Lab3 Bildanalys

Oscar Jacobson

December 29, 2022

1 Intro

In this exercise I am to find, classify and count the coins in the following figure 1. Using matlab, a short script was produced to perform the task.

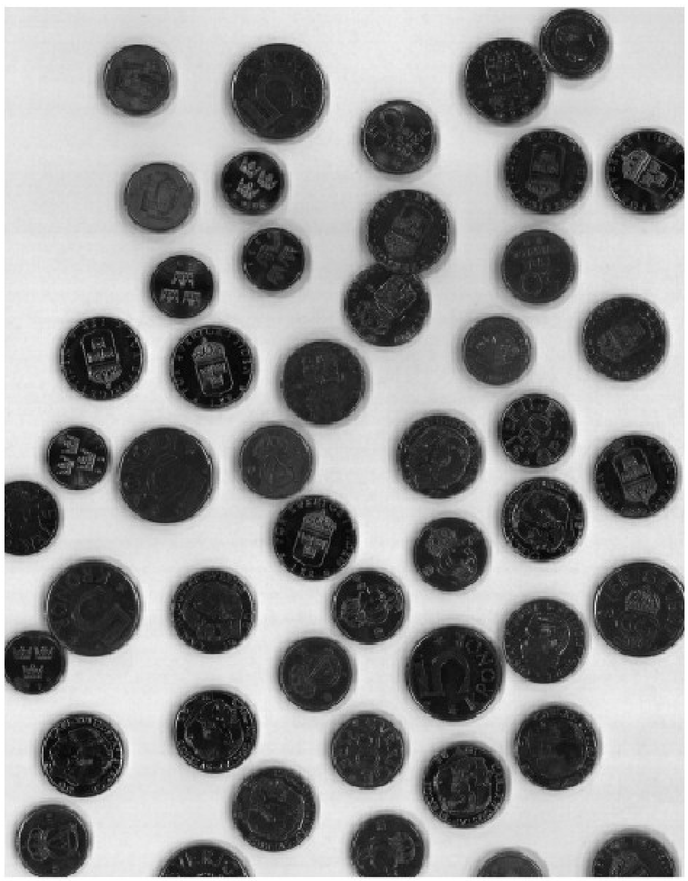

Figure 1: Original figure

2 Code

```matlab
clear all

% I = imread("bacteria.tif");
I = imread("coins.tif");
% figure(); imshow(I);

Ismall=I(1:100,200:350);
sz = size(I);
figure(); imshow(I);

% Preprocessing by: https://se.mathworks.com/help/images/marker-controlled-watershed-segmentation.html
% Gets rid of coin patterns. And can be used to reduce oversegmentation.


% Errode and reconstruct to get a reconstructed figure
% Not necessary to use in this case but still applied
%
se = strel('disk',10); % Morphological structuring element
Ie = imerode(I,se);
Iobr = imreconstruct(Ie,I);

% Dilate and reconstruct the figure using the above reconstructed figure as a complementary figure.
% This gets rid of coin patterns inside of the coins.
% Not necessary to use in this case but still applied
%
Iobrd = imdilate(Iobr,se);
I = imreconstruct(imcomplement(Iobrd),imcomplement(Iobr));
I = imcomplement(I);
figure(); imshow(I);


T = graythresh(I);
Thr = imbinarize(I,T); % Take T-0.1 to get a lower threshold, makes shadows less relevant but patterns inside coins visible.
% figure(); imshow(Thr);

