Exercise 3, HT2020
Segmentation and analysis of circular objects using MATLAB Image Processing Toolbox

Assignment in partial fulfilment of the requirements for the course

Computer-Assisted Image Analysis I

UPPSALA UNIVERSITET

Master's Programme in Computer Science

Department of Information Technology

Authors:
Abraham Mathewos Meja
Marcello Pietro Vendruscolo
Shubhomoy Biswas

December 10, 2020

Contents

List of Figures

1 Question 1

Write a script that solves the segmentation task of coin.tif. Include a comment for every function saying why that particular function was needed.

imageAnalysis_lab3.m

```
original = imread('images/coins.tif'); %Open the original image.
median = medfilt2(original,[3 3],'symmetric'); %Remove possible noise.

%Compute adequate thresholding value and apply to the original image. Output image has black foreground and white background (binary image).
T = graythresh(median);
binary = imbinarize(median,T);

%Calculate the distance transform of the complement of the binary image. Complement of the distance transformed for the watershed transform:
%light pixels -> high elevations, dark pixels -> low elevations. Pixels outside the region of interest are set to 0.
distanceT = bwdist(binary,'cityblock');
neg_distanceT = -distanceT;
watersheded = watershed(neg_distanceT);
watersheded(~distanceT) = 0;

%Labelling the watershed transf. with 8-neighbours and eroding for better identification of ROIs.
labelled = bwlabel(watersheded,8);
segmented = bwmorph(labelled, 'erode');

%Find local maxima of each ROI and map with values from distance transf. + paint similar regions (radius) with similar colours.
maxima = bwmorph(imextendedmax(distanceT,1), 'shrink', Inf);
CH = regionprops(segmented, 'ConvexHull');
[nrows, ncols] = size(maxima);
maxima = uint8(maxima);
segmented = uint8(segmented);
for row = 1:nrows
    for col = 1:ncols
        if maxima(row,col) == 1
            maxima(row,col) = distanceT(row,col);
        end
        if segmented(row,col) > 0 && maxima(row,col) > 0
            segmented(row,col) = maxima(row,col);
        end
    end
end
for index = 1:length(CH)
    roi = roipoly(segmented, CH(index).ConvexHull(:,1), CH(index).ConvexHull(:,2));
    segmented(roi) = max(segmented(roi));
end

%Extract the Area features from the labelled image and set up bins for the histogram.
F = regionprops(labelled, 'Area');
A = [F.Area];
bins = [];
for boundaries = 400:100:2600
    bins = [bins boundaries];
end
figure, imshow(label2rgb(segmented)); title('Segmented Image')
figure, histogram(A,bins); xlabel('area mesured in pixels'); ylabel('number of objects');
```

Figure 1: Figure showing segmentation process.

Figure 1 shows the code snippet for achieving the segmentation task on the *coin.tif* image. Following are the key functions used to achieve the same.

1. **medfilt2** - Function utilised to apply a 3 x 3 median filter not only to eliminate randomised noises that are smalled than 3 pixels in the background but also to eliminate most of reflection (i.e., bright spots) inside the coins due to the embrossed figures;
2. **graythresh** - Function utilised to get a suitable threshold value of the filtered image;
3. **imbinarize** - Function utilised to convert the filtered image to black and white using the specified threshold value;
4. **bwdist** - Function utilised to calculate the distance transform between the coins. This function allows the estimation of the radius of the coins and also the identification of connections between overlapping coins;
5. **watershed** - Function utilised to apply watershed segmentation to the image;
6. **bwlabel** - Function utilised to apply labels to the image using 8-neighbourhood method;

7. **bwmorph** - Function applied in two occasions to erode and reduce the coins to points, respectively. The purpose of the former was to better represent the segmentation while the purpose of the latter was to identify the maxima of the coins;
8. **regionprops** - Function utilised in two occasions to extract the area and convex hull (i.e., the smallest convex polygon containing the objects) features from the labelled images, respectively;
9. **roipoly** - Function utilised to define the regions of interest using the coordinates (x,y) of the vertices of polygons.
10. **label2rgb** - Function utilised to view the labelled image as color by colour-coding the labels.

2 Question 2

Include an image showing the final segmentation result.

