An Image-Processing Tool for Size and Shape Analysis of Manufactured Irregular Polyethylene Microparticles

Melanie Fritz [1,]*, Lukas F. Deutsch [1], Karunia Putra Wijaya [2], Thomas Götz [2] and Christian B. Fischer [1,3,]*

[1,2] Department of Physics, Faculty of Mathematics and Natural Sciences, University of Koblenz, Universitätsstr. 1, 56070 Koblenz, Germany
[2] Department of Mathematics, Faculty of Mathematics and Natural Sciences, University of Koblenz, Universitätsstr. 1, 56070 Koblenz, Germany
[3] Material Science, Energy and Nano-Engineering Department, Mohammed VI Polytechnic University, Lot 660, Hay Moulay Rachid, Ben Guerir 43150, Morocco
* Correspondence: fritz.melanie@web.de (M.F.); chrbfischer@uni-koblenz.de (C.B.F.); Tel.: +49-261-287-2345 (C.B.F.)

List of contents

SI1: Developed scripts

Script S1: MP_Analysis_Particle.m

The following algorithm was used for the evaluation of the 1st series with image format (TIF, resolution: 1406 x 1026 px).

```
%%%Clearing storage%%%
clear all; %clear all variables
##clc; %clear console
close all; %close all figures

%%%Initiatlizing Packages%%%
%Image Package Initialization: Core octave package
pkg load image;
%Computational Geometry Initialization: Requires one-time installation via "pkg install -forge matgeom" in prompt after downloading here: https://octave.sourceforge.io/download.php?package=matgeom-1.2.2.tar.gz
pkg load matgeom;

%%%Loading Images from folder%%%
[stat,img_idx]=fileattrib('*.tif');
%Number of images, is the length of the indexed vector
img_amount=length(img_idx);
%Helping variable for global shape parameters
length_old=0;
%Helping variable for global size evaluation
length_part_old=0;

%%%Settings%%%
%Select the contrast of the original image. If the background is very light, set bright image value 1
bright_image=0;
%Magnification of image enhancement
mag=20;
%Specify the border of image section to be analyzed in horizontal (x) direction or verical (y) direction
%Assumes, that SEM image labeling is superimposed in the bottom half.
dim_x=1406;
dim_y=1026;
%Select the edge detection method (Set edge_method "1" for Sobel or "2" for Canny or "3" for Thresholding by Otsu)
edge_method=2;
%Select shape analysis by setting the variable "1", otherwise, give it value "0"
shape_option=1;
%Select geometric overlay method by setting variable "1", otherwise give it different value
shape_geometry=0;
%Select value "1", if shape analysis statistics for single images are wanted, set it "0" otherwise
```

```
single_image_shape=0;
%Initialization of structuring elements
SE_tophat=strel("disk",10,0); %Structuring Element for top-hat Transformation
SE_close=strel("disk",4,0); %Structuring Element for closing after edge detection
SE_bound=strel("disk",3,0); %Thickness of the boundary
%Threshold value for the height of H-max transform
height_th=5;
%Select the value of the LDPE particle sample. It is found in the status bar of SEM images
LDPE=1.4;

switch(LDPE)
%Calculate the minimum and maximum sieve diameter in µm depending on sieve identifier
case{1.1}
  sieve_min=0;
  sieve_max=500;
case{1.2}
  sieve_min=400;
  sieve_max=500;
case{1.3}
  sieve_min=300;
  sieve_max=400;
case{1.4}
  sieve_min=200;
  sieve_max=300;
case{1.5}
  sieve_min=150;
  sieve_max=200;
case{1.6}
  sieve_min=125;
  sieve_max=150;
case{1.7}
  sieve_min=100;
  sieve_max=125;
case{1.8}
  sieve_min=50;
  sieve_max=100;
case{2.1}
  sieve_min=0;
  sieve_max=300;
case{2.2}
  sieve_min=200;
  sieve_max=300;
case{2.3}
  sieve_min=150;
  sieve_max=200;
case{2.4}
  sieve_min=125;
  sieve_max=150;
case{2.5}
  sieve_min=100;
  sieve_max=125;
case{2.6}
  sieve_min=50;
  sieve_max=100;
case{2.7}
  sieve_min=25;
  sieve_max=50;
endswitch

%%%Image Analysis%%%
for k=1:img_amount
    %%Selecting and cropping Image
    img_in=imread(img_idx(k).Name,"PixelRegion",{[1 dim_y],[1 dim_x]});

    %%TopHat-Filtering
    img_tophat_gray=imtophat(img_in,SE_tophat);
    img_tophat=im2double(img_tophat_gray);
##    img_tophat=imtophat(img_in,SE_tophat);
##    figure, imhist(img_tophat), axis "auto y";
    img_adj=imadjust(img_tophat);
##    img_adj=imadjust(img_tophat,[0.001;0.2], [0.5;1],1);
##    figure, imhist(img_adj), axis "auto y";
##    figure, imshow(img_adj);

    %%Edge-Finding
  switch(edge_method)
  case{1}
    %Sobel Edge-Finding algorithm
    img_edge=edge(img_adj,"Sobel");
##    figure,subplot(121),imshow(img_adj),subplot(122),imshow(img_edge);
  case{2}
    %Calculating the threshold via Otsu's method
    thresh=graythresh(img_adj);
    %Canny Edge-Finding algorithm
    img_edge=edge(img_adj,"Canny",0.5*thresh);
##    figure, imshow(img_edge);
##    figure,subplot(121),imshow(img_adj),subplot(122),imshow(img_edge);
  case{3}
```

```
    img_edge=im2bw(img_adj,graythresh(img_adj));
##    figure, imshow(img_edge);
  otherwise
    printf("Please enter a valid value for the Edge-Finding method or define another one in the code using your selected value under the variable edge_method");
  endswitch

    %%Closing holes on the edges
    img_closed=imclose(img_edge,SE_close);
##    figure, subplot(121), imshow(img_edge), subplot(122), imshow(img_closed);

    %%Flood filling the enclosed areas. Filling before the watershed can lead to issues with particle agglomerates
    img_filled=imfill(img_closed,"holes");
##    figure, imshow(img_filled);
##    figure, subplot(121), imshow(img_closed), subplot(122), imshow(img_filled);

  %%Calculate the minimum size of particles considered for morphology depending on sieve size
  switch(mag)
  % 80 percent of all particles evaluated possess an fitted ellipse reciprocal aspect ratio (rar) of 1/2 or higher, which is used to define upper thresholds
  case{20}
    %Min size particles have circular cross section the size of the sieve opening
##    th_min_metric=pi*(sieve_min/2)^2;
    th_min_metric=1.05*sieve_min^2;
    %Max size particles possess circular cross sections with cylindric shape and rar 1/3
    th_max_metric=pi*3/4*sieve_max^2;
    %Convert from metric to px size thresholds
    th_min_px=((188/1000)^2)*th_min_metric;
    th_max_px=((188/1000)^2)*th_max_metric;
  case{50}
    %Min size particles have circular cross section the size of the sieve opening
    th_min_metric=1.05*sieve_min^2;
    %Max size particles have elliptical cross sections with rar 1/3, resulting in rectangles on the image
    th_max_metric=pi*3/4*sieve_max^2;
    %Convert from metric to px size thresholds
    th_min_px=((281/600)^2)*th_min_metric;
    th_max_px=((281/600)^2)*th_max_metric;
  case{110}
    %Min size particles have circular cross section the size of the sieve opening
    th_min_metric=1.05*sieve_min^2;
    %Max size particles have elliptical cross sections with rar 1/3, with minor axis being the size of the cross section
    th_max_metric=pi*3/4*sieve_max^2;
    %Convert from metric to px size thresholds
    th_min_px=((309/300)^2)*th_min_metric;
```

```
   th_max_px=((309/300)^2)*th_max_metric;
  endswitch



   %%Removal of small particles/noise(2nd argument in bwareaopen enotes the boundary in px²
   img_clear1=bwareaopen(img_filled,th_min_px,4);
##    figure,imshow(img_clear1);

   %%Watershed iteration
   %Calculates the Negative of the inverse of the binary image created by edge detection, filling and noise removal
   distmap=bwdist(~img_clear1);
##    distmap_img=distmap./(max(max(distmap)));
##    figure, imshow(distmap_img);
   img_marker = imextendedmax(distmap,height_th,4);
##    mask2 = imextendedmin(distmap,4,4);
##    figure,imshow(img_marker);
   %Imimposemin makes sure only regional minima at non zero pixels of mask
   minima = imimposemin(-distmap,img_marker);
   img_ridgelines = watershed(minima);
##    figure, imshow(img_ridgelines);
   img_shed=img_clear1;
   %The preprocessed image is getting shed in the places, where L has value 0
   img_shed(img_ridgelines == 0) = 0;
##    figure,imshow(img_shed);

   %%Clearing the particles touching the borders(earlier application leads to removal of overlapping objects)
   img_clear=imclearborder(img_shed,4);
##    figure, imshow(img_clear);

   %Codepart for troubleshooting of filter model assumptions
##    img_bound_help=im2bw(img_clear-imerode(img_clear,SE_bound));
####    figure, imshow(img_bound_help);
##    img_overlay_help=img_in;
##    if bright_image==1
##      img_overlay_help(img_bound_help==1)=0;
##      figure, imshow(img_overlay_help);
##    else
##      img_overlay_help(img_bound_help==1)=255;
##      figure, imshow(img_overlay_help);
##    end

   %%Application of sieve-fraction sensitive size filtering
   %Remove particles belonging in a smaller sieve fraction and oversegmented particles
   img_clear2=bwareaopen(img_clear,th_min_px,4);
```

```
##    figure, imshow(img_clear2);
  %Retain particle agglomerations and particles belonging to a larger sieve fraction
  img_clear3=bwareaopen(img_clear,th_max_px,4);
##    figure, imshow(img_clear3);
  %Retain only particles which belong in the selected sieve fraction
  img_clear4=im2bw(img_clear2-img_clear3);
##    figure, imshow(img_clear4);

  %%Calculating the boundary of the particles
  img_bound=im2bw(img_clear4-imerode(img_clear4,SE_bound));
##    figure, imshow(img_bound);

  %%Superimpose the boundary of the particles on top of the original image
  img_overlay=img_in;
  if bright_image==1
    img_overlay(img_bound==1)=0;
##      figure, imshow(img_overlay);
##      figure, subplot(121), imshow(img_filled), subplot(122), imshow(img_overlay);
  else
    img_overlay(img_bound==1)=255;
##      figure, imshow(img_overlay);
##      figure, subplot(121), imshow(img_filled), subplot(122), imshow(img_overlay);
  end

  %%Generating statistics
  switch(shape_option)
  case{0}
  stats = regionprops(img_clear4,"all");
  sizeA = [stats.Area]; %size of regions in px²
  num_regions = length(stats); %number of regions

  case{1}
  stats = regionprops(img_clear4,"all");
  sizeA = [stats.Area]; %size of regions in px²
  majX = [stats.MajorAxisLength]; %length of the major axis of fitted ellipse
  minX = [stats.MinorAxisLength]; %length of the minor axis of fitted ellipse
  num_regions = length(stats); %number of regions

  %%Iteration that calculates the Max-Feret-Diameter and Bounding-Box of each particle by using the convex hull. Brute force
  bb_area = zeros(1,num_regions); %Vector that contains the Bounding-Box size for each particle
  maxferetdiam = zeros(1,num_regions); %Vector that contains the Max Feret Diameter for each particle
  for i=1:num_regions
    bb_info = stats(i).BoundingBox; %Read out the Bounding-Box info for the i-th particle
    bb_area(i) = bb_info(3)*bb_info(4); %Multiply the length with the width of the bounding box
```

```
   particle_convexhull = stats(i).ConvexHull; %Read out the convex hull coordinates of the i-th particle
