g. 0).
– `startingAngle`*	Starting angle, in degrees (e.g. 0).
– `startingDirection`*	Starting direction; should the angles decrease or increase (neg or pos).
– `doseSymmetricIncrement`*	The number of tilts before each switch in direction. 0=false, 2="normal" dose symmetric. ***Note 5:*** *The original dose symmetric scheme included 0 in the first group. For this, specify your doseSymmetricIncrement as a negative number.*
Defocus estimate	
– `defCutOff`*	The power spectrum used by `ctf estimate` is considered from slightly before the first zero past the first zero to this cutoff, in meter (e.g. 7e-10).

– defEstimate*	Initial rough estimate of the defocus, in meter. With defWindow, it defines the search window of defoci.
– defWindow*	Defocus window around defEstimate, in meter; e.g. if defEstimate = 2.5e-6 and defWindow = 1.5e-6, try a range of defocus between 1e-6 to 4e-6.
– deltaZtolerance	Includes the tiles at the tilt-axis $\pm\Delta Z$, in meter. See section 15.2 for more details. Default=50e-9.
– zShift	Used for the handedness check. Shift the evaluation region ($Z_{tilt-axis}$ $\pm$ deltaZtolerance) by this amount. See section 15.2 for more details. Default=150e-9.
– ctfMaxNumberOfTiles	Limits the number of tiles to include in the power spectrum. The more tiles, the stronger the signal but the longer it takes to compute the power spectrum. Default=3000.
– ctfTileSize	Size of the (square) tiles, in meter. Default=680e-10.
– paddedSize	The tiles are padded to this size, in pixel, in real space before computing the Fourier transform. Should be even, large (compared to the tiles), and preferably a power of 2. Default=768.

5.3 Run

The `ctf estimate` routine has the following signature:

```
>> emClarity <version> ctf estimate <param> <prefix>
```

`<param>` is the name of the parameter file (e.g. `param_ctf.m`), and `<prefix>` is the base-name of the tilt-series in `<projectDir>/fixedStacks` you wish to process.

For example, to run `ctf estimate` on the first tilt-series of the tutorial:

```
>> emClarity 150 ctf estimate param_ctf.m tilt1
```

If you have many tilt-series and you don't want to run all of them individually, you can do the following. This will select every available stack to emClarity and run `ctf estimate` on each one of them.

```
#!/bin/bash
for stack in fixedStacks/*.fixed; do
    prefix=${stack#fixedStacks}
    emClarity 150 ctf estimate param_ctf.m ${prefix%.fixed}
done
```

You may have noticed that most stacks have their first image blank. Fortunately, `ctf estimate` can remove images from the stack. For instance, to remove the first view of til1, run:

```
>> emClarity 150 ctf estimate param_ctf.m tilt1 [1]
```

and to remove the first and last view of tilt11, run:

```
>> emClarity 150 ctf estimate param_ctf.m tilt11 [1,41]
```

> ***Note 6:*** *You should remove the first view from the tilt-series tilt1 to tilt10, and the first and last views from tilt11 and tilt12.*

5.4 Outputs

You should make sure the average defocus (at the tilt axis) was correctly estimated. The best way to check this is to:

1. Open `fixedStacks/ctf/<prefix>_ali1_psRadial_1.pdf` and check that the theoretical CTF estimate (green) matches the radial average of the power spectrum of the tilt-series (black). Note that the amplitude doesn't matter here, the phase on the other hand, does.
2. If they don't match, it is likely that you will need to adjust the `defEstimate` and `defWindow` parameters. Open `fixedStacks/ctf/*_ccFIT.pdf`, which plots the cross-correlation score as a function of defocus. There is often an obvious correct peak, smoothly rising and falling. If you don't see this peak, try to change the sampled defoci with `defEstimate` ±`defWindow` and re-run `ctf estimate`

Note 7: *For a full description of the outputs generated by* `ctf estimate`, *you should refer to section* ***15.2***.

6 Select sub-regions

6.1 Objectives

Quite often, the regions of interests don't take up the entire field of view. To speed things up and save memory, you can select the sub-regions you would like emClarity to focus on. These sub-regions can be changed until the project is initialized (see section 8).

You may notice as we go through this tutorial taht we often thinks in term of sub-regions and not in terms of the full tomograms. For example, during picking (see section 7), each sub-regions is processed independently from the others and if you decide to ignore one sub-regions from a tomogram, you can.

