PyTomGUI for processing of cryo-Electron Tomograms

Gijs van der Schot & Friedrich Forster

V1.0 (June 2019)

1. Introduction

The aim of this tutorial is to introduce you to the computational workflow in cryo-electron tomography (Cryo-ET). Cryo-ET is an imaging technique used to obtain high-resolution three-dimensional images of biological objects such as macromolecules and cells in their near native environment. Samples are tilted as they are imaged, resulting is a set of 2D images (tilt series), that can be combined to form a three-dimensional (3D) reconstruction. In Cryo-ET samples are immobilized in non-crystalline ice (imaged at temperatures below -150 °C) allowing them to be imaged without dehydration or chemical fixation, processes which could disrupt or distort biological structures.

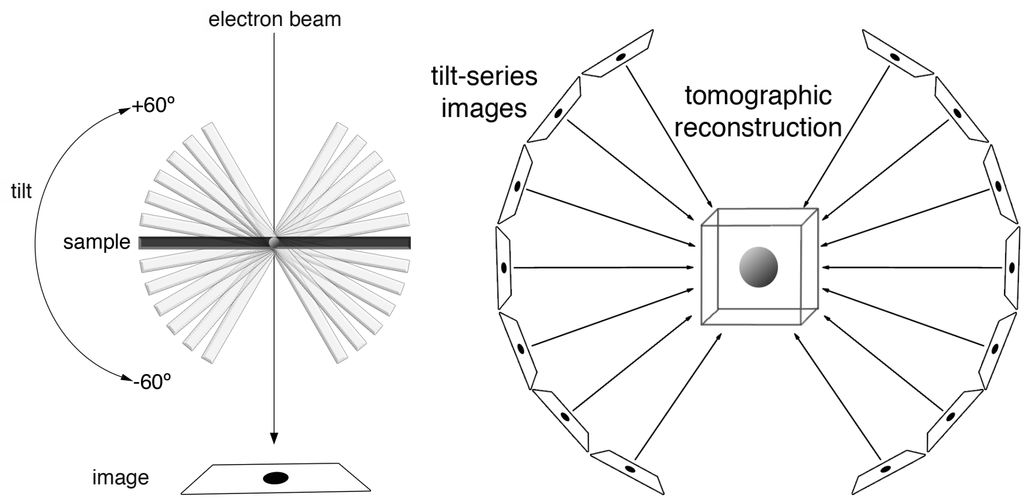


Figure 1. Schematic of the electron tomography setup. The sample is tilted between -60° and +60°. The recorded Tilt-series images are combined into a tomographic reconstruction. From: Eikos, Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45403034.

In the first step of the tutorial you will reconstruct several three-dimensional tomograms from measured tilt series. Using quantitative measures such as Fourier Shell Correlation (FSC) the resolution of a raw tomogram barely exceeds 50 Å – nevertheless, much higher resolution information will be present in your tomograms, but difficult to distinguish from the background – if you squint you tend to see quite some detail beyond this level. Larger macromolecular complexes can be identified visually in tomograms.

The limit on the resolution of the tomographic reconstruction is due to the maximal dose the sample tolerates. In order to extend the resolution, identical parts of the tomogram can be averaged. This is called subtomogram averaging. Resolutions beyond 4 Å have been achieved this way. In the second part of the tutorial you will select regions of interest within a reconstructed tomogram, and you will align the sub-tomograms to obtain a higher resolution structure of the object of interest.

*NOTES FOR OPERATION ON THE ANGSTROM CLUSTER @UU

In order for PytomGUI to work you will have to execute the following lines in the terminal that load the required modules for the program.

```
module load openmpi/2.1.1
module load python3/3.7
module load lib64/append
module load pytom/dev/gui_beta
```

(optional for frame alignment and CTF determination)

```
module load imod/4.10.28
module load motioncor2/1.2.3
```

Start the gui by executing:

```
pytomGUI.py
```

2. Data transfer and Preprocessing

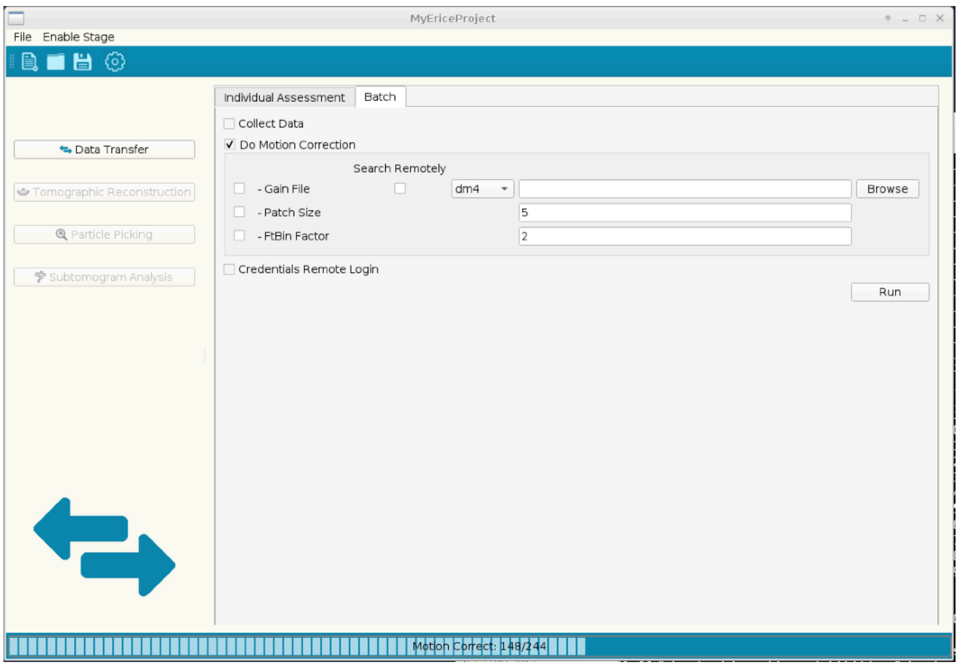
2.1. Data collection

In the GUI you will see a ‘’Stage Selection” panel on the left, with one active button *Data Transfer*. Click this button.

In the connected frame on the right you see two tabs (Figure 2). “Individual Assessment”, and “Batch Mode”. We will use *Batch Mode* to copy data to our project folder.

The data we want to collect is stored in `/data/tutorials/Erice/data`. These data are four tiltseries that contributed to (Braunger, Pfeffer et al, Science 2018); the sample was crude cell membrane extract from HEK293 cells, which primarily contain free ribosomes and Endoplasmic Reticulum (ER) derived (ribosome-studded) vesicles. The data were recorded using SerialEM on a K2 detector in counting mode as frames. The folder contains the recorded frames for each tilt stored as *tif* files and a parameter file (from SerialEM data acquisition) stored as an *mdoc*-file. Make sure you tick the box for the *mdoc* collection as well. This file is not strictly required as long as the tilt angle, and unique prefix to the tilt series, are part of the filename[1].

[1] Please do not use hyphens in your filename, and use ‘’_’’to delimit experimental parameters such a date, time, and angle.



- [ ] Collect Data
- [✓] Do Motion Correction
  - [ ] - Gain File [ ] dm4
  - [ ] - Patch Size 5
  - [ ] - FtBin Factor 2
- [ ] Credentials Remote Login

Motion Correct: 148/244

Figure 2. Data collection and Motion Correction. The left panel is the *Stage*, where *Data Transfer* is active. In the main panel *Batch* mode for transferring and preprocessing a series of tomograms is chosen. After importing the data using *Collect Data*, the frames can be motion corrected by checking the *Do Motion Correction* box.

2.2. Motion Correction

The input TIF files are composed of a series of frames for each tilt angle. In order to compensate for beam induced motion of the sample, a series of frames are recorded in rapid succession. The individual frames can be aligned during an initial preprocessing step called motion correction. The PyTom-GUI uses *Motioncor2* for motion correction [2]. For motion correction one can supply additional files, like a gain reference file, which *motioncor2* will use for correcting the different probabilities of pixels to record a signal. You can select if you want to align patches within the image, and whether you want to crop the images in Fourier space, i.e., to downsample the images (for example from super-resolution to normal-resolution). In the example, those options are not used.

*NOTE:

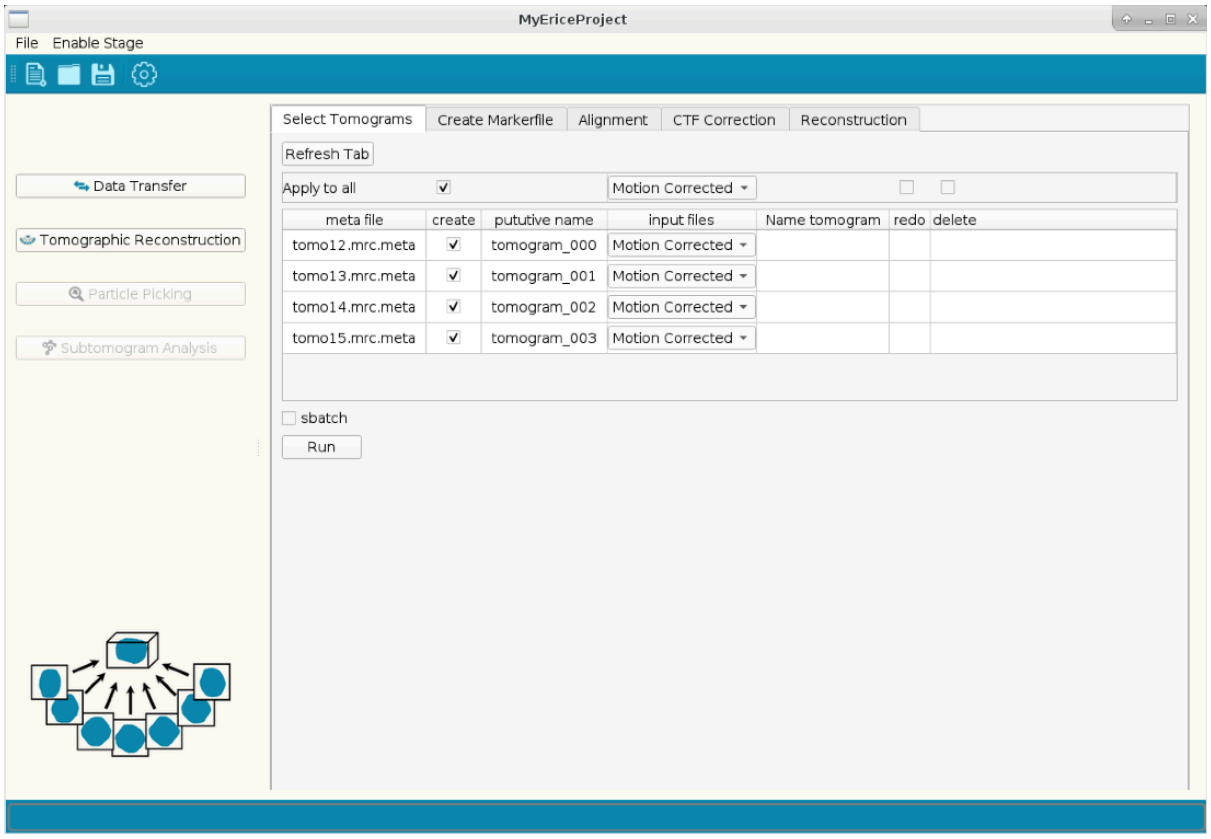
1) We will not use patches, nor binning or gain correction. If a file called *gain_ref.dm4* is present in your data folder, this can be used for gain correction.

3. Tomographic Reconstruction

3.1. Tomogram alignment

After successful data collection and motion correction, we are ready for the next step: Tomographic reconstruction. First you need to enable the *Tomographic Reconstruction* stage in the top menu, which then displays the *Tomographic Reconstruction* icon on the left panel in color (Figure 3).

Click on the second button in the stage selection. Choose *Select Tomograms* and click on *Refresh Tab* to update the table of prospect tomogram folders, in which the tilt images names will be numbered with respect to their tilt angle. As we have downloaded three tilt series, we only see three options. Select the four options and press *Run*. Now, tilt images will be squared (for K2 images x and y dimensions differ slightly), extreme pixels will be corrected to the mean value and the data are copied to separate folders named `tomogram_000`–`tomogram_003`.