   dist_vert = distancePoints(particle_convexhull, particle_convexhull); %Calculate the euclidian distance in px for each vertex of the convex hull of the i-th particle
   maxferetdiam(i) = max(max(dist_vert)); %Write the maximum entry of the distance Matrix for the i-th particle into the corresponding vector
  endfor

  %%Calculating shape parameters
  %(reciprocal axis ratio RAR)
  rar = minX./majX;
  %(rectangularity RTY)
  rty = sizeA./bb_area;
  %(feret major axis ratio FMR)
  fmr = maxferetdiam./majX;

  if shape_geometry == 1
   %%Shape analysis
   num_tri=num_ell=num_rec=num_und=zeros(1,num_regions);
   num_total=zeros(1,4); %Vector that's utilized to save the total count of all shapes identified
   for j=1:num_regions
    if (rty(j) <= 0.6 && fmr(j) >= 1.27)
     num_tri(j)=1;
    else
     num_tri(j)=0;
    endif
    if ((rar(j) >= 0.98 && rty(j) >= 0.98) || (rar(j) >= 0.97 && fmr(j) >= 1.2) || (rar(j) < 0.98 && rty(j) >= 0.98) || (fmr(j) < 0.98 || fmr(j) > 1.09))
     num_rec(j)=1;
    else
     num_rec(j)=0;
    endif
    if ((rar(j) >= 0.99 && (fmr(j) > 0.995 && fmr(j) <= 1.15)) || (fmr(j) >= 0.98 && fmr(j) <= 1.09))
     num_ell(j)=1;
    else
     num_ell(j)=0;
    endif
    %If Shape parameters overlap and don't allow clear identification, a particle is classified as undecided/inconclusive
    if ((num_tri(j)==1 && num_rec(j)==1) || (num_tri(j)==1&&num_ell(j)==1) || (num_rec(j)==1&&num_ell(j)==1))
     num_tri(j)=num_rec(j)=num_ell(j)=0;
     num_und(j)=1;
    endif
   endfor
   num_total(1)=sum(num_ell);
   num_total(2)=sum(num_rec);
```

```
num_total(3)=sum(num_tri);
num_total(4)=sum(num_und);

%%Histogram for particle shape distribution
figure
bar(num_total,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
xlabel('Shape');
ylabel('Count');
xticks(1:4);
xticklabels({'Ellipse','Rectangle','Triangle','Inconclusive'});
name_shape_hist=sprintf(['picture_',num2str(mag),'_',num2str(k),'_shape_histogram','.png']);
saveas(1,name_shape_hist);
close all;

%%Superimpose the detected ellipse or rectangle over the outline of the detected particle
figure
imshow(img_bound)

t = linspace(0,2*pi,100);
hold on
for i = 1:num_regions
  if (num_ell(i)==1)
    a = stats(i).MajorAxisLength/2;
    b = stats(i).MinorAxisLength/2;
    Xc = stats(i).Centroid(1);
    Yc = stats(i).Centroid(2);
    phi = deg2rad(-stats(i).Orientation);
    x = Xc + a*cos(t)*cos(phi) - b*sin(t)*sin(phi);
    y = Yc + a*cos(t)*sin(phi) + b*sin(t)*cos(phi);
    plot(x,y,'r','LineWidth',0.5)
  endif
  if (num_rec(i)==1)
    le = stats(i).MajorAxisLength/2*0.886227;
    wi = stats(i).MinorAxisLength/2*0.886227;
    Xc = stats(i).Centroid(1);
    Yc = stats(i).Centroid(2);
    center = transpose(stats(i).Centroid);
    theta = deg2rad(stats(i).Orientation);
    coords = [Xc-le Xc-le Xc+le Xc+le;...
              Yc-wi Yc+wi Yc+wi Yc-wi];
    R = [cos(theta) sin(theta);...
        -sin(theta) cos(theta)];
    rot_coords = R*(coords-repmat(center,[1 4]))+repmat(center,[1 4]);
    rot_coords(:,5)=rot_coords(:,1);
```

```
      line(rot_coords(1,:),rot_coords(2,:),'color','g','LineWidth',0.5);
    endif
  endfor
  name_overlay_shape=sprintf(['picture_',num2str(mag),'_',num2str(k),'_shape_overlay','.png']);
  saveas(1,name_overlay_shape);
  hold off

  close all;
endif

%%Calculating Convexity and Solidity of the particle to describe deviation from a convex cross section
solidity_particle=convexity_particle=perim_cvxpoly=zeros(1,num_regions);
for i=1:num_regions
  solidity_particle(i)=stats(i).Solidity;
  perim_cvxpoly(i)=polygonLength(stats(i).ConvexHull);
  convexity_particle(i)=perim_cvxpoly(i)./stats(i).Perimeter;
endfor

%%Global shape parameter distribution
if length_old==0
  rar_out=rar;
  fmr_out=fmr;
  cvx_out=convexity_particle;
  sol_out=solidity_particle;
  length_old=num_regions;
else
  %Save prior shape parameters in help array
  rar_old=rar_out;
  fmr_old=fmr_out;
  cvx_old=cvx_out;
  sol_old=sol_out;
  %Determine length of new shape parameter array
  length_new=length_old+num_regions;
  %Initialize new output arrays
  rar_out=fmr_out=sol_out=cvx_out=zeros(1,length_new);
  for i=1:length_old
    rar_out(i)=rar_old(i);
    fmr_out(i)=fmr_old(i);
    sol_out(i)=sol_old(i);
    cvx_out(i)=cvx_old(i);
  endfor
  for i=(length_old+1):(length_new)
    rar_out(i)=rar(i-length_old);
    fmr_out(i)=fmr(i-length_old);
```

```
      sol_out(i)=solidity_particle(i-length_old);
      cvx_out(i)=convexity_particle(i-length_old);
    endfor
    length_old=length(cvx_out);
  end

  %%Plotting Convexity and Solidity
  if single_image_shape==1
    [count_solid,bincenter_solid]=hist(solidity_particle,3);
    figure
    bar(bincenter_solid,count_solid,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Solidity A_{Particle}/A_{Convexhull}');
    ylabel ('Count');
    name_hist_solidity=sprintf(['picture_',num2str(mag),'_',num2str(k),'_solidity_histogram','.png']);
    saveas(1,name_hist_solidity);
    close all;

    [count_convex,bincenter_convex]=hist(convexity_particle,3);
    figure
    bar(bincenter_convex,count_convex,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Convexity P_{Convexhull}/P_{Particle}');
    ylabel ('Count');
    name_hist_convexity=sprintf(['picture_',num2str(mag),'_',num2str(k),'_convexity_histogram','.png']);
    saveas(1,name_hist_convexity);
    close all;
  endif

  otherwise
    printf("The currently selected value for shape_option is not implemented. Please redo with supported shape_option values or
add a new case");
  endswitch



  %%The following section calculates corresponding metric areas in [µm²] from digital unit in [px²]
  %%Tutorial for adding new magnification levels: Copy a case, select the desired magnification level
  %%in the argument of case{}. Afterwards use GIMP or a comparable software to measure the distance scale
  %%in px. Measure from the left pixel of the left border of the left end towards the left border of the
  %%right end. The "Zähler" of the fraction of the variable sizeA_metric describes the length of the scale in
  %%[µm], the "Nenner" contains the measured length of the scale as described above in [px]

  %%Hier werden zusammenhängende Pixelflächen[px²] in metrische Flächen [µm²] umgewandelt.
  %%Zur Erweiterung einen case kopieren, die gewünschte Vergrößerung angeben und mit
  %%Bildbearbeitungssoftware (z.B. Gimp) mit Lineal-Funktion die Skala zwischen den jeweils linken Balkenenden vermessen.
  %%Der Zähler des Bruchs in sizeA_metric beschreibt die Skalalänge in µm, der Nenner die Pixelanzahl.
```

```
switch(mag)
case{110}
  sizeA_metric=((300/309)^2)*sizeA;
  maxparticle=max(sizeA_metric);
  [count,bincenter]=hist(sizeA_metric,5);
  figure
  bar(bincenter,count,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [µm²]');
  ylabel ('Particles');
  name_hist=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram','.png']);
  saveas(1,name_hist);
  close all;

  %%Global size statistics generation
  %First image
  if length_part_old==0
    size_out=sizeA_metric;
    length_part_old=num_regions;
  %Subsequent images
  else
    %Save prior particle sizes in help array
    size_old=size_out;
    %Determine length of the new size output array
    length_part_new=length_part_old+num_regions;
    size_out=zeros(1,length_part_new);
    for i=1:length_part_old
      %Save old particle sizes in new array
      size_out(i)=size_old(i);
    endfor
    for i=(length_part_old+1):(length_part_new)
      %Add particle sizes from current image
      size_out(i)=sizeA_metric(i-length_part_old);
    endfor
    %Adjust the length of the array required to save the particle sizes
    length_part_old=length(size_out);
  end

if single_image_shape==1
  %Histogram for shape specific subplots
  sizeA_ell=zeros(1,num_total(1));
  sizeA_rec=zeros(1,num_total(2));
  sizeA_tri=zeros(1,num_total(3));
  sizeA_und=zeros(1,num_total(4));
  m=n=o=p=1;
```

```
for i=1:num_regions
  if num_ell(i)==1
    sizeA_ell(m)=sizeA_metric(i);
    m++;
  endif
  if num_rec(i)==1
    sizeA_rec(n)=sizeA_metric(i);
    n++;
  endif
  if num_tri(i)==1
    sizeA_tri(o)=sizeA_metric(i);
    o++;
  endif
  if num_und(i)==1
    sizeA_und(p)=sizeA_metric(i);
    p++;
  endif
endfor

if num_total(1) ~= 0
  [cnt_ell,center_ell]=hist(sizeA_ell,3);
  figure
  bar(center_ell,cnt_ell,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [μm²]');
  ylabel ('Particles');
  name_hist_ell=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_ellipse','.png']);
  saveas(1,name_hist_ell);
  close all;
endif
if num_total(2) ~= 0
  [cnt_rec,center_rec]=hist(sizeA_rec,3);
  figure
  bar(center_rec,cnt_rec,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [μm²]');
  ylabel ('Particles');
  name_hist_rec=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_rectangle','.png']);
  saveas(1,name_hist_rec);
  close all;
endif
if num_total(3) ~= 0
  [cnt_tri,center_tri]=hist(sizeA_tri,3);
  figure
  bar(center_tri,cnt_tri,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [μm²]');
```

```
    ylabel ('Particles');
    name_hist_tri=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_triangle','.png']);
    saveas(1,name_hist_tri);
    close all;
  endif
  if num_total(4) ~= 0
    [cnt_und,center_und]=hist(sizeA_und,3);
    figure
    bar(center_und,cnt_und,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [µm²]');
    ylabel ('Particles');
    name_hist_und=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_undecided','.png']);
    saveas(1,name_hist_und);
    close all;
  endif

endif

case{221}
  sizeA_metric=((100/207)^2)*sizeA;
  maxparticle=max(sizeA_metric);
  [count,bincenter]=hist(sizeA_metric,5);
  figure
  bar(bincenter,count,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [µm²]');
  ylabel ('Particles');
  name_hist=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram','.png']);
  saveas(1,name_hist);
  close all;

  %%Global size statistics generation
  %First image
  if length_part_old==0
    size_out=sizeA_metric;
    length_part_old=num_regions;
  %Subsequent images
  else
    %Save prior particle sizes in help array
    size_old=size_out;
    %Determine length of the new size output array
    length_part_new=length_part_old+num_regions;
    size_out=zeros(1,length_part_new);
    for i=1:length_part_old
      %Save old particle sizes in new array
      size_out(i)=size_old(i);
    endfor
    for i=(length_part_old+1):(length_part_new)
      %Add particle sizes from current image
      size_out(i)=sizeA_metric(i-length_part_old);
    endfor
    %Adjust the length of the array required to save the particle sizes
    length_part_old=length(size_out);
  end

if single_image_shape==1
  %Histogram for shape specific subplots
  sizeA_ell=zeros(1,num_total(1));
  sizeA_rec=zeros(1,num_total(2));
  sizeA_tri=zeros(1,num_total(3));
