ROME 1.1.2 Documentation

1 How to install

1.1 Prerequisite

The Intel compiler is required for compiling ROME 1.1.2 code. To download Intel compiler, please visit : https://software.intel.com/en-us/intel-parallel-studio-xe

To obtain free Intel software tools, please visit : https://software.intel.com/en-us/qualify-for-free-software

1.2 Compiling and Installing

Uncompress the tarball containing the sources, and go to the Makefile directory. The ROME execution binary file can be complied by some compliers, such as icpc, mpiicpc, mpicxx.

To compiling with icpc

```
export ROME_CC=icpc
make
```

To compiling with mpicxx

```
export ROME_CC=mpicxx
make
```

To compiling with mpiicpc

```
export ROME_CC=mpiicpc
make
```

To compiling rome_sml with OFFLOAD mode

```
export ROME_OFFLOAD=true
export ROME_CC=mpiicpc
make
```

To compiling rome_map2d and rome_map3d with single precision

```
export ROME_FLOAT=true
```

```
export ROME_CC=mpiicpc
make
```

The execution binary file will be generated on bin folder in current directory. type

```
./bin/rome_deep2d –help
```

to see how to use.

Notice: if you want to rebuild the code, you need to clean first

```
make clean
```

2 List of Programs

2.1 rome_map2d program:

Purpose: Perform 2D image alignment and classification using regularized maximum-likelihood or MAP algorithm.

Usage:

```
./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
```

Control parameters:

```
–i <metadatafile>
   Input metadata file with images needed to align
–o <metadatafile>
   Output metadata
–K <1>
   Number of classes needed to classify
–iter <100>
   Maximum number of iterations to perform
–angpix <1.0>
   Pixel size (in Angstroms)
–pool <50>
   Number of images to be processed together for each EM step
–random_seed <33>
   Number of the random seed generator
–offset_step <2>
   Sampling rate (before oversampling) for origin offsets (in pixels)
–offset_range <10>
   Search range for origin offsets (in pixels)
–psi_step <10>
   Sampling rate (before oversampling) for the in–plane angle
```

2.2 rome_sml program:

Purpose: Perform 2D image classification using statistical manifold learning or GTM algorithm.

Usage:

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

Control parameters:

```
-i <metadatafile>
   Input metadata file with images needed to align
-o <metadatafile>
   Output metadata
-K <1>
   Number of classes needed to classify
-iter <100>
   Maximum number of iterations to perform
-angpix <1.0>
   Pixel size (in Angstroms)
-alpha <0.01>
   Value of the variance of prior model Gaussian distribution
-updatebeta <1>
   Update the variance of noise whether or not, 0 or 1
-precision <10e-12>
   The condition of GTM convergence
-nummic <-1>
   The number of mic cards used to compute
-loadmic <0.3>
   The percentage of job put to compute in mic card
-weightedsum <1.0>
   Probability threshold (0~1) for weighted class averaging
-search <0>
   Whether perform second round of classification based on GTM on each class
or not
```

2.3 rome_deep2d usage:

Purpose: Perform 2D image classification using statistical manifold learning or GTM algorithm.

Usage:

```
./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
```

Control parameters:

```
-i <metadatafile>
   Input metadata file with images needed to align
-o <metadatafile>
```

```
   Output metadata
-ml2d_K <1>
   Number of classes needed to classify based on maximum likelihood method
-sml_K <1>
   Number of classes needed to classify based on GTM
-ml2d_iter <100>
   Maximum number of iterations to perform based on maximum likelihood method
-sml_iter <100>
   Maximum number of iterations to perform based on GTM
-angpix <1.0>
   Pixel size (in Angstroms)
-pool <50>
   Number of images to be processed together for each EM step
-random_seed <33>
   Number of the random seed generator
-offset_step <2>
   Sampling rate (before oversampling) for origin offsets (in pixels)
-offset_range <10>
   Search range for origin offsets (in pixels)
-psi_step <10>
   Sampling rate (before oversampling) for the in-plane angle
```

2.4 rome_map3d usage:

Purpose: Perform 3D image classification using regularized maximum-likelihood or MAP algorithm.

