# ROME 1.1.0 Tutorial

Copyright © 2017 | Intel® Parallel Computing Center for Structural Biology at Dana-Farber Cancer Institute, in association with Peking University.

# 1 Getting started

## 1.1 Recommended reading

Please cite the following article should you use ROME in your study.

- J. Wu, Y.B. Ma, C. Condgon, B. Brett, S. Chen, Q. Ouyang, Y. Mao. Unsupervised single-particle deep classification via statistical manifold learning. arXiv:1604.04539 [physics. Data-an] (2016). Download, Supplementary Data

Please cite the following articles should you use the MAP program in ROME in your study.

- S.H.W. Scheres. A Bayesian view on cryo-EM structure determination. J. Mol. Biol. 415, 406-418 (2012).
- F.J. Sigworth. A maximum-likelihood approach to single-particle image refinement. J. Struct. Biol. 122, 328-339 (1998).

## 1.2 Install MPI

Note that in order to run ROME within a reasonable amount of time, you will need a computing cluster equipped with an MPI (message passing interface) installation. Otherwise, a multi-core desktop machine may suffice for small data sets. Make sure that you have installed Intel compiler, Intel Math Kernel library and Intel MPI library.

## 1.3 Install ROME

ROME is an open-source software. Users can download ROME for free from our website, and install ROME following the installation instructions. Make sure that you have installed Intel C++ compiler, Intel Math Kernel Library and Intel MPI Library. To install the ROME, the first step is loading your Intel compiler, for example:

```
source /apps/mpss/intel_parallel_2015/composer_xe_2015.1.133/bin/compilervars.sh intel64
source /apps/mpss/intel_parallel_2015/impi/5.0.2.044/bin64/mpivars.sh
```

To build ROME, unzip rome1.0a.zip and go into folder first:

```
unzip rome1.1a.zip
cd rome1.1a
```

To build ROME with “mpiicpc” complier, using:

```
make
```

If you bind your intel c++ complier with other MPI complier like “mpicxx”, using:

```
export ROME_CC=mpicxx
make
```

If you want to build ROME with Non-MPI (single node), using:

```
export ROME_CC=icpc
make
```

The executable binary file will be generated in floder “rome1.0a/bin”.

# 2 **Unsupervised 2D classification**

2D classification is an important step to reject bad particles. Deep and quick classification contributes to purification of a homogenous dataset and possibly help discern subtle structural difference directly through class averages. Partitioning different classes with particles into different groups then becomes an efficient way to obtain a “purifier” data. In ROME, we equipped with 3 protocols for reference-free 2D classification applying in different version. The first one is “rome_map2d”, which is used for translation and in-plane angular determination. The basic algorithm of “rome_map2d” is MAP-based (maximum a posteriori or regularized maximum-likelihood) multi-reference 2D refinement. Thus, users can use “rome_map2d” for 2D classification based on MAP alone. The second one is “rome_sml”, which is used for deep classification by SML (statistical manifold learning). The basic algorithm of “rome_sml” is GTM. We suggest users to run “rome_map2d” before running jobs of “rome_sml”. The final one is “rome_deep2d”, which is an integrated, automated procedure used for translation, in-plane angular determination and deep classification. The basic algorithm of “rome_deep2d” is SML-based classification following MAP-based image alignment.

## 2.1 Image I/O

ROME reads the following image file format: MRC stacks (with extension .mrcs) (this is the recommended image format), SPIDER individual images (with extension .spi) and SPIDER stacks (with extension .spi). Preparation of the images is explained on the “Useful_tool” section. ROME writes individual images and image stacks in MRC format. Individual images in stacks are indicated by an integer number (ranging from 1 to the number of images in the stack) followed by an "@" sign and the filename of the stack. Thereby, the first three images in a stack named “test.mrcs” should read as:

```
1@test.mrcs
2@test.mrcs
3@test.mrcs
```

## 2.2 Running the job of “rome_map2d”

General options will be explained when you type “rome_map2d -help” as command in a Linux/Unix terminal. On the I/O tab, set:

Input images STAR file: particles.star
Output rootname: Class2D/map2d
Search range of origin offsets: 10 pixels
Sampling step of origin offsets: 1 pixel
Sampling step of in-plane angle: 5 degree
Number of classes: 30.

If users only want to use MAP-based multi-reference classification, the class number can be set for any value. However, in this case, we suggest that one typically does not use more than 200 classes. More classes specified in the input parameter does not necessilarily allow the rome_map2d progrom to retrieve the expected amount of useful classes. If users want to use MAP to determine translation and in-plane rotation, we suggest that class number should be in the range of 20 to 50. Before running rome_map2d, you need to prepare all “.mrcs” file and one “.star” input file and put them on same dictionary. This is the command to run rome_map2d:

./bin/rome_map2d -i $input_fn -o $output_fn -K $map2d_classes -angpix $pixel_size -iter $map2d_iter -pool $nr_pool -offset_range $offset_range -offset_step $offset_step -psi_step $psi_step

The output file will be a new “*.star” file, which is needed for the next SML step, including each image’s orientation, translation and the class averages mrcs file. More options could be found in “rome_map2d -help”. We suggest that users should run this job parallelly on a cluster.

## 2.3 Running the job of “rome_sml”

General options will be explained when you type “rome_sml –help”as command in a Linux/Unix terminal. On the I/O tab, set:

Input images STAR file: particles.star
Output rootname: Class2D/sml
Number of classes: 1000.

We suggest that if the number of images is relatively small, the class number should be set to no more than 300. Otherwise, a fraction of the classes will be blank. All “.mrcs” files and one “.star” input file should be put in the same directory. The basic command of SML is:

./bin/rome_sml -i $input_fn -o $output_fn -K $sml_classes -angpix $pixel_size -iter $sml_iter

More options can be found in “rome_sml -help”. We suggest that users should run this job parallelly on a cluster.

## 2.4 Running the job of “rome_deep2d”

General options will be explained when you type “rome_deep2d -help”as command in a Linux/Unix terminal. On the I/O tab, set:

Input images STAR file: particles.star
Output rootname: Class2D/deep2d
Number of classes: 300.

