# **Appendix: Single-cell image processing code (in MATLAB) with explanations**

1. Main file, for adipocyte cells (cells with lipid droplets):

| *Matlab Code* | *Explanation* |
| --- | --- |
| clc  clear all  close all |  |
| global Num\_of\_cells\_per\_pic  global Cell\_number  global Clear\_Cells\_Selected\_So\_Far  global btn1  global btn1\_disabled | Declaring variables as global:   * Number of cells per image * Cell number * Number of visibly clear cells, selected by the user * Button, to stop selecting clear cells * Controlling if ‘btn1’ is enabled or disabled |
| **%%Data inserted by the user**  im\_RGB1=imread('ImageName.jpg');  Num\_of\_cells\_per\_pic=A;  threshold=B; | * Loading the micrograph image * Number of cells in the image * Threshold value (ranges between 0-1) from red-green-blue (RGB) to grayscale, to produce a binary image, where (lipid droplets) LDs are colored white and everything else is blackened |
| **%% Noise Removal and converting to binary image**  im\_gray=rgb2gray(im\_RGB1);  im\_gray=double(im\_gray);  im\_gray=im\_gray/255;  im\_gray=wiener2(im\_gray); | * The image is converted from RGB to grayscale, eliminating hue and saturation properties, but maintaining luminance properties * Define double precision for the image * Normalize grayscale levels to range between 0 and 1 * Filter image using two-dimensional adaptive Gaussian noise-removal filter (Wiener filter) |
| **%% Region filling and connected component labeling**  Binary\_Image=im2bw(im\_gray,threshold);  Filled\_Image=imfill(Binary\_Image,'holes');  [labeled\_im,num\_of\_objects]=bwlabel(Filled\_Image,4);  RGB\_labeled\_im=label2rgb(labeled\_im); | * Converting to binary image, with the threshold value inserted by the user * Fill artifact cavities in LDs * Connected component labeling, each LD is identified as a “connected component” and uniquely labeled * Convert the labeled image to RGB |
| **%% Lipid droplet area calculations**  area=zeros(1,num\_of\_objects);  for i=1:length(labeled\_im(:))  if labeled\_im(i)~=0  object\_serial\_num=labeled\_im(i);  area(object\_serial\_num)=area(object\_serial\_num)+1;  end  end | * Array that contains areas of each LD * Run over pixels in the labeled image, implementing the following procedure: * If the pixel belongs to one of the LDs: * Save the label of the LD containing the pixel * The pixel is now considered a part of this LD area |
| **%% Labeling each object (lipid droplets) with its area (in pixels)**  labeled\_im\_area=labeled\_im;  for i=1:length(labeled\_im(:))  if labeled\_im(i)~=0  object\_serial\_num=labeled\_im(i);  labeled\_im\_area(i)=area(object\_serial\_num);  end  end | * Array that contains areas of each LD Define the area-labeled image * Run over pixels in the labeled image, implementing the following procedure: * If the pixel belongs to one of the LDs: * Save the label of the LD containing the pixel * The pixel is now associated with the area of the LD it belongs to |
| **%% Deleting connected components with area <5 pixels**  labeled\_im\_area(labeled\_im\_area==1)=0;  labeled\_im\_area(labeled\_im\_area==2)=0;  labeled\_im\_area(labeled\_im\_area==3)=0;  labeled\_im\_area(labeled\_im\_area==4)=0;  labeled\_im(labeled\_im\_area==0)=0; | * Connected components with area less than 5 pixels, considered as noise and are therefore blackened |