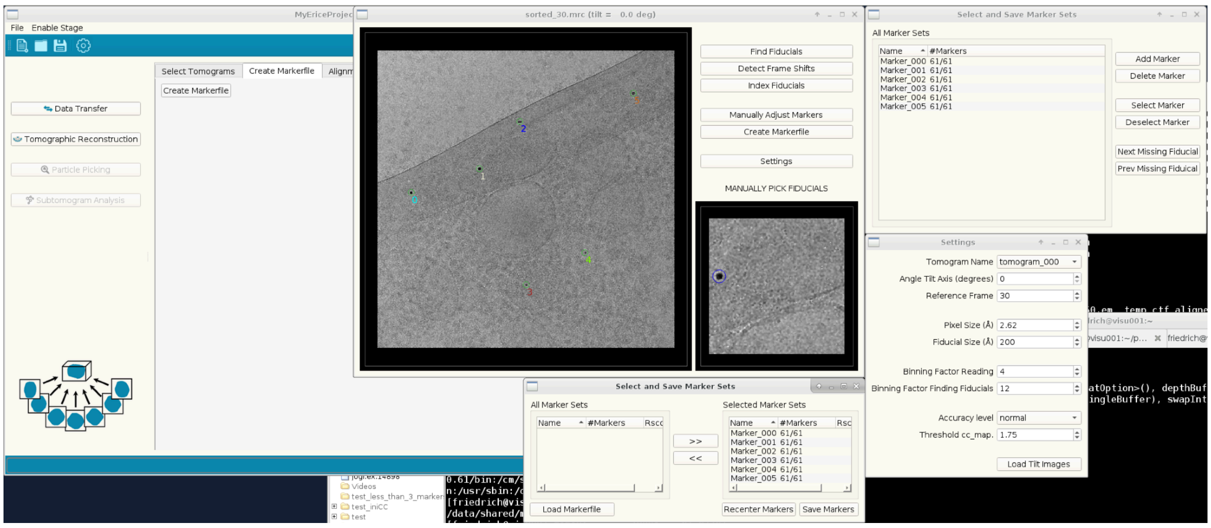


Apply to all ✓ Motion Corrected

meta file	create	pututive name	input files	Name tomogram	redo	delete
tomo12.mrc.meta	✓	tomogram_000	Motion Corrected			
tomo13.mrc.meta	✓	tomogram_001	Motion Corrected			
tomo14.mrc.meta	✓	tomogram_002	Motion Corrected			
tomo15.mrc.meta	✓	tomogram_003	Motion Corrected			

Figure 3. Start organizing your project. In the *Select Tomograms* menu you invoke a file structure, which facilitates organizing your data.

After this step we can go to the *Create Markerfile* tab (Figure 4). The rough alignment of individual tilt images is known from the experiment, but that is not sufficiently accurate for a meaningful reconstruction. Thus, we need to correct for differences in rotation, translation, and magnification between individual tilt images. For this we have added gold particles (fiducials) to our sample. They form points of reference in each tilt image. The aim of this step is to trace their relative position in each of the tilt images. You have to track at least five gold particles that are present in each tilt image. The GUI will aid you in this process.



All Marker Sets

Name	#Markers
Marker_000	61/61
Marker_001	61/61
Marker_002	61/61
Marker_003	61/61
Marker_004	61/61
Marker_005	61/61

Selected Marker Sets

Name	#Markers	Rsc
Marker_000	61/61	
Marker_001	61/61	
Marker_002	61/61	
Marker_003	61/61	
Marker_004	61/61	
Marker_005	61/61	

Settings

Tomogram Name	tomogram_000
Angle Tilt Axis (degrees)	0
Reference Frame	30
Pixel Size (Å)	2.62
Fiducial Size (Å)	200
Binning Factor Reading	4
Binning Factor Finding Fiducials	12
Accuracy level	normal
Threshold cc_map.	1.75

***Figure 4**. Fiducial selection windows*. From the main menu, you choose the *Tomographic Reconstruction* stage and the *Create Markerfile* tab therein. In there, *Create Markerfile* invokes the Settings menu. If you *Load Tilt Images* the display window (here image sorted_30.mrc is displayed) opens. Clicking *Manually Adjust Markers* opens the *Select and Save Marker Sets* (this name will be changed in the future to *Edit Markers*) window. The *Create Markerfile* button of the display menu opens Select and Save Marker Sets windows (which indeed does what its name says).

Select a tomogram name, set the binning factors and press *Load Tilt Images*. After this you can select the accuracy of finding fiducials. Now press *Find Fiducials*. Consecutively press *Detect Frameshift* to account for the frame shift between images. Press *Index Fiducials* to create sets of fiducials automatically.

Upon clicking an empty display window and a *Settings* window will open. In the Settings window you specify your tomogram (in *Tomogram Name*) as well as a number of parameters, which are wherever possible filled out automatically (e.g., pixelsize from specification in *mdoc* file). Recommended parameters for Binning Factor Reading and Finding Fiducials are 4 and 8 or 12, respectively. *Accuracy level* `normal` and *Threshold cc_map* `1.75` are good starting values. When you click *Load Tilt Images* the images are read into memory and eventually displayed.

It is handy to navigate through the display of the tilt images with hotkeys. Some hotkeys that make it easier to navigate through tilt series: *1,2,3* display min, reference (typically 0 degrees) and max tilt angles, respectively. You can shift between frames using the left and right arrow.

The marker detection and assignment bases on a semi-automated procedure. If you click *Find Fiducials* candidates for gold beads are detected in the images. The found candidates should correspond with beads you identify by eye, but it will inevitably also contain false positives and false negatives. Testing different values for the *Threshold_cc_map* may enable detecting more candidates (=lower threshold) or eliminating false positives (=higher threshold).

Based on the found markers the images can be brought into approximate register by clicking *Detect Frame Shifts*.

Good approximations of the shifts from one projection to another are the basis for the next step: *Index Fiducials*. Here, an algorithm seeks to assign indexes to markers throughout the tiltseries. If there are many candidates, this step can take really long – so it is advisable to limit the candidates in the reference projection (use hotkey '2' in display window). You select *Manually adjust Markers,* then you can focus on areas in the overview image with the left mouse button and remove markers in the focus image with the right button (or add with left, if required). If you are done adjusting your markers in the reference projection press *Index Fiducials*.

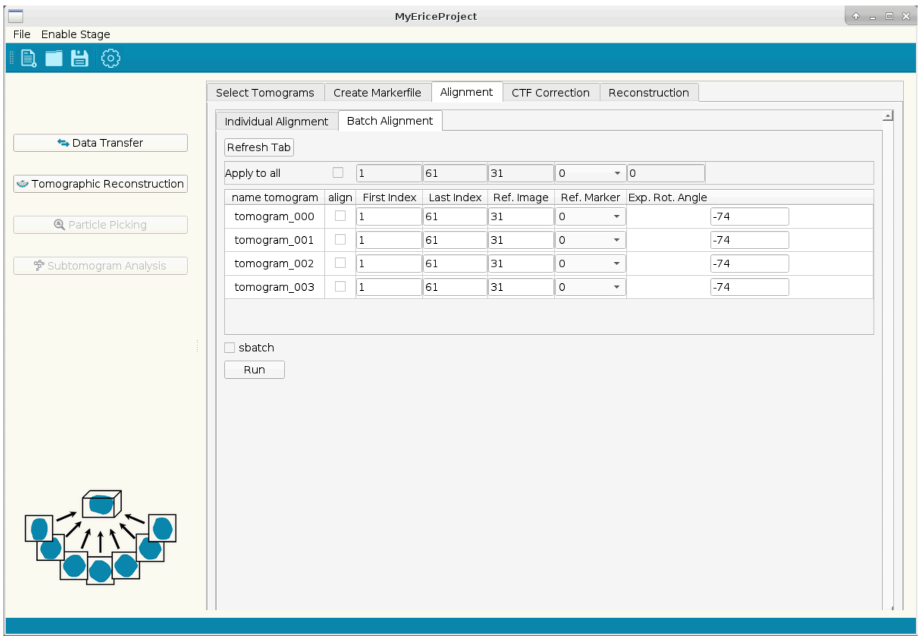
When the indexing is finalized, colored numbers are displayed next to those markers that could be indexed with confidence. Typically, for a number of markers this indexing will not succeed for all tilt angles. In some rare cases, the indexing might even jump from one marker to another. Both can be fixed by manual intervention. In the display window, the <(,) and >(.) buttons allow you to navigate to previous / next image. After some preliminary adjustment (e.g., adding/removing some markers in the reference projection using the *Manually adjust Markers* menu) you can press *Index Fiducials* again to help with indexing – however, do not be surprised if index numbers get changed. Occasionally, displaying the images as a loop using the 'l' button can be useful.

Finally, you save the markers using the *Create Markerfile* menu. Choose those markers you want to use for the tilt series alignment and *Save Markers*. The markers will be saved in a file `markerfile.em` in the `sorted` folder of the respective tomogram. The coordinates have only been determined coarsely in the binned images. You can refine these positions in the unbinned images using a simple iterative algorithm by clicking *Recenter Markers* and finally saving them again.

Note: If you do not want to use the fiducial localization from PyTomGUI you do not have to. In principle, everything required for the PyTom workflow is a file that contains the (indexed) marker coordinates. The underlying PyTom alignment functions also accept IMOD-generated wimp files instead of the pytom-generic `markerfile.em` format. You can also load these marker coordinates into the PyTomGUI when you load a WIMP markerfile in the Create Markerfile menu – PyTom also supports this format in addition to its native markerfile.em format. Alternatively, you can also use matlab function av3_wimp2em.m for conversion.

3.2. Alignment

After the creation of a markerfile, one can apply the image transformations to the tilt images. For the eventual high-resolution reconstructions, it is handy to generate the unbinned, aligned tilt images, which are created in this step. To this end we activate the *Alignment* panel and choose *Batch Alignment* to process all tomograms from the project. Clicking *Refresh Tab* should show all available tomograms. Select the folder where your sorted tilt images are (`[PROJECTDIR]/03_Tomographic_Reconstruction/tomogram_???/sorted/`), together with the respective metafile (located in the same folder). Make sure the binning factor is set to 1, as we will later use the unbinned aligned tiltseries. Check the *sbatch* box to submit the job to the queue or *Run* locally.