The basic command of Deep2D is:

./bin/rome_deep2d -i $input_fn -o $output_fn -map2d_K $map2d_classes -sml_K $deep2d_classes -angpix $pixel_size -map2d_iter $map2d_iter -sml_iter $sml_iter -pool $nr_pool -offset_range $offset_range -offset_step $offset_step -psi_step $psi_step

More options could be found in “rome_deep2d -help”. In default, we set 30 classes and 30 iterations for MAP alignment. The parameter ‘-iter’ in “rome_deep2d” means the number iteration of SML and ‘-K’ means the number of classes by SML. We suggest that users should run this job parallelly on a cluster. After the job is finished, the output file “.star” file from the last iteration can be displayed on the main GUI, ROME2D Viewer. Also, you can display results in each iteration through “_iter.star”. Sorting classes by image distribution will be useful to display the class averages. Thus, you can click on the nice-looking class averages to select the classes with corresponding particles, or you can select all the classes. All good classes can be selected to save as a new star for use in further analysis.

# 3 Supervised 3D classification

3D classification in ROME is based on Bayesian framework which proposed by Sjors Scheres.The first one is “rome_map3d”, which is used for 3D classification. The second one is “rome_reconstruct”, which is used for reconstruct 3D map.

## 3.1 Running the job of “rome_map3d”

General options will be explained when you type “rome_map3d -help”as command in a Linux/Unix terminal. On the I/O tab, set:

Input images STAR file: particles.star
Output rootname: Class3D/run01
Number of classes: 6.

The basic command of MAP3D is:

./bin/rome_map3d -i $input_fn -o $output_fn -K $map3d_classes –ref $reference -particle_diameter $dim -angpix $pixel_size -iter $iter --ini_high $ini_high -offset_range $offset_range -offset_step $offset_step -tau2_fudge 4 -oversampling 1 -healpix_order 2 -sym C1 -random_seed 1 -pool $nr_pool -firstiter_cc -flatten_solvent -zero_mask -norm -scale –ctf

More options could be found in “rome_map3d -help”.

## 3.2 Running the job of “rome_reconstruct”

General options will be explained when you type “rome_reconstruct -help” as command in a Linux/Unix terminal. On the I/O tab, set:

Input images STAR file: particles.star

The basic command of rome_reconstruct is:

./bin/rome_reconstruct -i $input_fn -o $output_fn -angpix $pixel_size

More options could be found in “ rome_reconstruct -help”.

# 4 Useful tools

We recommend that users should create a single working directory for each job, i.e. each structure that you want to determine. Before 2D classification, you first need to prepare the input file and corresponding parameters. ROME has implemented several useful tools such as format conversion, standalone 2D class averaging and low-pass filtering. General options will be explained when you type “rome_tool -help” as command in a Linux/Unix terminal.

## 4.1 Format conversion

ROME uses the STAR-file format to store any type of metadata. In default, stack of images as “mrcs” file is needed for 2D classification in ROME. However, users could use “rome_tool” to convert SPIDER images format into MRC stack format, or vice versa. On the I/O tab, set:

Input images Mrcs file: particles.mrcs (or *.dat)
Output rootname: particles.dat (or *.mrcs)

The basic command for format conversion is:

./bin/rome_tool -convert -i mrcs_file_name -o dat_file_name

or

./bin/rome_tool -convert -i dat_file_name -o mrcs_file_name

Details could be found in “rome_tool -convert -help”.

## 4.2 Class averaging

ROME could do 2D class averaging with CTF correction for each iteration alone. On the I/O tab, set:

Input images STAR file: particles.star
Output rootname: particles.mrcs

The basic command for doing class averaging is:

./bin/rome_tool -classaverage -i star_file_name –o output_file_name -K class_number -angpix pixel_size -averageAlpha (default 0.01) -averageBeta (default 1)

Details could be found in “rome_tool -classaverage -help”.

## 4.3 low-pass filtering

ROME could do low-pass filtering alone. On the I/O tab, set:

Input images Mrcs file: particles.mrcs
Output rootname: particles_filtered.mrcs

The basic command for low-pass filtering is:

./bin/rome_tool -applyfilter -i mrcs_file_name -o output_file_name -filter filter_radius -angpix pixel_size

Here, low-pass filter is a function that truncates the Fourier transform at spatial freaquency. Details could be found in “rome_tool -applyfilter -help”.

## 4.4 Rotate and shift images

ROME could rotate and shift images based on the translations and rotation angle in star file alone. On the I/O tab, set:

Input images STAR file: particles.star
Output rootname: particles_adjusted.mrcs

The basic command for rotating and shifting images is:

./bin/rome_tool -adjust -i star_file_name -o output_file_name

Details could be found in “rome_tool -adjust -help”.

