

# SIMPLE Tutorial 2017

Dominika Elmlund, Hans Elmlund

June 30, 2017

## Contributors:

cyril.reboul@monash.edu  
michael.eager@monash.edu  
dominika.elmlund@monash.edu  
hans.elmlund@monash.edu

## Address:

Dept. Biochemistry and Molecular Biology  
School of Biomedical Sciences  
Monash University, Bldg. 77  
Clayton, VIC, Australia, 3800

## Webpage:

[www.simplecryoem.com](http://simplecryoem.com)

## Contact:

<http://simplecryoem.com/contact.html>  
dominika@simplecryoem.com

## 1 From Movies to *ab initio* 3D reconstruction

These steps describe a typical SIMPLE workflow.

1. DDD (Direct Detector Device) movie alignment and frame-weighting using SIMPLE program `unblur`, executed with `simple_distr_exec`
2. CTF parameter identification with the SIMPLE program `ctffind`, wrapping CTFFIND4 (Rohou and Grigorieff, 2015), executed with `simple_distr_exec`
3. Particle identification using EMAN2 (Tang et al., 2007) to generate \*.box files
4. Particle extraction with SIMPLE program `extract`, executed with `simple_exec`
5. 2D analysis using the SIMPLE `prime2D` distributed workflow, executed with `simple_distr_exec`
6. *Ab initio* 3D reconstruction from class averages using the SIMPLE `ini3D_from_cavgs` distributed workflow, executed with `simple_distr_exec`
7. Mapping of class average selection and 3D class orientations to the particles using SIMPLE program `map2ptcls`, executed with `simple_exec`

8. Reconstruction of a 3D map from the individual particle images with SIMPLE program `recvol`, executed with `simple_distr_exec`
9. Map refinement will be part of release 3.0

## 1.1 This Tutorial will Cover

1. DDD (Direct Detector Device) movie pre-processing
2. 2D analysis with PRIME2D
3. *ab initio* 3D reconstruction from class averages using `ini3D_from_cavgs`

with the SIMPLE program package version 2.5 available for download on the SIMPLE homepage  
<http://simplecryoem.com>.

## 1.2 Execution of SIMPLE

SIMPLE is executed primarily via `simple_distr_exec` that implements higher level workflows intended for distributed execution on workstations and clusters using a hybrid parallelisation model (distributed *and* shared memory). SIMPLE can also be executed via `simple_exec`, which implements all individual SIMPLE programs and runs in shared-memory parallelisation mode. Please, beware that the high-level workflows (`prime2D` and `ini3D_from_cavgs`, for example) can *only* be executed via the distributed execution route. In cluster environments using a job scheduler (PBS, SLURM and SGE are supported by SIMPLE) the file `simple_distr_config.env` in the current working directory controls the execution.

```
bash-3.2$ cat simple_distr_config.env
# CONFIGURATION FILE FOR DISTRIBUTED SIMPLE EXECUTION

# ABSOLUTE PATH TO SIMPLE ROOT DIRECTORY
simple_path      = /scratch/m3earlyadopters/simple/simple/

# ESTIMATED TIME PER IMAGE (IN SECONDS)
time_per_image   = 600

# USER DETAILS
user_account     = e185
user_email       = hans.elmlund@monash.edu
user_project     =

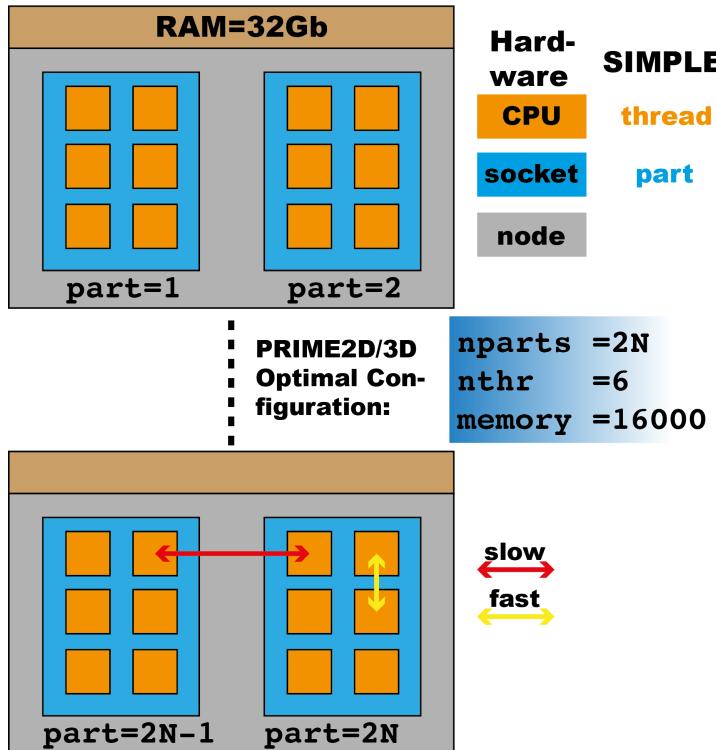
# QSYS DETAILS (qsys_name=<local|slurm|pbs>)
qsys_name        = slurm
qsys_partition   = m3a
qsys_qos         =
qsys_reservation = simple

# JOB DETAILS
job_ntasks       = 1
job_memory_per_task = 48000
job_name         = taf-dna-part2
job_ntasks_per_socket = 1
```

This file is auto-generated on workstations, but it needs to be edited by the user in cluster environments. If you need help configuring distributed SIMPLE execution, please file a help ticket on the webpage. The two most important parameters for distributed execution is the

number of partitions `nparts` and the number of shared-memory CPU threads `nthr`. A rule of thumb for good performance on multi-socket workstations is to minimise the number of parts and maximising the number of threads subject to never having fewer parts than sockets. The shared memory parallelisation does not benefit from the use of a larger number of threads than the number of logical threads available on a socket. To check the number of processors on a linux system, execute `nproc` in the terminal. Consider a heterogeneous cluster with  $N$  nodes, two CPU sockets per node and six CPUs per socket. If you are unsure how to configure your

### A heterogeneous cluster of $N$ nodes



**Figure 1: Configuration of the parallel PRIME-2D/3D execution on a heterogeneous cluster.** We here represent the nodes in a heterogeneous cluster by two sockets with six CPUs each and 32Gb RAM/node. The best performance of PRIME-2D/3D is going to be obtained by partitioning the jobs into `npart=2N` partitions, where  $N$  is the number of nodes. Each partition will then execute six threads `nthr=6` and these six threads will get access to half the RAM on the node (`memory=16000`) because we have two sockets per node that need to share the RAM between them

SIMPLE execution please file a help ticket.

## 2 DDD Movie Pre-processing

### 2.1 Motion Correction

During image acquisition, beam induced motion, charging and stage drift introduce blurring in the integrated movies. The individual frames need to be aligned with respect to one another in order to restore high-resolution information. Because of radiation damage, the electron dose is kept low during image acquisition, resulting in a very low signal-to-noise ratio (SNR) of the individual movie frames. SIMPLE implements a program called `unblur` for simultaneous motion-correction, frame-weighting and dose-weighting, with the objective of maximising the SNR of the integrated movie in a motion- and dose-dependent manner. Input is a text file `filetab`, simply listing the individual movies in `*.mrc` format, the pixel-to-pixel (or sampling) distance `smpd` (in Å), the number of CPU threads to use `nthr`, the template name of the outputted files `fbody`, in addition to a number of other optional parameters that we will not be concerned with here. If we execute `simple_distr_exec prg=unblur` in the prompt, we obtain brief instructions for how to run the program:

```
bash-3.2$ simple_distr_exec prg=unblur_movies
USAGE:
bash-3.2$ simple_distr_exec prg=simple_program key1=val1 key2=val2 ...
```