6.2 Create the boundaries for each sub-regions

- Download from the repository the recScript2.sh script and copy it inside the project directory.
- Prior to selecting the sub-regions, we need to create the tomograms containing the entire field of view. Running the following command from the project directory will do exactly that.

  ```
  ./recScript2.sh -1
  ```

  By default, this will create a new directory, called `<projectDir>/bin10`, which will have a bin 10 tomogram for every stack created by `ctf estimate` (i.e. a tomogram for every `aliStacks/*_ali1.fixed`).

- Now that we have the tomograms, we can start thinking about the sub-regions we want to select. The goal is to define 6 points which defines the boundaries of one sub-regions (x_{min}, x_{max}, y_{min}, y_{max}, z_{min} and z_{max}). So if you have 3 sub-regions you are interested in for a particular tomogram, you will need to define $6 \times 3 = 18$ points.

  This is most easily done by creating an IMOD model:

  - Go in `<projectDir>/bin10` and open the first tomogram with *3dmod* by running:

    ```
    3dmod tilt1_bin10.rec
    ```

  - Select the "Model" mode and select the 6 points in the order specified above. Each point must be in its own contour. For the tutorial dataset, as the ribosome are homogeneously spread across the entire tomogram, we recommend to divide the tomograms into 2 sub-regions of equal size.
  - Save the model (`File` → `Save model`) with the same name as the tomogram but with the `.mod` extension.
  - Repeat for each tomogram.

  At the end of this step, you should have in `<projectDir>/bin10` one `<prefix>.mod` file per tilt-series you whish to process.

  Note 8: *Each sub-regions will need to be transferred to GPU memory. Therefore, you should be careful not to select too big sub-regions as there might not be enough memory to hold them, especially at bin 1. If the particles of interest are homogeously spread, we recommend to divide the field of view into 2 equal sub-regions.*

- This `bin10` directory is not directly used by emClarity. You'll need to convert the `<prefix>.mod` files you just created into something emClarity understands. You can do this on the first stack by running:

  ```
  ./recScript2.sh tilt1
  ```

  Or to convert every sub-regions of every tomograms, run:

  ```
  #!/bin/bash
  for stack in bin10/*.mod; do
      prefix=${stack#bin10} ; ./recScript2.sh ${prefix%.mod}
  done
  ```

  This will create a directory called `<projectDir>/recon` with the file emClarity needs to extract the sub-regions you selected:

  Table 3: `recon/<prefix>_recon.coords`

Line	*tilt* parameter	Description
1		`<prefix>`; stack prefix.
2		Number of sub-regions within this stack.
3	`WIDTH`	Width in X of the first sub-region, in pixel.
4	`SLICE` 1	Starting Y coordinate of the first sub-region. Starts from 0.
5	`SLICE` 2	Ending Y coordinate of the first sub-region. Starts from 0.
6	`THICKNESS`	Thickness in Z of the first sub-region, in pixels.
7	`SHIFT` 1	Shift in X the reconstructed slice, in pixel. If it is positive, the slice will be shifted to the right and the output, the first sub-region, will contain the left part of the whole potentially reconstructable area.
8	`SHIFT` 2	Shift in Z the reconstructed slice, in pixel. If it is positive, the slice is shifted upward.
...	...	Same as line 3 to 8, but for the next sub-regions, if any.

  Note 9: *If you change the coordinates of the sub-regions, you must do it before running* `init` *(see section **8**) and you always need to refresh the* `<projectDir>/recon` *directory by re-running* `./recScript2.sh <prefix>`.

7 Picking

7.1 Objectives

It's time to pick the particles, i.e. sub-tomograms. There are many ways to pick particles, but they usually all rely on the tomograms. As such, each particle can be described by its x, y, z coordinates and ϕ, θ, ψ Euler angles. emClarity has a template matching routine that can pick the sub-tomograms for you, but it requires a template.

7.2 Parameters

Table 4: `templateSearch` parameters. Your parameter file should have the following parameters.