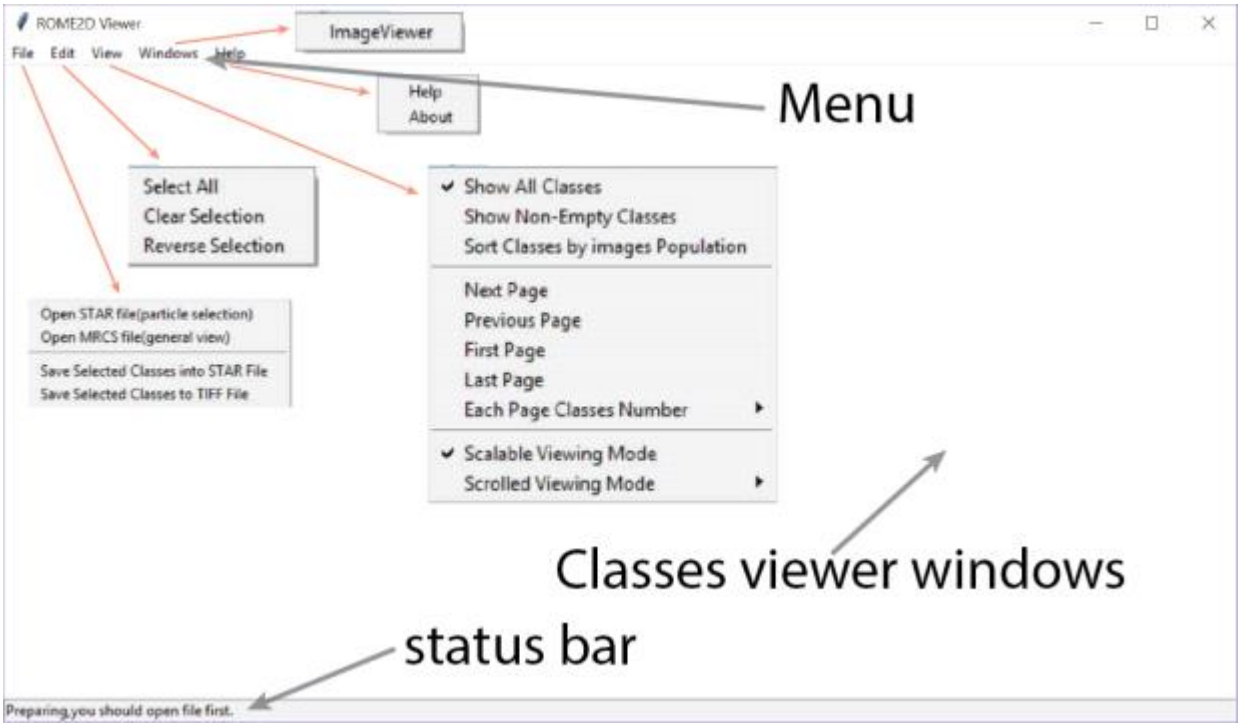
# 5 ROME GUI

The ROME GUI, ROME2D Viewer, is coded by python and is located in the folder “scripts”. This is a portable image and class viewer for MRCS file, which is based on python Tkinter GUI framework. In order to use rome_viewer, you should install python in your computer firstly. And you also need to implement Pillow, numpy and scipy libraries in your python. For Linux/Mac user, one easy way to install these library is using python-pip :

pip install Pillow
pip install numpy
pip install scipy

For Windows user, we suggest that users should use some Windows-friendly python distribution, like Anaconda and Enthought Canopy, which will save a lot of time to install these python libraries. After installing all these libraries, users could type “python rome_dict/script/rome_viewer.py” to open the ROME2D Viewer.

## 5.1 Main class viewer window



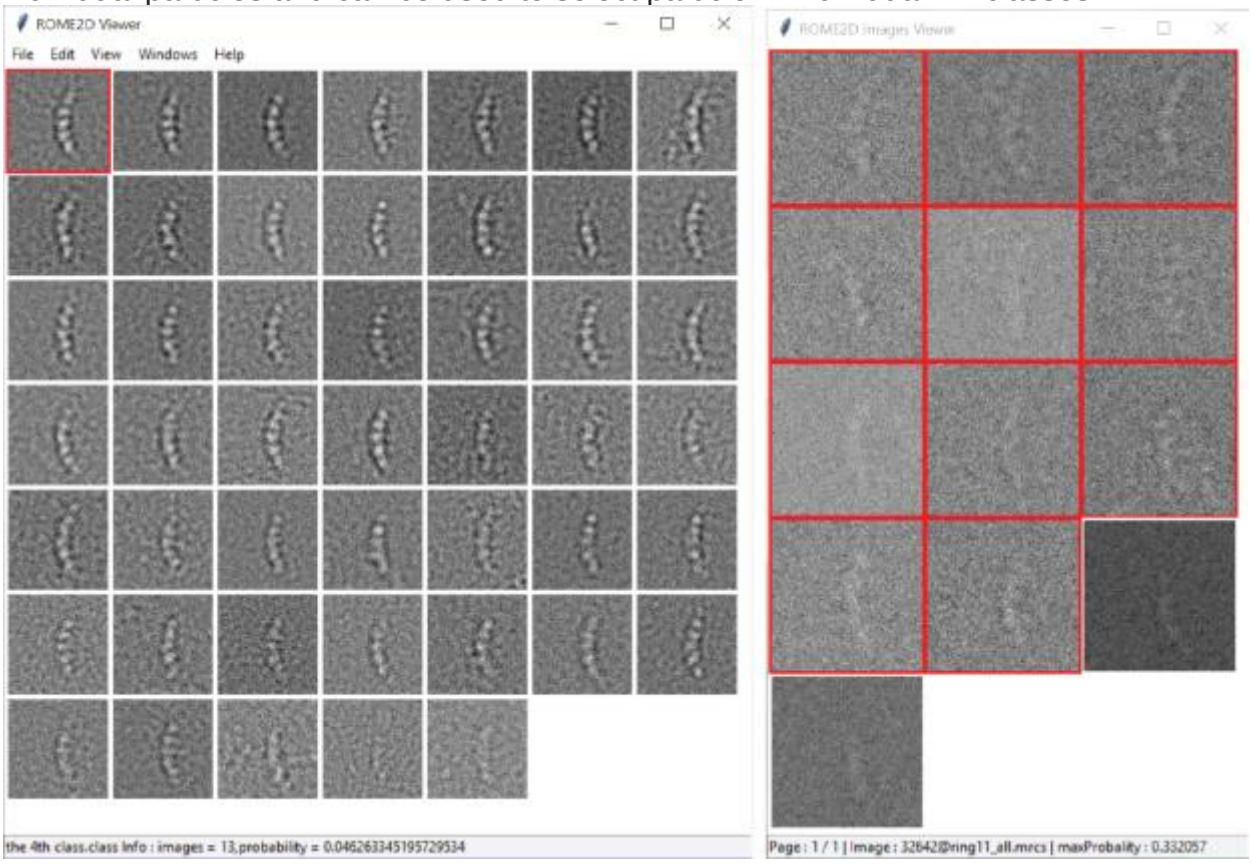

## 5.2 Menu lists

- Open STAR File : open *.star file for particle picking
- Open MRCS File : open *.mrcs file for general view
- Save selected Classes into STAR File : save selected Classes into *.star File
- Save selected Classes to TIFF File : save selected classes to *.tiff file
- Select All : select all classes in current page
- Clear Selection : unselect all classes in current page
- Reverse Selection : reverse select classes in current page
- Show All Classes : show all classes in all pages
- Show Non-Empty Classes : only show non empty classes in all pages
- Sort Classes by Images Population : sort all classes by its images population
- Next Page : goto next page
- Previous Page : goto previous page
- First Page : goto first page
- Last Page : goto last page
- Each Page Classes Number : choice how many classes show in each page(100,300,500,1000)