Usage:

```
./bin/rome_map3d -i $input_fn -o $output_fn -particle_diameter $dim -K
$map3d_classes -angpix $pixel_size -K $nr_classes -ini_high $ini_high -iter
$nr_iter -offset_range $offset_range -offset_step $offset_step
-oversampling $oversampling -healpix_order $healpix_order -random_seed
$random_seed -pool $nr_pool -tau2_fudge $tau2_fudge -sym C1 -zero_mask -
flatten_solvent -norm -scale -firstiter_cc -ctf
```

Control parameters:

```
-i <metadatafile>
   Input metadata file with images needed to align
-o <metadatafile>
   Output metadata
-ref <*.mrc>
   3D reference file
-particle_diameter
   Diameter of the circular mask that will be applied to the experimental
images (in Angstroms)
-K <1>
   Number of classes needed to classify
-iter <100>
   Maximum number of iterations to perform
-angpix <1.0>
   Pixel size (in Angstroms)
-ini_high
   Resolution (in Angstroms) to which to limit refinement in the first
```

```
iteration
-pool <50>
   Number of images to be processed together for each EM step
-random_seed <33>
   Number of the random seed generator
-offset_step <2>
   Sampling rate (before oversampling) for origin offsets (in pixels)
-offset_range <10>
   Search range for origin offsets (in pixels)
-oversampling
   Adaptive oversampling order to speed-up calculations (0=no oversampling,
1=2x, 2=4x, etc)
-healpix_order
   Healpix order for the angular sampling (before oversampling) on the (3D)
sphere: hp2=15deg, hp3=7.5deg, etc
-tau2_fudge
   Regularisation parameter (values higher than 1 give more weight to the
data,4 for 3D)
-sym
   Symmetry group
```

2.5 rome_reconstruct usage:

Purpose: Perform full-frequency 3D reconstruction.

Usage:

```
./bin/rome_reconstruct -o $output_fn -i $input_fn --sym C1 --angpix
$pixel_size -ctf
```

Control parameters:

```
-i <metadatafile>
   Input metadata file with images needed to align
-o <*.mrc>
   Name for output reconstruction
-angpix <1.0>
   Pixel size (in Angstroms)
```

2.6 rome_res usage:

Purpose: Calculate local resolution using the same algorithm as implemented in RESMAP.

Usage:

```
./bin/rome_res -res -i1 $mrc_file_1_name -i2 $mrc_file_2_name -minRes 10 -
maxRes 20 -stepRes 1.0
```

Control parameters:

```
-i <metadatafile>
   Input metadata file needed to calculate local resolution
-i1 <metadatafile >
   Input half1 metadata file needed to calculate local resolution
-i2 <metadatafile >
```

```
   Input half2 metadata file needed to calculate local resolution
-minRes <0.0>
   minimal resolution (A)
-maxRes <5.0>
   maximal resolution (A)
-vxSize <0.0>
   Voxel size of input map (A),
-stepRes <1.0>
   step size (A)
-variance <0.0>
   estimate variance
-pValue<0.05>
   P-value for likelihood ratio test
```

2.7 Useful tools

2.7.1 rome_tool -classaverage

Purpose: Compute class averaging from a given file.

Usage:

```
./bin/rome_tool -classaverage -i $star_file_name -o $output_file_name -K
class_number -angpix pixel_size
```

Control parameters:

```
-i <metadatafile>
   Input metadata file with images(*.star)
-o <metadatafile>
   Output metadata
-K <1>
   Number of classes needed to classify
-angpix <1.0>
   Pixel size (in Angstroms)
-averageBeta <1>
   The variance of noise when doing weighted class averaging
-averageAlpha <0.01>
   The variance of prior model Gaussian distribution when doing weighted
class averaging
```

2.7.2 rome_tool -convert

Purpose: Convert image data file from a given format (SPIDER form .dat or RELION form .mrcs) to other formats (SPIDER form .dat or RELION form .mrcs).

Usage:

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

Control parameters:

```
-i <images stack>
   Input file name(*.mrcs or *.dat)
-o <images stack>
   Output file name(*.dat or *.mrcs)
```

2.7.3 rome_tool -select
Purpose: Gather images from plenty of images into one stack.