  sizeA_und=zeros(1,num_total(4));
  m=n=o=p=1;
  for i=1:num_regions
    if num_ell(i)==1
      sizeA_ell(m)=sizeA_metric(i);
      m++;
    endif
    if num_rec(i)==1
      sizeA_rec(n)=sizeA_metric(i);
      n++;
    endif
    if num_tri(i)==1
      sizeA_tri(o)=sizeA_metric(i);
      o++;
    endif
    if num_und(i)==1
      sizeA_und(p)=sizeA_metric(i);
      p++;
    endif
  endfor

  if num_total(1) ~= 0
    [cnt_ell,center_ell]=hist(sizeA_ell,3);
    figure
    bar(center_ell,cnt_ell,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [μm²]');
    ylabel ('Particles');
    name_hist_ell=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_ellipse','.png']);
    saveas(1,name_hist_ell);
    close all;
  endif
  if num_total(2) ~= 0
    [cnt_rec,center_rec]=hist(sizeA_rec,3);
    figure
    bar(center_rec,cnt_rec,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [µm²]');
    ylabel ('Particles');
    name_hist_rec=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_rectangle','.png']);
    saveas(1,name_hist_rec);
    close all;
  endif
  if num_total(3) ~= 0
    [cnt_tri,center_tri]=hist(sizeA_tri,3);
    figure
    bar(center_tri,cnt_tri,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [µm²]');
    ylabel ('Particles');
    name_hist_tri=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_triangle','.png']);
    saveas(1,name_hist_tri);
    close all;
  endif
  if num_total(4) ~= 0
    [cnt_und,center_und]=hist(sizeA_und,3);
    figure
    bar(center_und,cnt_und,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [µm²]');
    ylabel ('Particles');
    name_hist_und=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_undecided','.png']);
    saveas(1,name_hist_und);
    close all;
  endif

endif

case{50}
  sizeA_metric=((600/281)^2)*sizeA;
  maxparticle=max(sizeA_metric);
  [count,bincenter]=hist(sizeA_metric,5);
  figure
  bar(bincenter,count,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [µm²]');
  ylabel ('Particles');
  name_hist=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram','.png']);
  saveas(1,name_hist);
  close all;

  %%Global size statistics generation
  %First image
  if length_part_old==0
    size_out=sizeA_metric;
    length_part_old=num_regions;
  %Subsequent images
  else
    %Save prior particle sizes in help array
    size_old=size_out;
    %Determine length of the new size output array
    length_part_new=length_part_old+num_regions;
    size_out=zeros(1,length_part_new);
    for i=1:length_part_old
      %Save old particle sizes in new array
      size_out(i)=size_old(i);
    endfor
    for i=(length_part_old+1):(length_part_new)
      %Add particle sizes from current image
      size_out(i)=sizeA_metric(i-length_part_old);
    endfor
    %Adjust the length of the array required to save the particle sizes
    length_part_old=length(size_out);
  end

if single_image_shape==1
  %Histogram for shape specific subplots
  sizeA_ell=zeros(1,num_total(1));
  sizeA_rec=zeros(1,num_total(2));
  sizeA_tri=zeros(1,num_total(3));
  sizeA_und=zeros(1,num_total(4));
  m=n=o=p=1;
  for i=1:num_regions
    if num_ell(i)==1
      sizeA_ell(m)=sizeA_metric(i);
      m++;
    endif
    if num_rec(i)==1
      sizeA_rec(n)=sizeA_metric(i);
      n++;
    endif
    if num_tri(i)==1
      sizeA_tri(o)=sizeA_metric(i);
      o++;
    endif
    if num_und(i)==1
      sizeA_und(p)=sizeA_metric(i);
      p++;
    endif
  endfor

  if num_total(1) ~= 0
    [cnt_ell,center_ell]=hist(sizeA_ell,3);
    figure
    bar(center_ell,cnt_ell,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [μm²]');
    ylabel ('Particles');
    name_hist_ell=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_ellipse','.png']);
    saveas(1,name_hist_ell);
    close all;
  endif
  if num_total(2) ~= 0
    [cnt_rec,center_rec]=hist(sizeA_rec,3);
    figure
    bar(center_rec,cnt_rec,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [μm²]');
    ylabel ('Particles');
    name_hist_rec=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_rectangle','.png']);
    saveas(1,name_hist_rec);
    close all;
  endif
  if num_total(3) ~= 0
    [cnt_tri,center_tri]=hist(sizeA_tri,3);
    figure
    bar(center_tri,cnt_tri,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [μm²]');
    ylabel ('Particles');
    name_hist_tri=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_triangle','.png']);
    saveas(1,name_hist_tri);
    close all;
  endif
  if num_total(4) ~= 0
    [cnt_und,center_und]=hist(sizeA_und,3);
    figure
    bar(center_und,cnt_und,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [μm²]');
```

```
   ylabel ('Particles');
   name_hist_und=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_undecided','.png']);
   saveas(1,name_hist_und);
   close all;
  endif

endif

case{20}
  sizeA_metric=((1000/188)^2)*sizeA;
  maxparticle=max(sizeA_metric);
  [count,bincenter]=hist(sizeA_metric,5);
  figure
  bar(bincenter,count,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [µm²]');
  ylabel ('Particles');
  name_hist=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram','.png']);
  saveas(1,name_hist);
  close all;

  %%Global size statistics generation
  %First image
  if length_part_old==0
   size_out=sizeA_metric;
   length_part_old=num_regions;
  %Subsequent images
  else
   %Save prior particle sizes in help array
   size_old=size_out;
   %Determine length of the new size output array
   length_part_new=length_part_old+num_regions;
   size_out=zeros(1,length_part_new);
   for i=1:length_part_old
    %Save old particle sizes in new array
    size_out(i)=size_old(i);
   endfor
   for i=(length_part_old+1):(length_part_new)
    %Add particle sizes from current image
    size_out(i)=sizeA_metric(i-length_part_old);
   endfor
   %Adjust the length of the array required to save the particle sizes
   length_part_old=length(size_out);
  end
```

```
if single_image_shape==1
 %Histogram for shape specific subplots
 sizeA_ell=zeros(1,num_total(1));
 sizeA_rec=zeros(1,num_total(2));
 sizeA_tri=zeros(1,num_total(3));
 sizeA_und=zeros(1,num_total(4));
 m=n=o=p=1;
 for i=1:num_regions
  if num_ell(i)==1
   sizeA_ell(m)=sizeA_metric(i);
   m++;
  endif
  if num_rec(i)==1
   sizeA_rec(n)=sizeA_metric(i);
   n++;
  endif
  if num_tri(i)==1
   sizeA_tri(o)=sizeA_metric(i);
   o++;
  endif
  if num_und(i)==1
   sizeA_und(p)=sizeA_metric(i);
   p++;
  endif
 endfor

 if num_total(1) ~= 0
  [cnt_ell,center_ell]=hist(sizeA_ell,3);
  figure
  bar(center_ell,cnt_ell,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [µm²]');
  ylabel ('Particles');
  name_hist_ell=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_ellipse','.png']);
  saveas(1,name_hist_ell);
  close all;
 endif
 if num_total(2) ~= 0
  [cnt_rec,center_rec]=hist(sizeA_rec,3);
  figure
  bar(center_rec,cnt_rec,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [µm²]');
  ylabel ('Particles');
  name_hist_rec=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_rectangle','.png']);
  saveas(1,name_hist_rec);
```

```
        close all;
      endif
      if num_total(3) ~= 0
        [cnt_tri,center_tri]=hist(sizeA_tri,3);
        figure
        bar(center_tri,cnt_tri,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
        xlabel('Area [μm²]');
        ylabel ('Particles');
        name_hist_tri=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_triangle','.png']);
        saveas(1,name_hist_tri);
        close all;
      endif
      if num_total(4) ~= 0
        [cnt_und,center_und]=hist(sizeA_und,3);
        figure
        bar(center_und,cnt_und,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
        xlabel('Area [μm²]');
        ylabel ('Particles');
        name_hist_und=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_undecided','.png']);
        saveas(1,name_hist_und);
        close all;
      endif

    endif


    otherwise
      printf("Area calculation isn't yet implemented for the selected magnification. Please redo with supported magnification or add
code for the required magnification by calculating area of 1px");
    endswitch

##  %Saving the B/W image of particle area and the perimeter of the particles superimposed on top of the original image
    name_img=sprintf(['picture_',num2str(mag),'_',num2str(k),'.png']);
    imwrite(img_clear4,name_img);
    name_overlay=sprintf(['picture_',num2str(mag),'_',num2str(k),'_overlay','.png']);
    imwrite(img_overlay,name_overlay);

endfor

%%Returning graphs containing information about particle size distribution over a series of images
  [count_size_global,bincenter_size_global]=hist(size_out,40);
  figure
  bar(bincenter_size_global,count_size_global,'FaceColor','r',"hist");
  xlabel('Area [μm²]');
  ylabel ('Count');
```

```
 name_hist_size_global=sprintf(['picture_',num2str(mag),'_size_global','.png']);
 saveas(1,name_hist_size_global);
 close all;
 csvwrite("size_global.csv",reshape(size_out,length_part_old,1));

%%The following function returns global statistics on shape parameters provided it is enabled
if (length_old > 0)
 [count_sol_global,bincenter_sol_global]=hist(sol_out,20);
 figure
 bar(bincenter_sol_global,count_sol_global,'FaceColor','r',"hist");
 xlabel('Solidity A_{Particle}/A_{Convexhull}');
 ylabel ('Count');
 name_hist_solidity_global=sprintf(['picture_',num2str(mag),'_solidity_global','.png']);
 saveas(1,name_hist_solidity_global);
 close all;
 csvwrite("solidity_global.csv",reshape(sol_out,length_part_old,1));

 [count_cvx_global,bincenter_cvx_global]=hist(cvx_out,20);
 figure
 bar(bincenter_cvx_global,count_cvx_global,'FaceColor','r',"hist");
 xlabel('Convexity P_{Convexhull}/P_{Particle}');
 ylabel ('Count');
 name_hist_convexity_global=sprintf(['picture_',num2str(mag),'_convexity_global','.png']);
 saveas(1,name_hist_convexity_global);
 close all;
 csvwrite("convexity_global.csv",reshape(cvx_out,length_part_old,1));

 [count_rar_global,bincenter_rar_global]=hist(rar_out,20);
 figure
 bar(bincenter_rar_global,count_rar_global,'FaceColor','r',"hist");
 xlabel('Reciprocal aspect ratio length(fitted minor axis)/length(fitted major axis)');
 ylabel ('Count');
 name_hist_rar_global=sprintf(['picture_',num2str(mag),'_rar_global','.png']);
 saveas(1,name_hist_rar_global);
 close all;
 csvwrite("reciprocalaspectratio_global.csv",reshape(rar_out,length_part_old,1));

 [count_fmr_global,bincenter_fmr_global]=hist(fmr_out,20);
 figure
 bar(bincenter_fmr_global,count_fmr_global,'FaceColor','r',"hist");
 xlabel('Feret major axis ratio length(feret major axis)/length(fitted major axis)');
 ylabel ('Count');
 name_hist_fmr_global=sprintf(['picture_',num2str(mag),'_fmr_global','.png']);
 saveas(1,name_hist_fmr_global);
```

```
  close all;
  csvwrite("feretmajoraxisratio_global.csv",reshape(fmr_out,length_part_old,1));
endif
```