Apply to all	☐	1	61	31	0	0

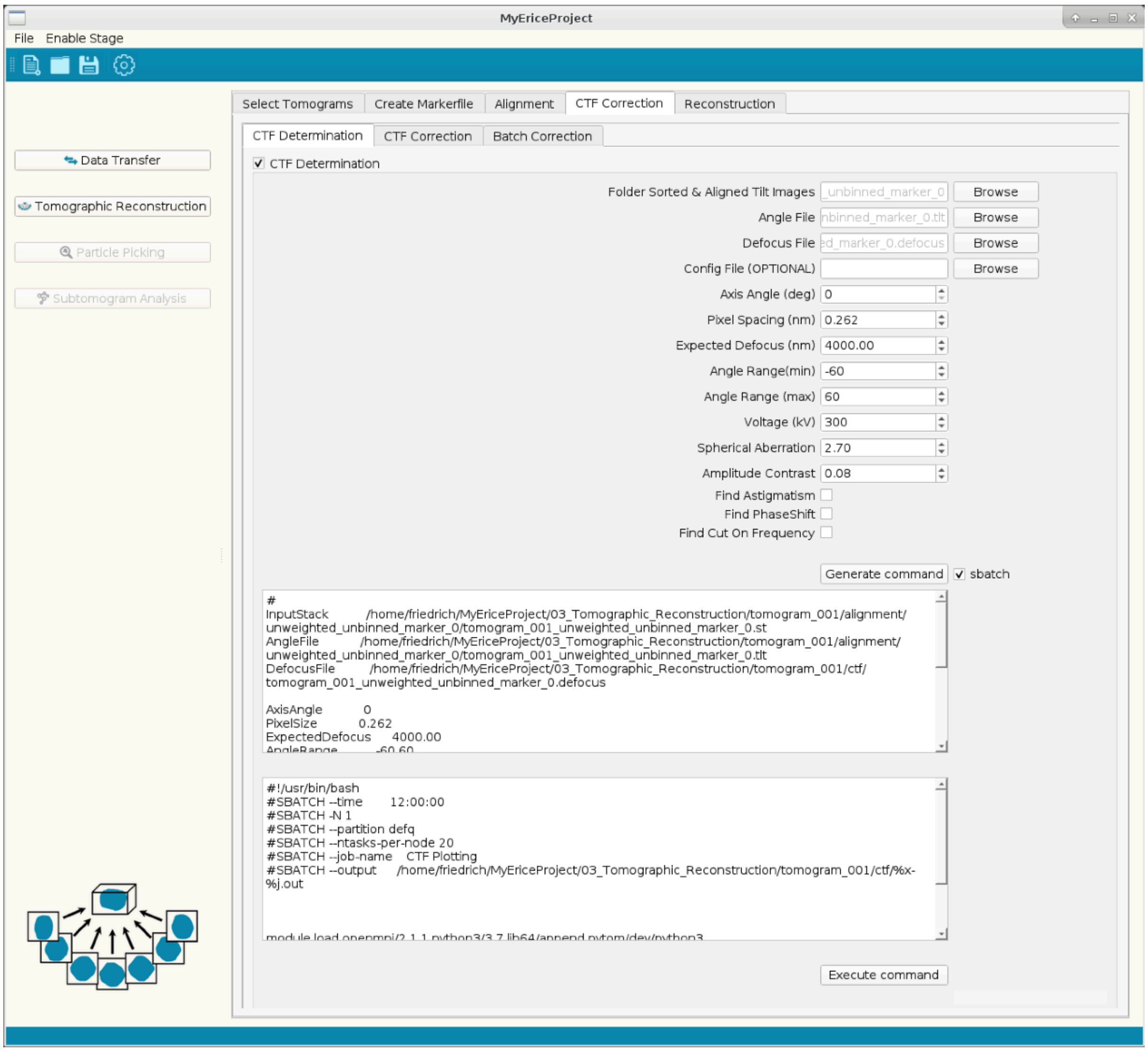
name tomogram	align	First Index	Last Index	Ref. Image	Ref. Marker	Exp. Rot. Angle
tomogram_000	☐	1	61	31	0	-74
tomogram_001	☐	1	61	31	0	-74
tomogram_002	☐	1	61	31	0	-74
tomogram_003	☐	1	61	31	0	-74

Figure 5. Transform alignment parameters to tilt series and write to disk.

3.3. CTF Correction

All TEM images are subject to convolution with a Contrast-Transfer Function (CTF). Approximate deconvolution is ultimately required to get a faithful image of the specimen, analogous to the typical workflow in single particle analysis. Peculiar about cryo-ET is that the images have a strong defocus gradient due to the tilting of the specimen. This defocus gradient has to be considered when determining the CTF parameters (defocus, astigmatism, phase shift) and also in the CTF correction. The PyTomGUI employs CTFplotter from IMOD to determine the CTF parameters (*CTF Determination*) and PyTom's own procedure for *CTF Correction* (phase flipping). For both, determination of CTF parameters and the actual CTF correction the tilt angle is required. Hence, CTF correction is performed after the tilt series alignment in the PyTomGUI workflow.

To start the CTF correction click the tab *CTF Correction* in the *Tomographic Reconstruction* stage. We start with the determination of the CTF parameters by CTF plotter in the *CTF determination* panel (Figure 6). The Folder *Sorted & Aligned Tilt Images* can be found in `[myProject]/03_Tomographic_Reconstruction/tomogram_xxx/alignment/unweighted_unbinned_marker_x/`. Enter the approximate *Expected Defocus* value, which is important for good starting parameters for the search.



☑ CTF Determination

Folder Sorted & Aligned Tilt Images	unbinned_marker_0
Angle File	nbinned_marker_0.tlt
Defocus File	ed_marker_0.defocus
Config File (OPTIONAL)	
Axis Angle (deg)	0
Pixel Spacing (nm)	0.262
Expected Defocus (nm)	4000.00
Angle Range(min)	-60
Angle Range (max)	60
Voltage (kV)	300
Spherical Aberration	2.70
Amplitude Contrast	0.08
Find Astigmatism	☐
Find PhaseShift	☐
Find Cut On Frequency	☐

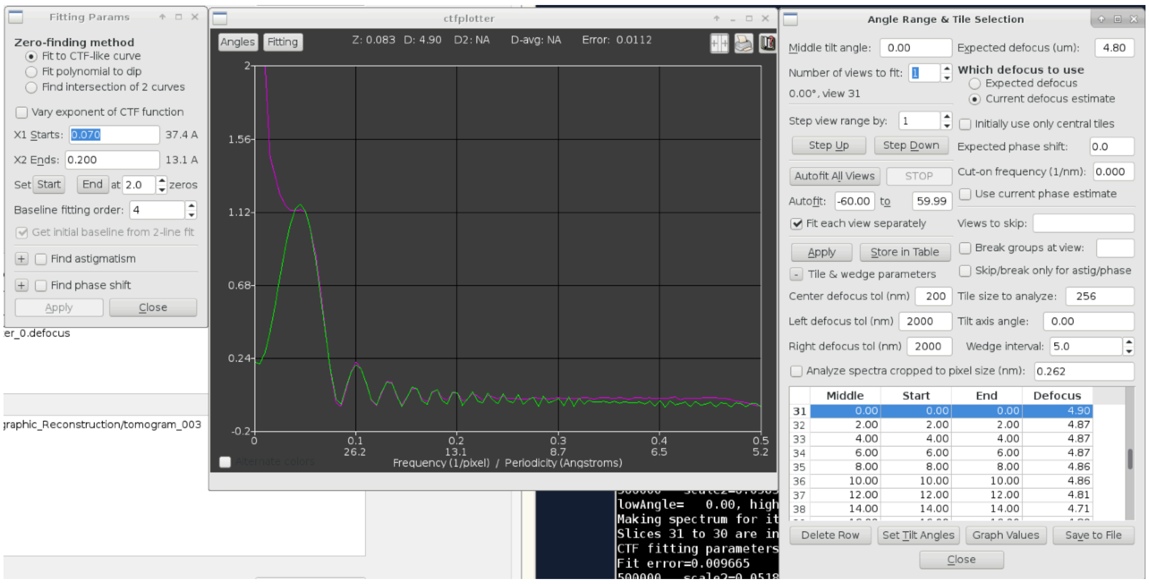
☑ sbatch

```
#
InputStack        /home/friedrich/MyEriceProject/03_Tomographic_Reconstruction/tomogram_001/alignment/
unweighted_unbinned_marker_0/tomogram_001_unweighted_unbinned_marker_0.st
AngleFile         /home/friedrich/MyEriceProject/03_Tomographic_Reconstruction/tomogram_001/alignment/
unweighted_unbinned_marker_0/tomogram_001_unweighted_unbinned_marker_0.tlt
DefocusFile       /home/friedrich/MyEriceProject/03_Tomographic_Reconstruction/tomogram_001/ctf/
tomogram_001_unweighted_unbinned_marker_0.defocus

AxisAngle         0
PixelSize         0.262
ExpectedDefocus   4000.00
AngleRange        -60 60
```

```
#!/usr/bin/bash
#SBATCH --time       12:00:00
#SBATCH -N 1
#SBATCH --partition defq
#SBATCH --ntasks-per-node 20
#SBATCH --job-name   CTF Plotting
#SBATCH --output     /home/friedrich/MyEriceProject/03_Tomographic_Reconstruction/tomogram_001/ctf/%x-
%j.out



module load openmpi/2.1.1 python3/3.7 lib64/append pytom/dev/python3
```

Figure 6. CTF determination. This panel sets the parameters for invoking CTFplotter (Figure 7).

Pressing *Execute command* opens CTF plotter (Figure 7). If agreement between data and simulation is reasonable save the parameter table in the interface (best *Fit each view separately*). This procedure has to be repeated for each tilt series.



Zero-finding method: [✓ Fit to CTF-like curve] Fit polynomial to dip / Find intersection of 2 curves

☐ Vary exponent of CTF function

X1 Starts: 0.070 37.4 A

X2 Ends: 0.200 13.1 A

Set Start End at 2.0 zeros

Baseline fitting order: 4

☑ Get initial baseline from 2-line fit

☐ Find astigmatism

☐ Find phase shift

er_0.defocus

raphic_Reconstruction/tomogram_003

Z: 0.083 D: 4.90 D2: NA D-avg: NA Error: 0.0112

```
lowAngle=   0.00, high
Making spectrum for it
Slices 31 to 30 are in
CTF fitting parameters
Fit error=0.009665
```

Middle tilt angle: 0.00

Expected defocus (um): 4.80

Number of views to fit: 1

0.00°, view 31

Which defocus to use: Expected defocus [✓ Current defocus estimate]

Step view range by: 1

☐ Initially use only central tiles

Expected phase shift: 0.0

Cut-on frequency (1/nm): 0.000

Autofit: -60.00 to 59.99

☐ Use current phase estimate

☑ Fit each view separately

Views to skip:

☐ Break groups at view:

☐ Skip/break only for astig/phase

Tile & wedge parameters

Center defocus tol (nm) 200

Tile size to analyze: 256

Left defocus tol (nm) 2000

Tilt axis angle: 0.00

Right defocus tol (nm) 2000

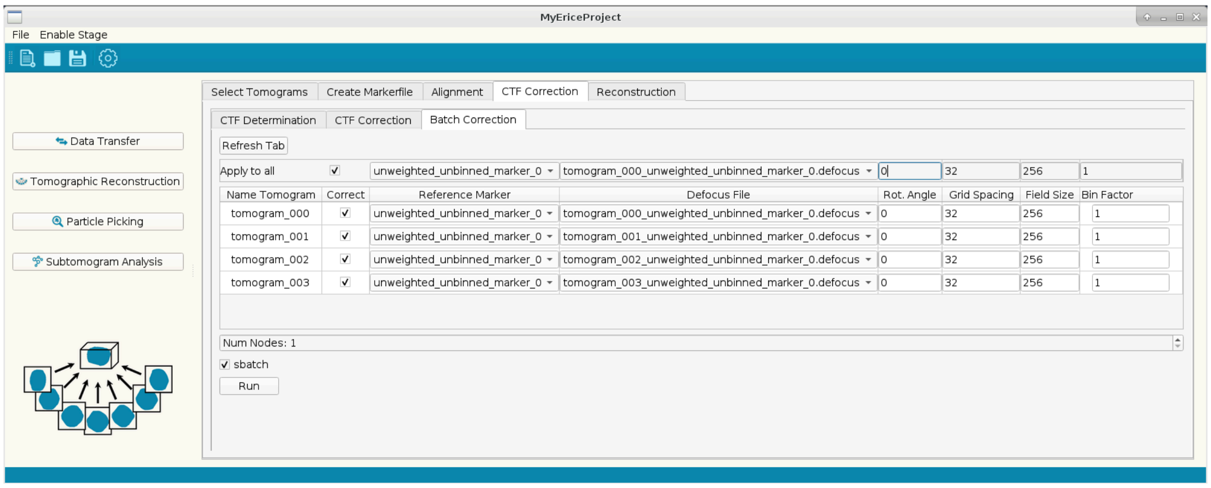
Wedge interval: 5.0

☐ Analyze spectra cropped to pixel size (nm): 0.262

	Middle	Start	End	Defocus
31	0.00	0.00	0.00	4.90
32	2.00	2.00	2.00	4.87
33	4.00	4.00	4.00	4.87
34	6.00	6.00	6.00	4.87
35	8.00	8.00	8.00	4.86
36	10.00	10.00	10.00	4.86
37	12.00	12.00	12.00	4.81
38	14.00	14.00	14.00	4.71

Figure 7. CTF plotter invoked by *Execute Command* button in *CTF determination*. The purple curve is the experimental radially averaged power spectrum and the green curve is the simulated one for the determined defocus value. Clicking *Store in Table* in the right menu and *Save to File* stores the determined parameters. Check *Fit each view separately* to get defocus of each micrograph rather than an averaged value and adjust *Number of views to fit* and *Step view range* for example to 1. Make sure the *Find Astigmatism* box is checked – otherwise the created fileformat will differ and create problems in the CTF correction.