Microscope settings	
– `PIXEL_SIZE`*	Pixel size in meter per pixel (e.g. 1.8e-10). Must match the header of the stacks in `fixedStacks/*.fixed`.
– `SuperResolution`*	Whether or not the raw stacks are in super-resolution (e.g. 0). If true/1, the `PIXEL_SIZE` is multiplied by 2, but we still expect the `aliStacks/*.fixed` stacks to be Fourier cropped already.
Particle	
– `particleRadius`*	Particle radii, in Å. Format is $[R_X,\ R_Y,\ R_Z]$. In this context, it defines the size of the cross-correlation peaks to select. See `Peak_mRadius` for more details.
– `Ali_mRadius`*	Alignment mask radii, in Å. Format is $[R_X,\ R_Y,\ R_Z]$. In this case, it is used to pad/trim the template to this size.
Template matching	
– `Tmp_samplingRate`*	Sampling (i.e. binning) at which the sub-region should be reconstructed to perform the template matching. The sampling rate should be chosen to give a running pixel size between 8 and 12Å/pix.
– `Tmp_angleSearch`*	Angular search, in degrees. Format is $[Range_{out},\ Step_{out},\ Range_{in},\ Step_{out}]$. For example, [180, 15, 180, 10], specifies a $\pm180°$ out of plane search (polar and azimuth angles) with 15° steps and $\pm180°$ in plane search (planar angles) with 10° steps. If you have a particle with C6 symmetry (with the symmetry axis corresponding to the Z-axis) you might search a more limited range, like [180,15,36,9], but for particles like ribosomes, there are no real constraints on the orientation, so searching a full grid in 12 or 15 degree increments is required.
–`Tmp_threshold`*	Number of particle to pick. This will be override by the command line argument `<threshold>` (see section 7.3).
–`Tmp_targetSize`	Size, in pixel, of the chunk to process. If the sub-region is too big, the processing will be split into individual chunks. Format is $[X, Y, Z]$. Default=[512,512,512].
–`lowResCut`	The sub-region is filtered before performing the template search and this defines the resolution (in Å) at which the Gaussian low-pass filter begins. Default=estimate of first zero of the CTF.

`–Tmp_medianFilter`	Apply a median filter to the sub-region before computing the cross-correlation. It defines the size of the kernel, either 3, 5 or 7. Default=false.
Cross-correlation	
`–Peak_mType`	Type (i.e. shape) of the cross-correlation peaks. Can be sphere, cylinder or rectangle. See section 15.3 for more details. Default=sphere.
`–Peak_mRadius`	Radius of the cross-correlation peaks, in Å. Format is $[R_X,\ R_Y,\ R_Z]$. See section 15.3 for more details. Default= $0.75 \times$ `particleRadius`.

7.3 Run

Before running the `templateSearch`, you need to **prepare a template**. This template should have the same pixel size as the `PIXEL_SIZE` parameter. It doesn't need to be low-pass filter, as emClarity will do it internally. If you want to re-scale a map, you can run:

```
>> emClarity 150 rescale <inName> <outName> <inPixel> <OutPixel>
   <method>
```

`<inName>` and `<outName>` are the name of your template and the output name of the re-scaled template, respectively. `<inPixel>` is the pixel size of your template and `<OutPixel>` is the desired pixel size. `<method>` can be GPU or cpu.

***Note 10:** We do provide a template for this tutorial, but any 70S ribosome map should work.*

The `templateSearch` routine has the following signature:

```
>> emClarity 150 templateSearch <param> <prefix> <region>
   <template> <symmetry> <threshold> <GPU>
```

`<param>` is the name of the parameter file, `<prefix>` is the base-name of the tilt-series in `<projectDir>/aliStacks` to process. `<region>` is the number of the sub-region to process (see section 6). If you have symmetry, C6 for example, `<symmetry>` should be 6. This will randomize the Euler angles to any one of the symmetry related positions, which helps to reduce wedge bias. At the same time, you can reduce in the $Range_{in}$ of the `Tmp_angleSearch` parameter to ∼36°. For ribosomes, it should be 1. `<threshold>` corresponds to the `<Tmp_threshold>` parameter and `<GPU>` is the GPU ID to use (starting from 1).

For example, to run `templateSearch` on the first tilt-series of the tutorial, where we defined 2 sub-regions:

```
# First region
>> emClarity 150 templateSearch param.m tilt1 1 template.mrc 1
   300 1
# Second region
>> emClarity 150 templateSearch param.m tilt1 2 template.mrc 1
   300 2
```

7.4 Outputs

The primary goal now is to remove false positives due to strong homogeneous features like carbon edges, membranes, or residual gold beads. The template matching produces a "cumulative correlation" map, which can be opened alongside a 3d model of the selected peaks.