| **%% Cells selection by the User**  figure(1)  imshow(im\_RGB1);title('Original Image');  btn1\_disabled=0;  Cell\_number=1;  labeled\_im\_area\_overlay=labeled\_im\_area+im\_gray;  size\_of\_pic=size(labeled\_im\_area);  label\_im\_cell=zeros(size\_of\_pic);  while Cell\_number<=Num\_of\_cells\_per\_pic  label\_im\_cell=[];  LDs\_area=[];  figure(2)  imshow(labeled\_im\_area\_overlay)  title('Please mark the cell') | * Display the original image * As long as ‘btn1’ wasn't pressed, ‘btn1\_disabled’ value is set to be = 0. * Cell counter, cells selected by the user * Creating an overlay area-labeled image, combined from the grayscale image and the binary image, in order for the user to be able to select the exact cell area and to improve lipid droplets visibility * Record the size (in pixels) of the area-labeled image * Define a new image for data storage, with size that equals the size of the original image * As long as the user didn't select all the cells, perform the following operations: * Each time the loop is repeated, these arrays are redefined for data storage * Display the area-labeled image each time the loop is repeated |
| **%% Defining buttons**  % "Stop Selecting Clear Cells" button  if btn1\_disabled==0  btn1=uicontrol('Style','pushbutton','String','Stop Selecting Clear Cells','Position',[20 20 130 30],'Callback', @Stop\_Selecting\_Clear\_cells);  else  btn1=uicontrol('Style','pushbutton','String','Stop Selecting Clear Cells','Position',[20 20 130 30],'Callback', @Stop\_Selecting\_Clear\_cells,'Enable','off');  end  % "How Many Cells Left?" button  btn2=uicontrol('Style','pushbutton','String','How Many Cells Left?','Position',[600 20 130 30],'Callback',@How\_many\_cells\_selected); | * Defining “Stop Selecting Clear Cells” button, for cases where the user finished selecting all the visibly clear cells in the image. This button is enabled as long as the user doesn't click it. * Clearly visible cells are cells with clear margins, clear lipid droplets, and cells that are entirely contained in the field of view. * Defining “How Many Cells Left?” button, for user convenience, when clicking on the button the user sees how many more cells he needs to select (clearly visible and not clearly visible). |
| **%% Cell parameters**  cell=roipoly;  stats\_cell=regionprops('table',cell,'Area','Eccentricity','Perimeter');  Circularity\_cells(Cell\_number)=(4\*pi\*stats\_cell.Area)/(stats\_cell.Perimeter.^2);  Eccentricity\_cells(Cell\_number)=stats\_cell.Eccentricity;  Areas\_cells(Cell\_number)=stats\_cell.Area; | * The user marks the cell, by specifying a polygonal region of interest within the image. Roipoly defines a new binary image, where the pixels marked by the user are assigned a ‘1’ label, and all other pixels are assigned a ‘0’ label * Regionprops: measure properties of the selected cell * Cell circularity calculation * Cell eccentricity calculation * Cell projected area calculation |
| **%% Lipid droplet calculations**  label\_im\_cell(find(cell))=labeled\_im(find(cell));  [y\_ind(Cell\_number),x\_ind(Cell\_number)]=ind2sub(size\_of\_pic,(find(label\_im\_cell,1,'first')));  while max(max(label\_im\_cell))>0  LDs\_area=[LDs\_area labeled\_im\_area(find(label\_im\_cell==max(max(label\_im\_cell)),1,'first'))];  label\_im\_cell(label\_im\_cell==…  max(max(label\_im\_cell)))=0;  end  average\_LD\_area(Cell\_number)=mean(LDs\_area);  num\_of\_LDs\_per\_cell(Cell\_number)= length(LDs\_area);  labeled\_im\_area\_overlay(find(cell))=0;  Cell\_number=Cell\_number+1;  end | * Defining data-storing array with dimensions corresponding to those of the image * Convert the linear