Contents

Cell Areas

```
% Abbott Lab at Carnegie Mellon University, Department of Biomedical
% Engineering
% Author: Tal Dassau, Nickia Muraskin, Liz Johnston
% Last Updated: 8/2/23

clear
clc
close all
```

Load in Image

Image is loaded in based on selection

```
filterSpec = {'*.jpg;*.tif;*.png;*.gif','All Image Files'};
[image_name, pathName] = uigetfile(filterSpec);
img = imread(image_name);

% Info is extracted from the file to obtain scale and make directory for the
% analyzed data

info = imfinfo(image_name);
format = info.Format;
[~, imageName, ~] = fileparts(image_name);
outputDir = fullfile(pwd, imageName);
mkdir(outputDir);

% Convert units based on the scale stored in the image
% This calculates the correction ratio in order to convert units
% from being in pixels to being in microns
% Check if the field 'XResolution' exists in the info structure

if isfield(info, 'XResolution')
    % Field exists, extract the value
    scale = info.XResolution;
    if isempty(scale)
        % Prompt the user to enter a value with a dialogue box
        prompt = {'Enter pixel/micron value:'};
        dlgtitle = 'Error: Scale not found in file structure';
        dims = [1 75];
        definput = {'0.9794'};
        answer = inputdlg(prompt,dlgtitle,dims,definput, "on");
        scale = str2double(answer{1});

        % check that input is numeric
        while isnan(scale)
            prompt = {'Enter a numeric pixel/micron value:'};
            dlgtitle = 'Error: Not a Number';
            dims = [1 75];
            definput = {'0.9794'};
            answer = inputdlg(prompt,dlgtitle,dims,definput, "on");
            scale = str2double(answer{1});
        end
    else
        % Field does not exist, handle the case accordingly
        prompt = {'Enter pixel/micron value:'};
        dlgtitle = 'Error: Scale not found in file structure';
        dims = [1 75];
        definput = {'0.9794'};
        answer = inputdlg(prompt,dlgtitle,dims,definput, "on");
        scale = str2double(answer{1});

        % check that input is numeric
        while isnan(scale)
            prompt = {'Enter a numeric pixel/micron value:'};
            dlgtitle = 'Error: Not a Number';
            dims = [1 75];
            definput = {'0.9794'};
            answer = inputdlg(prompt,dlgtitle,dims,definput, "on");
            scale = str2double(answer{1});
        end
    end
end

% Use the value of 'scale' for further calculations
```

```
Warning: Directory already exists.
```

High-pass

```
i = im2gray(img); % grayscale

bw = imbinarize(i,"adaptive"); % Thresholding

bwborder = imclearborder(bw); % suppresses lighter structures connected to the border
```

Stats

Convert to microns

```
ratio = 1 / scale;
ratio2 = ratio^2;

% Create table of stats
stats = regionprops('table', bwborder, 'Area', 'Centroid', 'MajorAxisLength', 'MinorAxisLength');

centers = stats.Centroid; % Collect centers of all regions
roundCenters = round(centers); % Round the centers

% Find unique centers using uniquetol
[uniqueCenters, uniqueIndices] = uniquetol(roundCenters, 0.01, 'highest', 'ByRows', true);

% Retrieve the corresponding rows from the original stats table
stats = stats(uniqueIndices, :);

% Calculate diameter and radii
diameters = mean([stats.MajorAxisLength, stats.MinorAxisLength], 2);

% Apply correction ratios
converted_diameters = diameters*ratio;
converted_area = stats.Area*ratio2;

% Add to table
stats.diameters = diameters;
stats.converted_area = converted_area;
stats.converted_diameters = converted_diameters;

% Remove unnecessary stats
cells = removevars(stats, {'Centroid', 'MajorAxisLength', 'MinorAxisLength'});
cells = addvars(cells, uniqueCenters, 'Before', 'Area');

% Find and store lipids in new table, artifacts are also removed
lipid_size = 20;
minimum = 5;
idx = cells.converted_diameters <= lipid_size;
lipids = cells(idx, :);
lipidx = lipids.converted_diameters >= minimum;
lipids = lipids(lipidx, :);

% Remove lipids from cell table
cells = cells(cells.converted_diameters >= lipid_size, :);

% Check to see if anything is too big to be an adipocyte
max_size = 300;
idx2 = cells.converted_diameters <= max_size;
cells = cells(idx2, :);

% Update Centers and radii variables
uniqueCenters = cells.uniqueCenters;
radii = cells.diameters / 2;
```