In `<projectDir>/convmap_wedgeType_2_bin<X>`, with `<X>` equal to `Tmp_sampling`:

- To look at the 3d IMOD model containing the coordinates of the selected particles, you can open the model with `3dmodv`. However, if your particles aren't organized in a lattice, it is quite difficult to do anything with this.
- Open `<prefix>_<region>_bin<X>_convmap.mrc` overlapped with `<prefix>_<region>_bin<X>.mod` to look and remove particles. Another possibility is to use the binned tomogram in `cache/<prefix>_<region>_bin<X>.rec`.

  ```
  3dmod <*>_convmap.mrc <*>.mod
  # or
  3dmod ../cache/<*>.rec <*>.mod
  ```

- You can always change the sampling, the angular search, the template or the threshold and re-run `templateSearch`, if you are not satisfied with the results. If you want to change the sub-region coordinates, change the boundaries in `../bin10/<prefix>_bin10.mod` and re-run the `recScrip2.sh` script as explained in section **6**.
- TODO Remove neighbours
- TODO Constrains

This directory cannot be seen by emClarity as it currently is. You will need to rename it to `<projectDir>/convmap` before going to the next step.

Note 11: *For a full description of the outputs generated by* `templateSearch`*, you should refer to section* ***15.3****.*

Note 12: *Changing the contours coordinates or adding new contours in the* `.mod` *file has no effect. You can only remove contours. See section* ***8*** *for more details.*

Note 13: *The* `wedgeType_2` *prefix comes from the an old version of emClarity and indicates the type of missing wedge mask to apply to the tomogram and the template before computing the cross-correlation scores. This parameter is no longer used.*

8 Initialize the project

8.1 Objectives

8.2 Parameters

8.3 Run

8.4 Outputs

9 Reconstruct the tomograms

9.1 Objectives

9.2 Parameters

9.3 Run

9.4 Outputs

10 Sub-tomogram averaging

10.1 Objectives

10.2 Parameters

10.3 Run

10.4 Outputs

11 Sub-tomogram alignment

11.1 Objectives

11.2 Parameters

11.3 Run

11.4 Outputs

12 Tilt-series refinement

12.1 Objectives

12.2 Parameters

12.3 Run

12.4 Outputs

13 Classification

14 Final reconstruction

15 Algorithms

15.1 Conventions

15.2 Defocus estimate

15.2.1 Transform the tilt-series

1. **Pre-processing** (for each view):
    (a) Low-pass filter at the Nyquist frequency.
    (b) Replace hot pixels (X-rays, etc.) with random scaled values.
2. **Transform** (for each view):
    (a) Load `fixedStacks/<prefix>.xf`, decompose the rotation matrix into magnification and rotation, and get the X and Y shifts.
    (b) Pad the view by a factor of 2 (if `SuperResolution`=0), compute the forward Fourier transform, apply the magnification, rotation and shifts to the transform, Fourier crop by a factor of 2 and compute the inverse Fourier transform to switch back to real space. Increasing the sampling for the transformation reduces noise due to aliasing.
    (c) Apply the same transformation to a mask of ones of the same size as the view. This mask is used to track down non-sampled regions of the view. The stack of masks (1 mask per view) is saved as `aliStacks/<prefix>_ali1.samplingMask`.
3. **Erase the beads** (optional):
   Try to load `fixedStacks/<prefix>.erase`. If it exists, replace the pixels of the aligned stack, at the coordinates specified in this file, by random scaled values.
4. **Post-processing** (for each view):
    (a) Remove hot pixels/outliers from the sampled region, ignoring un-sampled pixels thanks to the `samplingMask` (step **2.c**).
    (b) Scale down the intensity of each view relative to its cumulative dose and thickness (the thicker, the less elastic scattering, the less the projection should weight in the reconstruction). Estimate the thickness of the tomogram to 100nm (this will be later refined during `tomoCPR`, section 12) and mean-free path for in-elastics scattering of 400nm.
5. **Save aligned stacks**:
   Save the aligned, dose-weighted, bead-erased tilt-series as `aliStacks/<prefix>_ali1.fixed`. These stacks are used by `templateSearch` (section 7) and `ctf 3d` (section 9) to reconstruct the tomograms.

15.2.2 Defocus search

1. **Periodigram averaging**:
    (a) There are many defoci in a tilted image, which ultimately reduce the number of Thon rings in the power spectrum of the entire image, thus the accuracy of the ctf estimate. To reduce the negative interference between regions with different defoci (i.e. with different Z in the microscope), we exclude from the calculation of the power spectrum, and for each view, the regions that are too high or too low relative to the tilt axis. So, if the tilt

axis is along the Y axis, the selected coordinates satisfy:

$$-\texttt{deltaZtolerance} > -tan(\alpha) \times X_{coords} > +\texttt{deltaZtolerance}$$

with α being the tilt angle and X_{coords} being the pixel coordinates along the X axis, with the center of the axis (where the tilt axis is) equal to 0.