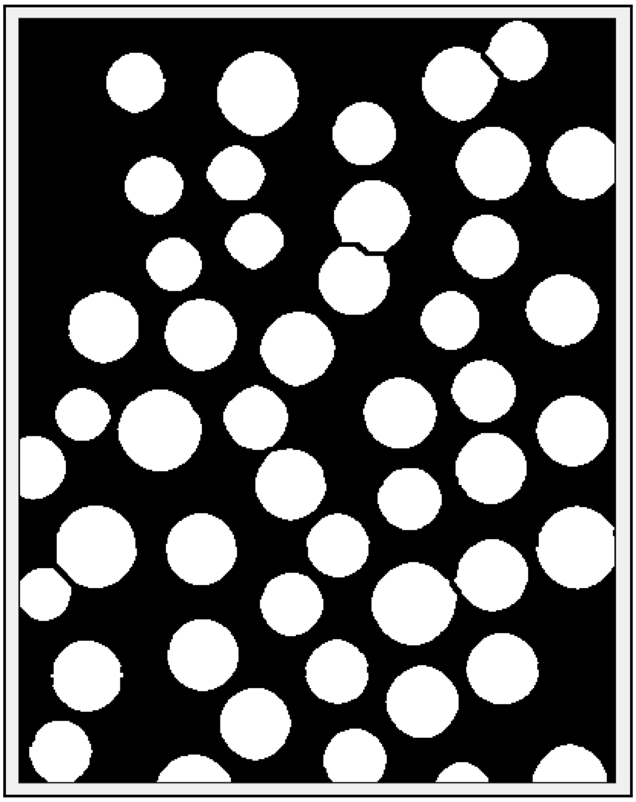

Figure 2: Figure showing binary classification of the coins image.

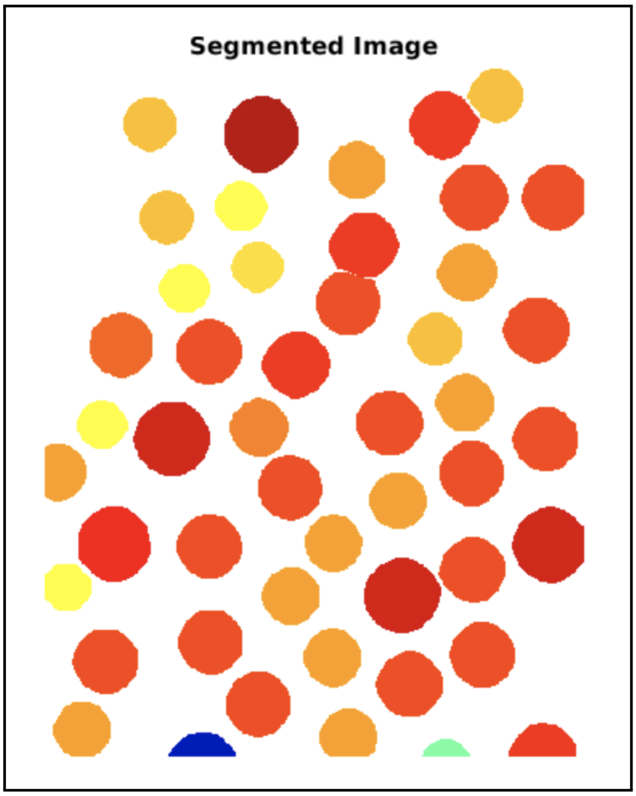

Figure 3: Figure showing final segmented image.

3 Question 3

Include a histogram showing the distribution of object size. Either area or radius could be used as a size measure.

The Figure 4 shows the histogram of the distribution of coins having similar area in pixels. The total number of objects is 48.

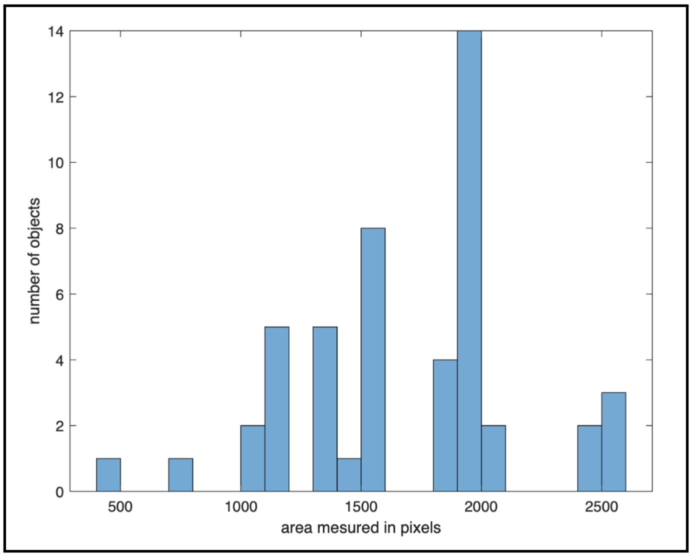

Figure 4: Histogram showing the frequency of coins having different areas.