REQUIRED

```

filetab = list of files(*.txt/*.asc)
smpd    = sampling distance, same as EMANs apix(in A)
nparts  = # partitions in distributed execution

OPTIONAL
nthr     = # OpenMP threads{1}
ncunits  = # computing units, can be < nparts{nparts}
fbody    = file body
dose_rate = dose rate(in e/A2/s)
exp_time = exposure time(in s)
lpstart   = start low-pass limit(in A){15}
lpstop    = stop low-pass limit(in A){8}
trs       = maximum halfwidth shift(in pixels)
kv        = acceleration voltage(in kV){300.}
pspecsz  = size of power spectrum(in pixels)
numlen   = length of number string
startit   = start iterating from here
scale     = image scale factor{1}
nframesgrp = # frames to group before unblur(Falcon 3){0}
fromf    = start frame index
tof      = stop frame index
nsig     = # sigmas

```

Output consists of four files per movie (`movie1.mrc` in this example):

1. `<fbody>_intg1.mrc` is the frame-weighted, motion-corrected sum
2. `<fbody>_forctf1.mrc` is the un-weighted sum of the aligned individual frames
3. `<fbody>_pspec1.mrc` is the power-spectrum of the uncorrected unweighted movie sum (left) and the corrected weighted movie sum (right)
4. `<fbody>_thumb1.mrc` is a down-scaled version of `<fbody>_intg1.mrc`

The `*forctf*` output is created because weighting the frames may degrade the contrast transfer function (CTF) signal and this file will therefore later be used to determine the CTF parameters of the integrated movie. The `*intg*` image is the dose-weighted one you will use for particle picking and extract your identified individual particle images from. The `*pspec*` and `*thumb*` outputs are diagnostic. Please, check so that the Thon rings are concentric and have similar radial intensity distribution in any given resolution shell after alignment (right part of power-spectrum)—if not, trash the image. If you have downloaded the example data folder, please execute

```

bash-3.2$ cd 1_DDD-movie-proc/
bash-3.2$ ls
data
bash-3.2$ ls ./data/
4comparison boxfiles info.txt movie1.mrc movie2.mrc
bash-3.2$ cat ./data/info.txt
molecule: proteasome
exp_time=7.6s
dose_rate=7e/A2/s
kv=300
cs=2.7
smpd=5.26
fraca=0.1

```

```

bash-3.2$ ls ./data/movie* > movies.txt
bash-3.2$ cat movies.txt
./data/movie1.mrc
./data/movie2.mrc

```

These movies are of proteasome molecules in ice (EMPIAR-10025), exposed for `exp_time=7.6` s with a dose rate of `dose_rate=7 e/Å2/s`. The electron microscope used to acquire these images has an acceleration voltage of `kv=300` kV and a spherical aberration constant of `cs=2.7` mm. We have downsampled these images for more rapid processing to a sampling distance of `smpd=5.26` Å and we will assume 10% amplitude contrast when fitting the CTF `fraca=0.1`. First, we need to create a file table, listing the movies to be processed with `unblur`.

```
bash-3.2$ ls data/movie* > movies.txt
```

To see what this file contains, execute:

```

bash-3.2$ cat movies.txt
data/movie1.mrc
data/movie2.mrc

```

Next, we execute the program.

```

bash-3.2$ simple_distr_exec prg=unblur filetab=movies.txt smpd=5.26
nparts=2 nthr=12 fbody=proteasome dose_rate=7 exp_time=7.6 kv=300
fbody=proteasome

```

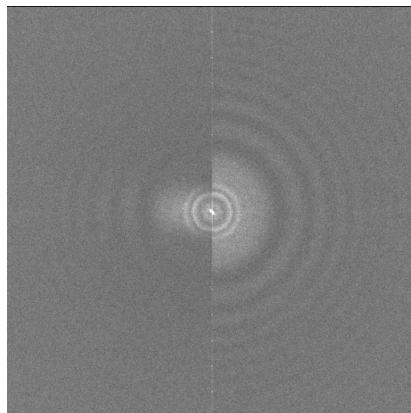
The first thing that happens is that all individual movie frames are read in and Fourier transformed. The algorithm is iterative and based on registration of the individual frames to a weighted average, where the weights are determined based on the agreement between the frames and the average. The algorithm automatically updates the resolution limit as the alignment accuracy improves, *i.e.* using only low-resolution Fourier components at first when the alignment errors are large and large movements need to be identified. It then successively updates the resolution limit as the alignment accuracy improves. When no improvements can be identified, the procedure stops. The outputted image files are:

```

bash-3.2$ ls
data                      proteasome_forctf2.mrc  proteasome_pspec2.mrc
movies.txt                 proteasome_intg1.mrc   proteasome_thumb1.mrc
simple_unidoc.txt          proteasome_intg2.mrc   proteasome_thumb2.mrc
proteasome_forctf1.mrc    proteasome_pspec1.mrc  simple_distr_config.env

```

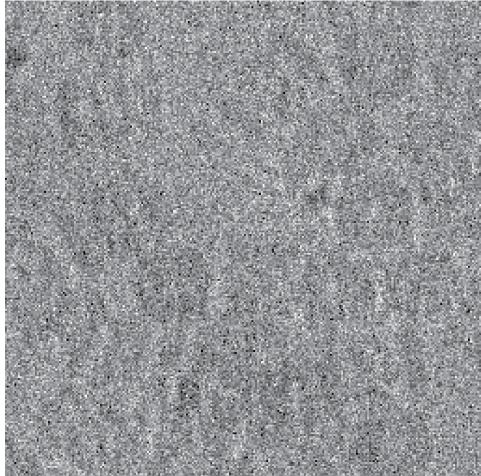
The first images we inspect are the power spectra. For visualisation, use the EMAN2 program `e2display.py`. Execute `e2display.py` in the current working directory, click on one of the `*pspec*` files and then `Show 2D`. To get an idea of the effect of the movie pre-processing in real-



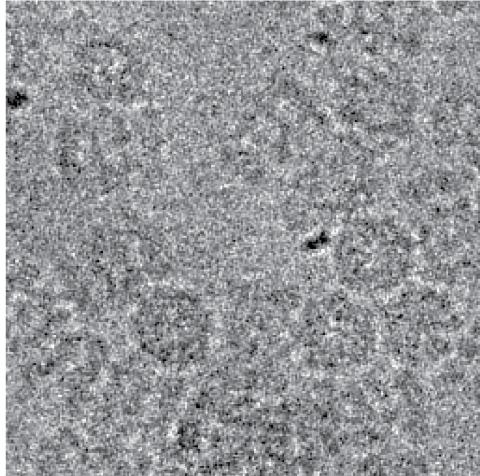
**Figure 2:** The left part of the power spectrum represents the movie integrated without applying the alignment parameters, whereas the right part is the aligned integrated movie. We see that the motion correction restores coherence, giving radially symmetric Thon rings extending to much higher resolution.

space, expand the `data` folder from within `e2display.py`, then expand the `4comparison` folder and click on the `proteasome_straight_intg1.mrc` file, followed by `Show 2D+`. Do the same for the corrected movie `proteasome_intg1.mrc` and compare the two images.

Uncorrected



Corrected



**Figure 3:** The uncorrected movie, integrated without alignment, is blurred and lacks high-resolution contrast, whereas the corrected movie has much higher contrast and better definition.

