



Project Report

Coin Localization using MATLAB

Table of Contents

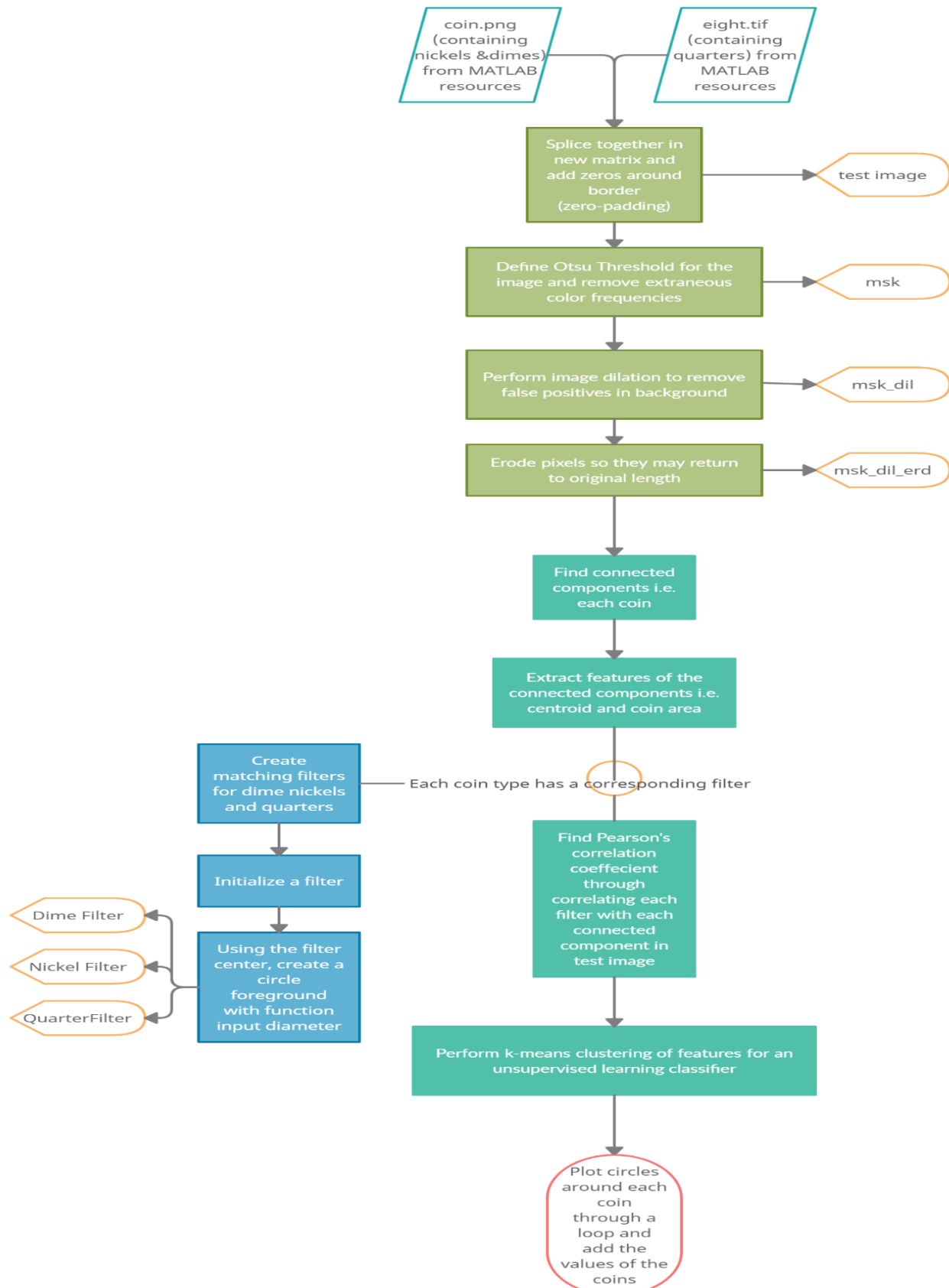
I. Abstract.....	2
II. Flow Diagram	2
III. Working	4
a. Image Synthesis	4
b. Measuring features for the coins.....	9
c. Coin segmentation	10
IV. Expected Outcome.....	11
V. Final Result.....	13
VI. References	14

I. Abstract

The main objective of our project is to use image recognition techniques to detect different coins and automatically indicate their total value. This is done through a camera placed on top of a banker's desk which captures the picture of the coins placed and passes it to the program. Thereafter, the program, whose algorithm is built using MATLAB, processes and analyzes the image and the output is a image that has different color circles, based on coin types, and the total value of the money. This mitigates the manual process of the banker having to count the coins which is prone to human errors. This report discusses the algorithm behind the project in detail so that it may be emulated in future.