```

```
% T2 = graythresh(Ismall);
% Thr2 = imbinarize(Ismall,T2);

figure(); imshow(Thr); % White background binary
Idist=bwdist(Thr);

Thr3 = medfilt2(Thr,'symmetric');
figure(); imshow(Thr3); % White background binary with median filter


Thr4 = 1-Thr3;
figure(); imshow(Thr4); % Black background binary with median filter


dist = mat2gray(bwdist(Thr3));
figure(); imshow(dist); % Distance in white with black background


h = fspecial("disk",2);
distavg = imfilter(dist,h); % Use "disk" average here to prevent oversegmentation of segments between coins
figure(); imshow(distavg);


dist2 = 1-distavg;
figure(); imshow(dist2); % Distance in black with white background for watershed function



wa = watershed(dist2,8); % Splits all coins and classifies them 1:n
wa(~Thr4) = 0; % ~ means not in
rgb = label2rgb(wa,'jet',[.5 .5 .5]);
figure(); imshow(rgb) % colored image


% Find the boundaries of all labeled coins
```

```
B = bwboundaries(wa,'noholes');


% Find which coin indexes are on the border of the figure
border = zeros(size(B,1),1);
for k = 1:length(B)
  boundary = B{k};
  for i = 1:size(boundary,1)
      if boundary(i,1) == 1 || boundary(i,2) == 1 || boundary(i,1) == sz(1,1) || boundary(i,2) == sz(1,2)
          border(k,1) = 1;
      end
  end
end


% Plot the boundaries
figure();
imshow(rgb) % colored image
hold on
for k = 1:length(B)
    boundary = B{k};
  plot(boundary(:,2),boundary(:,1),'w','LineWidth',2)
end
hold off



% Get centroids and diameters, weird at the edges
stats = regionprops('table',wa,'Centroid','MajorAxisLength','MinorAxisLength'); % Finds and outlines circles
centers = stats.Centroid;
diameters = mean([stats.MajorAxisLength stats.MinorAxisLength],2);
radii = diameters/2;

for i = length(border):-1:1
    if border(i,1) == 1
        radii(i) = nan;
    end
end
```

```
fives = [];
one = [];
halfs = [];

for i = 1:length(radii)
    if radii(i) > 26.75
        fives = [fives; [radii(i) centers(i,:)]];
    elseif radii(i) < 23.25
        halfs = [halfs; [radii(i) centers(i,:)]];
    else
        one = [one; [radii(i) centers(i,:)]];
    end
end

figure();
imshow(rgb) % colored image
hold on
viscircles(fives(:,2:3),fives(:,1),'color','b');
viscircles(one(:,2:3),one(:,1),'color','r');
viscircles(halfs(:,2:3),halfs(:,1),'color','g');
%
% viscircles(centers,radii);
hold off

figure(); histogram(radii, 15);
title('Histogram of radii')
xlabel('Circle radius')
ylabel('Frequency of coin size')


% Morphological tophat can probably be used instead of meadian filtering
```

2.1 Code Comments and figures

The original figure is read into matlab using the *imread* function which produces a matrix with all of the intensity values of each pixel. The original figure is shown above in figure 1.

Some of the coins contains patterns which might disturb the classification process. A coin should ideally be a homogeneous object when analyzed and the brighter pixels in the patterns might make sure that this is impossible. To produce a figure with homogeneous coins the figure was dilated (Line 26) and reconstructed (Line 27-28) using a eroded and reconstructed figure (Line 19-20) as a complementary figure. This was all done using a disk-shaped morphological strurcturing unit with the radius of 10 (Line 18). The morphphological structuring unit decides what area and shape the filers applied will affect for each pixel. These lines of code was copied from an article from the *Mathworks* website [1]. During testing the homogeneous transformation of the coins was not needed for the classification to work, but was included anyways to prove a concept.