(b) Extract the tiles that are within the selected regions, pad them and compute the average 2d power spectrum of these tiles. We now have one average power spectrum of the stack, gathering the signal of only the regions close to the tilt axis height.

2. **Initial defocus search**:

   (a) This first search consists to find a first rough estimate of the average defocus of the tilt-series, at the tilt axis. Therefore, we can ignore the astigmatism for now and compute the radial averaging of the 2d power spectrum computed at the previous step.

   (b) For each defocus (`defEstimate` $\pm$ `defWindow`, 1nm increment):

      i. Compute the 1d theoretical CTF (with envelope).

      ii. Background estimate: if the current tested defocus is correct, the zeros of the CTF should be at the background level. Therefore, extract the values on the 1d radial average at the frequencies where the zeros should be, based on the theoretical CTF. Fit a smooth curve on these sampled points and use it as background estimation. The position of the zeros varies depending on the defocus, therefore the closer we get from the true defocus, the more accurate the background estimation becomes, the better the theoretical and experimental background subtracted CTFs fit.

      iii. Band-pass the theoretical CTF and 1d radial average to consider only the frequencies from slightly before the first zero to the first zero after `defCutoff`.

      iv. Subtract the background from the 1d radial average and compute the normalized cross-correlation coefficient (CC) between the theoretical CTF and the 1d radial average.

   (c) Select the defocus that gave the best fit and save the following files:

      - `fixedStacks/ctf/<prefix>_ali1_ccFIT.pdf`: the CC for each defoci that were tested.
      - `fixedStacks/ctf/<prefix>_ali1_bgFit.pdf`: the 1d radial average (before background subtraction) and the estimated background (computed using the best defocus). For visualization, the background curve is slightly shifted towards the bottom.
      - `fixedStacks/ctf/<prefix>_ali1_psRadial_1.pdf`: the background subtracted, band-passed, radial averaging (in black) and the theoretical CTF that gave the best fit (in green).

   (d) **Astigmatism search**:
   Until now, we have used the 1d radial average of the 2d power spectrum to estimate the CTF. We will now use the 2d power spectrum, computed in step **6**, in order to account for astigmatism.

      i. Background estimate: Using the defocus measured at the step **7.c**, we compute the 1d theoretical CTF to find at which frequencies the zeros are. Then, we extract lines of the 2d power spectrum at different angles. For each line, extract the values at the calculated zero positions and fit a smooth curve along these values. Average the smoothed curve of each line and fit a cubic spline to this average. This is the estimated background of the 2d power spectrum.

ii. Band-pass the power spectrum as in step **7.b.iii** and save the following files:
`fixedStacks/ctf/<prefix>_avgPS.fixed`: 2d band-passed power spectrum before background subtraction.
`fixedStacks/ctf/<prefix>_avgPS-bgSub.fixed`: 2d band-passed power spectrum after background subtraction.

iii. Coarse search. For each tested astigmatism angle α_{ast} ($\pm$45°, 10°step), for each tested Δf_1 and Δf_2 (current defocus $\pm$200nm, 15nm step):
  A. Compute the 2d band-passed CTF, with envelope.
  B. Compute the CC between the background subtracted, band-passed, 2d power spectrum and the CTF calculated at the previous step.

iv. Select the trio (α_{ast}, Δf_1 and Δf_2) that gave the best fit.

v. Fine search. Identical to the coarse search, but refine around the selected trio (α_{ast} $\pm$5°, 0.5° step and Δf_1, Δf_2 $\pm$7.5nm, 3.75nm step).

vi. Select the trio (α_{ast}, Δf_1 and Δf_2) that gave the best fit and save the following file:

Table 5: `fixedStacks/ctf/<prefix>_ali1_ctf.tlt`

Column	Description	Column	Description
1	view number	14	empty
2	X shift (in pixels)	15	defocus
3	Y shift (in pixels)	16	`PIXEL_SIZE`
4	tilt angle (in degrees)	17	`Cs`
5	empty	18	`WAVELENGTH`
6	additional rotation (90°)	19	`AMPCONT`
7	rotation matrix, a_{11}	20	Number of pixels in X
8	rotation matrix, a_{21}	21	Number of pixels in Y
9	rotation matrix, a_{12}	22	Number of sections (Z)
10	rotation matrix, a_{22}	23	...
11	cumulative dose		
12	Δf_1		
13	Δf_2		