After determination of the CTF parameters the actual CTF correction can be performed, optionally in batch for a set of tomograms (Figure 8). In Fourier space phases are flipped in specific areas, whereby the image is divided into small stripes to account for the defocus gradient.



Name Tomogram	Correct	Reference Marker	Defocus File	Rot. Angle	Grid Spacing	Field Size	Bin Factor
Apply to all	✓	unweighted_unbinned_marker_0	tomogram_000_unweighted_unbinned_marker_0.defocus	0	32	256	1
tomogram_000	✓	unweighted_unbinned_marker_0	tomogram_000_unweighted_unbinned_marker_0.defocus	0	32	256	1
tomogram_001	✓	unweighted_unbinned_marker_0	tomogram_001_unweighted_unbinned_marker_0.defocus	0	32	256	1
tomogram_002	✓	unweighted_unbinned_marker_0	tomogram_002_unweighted_unbinned_marker_0.defocus	0	32	256	1
tomogram_003	✓	unweighted_unbinned_marker_0	tomogram_003_unweighted_unbinned_marker_0.defocus	0	32	256	1

Num Nodes: 1

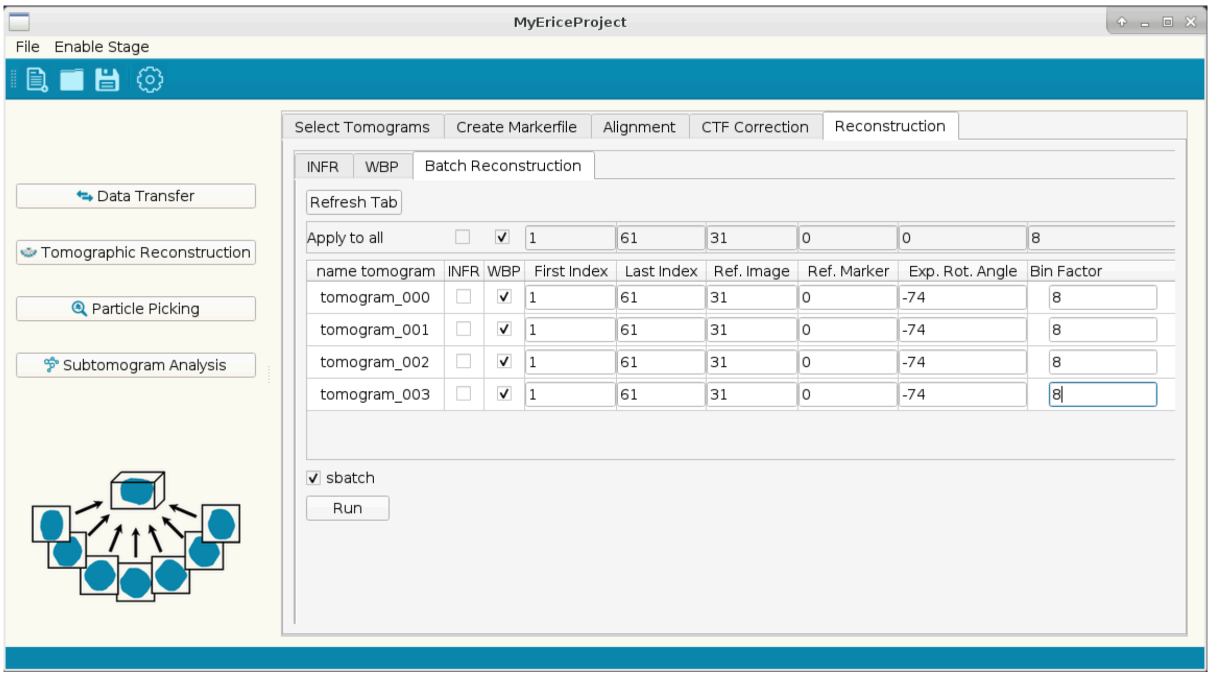
☑ sbatch

Figure 8. CTF correction. In *Batch Correction* mode the phases of the Fourier transforms of the tilt series images are flipped according to the determined defocus values and taking into account of the defocus gradient in the images of the tilted specimen. *Field size* denotes the tile size for the Fourier transform and *Grid spacing* determines the width of stripes for the correction.

3.4. Reconstruction

After creation of the markerfile we compute overview reconstructions. PyTom supports two types of 3D reconstruction: weighted backprojection (WBP) and Iterative Nonuniform Fourier Reconstruction (INFR). WBP comes with two different weighting schemes: r*-weighting, which linearly enhances weights with increasing frequency, and a scheme which accounts for the precise angular sampling (going back to Harausz and van Heel). The former typically results in over-amplification of high-frequency noise, which is reduced with the alternative weighting scheme. The iterative INFR can achieve more accurate weighting and reconstruction at the expense of computation time.

Under the *Reconstruction* tab you can choose *Batch Reconstruction* to reconstruct an entire set of tomograms using specified parameters (Figure 9). This mode support WBP (precise angular sampling) and INFR. The choice of the reference marker determines the coordinate system for the final reconstruction, as also the case for tilt series alignment. We typically use a *Bin Factor* of 8 for overview reconstructions. If *sbatch* is checked, the reconstruction will be submitted to the slurm queue.



name tomogram	INFR	WBP	First Index	Last Index	Ref. Image	Ref. Marker	Exp. Rot. Angle	Bin Factor
Apply to all	☐	✓	1	61	31	0	0	8
tomogram_000	☐	✓	1	61	31	0	-74	8
tomogram_001	☐	✓	1	61	31	0	-74	8
tomogram_002	☐	✓	1	61	31	0	-74	8
tomogram_003	☐	✓	1	61	31	0	-74	8

✓ sbatch

Figure 9. Reconstruction of tomograms. Make sure that Ref. Marker is the same as for the unbinned aligned images created previously (in this case marker 0).

4. Particle Picking

The next step of the processing workflow is to determine the coordinates of potentially interesting features, typically particles of a specific type. There are two options supported to obtain these coordinates: particle coordinates can be determined manually or automatically using template matching.

4.1. Manual Picking

Manual picking of particles is a good option to avoid possible bias due to external references. A typical task can be to localize some particles manually, to create a first subtomogram average from these particles, which may then serve as a reference for further exhaustive searching of tomograms and again subtomogram averaging and even classification.

To start manual picking you enable the stage *Particle Picking*. Then you choose the panel *Manual Picking* (Figure 10) where you can load tomograms and pick particles interactively.

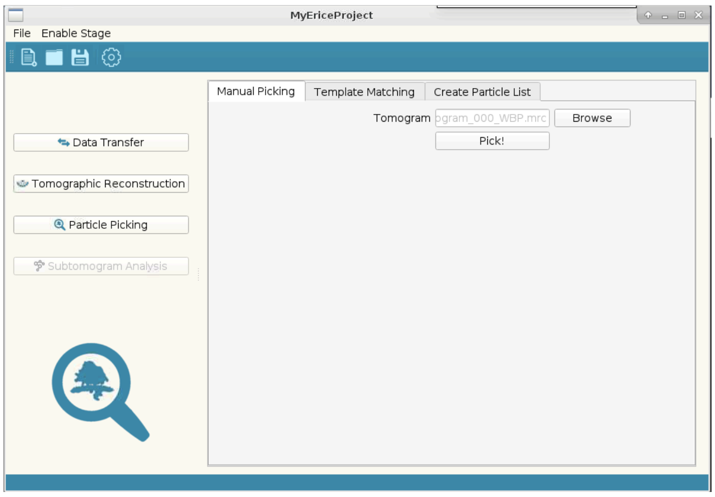
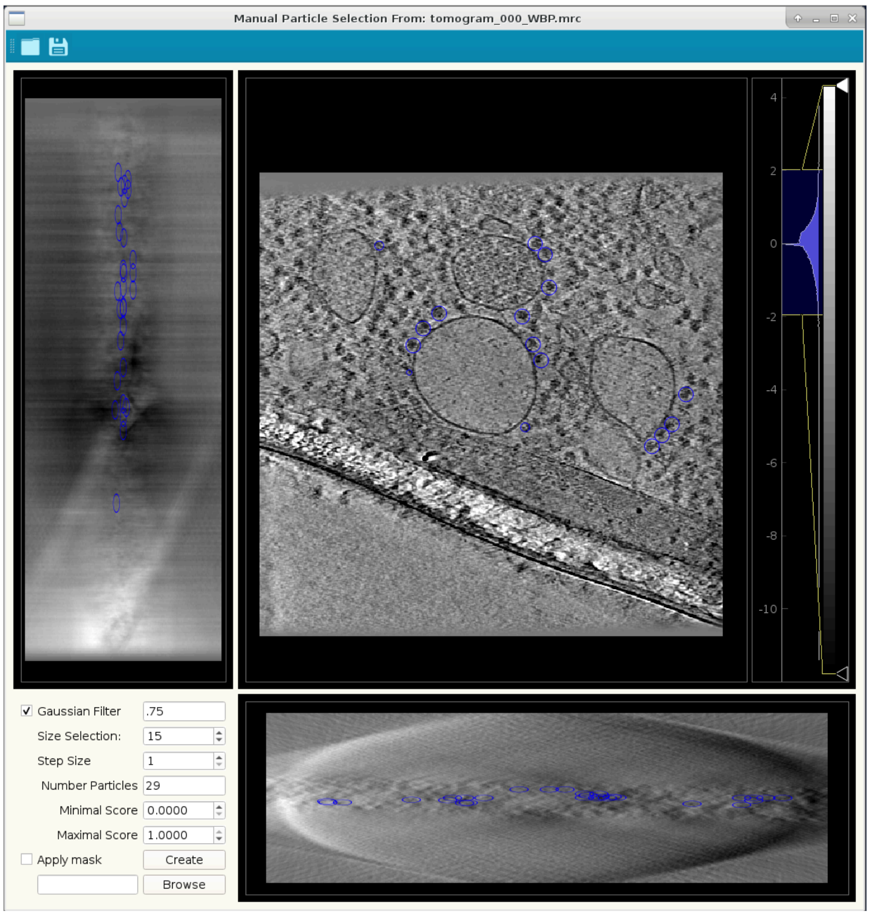


Figure 10. Manual picking panel. *Browse* allows choosing an overview tomogram and *Pick* will open an interactive display window for manually marking particles.

An interactive viewer allows marking features with the mouse in 2D xy-slices (Figure 11). The 3D coordinates can be stored in a file for further processing. The determined coordinates are be saved in ascii (*.txt*) files.

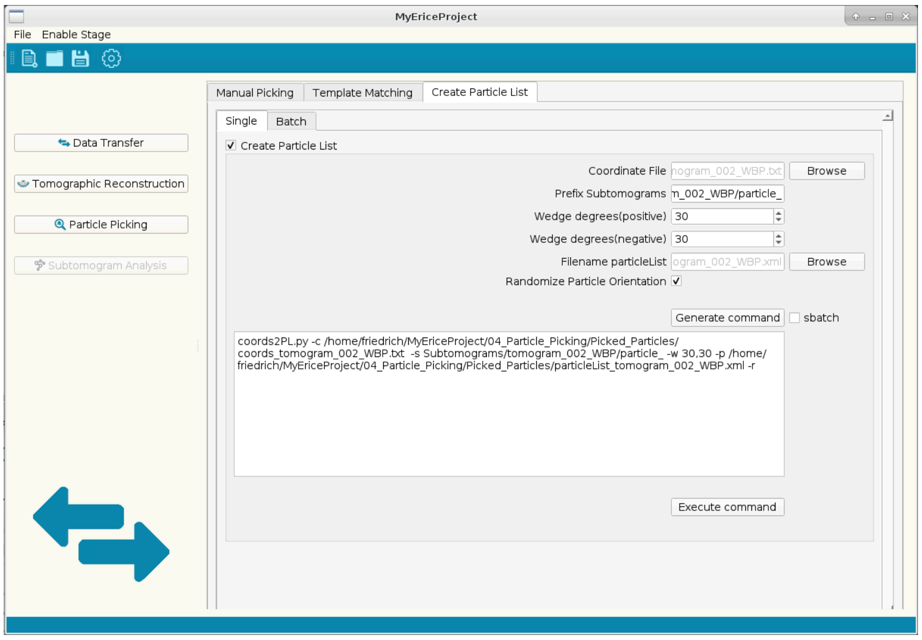


☑ Gaussian Filter	.75
Size Selection:	15
Step Size	1
Number Particles	29
Minimal Score	0.0000
Maximal Score	1.0000

Figure 11. Viewer for selecting particles. To enhance contrast a *Gaussian Filter* can be applied. The *Size Selection* specifies the diameter of your particles (in pixels) and hence the displayed circles. *Step Size* specifies the number of z-slices to be moved upon pressing the arrow buttons.