Script S2: MP_Analysis_Particle_RGB.m

The following algorithm was used for the evaluation of the validation series (2nd series). Small changes had to be made to accommodate the different image format (JPG, RGB, resolution: 1024 x 768 px).

```
%%%Clearing storage%%%
clear all; %clear all variables
##clc; %clear console
close all; %close all figures

%%%Initiatlizing Packages%%%
%Image Package Initialization: Core octave package
pkg load image;
%Computational Geometry Initialization: Requires one-time installation via "pkg install -forge matgeom" in prompt after
downloading here: https://octave.sourceforge.io/download.php?package=matgeom-1.2.2.tar.gz
pkg load matgeom;

%%%Loading Images from folder%%%
[stat,img_idx]=fileattrib('*.jpg');
%Number of images, is the length of the indexed vector
img_amount=length(img_idx);
%Helping variable for global shape parameters
length_old=0;
%Helping variable for global size evaluation
length_part_old=0;

%%%Settings%%%
%Select the contrast of the original image. If the background is very light, set bright image value 1
bright_image=0;
%Magnification of image enhancement
mag=20;
%Specify the border of image section to be analyzed in horizontal (x) direction or verical (y) direction
%Assumes, that SEM image labeling is superimposed in the bottom half.
dim_x=1024;
dim_y=768;
%Select the edge detection method (Set edge_method "1" for Sobel or "2" for Canny or "3" for Thresholding by Otsu)
edge_method=2;
%Select shape analysis by setting the variable "1", otherwise, give it value "0"
shape_option=0;
%Select geometric overlay method by setting variable "1", otherwise give it different value
shape_geometry=0;
```

```
%Select value "1", if shape analysis statistics for single images are wanted, set it "0" otherwise
single_image_shape=0;
%Initialization of structuring elements
SE_tophat=strel("disk",10,0); %Structuring Element for top-hat Transformation
SE_close=strel("disk",6,0); %Structuring Element for closing after edge detection
SE_bound=strel("disk",3,0); %Thickness of the boundary
%Threshold value for the height of H-max transform
height_th=5;
%Select the value of the LDPE particle sample. It is found in the status bar of SEM images
LDPE=1.4;

switch(LDPE)
%Calculate the minimum and maximum sieve diameter in µm depending on sieve identifier
case{1.1}
  sieve_min=0;
  sieve_max=500;
case{1.2}
  sieve_min=400;
  sieve_max=500;
case{1.3}
  sieve_min=300;
  sieve_max=400;
case{1.4}
  sieve_min=200;
  sieve_max=300;
case{1.5}
  sieve_min=150;
  sieve_max=200;
case{1.6}
  sieve_min=125;
  sieve_max=150;
case{1.7}
  sieve_min=100;
  sieve_max=125;
case{1.8}
  sieve_min=50;
  sieve_max=100;
case{3}
  sieve_min=500;
  sieve_max=900;
case{2.1}
  sieve_min=0;
  sieve_max=300;
case{2.2}
```

```
  sieve_min=200;
  sieve_max=300;
case{2.3}
  sieve_min=150;
  sieve_max=200;
case{2.4}
  sieve_min=125;
  sieve_max=150;
case{2.5}
  sieve_min=100;
  sieve_max=125;
case{2.6}
  sieve_min=50;
  sieve_max=100;
case{2.7}
  sieve_min=25;
  sieve_max=50;
case{4}
  sieve_min=300;
  sieve_max=450;
endswitch

%%%Image Analysis%%%
for k=1:img_amount
    %%Selecting and cropping Image
    img_in=imread(img_idx(k).Name,"PixelRegion",{[1 dim_y],[1 dim_x]});

    %%TopHat-Filtering
    img_tophat_gray=imtophat(img_in,SE_tophat);
    img_tophat=im2double(img_tophat_gray);
##    img_tophat=imtophat(img_in,SE_tophat);
##    figure, imhist(img_tophat), axis "auto y";
    img_adj2=imadjust(img_tophat);
    img_adj=rgb2gray(img_adj2);
##    img_adj=imadjust(img_tophat,[0.05;0.4], [0;1],1);
##    figure, imhist(img_adj), axis "auto y";
##    figure, imshow(img_adj);
##
figure,subplot(221),imshow(img_in),subplot(222),imshow(img_tophat_gray),subplot(223),imshow(img_tophat),subplot(224),imsho
w(img_adj);
##    figure, subplot(131),imshow(img_tophat),subplot(132),imshow(img_adj),subplot(133),imshow(img_adj2);


    %%Edge-Finding
  switch(edge_method)
  case{1}
    %Sobel Edge-Finding algorithm
    img_edge=edge(img_adj,"Sobel");
##    figure,subplot(121),imshow(img_adj),subplot(122),imshow(img_edge);
  case{2}
    %Calculating the threshold via Otsu's method
    thresh=graythresh(img_adj);
    %Canny Edge-Finding algorithm
    img_edge=edge(img_adj,"Canny",0.5*thresh);
##    figure, imshow(img_edge);
##    figure,subplot(121),imshow(img_adj),subplot(122),imshow(img_edge);
  case{3}
    img_edge=im2bw(img_adj,graythresh(img_adj));
##    figure, imshow(img_edge);
  otherwise
    printf("Please enter a valid value for the Edge-Finding method or define another one in the code using your selected value under the variable edge_method");
  endswitch

    %%Closing holes on the edges
    img_closed=imclose(img_edge,SE_close);
##    figure, subplot(121), imshow(img_edge), subplot(122), imshow(img_closed);

    %%Flood filling the enclosed areas. Filling before the watershed can lead to issues with particle agglomerates
    img_filled=imfill(img_closed,"holes");
##    figure, imshow(img_filled);
##    figure, subplot(121), imshow(img_closed), subplot(122), imshow(img_filled);

  %%Calculate the minimum size of particles considered for morphology depending on sieve size
  switch(mag)
  % 80 percent of all particles evaluated possess an fitted ellipse reciprocal aspect ratio (rar) of 1/2 or higher, which is used to define upper thresholds
  case{20}
    %Min size particles have circular cross section the size of the sieve opening
##    th_min_metric=pi*(sieve_min/2)^2;
    th_min_metric=1.05*sieve_min^2;
    %Max size particles possess circular cross sections with cylindric shape and rar 1/3
    th_max_metric=pi*3/4*sieve_max^2;
    %Convert from metric to px size thresholds
    th_min_px=((472/3000)^2)*th_min_metric;
    th_max_px=((472/3000)^2)*th_max_metric;
  case{50}
    %Min size particles have circular cross section the size of the sieve opening
    th_min_metric=1.05*sieve_min^2;
    %Max size particles have elliptical cross sections with rar 1/3, resulting in rectangles on the image
    th_max_metric=pi*3/4*sieve_max^2;
    %Convert from metric to px size thresholds
    th_min_px=((281/600)^2)*th_min_metric;
    th_max_px=((281/600)^2)*th_max_metric;
  case{110}
    %Min size particles have circular cross section the size of the sieve opening
    th_min_metric=pi*(sieve_min/2)^2;
    %Max size particles have elliptical cross sections with rar 1/3, with minor axis being the size of the cross section
    th_max_metric=pi*3/4*sieve_max^2;
    %Convert from metric to px size thresholds
    th_min_px=((309/300)^2)*th_min_metric;
    th_max_px=((309/300)^2)*th_max_metric;
  endswitch



    %%Removal of small particles/noise(2nd argument in bwareaopen enotes the boundary in px²
    img_clear1=bwareaopen(img_filled,th_min_px,4);
##    figure,imshow(img_filled);
##    figure,imshow(img_clear1);

    %%Watershed iteration
    %Calculates the Negative of the inverse of the binary image created by edge detection, filling and noise removal
    distmap=bwdist(~img_clear1);
##    distmap_img=distmap./(max(max(distmap)));
##    figure, imshow(distmap_img);
    img_marker = imextendedmax(distmap,height_th,4);
##    mask2 = imextendedmin(distmap,4,4);
##    figure,imshow(img_marker);
    %Imimposemin makes sure only regional minima at non zero pixels of mask
    minima = imimposemin(-distmap,img_marker);
    img_ridgelines = watershed(minima);
##    figure, imshow(img_ridgelines);
    img_shed=img_clear1;
    %The preprocessed image is getting shed in the places, where L has value 0
    img_shed(img_ridgelines == 0) = 0;
##    figure,imshow(img_shed);

    %%More watershed attempts
##    distmap_test=bwdist(~img_clear1);
##    figure, imshow(distmap_test./max(max(distmap_test)));
##    filter_test=fspecial("gaussian",6,2);
##    distmap_test_filtered=imfilter(-distmap_test,filter_test);
##    figure, imshow(distmap_test_filtered./(min(min(distmap_test_filtered))));
##    img_ridgeline_test=watershed(distmap_test_filtered);
##    figure, imshow(img_ridgeline_test);
##    img_shed_test=img_clear1;
##    img_shed_test(img_ridgeline_test == 0) = 0;
##    figure, imshow(img_shed_test);

   %%Trashy Watershed
##    C=~img_clear1;
##    D=-bwdist(C);
##    distmap_img=D./(min(min(D)));
##    figure, imshow(distmap_img);
##    D(C)=-Inf;
##    L=watershed(D);
##    Wi=label2rgb(L,'hot','w');
##    figure, imshow(Wi);
##    img_shed=img_clear1;
##    img_shed(L==0)=0;
##    figure, imshow(im);

   %%New Watershed segmentation attempt
##    hy = fspecial("sobel");
##    hx = hy';
##    Iy = imfilter(double(img_in), hy, "replicate");
##    Ix = imfilter(double(img_in), hx, "replicate");
##    gradmag = sqrt(Ix.^2 + Iy.^2);
##    SE_test=strel("disk",20,0);
##    Ie=imerode(img_in,SE_test);
##    Iobr=imreconstruct(Ie,img_in);
##    Iobrd=imdilate(Iobr,SE_test);
##    Iobrcbr=imreconstruct(imcomplement(Iobrd),imcomplement(Iobr));
##    Iobrcbr=imcomplement(Iobrcbr);
##    fgm=imregionalmax(Iobrcbr);
##    I2=img_clear1;
##    I2(fgm)=255;
##    se2=strel(ones(5,5));
##    fgm2=imclose(fgm,se2);
##    fgm3=imerode(fgm2,se2);
##    fgm4=bwareaopen(fgm3,20);
##    I3=img_clear1;
##    I3(fgm4)=255;
##    bw=im2bw(Iobrcbr,graythresh(Iobrcbr));
##    D=bwdist(bw);
##    DL=watershed(D);
##    bgm=DL==0;
##    gradmag2 = imimposemin(gradmag, bgm | fgm4);
##    L=watershed(gradmag2);
##    Lrgb=label2rgb(L,'jet','w','shuffle');
##    figure,subplot(121),imshow(Lrgb),subplot(122),imshow(img_in),
##    hold on
##    himage=imshow(Lrgb);
##    set(himage,'AlphaData',0.3);
##    figure,
subplot(231),imshow(Iobr),subplot(232),imshow(Iobrcbr),subplot(233),imshow(I2),subplot(234),imshow(I3),subplot(235),imshow(b
w),subplot(236),imshow(bgm);


  %%Clearing the particles touching the borders(earlier application leads to removal of overlapping objects)
  img_clear=imclearborder(img_shed,4);
##
figure,subplot(231),imshow(img_in),subplot(232),imshow(img_edge),subplot(233),imshow(img_closed),subplot(234),imshow(img_f
illed),subplot(235),imshow(img_clear);
##    figure, imshow(img_clear);

  %Codepart for troubleshooting with filter model assumptions
##    img_bound_help=im2bw(img_clear-imerode(img_clear,SE_bound));
####    figure, imshow(img_bound_help);
##    img_overlay_help=img_in;
##    if bright_image==1
##      img_overlay_help(img_bound_help==1)=0;
##      figure, imshow(img_overlay_help);
##    else
##      img_overlay_help(img_bound_help==1)=255;
##      figure, imshow(img_overlay_help);
##    end

  %%Removal of small particles and noise after watershed and border clearing
  %Remove particles belonging in a smaller sieve fraction and oversegmented particles
  img_clear2=bwareaopen(img_clear,th_min_px,4);
##    figure, imshow(img_clear2);
  %Retain particle agglomerations and particles belonging in a higher sieve fraction
  img_clear3=bwareaopen(img_clear,th_max_px,4);
##    figure, imshow(img_clear3);
  %Retain only particles which belong in the selected sieve fraction
  img_clear4=im2bw(img_clear2-img_clear3);
##    figure, imshow(img_clear4);
##    figure, subplot(221), imshow(img_clear1), subplot(222), imshow(img_clear2), subplot(223), imshow(img_clear3), subplot(224),
imshow(img_clear4);
```

```
  %%Calculating the boundary of the particles
  img_bound=im2bw(img_clear4-imerode(img_clear4,SE_bound));
##  subplot(236),imshow(img_bound);
##  figure, imshow(img_bound);

  %%Superimpose the boundary of the particles on top of the original image
  img_overlay=img_in;
  if bright_image==1
    img_overlay(img_bound==1)=0;
##    figure, imshow(img_overlay);
##    figure, subplot(121), imshow(img_filled), subplot(122), imshow(img_overlay);
  else
    img_overlay(img_bound==1)=255;
##    figure, imshow(img_overlay);
##    figure, subplot(121), imshow(img_filled), subplot(122), imshow(img_overlay);
  end

  %%Generating statistics
  switch(shape_option)
  case{0}
  stats = regionprops(img_clear4,"all");
  sizeA = [stats.Area]; %size of regions in px²
  num_regions = length(stats); %number of regions

  case{1}
  stats = regionprops(img_clear4,"all");
  sizeA = [stats.Area]; %size of regions in px²
  majX = [stats.MajorAxisLength]; %length of the major axis of fitted ellipse
  minX = [stats.MinorAxisLength]; %length of the minor axis of fitted ellipse
  num_regions = length(stats); %number of regions

  %%Iteration that calculates the Max-Feret-Diameter and Bounding-Box of each particle by using the convex hull. Brute force
  bb_area = zeros(1,num_regions); %Vector that contains the Bounding-Box size for each particle
  maxferetdiam = zeros(1,num_regions); %Vector that contains the Max Feret Diameter for each particle
  for i=1:num_regions
    bb_info = stats(i).BoundingBox; %Read out the Bounding-Box info for the i-th particle
    bb_area(i) = bb_info(3)*bb_info(4); %Multiply the length with the width of the bounding box
    particle_convexhull = stats(i).ConvexHull; %Read out the convex hull coordinates of the i-th particle
    dist_vert = distancePoints(particle_convexhull, particle_convexhull); %Calculate the euclidian distance in px for each vertex of the convex hull of the i-th particle
    maxferetdiam(i) = max(max(dist_vert)); %Write the maximum entry of the distance Matrix for the i-th particle into the corresponding vector
  endfor
  %%Calculating shape parameters
  %(reciprocal axis ratio RAR)
  rar = minX./majX;
  %(rectangularity RTY)
  rty = sizeA./bb_area;
  %(feret major axis ratio FMR)
  fmr = maxferetdiam./majX;

  if shape_geometry == 1
   %%Shape analysis
   num_tri=num_ell=num_rec=num_und=zeros(1,num_regions);
   num_total=zeros(1,4); %Vector that's utilized to save the total count of all shapes identified
   for j=1:num_regions
    if (rty(j) <= 0.6 && fmr(j) >= 1.27)
     num_tri(j)=1;
    else
     num_tri(j)=0;
    endif
    if ((rar(j) >= 0.98 && rty(j) >= 0.98) || (rar(j) >= 0.97 && fmr(j) >= 1.2) || (rar(j) < 0.98 && rty(j) >= 0.98) || (fmr(j) < 0.98 || fmr(j) > 1.09))
     num_rec(j)=1;
    else
     num_rec(j)=0;
    endif
    if ((rar(j) >= 0.99 && (fmr(j) > 0.995 && fmr(j) <= 1.15)) || (fmr(j) >= 0.98 && fmr(j) <= 1.09))
     num_ell(j)=1;
    else
     num_ell(j)=0;
    endif
    %If Shape parameters overlap and don't allow clear identification, a particle is classified as undecided/inconclusive
    if ((num_tri(j)==1 && num_rec(j)==1) || (num_tri(j)==1&&num_ell(j)==1) || (num_rec(j)==1&&num_ell(j)==1))
     num_tri(j)=num_rec(j)=num_ell(j)=0;
     num_und(j)=1;
    endif
   endfor
   num_total(1)=sum(num_ell);
   num_total(2)=sum(num_rec);
   num_total(3)=sum(num_tri);
   num_total(4)=sum(num_und);

   %%Histogram for particle shape distribution
   figure
   bar(num_total,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
```

```
xlabel('Shape');
ylabel('Count');
xticks(1:4);
xticklabels({'Ellipse','Rectangle','Triangle','Inconclusive'});
name_shape_hist=sprintf(['picture_',num2str(mag),'_',num2str(k),'_shape_histogram','.png']);
saveas(1,name_shape_hist);
close all;

%%Superimpose the detected ellipse or rectangle over the outline of the detected particle
figure
imshow(img_bound)

t = linspace(0,2*pi,100);
hold on
for i = 1:num_regions
 if (num_ell(i)==1)
  a = stats(i).MajorAxisLength/2;
  b = stats(i).MinorAxisLength/2;
  Xc = stats(i).Centroid(1);
  Yc = stats(i).Centroid(2);
  phi = deg2rad(-stats(i).Orientation);
  x = Xc + a*cos(t)*cos(phi) - b*sin(t)*sin(phi);
  y = Yc + a*cos(t)*sin(phi) + b*sin(t)*cos(phi);
  plot(x,y,'r','LineWidth',0.5)
 endif
 if (num_rec(i)==1)
  le = stats(i).MajorAxisLength/2*0.886227;
  wi = stats(i).MinorAxisLength/2*0.886227;
  Xc = stats(i).Centroid(1);
  Yc = stats(i).Centroid(2);
  center = transpose(stats(i).Centroid);
  theta = deg2rad(stats(i).Orientation);
  coords = [Xc-le Xc-le Xc+le Xc+le;...
       Yc-wi Yc+wi Yc+wi Yc-wi];
  R = [cos(theta) sin(theta);...
    -sin(theta) cos(theta)];
  rot_coords = R*(coords-repmat(center,[1 4]))+repmat(center,[1 4]);
  rot_coords(:,5)=rot_coords(:,1);
  line(rot_coords(1,:),rot_coords(2,:),'color','g','LineWidth',0.5);
 endif
endfor
name_overlay_shape=sprintf(['picture_',num2str(mag),'_',num2str(k),'_shape_overlay','.png']);
saveas(1,name_overlay_shape);
hold off
```

```
  close all;
endif

%%Calculating Convexity and Solidity of the particle to describe deviation from a convex cross section
solidity_particle=convexity_particle=perim_cvxpoly=zeros(1,num_regions);
for i=1:num_regions
  solidity_particle(i)=stats(i).Solidity;
  perim_cvxpoly(i)=polygonLength(stats(i).ConvexHull);
  convexity_particle(i)=perim_cvxpoly(i)./stats(i).Perimeter;
endfor

%%Global shape parameter distribution
if length_old==0
  rar_out=rar;
  fmr_out=fmr;
  cvx_out=convexity_particle;
  sol_out=solidity_particle;
  length_old=num_regions;
else
  %Save prior shape parameters in help array
  rar_old=rar_out;
  fmr_old=fmr_out;
  cvx_old=cvx_out;
  sol_old=sol_out;
  %Determine length of new shape parameter array
  length_new=length_old+num_regions;
  %Initialize new output arrays
  rar_out=fmr_out=sol_out=cvx_out=zeros(1,length_new);
  for i=1:length_old
    rar_out(i)=rar_old(i);
    fmr_out(i)=fmr_old(i);
    sol_out(i)=sol_old(i);
    cvx_out(i)=cvx_old(i);
  endfor
  for i=(length_old+1):(length_new)
    rar_out(i)=rar(i-length_old);
    fmr_out(i)=fmr(i-length_old);
    sol_out(i)=solidity_particle(i-length_old);
    cvx_out(i)=convexity_particle(i-length_old);
  endfor
  length_old=length(cvx_out);
end
```

```
  %%Plotting Convexity and Solidity
  if single_image_shape==1
   [count_solid,bincenter_solid]=hist(solidity_particle,3);
   figure
   bar(bincenter_solid,count_solid,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
   xlabel('Solidity A_{Particle}/A_{Convexhull}');
   ylabel ('Count');
   name_hist_solidity=sprintf(['picture_',num2str(mag),'_',num2str(k),'_solidity_histogram','.png']);
   saveas(1,name_hist_solidity);
   close all;

   [count_convex,bincenter_convex]=hist(convexity_particle,3);
   figure
   bar(bincenter_convex,count_convex,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
   xlabel('Convexity P_{Convexhull}/P_{Particle}');
   ylabel ('Count');
   name_hist_convexity=sprintf(['picture_',num2str(mag),'_',num2str(k),'_convexity_histogram','.png']);
   saveas(1,name_hist_convexity);
   close all;
  endif

  otherwise
   printf("The currently selected value for shape_option is not implemented. Please redo with supported shape_option values or add a new case");
  endswitch



  %%The following section calculates corresponding metric areas in [µm²] from digital unit in [px²]
  %%Tutorial for adding new magnification levels: Copy a case, select the desired magnification level
  %%in the argument of case{}. Afterwards use GIMP or a comparable software to measure the distance scale
  %%in px. Measure from the left pixel of the left border of the left end towards the left border of the
  %%right end. The "Zähler" of the fraction of the variable sizeA_metric describes the length of the scale in
  %%[µm], the "Nenner" contains the measured length of the scale as described above in [px]

  %%Hier werden zusammenhängende Pixelflächen[px²] in metrische Flächen [µm²] umgewandelt.
  %%Zur Erweiterung einen case kopieren, die gewünschte Vergrößerung angeben und mit
  %%Bildbearbeitungssoftware (z.B. Gimp) mit Lineal-Funktion die Skala zwischen den jeweils linken Balkenenden vermessen.
  %%Der Zähler des Bruchs in sizeA_metric beschreibt die Skalalänge in µm, der Nenner die Pixelanzahl.
  switch(mag)
  case{110}
   sizeA_metric=((300/309)^2)*sizeA;
   maxparticle=max(sizeA_metric);
   [count,bincenter]=hist(sizeA_metric,5);
   figure
```

```
 bar(bincenter,count,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
 xlabel('Area [µm²]');
 ylabel ('Particles');
 name_hist=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram','.png']);
 saveas(1,name_hist);
 close all;

 %%Global size statistics generation
 %First image
 if length_part_old==0
  size_out=sizeA_metric;
  length_part_old=num_regions;
 %Subsequent images
 else
  %Save prior particle sizes in help array
  size_old=size_out;
  %Determine length of the new size output array
  length_part_new=length_part_old+num_regions;
  size_out=zeros(1,length_part_new);
  for i=1:length_part_old
   %Save old particle sizes in new array
   size_out(i)=size_old(i);
  endfor
  for i=(length_part_old+1):(length_part_new)
   %Add particle sizes from current image
   size_out(i)=sizeA_metric(i-length_part_old);
  endfor
  %Adjust the length of the array required to save the particle sizes
  length_part_old=length(size_out);
 end

if single_image_shape==1
 %Histogram for shape specific subplots
 sizeA_ell=zeros(1,num_total(1));
 sizeA_rec=zeros(1,num_total(2));
 sizeA_tri=zeros(1,num_total(3));
 sizeA_und=zeros(1,num_total(4));
 m=n=o=p=1;
 for i=1:num_regions
  if num_ell(i)==1
   sizeA_ell(m)=sizeA_metric(i);
   m++;
  endif
  if num_rec(i)==1
```

```
    sizeA_rec(n)=sizeA_metric(i);
    n++;
  endif
  if num_tri(i)==1
    sizeA_tri(o)=sizeA_metric(i);
    o++;
  endif
  if num_und(i)==1
    sizeA_und(p)=sizeA_metric(i);
    p++;
  endif
endfor

if num_total(1) ~= 0
  [cnt_ell,center_ell]=hist(sizeA_ell,3);
  figure
  bar(center_ell,cnt_ell,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [µm²]');
  ylabel ('Particles');
  name_hist_ell=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_ellipse','.png']);
  saveas(1,name_hist_ell);
  close all;
endif
if num_total(2) ~= 0
  [cnt_rec,center_rec]=hist(sizeA_rec,3);
  figure
  bar(center_rec,cnt_rec,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [µm²]');
  ylabel ('Particles');
  name_hist_rec=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_rectangle','.png']);
  saveas(1,name_hist_rec);
  close all;
endif
if num_total(3) ~= 0
  [cnt_tri,center_tri]=hist(sizeA_tri,3);
  figure
  bar(center_tri,cnt_tri,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [µm²]');
  ylabel ('Particles');
  name_hist_tri=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_triangle','.png']);
  saveas(1,name_hist_tri);
  close all;
endif
if num_total(4) ~= 0
```

```
    [cnt_und,center_und]=hist(sizeA_und,3);
    figure
    bar(center_und,cnt_und,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [µm²]');
    ylabel ('Particles');
    name_hist_und=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_undecided','.png']);
    saveas(1,name_hist_und);
    close all;
  endif

endif

case{221}
  sizeA_metric=((100/207)^2)*sizeA;
  maxparticle=max(sizeA_metric);
  [count,bincenter]=hist(sizeA_metric,5);
  figure
  bar(bincenter,count,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [µm²]');
  ylabel ('Particles');
  name_hist=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram','.png']);
  saveas(1,name_hist);
  close all;

  %%Global size statistics generation
  %First image
  if length_part_old==0
    size_out=sizeA_metric;
    length_part_old=num_regions;
  %Subsequent images
  else
    %Save prior particle sizes in help array
    size_old=size_out;
    %Determine length of the new size output array
    length_part_new=length_part_old+num_regions;
    size_out=zeros(1,length_part_new);
    for i=1:length_part_old
      %Save old particle sizes in new array
      size_out(i)=size_old(i);
    endfor
    for i=(length_part_old+1):(length_part_new)
      %Add particle sizes from current image
      size_out(i)=sizeA_metric(i-length_part_old);
    endfor
  %Adjust the length of the array required to save the particle sizes
  length_part_old=length(size_out);
 end

if single_image_shape==1
 %Histogram for shape specific subplots
 sizeA_ell=zeros(1,num_total(1));
 sizeA_rec=zeros(1,num_total(2));
 sizeA_tri=zeros(1,num_total(3));
 sizeA_und=zeros(1,num_total(4));
 m=n=o=p=1;
 for i=1:num_regions
  if num_ell(i)==1
   sizeA_ell(m)=sizeA_metric(i);
   m++;
  endif
  if num_rec(i)==1
   sizeA_rec(n)=sizeA_metric(i);
   n++;
  endif
  if num_tri(i)==1
   sizeA_tri(o)=sizeA_metric(i);
   o++;
  endif
  if num_und(i)==1
   sizeA_und(p)=sizeA_metric(i);
   p++;
  endif
 endfor

 if num_total(1) ~= 0
  [cnt_ell,center_ell]=hist(sizeA_ell,3);
  figure
  bar(center_ell,cnt_ell,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [µm²]');
  ylabel ('Particles');
  name_hist_ell=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_ellipse','.png']);
  saveas(1,name_hist_ell);
  close all;
 endif
 if num_total(2) ~= 0
  [cnt_rec,center_rec]=hist(sizeA_rec,3);
  figure
  bar(center_rec,cnt_rec,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
```

```
    xlabel('Area [µm²]');
    ylabel ('Particles');
    name_hist_rec=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_rectangle','.png']);
    saveas(1,name_hist_rec);
    close all;
  endif
  if num_total(3) ~= 0
    [cnt_tri,center_tri]=hist(sizeA_tri,3);
    figure
    bar(center_tri,cnt_tri,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [µm²]');
    ylabel ('Particles');
    name_hist_tri=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_triangle','.png']);
    saveas(1,name_hist_tri);
    close all;
  endif
  if num_total(4) ~= 0
    [cnt_und,center_und]=hist(sizeA_und,3);
    figure
    bar(center_und,cnt_und,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [µm²]');
    ylabel ('Particles');
    name_hist_und=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_undecided','.png']);
    saveas(1,name_hist_und);
    close all;
  endif

endif

case{50}
  sizeA_metric=((600/281)^2)*sizeA;
  maxparticle=max(sizeA_metric);
  [count,bincenter]=hist(sizeA_metric,5);
  figure
  bar(bincenter,count,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [µm²]');
  ylabel ('Particles');
  name_hist=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram','.png']);
  saveas(1,name_hist);
  close all;

  %%Global size statistics generation
  %First image
  if length_part_old==0
```

```
    size_out=sizeA_metric;
    length_part_old=num_regions;
  %Subsequent images
  else
    %Save prior particle sizes in help array
    size_old=size_out;
    %Determine length of the new size output array
    length_part_new=length_part_old+num_regions;
    size_out=zeros(1,length_part_new);
    for i=1:length_part_old
      %Save old particle sizes in new array
      size_out(i)=size_old(i);
    endfor
    for i=(length_part_old+1):(length_part_new)
      %Add particle sizes from current image
      size_out(i)=sizeA_metric(i-length_part_old);
    endfor
    %Adjust the length of the array required to save the particle sizes
    length_part_old=length(size_out);
  end

if single_image_shape==1
  %Histogram for shape specific subplots
  sizeA_ell=zeros(1,num_total(1));
  sizeA_rec=zeros(1,num_total(2));
  sizeA_tri=zeros(1,num_total(3));
  sizeA_und=zeros(1,num_total(4));
  m=n=o=p=1;
  for i=1:num_regions
    if num_ell(i)==1
      sizeA_ell(m)=sizeA_metric(i);
      m++;
    endif
    if num_rec(i)==1
      sizeA_rec(n)=sizeA_metric(i);
      n++;
    endif
    if num_tri(i)==1
      sizeA_tri(o)=sizeA_metric(i);
      o++;
    endif
    if num_und(i)==1
      sizeA_und(p)=sizeA_metric(i);
      p++;
    endif
  endfor

  if num_total(1) ~= 0
    [cnt_ell,center_ell]=hist(sizeA_ell,3);
    figure
    bar(center_ell,cnt_ell,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [µm²]');
    ylabel ('Particles');
    name_hist_ell=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_ellipse','.png']);
    saveas(1,name_hist_ell);
    close all;
  endif
  if num_total(2) ~= 0
    [cnt_rec,center_rec]=hist(sizeA_rec,3);
    figure
    bar(center_rec,cnt_rec,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [µm²]');
    ylabel ('Particles');
    name_hist_rec=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_rectangle','.png']);
    saveas(1,name_hist_rec);
    close all;
  endif
  if num_total(3) ~= 0
    [cnt_tri,center_tri]=hist(sizeA_tri,3);
    figure
    bar(center_tri,cnt_tri,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [µm²]');
    ylabel ('Particles');
    name_hist_tri=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_triangle','.png']);
    saveas(1,name_hist_tri);
    close all;
  endif
  if num_total(4) ~= 0
    [cnt_und,center_und]=hist(sizeA_und,3);
    figure
    bar(center_und,cnt_und,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [µm²]');
    ylabel ('Particles');
    name_hist_und=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_undecided','.png']);
    saveas(1,name_hist_und);
    close all;
  endif
```

```
endif

case{20}
  sizeA_metric=((3000/472)^2)*sizeA;
  maxparticle=max(sizeA_metric);
  [count,bincenter]=hist(sizeA_metric,5);
  figure
  bar(bincenter,count,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [µm²]');
  ylabel ('Particles');
  name_hist=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram','.png']);
  saveas(1,name_hist);
  close all;

  %%Global size statistics generation
  %First image
  if length_part_old==0
    size_out=sizeA_metric;
    length_part_old=num_regions;
  %Subsequent images
  else
    %Save prior particle sizes in help array
    size_old=size_out;
    %Determine length of the new size output array
    length_part_new=length_part_old+num_regions;
    size_out=zeros(1,length_part_new);
    for i=1:length_part_old
      %Save old particle sizes in new array
      size_out(i)=size_old(i);
    endfor
    for i=(length_part_old+1):(length_part_new)
      %Add particle sizes from current image
      size_out(i)=sizeA_metric(i-length_part_old);
    endfor
    %Adjust the length of the array required to save the particle sizes
    length_part_old=length(size_out);
  end

if single_image_shape==1
  %Histogram for shape specific subplots
  sizeA_ell=zeros(1,num_total(1));
  sizeA_rec=zeros(1,num_total(2));
  sizeA_tri=zeros(1,num_total(3));
  sizeA_und=zeros(1,num_total(4));
```

```
m=n=o=p=1;
for i=1:num_regions
  if num_ell(i)==1
    sizeA_ell(m)=sizeA_metric(i);
    m++;
  endif
  if num_rec(i)==1
    sizeA_rec(n)=sizeA_metric(i);
    n++;
  endif
  if num_tri(i)==1
    sizeA_tri(o)=sizeA_metric(i);
    o++;
  endif
  if num_und(i)==1
    sizeA_und(p)=sizeA_metric(i);
    p++;
  endif
endfor

if num_total(1) ~= 0
  [cnt_ell,center_ell]=hist(sizeA_ell,3);
  figure
  bar(center_ell,cnt_ell,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [μm²]');
  ylabel ('Particles');
  name_hist_ell=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_ellipse','.png']);
  saveas(1,name_hist_ell);
  close all;
endif
if num_total(2) ~= 0
  [cnt_rec,center_rec]=hist(sizeA_rec,3);
  figure
  bar(center_rec,cnt_rec,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
  xlabel('Area [μm²]');
  ylabel ('Particles');
  name_hist_rec=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_rectangle','.png']);
  saveas(1,name_hist_rec);
  close all;
endif
if num_total(3) ~= 0
  [cnt_tri,center_tri]=hist(sizeA_tri,3);
  figure
  bar(center_tri,cnt_tri,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
```

```
    xlabel('Area [µm²]');
    ylabel ('Particles');
    name_hist_tri=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_triangle','.png']);
    saveas(1,name_hist_tri);
    close all;
   endif
   if num_total(4) ~= 0
    [cnt_und,center_und]=hist(sizeA_und,3);
    figure
    bar(center_und,cnt_und,'FaceColor','r','EdgeColor','k','LineWidth',2,"hist");
    xlabel('Area [µm²]');
    ylabel ('Particles');
    name_hist_und=sprintf(['picture_',num2str(mag),'_',num2str(k),'_size_histogram_undecided','.png']);
    saveas(1,name_hist_und);
    close all;
   endif

  endif


  otherwise
   printf("Area calculation isn't yet implemented for the selected magnification. Please redo with supported magnification or add
code for the required magnification by calculating area of 1px");
  endswitch

## %Saving the B/W image of particle area and the perimeter of the particles superimposed on top of the original image
  name_img=sprintf(['picture_',num2str(mag),'_',num2str(k),'.png']);
  imwrite(img_clear4,name_img);
  name_overlay=sprintf(['picture_',num2str(mag),'_',num2str(k),'_overlay','.png']);
  imwrite(img_overlay,name_overlay);

endfor

%%Returning graphs containing information about particle size distribution over a series of images
 [count_size_global,bincenter_size_global]=hist(size_out,40);
 figure
 bar(bincenter_size_global,count_size_global,'FaceColor','r',"hist");
 xlabel('Area [µm²]');
 ylabel ('Count');
 name_hist_size_global=sprintf(['picture_',num2str(mag),'_size_global','.png']);
 saveas(1,name_hist_size_global);
 close all;
 csvwrite("size_global.csv",reshape(size_out,length_part_old,1));

%%The following function returns global statistics on shape parameters provided it is enabled
```

```
if (length_old > 0)
  [count_sol_global,bincenter_sol_global]=hist(sol_out,20);
  figure
  bar(bincenter_sol_global,count_sol_global,'FaceColor','r',"hist");
  xlabel('Solidity A_{Particle}/A_{Convexhull}');
  ylabel ('Count');
  name_hist_solidity_global=sprintf(['picture_',num2str(mag),'_solidity_global','.png']);
  saveas(1,name_hist_solidity_global);
  close all;
  csvwrite("solidity_global.csv",reshape(sol_out,length_part_old,1));

  [count_cvx_global,bincenter_cvx_global]=hist(cvx_out,20);
  figure
  bar(bincenter_cvx_global,count_cvx_global,'FaceColor','r',"hist");
  xlabel('Convexity P_{Convexhull}/P_{Particle}');
  ylabel ('Count');
  name_hist_convexity_global=sprintf(['picture_',num2str(mag),'_convexity_global','.png']);
  saveas(1,name_hist_convexity_global);
  close all;
  csvwrite("convexity_global.csv",reshape(cvx_out,length_part_old,1));

  [count_rar_global,bincenter_rar_global]=hist(rar_out,20);
  figure
  bar(bincenter_rar_global,count_rar_global,'FaceColor','r',"hist");
  xlabel('Reciprocal aspect ratio length(fitted minor axis)/length(fitted major axis)');
  ylabel ('Count');
  name_hist_rar_global=sprintf(['picture_',num2str(mag),'_rar_global','.png']);
  saveas(1,name_hist_rar_global);
  close all;
  csvwrite("reciprocalaspectratio_global.csv",reshape(rar_out,length_part_old,1));

  [count_fmr_global,bincenter_fmr_global]=hist(fmr_out,20);
  figure
  bar(bincenter_fmr_global,count_fmr_global,'FaceColor','r',"hist");
  xlabel('Feret major axis ratio length(feret major axis)/length(fitted major axis)');
  ylabel ('Count');
  name_hist_fmr_global=sprintf(['picture_',num2str(mag),'_fmr_global','.png']);
  saveas(1,name_hist_fmr_global);
  close all;
  csvwrite("feretmajoraxisratio_global.csv",reshape(fmr_out,length_part_old,1));
endif
```

SI2: Global thresholding versus edge detection

Large particles split in multiple, inaccurate small particles for global thresholding (Figure S1a) and yielded better results using edge detection (Figure S1b).

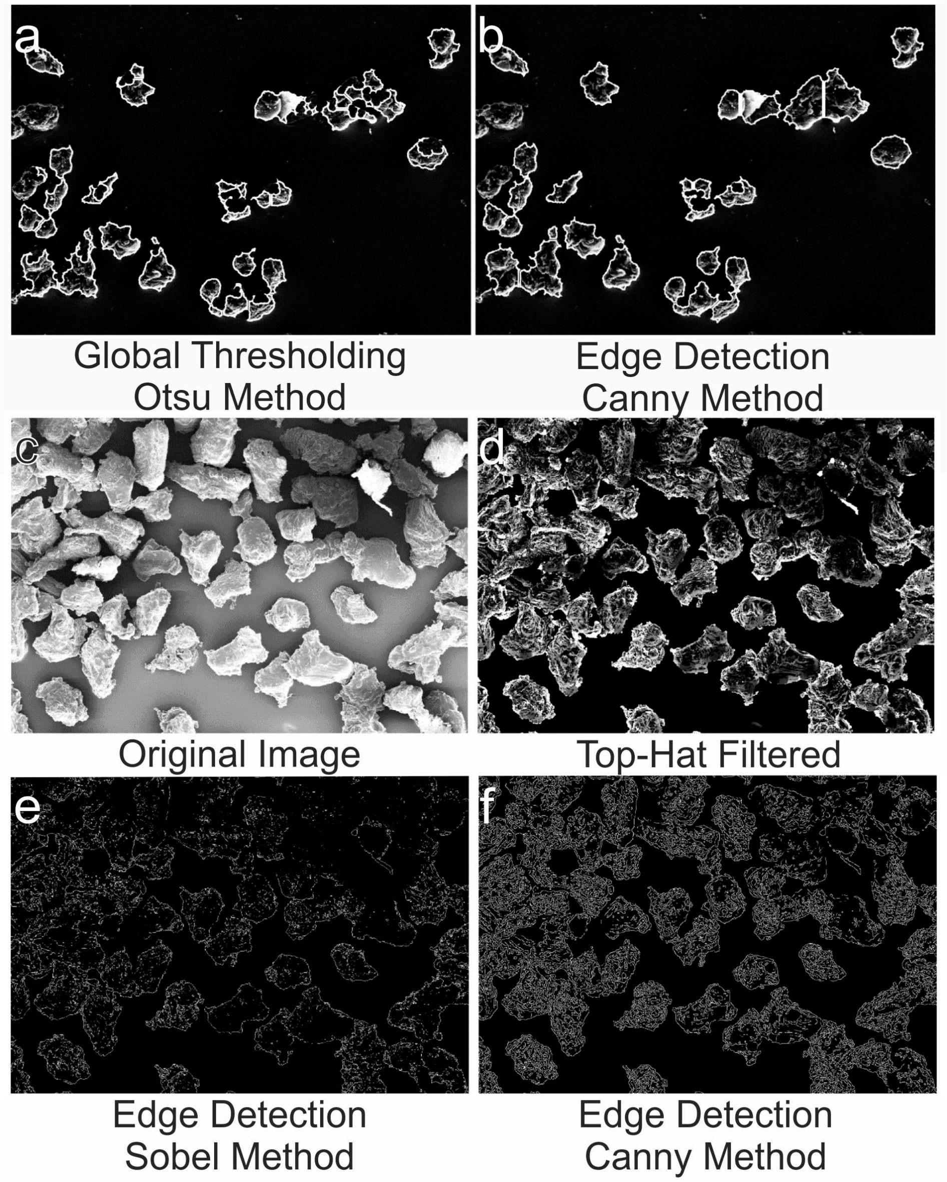

Figure S1: Original SEM-image (**c**), top-hat-filtered and histogram-stretched (**d**), after Sobel edge detection (**e**) and after Canny edge detection (**f**).

A comparison between the two edge detection methods *Sobel* and *Canny* shows that no method is a clear winner. *Canny* detected more edges in areas of high and medium contrast (Figure S1f), while *Sobel* can completely miss darker objects near a bright object (Figure S1e). The edges identified by *Canny* were generally not closed and require further DIP steps for their closure (see Article, Figure 2g-i).

SI3: Evaluation of “best” setting

Under the specified settings (Table S2, Table S3: columns 1-3), a manual count of all incorrectly detected particles of two pilot images (Figure 7g,h) was performed to find the ”best” setting for running an image series. The high number of settings were narrowed down to 13 respectable settings (Table S1, Table S2) and the first four were tested on an image series. LDPE 1.4 are particles of the sieve fraction 200-300 µm (measuring bar 1 mm) and LDPE 1.8 of the fraction 50-100 µm (0.6 mm).

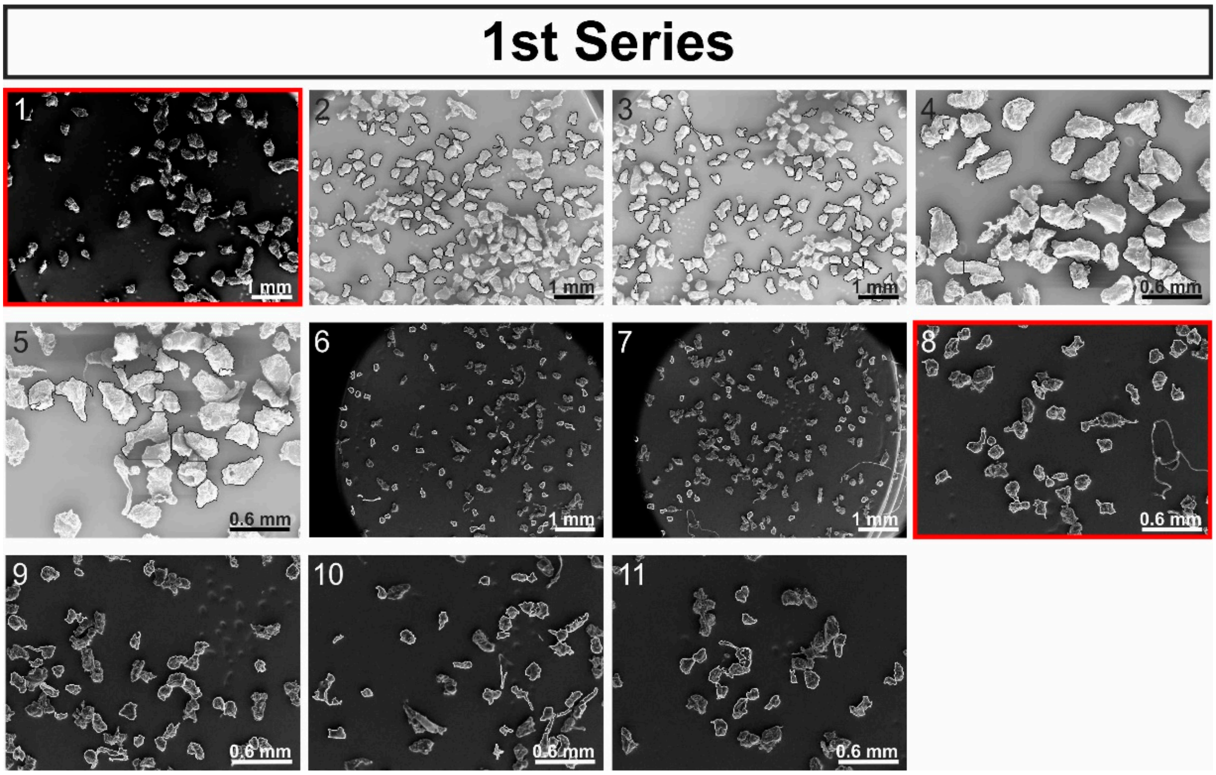

Figure S2: Overlay of input and output images of the 1st image series tested, LDPE particles of the sieve fraction 200-300 µm (No. 1-5) and fraction 50-100 µm (No. 6-11) with the two pilot images (No. 1+8, red framed) imaged in MAG 20x (1 mm) and MAG 50x (0.6 mm).

Table S1: Tested settings on the pilot image of fraction 200-300 µm with MAG 20x.

Structuring element size top-hat filter [px]	Structuring element size edge-closing [px]	Segmentation threshold *height_th*	Total particles	Detected particles	Errors
10	4	5	56	46	7
10	4	4	56	48	8
10	6	5	56	42	8
10	6	3	56	46	11
10	5	4	56	46	10
10	6	4	56	43	9
10	5	5	56	44	10
10	5	3	56	48	12
10	4	3	56	50	10
9	4	4	56	47	8
11	4	4	56	49	10
12	4	4	56	51	11
15	4	4	56	49	9

Table S2: Tested settings on the pilot image of fraction 50-100 µm with MAG 50x.

Structuring element size top-hat filter [px]	Structuring element size edge-closing [px]	Segmentation threshold *height_th*	Total particles	Detected particles	Errors
10	4	5	40	30	0
10	4	4	40	30	0
10	6	5	40	30	1
10	6	3	40	34	5
10	5	4	40	30	1
10	6	4	40	31	2
10	5	5	40	30	1
10	5	3	40	35	9
10	4	3	40	35	9
9	4	4	40	29	0
11	4	4	40	29	0
12	4	4	40	30	0
15	4	4	40	31	2

Table S3: Evaluated results using structuring element size 4, morphological closing and segmentation threshold 5.

Identifier (Magnification)	Total particles	Identified particles	Total errors	Identification ratio [%]	Error ratio [%]
LDPE1.4_1(20)	56	46	7	82.1	15.2
LDPE1.4_2(20)	161	81	16	50.3	19.8
LDPE1.4_3(20)	143	87	18	60.8	20.7
LDPE1.4_4(50)	29	21	4	72.4	19.0
LDPE1.4_5(50)	27	14	6	51.9	42.9
LDPE1.8_1(20)	149	57	6	38.3	10.5
LDPE1.8_2(20)	183	44	5	24.0	11.4
LDPE1.8_3(50)	40	30	0	75.0	0.0
LDPE1.8_4(50)	58	35	4	60.3	11.4
LDPE1.8_5(50)	40	38	13	95.0	34.2
LDPE1.8_6(50)	33	19	2	57.6	10.5
Sum	919	472	81	51.4	17.2

Table S4: Evaluated results using structuring element size 4, morphological closing and segmentation threshold 4.

Identifier (Magnification)	Total particles	Identified particles	Total errors	Identification ratio [%]	Error ratio [%]
LDPE1.4_1(20)	56	48	8	85.7	16.7
LDPE1.4_2(20)	161	87	21	54.0	24.1
LDPE1.4_3(20)	143	94	21	65.7	22.3
LDPE1.4_4(50)	29	21	5	72.4	23.8
LDPE1.4_5(50)	27	15	6	55.6	40.0
LDPE1.8_1(20)	149	66	8	44.3	12.1
LDPE1.8_2(20)	183	55	5	30.1	9.1
LDPE1.8_3(50)	40	30	0	75.0	0.0
LDPE1.8_4(50)	58	37	4	63.8	10.8
LDPE1.8_5(50)	40	40	17	100.0	42.5
LDPE1.8_6(50)	33	20	2	60.6	10.0
Sum	919	513	97	55.8	18.9

Table S5: Evaluated results using structuring element size 6, morphological closing and segmentation threshold 5.

Identifier (Magnification)	Total particles	Identified particles	Total errors	Identification ratio [%]	Error ratio [%]
LDPE1.4_1(20)	56	42	8	75.0	19.0
LDPE1.4_2(20)	161	60	12	37.3	20.0
LDPE1.4_3(20)	143	72	19	50.3	26.4
LDPE1.4_4(50)	29	23	6	79.3	26.1
LDPE1.4_5(50)	27	14	6	51.9	42.9
LDPE1.8_1(20)	149	51	6	34.2	11.8
LDPE1.8_2(20)	183	37	3	20.2	8.1
LDPE1.8_3(50)	40	30	1	75.0	3.3
LDPE1.8_4(50)	58	29	4	50.0	13.8
LDPE1.8_5(50)	40	33	10	82.5	30.3
LDPE1.8_6(50)	33	18	1	54.5	5.6
Sum	919	409	76	44.5	18.6

Table S6: Evaluated results using structuring element size 6, morphological closing and segmentation threshold 3.

Identifier (Magnification)	Total particles	Identified particles	Total errors	Identification ratio [%]	Error ratio [%]
LDPE1.4_1(20)	56	46	11	82.1	23.9
LDPE1.4_2(20)	161	72	23	44.7	31.9
LDPE1.4_3(20)	143	85	24	59.4	28.2
LDPE1.4_4(50)	29	24	8	82.8	33.3
LDPE1.4_5(50)	27	15	6	55.6	40.0
LDPE1.8_1(20)	149	65	10	43.6	15.4
LDPE1.8_2(20)	183	54	6	29.5	11.1
LDPE1.8_3(50)	40	34	5	85.0	14.7
LDPE1.8_4(50)	58	40	5	69.0	12.5
LDPE1.8_5(50)	40	43	19	107.5	44.2
LDPE1.8_6(50)	33	20	3	60.6	15.0
Sum	919	498	120	54.2	24.1

SI4: Influences of magnification changes

Size measurements of the LDPE particles (No. 1-27) counted in the SEM images (Figure 8) and the determination of the deviation from the magnifications (MAG 50x/110x) to baseline (MAG 20x) are presented in Table S7.

Table S7: Measurement values of magnification (MAG) compatibility analysis.

