AlphaCryo4D v0.1.0d Tutorial

To demonstrate the usage of the current version of AlphaCryo4D, the following example of procedure is explained. The inputs of AlphaCryo4D are cryo-EM particles with the parameters of CTF and alignment. The results of AlphaCryo4D are energy landscape and the corresponding 3D classes. In this tutorial, particle images of substrate-translocating 26S proteasome, which the user can replace with his/her own cryo-EM data, are processed to exploit intermediate conformations and to compute the pseudo energy landscape, as described in the following reference:

Zhaolong Wu, Enbo Chen, Shuwen Zhang, Yinping Ma, Congcong Liu, Chang-Cheng Yin, Youdong Mao. Visualizing conformational space of functional biomolecular complexes by deep manifold learning. bioRxiv preprint doi: https://doi.org/10.1101/2021.08.09.455739.

1. Bootstrapping of 3D volumes

First, split the .star file of 2,521,686 proteasome particles into several batches, with each batch having 100,000 particles. Data with low SNR should contains more particles per batch for bootstrapping.

```
cd Bootstrap
python ../../Bootstrap/randsf.py --star initial.star --number 100000
```

For each batch, do a M-fold particle shuffling. Here we apply M = 3 to the star file batch1.star. The smaller the dataset is, the larger value of M should be set.

```
python ../../Bootstrap/bootstrap.py --star batch1.star --fold 3
```

Then for each bootstrap .star file, the bootstrapped 3D volumes via 3D classification are produced in Relion.

```
. fit.sh
```

Then we prepare all bootstrapped 3D volumes for the subsequent process.

```
python ../../Bootstrap/bigdata.py --folder maps_aligned/ --std True
```

Or the preprocessing of low-pass filtering (5 Å is recommended) can be conducted instead of standardization. Three typical 3D bootstrapped density maps of proteasome are shown below. The user can check his/her own bootstrapped 3D volumes in this step.

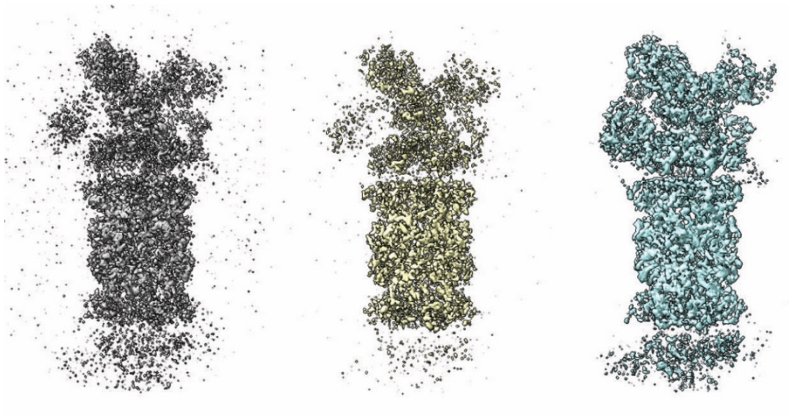

Fig. 1. Typical bootstrapped 3D volumes.

2. Feature extraction by 3D autoencoder

To meet the need of 3D Autoencoder, the dataset of 3D volumes is preprocessed at this step.

```
    cd ../DeepFeature
    python ../../DeepFeature/run_prepare.py --data ../Bootstrap/rdata_3d.npy
```

When training the network, we can define the parameters such as the batch size and GPUs according to our requirement. Note that the batch size has to be an integral multiple of the number of GPUs.

```
    python ../../DeepFeature/run_resnet.py --batchsize 2 --validationsize 2 -r
0.001 --data data_dl.npy --gpu 0,1
```

After training, we apply the 3D CNN to obtain the feature of the dataset of 3D volumes.

```
    python ../../DeepFeature/run_predict.py --data data_dl.npy --batchsize 3
```

3. Energy landscape by manifold learning

To prepare the input data to the energy landscape, the 3D volumes and their feature maps are preprocessed at first.

```
    cd ../ManifoldLandscape
    python ../../ManifoldLandscape/tsne_prepare.py --
data ../Bootstrap/rdata.npy --feature ../DeepFeature/result/feature.npy
```

Then t-SNE is used to plot the low-dimension mapping of 3D volumes with a random seed we set. In many cases, the default values of perplexity and maximum number of iterations in t-SNE work well, but can be empirically tuned to improve the results if desired.

```
    python ../../ManifoldLandscape/tsne_rd.py --input input.npy -s 0 --perplexity 30.0 --niter 1000
```