PyTom internally works with ParticleLists. These are xml files that not only contain particle locations, but also their orientations, fine shifts and other useful information like filenames and missing wedge. To convert the coordinate list into a particleList you choose the *Create*

Particle List panel and the *Single* mode, which allows you to create a particleList with a coordinate file as input (Figure 12). Best stick with the suggested filenames for the (unbinned) Subtomograms, which you will the reconstruct in the *Subtomogram Analysis* stage.



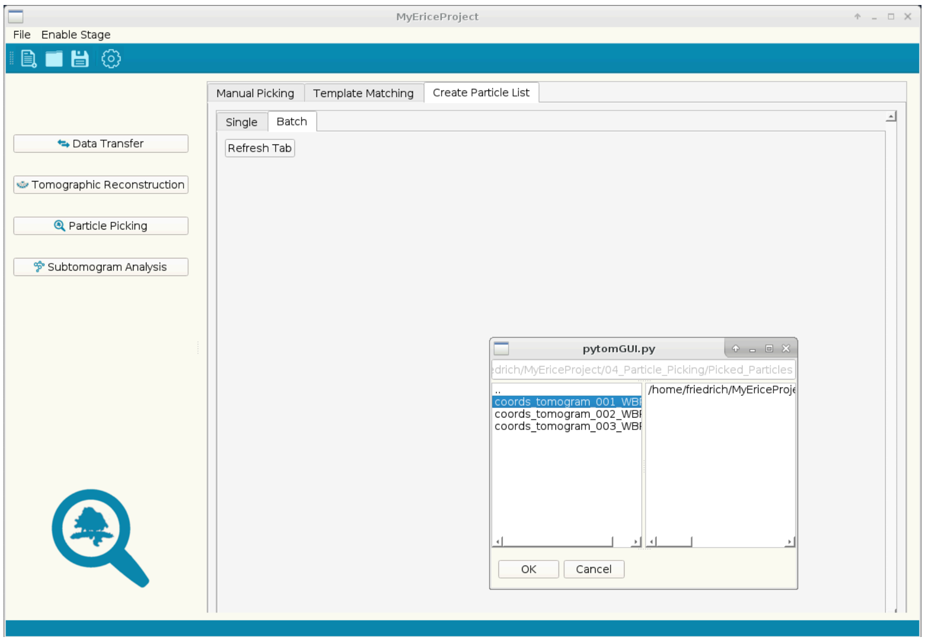
✓ Create Particle List

Coordinate File nogram_002_WBP.txt
Prefix Subtomograms n_002_WBP/particle_
Wedge degrees(positive) 30
Wedge degrees(negative) 30
Filename particleList ogram_002_WBP.xml
Randomize Particle Orientation ✓
sbatch ☐

```
coords2PL.py -c /home/friedrich/MyEriceProject/04_Particle_Picking/Picked_Particles/
coords_tomogram_002_WBP.txt -s Subtomograms/tomogram_002_WBP/particle_ -w 30,30 -p /home/
friedrich/MyEriceProject/04_Particle_Picking/Picked_Particles/particleList_tomogram_002_WBP.xml -r
```

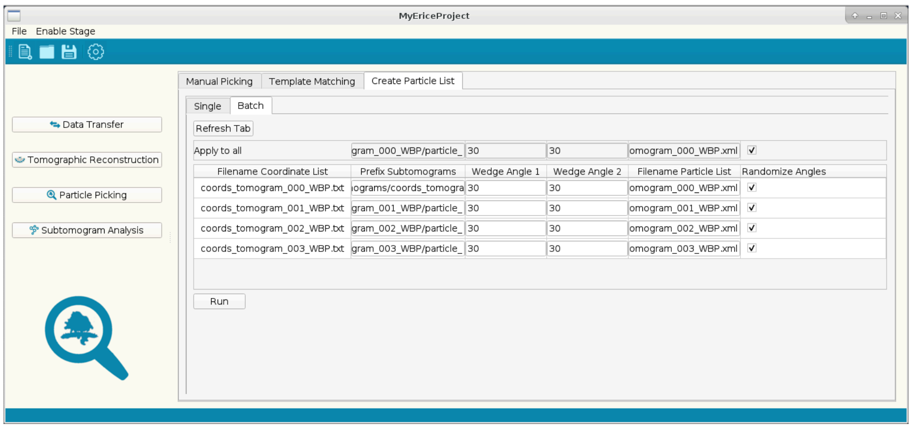
Figure 12. Converting a single coordinate list to a particleList.

Several coordinate lists and respective tomograms are combined into a single unified particleList using the *Batch* option (Figure 13). The resulting particleList then contains particles from different tomograms.



edrich/MyEriceProject/04_Particle_Picking/Picked_Particles
..
coords_tomogram_001_WBF
coords_tomogram_002_WBF
coords_tomogram_003_WBF
/home/friedrich/MyEriceProje

Figure 13. Converting coordinates to particleLists in Batch. Upon clicking the *Refresh Tab*, a dialog window will open. In there you need to double-click on coordinate files to add them to the list of processed coordinate files on the right. Pressing *OK* will invoke Figure 14.



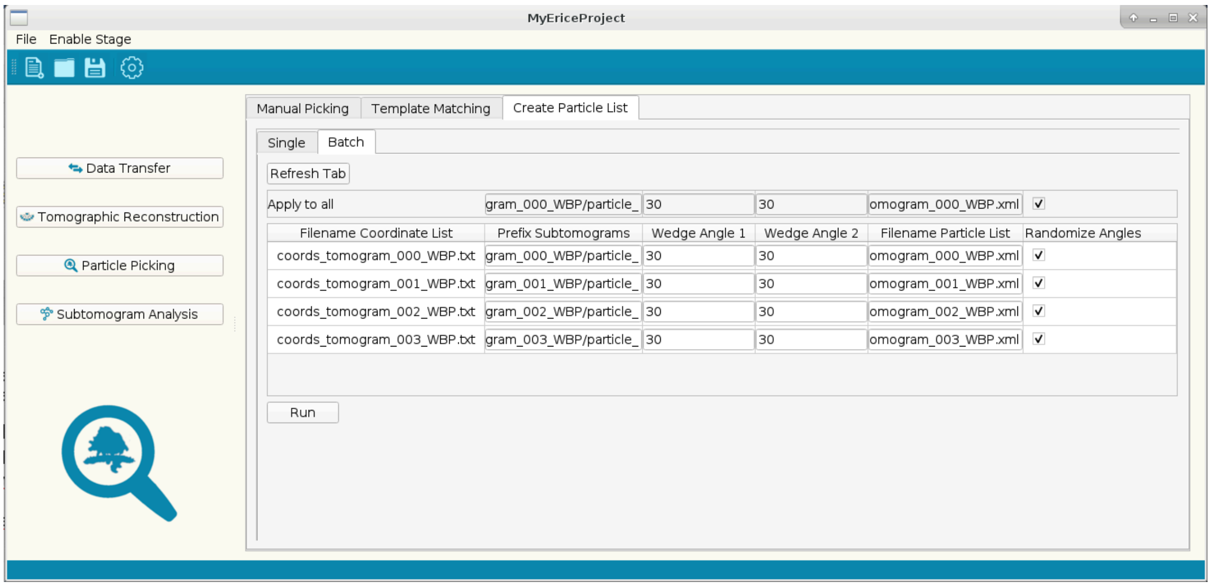
Apply to all	gram_000_WBP/particle_	30	30	omogram_000_WBP.xml	✓
Filename Coordinate List	Prefix Subtomograms	Wedge Angle 1	Wedge Angle 2	Filename Particle List	Randomize Angles
coords_tomogram_000_WBP.txt	iograms/coords_tomogra	30	30	omogram_000_WBP.xml	✓
coords_tomogram_001_WBP.txt	gram_001_WBP/particle_	30	30	omogram_001_WBP.xml	✓
coords_tomogram_002_WBP.txt	gram_002_WBP/particle_	30	30	omogram_002_WBP.xml	✓
coords_tomogram_003_WBP.txt	gram_003_WBP/particle_	30	30	omogram_003_WBP.xml	✓

Figure 14. Continuation of converting coordinates to particleLists in Batch.

4.2. Template matching

Open the *Batch* panel in *Template Matching*. Clicking *Refresh Tab* will open a dialog window, which first asks for Tomograms (double clicking adds a tomogram to the selection!), then template(s), and finally mask(s) to be used in the batch processing. In the tutorial `data` folder, you find a folder `templates` with a ribosome template (21 Å voxel size, corresponding to the 8x-downsampled overview reconstruction) and a corresponding spherical mask.

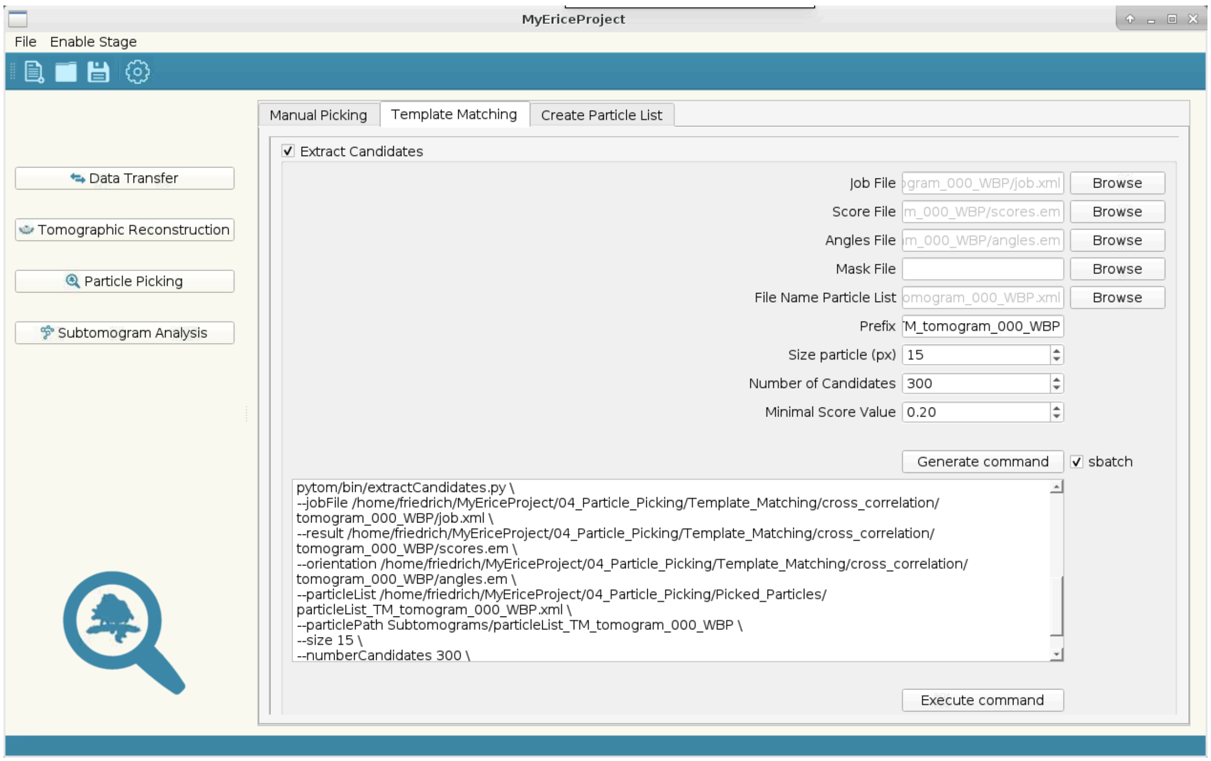
You can specify the parameters in the window that appears after the dialog window (Figure 15). For example, *Angle List* is a list of predefined Euler angles to be probed. Its angular increment and the total number of orientations characterize the respective lists (e.g., `angles_12.85_7112.em` samples 7112 orientations with 12.85 deg increment).