Particle identifier	Measured particle area [µm²]			Deviation to baseline [%]
	MAG 20 (baseline)	MAG 50	MAG 110	
1	34009	-	27695	-18.6
2	17881	-	14557	-18.6
3	10440	-	7187	-31.2
4	30981	-	21919	-29.3
5	13128	-	12114	-7.7
6	23144	-	19722	-14.8
7	13185	-	9935	-24.6
8	14345	-	11048	-23.0
9	46260	-	35652	-22.9
10	15109	-	11146	-26.2
11	30585	-	23853	-22.0
12	34603	-	23125	-33.2
13	36131	-	29882	-17.3
14	21248	-	16279	-23.4
15	24247	23553	-	-2.9
16	16410	15606	-	-4.9
17	15816	15570	-	-1.6
18	15618	15196	-	-2.7
19	19890	20170	-	+1.4

20	16071	13965	-	-13.1
21	27133	20763	-	-23.5
22	13383	10783	-	-19.4
23	16495	11822	-	-28.3
24	28435	21561	-	-24.2
25	13637	9674.6	-	-29.1
26	25492	19819	-	-22.3
27	13949	10559	-	-24.3

SI5: Second image series

Figure S3 shows the analyzed SEM images of the LDPE particles used for determination of the particle size distributions (Figure 10a). The images of 2nd series in Figure S4 and Figure S5 were used to classify the shape. The shape distributions of all combined sieve fractions of both batches (< 300 μm and < 800 μm) can be seen in Figure 10c-h. The 2nd series of the LDPE particles in Figure S5 were visually inspected after the full-automatic analysis and misidentified particles were counted.

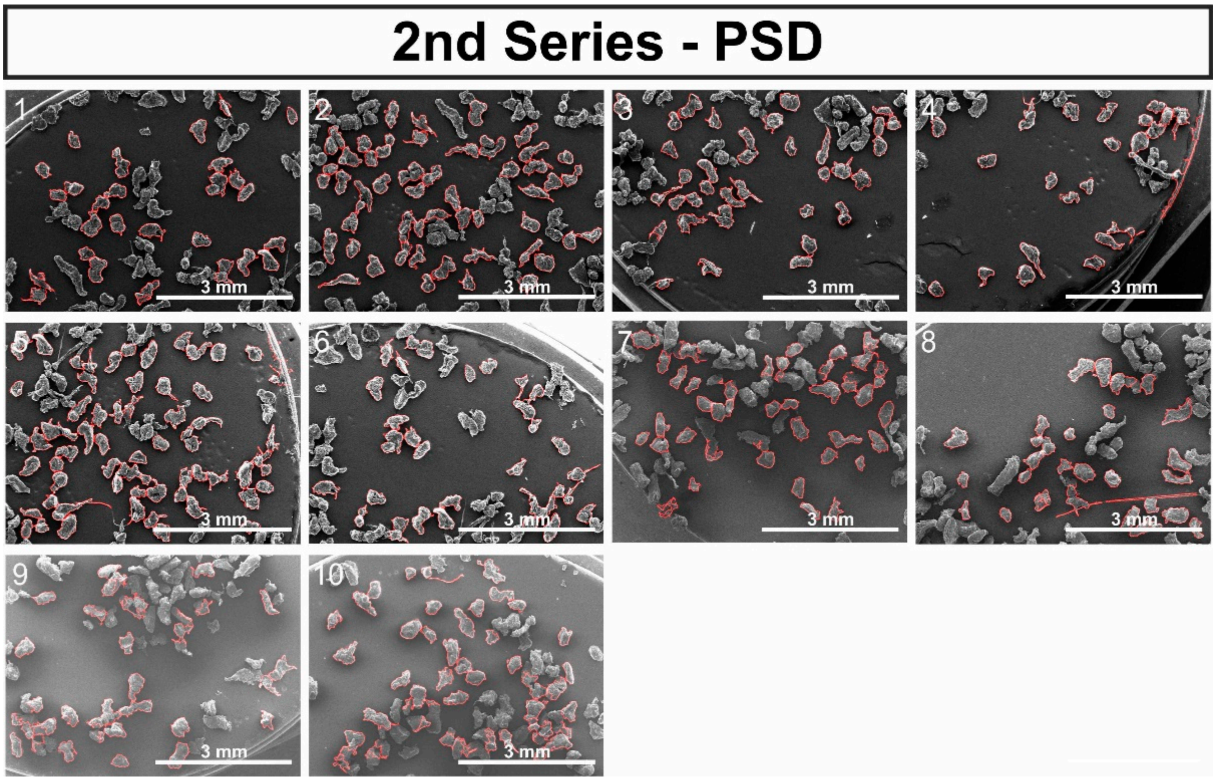

Figure S3: SEM images (No. 1-10) with detected LDPE particles (200-300 μm, red outlined) of 2nd series used for PSD analysis**.**

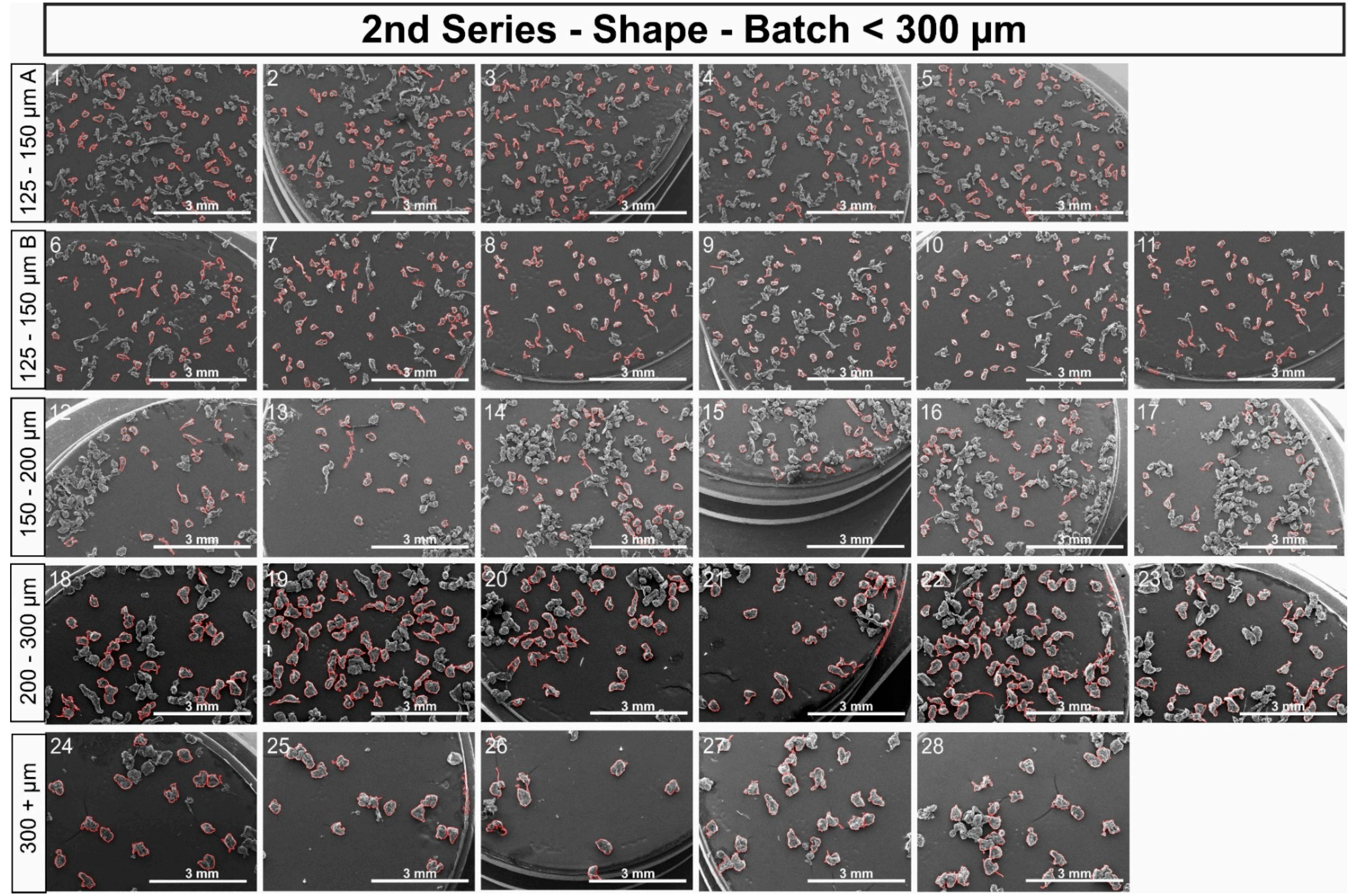

Figure S4: SEM images (No. 1-28) with detected particles (red outlined) of 2nd series from the batch smaller 300 µm used for shape classification.

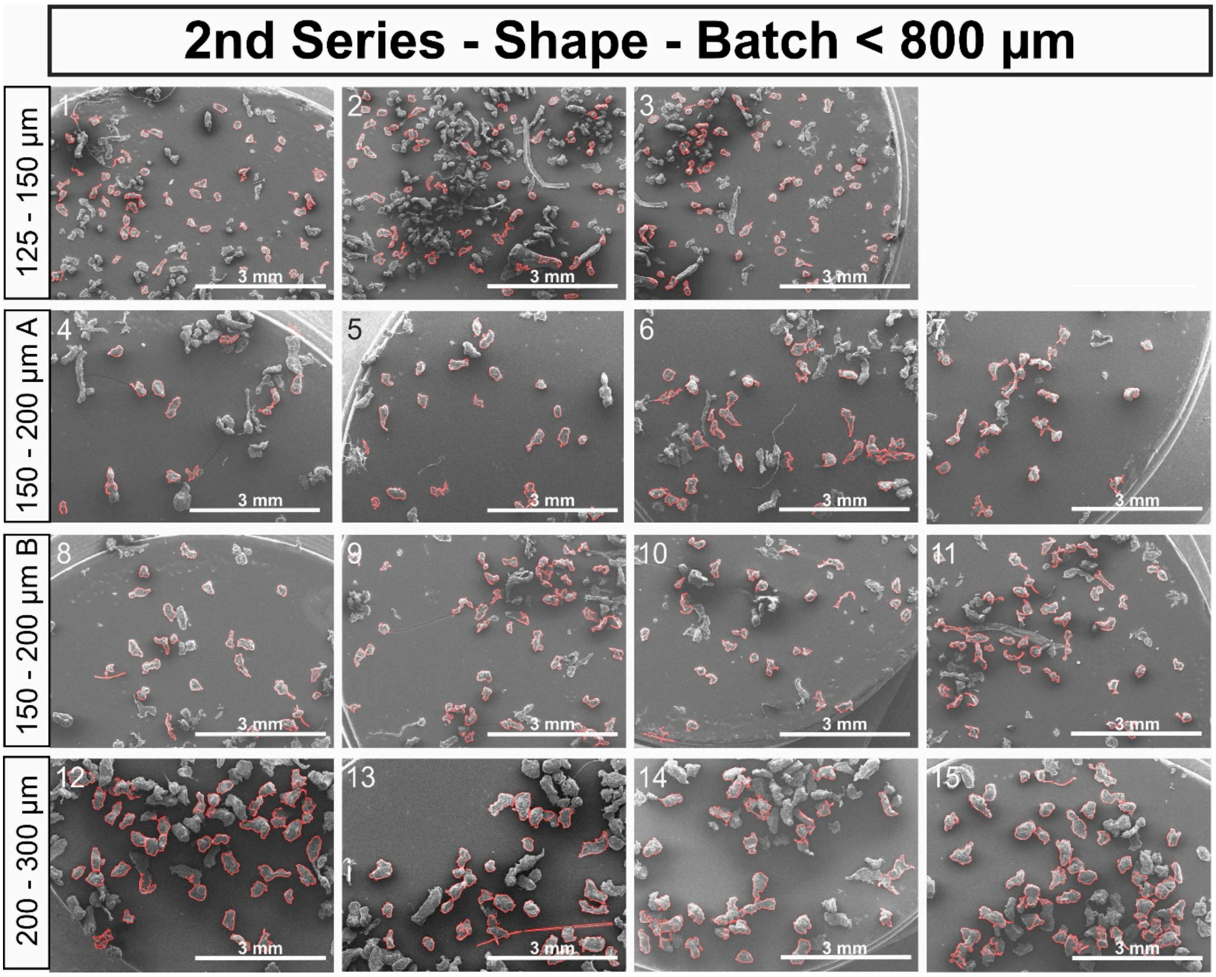

Figure S5: SEM images (No. 1-15) with detected particles (red outlined) of 2nd series from the batch smaller 800 µm used for shape analysis.