College of Computer and Information Sciences
Computer Science Department

CSC 478
Image Processing

HW3: MATLAB 3

Prepared by:

Name: Sarah Alajlan

Student ID: 442202314

Dr. Seetah Alsalamah

Task 1:

MATLAB Code:

```
% The input image will be saved into the variable "Input."
Input = imread('riceTest.png');
% Creating a figure so I can display all figures in one window.
figure;
% Showing the image before any processing steps.
subplot(2, 3, 1);
imshow(Input);
title('Image before processing');
% Converting the image from RGB to grayscale.
Input2Gray = rgb2gray(Input);
% Showing the converted image.
subplot(2, 3, 2);
imshow(Input2Gray);
title('Grayscale image:');
% Calculate the mode gray level using hist and max. It basically returns the
most frequent intensity value. I can either do it manually or use the mode
function, which will automatically bring the mode gray level of the image. So,
I can replace these three lines with mode = mode(Input2Gray(:)).
[counts, grayLevels] = imhist(Input2Gray); % It returns two arrays: one for
the number of pixels in each intensity level and one where we store the
different intensity levels.
[~, index] = max(counts); % It returns the index of the intensity level that
has the most number of pixels.
mode = grayLevels(index); % Retrieves the intensity level using the index we
obtained from the array count.
% Showing the image with its mode gray level.
subplot(2, 3, 3);
imshow(Input2Gray);
hold on;
plot(size(Input2Gray, 2) / 2, size(Input2Gray, 1) / 2, 'r*'); % Mark the
center.
title(['Grayscale image with mode: ', num2str(mode)]);
hold off;
% Create a binary mask that highlights pixels with intensity values deviating
from the mode of the image by a small margin of 2.
% This mask combines two arrays of intensity values: one for values lower than
the mode minus 2, and another for values higher than the mode plus 2.
% The purpose of this operation is to identify and capture pixels that are
close to the mode but not exactly at the mode.
% The resulting mask will split the image into two regions: pixels close to
the mode (within a margin of 2) will be in white, and the mode itself
(representing the background) will be in black.
```

```
mask = (Input2Gray < mode - 2) | (Input2Gray > mode + 2);
% Showing the binary mask.
subplot(2, 3, 4);
imshow(mask);
title('Binary mask');
% Performing erosion to separate touching objects using imerode.
SE = strel('disk', 5); % Creates a disk-shaped structuring element where 5 is
the radius. I have tried different structures and different radius, and a disk
with a radius of 5 gave the best results.
mask = imerode(mask, SE); % Performs the erosion where it uses the structuring
element as a template that slides over the binary mask.
% Label connected components in the mask using bwconncomp
cc = bwconncomp(mask); %It searches and labels the connected components, then
saves the information of the connected components in the structure cc.
numOfObjects = cc.NumObjects; % It gets the number of the connected
components.
% Showing the image before all the processing steps with the number of objects
in the image.
subplot(2, 3, 6);
imshow(Input);
title(['Number of objects: ', num2str(numOfObjects)]);
```

Output:

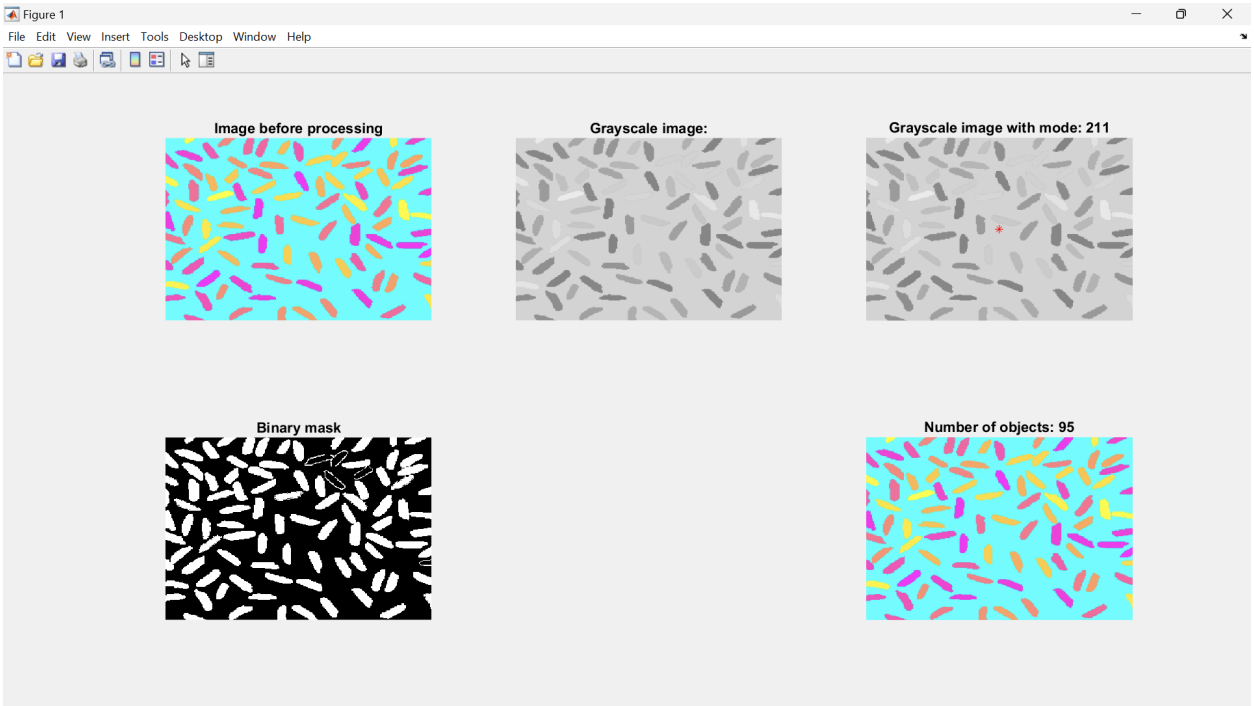

Figure 1: riceTest.png output

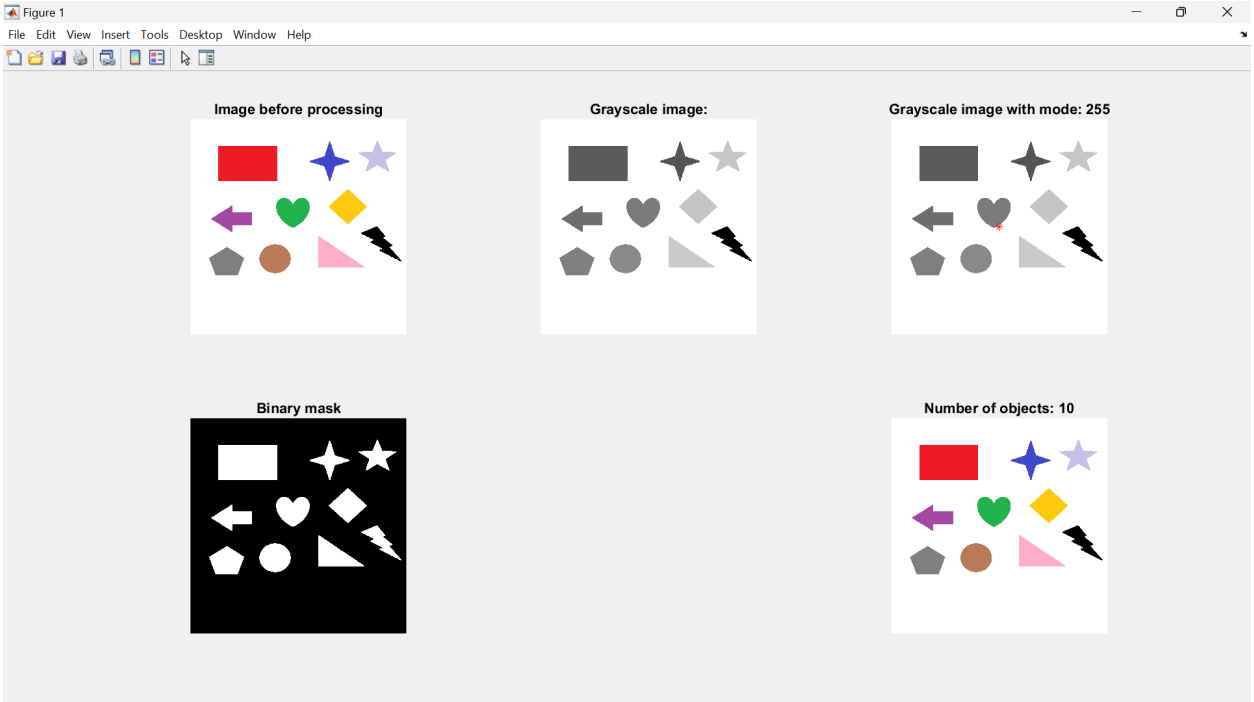

Figure 2: obj.png output

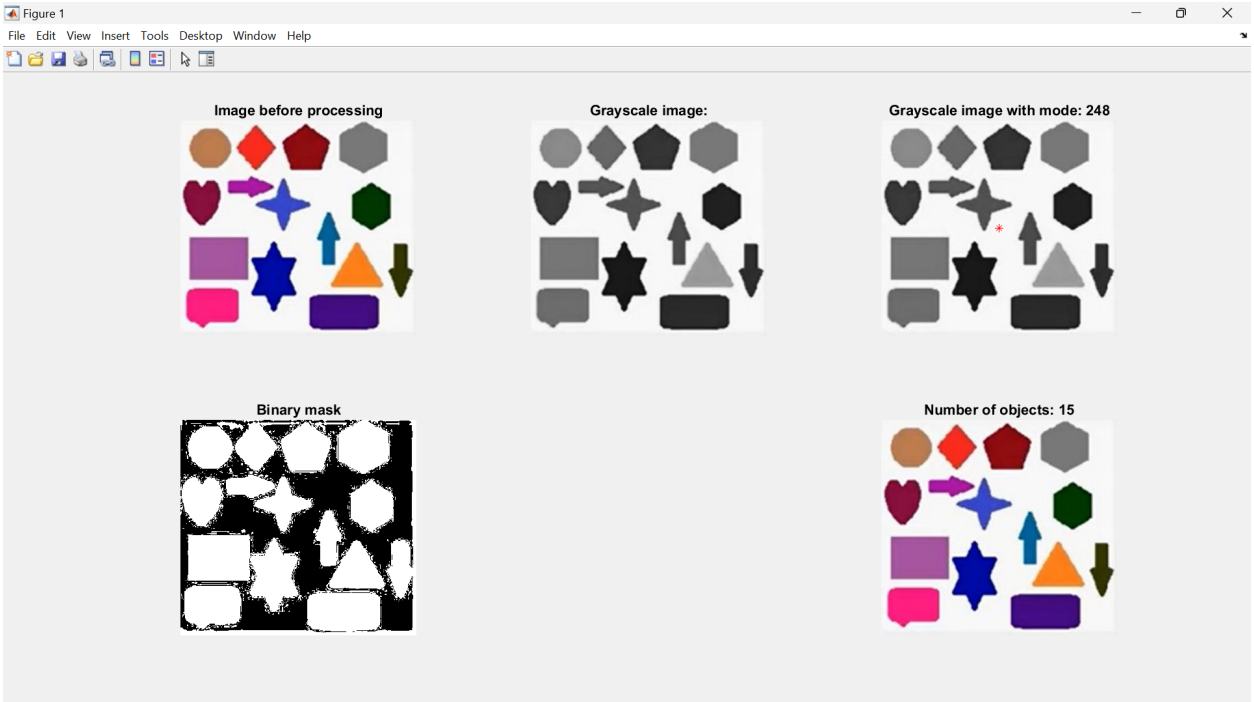

Figure 3: shapes.jpeg output

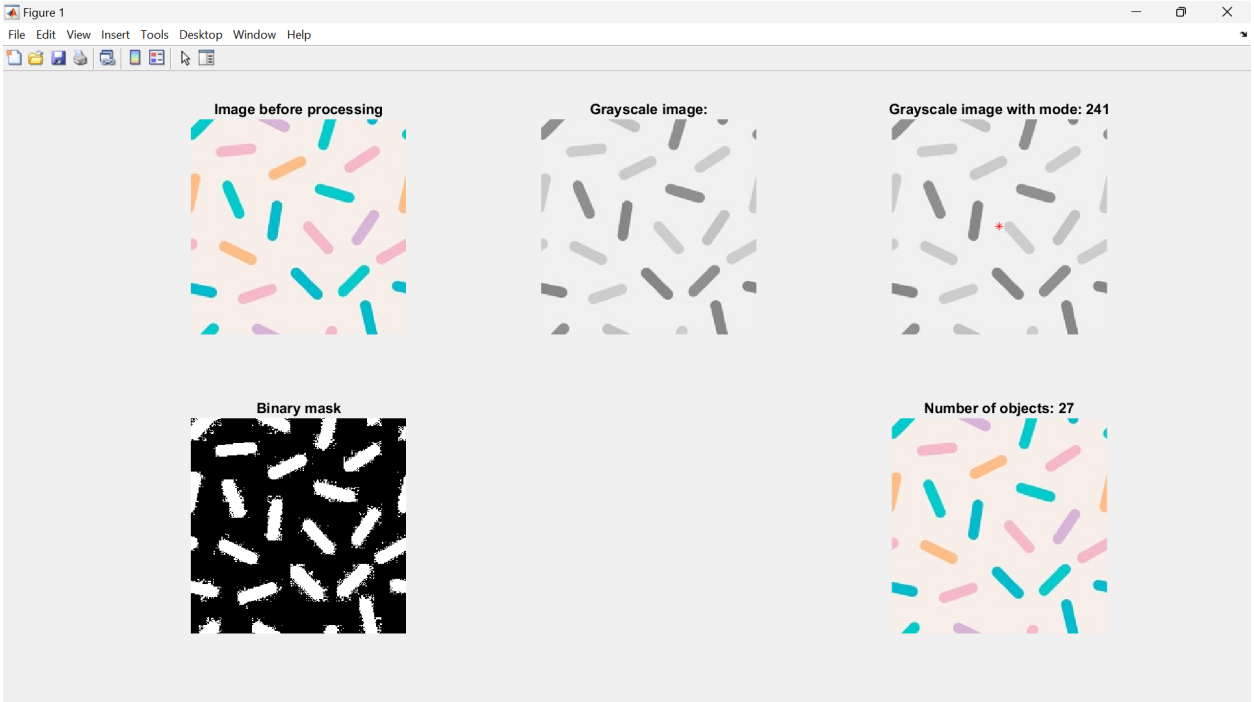

Figure 4: images.jpg

Task 2:

MATLAB Code:

```
% Read the image and convert it to a binary image so that we can distinguish
the coins from the background.
binaryImage = im2bw(imread('coins.png'));
% Subplot 1: Display the binary image
subplot(2, 3, 1);
imshow(binaryImage);
title('Binary image');
% % Filling in holes within a binary image is a vital step. It enables us to
accurately identify and analyze connected components, separate objects from
the background, and ensures that objects remain complete for further
processing.
filledImage = imfill(binaryImage, 'holes');
% Subplot 2: Display the image after filling holes
subplot(2, 3, 2);
imshow(filledImage);
title('Image after filling holes');
%The 'bwlabel' function is employed to label connected components in the
binary image.
% Each labeled region corresponds to a distinct object, and 'numObjects'
stores the total count.
% This labeling step is essential for subsequent object-specific analysis and