II. Flow Diagram

The flow chart has been made using creately.com and may be accessible on the following link: <https://app.creately.com/diagram/ucstbtX8RhE/edit>. A picture of the chart has also been added on the next page.



III. Working

The project has been divided into 3 main parts:

1. Image synthesis
2. Measuring features for the coins
3. Coin segmentation

Each part is discussed in detail to gain a grasp of the various techniques used in getting the final output.

a. Image Synthesis

The first part of the project involves loading the built-in images in MATLAB and splicing them together to create a final matrix image that can be analyzed using mathematical functions available in MATLAB.

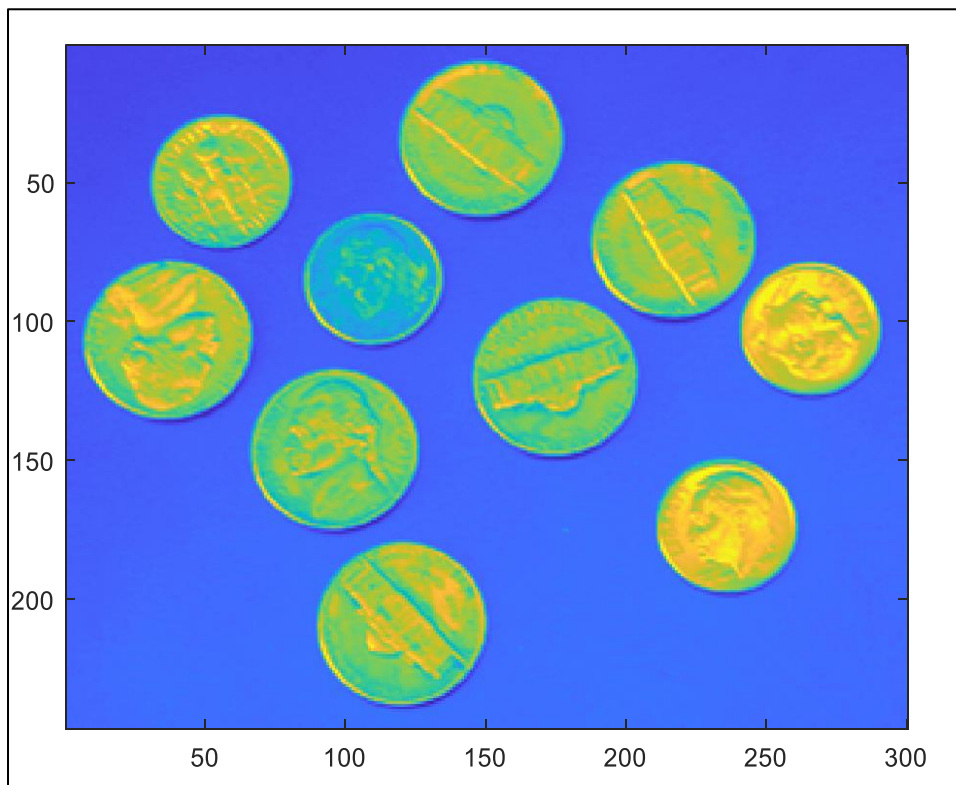


Figure 1 coins.png from the MATLAB database

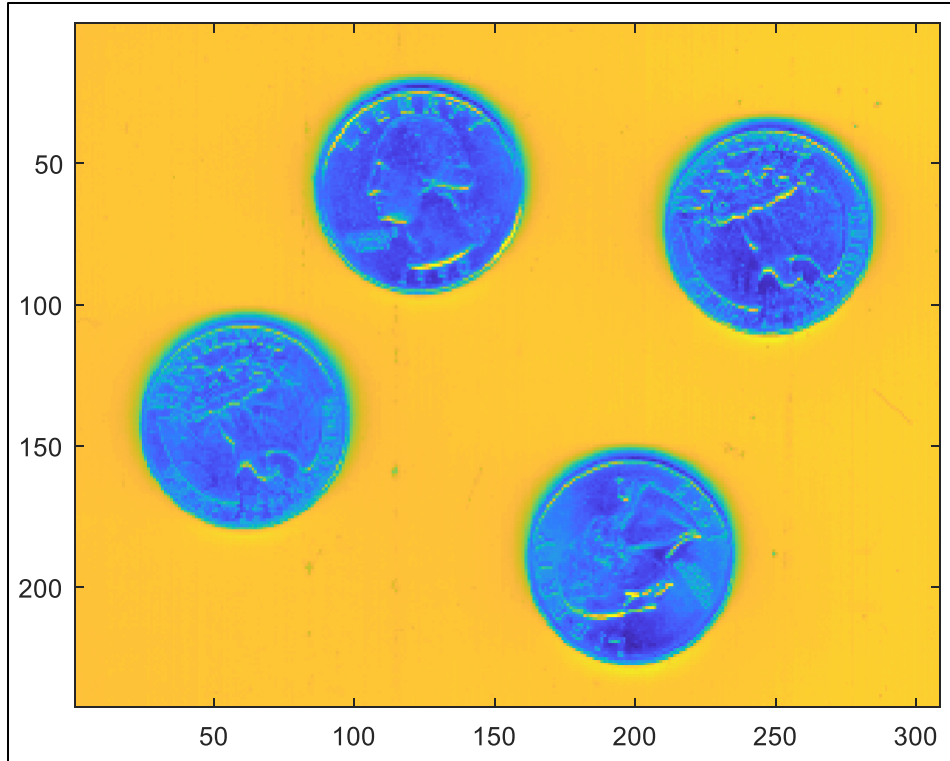


Figure 2 eight.tif from the MATLAB database

To combine these images, we create a new matrix which has the dimensions

$$r \times c$$

where $r = \text{rows of image1} + \text{rows of image2} + \text{filter size}$

$c = \text{columns of image 1} + \text{filter size} = \text{columns of image 2} + \text{filter size}$

The filter references to the filters that we will create in step 2 for the image analysis. The final test image produced is as follows (referenced as **test image** hereafter)