## 2.2 Using SIMPLE in the Wild—Power Spectrum Analysis and Movie Selection

In a real-life scenario, all images in a data set are never perfect. A truck may come driving by the EM suite and cause drift, someone may pass the EM room transporting a big piece of metal, causing the energy filter to become misaligned, or the pump of the autoloader may have been accidentally turned on. The list could be made long. Therefore, it is important that we take a good look at our data. The first round of screening is usually done by looking at power spectra—the `*pspec*` files produced by `unblur`. Typically, you will have executed `unblur` in distributed mode in a cluster environment (described below) and a stack of power spectra called `unblur_pspeccs.mrc` has then been produced. We use EMAN2 to look at the power spectra but you can use whatever GUI you like to make the selection. SIMPLE supports MRC and SPIDER file formats but does not allow you to mix formats, so if you need to convert between MRC and SPIDER file formats, use SIMPLE program `convert`

```
bash-3.2$ simple_exec prg=convert  
USAGE:  
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...
```

### OPTIONAL

```
stk      = particle stack with all images(ptcls.ext)  
vol1    = input volume no1(invvol1.ext)  
outstk  = output image stack  
outvol  = output volume{outvol.ext}
```

Once you have made the selection and have the two stacks `unblur_pspeccs.mrc` and `good_pspeccs.mrc` or whatever you select to call them you can propagate the selection to your `*intg*` and `*forctf*` files using SIMPLE program `select`

```
bash-3.2$ simple_exec prg=select  
USAGE:  
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...
```

### REQUIRED

```
stk  = particle stack with all images(ptcls.ext)
```

```
stk2 = 2nd stack(in map2ptcls/select: selected(cavgs).ext)
```

#### OPTIONAL

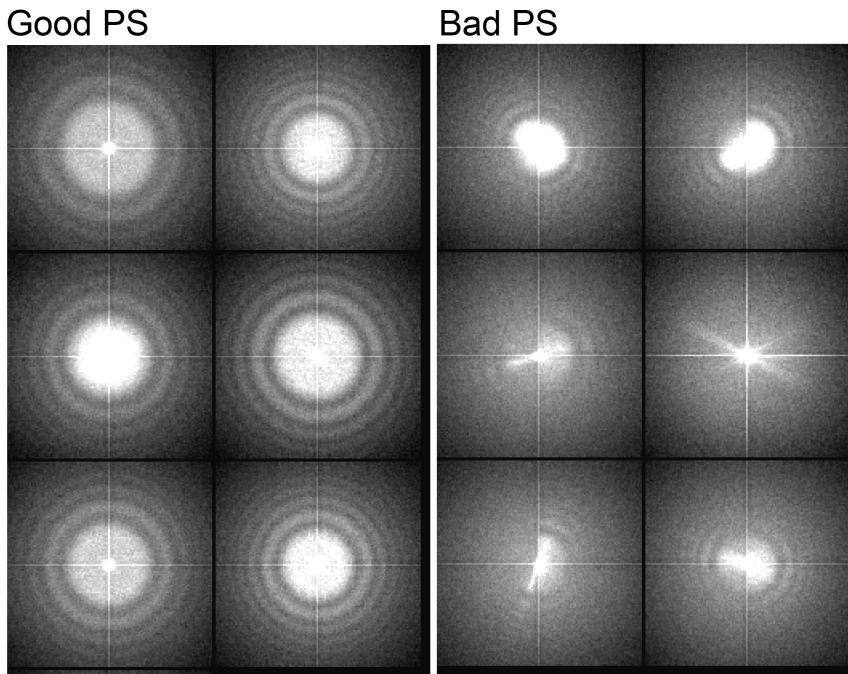
```
nthr      = nr of OpenMP threads{1}
stk3      = 3d stack (in map2ptcls/select: (cavgs)2selectfrom.ext)
filetab   = list of files(*.txt/*.asc)
outfile   = output document
outstk    = output image stack
dir_select = move selected files to here{selected}
dir_reject = move rejected files to here{rejected}
```

You would then execute `select` twice. In the first pass to select your `*intg*` files:

```
bash-3.2$ ls *intg* > intg_movies.txt
bash-3.2$ simple_exec prg=select stk=unblur_pspegs.mrc stk2=good_pspegs.mrc nthr=8
filetab=intg_movies.txt dir_select=intg_selected dir_reject=intg_rejected
```

and in the second pass to select your `*forctf*` files:

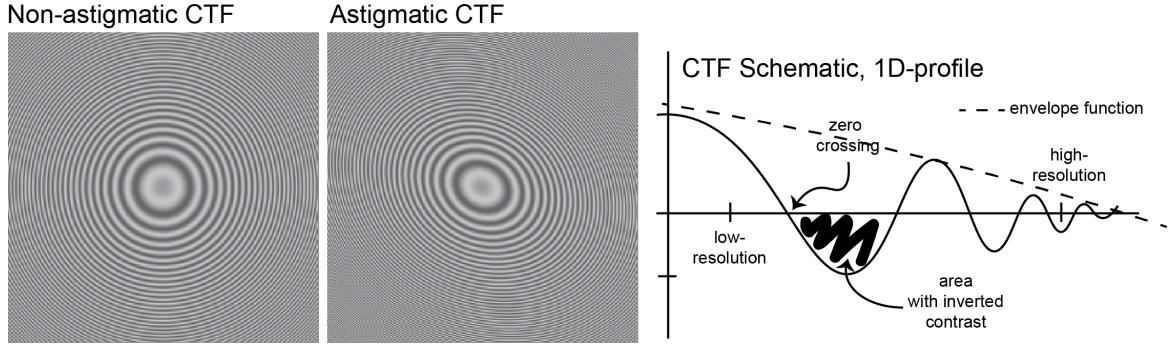
```
bash-3.2$ ls *forctf* > forctf_movies.txt
bash-3.2$ simple_exec prg=select stk=unblur_pspegs.mrc stk2=good_pspegs.mrc nthr=8
filetab=forctf_movies.txt dir_select=forctf_selected dir_reject=forctf_rejected
```



**Figure 4:** Examples of good power spectra (left) and bad power spectra (right)

## 2.3 CTF Parameter Determination

The contrast transfer function (CTF) mathematically describes how aberrations in a transmission electron microscope modify the recorded image. The assumption we rely on is that the image is a projection of the Coulomb potential distribution of the specimen, convoluted (multiplied in Fourier space) with the CTF and with Gaussian noise added. All 3D reconstruction methodology rely on this or slight variations of this assumption. The CTF is an oscillatory function (we saw the effect of it in the power spectrum in Figure 1) and the frequency of the oscillations depends on the spherical aberration constant `cs` in mm, which is an instrument specific constant, and the defocus in microns, which we systematically vary during image acquisition. The reason that we vary the defocus during data collection is that we want to avoid systematic lack of information



**Figure 5:** Non-astigmatic CTF (left) vs. astigmatic CTF (middle) and schematic 1D CTF profile (right)

in certain resolution regions, as no information other than noise is present in the zero crossings of the CTF. By varying the defocus we shift the positions of the zero crossings of the CTF. If astigmatism is present in the image, the Thon rings are no longer radially symmetric but elliptical and we need to take this into account as well.

The CTF has two components: one phase contrast component (giving rise to the same kind of contrast as in a medical X-ray image) and one amplitude contrast component (giving rise to the same kind of contrast as when you put your hand on a projector, omitting photons to create a shadow image). Most of the contrast in biological cryo-EM are due to change in phase. In fact, the fraction of amplitude contrast has been measured to be around 10% for proteins. In summary, we need six parameters to determine the parameters of the CTF: the acceleration voltage `kv`, the spherical aberration constant `cs`, the fraction of amplitude contrast `fraca`, the defocus in the x-direction `dfx`, the defocus in the y-direction `dfy` and the angle of astigmatism `angast`. The `kv`, `cs`, and `fraca` we treat as constants. Although we measure the defocus and astigmatism during data acquisition, the accuracies of these measurements are too crude to be useful for image processing. However, the protein itself, the carbon support and the ice scatters sufficiently for us to be able to generate a power spectrum that accurately represents the CTF power. This is typically done by extracting overlapping boxes from the micrograph, for every box calculating a power spectrum and averaging all power spectra over a micrograph to enhance the CTF signal. Using a parametric model for the CTF of the form

$$CTF = A_{phase} \sin(\phi) + A_{amp} \cos(\phi) \quad (1)$$

where  $A_{phase} = 0.9$ ,  $A_{amp} = 0.1$  and the phase shift  $\phi$

$$\phi = \pi \lambda s^2 (D - \frac{\lambda^2 s^2 C_s}{2}) \quad (2)$$

where  $\lambda$  is the electron wavelength,  $s$  is spatial frequency,  $D$  is defocus and  $C_s$  is the spherical aberration constant, we can determine the parameters we need.