4 Question 4

Discussion of errors and limitations in our final method.

Limitations - Image segmentation is a complex activity and there is not such an universal solution that solves all segmentation cases. Some of the limitations of the final method come from the image acquisition itself. When a digital image of an object is taken, the borders are not always sharp (i.e., there is some smoothness along the borders of the object in the image) which, ultimately, results in approximation rather than true reality, affecting the result of the segmentation. A setup where the source image is well differentiated from the background with proper illumination and no shading minimises the limitations of segmentation.

However, results that are not perfectly accurate can still be produced when objects are overlapped or noises are not completely removed, as illustrated by our first attempt of coin segmentation in Figure (5). This limitation was partially overcome by changing the algorithm utilised to compute the distance transform. However, the final outcome still does not precisely reflect reality.

Another limitation of the final method is that it only takes into consideration one feature (i.e., the area) of objects for identification purposes. This limitation reduces the applicability of the method in domains where different objects can have the same area value.

Errors - The area calculation for the coins probably introduced significant errors in comparison to the true output because of operations required to segment the image. Moreover, in Figure 3, it is possible to observe that some coins which seems to have the same currency value in the orignal image have been labelled as different objects, specially in the top-left quadrant of the image. Also, the overlapping coins have been identified by different shades of red (or orange) while they seem to have the same currency value. However, it is important to consider that coins of same currency value can, in reality, differ in area due to manufacturing errors and usage damaging.

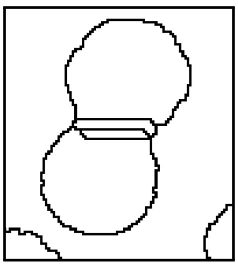

Figure 5: Segmentation error in overlapping objects.

5 Question 5

Is it possible to count the total amount of money using the implemented algorithm?

Given a labelled data mapping the currency value and the area of each coin, it is possible to count the amount of money using the algorithm with certain limitations, as follow.

1. The coins cut by the image boundaries can induce inaccuracies in the total value.
2. The overlapping of coins can introduce errors when computing the area of coins.
3. If two coins have little difference in area, for example in the cases where coins are damaged, it can happen that they are not accurately determined in certain cases.

Figure 3 shows coins of equal areas identified by the same colours. It is possible to extend this algorithm to calculate the total value based on the area of the coins. First, sum the number of coins that have a same approximate area, then multiply the sum with their currency value. We can then repeat the process for all coin types.

Limitation - It may be possible that two coins with the same area/structure have different currency values. In such cases, the segmentation algorithm cannot be utilised to count the total value of coins.

6 Question 6

How are the coins treated on the image border?

The first filter applied to the original image is the median filter to remove randomised and small possible noise. The filter size is 3 x 3, which is relatively small in comparison to the size of the image 505 x 394. Nevertheless, the argument *"symmetric"* was specified for the function to symmetrically extend the pixels at the borders of the image. The reason supporting this choice instead of padding the image with 0s (default behaviour) is the expected shape of coins, since circles are symmetric.

However, the erosion of the image (reference to line 17 in Figure 1), also reduces the area of coins cut by the borders by one row or column, depending on which border they are located. The remaining transformations and operations are not immensely affected by the manner in which the image borders are handled because the outcome producing by them are not affected by the borders.

For the coins at the borders which were not possible to identify as whole coins, we labelled them as outliers (blue and green coins in Figure 3) but taken them into area calculation to count the total number of coins. In case where we need to calculate the total currency value of coins, we would not include such coins inside the calculation.