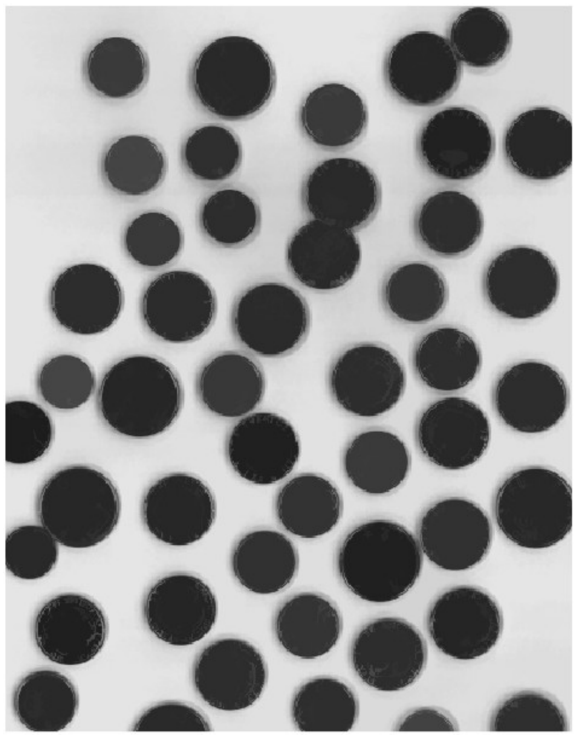

Figure 2: Figure with homogeneous coins.

(Line 32) takes the figure and evaluates a suitable threshold for where the intensities can be binarized. When binarized (Line 33) all intensities above the threshold T are set to 1 and all intensities below are set to 0. The resulting figure 4.

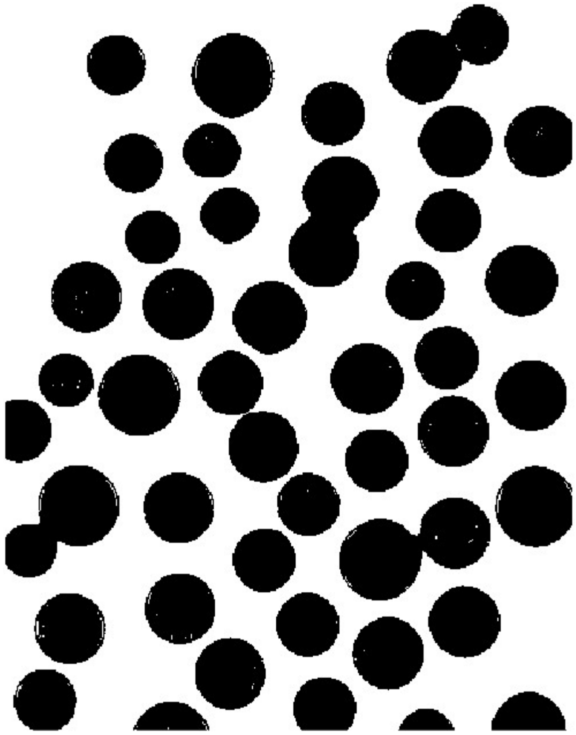

Figure 3: Binary figure.

A median filter was then applied (Line 43) to the binary figure to make sure that coins are fully homogeneous before analyzing them. This is required since the binary figure still include some small white dots inside of some coins resulting from shadows and imperfect edges etc. The figure was also inverted (Line 47).

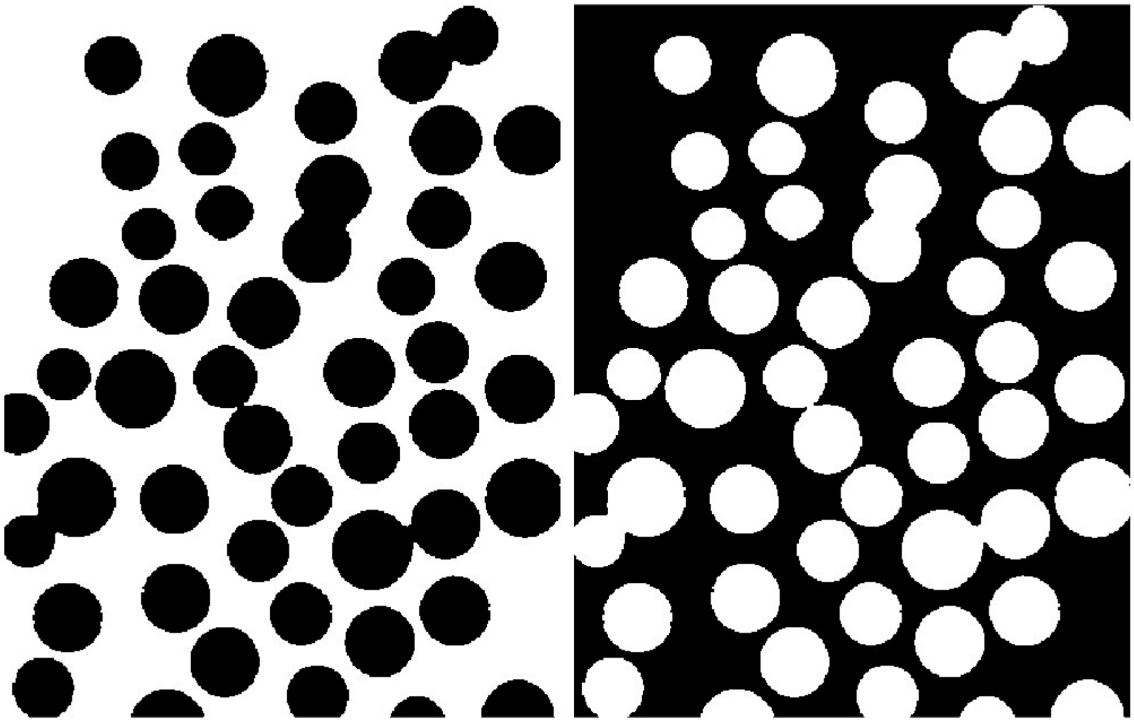

Figure 4: Binary figure with median filtering.

The distance of every pixel to the nearest "empty" pixel is the measured using the matlab function *bwdist* (Line 51) which produces potential-field-like patterns where the central points of every coin can easily be identified. A disk-shaped averaging filter was also applied on this figure (Line 55-56) to reduce oversegmentation in the next step. Results shown if figure 5

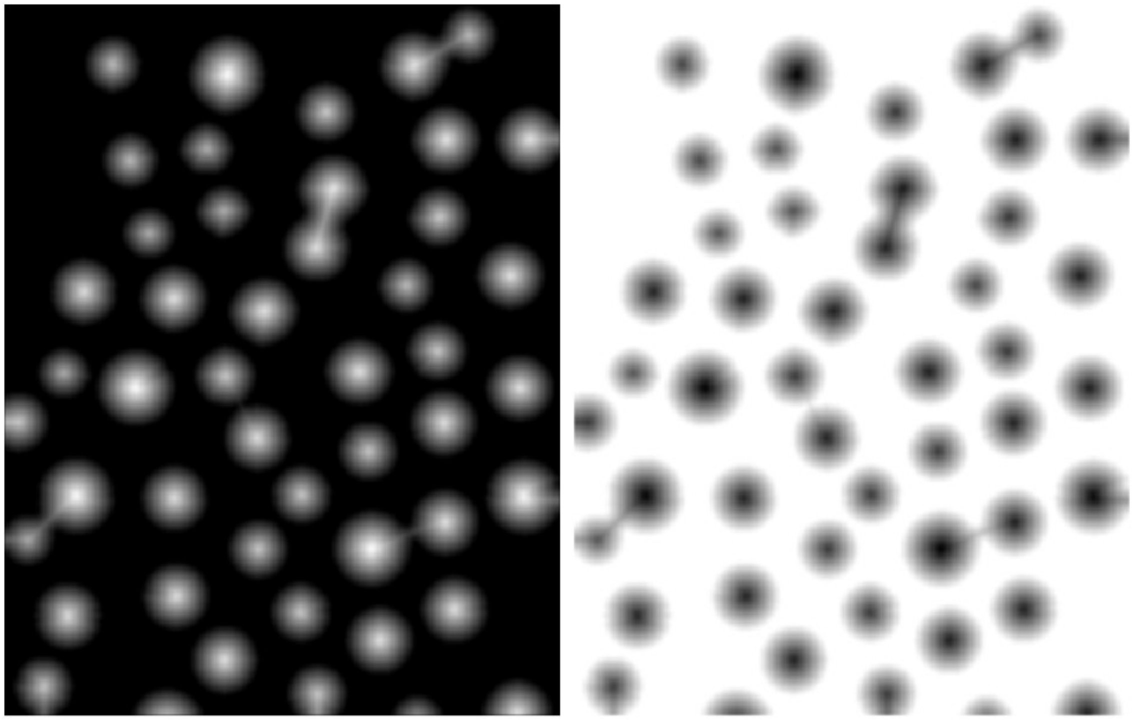

Figure 5: Binary figure with median filtering.

The matlab function *watershed* is used to find the local maxima in each coin to segment and label each one of them (Line 65). A logical NOT IN is then applied to the result to apply zeros to every uninteresting pixel (Line 66). The segmentation is presented in figure 6 below. The matlab function *bwboundaries* was then used to find the boundaries of all the already labeled coin sections.

(Line 76-84) determines if any of the labeled coin sections has a boundary that goes outside of the figure boundary. The right image in figure 6 shows the detected boundaries.

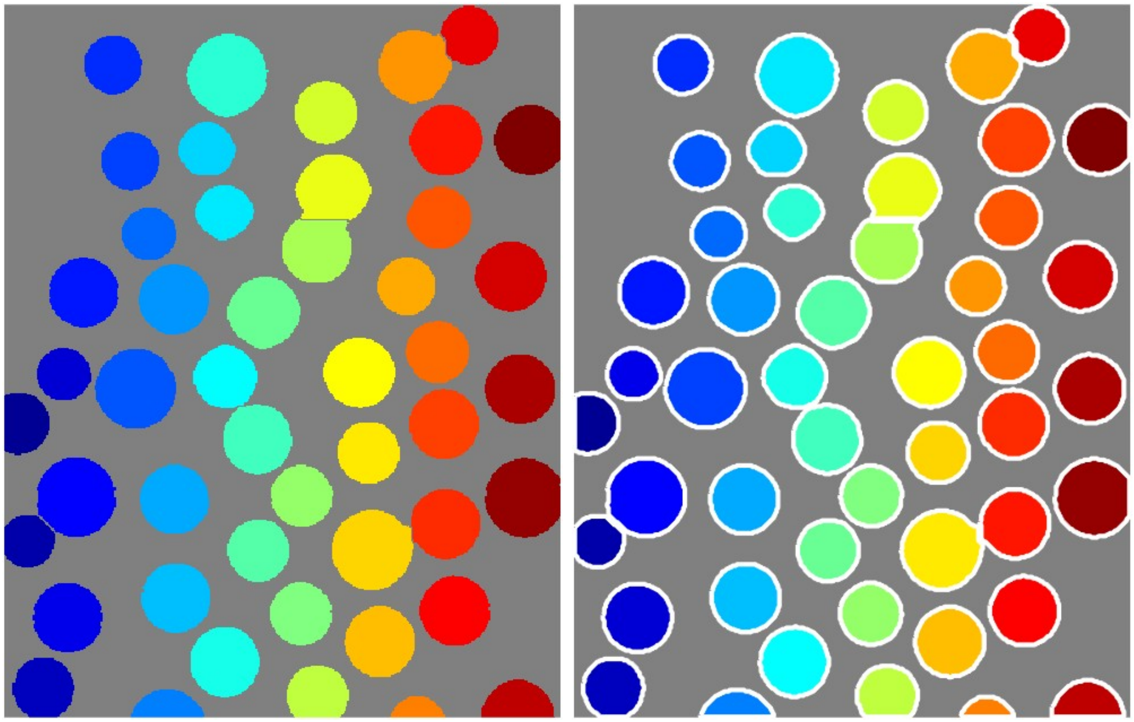

Figure 6: Watershed function segmentation and border detection.

The matlab functon *regionprops* was then used to find the properties of the defined coin segments, retrieving radii and center points of each coin (Line 100). (Line 105-109) then discards the values for coins that were detected to have a border outside of the figure border.

The radii was then plotted into a histogram which clearly show that there are 3 large blocks of coin sizes. Using visually determined values between each stack in the histogram the number of coins of each size could be determined, in the figure 6 above, 5kr coins are marked with blue, 1kr coins are marked with red and 0.5kr coins are marked with green.

Coins detected at the border are not marked at all and also not counted in the histogram. This was because the *regionprops* function could not handle the circles at the edges well and contributed with false information and outliers in the histogram.

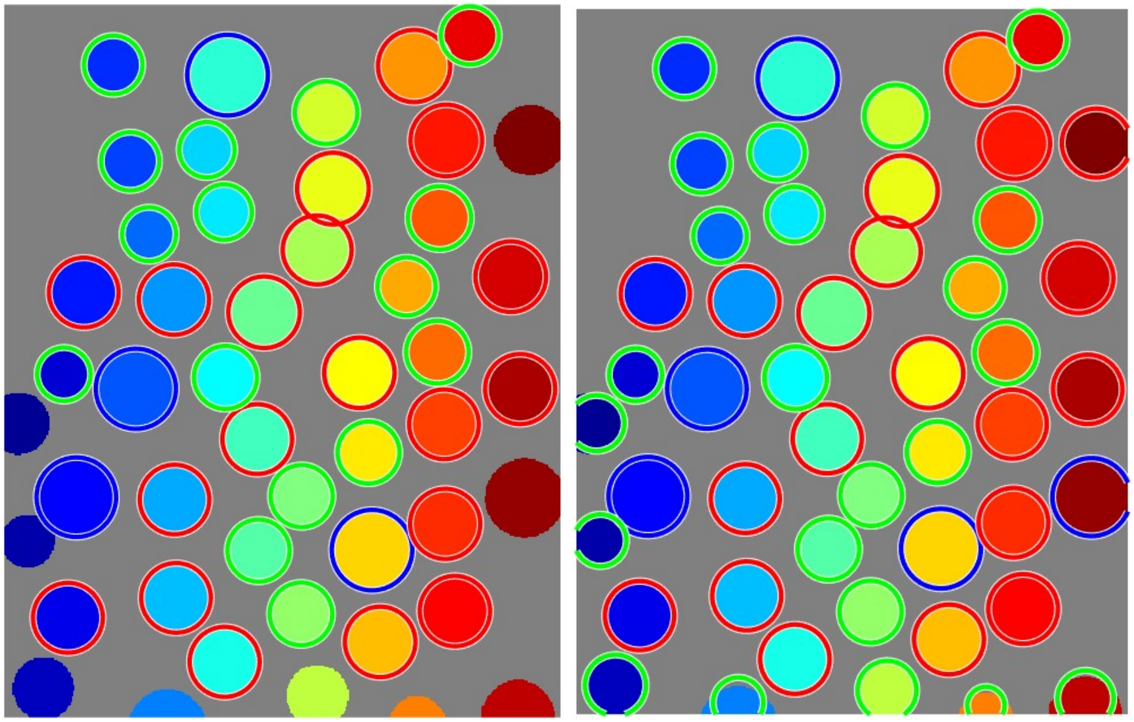

Figure 7: Final classification of coins and visualization of errors at border.

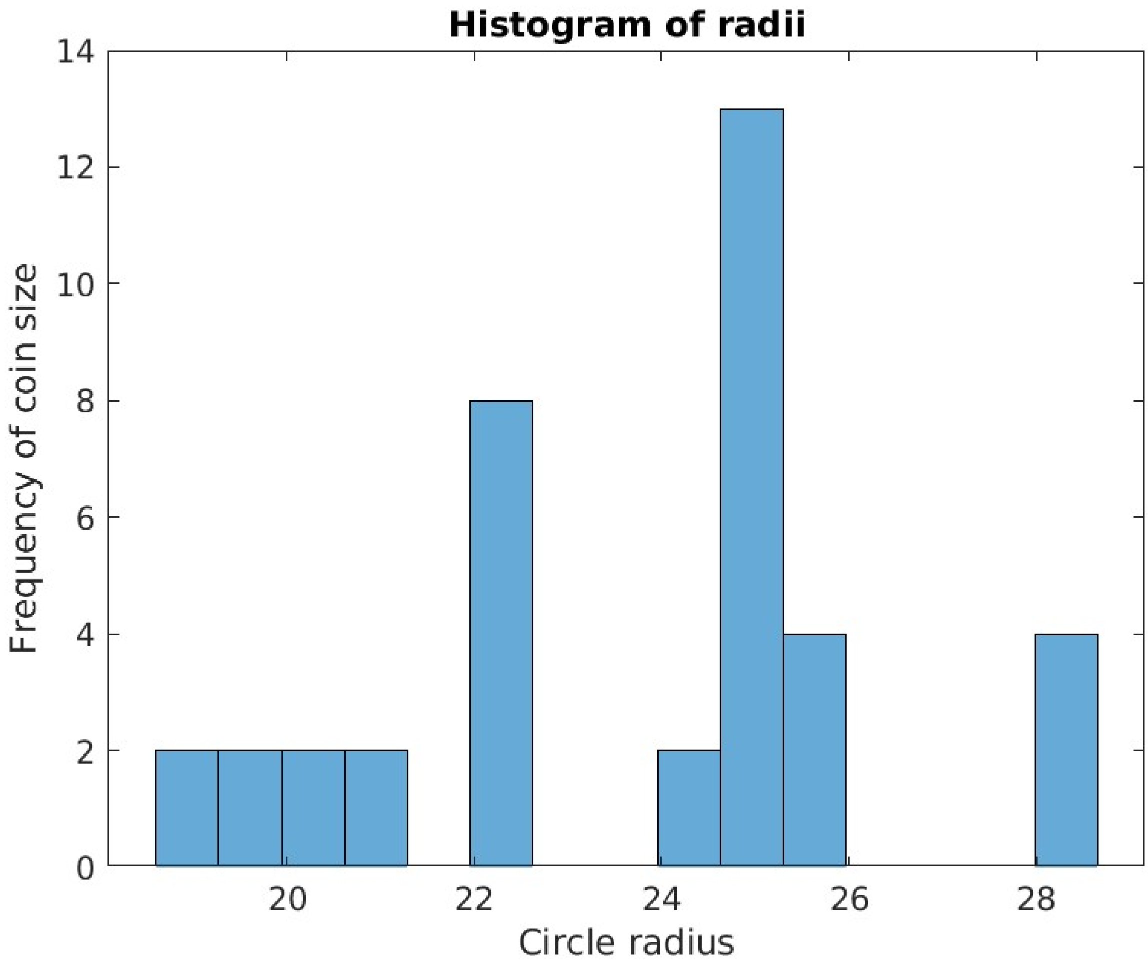

Figure 8: Histogram of coin radii.

3 Discussion

The coins on the edges are currently not handled at all. This is clearly visible in the bottom of the right image in figure 7 where three larger coins are incorrectly labeled as small because the fitted circles are clearly smaller than the actual coins. This is the cause of *regionprops* on line 100 assuming that the information given is the entire coin and incorrectly measures central point and size. This would also causes a very small outlier in the histogram in figure 8 as well as incorrectly contributing large coins to the smallest block in the histogram (Currently not as these are discarded).

By the histogram in figure 8 the size of the smaller coins seems to be more affected by randomness compared to the larger coins. This might be because a larger portion of their size is affected by shadows or overlapping etc. All the sizes are in fact subject to the radius approximations made by *regionprops* which will cause the results to be somewhat random.