15.2.3 Handedness check

(a) If the sample is tilted, the regions higher than the tilt axis have a smaller defocus and the regions lower than the tilt axis have a larger defocus. If it is the opposite, it means the projections were flipped or the tilt axis used for alignment is 180° off. In step **6**, we've selected regions in the same height than the tilt axis ($\pm$ `deltaZtolerance`) to reduce interference between regions with "significantly" different defoci. Here, we do the same, but we apply a Z shift (`zShift`) to only select the regions that are significantly lower than the tilt axis.

(b) Compute the 2d power spectrum of only these "low" regions and estimate the defocus as explained in step **7**. We now have an estimate of the defocus of the regions of the

specimen that were imaged while being below the tilt axis (or at least we assume so). We can call this defocus, Δf_{below}.
Save `fixedStacks/ctf/<prefix>_ali1_psRadial_2.pdf`: the background subtracted, band-passed, radial averaging of the field of view below the tilt axis (in black) and the theoretical CTF that gave the best fit (in green).

(c) Repeat the previous two steps but this time apply a Z shift in the other direction, to only select regions that are significantly higher than the tilt axis. We now have defocus estimate of the regions of the specimen that were imaged while being above the tilt axis (again, we assume so). We can call this defocus, Δf_{above}.
Save `fixedStacks/ctf/<prefix>_ali1_psRadial_3.pdf`: the background subtracted, band-passed, radial averaging of the field of view above the tilt axis (in black) and the theoretical CTF that gave the best fit (in green).

(d) The regions below the tilt axis should have a stronger defocus than the regions above the tilt axis. So, if $\Delta f_{above} < \Delta f < \Delta f_{below}$, then the handedness is probably correct. On the other hand, if $\Delta f_{above} > \Delta f > \Delta f_{below}$, the handedness is probably wrong.

15.2.4 Defocus refinement

So far, every 2d power spectra and 1d radial averages were the result of an average across the entire tilt-series. Therefore, we could only estimate an average defocus and average astigmatism. In practice, each view can be at a different defocus and have a different astigmatism. Consequently, we should work on one projection at a time.

There are two main problems to look at individual images. First, there is considerably less signal in one single image than in the entire tilt-series. Second, we mostly deal with tilted images, which inherently have less Thon rings than non-tilted images, making the ctf estimate even more complicated.

Fortunately, we can almost entirely resolve the second problem. As a matter of fact, if an image is tilted, we can account for its defocus ramp and correct for it. By stretching/compressing the power spectrum of the different sub-regions (tiles) of a tilted image, we can make them all constructively interfere with each other. As a result, the analysis of tilted specimens becomes as good as for non-tilted specimens, up to the point where the signal is getting weaker due to the increased thickness.

Padding an image in real space is effectively stretching its power spectrum. On the other hand, cropping an image in real space is effectively compressing its power spectrum. As such, with the correct padding/cropping factor, one can make two power spectra with different defoci "overlap". The tiles below the tilt axis have a stronger defocus. To make their power spectrum constructively interfere with the tiles at the tilt axis, it is likely that we'll need to pad them in order to stretch their power spectrum. On the other hand, the tiles above the tilt axis have a lower defocus, so we'll need to crop the tiles in order to compress their power spectrum. To prevent loosing some information during cropping, all the tiles can be padded in advance by a given factor (super-sampling) to make sure the cropping will not intrude actual data.

The goal of the next steps is to find the padding/cropping factor (referred as S) to apply to the super-sampled tiles.

(a) If the `PIXEL_SIZE` is lower than 2Å, the Thon rings start to be compressed in a very small frequency window, which makes the analysis more complicated. In this case, re-sample the tilt-series at 2Å, by Fourier cropping. This effectively stretches the power spectrum of the image.