Masking

```
% Mask high filter cells (fix scalar adjustment)
filtered_bw = insertShape(img, 'FilledCircle', [uniqueCenters(:, 1), uniqueCenters(:, 2), radii * 1.18], 'Color', 'black', 'Opacity', 1);
```

Low-pass

```
% Composite image: original image with masked cells
merged = imfuse(img, filtered_bw);

% Turn merged image grey
merged_gray = rgb2gray(merged);

% Binarize merged grey
merged_bw = imbinarize(merged_gray, 'adaptive');
border_merged_bw = imclearborder(merged_bw);
```

Stats on Merged Image

```
stats2 = regionprops('table', border_merged_bw, 'Area', 'Centroid', 'MajorAxisLength', 'MinorAxisLength');

% Collect centers of "cells" for later
centers2 = stats2.Centroid;

roundCenters2 = round(centers2); % Round the centers

% Find unique centers using uniquetol
[uniqueCenters2, uniqueIndices2] = uniquetol(roundCenters2, 0.1, 'highest', 'ByRows', true);

% Retrieve the corresponding rows from the original stats table
stats2 = stats2(uniqueIndices2, :);

% Calculate diameter and radii
diametersnew = mean([stats2.MajorAxisLength, stats2.MinorAxisLength], 2);

% Apply correction ratios
converted_diameters2 = diametersnew*ratio;
converted_area2 = stats2.Area*ratio2;

% Add to table
stats2.diameters = diametersnew;
stats2.converted_area = converted_area2;
stats2.converted_diameters = converted_diameters2;

% Remove unnecessary stats
cells2 = removevars(stats2, {'Centroid', 'MajorAxisLength', 'MinorAxisLength'});
cells2 = addvars(cells2, uniqueCenters2, 'Before', 'Area');

% Find and store lipids in new table
idx3 = cells2.converted_diameters <= lipid_size;
lipids2 = cells2(idx3, :);
lipidx2 = lipids2.converted_diameters >= minimum;
lipids2 = lipids2(lipidx2, :);

% Remove lipids from cell table
cells2 = cells2(cells2.converted_diameters >= lipid_size, :);

% Rename the variable so we can concatenate the table of lipids
cells2 = renamevars(cells2, 'uniqueCenters2', 'centers');
cells = renamevars(cells, 'uniqueCenters', 'centers');
AdipocyteTable = [cells; cells2];

% Compare High and lowpass and only keep unique
% Round the centers of the cells to approximate uniqueness
roundedcenters = round(AdipocyteTable.centers);

% Check to see if these centers are unique or if there are cells in both high and lowpass
[~, ia] = uniquetol(roundedcenters, 0.01, 'highest', 'ByRows', true);
AllUnique = AdipocyteTable(ia, :);
AdipocyteCenter = AllUnique.centers;
AdipocyteRadius = AllUnique.diameters / 2;
```

Extracellular Lipids

Combine lipid tables from high and low pass

```
lipids2 = renamevars(lipids2, 'uniqueCenters2', 'centers');
lipids = renamevars(lipids, 'uniqueCenters', 'centers');
AllEL = [lipids; lipids2];

%Ensure Uniqueness
roundedELcenters = round(AllEL.centers);
[~, ialipids] = uniquetol(roundedELcenters, 0.01, 'highest', 'ByRows', true);
AllUniqueEL = AllEL(ialipids, :);
ExtracellularLipidCenter = AllUniqueEL.centers;
ExtracellularLipidRadius = AllUniqueEL.diameters / 2;
```