SIMPLE implements a wrapper program called `ctffind` that executes CTFFIND4 (Rohou and Grigorieff, 2015) version 4.1.X, producing a SIMPLE conforming CTF parameter document `deftab`. Input is a text file `filetab`, simply listing the individual integrated movies in `*.mrc` format, the pixel-to-pixel (or sampling) distance `smpd` (in Å), the acceleration voltage of the electron microscope `kv` (in kV), the spherical aberration constant `cs` in mm and the fraction of amplitude contrast `fraca` in addition to a number of other optional parameters that we will not be concerned with here. If we execute `simple_distr_exec prg=ctffind` in the prompt, we obtain brief instructions for how to run the program:

```
bash-3.2$ simple_exec prg=ctffind
USAGE:
bash-3.2$ simple_distr_exec prg=simple_program key1=val1 key2=val2 ...
```

```

REQUIRED
filetab = list of files(*.txt/*.asc)
smpd    = sampling distance, same as EMANs apix(in A)
kv      = acceleration voltage(in kV){300.}
cs      = spherical aberration constant(in mm){2.7}
fraca   = fraction of amplitude contrast used for fitting CTF{0.07}
nparts  = # partitions in distributed execution

```

```

OPTIONAL
ncunits   = # computing units, can be < nparts{nparts}
pspecsz   = size of power spectrum(in pixels)
hp        = high-pass limit(in A)
lp        = low-pass limit(in A)
dfmin     = minimum expected defocus(in microns)
dfmax     = maximum expected defocus(in microns)
astigstep = step size for astigmatism search(in microns)
expastig   = expected astigmatism(in microns)
phaseplate = images obtained with phaseplate(yes|no){no}

```

Output is a text file with defocus parameters `ctffind_output.txt`. First, we need to create a file table, listing the integrated movies to be processed with `ctffind`.

```
bash-3.2$ ls proteasome_forctf* > ctf_movies.txt
```

As explained above, the `*forctf*` output is created because weighted averaging may degrade the contrast transfer function (CTF) signal and this un-weighted average is therefore used to determine the CTF parameters of the integrated movie. Next, we execute the program.

```
bash-3.2$ simple_distr_exec prg=ctffind filetab=ctf_movies.txt smpd=5.26
kv=300 cs=2.7 fraca=0.1 nparts=2
```

Output is the file `ctffind_output.txt`.

In SIMPLE, we have yet to implement a particle picker but rely on EMAN1.9 and EMAN2 for particle identification. However, if you have `*.box` files obtained with EMAN you can extract the particle images and their associated CTF parameters using SIMPLE. To illustrate this functionality, we have prepared box files in the `data` folder for use in conjunction with SIMPLE program `extract`. As before, we need to prepare file tables. First, we prepare a file table for the motion corrected integrated movies:

```
bash-3.2$ ls proteasome_intg* > intg_movies.txt
```

Second, for the box files:

```
bash-3.2$ ls ./data/boxfiles/*box > boxfiles.txt
```

Finally, we execute the program:

```
bash-3.2$ simple_exec prg=extract filetab=intg_movies.txt boxtab=boxfiles.txt
smpd=5.26 ctffind_doc=ctffind_output.txt
>>> DONE PROCESSING PARAMETERS
>>> DONE BUILDING GENERAL TOOLBOX
100% |=====| done.

----- ----- -----
|_____| | | | | |_____| | |____|
-----|__|_| | | | | |_____| |____|
-----
```

\_)\_ ( \_ - ) o \_ - - - - o - - -

(\_ ) ) )\_) ( ( ) ) (\_( \) )\_) ) ) (\_( ( ) ) )\_)  
(\_ ) ) )\_) ( ( ) ) (\_( \) )\_) ) ) (\_( ( ) ) )\_)

\*\*\*\* SIMPLE\_EXTRACT NORMAL STOP \*\*\*\*

Output is a particle stack `sumstack.mrc` and its associated parameter file `extract_params.txt`.

### 3 2D Analysis with PRIME2D

Algorithms that can rapidly discover clusters corresponding to sets of images with similar projection direction and conformational state play an important role in single-particle analysis. Identification of such clusters allows for enhancement of the signal-to-noise ratio (SNR) by averaging and gives a first glimpse into the character of a dataset. Therefore, clustering algorithms play a pivotal role in initial data quality assessment, *ab initio* 3D reconstruction and analysis of heterogeneous single-particle populations. SIMPLE implements a probabilistic algorithm for simultaneous 2D alignment and clustering, called [prime2D](#). The version we are going to use here is an improved version of the published code released in SIMPLE 2.1 (submitted manuscript). Grouping tens of thousands of images into several hundred clusters is a computationally intensive job. Therefore, we will now introduce the execution route for running SIMPLE jobs in a distributed computing environment.

### 3.1 PRIME2D Analysis of TRPV1 Membrane Receptor Images

The computers we are currently using have four i7 processors in a single socket, hyper-threaded so it looks like we have eight. We are now going to pretend that we have two CPU sockets with four CPUs each for the purpose of demonstration. Go to the PRIME2D directory and check things out.

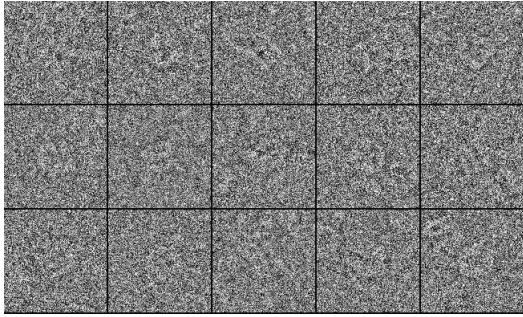
```
bash-3.2$ cd ../_2_PRIME2D/
bash-3.2$ ls
data/
bash-3.2$ ls data/
info.txt          trpv1_extract_params.txt trpv1_ptcls.mrc
bash-3.2$ cat data/info.txt
smpd=2.43
kv=300.
cs=2.0
fraca=0.1
ctf=flip
msk=36
box=128
```

The key-value pair `ctf=flip` means that the images have been “phase-flipped”, *i.e.* the contrast bands inverted by the CTF have been reverted simply by multiplying the Fourier transform of the image with the sign of the CTF. The individual particle images are quite noisy and it is difficult to gain any understanding about the imaged structure from these images, which represent a subset of particles from a larger TRPV1 data set (EMPIAR-10005). In order to cluster the images, execute:

```
bash-3.2$ simple_distr_exec prg=prime2D stk=./data/trpv1_ptcls.mrc smpd=2.43 msk=36  
ncls=5 ctf=flip nparts=2 nthr=4 deftab=./data/trpv1_extract_params.txt
```

For every iteration, we expect to see the following output in the terminal:

```
>>>  
>>> ITERATION      10
```