Usage:
```
./bin/rome_tool -select -i $star_file_name -o $output_file_name
```

Control parameters:
```
-i <metadatafile>
   Input metadata file with images(*.star)
-o <metadatafile>
   Output metadata
```

2.7.4 rome_tool -adjust
Purpose: Shift and rotate images based on translations and rotation angle in star file.

Usage:
```
./bin/rome_tool -adjust -i $star_file_name -o $output_file_name
```

Control parameters:
```
-i <metadatafile>
   Input metadata file with images(*.star)
-o <metadatafile>
   Output metadata
```

2.7.5 rome_tool -applyfilter
Purpose: Perform low-pass filtering.

Usage:
```
./bin/rome_tool -applyfilter -i $mrcs_file_name -o $output_file_name -filter
filter_radius -angpix pixel_size
```

Control parameters:
```
-i <image stack>
   Input file name(*.mrcs)
-o <image stack>
   Output file name(*.mrcs)
-filter
   Filter radius in frequency
```

3 ROME GUI
ROME has a display program (called rome_viewer.py) to display particle images or 2D class averages. It could be launched from the command-line "python rome_viewer.py".

3.1 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 a *.star File
- Save selected classes to TIFF File : save selected classes to *.tiff files
- 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 : go to next page
- Previous Page : go to previous page
- First Page : go to first page
- Last Page : go to 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 belonging to each class

3.2 Particle selection

In particle picking mode, users should put "*.star*" and "*.mrcs*" files that you got from folder "rome_ml2d", "rome_sml" or "rome_deep2d" and the original input mrcs data in the same directory. Then open the "*.star" file in rome_viewer. "ImageViewer" can be needed to select particle classes.

4 Image I/O

ROME reads the following image file formats: MRC stacks (with extension .mrcs) (this is the recommended image format) SPIDER individual images (with extension .spi) SPIDER stacks (with extension .spi). ROME writes individual images and image stacks in MRC format. For SPIDER image format, users can use "rome_tool" to convert SPIDER images or stacks to MRC format stacks. 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. For instance, the first three images in a stack file "test.mrcs" should read as:

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

5 File formats and parameter conventions

We have inherited all the STAR conventions from RELION. ROME uses the STAR (Self-defining Text Archiving and Retrieval) format (Hall, Allen and Brown, 1991) for the storage of label-value pairs for all kinds of input and output metadata. The STAR format

has been adopted by the crystallographic community in the form of CIF (Crystallographic Information Framework), and Bernard Heymann's BSOFT package was the first to use STAR in the field of 3D-EM. Also Xmipp-3.0 now uses the STAR format.

5.1 Metadata label definitions

An example

A STAR file that could be used as input for refinement in ROME that includes CTF information about each particle. A STAR input file should look like this:

```
data_

loop_
_rlnVoltage #1
_rlnDefocusU #2
_rlnDefocusV #3
_rlnDefocusAngle #4
_rlnSphericalAberration #5
_rlnAmplitudeContrast #6
_rlnImageName #7
200.000 38739.3 38739.3 0.00000 2.70000 0.100000 1@data1.mrcs
200.000 38739.3 38739.3 0.00000 2.70000 0.100000 2@data1.mrcs
200.000 38739.3 38739.3 0.00000 2.70000 0.100000 3@data1.mrcs
```

In each iteration of classification, ROME will write out orientations and class number of each particle into a STAR file. All other alignment parameters will be stored in this STAR file.

5.2 Orientations

5.2.1 In-plane rotation angle

The definition of in-plane rotational angle is in compliance with the Heymann, Chagoyen and Belnap (2005). Right-handed rotations are called positive. This is consistent with that is used in RELION.

5.2.2 Image center

When a 2D image is shifted and rotated, the center of this 2D image of dimensions xdim x ydim is defined by ((int)xdim/2, (int)(ydim/2)) (with the first pixel in the upper left being (0,0). Note that for both xdim=ydim=101 and for xdim=ydim=100, the center will be at (50,50). This is the same convention as used in SPIDER and XMIPP. Origin offsets reported for individual images translate the image to its center and are to be applied BEFORE rotations in Fourier Space.

5.2.3 Contrast Transfer Function

CTF parameters are defined as in those used in CTFFIND3, also see the publication by Mindell et al (2003).