- Scalable Viewing Mode : classes viewer window is scalable,so you can adjust the windows size
- Scrolled Viewing Mode : classes viewer window is fix,you can set the classes' photo size
- ImageViewer : using to view images belong to each class

## 5.3 Particle picking

In the particle selection mode, users should move STAR and MRCS files output from the working folders such as "rome_map2d", "rome_sml" or "rome_deep2d" and the original input MRCS data in the same directory. Then open the "*.star" file in ROME2D Viewer. In the "Windows" menu, select "Single-Particle Viewer", which will open another GUI window ROME2D Single-Particle Viewer. This window is dedicated to display individual particles and can be used to select particle in individual 2D classes.




# 6 Example for 2D classification

## 6.1 Testing dataset files

We have two small datasets (data1.mrcs and data2.mrcs) to test performance of RELION and ROME. All these two dataset run on a computing cluster with 32 nodes, hereafter ib for each node's name.

data1.mrcs: 28201 particle images, 180x180 in size
data2.mrcs: 24504 particle images, 224x224 in size

The pixel size for both datasets is 1.72 angstram.

## 6.2 Scripts

We have prepared scripts for these two datasets. Reference-free 2D deep classification method based on statistical manifold learning (SML) for data1: runrome_deep2d_data1.sh Reference-free 2D classification based on maximum-likelihood method for data1: runrome_map_data1.sh. SML-based classification for data1: runrome_sml_data1.sh. You only change the name "data1" to "data2" in scripts above to get the scripts for data2.

## 6.3 GUI for Display

Using rome_viewer.py to display the result.

## 6.4 MPI machine file

The machine file is all_machines_ib:

```
$ cat all_machines_ib |head -n 6
$ ib001-ib:1
  ib002-ib:1
  ib003-ib:1
  ib004-ib:1
  ib005-ib:1
  ib006-ib:1
```

Here, "1" means that we allocate one MPI rank to each node.

## 6.5 Testing examples

Reference-free 2D deep classification method based on statistical manifold learning (SML) for data1:

```
$ cat runrome_deep2d_data1.sh
source /apps/mpss/intel_parallel_2015/composer_xe_2015.1.133/bin/compilervars.sh intel64
source /apps/mpss/intel_parallel_2015/impi/5.0.2.044/bin64/mpivars.sh
mkdir ./Class2D/data1_deep2d
input_fn="./data1"
output_fn="./Class2D/data1_deep2d/data1_deep2d_K100"
deep2d_classes=100
map2d_classes=30
offset_range=10
offset_step=2
psi_step=10
map2d_iter=30
sml_iter=30
pixel_size=1.72
nr_pool=20
mpirun -n 32 -f ~/ragon06/yb/apps/all_machines_ib -perhost 1 ./bin/rome_deep2d -i $input_fn -o
$output_fn
-map2d_K $map2d_classes -sml_K $deep2d_classes -angpix $pixel_size -map2d_iter $map2d_iter -
```

```
sml_iter $sml_iter
-pool $nr_pool -offset_range $offset_range -offset_step $offset_step -psi_step $psi_step >
deep2d_K100_data1.log
```

Reference-free 2D classification based on the MAP method for data1:

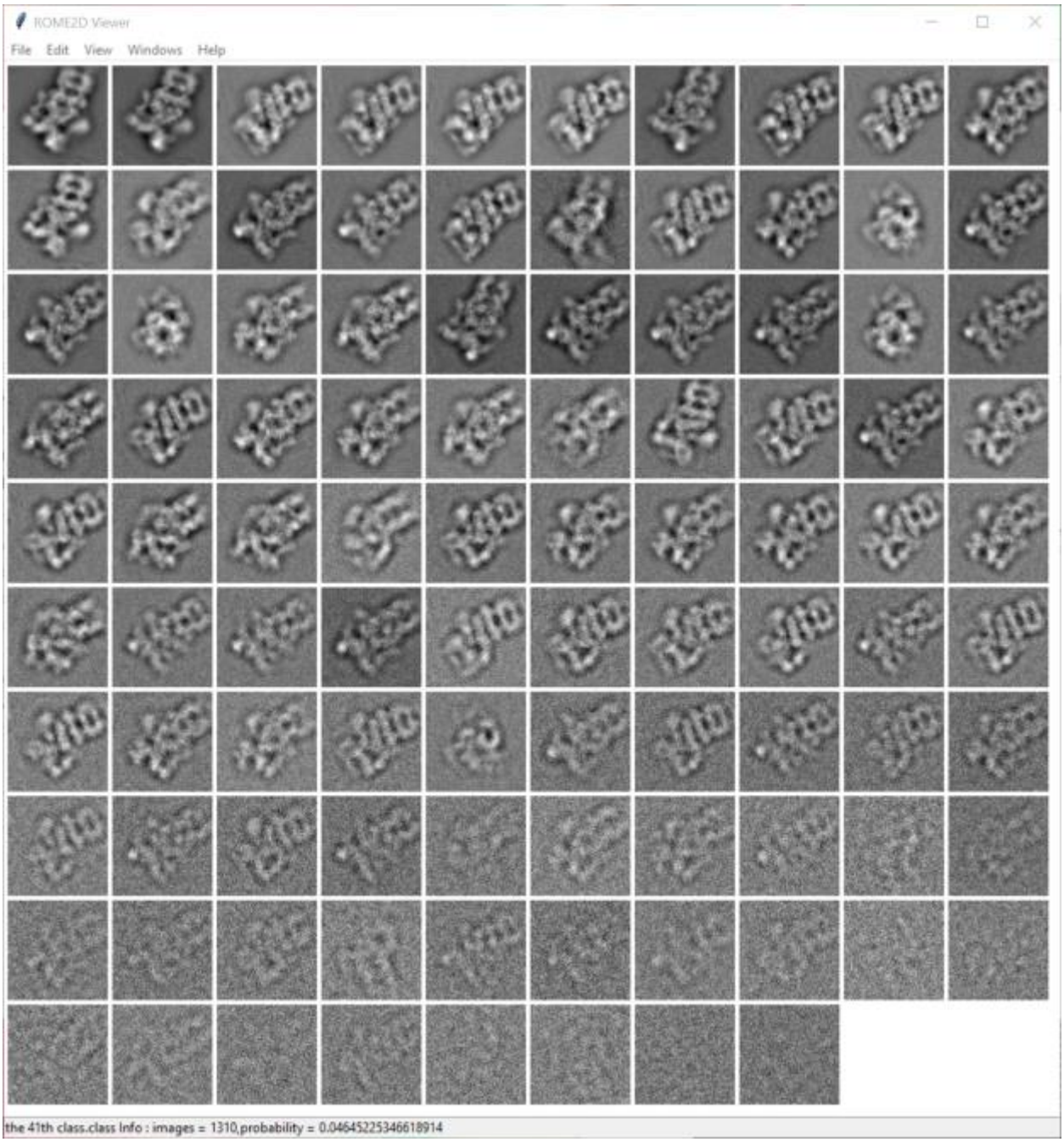
```
$ cat runrome_map2d_data1.sh
source /apps/mpss/intel_parallel_2015/composer_xe_2015.1.133/bin/compilervars.sh intel64
source /apps/mpss/intel_parallel_2015/impi/5.0.2.044/bin64/mpivars.sh
mkdir ./Class2D/data1_map2d_K30
input_fn="./data1"
output_fn="./Class2D/data1_map2d_K30/data1_map2d_K30"
map2d_classes=30
pixel_size=1.72
offset_range=10
offset_step=2
psi_step=10
map2d_iter=30
nr_pool=20
mpirun -n 32 -f ~/ragon06/yb/apps/all_machines -perhost 1  ./bin/rome_map2d -i $input_fn -o $output_fn
-map2d_k $map2d_classes -angpix $pixel_size -map2d_iter $map2d_iter -pool $nr_pool -offset_range
$offset_range
-offset_step $offset_step -psi_step $psi_step > map2d_K30_data1.log
```

SML-based classification for data1:

```
$ cat runrome_sml_data1.sh
source /apps/mpss/intel_parallel_2015/composer_xe_2015.1.133/bin/compilervars.sh intel64
source /apps/mpss/intel_parallel_2015/impi/5.0.2.044/bin64/mpivars.sh
mkdir ./Class2D/data1_sml
input_fn="./data1_map2d_K30"
output_fn="./Class2D/data1_sml/data1_sml_K100"
sml_classes=100
sml_iter=30
pixel_size=1.72
nr_pool=20
mpirun -n 32 -f ~/ragon06/yb/apps/all_machines_ib -perhost 1 ./bin/rome_sml -i $input_fn -o $output_fn
-sml_K $sml_classes -angpix $pixel_size -sml_iter $sml_iter -pool $nr_pool> sml_K100_data1.log
```