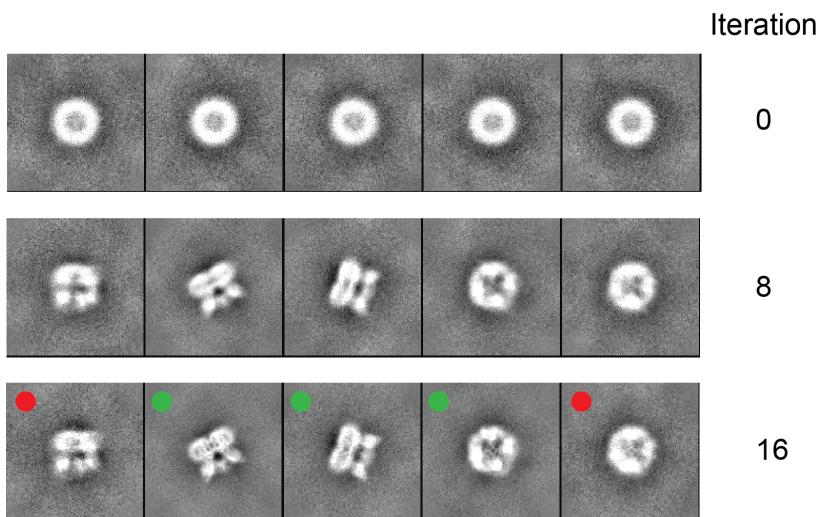


**Figure 6:** Individual TRPV1 particle images.

```
>>>
DISTRIBUTED MODE :: submitting scripts:
distr_simple_script_1
distr_simple_script_2
appending output to nohup.out
appending output to nohup.out
>>> DONE PROCESSING PARAMETERS
**** SIMPLE_MERGE_ALGNDOS NORMAL STOP ****
>>> DONE PROCESSING PARAMETERS
>>> DONE BUILDING GENERAL TOOLBOX
>>> DONE BUILDING HADAMARD PRIME2D TOOLBOX
**** SIMPLE_CAVGASSEMBLE NORMAL STOP ****
>>> DONE PROCESSING PARAMETERS
>>> DONE BUILDING GENERAL TOOLBOX
>>> JOINT      DISTRIBUTION OVERLAP:      0.6757
>>> CLASS      DISTRIBUTION OVERLAP:      0.8030
>>> IN-PLANE DISTRIBUTION OVERLAP:      0.5485
>>> AVERAGE IN-PLANE ANGULAR DISTANCE:   13.8
>>> PERCENTAGE OF SEARCH SPACE SCANNED:  89.6
>>> CORRELATION:                         0.5618
>>> CONVERGED: .NO.
**** SIMPLE_CHECK2D_CONV NORMAL STOP ****
```

The distribution overlap represents the agreement between the alignment and clustering parameters from the present and previous iteration. The most important metric, however, is the fraction of search space scanned. [prime2D](#) is based on stochastic hill climbing optimisation, which relies on the so-called first improvement heuristic. This means that not all references are matched for each particle in each round but only as many as is required to find an improving solution, which is instantly accepted. Therefore, in the beginning of the search only few references need to be evaluated to find improving configurations (percentage of search space is low) whereas when convergence to a local (or global) optimum has been achieved most if not all references need to be evaluated (percentage of search space is high). Convergence is based on the class overlap and the fraction of search space scanned. Upon completion of the calculation, we see

```
>>> CONVERGED: .YES.
**** SIMPLE_CHECK2D_CONV NORMAL STOP ****
>>> DONE PROCESSING PARAMETERS
>>> DONE BUILDING GENERAL TOOLBOX
CLASS:      4 POP:  489
CLASS:      2 POP:  433
CLASS:      3 POP:  301
CLASS:      5 POP:  202
CLASS:      1 POP:  194
**** SIMPLE_RANK_CAVGS NORMAL STOP ****
-----
```



**Figure 7:** TRPV1 class averages from the 0th (random initialisation), 8th, and 16th (final) iteration of simultaneous 2D alignment and clustering with prime2D. Good class averages with clearly discernible projected secondary structure are marked with a green dot and class averages of lower quality are marked with a red dot.

### 3.2 Using SIMPLE in the Wild—Selecting Good Class Averages and Mapping the Selection to the Particles

Most data sets contain images other than perfect particle images (false positives). Sometimes the particles tend to form micro-aggregations and are lying too close to each other or on top of each other. When automatic or semi-automatic particle identification procedures are used, ice-contaminations and carbon edges are often mistaken for particles. Therefore, it is good practice to first cluster the images into 100-200 classes and clean out the unwanted images before proceeding with grouping the data into a larger number of classes and calculating an initial 3D model. You can map the selection done on class averages back to the particle images using the SIMPLE program `map2ptcls`.

```
bash-3.2$ simple_exec prg=map2ptcls
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...
```

**REQUIRED**

```

stk      = particle stack with all images(ptcls.ext)
stk2     = 2nd stack(in map2ptcls/select: selected(cavgs).ext)
stk3     = 3d stack (in map2ptcls/select: (cavgs)2selectfrom.ext)
oritab   = table (text file) of orientations(*.asc/*.txt)

```

**OPTIONAL.**

```
oritab2    = 2nd table (text file) of orientations (*.asc/*.txt)
comlindoc = shc_clustering_ncclsX.txt
doclist   = list of oritabs for different states
deftab    = text file with CTF info (*.txt/*.asc)
outfile   = output document
mul       = origin shift multiplication factor{1}
```

```
nthr      = nr of OpenMP threads{1}
```

If your converged class averages from `prime2D` are `cavgs_iter021.mrc` and your selected ones are `selected.mrc`, then you map the selection back to the particle images with the command:

```
bash-3.2$ simple_exec prg=map2ptcls stk=sumstack.mrc stk2=selected.mrc  
stk3=cavgs_iter21.mrc oritab=prime2Ddoc_021.txt nthr=8
```

creating a file `mapped_ptcls_params.txt` that you can input with flag `oritab` to `simple_distr_exec prg=prime2D` to execute a second pass of clustering using the clean set of particles.

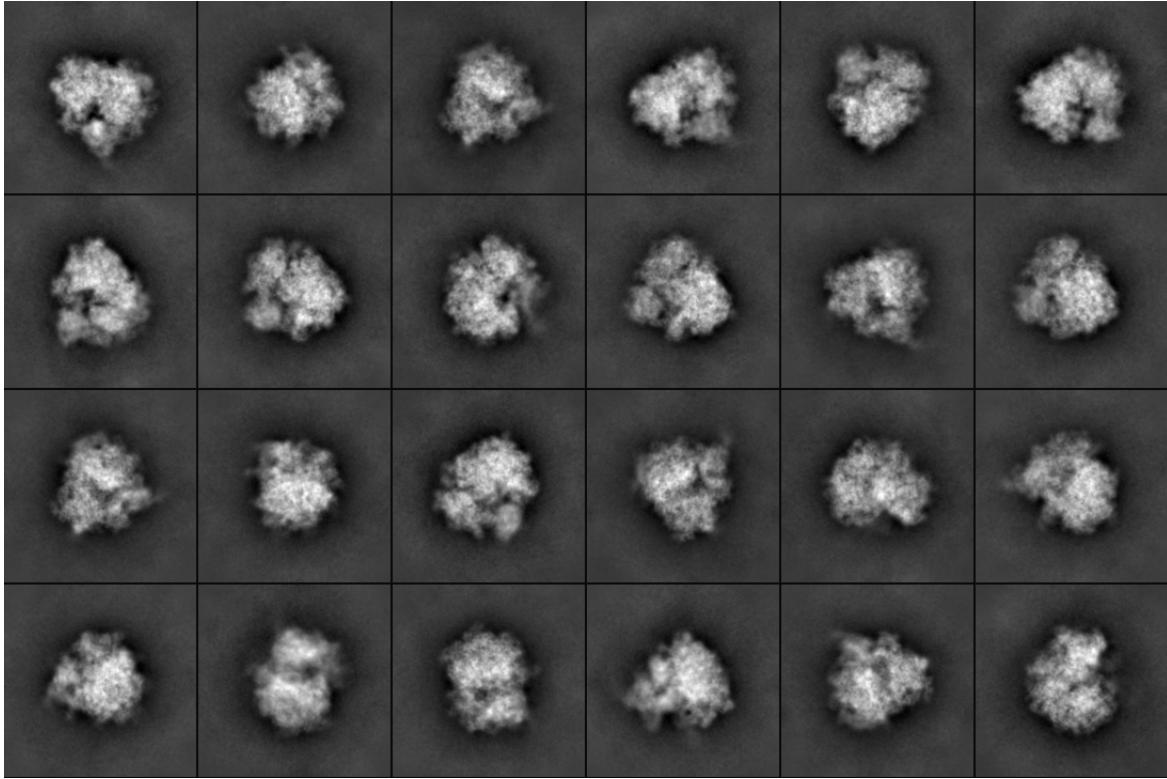
## 4 *Ab initio* 3D Reconstruction from Class Averages Using PRIME3D