index of the array to subscript indices * The following loop repeats until all LDs in the cell are taken into consideration * Array which stores area of the LDs within the cell area marked by the user * Eliminate pixels that have already been considered * A vector which contains the average LD area per cell * A vector which contains the number of LDs per cell * The area of the selected cell is blackened for display purpose * Update the counter ‘Cell\_number’ * The above procedure is repeated until the user selects all the cells |
| **%% Lipid area per field of view calculations**    total\_area=sum(area);  image\_area=length(im\_gray(:));  total\_area\_noise=0;  for i=1:length(labeled\_im\_area\_overlay(:))  if labeled\_im\_area\_overlay(i)>=1;  total\_area\_noise=total\_area\_noise+1;  end  end  new\_total\_clean\_area=total\_area-total\_area\_noise;  area\_ratio=new\_total\_clean\_area/image\_area; | * ‘total\_area’ is the number of white pixels in the image * ‘image\_area’ is the number of pixels in the image * ‘total\_area\_noise’ is the number of white pixels outside of the selected cells, which is not representing LDs * The loop runs over all the pixels in the image (which is now with black area instead of the selected cells) for implementing the following procedure: * Count how many white pixels there are in the image (outside the selected cells only) each white pixel, outside the selected cells is considered as noise. * The above process is repeated for all pixels in the image * ‘new\_total\_clean\_area’ is the total number of white pixels which represent LDs, it's the number of all white pixels in the image, minus the noise white pixels * Calculate the percentage of the image area occupied by LDs |
| **%%** **Processing results for "Number of LDs per cell"**  if btn1\_disabled~=0  num\_of\_LDs\_per\_cell=num\_of\_LDs\_per\_cell(1:Clear\_Cells\_Selected\_So\_Far);  else  Clear\_Cells\_Selected\_So\_Far=Cell\_number-1;  end | * If “Stop Selecting Clear Cells” button was pressed, the vector is cut and unclear cells data is not included, otherwise the vector remains the same |
| **%% Processing results for "Radius LD**"  R\_of\_LD\_per\_cell=sqrt(average\_LD\_area./pi);  R\_of\_LD\_per\_cell=R\_of\_LD\_per\_cell\*0.1718;  if btn1\_disabled~=0  R\_of\_LD\_per\_cell=R\_of\_LD\_per\_cell(1:Clear\_Cells\_Selected\_So\_Far);  end | * Calculating average LD radius per cell from the average LD area per cell * Pixels to μm conversion * If “Stop Selecting Clear Cells” button was pressed, the vector is cut and unclear cells data is not included, otherwise the vector remains the same |
| **%% Processing Results for "Cell Area"**    Cells\_Areas\_micrometer=Areas\_cells\*0.029515;  if btn1\_disabled~=0  Cells\_Areas\_micrometer=Cells\_Areas\_micrometer(1:Clear\_Cells\_Selected\_So\_Far);  end | * Pixels to μm² conversion * If “Stop Selecting Clear Cells” button was pressed, the vector is cut and unclear cells data is not included, otherwise the vector remains the same |
| **%% Processing Results for "Cell Circularity" and "Cell Eccentricity"**  if btn1\_disabled~=0  Circularity\_cells=Circularity\_cells(1:Clear\_Cells\_Selected\_So\_Far);  Eccentricity\_cells=Eccentricity\_cells(1:Clear\_Cells\_Selected\_So\_Far);  end  delete(findall(0,'Type','figure')) | * If “Stop Selecting Clear Cells” button was pressed, the vector is cut and unclear cells data is not included, otherwise the vector remains the same * All image windows are closed |

