ISAC

To make image stack

1. copy "sxrelion2sparx.py" to your working folder

2../sxrelion2sparx.py particles_autopick_thr3.star --output_dir=isac_output --star_section=dEta_--box_size=256 --create_stack

3. convert hdf file to bdb format

sxcpy.py sparx_stack.hdf bdb:test

4. phase-flip all particles in the stack.

sxprocess.py bdb:test bdb:test_flip --phase_flip

5. To reduce image size to 64 x 64

sxprocess.py bdb:test_flip bdb:test_bin --ratio=0.25 --changesize

6. initialize the header information: set the attribute "active" to 1 and the alignment parameters to zero

sxheader.py bdb:test_bin --params=active --one sxheader.py bdb:test_bin --

params=xform.align2d --zero

7. pre-align particles

sxali2d.py bdb:test_bin prealign --ou=22 --xr="2 1" --ts="1 0.5" --maxit=20 --dst=90 --MPI

sxtransform2d.py bdb:test_bin bdb:test_prealign

ou: radius of the alignment area in pixel. For 8x binned, 1 pixel = 5.4 A, ou=22: 119 A
xr: range of translation search in x direction. the Range of 1st and 2st iteration is 2 and 1
ts: step of translation search
maxit: maxiumum number of iterations

8.run ISAC using the "srun_isac.sh" script

It needs several runs. Particles not assigned to the class averages will be used for the next-run classificaiton until no or very few class averages left

The parameters to change:

ou: radius of the alignment area
img_per_grp: max # of images per class (default = 100). This depends on the # of particles in the in)ut file
stab_ali: # of the alignment when checking the stability (default = 5)
thld_err: the threhold of pixel error when checking the stability (default = 0.7), the most important parameter.
n_generation: the # of approach on the dataset.

#!/bin/bash
^ yep, you need a shebang
#4t
specify queue
tt#SBATCH -p normal
#r# run time
#SBATCH -t 202:00:00
number of nodes, this was on stampede, may not need this at your site
#SBATCH -N 1
number of cores
#SBATCH -n 64
error and output files, %J is a handy variable.
#SBATCH -o isac_%J.out
#SBATCH -e isac_%J.err
#ztt Job Name
#SBATCH -J isac
ti=1* email when it is done
#SBATCH --mail-type=end
#SBATCH --mail-user=moldham@rockefeller.edu

mpirun.eman2 -np 64 sxisac.py bdb:test_prealign --radius=22 --img_per_grp=50 --thld_err=1 0 --n_generation=5

9. isac output information

An output directory is generated as: "master2015_12_22 14_07_52"
The directory name include year.month,day and the starting time of the job.

class_averages_candidate_generation_n.hdf: The candidate class averages are stored in

class_averages_generation_n.hdf : class averages generated in this generation

generation_n_accounted.txt : IDs of accounted particles in this generation

generation_n_unaccounted.txt : IDs of unaccounted particles in this generation

To combine the classes from all generation, move them into one directory, then

sxcpy.py class_averages_generation_*.hdf class_averages.hdf

Retrieval of images signed to selected group averages

1 Open in e2display.py file class_averages.hdf located in the main directory.
2 Delete averages whose member particles should not be included in the output.
3 Save the selected subset under a new name,say select1.hdf
4 Retrieve IDs of member particles and store them in a text file ohk.txt:
sxprocess.py --isacselect class_averages.hdf ok.txt
5 Create a vritual stack containng selected particles:
e2bdb.py bdb:data --makevstack:bdb:selectl --list=ohk.txt

The same steps can be performed on files containing candidate class averages.

Note on image size:

Isac_####.out contains information on how many particles are used each generation to make how n- any stable classes. It also stated that the images are changed by a shrink_ratio. To get the ISCA output pixel size, user need to divide the original image size by this number.

ISAC resize the input images as follows:

mpirun.eman2 -np 64 sxisac.py bdb:test_prealign **--radius=radius** --img_per_grp=50 —CTF --thld_err=1.0 **— target_radius=target_radius --target_nx=C** --n_generation=5

shrink_ratio = target_radius/radius (target_radius / radius)
new_image_size = original_image_size x shrink_ratio (in pixel)
if new image size > target_nx: cut image to be target_nx in size
if new image size < target_nx: pad image to be target_nx in size

For example,

input file is 64 x 64 at 4 A/pixel
radius = 25 (which means particle radius 25 x 4 = 100 A)
target_radius = 29 (default, new version should be able to define this by user)
shrink_ratio = 29/25 = 1.16
new_pixel_size = 4 / 1.16 = 3.45 A/pixel
new_image_size = 64 x 1.16 = 74.5 < target_nx (76, default)

output will be: 76 x 76 at 3.45 A/pixel