A major obstacle to achieving near-atomic resolution with single-particle cryo-EM is the problem of generating an accurate *de novo* 3D reconstruction. Many cryo-EM structures are therefore solved by alignment of the images to *a priori* models. The use of prior information, in the form of either a starting model from an independent source or an assumption of a particular point-group symmetry, is associated with the risk of introducing model bias. The model bias phenomenon is often illustrated by alignment of pure noise images to an image of Einstein. The image of Einstein is almost perfectly reproduced when the aligned noise images are averaged. It is often stated that low-pass filtering of X-ray maps, before they are used as starting models, eliminates model bias. This is a misunderstanding, as any model can be convincingly reproduced from noisy images (Henderson, 2013). Most refinement software, such as FREALIGN (Grigorieff, 2007), RELION (Scheres, 2012), or projection matching (Hohn et al., 2007; Penczek et al., 1994; Tang et al., 2007), depends on an accurate starting model for convergence to a high-resolution map. If the starting model is not supported by the images, there is a potent risk of introducing model bias. To what degree a starting model can bias the final 3D structure needs to be better characterised by methodological studies. We introduced the PRIME3D algorithm (Elmlund et al., 2013) to remove the requirement for *a priori* structural knowledge and open the method to the study of particles with novel structure. Robust algorithms for *ab initio* 3D reconstruction are particularly important for the analysis of small particles with low symmetry.

### 4.1 Analysis of Ribosome Class Averages Using the `ini3D_from_cavgs` Program

SIMPLE was designed primarily for processing single-particle images of molecules with low or no internal symmetry. There are not an awful lot of data sets of asymmetric molecules publicly available, but one of the most popular specimens—one that we all recognise the structure of—is the ribosome. Although the initial idea with PRIME3D was to overcome the need for 2D clustering and generate an accurate 3D map straight from the noisy individual images, we have found that 2D analysis is extremely useful for weeding out bad data and enhancing SNR prior to 3D analysis. Processing class averages also makes 3D model validation easier as we can simply compare re-projections of the 3D map with the signal-enhanced class averages used to obtain the 3D reconstruction. We pre-calculated 2D class averages with PRIME 2D from a publicly available ribosome data set (EMPIAR-10028). These are located in the `3_PRIME3D/data` folder. Please `cd` to this folder and have a look at the class averages with EMAN2. As before, the parameters associated with the class averages are listed in the `info.txt` file.

```
bash-3.2$ cat ./data/info.txt  
smpd=2.68  
msk=80  
pgrp=c1
```



**Figure 8:** Ribosome class averages obtained with PRIME2D.

Input to `ini3D_from_cavgs` is a stack of class averages `stk`, sampling distance `smpd` in Å, mask radius `msk` in pixels, point-group symmetry `pgrp` (`c1` in this case), the number of CPU threads `nthr` and the number of partitions to divide the job into `nparts`. The output from `ini3D_from_cavgs` is a 3D model `rec_final.mrc` and re-projections of this model in the orientations assigned to the class averages `reprojs.mrc`. If we execute `simple_distr_exec prg=ini3D_from_cavgs` in the prompt, we obtain brief instructions for how to run the program:

```
bash-3.2$ simple_distr_exec prg=ini3D_from_cavgs
USAGE:
bash-3.2$ simple_distr_exec prg=simple_program key1=val1 key2=val2 ...
```

#### REQUIRED

```
stk      = particle stack with all images(ptcls.ext)
smpd    = sampling distance, same as EMANs apix(in A)
msk      = mask radius(in pixels)
pgrp    = point-group symmetry(cn|dn|t|o|i)
nparts  = # partitions in distributed execution
```

#### OPTIONAL

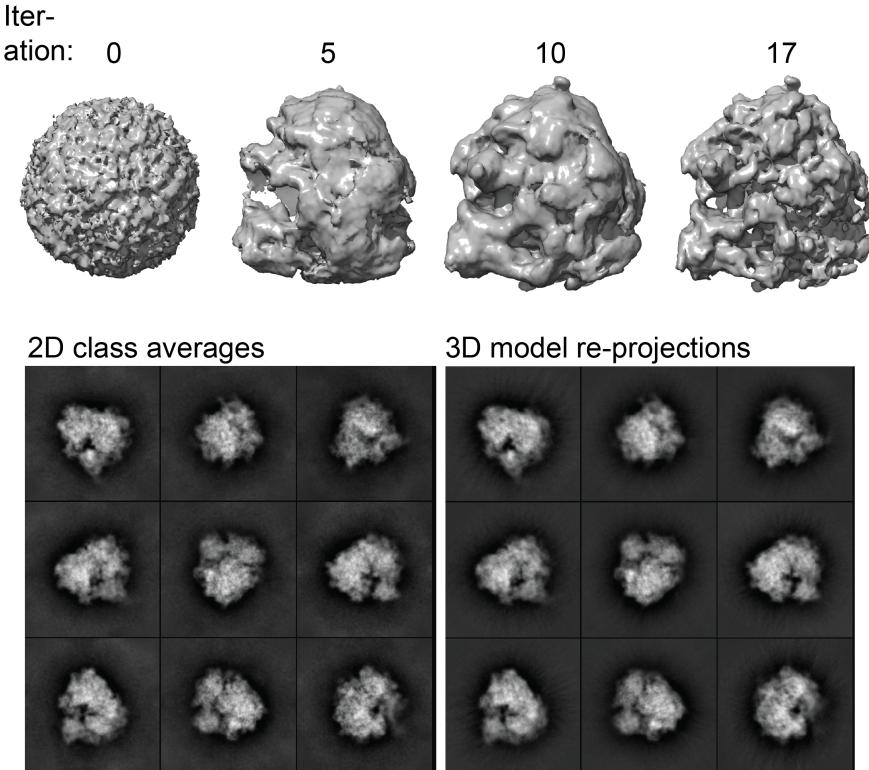
```
nthr     = # OpenMP threads{1}
ncunits  = # computing units, can be < nparts{nparts}
hp       = high-pass limit(in A)
lp       = low-pass limit(in A)
lpstop   = stop low-pass limit(in A){8}
inner    = inner mask radius(in pixels)
width    = falloff of inner mask(in pixels){10}
nspace   = # projection directions
autoscale = automatic down-scaling(yes|no){yes}
pgrp_known = point-group known a priori(yes|no){no}
```

In order to reconstruct an *ab initio* 3D reconstruction, please execute:

```
bash-3.2$ simple_distr_exec prg=ini3D_from_cavgs stk=./data/pfrib80S_cavgs.mrc
smpd=2.68 msk=80 pgrp=c1 nthr=4 nparts=2 nthr_master=8
```

For every iteration, we expect to see an output similar to that of PRIME2D in the terminal:

```
>>>
>>> ITERATION      4
>>>
DISTRIBUTED MODE :: submitting scripts:
distr_simple_script_1
distr_simple_script_2
appending output to nohup.out
appending output to nohup.out
>>> DONE PROCESSING PARAMETERS
**** SIMPLE_MERGE_ALGNDOS NORMAL STOP ****
>>> DONE PROCESSING PARAMETERS
>>> DONE BUILDING GENERAL TOOLBOX
>>> DONE BUILDING RECONSTRUCTION TOOLBOX
**** SIMPLE_VOLASSEMBLE NORMAL STOP ****
>>> DONE PROCESSING PARAMETERS
>>> DONE BUILDING GENERAL TOOLBOX
>>> ANGLE OF FEASIBLE REGION:          14.6
>>> JOINT    DISTRIBUTION OVERLAP:     0.0775
>>> CLASS    DISTRIBUTION OVERLAP:     0.0600
>>> IN-PLANE DISTRIBUTION OVERLAP:    0.0950
>>> AVERAGE ANGULAR DISTANCE BTW ORIS: 63.7
>>> AVERAGE IN-PLANE ANGULAR DISTANCE: 16.8
>>> PERCENTAGE OF SEARCH SPACE SCANNED: 53.9
>>> CORRELATION:                      0.9536
>>> ANGULAR SDEV OF MODEL:            28.06
>>> UPDATE LOW-PASS LIMIT: .NO.
>>> CONVERGED: .NO.
**** SIMPLE_CHECK3D_CONV NORMAL STOP ****
```



**Figure 9:** Progress of the 3D model throughout the stochastic PRIME3D search (top). Class averages vs. re-projections of the converged 3D reconstruction (bottom).