(b) For each tile:

i. If we know the defocus difference between a tile and the tilt axis and if the tilt axis is along the Y axis, then we can compute:

- $\Delta Z = -tan(\alpha) \times (X_{tile} - X_o)$; α is the tilt angle, X_{tile} is the center of the tile in X and X_o is the position of the tilt axis in X.
- $S = \sqrt{(1 + \frac{\Delta Z}{\Delta f})}$, with Δf being the defocus at the tilt axis.

ii. Compute the theoretical 1d CTF at the tilt axis (CTF_o) and at the tile height (CTF_{tile}).

iii. Refine S. For each $S \pm 1$, 0.01 step: apply S to CTF_{tile} (stretch by linear interpolation) and compute the CC between CTF_o and CTF_{tile}. Select the S that gave the best fit and refine again around this new factor ($S \pm 0.1$, 0.001 step) as in the previous step.

iv. Select the S that gave the best fit and compute the padded/cropped tile such as: $size_{adjusted} = size_{tile} \times S$

(c) Compute the averaged 2d power spectrum of each adjusted tile, and band-pass it as in step **7.c.iii**. Repeat the previous step for each view of the tilt-series and gather these stretched power spectra into one stack:

- `fixedStacks/ctf/<prefix>_ali1-PS2.mrc`: averaged band-passed 2d power spectrum. Each view was processed independently.
- `fixedStacks/ctf/<prefix>_ali1-PS.mrc`: averaged 2d power spectrum. Each view was processed independently.

(d) Send this stack (`<prefix>_ali1-PS2.mrc`) to CTFFIND4 (4.10.14) for analysis and update columns 12, 13 and 15 of `fixedStacks/ctf/<prefix>_ali1_ctf.tlt` (table 5) with the results.

15.3 Template matching

1. **Reconstruct the tomograms**:
   Reconstruct by weighted back-projection the desired sub-region (defined in section 6).
   (a) Load the dose weighted aligned stack in `aliStacks/<prefix>_ali1.fixed` and bin it to the desired sampling (`Tmp_sampling`).
   (b) Load the region coordinates in `recon/<prefix>_recon.coords`, the tilt angles in `fixed-Stacks/<prefix>.tlt` and the local alignments in `fixedStacks/<prefix>.local` if it exists, and send them to tilt to reconstruct the tomograms.
2. **Prepare the template**:
   Load the template and pad/trim it to make it squared ($size = 2\times$`Ali_mRadius`), standardize it ($\mu = 0$, $\sigma = 1$) and bin it to the desired sampling (`Tmp_sampling`).
3. **Optimize the size of the tomogram**:
   If the tomogram is larger than the desired size (`Tmp_targetSize`), split it into chunks with the most optimised size to perform Fast Fourier Transforms.
4. **Pre-processing**:
   For each chunk:
   (a) Apply a band-pass filter:
      - High-pass: remove very low resolutions (600Å) to deal with ice/intensity gradients; mean is set to zero.
      - Low-pass: start the Gaussian roll off from `lowResCut`. If it is not defined, set it to an estimate of the first CTF zero based on the defocus saved in `<fixedStack-/ctf/<prefix>_ali1_ctf.tlt>` (table 5).

   (b) If `Tmp_medianFilter` is defined, apply a median filter to the chunk.
   (c) Standardize the chunk ($\mu = 0$, $\sigma = 1$)
   (d) Phase flip the chunk. We assume that `lowResCut` before the first zero, so flip in real space by multiplying the chunk by -1.

   Finally, normalize the global variance (i.e. variance of the entire sub-region).
5. **Angle search**:
   Computing the cross-correlation (CC) between the tomogram and the template gives us a CC map. Each pixel of this map corresponds to the CC score between the template and the tomogram centered on this pixel. If we sample different rotations, how do we keep track of CC scores? After all, the CC scores do not tell us what was the rotation of the template.
   (a) Extract all the rotations to sample (ϕ, θ, ψ) from the `Tmp_angleSearch` parameter. Each trio of Euler angles is assigned to a number (1 to n), referred as "**angle$_{\textbf{ID}}$**".
   (b) Prepare two empty arrays, referred as "**coordinates**" and "**angles**", of the same size as the tomogram (i.e of the same size as the CC map).
      - **coordinates**: Will store the cross-correlation scores.
      - **angles**: Will store the **angle$_{\textbf{ID}}$**s.