And the particle number of each 3D volume is calculated.

```
    . enumerate.sh
```

Based on the low-dimensional mapping and the particle number of 3D volumes, we can estimate the energy landscape using the Boltzmann relationship. Moreover, the minimum-energy path (MEP) can be found on the landscape using the String method by defining the starting point, the end point, the step size, etc. Here we use the landscape of all 1,000 volumes to demonstrate the application of the String method.

```
    python ../../ManifoldLandscape/string_method.py --landscape output_big.npy --number num_big.txt --range 160 --start -50 100 --end -80 -30 --stepsize 0.1 --interpolate cubic --kcenters 7
```

In this example, we plot the energy landscape of substrate-bound 26S proteasome. Moreover, an MEP corresponding to substrate translocation initiation is also marked on the landscape (lemon line).

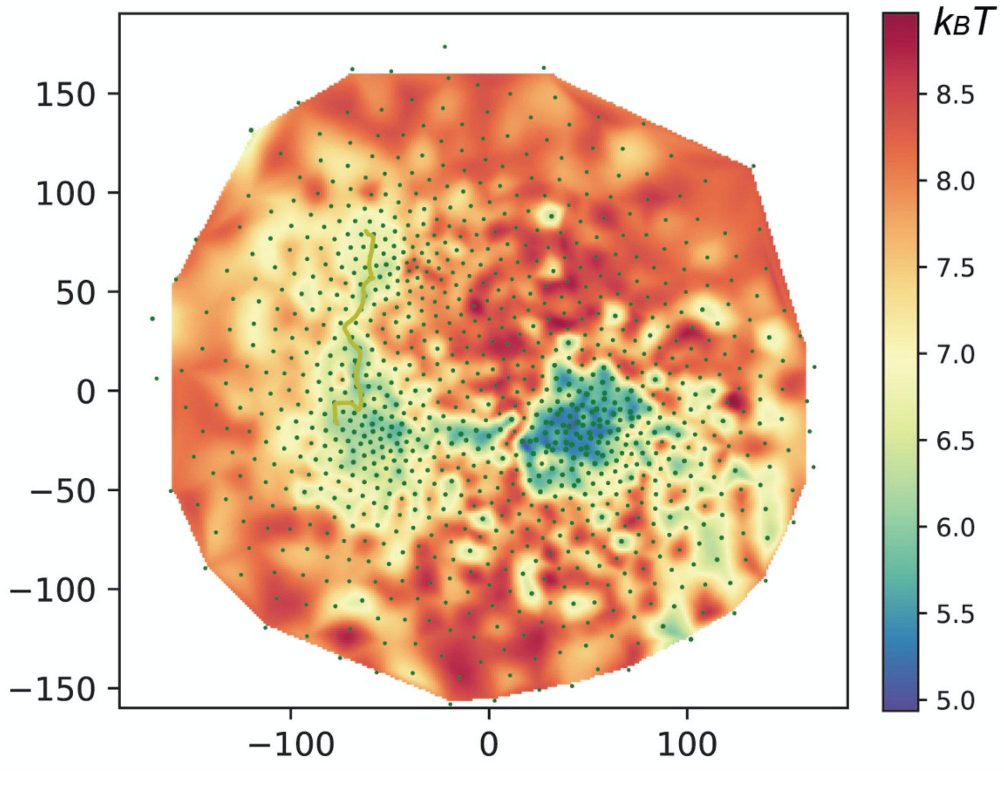

Fig. 2. Energy landscape of 26S proteasome during substrate translocation

4. Particle voting based on energy

At this step, the clusters are prepared according to the centers and radii of the voting boundary on the pseudo energy landscape. Alternatively, we can just define the clusters manually if the landscape is simple enough, skipping minimum-energy path plotting.

```
    cd ../ParticleVoting
    python ../../ParticleVoting/clustering.py --
points ../ManifoldLandscape/output_big.npy --
centers ../ManifoldLandscape/centers_k.npy --radius 30
```

Then for each cluster, the particle star files are linked to their folder respectively. Here also we use a small cluster c1_small for particle voting.

```
    cd c1
    . vote_prepare.sh c1_small
```

Before voting on particles, the header of these particle star files are extracted to head.star.

```
    python ../../../ParticleVoting/gethead.py --star u123_b1_c1.star
```

Lastly, we need to run the voting script to obtain the voted particles for high resolution refinement. Note that the voting threshold (M/2 as default) can be defined in this script, and increasing this threshold may improve the classification accuracy and decrease the particle number of each class. The .star file post_and.star is what we need.

```
    . post_and_f.sh
```