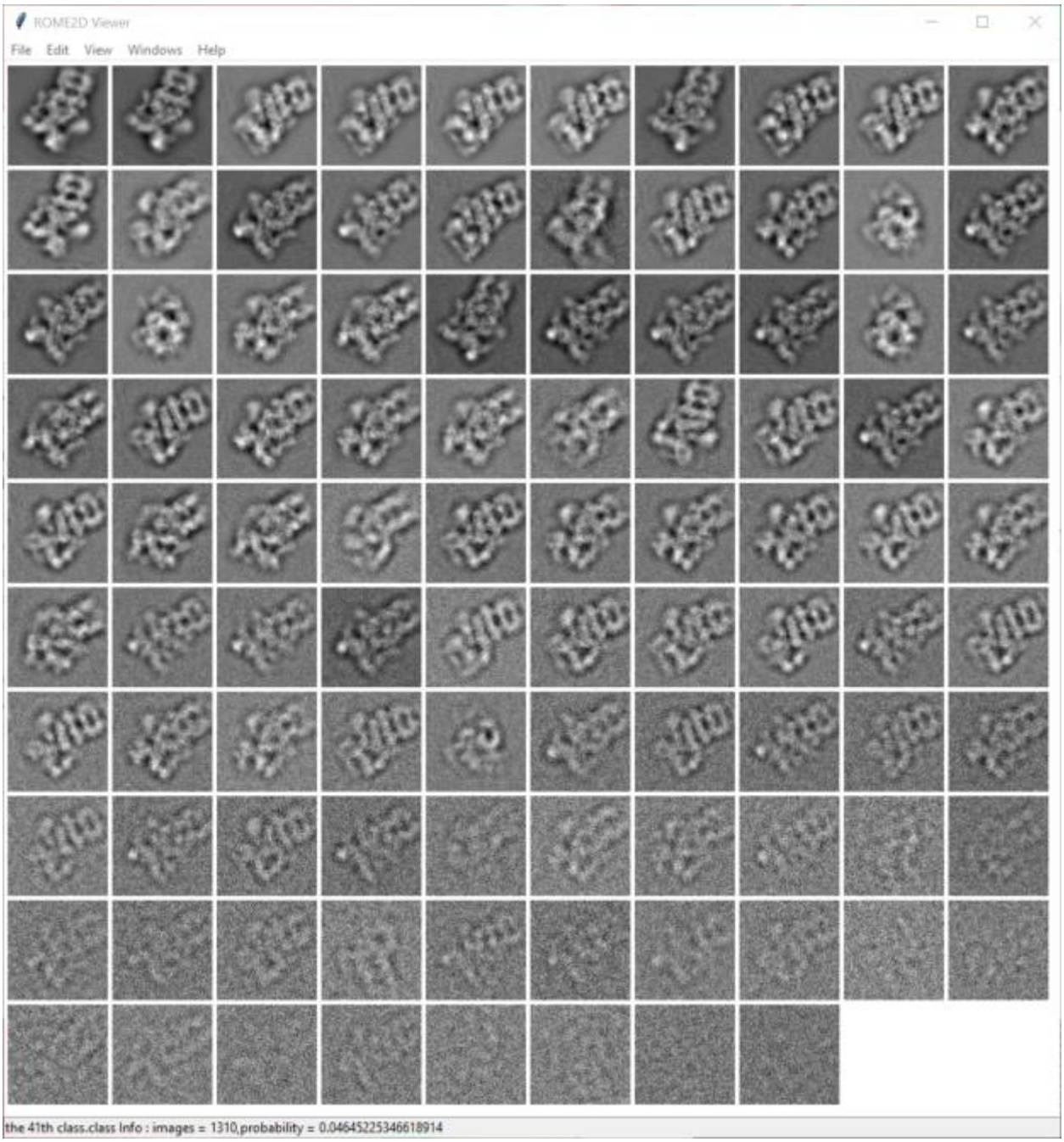
# 6.6 Results

All results were displayed by ROME2D Viewer. Class averages are sorted by the number of images included in each class. For data1, we classify data1 into 100 subsets by "runrome_deep2d_data1.sh". Results should be like this:

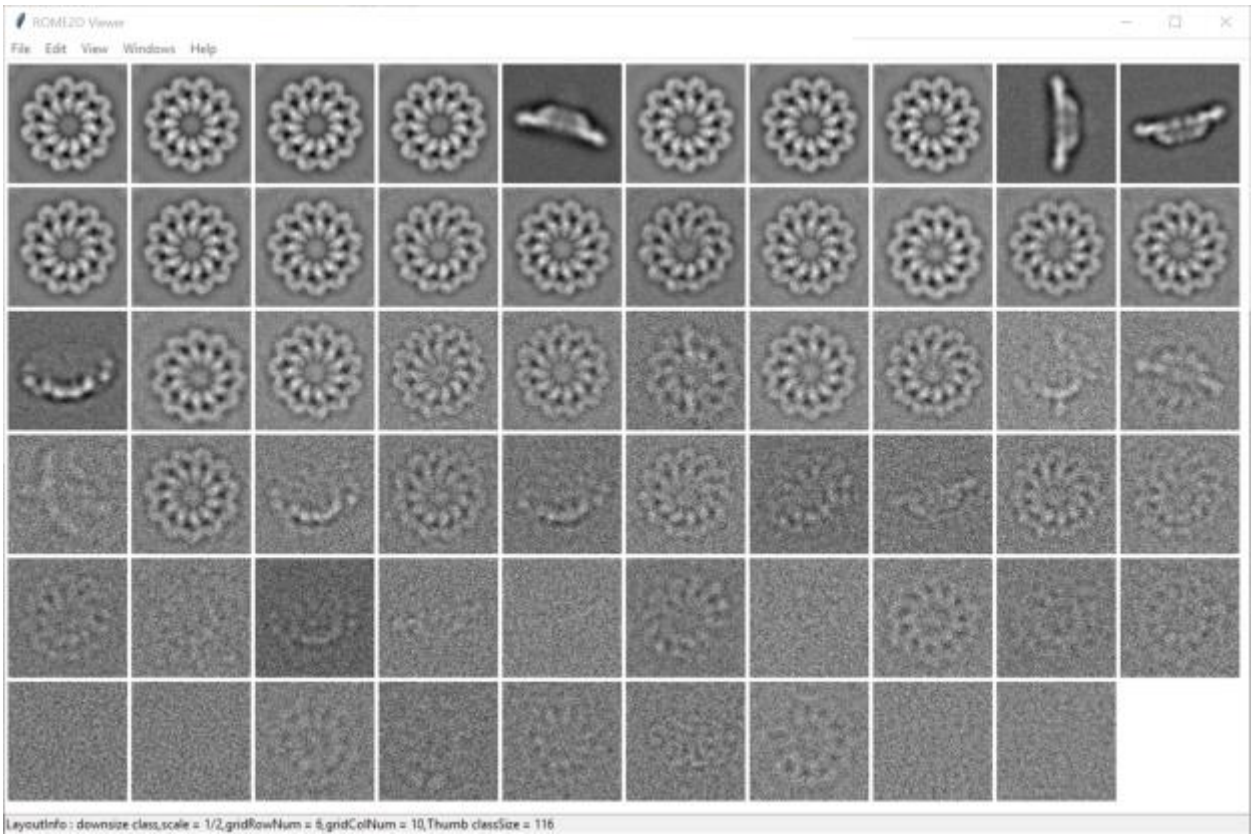



We could first do alignment for data1 by "runrome_map_data1.sh" into 30 classes. Results should be like this:

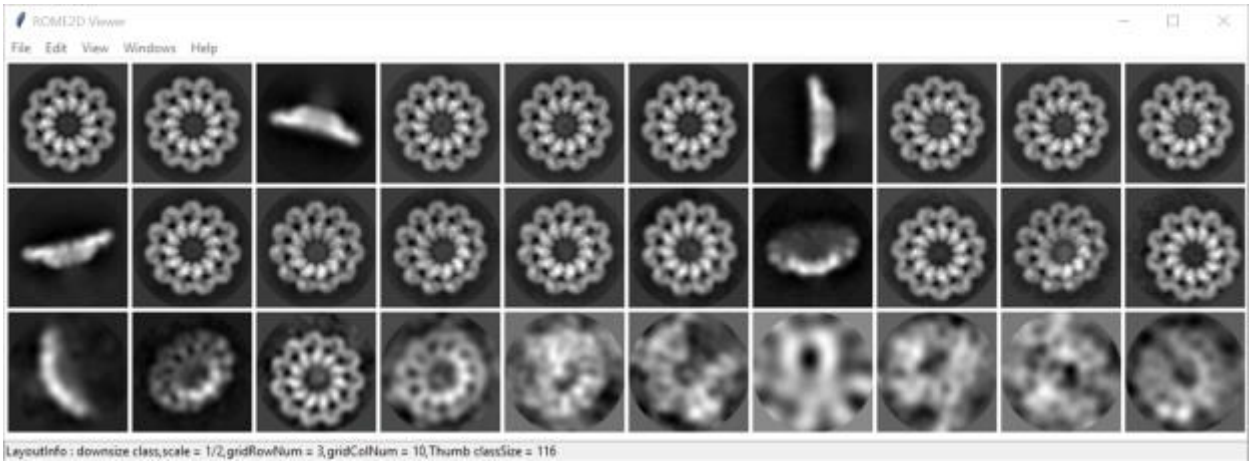

Then we use the results of ML and obtain 100 classes by "runrome_sml_data1.sh". Results are the same as "runrome_deep2d_data1.sh":



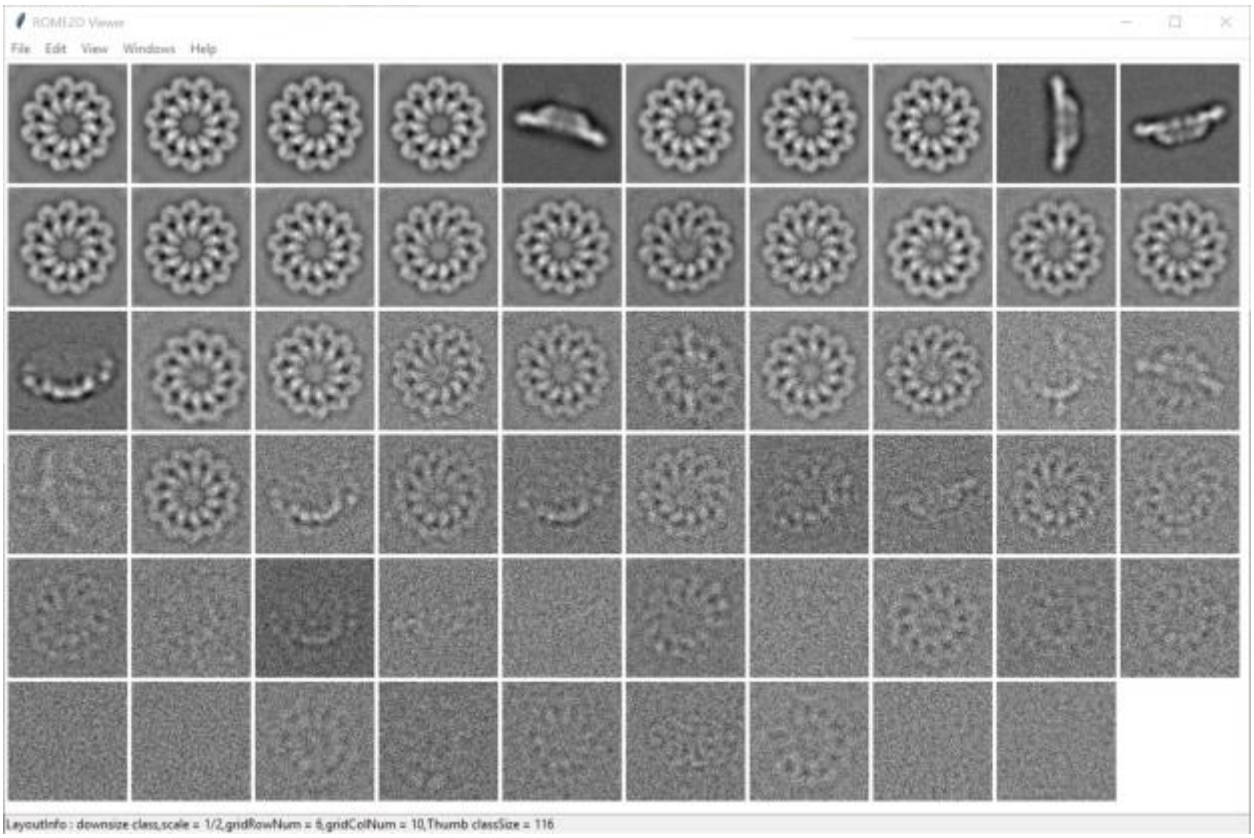

For data2: Results after "runrome_deep2d_data2.sh" into 100 classes should be like this:




Results after "runrome_map_data2.sh" into 30 classes should look like this:




Similar to the results of "rundeep2d_data2.sh", the results after "runrome_sml_data2.sh" classification into 100 classes should look like this:




# 1.7 Example for 3D classification

## 1.7.1 benchmark platform

The CPU information for each node is:

```
$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                72
On-line CPU(s) list:   0-71
Thread(s) per core:    2
Core(s) per socket:    18
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 79
Model name:            Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz
Stepping:              1
CPU MHz:               2300.000
BogoMIPS:              4595.19
Virtualization:        VT-x
```

L1d cache: 32K
L1i cache: 32K
L2 cache: 256K
L3 cache: 46080K
NUMA node0 CPU(s): 0-17,36-53
NUMA node1 CPU(s): 18-35,54-71

We use five nodes to run benchmark dataset.

## 1.7.2 benchmark dataset 1 : ribosomes wi/wo EFG

This is standard classification dataset which is public in the Scheres (2012) JMB paper.it can be downloaded from here ,and the reference map can be downloaded from here.