Apply to all	gram_000_WBP/particle_	30	30	omogram_000_WBP.xml	✓
Filename Coordinate List	Prefix Subtomograms	Wedge Angle 1	Wedge Angle 2	Filename Particle List	Randomize Angles
coords_tomogram_000_WBP.txt	gram_000_WBP/particle_	30	30	omogram_000_WBP.xml	✓
coords_tomogram_001_WBP.txt	gram_001_WBP/particle_	30	30	omogram_001_WBP.xml	✓
coords_tomogram_002_WBP.txt	gram_002_WBP/particle_	30	30	omogram_002_WBP.xml	✓
coords_tomogram_003_WBP.txt	gram_003_WBP/particle_	30	30	omogram_003_WBP.xml	✓

Figure 15. Template Matching in *Batch* mode. This window displays after tomograms, template(s) and mask(s) have been specified in a dialog window. The *Randomize Angles* option randomized the particle orientations.

Templating matching correlates the tomograms with the template, confined to a mask. The template has a much smaller dimension than the tomogram and the mask has to be of the same dimension as the template. EM and mrc files are accepted as input. Maxima of the resulting correlation volume specify likely particle localization. To determine coordinates of maxima of the correlation volume and the corresponding orientations that yielded the highest correlation scores the corresponding output files need to be analyzed using the *Extract Candidates* box (Figure 16).



☑ Extract Candidates

Job File	ɔgram_000_WBP/job.xml
Score File	m_000_WBP/scores.em
Angles File	m_000_WBP/angles.em
Mask File	
File Name Particle List	omogram_000_WBP.xml
Prefix	'M_tomogram_000_WBP
Size particle (px)	15
Number of Candidates	300
Minimal Score Value	0.20

☑ sbatch

```
pytom/bin/extractCandidates.py \
--jobFile /home/friedrich/MyEriceProject/04_Particle_Picking/Template_Matching/cross_correlation/
tomogram_000_WBP/job.xml \
--result /home/friedrich/MyEriceProject/04_Particle_Picking/Template_Matching/cross_correlation/
tomogram_000_WBP/scores.em \
--orientation /home/friedrich/MyEriceProject/04_Particle_Picking/Template_Matching/cross_correlation/
tomogram_000_WBP/angles.em \
--particleList /home/friedrich/MyEriceProject/04_Particle_Picking/Picked_Particles/
particleList_TM_tomogram_000_WBP.xml \
--particlePath Subtomograms/particleList_TM_tomogram_000_WBP \
--size 15 \
--numberCandidates 300 \
```

Figure 16. Determining coordinates and orientations of putative particles. Choose the *Job File* from the template matching run in the dialog invoked by *Browse*, which fills *File Name Particle List* and *Prefix* (names of subtomograms) automatically. In this case, coordinates and orientations for the top 300 hits will be stored (if above a correlation coefficient of 0.2). *Size particle* is important to prevent particles from being picked twice. This radius should correspond to the diameter of the particle.

The underlying extraction scripts requires the input parameters of the template matching run. The *Job, Score and Angles File* are found in the folder `[myProject]/04_Particle_Picking/Template_Matching/cross_correlation/tomogram_xxx_WBP/`. The Mask File is optional to restrict the search to specific areas of interest.

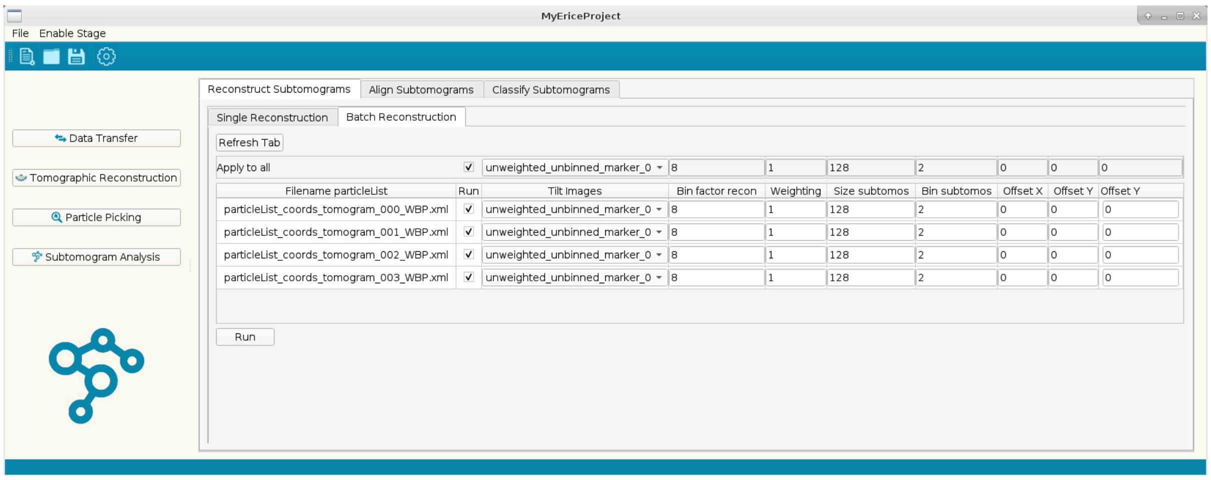
The individual particleLists can be combined into a single one using a script from the command line. Go to the directory, where you saved the particleLists (default: `[myProject]/04_Particle_Picking/Picked_Particles`) and then type `combineParticleLists.py`, which will display the help of this command. Use the argument `-f` followed by the names of the particleLists (separated by commas, no spaces) and `-o` to specify the output particleList (e.g., `combineParticleLists.py -f particleList_TM_tomogram_000_WBP.xml,particleList_TM_tomogram_001_WBP.xml,particleList_TM_tomogram_002_WBP.xml,particleList_TM_tomogram_003_WBP.xml -o particleList_TM_tomogram_combined.xml`).

5. Subtomogram Analysis

5.1. Reconstruct Subtomograms

First you need to reconstruct the subtomograms corresponding to the coordinates specified in the respective particleList. After enabling the stage *Subtomogram Analysis* you select

Reconstruct Subtomograms, and therein *Batch Reconstruction* to reconstruct particles from a number of particleLists. For the reconstruction you now make use of the CTF corrected aligned projections created previously (Figure 17).



Filename particleList	Run	Tilt Images	Bin factor recon	Weighting	Size subtomos	Bin subtomos	Offset X	Offset Y	Offset Y
Apply to all	✓	unweighted_unbinned_marker_0	8	1	128	2	0	0	0
particleList_coords_tomogram_000_WBP.xml	✓	unweighted_unbinned_marker_0	8	1	128	2	0	0	0
particleList_coords_tomogram_001_WBP.xml	✓	unweighted_unbinned_marker_0	8	1	128	2	0	0	0
particleList_coords_tomogram_002_WBP.xml	✓	unweighted_unbinned_marker_0	8	1	128	2	0	0	0
particleList_coords_tomogram_003_WBP.xml	✓	unweighted_unbinned_marker_0	8	1	128	2	0	0	0

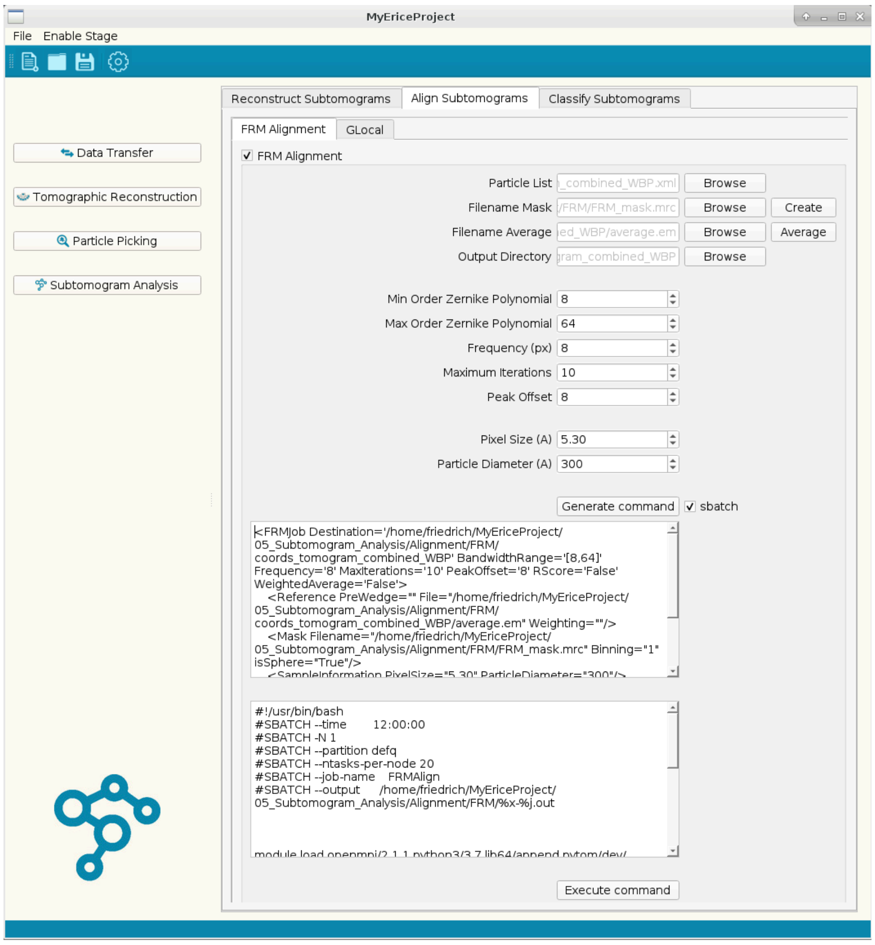
Figure 17. Batch reconstruction of subtomograms. The projections used for the reconstructions are in the third column. A pop-up window provides you with different options. Until recently, the folder name for uncorrected and CTF corrected aligned projections were the same – the last option refers to the CTF corrected one. In the present PyTomGUI this is changed to make distinction easier ;). *Bin factor recon* is the downsampling factor of the volume used for (manual) particle picking, in this case 8. *Weighting* is the weighting applied to the projections prior to reconstruction; '1' is the weighting that accounts for the angular sampling, '-1' the ramp-weighting and '0' no weighting (if projections are already weighted).

5.2. Subtomogram alignment

Alignment of subtomograms to a common coordinate system, which then allows meaningful averaging, relies on iterative algorithms. In PyTom, relatively simple quasi-expectation maximation approaches iteratively optimize the correlation of particles with an aligned average. In particular, the 3D rotational search is vast as three Euler angles need to be sampled. PyTom supports two flavors of iterative subtomogram alignment: Fast Rotational Matching (FRM) and real space alignment.

5.2.1. Fast Rotational Matching alignment

Subtomogram alignment by FRM allows exhaustive orientation sampling with high computational speed. This is achieved through transformation of the Fourier transformed of particles and reference to Spherical Harmonics space. The tab *Align Subtomograms* and the selection *FRM Alignment* opens the interface for this subtomogram alignment option (Figure 18). A main strength of the FRM alignment is that subtomogram alignment can succeed without an external reference. To this end we have randomized the orientations in the particleList (Figure 15), which allows generating an initial reference that would typically be some kind of sphere. Starting from this unbiased reference the iterative search should, at least for large complexes, converge.



✓ FRM Alignment

Particle List	_combined_WBP.xml
Filename Mask	/FRM/FRM_mask.mrc
Filename Average	ed_WBP/average.em
Output Directory	gram_combined_WBP
Min Order Zernike Polynomial	8
Max Order Zernike Polynomial	64
Frequency (px)	8
Maximum Iterations	10
Peak Offset	8
Pixel Size (A)	5.30
Particle Diameter (A)	300

✓ sbatch

```
<FRMJob Destination='/home/friedrich/MyEriceProject/
05_Subtomogram_Analysis/Alignment/FRM/
coords_tomogram_combined_WBP' BandwidthRange='[8,64]'
Frequency='8' MaxIterations='10' PeakOffset='8' RScore='False'
WeightedAverage='False'>
   <Reference PreWedge="" File="/home/friedrich/MyEriceProject/
05_Subtomogram_Analysis/Alignment/FRM/
coords_tomogram_combined_WBP/average.em" Weighting=""/>
   <Mask Filename="/home/friedrich/MyEriceProject/
05_Subtomogram_Analysis/Alignment/FRM/FRM_mask.mrc" Binning="1"
isSphere="True"/>
   <SampleInformation PixelSize="5.30" ParticleDiameter="300"/>
```

```
#!/usr/bin/bash
#SBATCH --time       12:00:00
#SBATCH -N 1
#SBATCH --partition defq
#SBATCH --ntasks-per-node 20
#SBATCH --job-name    FRMAlign
#SBATCH --output      /home/friedrich/MyEriceProject/
05_Subtomogram_Analysis/Alignment/FRM/%x-%j.out



module load openmpi/2.1.1 python3/3.7 lib64/append pytom/dev/
```

Figure 18. FRM-based subtomogram alignment and averaging. As *Particle List* choose the previously generated one, for which you also reconstructed the subtomogram. The *Mask* can be generated from scratch by clicking the *Create* button, which opens a dialog, or by choosing a pre-generated one (for example, generated in matlab using tom/av3). The initial reference (*Filename Average*) can also be auto-generated from the (randomized) particleList using the *Average* button. The spherical harmonics expansion is accelerated by restricting the computation to those bands that are needed. The limits of those bands ('quantum number' *l* in spherical harmonics). Broadly speaking, the internally used maximum *l* corresponds to $\sqrt{2\pi \times k}$, where k denotes the maximum Fourier frequency determined by FSC, or the bounds provided by *Max* and *Min Order*. *Frequency* is the low-pass used in the first iteration (in pixels), *Maximum iterations* specifies the number of iterations for the alignment, and *Peak offset* limits the shifts of subtomograms in the alignment procedure.

Above example applies the FRM alignment to membrane-associated ribosomes from the tutorial data. In this case, the procedure converges from a sphere to a ribosomal structure bound to a membrane (Figure 19).

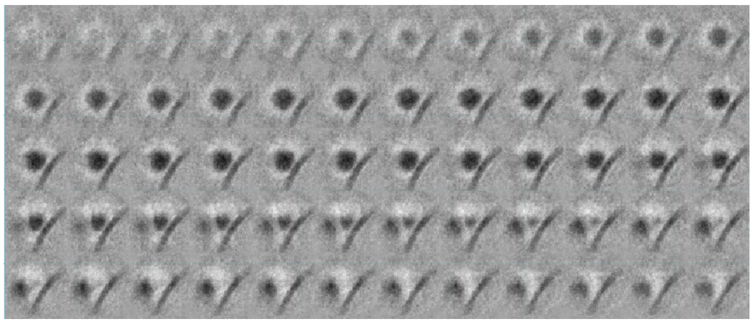
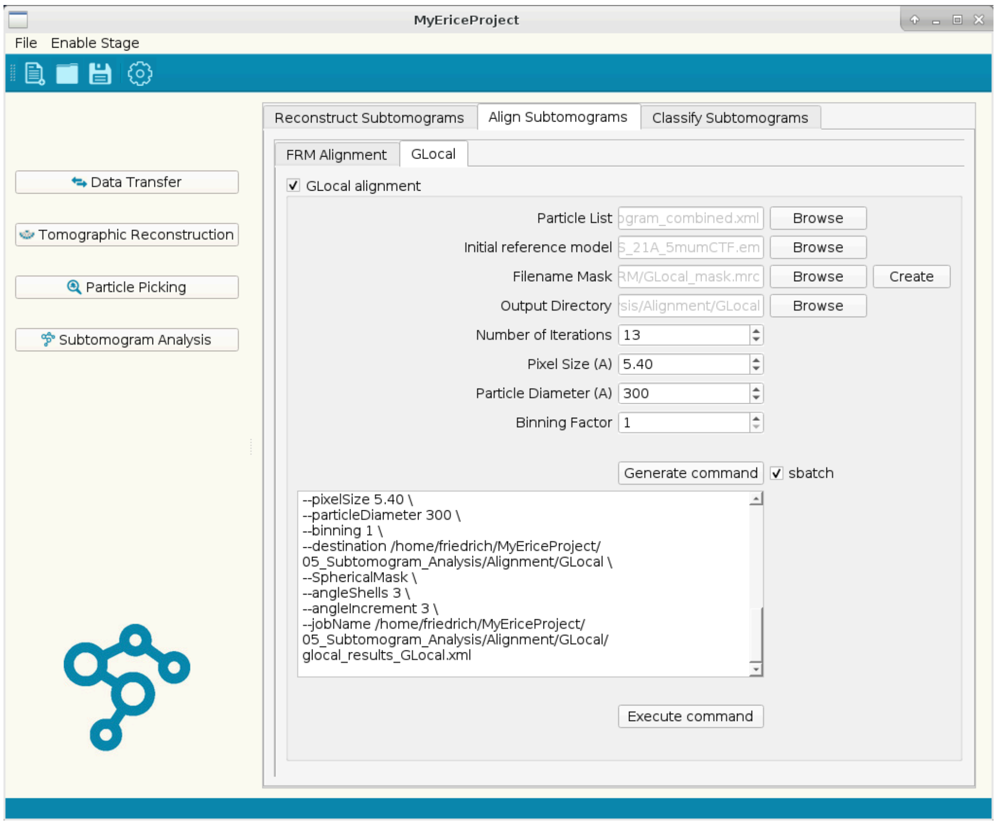


Figure 19. intermediate stage of FRM alignment for hand-picked membrane-bound ribosomes. The figure shows subsequent slices through the average after 10 FRM iterations (displayed with tom_dspcub in matlab/tom).

5.2.2. Real space alignment

Real space alignment tends to be more accurate because it can be focused on specific features of the molecule of interest with a mask, but the orientation sampling takes much longer in real space. As a consequence, only a limited range of orientations can be sampled in a single iteration, decreasing the radius of converge compared to the FRM alignment. Thus, real space alignment is a good option if the approximate orientations of the particles are known.

To use this approach, choose the tab *Align Subtomograms* with the specification *GLocal* (Gold-standard Local alignment) (Figure 20). Required input are a *Particle List* (with good approximations of orientations, for example from template matching or initial FRM alignment), a *Mask* (spherical mask can also be created in GUI). An initial external *reference model* can be provided or the average from the particleList will be used.



✓ GLocal alignment

Particle List	pgram_combined.xml
Initial reference model	S_21A_5mumCTF.em
Filename Mask	RM/GLocal_mask.mrc
Output Directory	sis/Alignment/GLocal
Number of Iterations	13
Pixel Size (A)	5.40
Particle Diameter (A)	300
Binning Factor	1

✓ sbatch

```
--pixelSize 5.40 \
--particleDiameter 300 \
--binning 1 \
--destination /home/friedrich/MyEriceProject/
05_Subtomogram_Analysis/Alignment/GLocal \
--SphericalMask \
--angleShells 3 \
--angleIncrement 3 \
--jobName /home/friedrich/MyEriceProject/
05_Subtomogram_Analysis/Alignment/GLocal/
glocal_results_GLocal.xml
```

Figure 20. Real space alignment menu. The menu *generates* the *command* for the GLocal alignment. At this point the GUI still requires you to specify a reference file – if you do want to use the average created from the data from the beginning you can just delete the reference in the generated command (this 'hack' will soon be obsolete …).

Results of the GLocal subtomogram alignment will be stored in `[myProject]/05_Subtomogram_Analysis/Alignment/GLocal/`. For each iteration, the average, the average filtered to the resolution according to FSC and the respective particleList will be stored. When applied to the particles found by template matching the average rapidly converges (Figure 21).

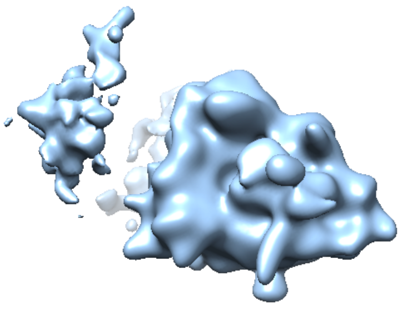


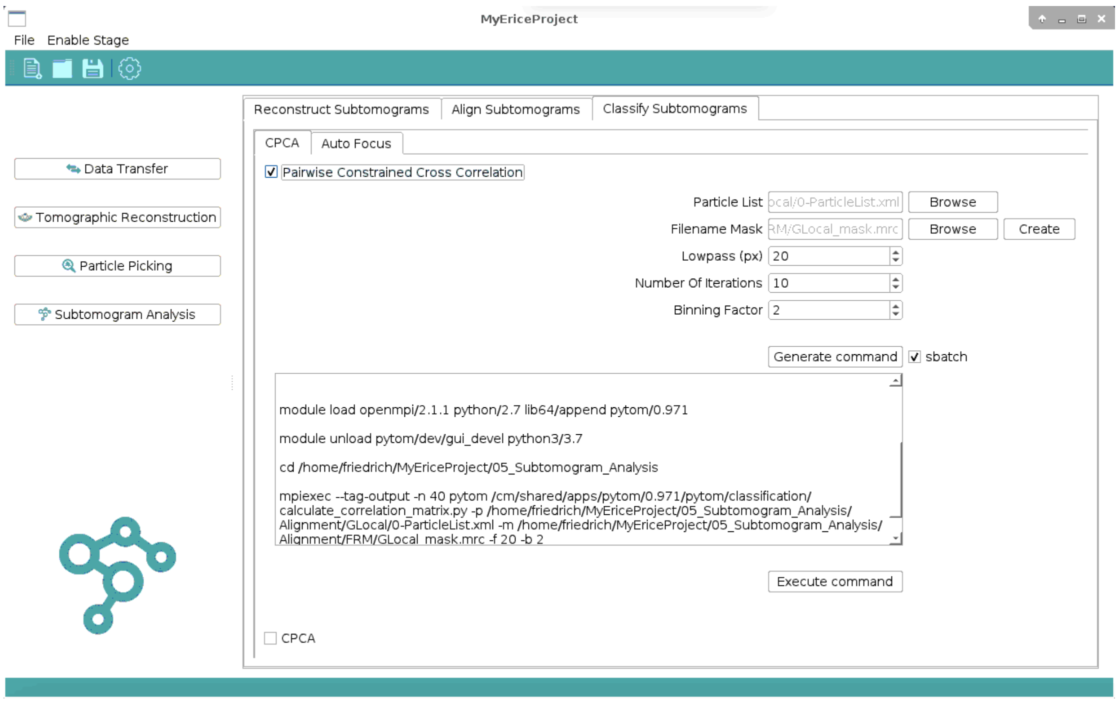
Figure 21. Subtomogram average resulting from template matching. The resolution rapidly converges to ~25 A. In proximity to the primary ribosome the density of neighboring ribosomes is visible.

5.3. Classification

Part of PyTom are two distinct classification approaches: Constrained Principal Component analysis-based classification (CPCA) and auto-focused classification. CPCA is designed to classify previously aligned subtomograms according to main features in a user-defined mask. Auto-focused classification aims to align subtomograms simultaneously with classification and to automatically focus the classification on the most variable parts of the data.

5.3.1. Constrained principal component analysis

CPCA classification is accessible by choosing *CPCA* in the *Classify Subtomograms* panel. CPCA classification comprises two major steps. The by far most time-consuming part is the computation of the constrained correlation matrix; each possible particle pair is correlated, considering the overlap of the particles in Fourier space. The result is a matrix, which is the basis of the subsequent classification. This step is accessed by checking the *Pairwise Constrained Cross Correlation* box (Figure 22).