The overlap parameters are very similar to those in `prime2D` but the **CLASS DISTRIBUTION OVERLAP** now corresponds to projection directions rather than 2D classes. New statistics include **AVERAGE ANGULAR DISTANCE BTW ORIS**, which is the average angular distance (in degrees) between the present and the previous sets of best orientations. Since every particle image is assigned a distribution of orientations with associated weight factors, there is also an estimate of the angular standard deviation of the model **ANGULAR SDEV OF MODEL**. For asymmetric molecules, we expect the angular standard deviation to decrease as the procedure approaches convergence. However, if we reconstruct symmetric molecules with `pgrp=c1` or asymmetric molecules with pseudo symmetries, the angular standard deviation may increase with the number of iterations as the orientations become randomly distributed over the symmetry related configurations. Please, use UCSF Chimera to view the reconstructed density. The `*pproc*` output is the post-processed volume (for initial model generation this is simply the volume low-pass filtered with the low-pass limit used for alignment).

## 4.2 Using SIMPLE in the Wild—Mapping Class Orientations and Selection to the Particles

In a real-life scenario, you would have used a carefully selected set of class averages `selected.mrc` to obtain your initial 3D model. In order to map this selection and to compose the 3D orientations assigned to the class averages with the 2D orientations/class parameters obtained with `prime2D` and produce a 3D alignment/selection at the particle level, use `map2ptcls`. The command is similar to that introduced above. If your converged class averages from `prime2D` are `cavgs_iter021.mrc`, your selected ones are `selected.mrc`, and your 3D alignment doc is `prime3Ddoc_030.txt` then you do the mapping with the command:

```
bash-3.2$ simple_exec prg=map2ptcls stk=sumstack.mrc stk2=selected.mrc
stk3=cavgs_iter21.mrc oritab=prime2Ddoc_021.txt oritab2=prime3Ddoc_030.txt nthr=8
creating a file mapped_ptcls_params.txt that you can input with flag oritab to
simple_distr_exec prg=recvol to reconstruct a volume from particle images.
```

## 5 Reconstruction of Symmetrical Structures with SIMPLE

To illustrate how to use the SIMPLE suite, we provide the following workflow including all commands executed when reconstructing the D2 symmetric beta-galactosidase particle. It consists of three steps

1. 2D alignment and clustering of the images
2. Symmetric *ab initio* 3D reconstruction and refinement on class averages in the D2 symmetry point group in one automated step
3. Final resolution estimate from single particles images

We used a dataset of 5,500 256x256 images. After extraction of the images from the micrographs, the image stack is named `sumstack.mrc` and the corresponding CTF parameters (obtained with CTFFIND) are contained in a document called `extract_params.txt`. Both files are located in the `./stack` folder.

### 5.1 In-plane alignment and clustering

We first create a working directory called `2d` where PRIME2D will perform the 2D alignment and clustering of the images.

```
$ mkdir 2d  
$ cd 2d
```

Now, we execute the PRIME2D workflow with a single command line

```
$ nohup simple_distr_exec prg=prime2D stk=../stack/sumstack.mrc  
smpd=1.62 msk=88 ncls=100 ctf=yes deftab=../stack/extract_params.txt  
nparts=2 nthr=4 >& PRIME2DOUT &
```

The `stk` argument points to the location of the images. Throughout the different steps of this workflow we will use a circular mask radius of 88 pixels (`msk=88`) and a sampling distance of 1.62 Angstroms/pixel (`smpd=1.62`). With the `ncls` argument we specify how many cluster centres (or class averages) will be produced. In this case one hundred clusters means we aim at producing 55 images per cluster. Values up to 100-150 are also valid choices but in this case would only produce 30-40 cluster centres. With the argument `ctf=yes` we turn on the Wiener restoration of the cluster centres. Accordingly, we provide the defocus parameters with the `deftab` argument.

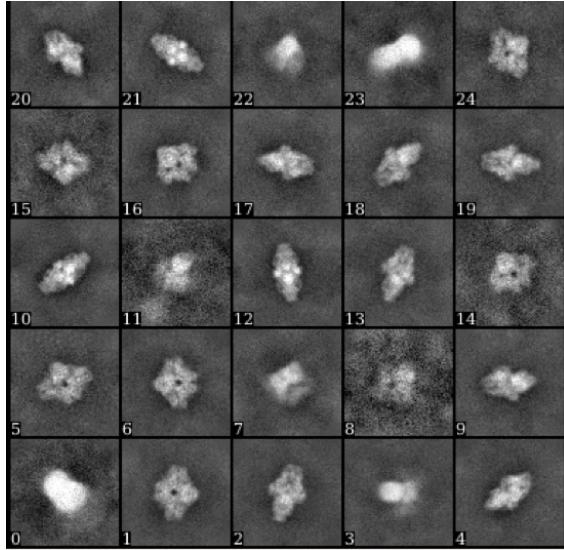
Finally, we specify how to perform the calculations. With `nparts=2` and `nthr=4` all the computations will be split into two batches of 4 threads each. All PRIME2D standard output is written to the PRIME2DOUT log file. At each PRIME2D iteration class averages are produced along with the corresponding document containing in-plane parameters and cluster assignment of all the images, named `cavgs_iterXXX.mrc` and `prime2Ddoc_XXX.txt`, respectively.

Upon completion after 15-20 iterations, PRIME2D produces the classes averages (`cavgs_final.mrc`) and a final document containing in-plane parameters and cluster assignment of all the images (`prime2Ddoc_final.txt`). An additional stack is also generated that contains the class averages ranked by population (`cavgs_final_ranked.mrc`).

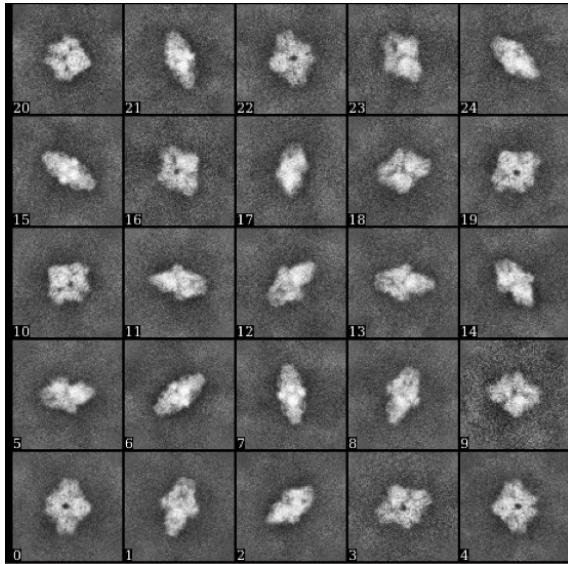
You may remove the temporary files generated by PRIME2D by typing (this should of course never be done while the application is running)

```
$ prime_cleanup.pl
```

Visual examination of the 100 class averages shows numerous images with distinctive, symmetric, features of beta-galactosidase. One can also note blurrier images with less contrast. Typically, these correspond to lowly populated classes where the weaker SNR is likely to contribute little to the subsequent 3D reconstruction. We therefore discard them from the next *ab initio* 3D reconstruction step, together with class averages that correspond to ice and likely inaccurate particle picking. This can be done with your image visualisation software of choice, such as EMAN. The resulting class averages stack ( 70-80 images) is named `cavgs_selected.mrc`



