1 Introduction

NEighboring **MO**lecule **TO**pology **C**lustering (**NEMO-TOC**) system is designed to cluster *in situ* neighboring macromolecules spatial arrangements acquired by cryo-Electron Tomography (cryo-ET). After that, we can learn how neighboring macromolecules are arranged spatially *in situ*. Besides some additional functions like linking neighboring macromolecules into long chains and undetected particles automatically interpolation are added.

2 System requirements

Linux: centos 7 (only test)
Python packages

- numpy: 1.16.1(1.18.5/1.19.2)
- pandas: 1.0.5(1.1.1/1.2.1)
- matplotlib: 3.3.3
- cupy: 9.0.0(for GPU calculation)
- cudatoolkit: 11.0.3(for GPU calculation)
- pytest: 6.2.4
- networkx:2.6.2
- seaborn:0.11.2
- dill:0.3.4

We tested GPU acceleration in GeForce RTX 3090 with CUDA version 11.1.

3 Installation

In the command line window, input following command,
git clone https://github.com/GuoQLabPKU/polysomeTracking.git. After that the source codes are downloaded. For current version, you should do the following analysis where codes are downloaded.

4 Testing

For test, an example star file with 11,547 particles(ribosomes) from 18 collected tomograms are available in the folder named ***data***. For each particle, the Euler angle and the coordinate are included, which was acquired by the custom method like template matching and further refinement in ***Relion***. The parameters which are needed modified for following analysis are stored in the script ***main.py***. Should edit the main.py.

4.1 basic parameters
input_star = ‘data/ all_particles_neuron_warp.star’ (the pathway pointing to the star file to be analysed)
run_time = ‘run0’ (this will create a folder named ‘run0’)

pixelSize = 3.42 (unit angstrom)
cluster_threshold = 20 (a threshold that used to cut the linkage during the step of hierarchical clustering. 20~35 usually works well) ***Need multiple tunes*** for better clustering results.
relinkWithOutSmallClasses = 1 (if remove non-meaningful clusters with small number of transformations. Set to 0 if want to keep all the clusters)
minNumTransformPairs = -1 (if the parameter **relinkWithOutSmallClasses** is set to 0, this parameter is useless. Or else, this parameter set the number of transformations to filter non-meaningful clusters, that is to say, if one cluster has transformations smaller than this number, then this cluster will be removed for further analysis. -1 is the default parameter corresponding to 0.05% fraction of the total number of transformations detected)
remove_branch = 0 (if remove the branches of detected transformation chains based on the assumption that the theoretical transformation chain is linear without any branch)
average_particles = 1 (if average the particles of each cluster). ***For more detailed parameters of averaging, see the parameters section of advanced parameters.***

4.2 parallel computation parameters
cpuNr = 15 (the number of cores to parallelly run the script)
gpuList = None (If no GPU is available, just set this parameter to None. Or else pass the list of available GPUs, like [0], [0,1])
avg_cpuNr = 15 (the number of cores to run the particle averaging step)

4.3 visual parameters
vectorfield_potting = 'basic' (this parameter is used to present the polysomes information in the rendered vector field figures. Setting this parameter to 'advance' will present the detailed information of polysomes like the cluster labels belonging to). *Highly recommend to set as **basic** for clear visualization.*
show_longestPoly = 1 (this parameter is used to show the longest polysome in the rendered figures. If set to 1, of each cluster, the longest polysome will be rendered into one figure. Set to 0 to switch off such function)
longestPoly_ClassNr = np.array([-1]) (if the parameter **show_longestPoly** is set to 1, then only the longest polysome will be rendered in these clusters)
if_vispoly = 0 (if set this parameter bigger than 0 like 10, then only the polysomes with length bigger than 10 will be rendered for display. Setting this parameter to 0 is default to switch off display)

4.4 advanced parameters
maxDist = 3.52*100 (in angstrom unit. The search radius for adjacent macromolecules in the tomgram)
link_depth = 2 (the searching depth for linking adjacent transformations into long chains. The higher value this parameter is, the longer time it will consume.