[✓] Pairwise Constrained Cross Correlation

Particle List	ocal/0-ParticleList.xml
Filename Mask	RM/GLocal_mask.mrc
Lowpass (px)	20
Number Of Iterations	10
Binning Factor	2

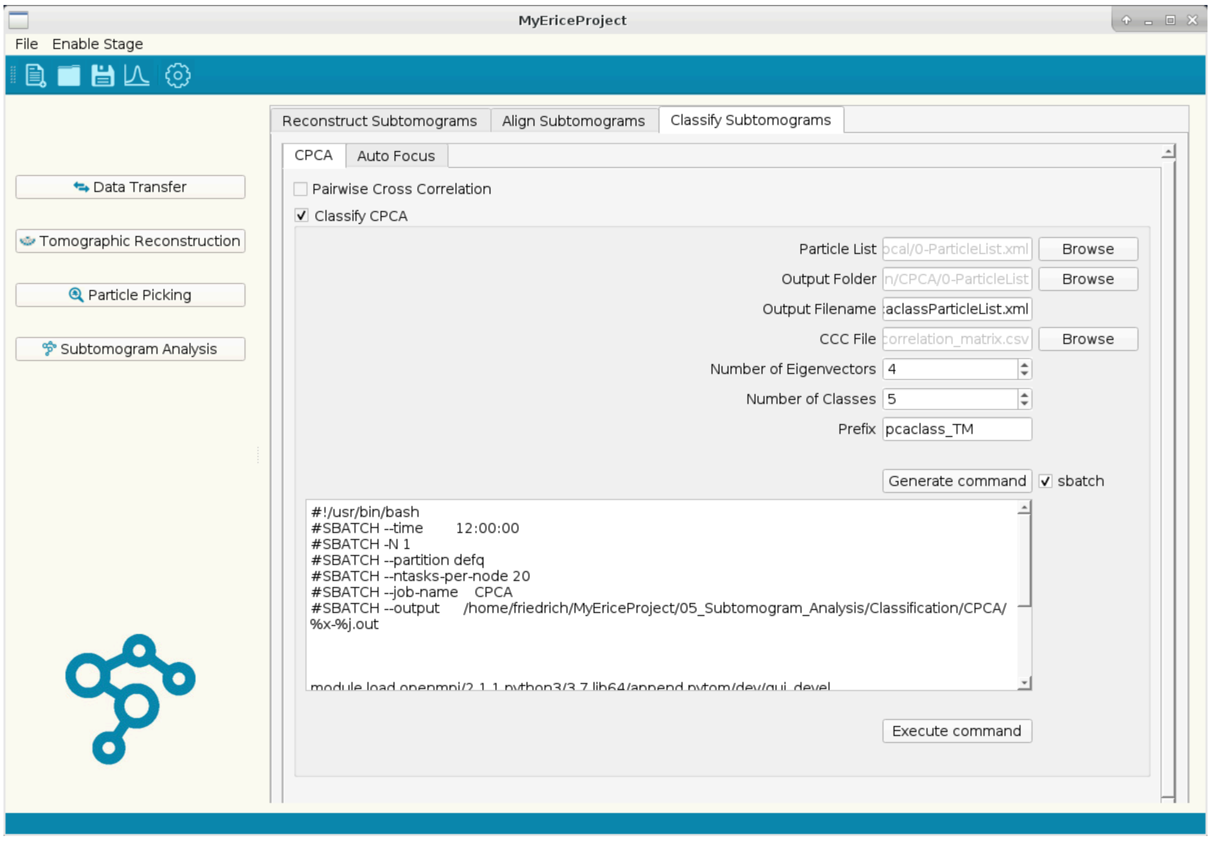
[✓] sbatch

```
module load openmpi/2.1.1 python/2.7 lib64/append pytom/0.971

module unload pytom/dev/gui_devel python3/3.7

cd /home/friedrich/MyEricProject/05_Subtomogram_Analysis

mpiexec --tag-output -n 40 pytom /cm/shared/apps/pytom/0.971/pytom/classification/
calculate_correlation_matrix.py -p /home/friedrich/MyEriceProject/05_Subtomogram_Analysis/
Alignment/GLocal/0-ParticleList.xml -m /home/friedrich/MyEriceProject/05_Subtomogram_Analysis/
Alignment/FRM/GLocal_mask.mrc -f 20 -b 2
```

[ ] CPCA

Figure 22. Computing the cross correlation matrix. *Particle List* comprises the particles to be classified. A *Mask* must be specified to target the correlation – and classification – to a specific area. A lowpass reduces noise beyond the resolution of interest and Binning is recommended to accelerate computations whenever possible (downscaling is performed in Fourier space).

Subsequently, the particles are classified based on the resulting matrix by checking the *Classify CPCA* box (Figure 23). Firstly, the eigenvectors of the matrix are computed and the projections of the particles to the selected set. K-means classification in this highly reduced space then groups the data into a desired number of bins. The outcome of the classification procedure are class averages and corresponding particle lists, which can be found in the pre-generated folders.



Pairwise Cross Correlation

✓ Classify CPCA

Particle List	pcal/0-ParticleList.xml
Output Folder	n/CPCA/0-ParticleList
Output Filename	aclassParticleList.xml
CCC File	correlation_matrix.csv
Number of Eigenvectors	4
Number of Classes	5
Prefix	pcaclass_TM

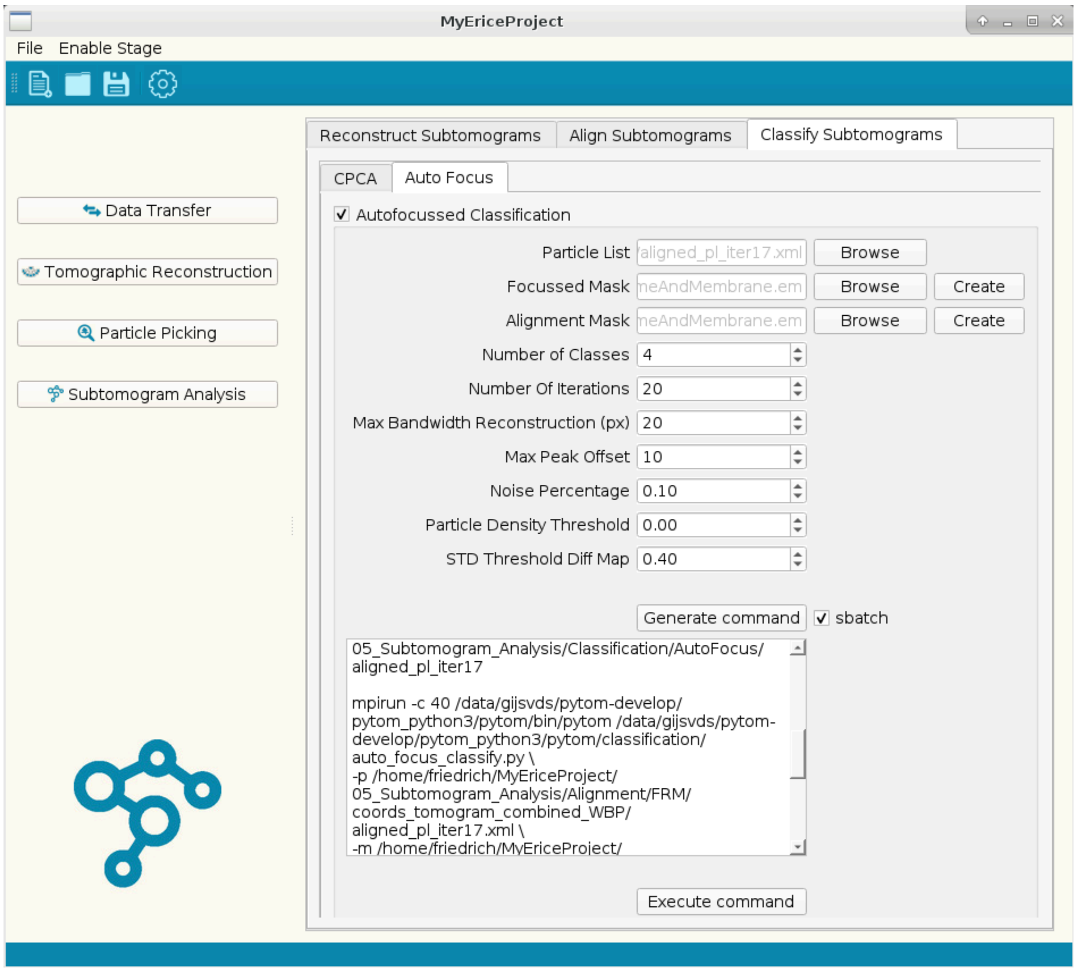
✓ sbatch

```
#!/usr/bin/bash
#SBATCH --time       12:00:00
#SBATCH -N 1
#SBATCH --partition defq
#SBATCH --ntasks-per-node 20
#SBATCH --job-name    CPCA
#SBATCH --output      /home/friedrich/MyEriceProject/05_Subtomogram_Analysis/Classification/CPCA/
%x-%j.out



module load openmpi/2.1.1 python3/3.7 lib64/append pytom/dev/gui_devel
```

Figure 23. CPCA-based k-means classification. *Particle List* is list of particles that has been used to compute the correlation matrix. The Output Folder is then auto-generated to make sure it refers to the particleList you have worked with. *CCC File* is the correlation matrix, which has been computed with above box (Figure 21*). Number of Eigenvectors* specifies the principal components; here the four eigenvectors corresponding to the four highest eigenvalues of the correlation matrix are used to project the data. Subsequently, k-means clustering with the specified *Number of Classes* is carried out.

5.3.1. Autofocused-classification

Above CPCA classification requires accurately pre-aligned particles and typically prior knowledge on which feature to classify for. Auto-focused classification is an attempt to alleviate these requirements. It uses the FRM subtomogram alignment to align the particles during the alignment. Furthermore, the procedure continuously monitors the variance between the class averages, where the classification is focused on (hence the name: auto-focused). Class assignments are based on a voting procedure: pairwise comparisons are made for each investigated particle (i.e., the particle is compared to class 1 and 2, class 1 and 3, class 2 and 3, and so forth) and the particle is assigned to the class that has most 'wins' in these comparisons. You access this classification option through the tab *Auto Focus* in *Classify Subtomograms* (Figure 24).



☑ Autofocussed Classification

Particle List: aligned_pl_iter17.xml
Focussed Mask: neAndMembrane.em
Alignment Mask: neAndMembrane.em
Number of Classes: 4
Number Of Iterations: 20
Max Bandwidth Reconstruction (px): 20
Max Peak Offset: 10
Noise Percentage: 0.10
Particle Density Threshold: 0.00
STD Threshold Diff Map: 0.40
☑ sbatch

```
05_Subtomogram_Analysis/Classification/AutoFocus/
aligned_pl_iter17

mpirun -c 40 /data/gijsvds/pytom-develop/
pytom_python3/pytom/bin/pytom /data/gijsvds/pytom-
develop/pytom_python3/pytom/classification/
auto_focus_classify.py \
-p /home/friedrich/MyEriceProject/
05_Subtomogram_Analysis/Alignment/FRM/
coords_tomogram_combined_WBP/
aligned_pl_iter17.xml \
-m /home/friedrich/MyEriceProject/
```

Figure 24. Auto-focused classification. Autofocused classification is performed on the specified *Particle List*. The *Focussed Mask* allows to restrict the classification area; for example, membranes can be flexible and you may want to exclude this area from the classification to rather focus on other structural features. The *Alignment Mask* is required for the subtomogram alignment at each classification iteration. The *Noise Percentage* is an estimation of particles that likely do not correspond to any meaningful class and should rather be assigned to a 'junk' basket (here 10%). The *STD Threshold Diff Map* determined the auto-focused mask: areas above this threshold are used for the classification (here 0.4 times the standard deviation). The Particle Density threshold focuses on particularly negative densities – typically classification targets densities that are present / absent in the classes at this resolution.

This user guide is meant to serve as a first introduction into PyTom and its GUI. The aim is to familiarize with the workflow and commands. Experienced users will eventually use commands from the command line, making use of the pre-defined folder structure of the GUI and in particular of its batch functions.

Have Fun!