**Figure 10:** cavgs\_final\_ranked.mrc



**Figure 11:** cavgs\_selected.mrc

## 5.2 *Ab initio* 3D reconstruction from class averages

We are now ready to generate an *ab initio* 3D reconstruction based on the selected cluster centres. With the following we create a new working directory for this step

```
$ cd ..
$ mkdir 3d_cavg
$ cd 3d_cavg
```

and execute the automated workflow `ini3D_from_cavgs`

```
$ nohup simple_distr_exec prg=ini3D_from_cavgs stk=../2d/cavgs_selected.mrc
smpd=1.62 msk=88 pgrp=d2 pgrp_known=yes nparts=2 nthr=4 >&INI3DOUT &
```

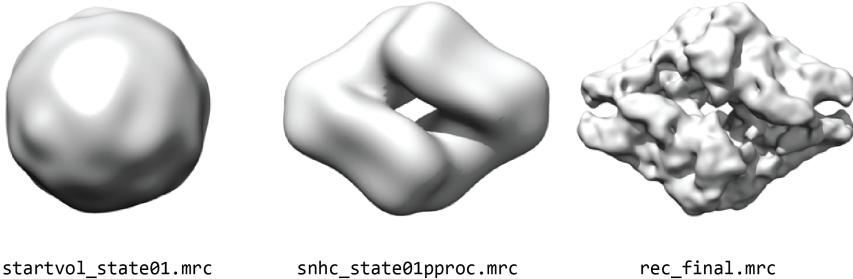
This command line is very similar to the one used to run PRIME2D. The image stack (`stk=?`) is the previously selected class averages. In our 2x4 CPUs workstation this step is completed after a dozen iterations in just 8 to 9 minutes.

This automated workflow uses its own scheme to low-pass filter images and volumes. For more challenging cases, it may be preferable to set the low-pass limit to 20-25 Angstroms; for class averages derived from negative stain images to 30-35 Angstroms. This is achieved by setting the `lp` argument to the desired value (e.g. `lp=25.0`).

It is possible to start with no symmetry assumed, perform a symmetry search in a specific point-group and use it for refinement by simply removing the `pgrp_known=yes` argument. In cases

of low symmetry order (e.g. C2, C4, D2) the `pgrp_known` key performs well but can sometimes fail, especially for small proteins whose shape exhibit pseudo-axes of symmetry. Given that `ini3D_from_cavgs` takes less than 10 minutes to complete, it is a good strategy to restart this step a handful of times to assess the robustness of the volume produced. One can also start from an asymmetric reconstruction and then search for the symmetry as mentioned above.

`ini3D_from_cavgs`, like PRIME2D, generates files at each iteration: documents with the 3D alignments parameters (`prime3Ddoc_XXX.txt`) and the corresponding volumes. The volumes with the `pproc` suffix (standing for *post-processed*) are masked and low-pass filtered to the current resolution and are thus amenable to visual examination. Updates to the resolution are reported in the log file produced (`INI3DOUT`). A final volume (`rec_final.mrc`) is also reconstructed to the original scale of the images/class averages. Temporary files can be deleted as done in step 1. The



**Figure 12:** 3D reconstructions obtained

*ab initio* 3D model generated is indeed in excellent agreement with previous beta-galactosidase maps and crystallographic structures, but the volumes display the alternate known handedness. To mirror one volume to the correct handedness (then named for instance `vol_mirr.mrc`) type

```
$ simple_exec prg=volops smpd=1.62 vol1=recvol_state01_iter012pproc.mrc
outvol=vol_mirr.mrc
```

`ini3D_from_cavgs` also automatically generates re-projections of `rec_final.mrc` (`reprojs.mrc`) in orientations directly comparable to the selected cluster centres obtained in step 1. Excellent agreement between re-projections and cluster centres validates that the final reconstruction is representative of the class averages.

### 5.3 Resolution estimate from single-particle images

To calculate the resolution of the map obtained we calculate the Fourier Shell Correlation (FSC) from even and odd half-maps with the 0.143 resolution threshold. To do so, we need to reconstruct the half-maps from particles. We do this in a separate folder

```
$ cd ..
$ mkdir rec_ptcls
$ cd rec_ptcls
```

First, we need to combine the 2D parameters of the particle images (cluster assignment and in-plane parameters; from step 1) with the 3D parameters derived for the class averages (from step 2):

```
$ simple_exec prg=map2ptcls stk=../stack/sumstack.mrc
stk2=../2d/cavgs_selected.mrc stk3=../2d/cavgs_final.mrc
oritab=../2d/prime2Ddoc_final.txt
oritab3D=../3d_cavg/prime3Ddoc_final.txt nthr=4
```

The arguments used are as follows

- `stk`: original stack of images
- `stk2`: stack of *selected* cluster centres

- **stk3**: stack of cluster centres used for the selection (e.g., to produce **stk2**)
- **oritab**: document produced by PRIME2D corresponding to **stk3**
- **oritab3D**: document produced by **ini3D\_from\_cavgs** corresponding to **stk2**

This generates a document named **mapped\_params\_ptcls.txt** that we can directly use to reconstruct a new map from the particle images and calculate the resolution

```
$ simple_distr_exec prg=recvol eo=yes stk=../stack/sumstack.mrc
oritab=mapped_params_ptcls.txt smpd=1.62 msk=88 ctf=yes
pgrp=d2 nparts=2 nthr=4 >& EOREC
```

This will only takes a few minutes and will print out the FSC values in the EOREC file. The **eo=yes** option specifies that the resolution will be calculated from the FSC between the even/odd halves of the dataset. The final reconstruction is the **recvol\_state01.mrc**. The end of the RESOLUTION file produced gives you the calculated resolution

```
>>> RESOLUTION AT FSC=0.143 DETERMINED TO:      9.87
>>> RESOLUTION AT FSC=0.500 DETERMINED TO:    13.38
```

## References

- Elmlund, H., Elmlund, D., Bengio, S., Aug 2013. Prime: probabilistic initial 3d model generation for single-particle cryo-electron microscopy. *Structure* 21 (8), 1299–306.
- Grigorieff, N., Jan 2007. Frealign: high-resolution refinement of single particle structures. *J Struct Biol* 157 (1), 117–25.
- Henderson, R., 2013. Avoiding the pitfalls of single particle cryo-electron microscopy: Einstein from noise. *Proceedings of the National Academy of Sciences* 110 (45), 18037–18041.
- Hohn, M., Tang, G., Goodyear, G., Baldwin, P. R., Huang, Z., Penczek, P. A., Yang, C., Glaeser, R. M., Adams, P. D., Ludtke, S. J., Jan 2007. Sparx, a new environment for cryo-em image processing. *J Struct Biol* 157 (1), 47–55.
- Penczek, P. A., Grassucci, R. A., Frank, J., 1994. The ribosome at improved resolution: new techniques for merging and orientation refinement in 3d cryo-electron microscopy of biological particles. *Ultramicroscopy* 53 (3), 251–270.
- Reboul, C. F., Bonnet, F., Elmlund, D., Elmlund, H., 2016. A stochastic hill climbing approach for simultaneous 2d alignment and clustering of cryogenic electron microscopy images. *Structure* 24 (6), 988–996.
- Rohou, A., Grigorieff, N., 2015. Ctffind4: Fast and accurate defocus estimation from electron micrographs. *Journal of Structural Biology* 192 (2), 216–221.
- Scheres, S. H. W., Dec 2012. Relion: implementation of a bayesian approach to cryo-em structure determination. *J Struct Biol* 180 (3), 519–30.
- Tang, G., Peng, L., Baldwin, P. R., Mann, D. S., Jiang, W., Rees, I., Ludtke, S. J., Jan 2007. Eman2: an extensible image processing suite for electron microscopy. *J Struct Biol* 157 (1), 38–46.