   (c) For each chunk, for each **angle$_{\textbf{ID}}$**:

i. Rotate the template, pad it to match the size of the tomogram chunk, band-pass filter it with the band-pass filter computed in step **4.a** and correct for any change in power due to rotation/interpolation.
ii. Compute the cross-correlation (CC), in Fourier space, between the tomogram chunk and the rotated template. Normalize this CC map by its standard deviation.
iii. Save the results:
   - If it is the first rotation to be sampled, set **coordinates** equal to the CC scores and **angles** equal to **angle$_{\mathbf{ID}}$** (which is equal to 1 in this case).

```
# matlab
coordinates = CCmap;
angles = 1;
```

   - If it is *not* the first rotation to be sampled, update the CC scores only if they are higher than the previous iteration. For each pixel updated in **coordinates**, update the corresponding pixel in **angles** with the current **angle$_{\mathbf{ID}}$** to keep track of the rotation that gave this CC score.

```
# matlab
mask = coordinates < CCmap;
coordinates(mask) = CCmap(mask);
angles(mask) = angleID;
```

6. **Save the results of the angle search**:
At the end of the angle search, **coordinates** contains, for each pixel of the tomogram, the best CC score that was calculated and **angles** contains, for each CC score stored in **coordinates**, the rotation (i.e. **angle$_{\mathbf{ID}}$**) the template had when computing the cross-correlation.

Save in `convmap_wedgeType_2_bin<X>` (`<X>` equal to `Tmp_sampling`):

- **coordinates** as `<prefix>_<region>_bin<X>_convmap.mrc`.
- **angles** as `<prefix>_<region>_bin<X>_angles.mrc`.
- `<prefix>_<region>_bin<X>_angles.list`: Every sampled ϕ, θ, ψ Euler angles.

7. **Extract the strongest peaks**:
Select the x, y, z coordinates of the n strongest peaks (i.e. CC scores) from **coordinates**, with n equal to `Tmp_threshold`. For each peak, extract the corresponding **angle$_{\mathbf{ID}}$** from **angles** and convert the **angle$_{\mathbf{ID}}$** back to the corresponding three Euler angles.

(a) Select the x, y, z coordinates of the strongest peak registered in **coordinates**. To take into account the neighbouring pixels, these coordinates are shifted to the local center of mass ($3 \times 3 \times 3$ matrix, centered on the strongest peak).

(b) Using the peak x, y, z coordinates, catch the peak's corresponding **angle$_{\mathbf{ID}}$** in **angles** and convert these Euler angles into a rotation matrix. If `<symmetry>`, generate a uniform distribution over the in-plane symmetry related angles to reduce missing-wedge bias.

(c) Erase the selected peak and its neighbours by masking them out:
   i. Apply the rotation from step (b) to a mask of shape `Peak_mType` and size `Peak_-mRadius` ($2 \times$ `particleRadius` by default).
   ii. Apply the rotated mask at the peak coordinates to remove it, as well as the neighbouring peaks.

(d) Standardize the CC score: $score = (score - \mu)/\sigma$, μ is the average, and σ is the standard deviation, of all the peaks (i.e. of **coordinates**).

(e) Repeat step (a) and (b), `Tmp_threshold` times.

(f) Every peak is described by its x, y, z coordinates, its ϕ, θ, ψ Euler angles and its standardized CC score. Gather all the information into the following file. For each peak (i.e. line):

Table 6: `convmap_wedgeType_2_bin<X>/<prefix>_<region>_bin<X>.csv`

Column	Description	Column	Description
1	standardized CC score	14	Euler angle ϕ
2	`Tmp_sampling`	15	Euler angle θ
3	empty (0)	16	Euler angle ψ
4	ID (unique per sub-region)	17	Rotation matrix r_{11}
5	empty (1)	18	Rotation matrix r_{21}
6	empty (1)	19	Rotation matrix r_{31}
7	empty (1)	20	Rotation matrix r_{12}
8	empty (1)	21	Rotation matrix r_{22}
9	empty (1)	22	Rotation matrix r_{32}
10	empty (0)	23	Rotation matrix r_{13}
11	x coordinates (unbinned, from 1, origin at left corner)	24	Rotation matrix r_{23}
12	y coordinates (unbinned, from 1, origin at left corner)	25	Rotation matrix r_{33}
13	$z$coordinates (unbinned, from 1, origin at left corner)	26	Class (1)

(g) Bin the x, y, z coordinates to `Tmp_sampling`, save them into `<prefix>_<region>_-bin<X>.pos` and convert them into an IMOD mod file with the following command:

```
point2model -number 1 -sphere 3 -scat *.pos  *.mod
```

References

[1] Fabian Eisenstein, Radostin Danev, and Martin Pilhofer. Improved applicability and robustness of fast cryo-electron tomography data acquisition. *Journal of Structural Biology*, 208(2):107–114, November 2019.