01.02.2022 by Laura Martin

CLUSTER\_ANALYSIS\_main. In details.

**>defineClustersStruct**

**>FindClusterStruct** Set analysis **parameters**

**>FindClusters**

Identifies Islands and clusters

**Calculates clusters’ features**:

* n° localizations
* Areas
* NND (of In-Island only)

**>saveClusterMetricData**

**>extractClusterStats**

**>DistanceDualColor**

Calculates the **global NND**

between ALL clusters

**Saves** output files

(.xlsx, .mat, .bin, .dcc, .fig, .png)

**Detailed description of CLUSTER\_ANALYSIS\_main workflow**

In **CLUSTER\_ANALYSIS\_main** we can define “rootdir”, if to “plotstats” or not, and the “pix2nm” size of the microscope we used to acquire the images.

Then, CLUSTER\_ANALYSIS\_main calls **defineFindClustersStruct**.

defineFindClustersStruct defines the GUI for entering the analysis parameters. Here we can change the size of the GUI (ln 12) and the names of the parameters to insert (e.g. ln 52).

defineFindClustersStruct calls **FindClustersStruct**.

FindClustersStruct defines the values of the parameters which will appear in the GUI as default.

CLUSTER\_ANALYSIS\_main calls **FindClusters**.

**FindClusters.m:**

Extracts information from localization list.

Resizes the image of a chosen zoom\_factor = original\_pixel\_size/analysis\_pixel\_size;

Xn are the locs coordinates changes for the new zoom

M is the matrix of the number of locs inside each pixel

Mf = imfilter(full(M),ones(roi)); imfilter sums the number of molecules of each pixel of a roi to the central pixel of the roi. Thus, Mf is the matrix of all pixels values after the summation of the number of localizations.

Mf2 is an empty matrix of the same size of M. It is filled with only the pixels which contain a number of locs >threshold.

* Alert! If Mf2 is not filled, it means that the code found no pixels with locs >threshold. Thus the threshold values is too high and should be reduced.
* If we put True for Use Iterative Segmentation, go to ln 920-1104, function segment\_nn(Mf,Mf2,BW,th,max\_area). If there is any Island with an area >max\_segmentation\_area, then this Island is segmented to reduce it.

Mf2 is converted to a Black & White image (BWL). Pixels ≠ 0 are White (1), pixels = 0 are Black (0).

Regionprops(Mf2) automatically identifies Islands in the B&W image of Mf2.

m is the matrix containing all the Islands (j).

BB=m(j).BoundingBox identifies each white block [=Island, =m(j)] and isolates it by the function Bounding Box (square).

Mi=m(j).Image. Each Island [m(j)] is converted into an image.

xi\_b is the matrix containing the coordinates of localizations of each Island.

[iok, jok] is the matrix of the pixels of each Island [only the White (1) pixels of each Bounding Box].

* If an Island has a number of pixels (iok) > ignoreNumPeakThreshold, this Island will be discarded from the analysis. If you want to analyze every Island, increase the ignoreNumPeakThreshold (keep in mind that the analysis will take very long).

xi\_c is the matrix of localization coordinates belonging to each Island (Mi), updated after the filtering of Islands [(iok) > ignoreNumPeakThreshold]. See function xi\_c in ln 866-918.

Mi\_c is a new empty matrix. It has the size of the Mi image, after resizing of a chose factor (defined in the parameters at the beginning).

* If an Island has a number of locs (xi\_c) > ignoreNumIslandThreshold, this Island will be discarded.

xo, yo are the new coordinates of locs of each Island (Mi), once resized of the factor.

sigma = precision / (analysis\_pixel\_size/factor). Sigma is defined as the Localization precision (in nm) (defined at the beginning) normalized to the new pixel size.

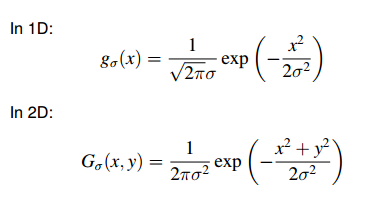
minx, maxx, miny, maxy. These coordinates define a square roi around each localization.  
The extremities of this square are:  
 - mix x = left  
 - max x = right  
 - min y = up  
 - max y = down  
These extremities are located at numSigma\*sigma from the coordinates of the localization.  
numSigma is set as 3 but can be changed. The smaller the numSigma, the smaller the area associated to each localization.

In each square roi localization, a kernel Gaussian is drawn, with the peak corresponding to the coordinates x,y of the localization.  
Mi\_c(yval,xval) = Mi\_c(yval,xval) + G(xval, yval, x0, y0, sigma).

The formula for the kernel 2D Gaussian is the following:  
G= @(x,y,x0,y0,rx) (1./((2\*pi)\*(rx\*rx))).\*exp(-(((x-x0)).^2)/(2\*rx^2).\*exp(-(((y-y0)).^2)/(2\*rx^2))

Function handle

Formula



All kernel Gaussias belonging to the same Island are summed together, generating a local density map. This matrix is called Mi\_c.

IM\_BW is a new empty matrix of the same size of Mi\_c.

The functions bwlabel and regionprops are used to extract the feature of the Centroids from the Mi\_c. This process is repeated for each Island (Mi\_c).

Function lik\_sig associates the localizations (N) to each Centroid previously automatically found through regionprops. See function lik\_sig in ln 784-862.

Then N (localizations belonging to one cluster) are filtered. Only clusters having N ≥ minCluster are kept, the others are discarded. minCluster is defined in the parameters at the beginning as minimum\_molecules\_per\_cluster.

The Centroid of each cluster (previously automatically extracted by the function regionprops) is re-calculated as the mean of the coordinates of N localizations. These new coordinates should agree with the ones previously found by regionprops.

Then, all features calculated are summarized:

* X = x coordinate of the centroid
* Y = y coordinate of the centroid
* NN = Number of total localizations of the cluster
* SIGX = Standard deviation of x coordinates of cluster localizations
* SIGY = Standard deviation of y coordinates of cluster localizations
* SIG = Average of standard deviation of x and y coordinates.
* SIGQUAD = sqrt((xstd\*xstd + ystd\*ystd)) ]; only for clusters inside Islands (thus, NOT for single clusters), it calculates the Nearest Neighbour Distance (NND).
* Z = z coordinate of the centroid.   
  (Calculated as the average of z- coordinates of localizations belonging to the cluster.)
* SIGZ = Standard deviation of z coordinates of cluster localizations
* xyzNND = X and Y of centroid(s) of the cluster(s) of one Island, in nm. (Temporary)
* IslandIdx = index of the Island to which the cluster belongs to

xyzNND is a matrix containing the x-y-z-coordinates of the centroid(s) of one Island.  
If ncisland is the dimension of xynNND matrix. If ncisland >1 it means that the Island contains more than >1 cluster, so the NND can be computed, as:  
squareform(pdist(xyzNND)) + diag(Inf\*ones(size(xyzNND,1)  
Otherwise, it means that xynNND has only one cluster, so it is not an Island but a Single Cluster, and thus NND is “Inf”.

NCI is the index of the clusters inside the Island (1, 2, 3, ...) or Single (1).

Features of every cluster identified will be summarized in columns in an Excel.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X | Y | NN | SIGX | SIGY | SIG | SIGQUAD | Z | SIGZ | NND | NCI | IslandIdx |

Finally, the code extracts few other parameters:

* medN = Median of number of localizations per cluster
* medArea = Median of cluster area.   
  Cluster area is calculated as π\*(SIG)2.
* Median density = medN / medArea

Additionally, in the case of 3D STORM images, the code will do a further step of analysis for better data visualization with software Insight3.   
In details:  
- if in the initial GUI we put “True” for drawUsingZRange, the localizations of each cluster will be assigned to 9 different channels, based on the z-coordinate of theri centroid. The z- range is defined as the interval between the max and min z-coordinate of all centroids, divided per 9. z-min correspond to magenta color, while z-max corresponds to red color. So, localizations will be displayed with a different color depending on the z-coordinate of the centroid of the cluster to which they belong. Localizations of different clusters will be displayed with the same color if the z-coordinate of their centroids belong to the same interval.  
- if in the initial GUI we put “False” instead, all localizations belonging to the same cluster will be displayed with the same color, randomly chosen among the 9 available (with no correlation with the z coordinate of each centroid).

Then CLUSTER\_ANALYSIS\_main calls **saveClusterMetricData.**

SaveClusterMetricData creates a temporary .xyn file containing the parameters used for the analysis (ln 18). It then calls **extractClusterStats**.

extractClusterStats calls **DistanceDualColor**,and defines parameters of cluster files, categories and minimum number of molecules per cluster in ln 151-152.

**DistanceDualColor:**

It was originally created to measure the distance between two different protein clusters (labelled with 2-colors). Inside this code, it is used for 1-color cluster images.

It takes one cluster file (.xyn) generated from FindClusters.m and doubles it, as they were two different lists of clusters (=data1; data2).

Clusters listed in data1 and data2 are checked for dimension. They have to contain > minimum\_localizations to be further analysed.

It calculates the distance in x-y-z dimensions between data1 and data2 in nm:

Dist = sqrt( ((data1(i,1)-data2(j,1))\*pixel\_size\_nm)^2 + ((data1(i,2)-data2(j,2))\*pixel\_size\_nm)^2 + (data1(i,8)-data2(j,8))^2);

Every time the same cluster is picked, distance is set as “Inf” and ignored.

if isSameFile dist(j,k) = Inf;

All distances are sorted (function srt) in order to pick the closest neighbour for every cluster.

The clusters can be separated into categories, depending on their distance. For this code, categories are not defined (ln 181-182), so NNDs are listed all together.

{Otherwise, categories can be defined (e.g. [ 70 250 ] so that NND are divided into 3 groups: NNDs < 70 nm, 70 < NNDs < 250 nm, NNDs > 250 nm), see from ln 184}.

**extractClusterStats** saves the list of NND global distances in a ddc file called “ddcData.nndXY”.

**SaveClusterMetricData** appends the results “ddcData.nndXY” to the results generated in FindClusters.m as a 13th column.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X | Y | NN | SIGX | SIGY | SIG | SIGQUAD | Z | SIGZ | NND | NCI | IslandIdx | NND global |

This table of 13 columns is temporarily saved as the matrix “allResults”.

The .xyn file previously generated in ln 18 is temporarily saved with a name containing the original name of the list file plus the abbreviations of the parameters used for the analysis (ln 55 or ln 57).

* Warning! If the original list file has a long name, the code could give error when saving the final .xyn. Apparently .xyn cannot be open correctly when they have a name > 256 characters. If .xyn file gives problem, rename the .xyn (reducing the number of characters) or rename the original list and re-run the cluster analysis.

Inside the .xyn file, the results of “medianNum, “median Area[nm^2]” and “medianDensity” are appended (ln 79, 81, 82).

Finally, the 13 columns table with all clusters feautures is appended (ln 85). The .xyn in now completed and saved for the last time.

At the very end, all the results calculated up until now are saved in a .mat file (ln 103).

Back in **CLUSTER\_ANALYSIS\_main** the plot statistics (if we had put “true”in ln 5) are saved as .fig and .png files.