2 usually works well)
fillUpPoly_classNr = np.array([-1]) (the clusters for filling up gaps. -1:all clusters)
fillUpPoly_addNum = 0 (the number of particles added at the tail of each polysome for filling up steps. 0:switch off the filling up step)
fillUpPoly_model = 'longnorm' (the theoretical distribution model to judge if accept interpolated particles during the filling up step. **genFit**: the distribution generated based on the data; **lognorm**: the distribution generated based on the lognorm model, **max**:no specific model distribution fitted. It is suitable if the clusters detected have few transformations)
fillUpPoly_threshold = 0.05 (the threshold to accept interpolated particles during the filling up step. The threshold ranges from 0 to 1. The higher value is, the more confident is)
avg_pixS = 3.42 (the pixel size of particles for ***Relion*** alignment)
avg_minPart = 50 (the minimal number of particles requirement for average. If one cluster contains particles less than this threshold, then the average step will be ignored)
avg_maxRes = 50 (the parameter of ***Relion***. The highest resolution for reconstruction)
avg_callByPython = 0 (If use python to call the average function of ***Relion***. If set to 1, the ***Relion*** average function should be available in the environment. If set to 0, the executable averaging scripts are located at the relative pathway 'run0/avg/exp/run0'. Can directly execute these scripts to get the averaging reconstructions)

After modification of these parameters, you can directly run the script by ***python main.py***. For the test data, it usually consumes ~30min. If you find the speed is slow, you can use GPU or increase the number of cores.

5 Result analysis

After ***python main.py***, it will automatically create several folders to store results. The parent folder will be created according to the ***run_time*** parameter set in the main.py.

5.1 The file ***allTransformFillUp.star*** stores all the information of transformations. Each row represents one transformation. The detail information for each column is listed below.

pairIDX1	the index of the particle 1 in the input particle star file
pairIDX2	the index of the particle 2 in the input particle star file
pairTomoID	the tomogram ID that the transformation belongs to
pairTransVectX	transformation vector X (pixel)
pairTransVectY	transformation vector Y (pixel)
pairTransVectZ	transformation vector Z (pixel)
pairTransAngleZXZPhi	transformation angle phi (degree)
pairTransAngleZXZPsi	transformation angle psi (degree)
pairTransAngleZXZTheta	transformation angle theta (degree)
pairInvTransVectX	invert transformation vector X (pixel)
pairInvTransVectY	invert transformation vector Y (pixel)
pairInvTransVectZ	invert transformation vector Z (pixel)
pairInvTransAngleZXZPhi	invert transformation angle phi (degree)
pairInvTransAngleZXZPsi	invert transformation angle psi (degree)
pairInvTransAngleZXZTheta	invert transformation angle theta (degree)
pairLenTrans	transformation vetor length (pixel)
pairAngDist	transformation angle distance compared with the average transformation
pairCoordinateX1	the coordinate X of the particle 1 (pixel)
pairCoordinateY1	the coordinate Y of the particle 1 (pixel)
pairCoordinateZ1	the coordinate Z of the particle 1 (pixel)
pairAnglePhi1	the Euler angle of phi of the particle 1 (degree)
pairAnglePsi1	the Euler angle of psi of the particle 1 (degree)
pairAngleTheta1	the Euler angle of theta of the particle 1 (degree)
pairClass1	the conformation class of the particle 1
pairPsf1	the CTF image of the paritcle 1
pairPosInPoly1	the position of this transformation in one polysome that it belongs to
pairCoordinateX2	the coordinate X of the particle 2 (pixel)
pairCoordinateY2	the coordinate Y of the particle 2 (pixel)
pairCoordinateZ2	the coordinate Z of the particle 2 (pixel)
pairAnglePhi2	the Euler angle of phi of the particle 2 (degree)
pairAnglePsi2	the Euler angle of psi of the particle 2 (degree)
pairAngleTheta2	the Euler angle of theta of the particle 2 (degree)
pairClass2	the conformation class of the particle 2
pairPsf2	the CTF image of the paritcle 2
pairTomoName	the name of the tomogram that the transformation belongs to
pairPixelSizeAng	the pixel size
pairOriPartList	the input particle star file
pairMaxDist	the distance threshold for neighboring particles
pairClass	the cluster number the transformation belongs to
pairClassColour	the color for this cluster for figure rendering
pairLabel	the polysome ID the transformation belongs to
fillUpProb	1:interpolated particles /-1: particles from the input particle star file

5.2 In the subfolder named ***classes***, the particles information belonging to each cluster is located at the subfolder named ***particleCenter.*** Besides, the center coordinate of one pair of particles belonging to each cluster is located at the subfolder named ***pairCenter***. And the transformation information is stored at the file named ***transList.star***. The typical files created in ***classes*** folder is listed below.

```
├── c1
│   ├── pairCenter
│   │   ├── pairCenter_101.coords
│   │   ├── pairCenter_101.coords.angles.zyz
│   │   ├── pairCenter_102.coords
│   │   ├── pairCenter_102.coords.angles.zyz
│   ├── particleCenter
│   │   ├── allParticles.star
│   └── transList.star
```

5.3 In the subfolder named ***scores***, the file named ***tree.npy*** stores the transformation distance between any pair of transformations. This file can be used for following clustering.

5.4 In the subfolder named ***stat,*** from the file named ***statPerClass.star,*** you can read the statistical information of each cluster. The basic information in this file is listed below,

stdTransVect: standard deviation of translocation vector compared with the average
stdTransAng: standard deviation of rotation vector compared with the average
meanTransVectX: average translocation vector X
meanTransVectY: average translocation vector Y
meanTransVectZ: average translocation vector Z
meanTransAngPhi: average rotation angle phi
meanTransAngPsi: average rotation angle psi
meanTransAngTheta: average rotation angle theta

From the file named ***statPerPoly.star***, you can read the detail information of each polysome. The basic information in this file is listed below,

num: the number of particles for each polysome detected
tomoNr: the tomo ID of each polysome detected
classNr: the cluster number of each polysome detected
polyID: the ID number of each polysome
hasBranch: if the polysome has any branch(0: without any branch, 1:with at least one branch)
confClassVect: the conformation states of each particle in each polysome
posInListVect: the index of each particle of each polysome

Besides, the number of particles which are not included in the final detected cluster is stored in the file named ***statDropRibos.star***. The number of overlapped particles between any pair of clusters is stored in the file named ***statOverlapRibos.star***.

5.5 In the subfolder named ***vis***, the figures are rendered for better visualization.

5.5.1 In the folder named ***clustering***, one figure named ***linkLevel.png*** like below shows the distribution of calculated transformation distances.

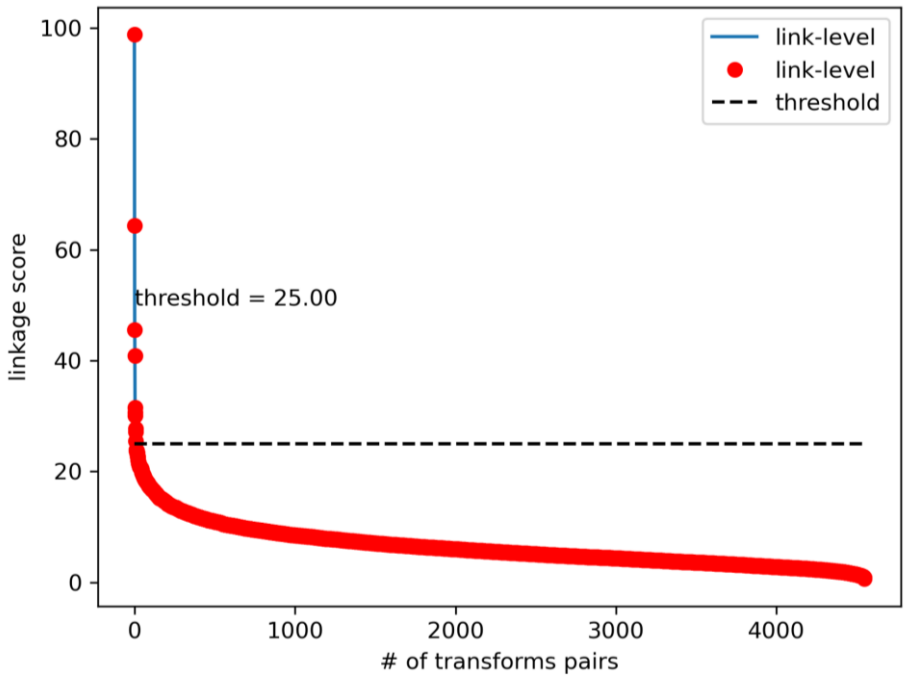

Figure 1. The distribution of transformations distances. The threshold labeled is the threshold used to cut the linkage.

Besides, the figure named ***tree.png*** shows the linkage results of detected clusters like below,

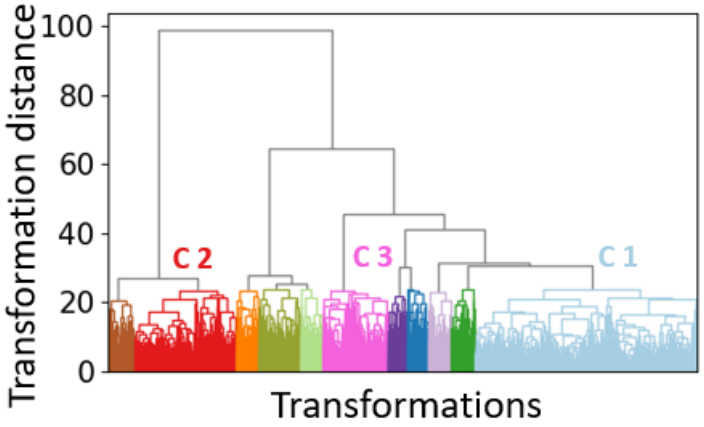

Figure 2. The hierarchical clustering results of transformations.