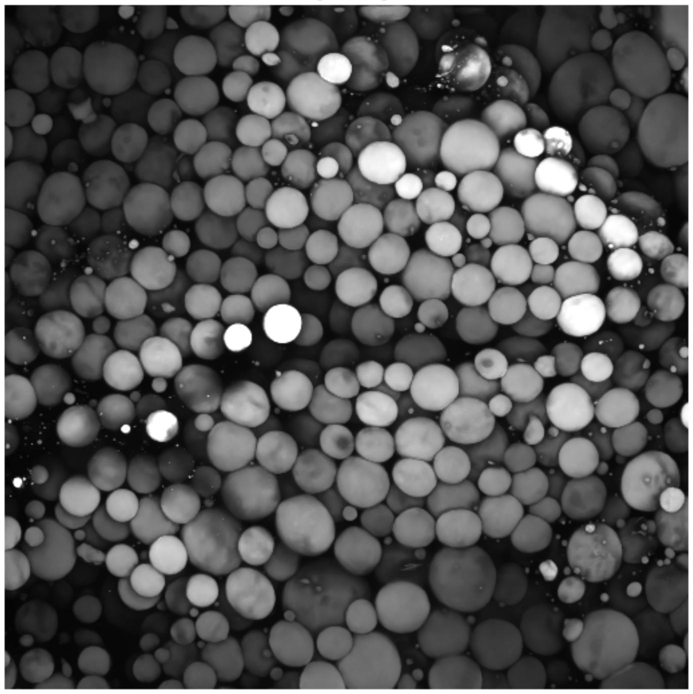
All Lipids together

```
Total = [AllUnique; AllUniqueEL];
roundedallcenters = round(Total.centers);
[~, ia2] = uniquetol(roundedallcenters, 0.01, 'highest', 'ByRows', true);
AllTotalUnique = Total(ia2, :);

%Making sure there is no overlap between the extracellular lipids and
%adipocytes
FinalIndex = AllTotalUnique.converted_diameters <= lipid_size;
FinalExtracellularlipids = AllTotalUnique(FinalIndex, :);

% Remove lipids from cell table
FinalAdipocyte = AllTotalUnique(AllTotalUnique.converted_diameters >= lipid_size, :);

% Final Lipid Quantification
UniqueELCenter = FinalExtracellularlipids.centers;
UniqueELRadius = FinalExtracellularlipids.diameters / 2;
UniqueAdipoCenter = FinalAdipocyte.centers;
UniqueAdipoRadius = FinalAdipocyte.diameters / 2;
```

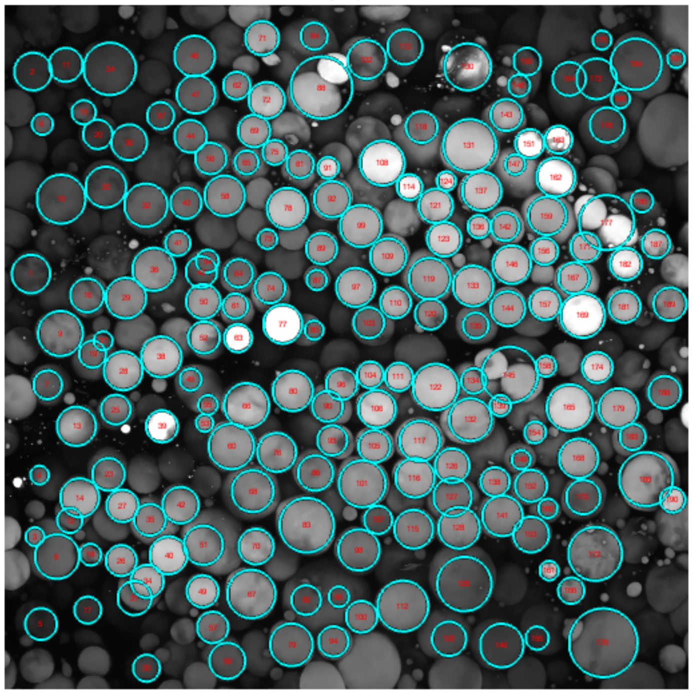
Create All Figures and Save them in the directory

```
figure(1);
imshow(img);
title('Original Image');
outputPath = fullfile(outputDir, 'Original_image.png');
saveas(gcf, outputPath);

figure(2);
imshow(img);
hold on;
title('Adipocytes');
viscircles(UniqueAdipoCenter, UniqueAdipoRadius, 'Color', 'c');
for k = 1:size(UniqueAdipoCenter, 1)
    text(UniqueAdipoCenter(k, 1), UniqueAdipoCenter(k, 2), sprintf('%d', k), 'Color', 'r','FontSize',6, 'HorizontalAlignment', 'center', 'VerticalAlig
end
hold off;
Adipocytes = fullfile(outputDir, 'Adipocytes.png');
saveas(gcf, Adipocytes);

figure(3);
imshow(img);
hold on;
title('Extracellular Lipids');
viscircles(UniqueELCenter, UniqueELRadius, 'Color', 'm');
hold off;
extracellularlipidfig = fullfile(outputDir, 'ExtracellularLipids.png');
saveas(gcf, extracellularlipidfig);

figure(4);
imshow(img);
hold on;
title('All Lipids');
viscircles(UniqueAdipoCenter, UniqueAdipoRadius, 'Color', 'c');
viscircles(UniqueELCenter, UniqueELRadius, 'Color', 'm');
hold off;
end_fig = fullfile(outputDir, 'AllLipids.png');
saveas(gcf, end_fig);

%Add Count Column for adipocytes
countColumn = (1:height(FinalAdipocyte))';
newTable = table(countColumn, 'VariableNames', {'Count'});

% Concatenate the new table with the existing table
FinalAdipocyte = [newTable,FinalAdipocyte];
newColumnNames = {'Count', 'Center','Area (pixel^2)','Diameter (pixel)','Area (um^2)','Diameter (um)'};
FinalAdipocyte.Properties.VariableNames = newColumnNames;

% Put Excel Files into the directory
adipocyteDataPath = fullfile(outputDir, 'Adipocyte_data.xlsx');
writetable(FinalAdipocyte, adipocyteDataPath);

extracellularLipidDataPath = fullfile(outputDir, 'ExtracellularLipid_data.xlsx');
writetable(FinalExtracellularlipids, extracellularLipidDataPath);

allLipidDataPath = fullfile(outputDir, 'AllLipid_data.xlsx');
writetable(AllTotalUnique, allLipidDataPath);
```