identification.
[labeledImage, numObjects] = bwlabel(double(filledImage));
% The 'regionprops' function is employed to extract specific properties, such
as 'Area' (representing object size)
% and 'Centroid' (providing the center of mass coordinates), for each labeled
object in the image.
% These properties are stored in a structure array ('props') and are used
later on for object identification.
props = regionprops(labeledImage, 'Area', 'Centroid');
% Initialize counters for coins of Dime and Nickel and show the original image
with coin areas
countDime = 0;
countNickel = 0;
subplot(2, 3, 3);
imshow(imread('coins.png'));
title('Image with each coins area');
hold on
% Loop through each coin's properties
for n = 1: size(props, 1)
centroid = props(n).Centroid; %gets the centroid information of the nth
element
```

```
X = centroid(1); %retrives X coordinates
Y = centroid(2);%retrives Y coordinates
% Display the size (area) of each coin
text(X - 20, Y, ['Area:', num2str(props(n).Area)], 'Color' , 'Cyan',
'FontWeight', 'bold');
end
% Subplot 4: Display the image with marked coins. After distinguishing between
the two types of coins, I will use the area of the coins to my advantage to
differentiate between them.
subplot(2, 3, 4);
imshow(imread('coins.png'));
title('Image with Marked Coins');
for n = 1:size(props, 1)
centroid = props(n).Centroid; % gets the centroid information of the nth
element
X = centroid(1); % retrieves X coordinates
Y = centroid(2); % retrieves Y coordinates
% Determine if the coin is a Nickel or a Dime based on its area.
% if its gretaer than 2000 than its a nickle otherwise its a dime
if props(n).Area > 2000
text(X - 10, Y, 'N', 'Color', 'blue' , 'FontWeight', 'bold'); % Marks Nickels
with blue N
rectangle('Position', [X - 20, Y - 20, 40, 40], 'EdgeColor', 'b', 'LineWidth',
2); % Add a blue bounding box
countNickel = countNickel + 1;
else
text(X - 10, Y, 'D', 'Color', 'cyan' , 'FontWeight', 'bold'); % Marks Dimes
with cyan D
rectangle('Position', [X - 20, Y - 20, 40, 40], 'EdgeColor', 'c', 'LineWidth',
2); % Add a light blue bounding box
countDime = countDime + 1;
end
end
totalCoins = n; % Get the total number of coins
% Subplot 6: Display the total number of coins, Number of coins for Nickles
and Dimes
subplot(2, 3, 6);
imshow(imread('coins.png'));
title(strcat('Number of Coins = ', num2str(totalCoins), ' Nickles = ',
num2str(countNickel), ' Dimes = ', num2str(countDime)));
hold off
```