RELION 1.4 running scripts:

```
mpirun -np 45 -perhost 9 -machinefile nodes relion_refine_mpi --o K4_order2 --i all_images_norm.star --particle_diameter 340 --angpix 2.82 --ref ed_1056.mrc --offset_range 6 --offset_step 1 --ini_high 100 --iter 25 --tau2_fudge 4 --K 4 --oversampling 1 --healpix_order 2 --sym C1 --j 8 --memory_per_thread 2 --random_seed 1 --firstiter_cc --flatten_solvent --zero_mask --norm --scale –ctf
```

Here is the running time for different MPI Rank(-np) and threads(--j):

| MPI_Rank | Threads | Running_time(minutes) |
|---|---|---|
| 10 | 18 | 222.95 |
| 10 | 36 | 166.03 |
| 20 | 9 | 187.03 |
| 20 | 18 | 139.37 |
| 45 | 4 | 160.9 |
| 45 | 8 | 125.68 |
| 90 | 2 | 151.08 |
| 90 | 4 | 120.0 |

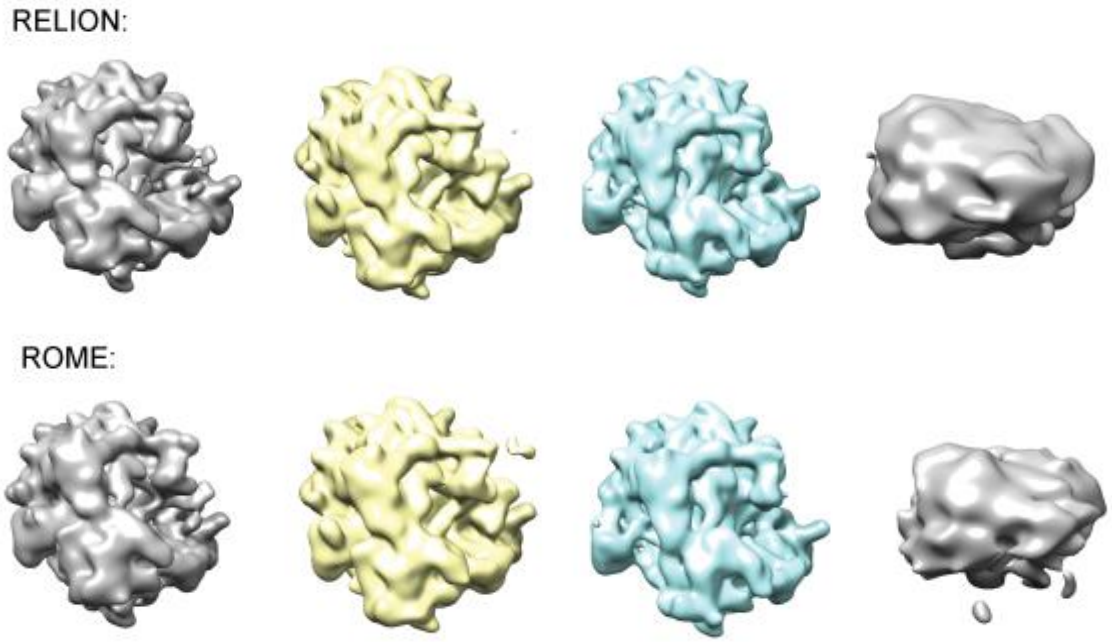
ROME running scripts:

```
export OMP_NUM_THREADS=72
mpirun -n 5 -perhost 1 -machinefile nodes rome_map3d -nr_pool 72 --o K4_order2 --i all_images_norm.star --particle_diameter 340 --angpix 2.82 --ref ed_1056.mrc --offset_range 6 --offset_step 1 --ini_high 100 --iter 25 --tau2_fudge 4 --K 4 --oversampling 1 --healpix_order 2 --sym C1 --random_seed 1 --firstiter_cc --flatten_solvent --zero_mask --norm --scale --ctf
```

Here is the running time for different MPI Rank(-n) and threads:

| MPI_Rank | Threads | Running_time(minutes) |
|---|---|---|
| 10 | 36 | 26.1 |
| 5 | 72 | 21.98 |

Result:




## 1.7.3 benchmark dataset 2 : Plasmodium ribosome

This dataset was public by Wong et al, eLife 2014,it can be downloaded on EMPIAR and EMDB.

RELION 1.4 running scripts:

```
mpirun -np 45 -perhost 9 -machinefile nodes relion_refine_mpi --o K6_order2 --i Particles/shiny_2sets.star --particle_diameter 360 --angpix 1 --ref emd_2660.mrc --offset_range 5 --offset_step 2 --ini_high 60 --iter 25 --tau2_fudge 4 --K 6 --oversampling 1 --healpix_order 2 --sym C1 --j 4 --memory_per_thread 2 --random_seed 0 --firstiter_cc --flatten_solvent --zero_mask --norm --scale --ctf --ctf_corrected_ref
```

Here is the running time for different MPI Rank(-np) and threads(--j):

| MPI_Rank | Threads | Running_time(minutes) |
|---|---|---|
| 10 | 18 | 857.6 |
| 10 | 36 | 644.02 |
| 20 | 9 | 769.15 |
| 20 | 18 | 624.23 |
| 45 | 4 | 835.37 |
| 45 | 8 | 658.73 |
| 90 | 2 | crash |
| 90 | 4 | crash |

ROME running scripts:

```
export OMP_NUM_THREADS=72
mpirun -n 5 -perhost 1 -machinefile nodes rome_map3d -nr_pool 72 --o K6_order2 --i
Particles/shiny_2sets.star --particle_diameter 360 --angpix 1 --ref emd_2660.mrc --offset_range 5 --
offset_step 2 --ini_high 60 --iter 25 --tau2_fudge 4 --K 6 --oversampling 1 --healpix_order 2 --sym C1 --
random_seed 0 --firstiter_cc --flatten_solvent --zero_mask --norm --scale --ctf --ctf_corrected_ref
```

Here is the running time for different MPI Rank(-n) and threads:

| MPI_Rank | Threads | Running_time(minutes) |
|---|---|---|
| 10 | 36 | 180.55 |
| 5 | 72 | 166.2 |

Result: