**Scripts**

*Analysis script for particle clustering in SEM images using MATLAB 2021a*

%%% the script reads .txt-output from ImageJ formatted with the following columns in that order: Area,Mean,Min,Max,X,Y,Circ.,Feret,FeretX,FeretY,FeretAngle,MinFeret,AR,Round,Solidity

clear all; close all; version=3.3;

mainfp='U:\Post Doc\Databehandling\SEM\Data for clustering analysis ....'; % folder for looking for files

foldername = uigetdir(mainfp,'Point me to the results DIRECTORY'); %opens GUI for selecting folder containing ImageJ-output

fp=[foldername '\'];

pxlnm=0.1374; %pixel per nanometer value

Rtol=15; %radial tolerance

Rshrink=10; %shrinking of radius

maxdist=13.74; %distance criterion for determining cluster relations

resolution=[1020 882]; %Resolution WxH of picture

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% script start %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if exist([ fp 'analysisoutput'])~=7; %create folder for output

mkdir(fp,'analysisoutput') ;

end

outputfolder='analysisoutput\'; outputfp=[ fp outputfolder ];

save([ outputfp 'geometryworkspace']); %save workspace

fname=dir([ fp '\*RES.txt']);

maxn=size(fname,1); %maxn holds the number of files for analysis

for n=1:maxn;

content{n} = importdata([ fp fname(n).name ],'\t',1); %load data from inputfiles

X=sprintf('%s loaded',fname(n).name(1:end-12)); %print the progress

disp(X)

end

%%%%%% analysis starts here %%%%%

disp('Starting analysis...')

for n=1:maxn;

X=sprintf('Working on %i / %i file %s',n,maxn,fname(n).name(1:end-12)); disp(X); %print progress

%%%%%%%%%%%%%%%%%% exclusion on size %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ressize{n}=size(content{n}.data,1); %results size M

maxsize=pi\*(R\_PS+Rtol)^2; %maximum tolerated size

minsize=0.0001\*pi\*(R\_PS-Rshrink-Rtol)^2; %minimum tolerated size (0.0001 inserted)

indexSize{n}=find(content{n}.data(:,2)>=minsize & content{n}.data(:,2)<=maxsize); %filter for sizes defined in the beginning of script

discardsSize{n}=find(content{n}.data(:,2)>maxsize & content{n}.data(:,2)<minsize); %discarded based on size

filteredcontentS{n}=content{n}.data(indexSize{n},:); %S for size filtered

NexclS{n}=numel(discardsSize{n}); %number of entries discarded based on size

NpassS{n}=numel(indexSize{n}); %number of entries passed based on size

%%% making things manageable

x0{n}=filteredcontentS{n}(:,6);

y0{n}=filteredcontentS{n}(:,7);

Area{n}=filteredcontentS{n}(:,2);

Circ{n}=filteredcontentS{n}(:,8);

distances{n}=squareform(pdist([x0{n} y0{n}])); %setup n x m matrix with (n,m)'th entry having the distance between particle n and particle m

neighbours{n}=distances{n}<=maxdist & distances{n}~=0; %evaluate which particle distances are lower than the user defined threshold value

figure

bar(distances{n})

%setup empty vectors/matrices for data collection

Nneighbours{n}=zeros(1,NpassS{n});

pairs{n}=zeros(NpassS{n},2);

trickycluster{n}=zeros(NpassS{n},1);

lonely{n}=zeros(NpassS{n},1);

%%%%%%%%%%%%% evaluating neighbours particle by particle %%%%%%%%%%%%%%%%%

for k=1:size(neighbours{n},1);

neighbourindex=find(neighbours{n}(k,:)==1); %vector containing the index of particle k's neighbours

Nneighbours{n}(k)=numel(neighbourindex); %number of neighbours of particle k

% sort particles into pairs, loners and clusters

if Nneighbours{n}(k)==1 %if only one neighbour, evaluate next neighbour's neighbours

neighbourneighbourindex=find(neighbours{n}(neighbourindex,:)==1); %index of particle k's neighbour's neighbours

if numel(neighbourneighbourindex)==1 & neighbourneighbourindex==k & neighbourindex>k;% if only one neighbour and neighbours neighbours is particle k, this must be a pair

pairs{n}(k,:)=[k neighbourindex]; %pairs contains index of observations in a pair sorted in rows

elseif numel(neighbourneighbourindex)>1 %if next neighbour has more than one neighbour pairs are clustered

trickycluster{n}(k)=1;

end

elseif Nneighbours{n}(k)==0 %if no neighbours observed save index in lonely

lonely{n}(k)=1;

elseif Nneighbours{n}(k)>1 %if number of neighbours is greater than one, register as a tricky cluster

trickycluster{n}(k)=Nneighbours{n}(k); %the k'th entry of tricky cluster holds the number of neighbours of particle k

end

end

%remove zeros from lists

Npairs{n}=nnz(pairs{n}(:,1));

Npairparticles{n}=nnz(pairs{n});

pairlist{n}=[ pairs{n}((pairs{n}(:,1)~=0),:) ];

Ntrickycluster{n}=nnz(trickycluster{n});

trickyclusterlist{n}=find(trickycluster{n}~=0);

Nlonely{n}=nnz(lonely{n});

lonelylist{n}=lonely{n}(lonely{n}~=0);

exclmultipct{n}=Ntrickycluster{n}\*100/NpassS{n}; %evaluate exclusion percentage

exclpct{n}=(Ntrickycluster{n}+Nlonely{n})\*100/NpassS{n};

if Npairparticles{n}+Ntrickycluster{n}+Nlonely{n}~=NpassS{n} %check that all particle are accounted for and issue warning if not

X=sprintf('Warning! Not all particles accounted for \r\n %i entries in file vs %i particles in pairs, %i particles had multiple neighbours and %i single particles found',NpassS{n},Npairparticles{n},Ntrickycluster{n},Nlonely{n}); %print

disp(X)

elseif Npairparticles{n}+Ntrickycluster{n}+Nlonely{n}==NpassS{n}

X=sprintf('All particles accounted for'); disp(X);

X=sprintf('%i entries in file',NpassS{n}); disp(X);

X=sprintf('%i particles in pairs, %i particles had multiple neighbours and %i single particles found',Npairparticles{n},Ntrickycluster{n},Nlonely{n}); disp(X);%print

end

end

%%%%%%%%%%%%%%%% create xls file with all particles neighbours %%%%%%%%%%%%

for u=1:maxn

theArray = Nneighbours{u}';

for row = 1 : size(theArray, 1)

ca{row,u} = theArray(row);

end

end

xlswrite('Nneighbours.xls', ca);

%%%%%%%%%%%%%%%%%% count the number of each unique value in Nneighbours %%%%%%%%%%%%%%%%%%

ca2 = cell2mat(Nneighbours);

xx = unique(ca2); % temp vector of vals

x = sort(ca2); % sorted input aligns with temp (lowest to highest)

t = zeros([size(xx),2]); % vector for freqs

% frequency for each value

for i = 1:length(xx)

t(i,1) = sum(x == xx(i));

t(i,2)= xx(i);

end

disp('Number of particles with n neighbours')

disp(t)

%%%%%%%%%%%%%%%% create xls file with all tricky clusters neighbours %%%%%%%%%%%%

for u=1:maxn

theArray = trickycluster{u};

for row = 1 : size(theArray, 1)

ca{row,u} = theArray(row);

end

end

xlswrite('trickycluster.xls', ca);

close all