This method is simple and suited to work on this problem only. The averaging step on line 55-56 for example was only a quick fix to the oversegmentation of this particular problem and would probably not be able to solve the problem when applied to other images.

This method has labeled and classified all coins within the border correctly (As far as I can tell) and from there the total amount of money can be calculated by multiplying the magnitudes in the histogram with the respective values of the coins. Most of the discarded coins on the border are also labeled correctly which would make the prediction of that method pretty accurate as well.

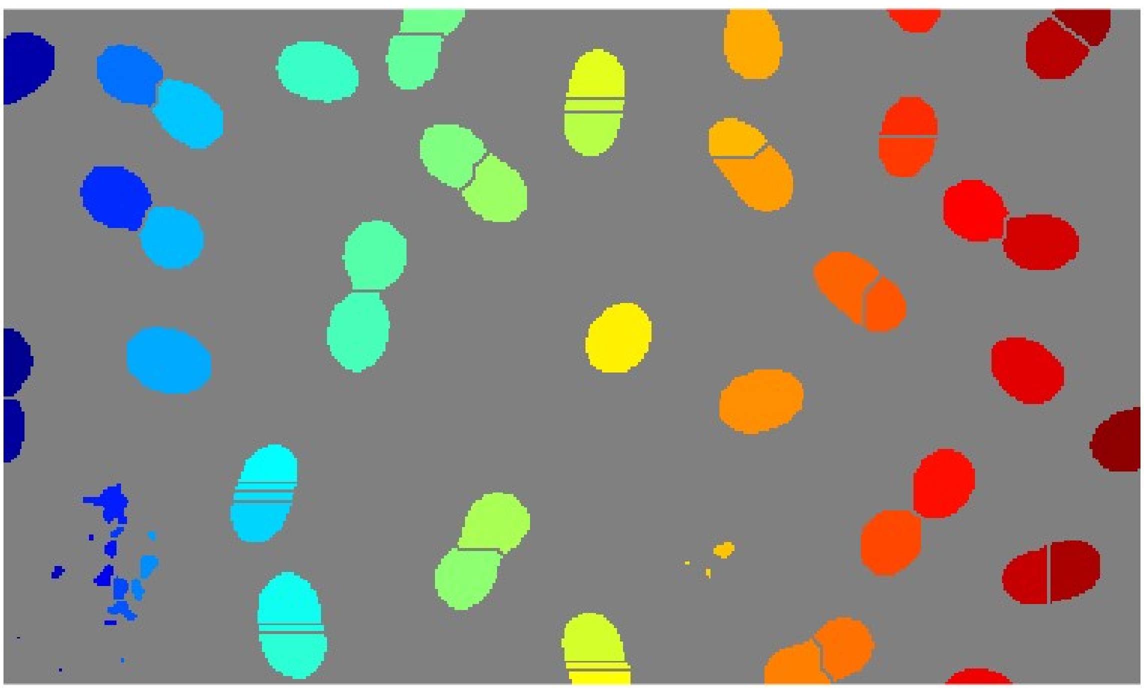

Figure 9: Test run on bacteria.jpg.

The method is not very general and would only work on almost perfect circles as of now. As seen in figure 9 there is some oversegmentation in the bacteria. A fix to this would probably include not using the *watershed* function directly on the bacteria but make sure to label them manually. This could probably be done with the *bwboundaries* function as the bacteria does not seem to overlap as much as the coins do. This would probably have to be coupled with some modification of the data from *regionprops* as the radius of a bacteria might not be valuable information in this case. The data of the size of the bacteria might in fact have to be manually extracted from the actual boundaries given from *bwboundaries*.

There also exists some dirt or noise in the bacteria image which has to be cleaned up, a morphological tophat filter is great at identifying small noise-particles and could probably be used to identify the features of importance.

References

[1] Marker-controlled watershed segmentation, mathworks. https://se.mathworks.com/help/images/marker-controlled-watershed-segmentation.html. Accessed: December 29, 2022.