Output:

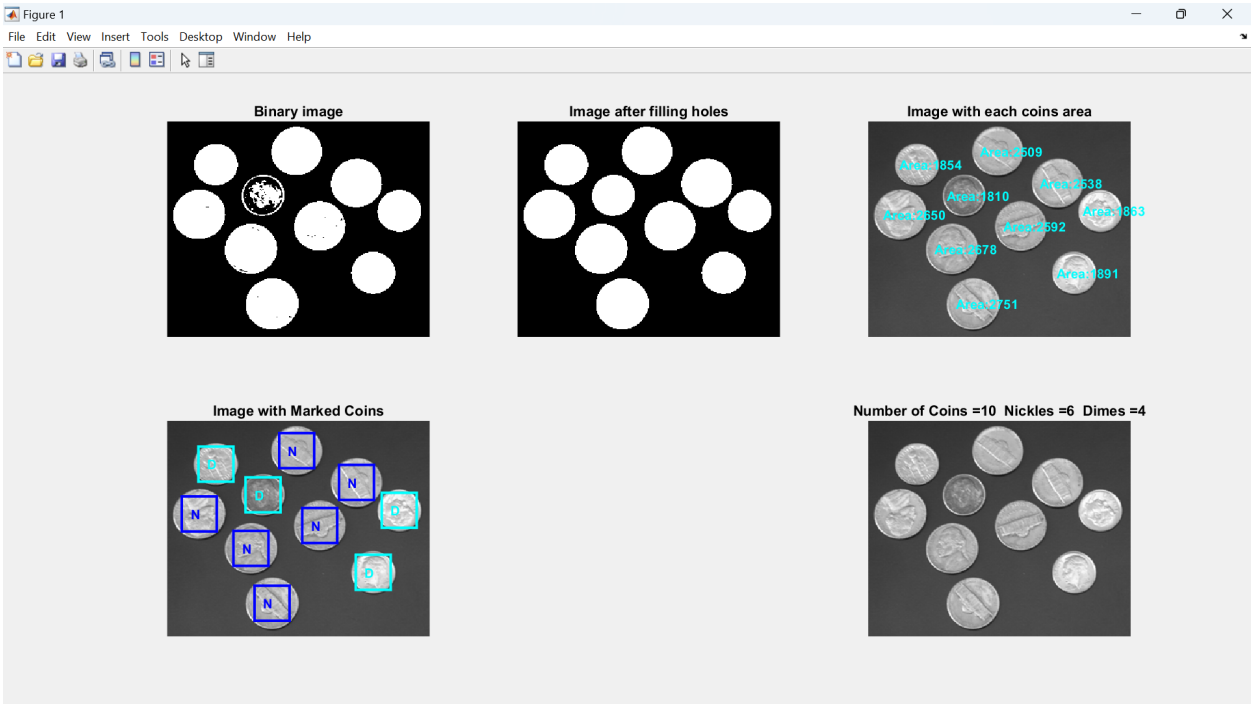

Figure 5: Displays the image after converting it to binary, filling, displaying the areas, marking them and displaying the total number of coins with number of coins for each type.