5.5.2 In the subfolder named ***fitDistanceDist***, each figure represents the distribution of transformation distances (vector distances, angle distances and the combined distances) of each cluster.

5.5.3 In the subfolder named ***noiseEstimate***, of each cluster, the transformation distances from the same cluster and from the different clusters are shown as two distributions like below. The green color presents transformation distances from the different clusters, and blue color from the same cluster. We used the gauss kernel to fit the distribution (colored in orange and in red)

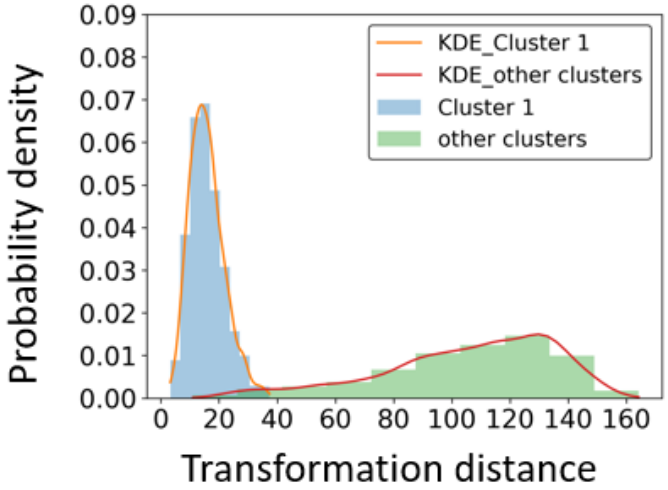

Figure 3. The distribution of transformation distances from the same cluster (in blue) and different clusters (in green).

Tips: The figures in this subfolder is very useful to assess the purity of clustering. Like the figure posted here, the transformation distances from different clusters are bigger than from the same cluster, which means the higher similarity of transformations from the same cluster.

5.5.4 In the subfolder of ***vectfield***. The ***tomoID.png*** contain all the polysomes in that tomogram. Like below, one line represents two particles, and the color represents the cluster these two particles belonging to. It is very easy to find the long helical polysome with the black arrow pointing to.

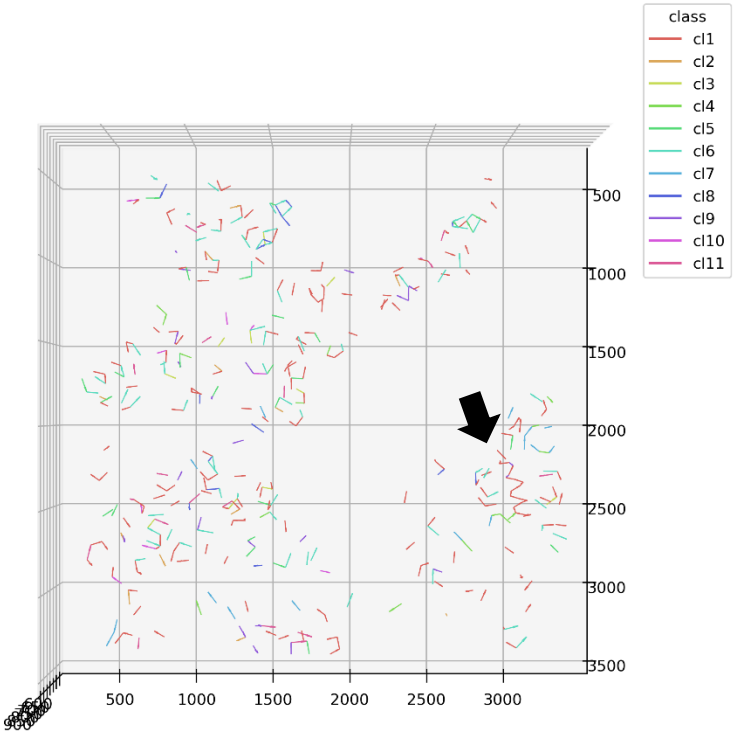

Figure 4. The overall presentation of all the transformation chains(polysomes). The longest helical polysome is indicated with the black arrow.

Besides, of each cluster, the longest polysome are rendered named ***longestPoly.png*** and the polysome length distribution are shown in the figure named ***polyLengthDist.png.***

5.5 Another important file generated is ***particlesFillUp.star***, of which you can find the interpolated particles (labeled as 1 in the column of ***fillUpProb***). Then you can extract the information of interpolated particles for manually inspection in the tomogram.