7 Question 7

Is the implemented solution general in the sense that it can be used when analyzing images with arbitrary circular objects (i.e. not only coins.tif)? Try it on bacteria.tif to check whether it works on arbitrary ellipsoid or convex objects. Also, describe some possible modifications made in the program.

```
I = imread('bacteria.tif');

% Applying median filter to reduce noise
I_med = medfilt2(I, [3, 3], 'symmetric');

% % Applying the threshold level to get the BW image
% % The resultant image has black color for the object
bw = im2bw(I_med, graythresh(I_med));

% Closing the image to eliminate some background darkness
se = strel('disk', 7);
bw = imclose(bw, se);

% % Calculating the distance transform
% % The object connections becomes white
D = bwdist(bw, 'cityblock');

% % Converting the white objects and the connections to black again
D = -D;

% % Applying watershed
water = watershed(D);
water(~D) = 0;

% % Labelling
Ilabel = bwlabel(water, 8);

% Opening for segmentation
segmented = bwmorph(Ilabel, 'open');

subplot(2,3,1); imshow(I); title('Original Image');
subplot(2,3,2); imshow(I_med); title('Median Image');
subplot(2,3,3); imshow(bw); title('BW Image');
subplot(2,3,4); imshow(D, []); title('Distance Transformed Image');
subplot(2,3,5); imshow(Ilabel,[]); title('Watershed Applied');
subplot(2,3,6); imshow(segmented,[]); title('Segmented');
```

Figure 6: Code snippet for segmentation of bacteria.tif.

Most steps of the above solution can be applied in segmenting images similar to *coins.tif* image (Figure 6), such as:

1. Applying median filter to remove noise.
2. Converting to BW image using a threshold.
3. Calculating distance transform.

4. Applying *Watershed* on *Distance Transform*.
5. Labelling the watershed image.

Applying the same process in the *bacteria.tif* image does most of the work. However, it was needed to do certain modifications according to the properties of the given image. Since the image have few bacteria which are in the verge of splitting or combining, these can be counted as separate baterias rather than one single bacteria. It was required to apply *closing* before applying distance transform to account for this change.

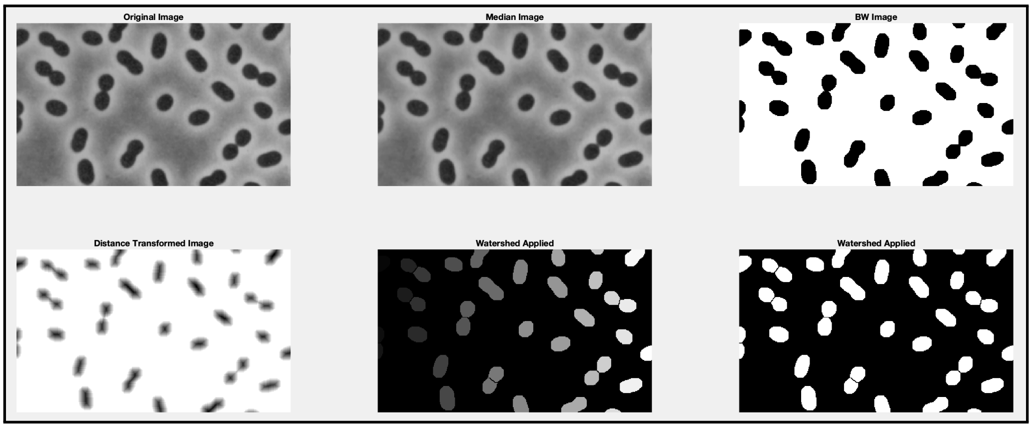

Figure 7: Binary Segmentation process.

Figure 8 illustrates the result obtained after the identification of bacteria by area feature, i.e. bacteria of similar areas have the same color. However, as it can be observed through the dark-blue coloured bacterium, the identification result is not perfect.

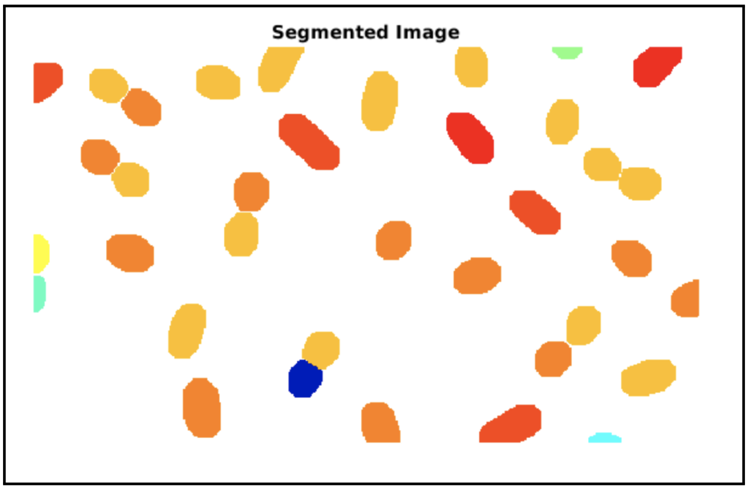

Figure 8: Colour-coded identification of bacteria.