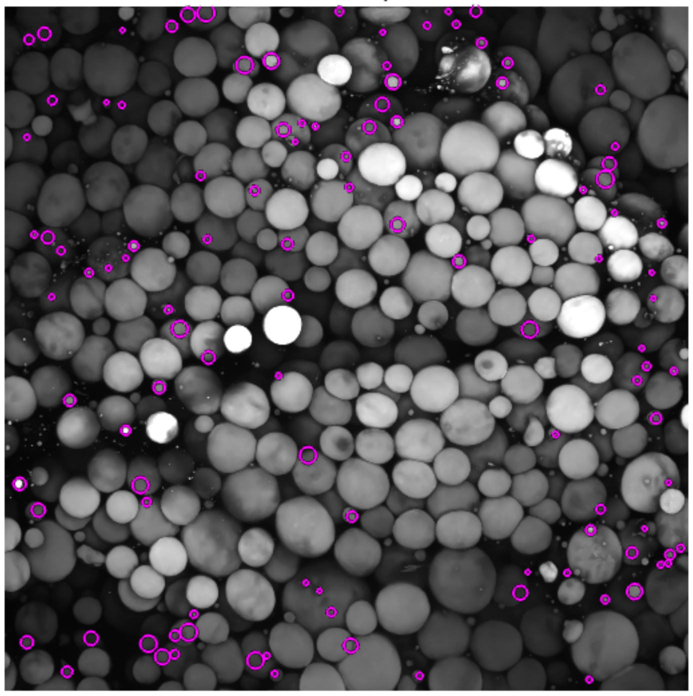
Original Image



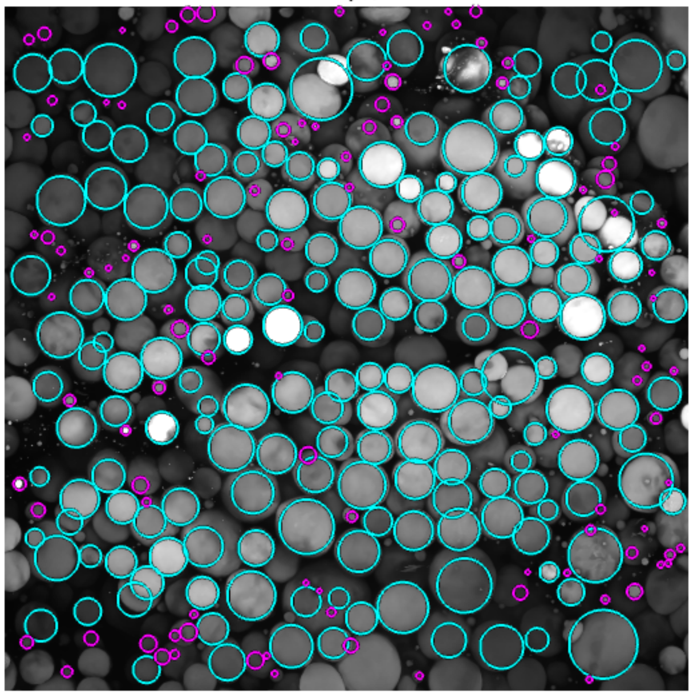
Adipocytes



Extracellular Lipids



All Lipids



Published with MATLAB® R2023b