1. Main file, for fibroblast cells (cell with no lipid droplets):

| *Matlab Code* | *Explanation* |
| --- | --- |
| clc  clear all  close all |  |
| global Num\_of\_cells\_per\_pic  global Cell\_number | Declaring variables as global:   * Number of cells per image * Cell number |
| **% Data inserted by the user**  im\_RGB1=imread('ImageName.jpg');  Num\_of\_cells\_per\_pic=A; | * Loading the micrograph image * Number of cells in the image |
| **%% Converting to binary image**  im\_gray=rgb2gray(im\_RGB1);  im\_gray=double(im\_gray);  im\_gray=im\_gray/255;  im\_gray=wiener2(im\_gray); | * The image is converted from RGB to grayscale, eliminating hue and saturation properties, but maintaining luminance properties * Define double precision for the image * Normalize grayscale levels to range between 0 and 1 * Filter image using two-dimensional adaptive Gaussian noise-removal filter (Wiener filter) |
| %% **Cells selection by the user**  figure(1)  imshow(im\_RGB1); title('Original Image');  Cell\_number=1;  labeled\_im\_area=im\_gray;  while Cell\_number<=Num\_of\_cells\_per\_pic  figure(2)  imshow(labeled\_im\_area)  title('Please mark the cell') | * Display the original image * Cell counter, cells selected by the user * The grayscale filtered image * As long as the user didn't select all the cells, perform the following operations: * Display the area-labeled image each time the loop is repeated |
| **%% Defining button**  % "How Many Cells Left?" button  btn2=uicontrol('Style','pushbutton','String','How Many Cells Left?','Position',[600 20 130 30],'Callback',@How\_many\_cells\_selected); | * Defining “How Many Cells Left?” button, for user convenience, when clicking on the button the user sees how many more cells he needs to select. |
| **%% Cell parameters**  cell=roipoly;  stats\_cell=regionprops('table',cell,'Area','Eccentricity','Perimeter');  Circularity\_cells(Cell\_number)=(4\*pi\*stats\_cell.Area)/(stats\_cell.Perimeter.^2);  Eccentricity\_cells(Cell\_number)=stats\_cell.Eccentricity;  Areas\_cells(Cell\_number)=stats\_cell.Area; | * The user marks the cell, by specifying a polygonal region of interest within the image. Roipoly defines a new binary image, where the pixels marked by the user are assigned a ‘1’ label, and all other pixels are assigned a ‘0’ label * Regionprops: measure properties of the selected cell * Cell circularity calculation * Cell eccentricity calculation * Cell area calculation |
| labeled\_im\_area(find(cell))=0;  Cell\_number=Cell\_number+1;  end | * The area of the selected cell is blackened for display purpose * Update the counter ‘Cell\_number’ * The above procedure is repeated until the user selects all the cells |
| **%% Processing Results for Cell Area**  Cells\_Areas\_micrometer=Areas\_cells\*0.029515;  delete(findall(0,'Type','figure')) | * Pixels to μm² conversion * All image windows are closed |

1. Function – “Stop\_Selecting\_Clear\_cells”:

| *Matlab Code* | *Explanation* |
| --- | --- |
| function Stop\_Selecting\_Clear\_cells(~,~)  global Cell\_number  global Clear\_Cells\_Selected\_So\_Far  global btn1  global btn1\_disabled | * Defining function   Global variables:   * Cell number * Amount of visibly clear cells, selected by the user * Button, to stop selecting clear cells * Controlling if ‘btn1’ is enabled or disabled |
| Clear\_Cells\_Selected\_So\_Far=Cell\_number-1;  set(btn1,'Enable','off')  btn1\_disabled=1;  end | * Calculation of how many clear cells were selected by the user until the user clicked on “Stop Selecting Clear Cells” button * “Stop Selecting Clear Cells” button (‘btn1’) is disabled, so the user will not be able to click those buttons once more * When ‘btn1’ is disabled, the value of ‘btn1\_disabled’ is set to be different than 0 |

1. Function – “How\_many\_cells\_selected”:

| *Matlab Code* | *Explanation* |
| --- | --- |
| function How\_many\_cells\_selected(~,~)  global Num\_of\_cells\_per\_pic  global Cell\_number | * Defining function   Global variables:   * Number of cells per image * Cell number |
| temp\_cells\_left\_to\_select= Num\_of\_cells\_per\_pic-Cell\_number+1;  temp\_cells\_selected\_so\_far=Cell\_number-1;  display(Num\_of\_cells\_per\_pic);  display(temp\_cells\_selected\_so\_far);  display(temp\_cells\_left\_to\_select);  end | * Calculation of how many cells are left to select * Calculation of how many cells were selected so far * Displaying the results |