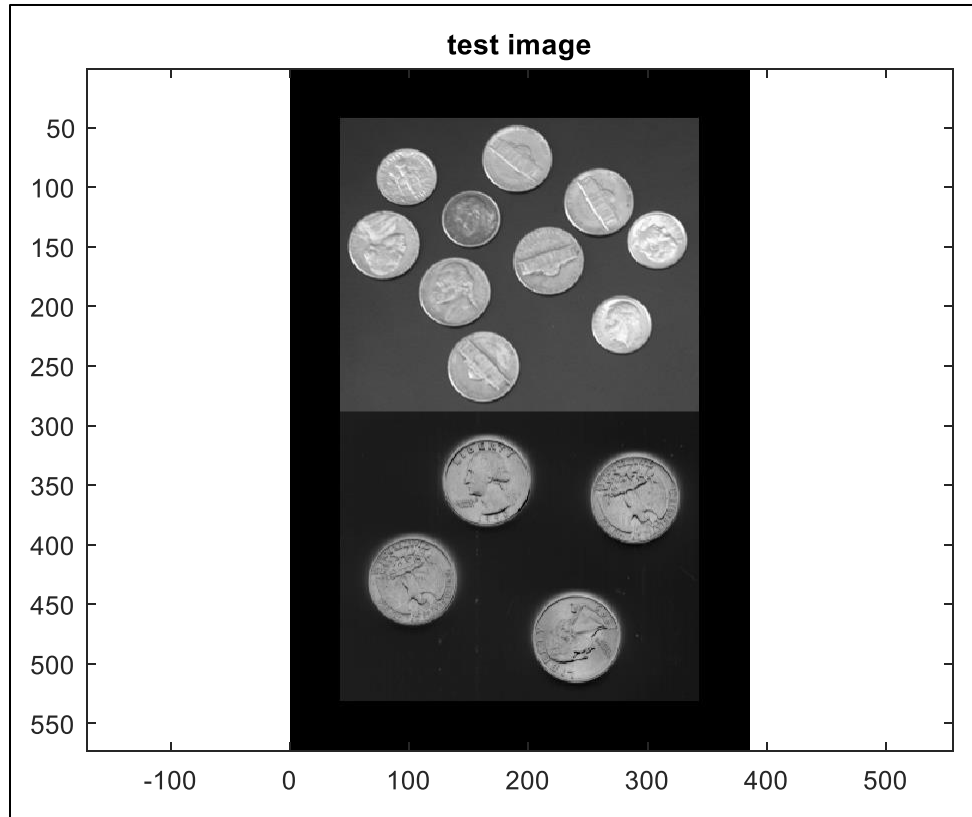


Figure 3 Final image matrix, 'im' with zero-padding added and displayed in the gray colormap

First, we perform a pixel classification method called thresholding where we partition the space in the image histogram and thus obtain two classes in the image where one class has the image pixels that have an intensity higher than the threshold value and the other class contains pixels with lower intensity values compared to the threshold value.

In this image for example, we obtain a histogram as follows:

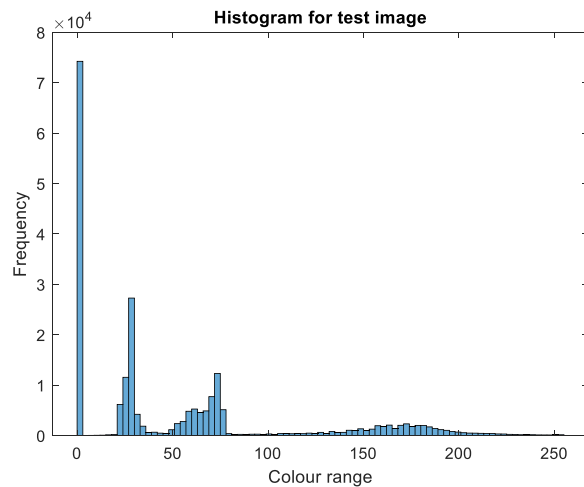


Figure 4 Histogram for the test image

Analyzing this histogram, we can see that there is a relatively high frequency of pixels in the range 0 -75 and then another contingency of pixels can be found between the range of 150 to 200. We obtain an Otsu threshold of 127. Hence, we can classify the image pixels as background and foreground and store the result as a new binary image which represents the segmentation. The binary image will now consist of white pixels representing the foreground (coins) and black pixels representing the background. This image is referred to as a mask.

This entire process is done by the `OtsuThreshold` function which we have created to make the main code simpler. The binary mask produced is as follows:

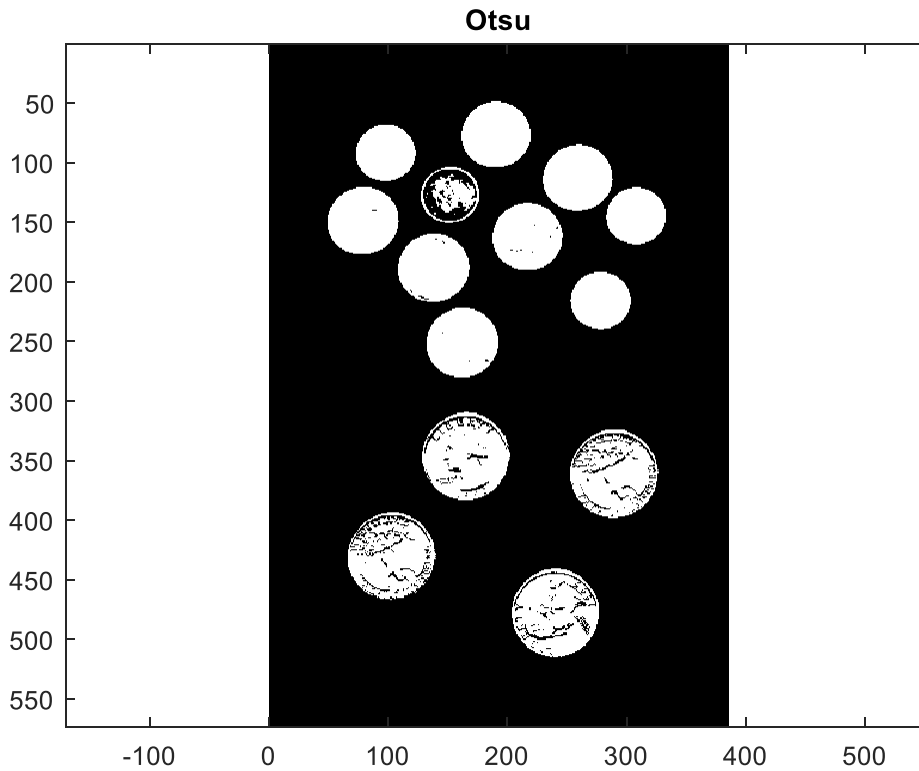


Figure 5 Segmentation mask on the basis of the Otsu Threshold

This mask image contains many false negatives (pixels of coins are represented as black) since the coins have a relief that has a relatively lower intensity in the image due to reflections on the surface. Hence, to remove these false negatives, we proceed to perform a binary dilation of the mask. The result of this operation can be effectively inferred from the resulting image after the dilation.

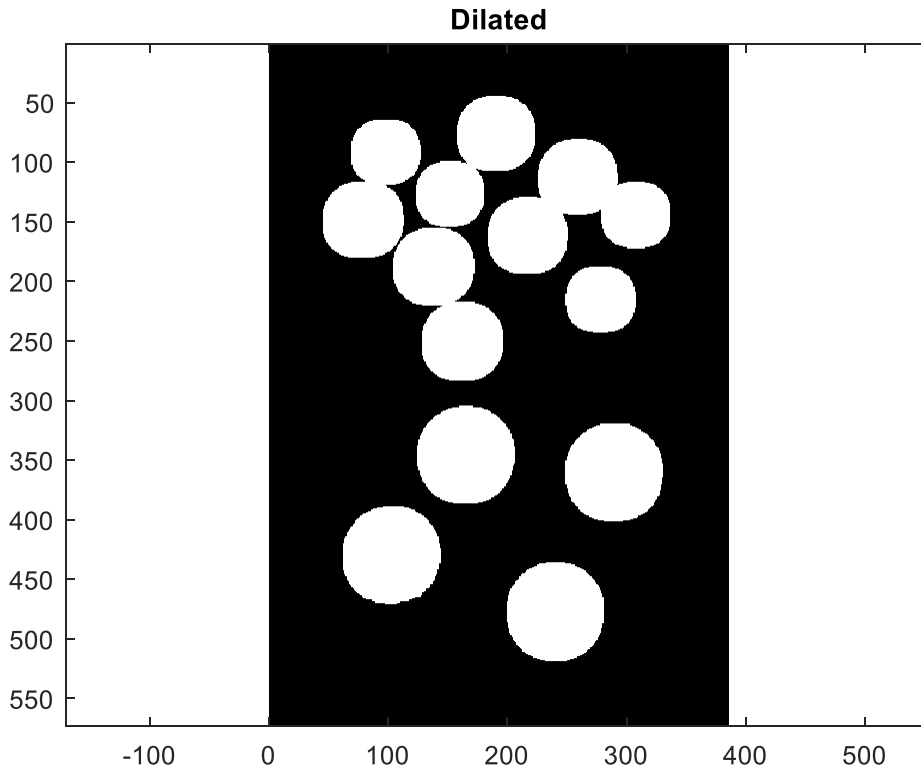


Figure 6 Dilation filtering on the segmentation mask

The noise in the image has thus effectively been removed through growing the size of the coins. Dilation flips a certain number of background pixels in the region surrounding the coins to the foreground. This number is provided by a structuring element that is given to the function as 9 by 9 matrix of 0s.

Although we have removed the noise in the image, we see that the foreground pixels of distinct coins are now connected such as the region of the group of 3 coins observed in the top right of the image above. To mitigate this, we perform our last image synthesis operation of erosion.

The `imerode` function is a complement to the dilation function. It shrinks the coin size to obtain the following image:

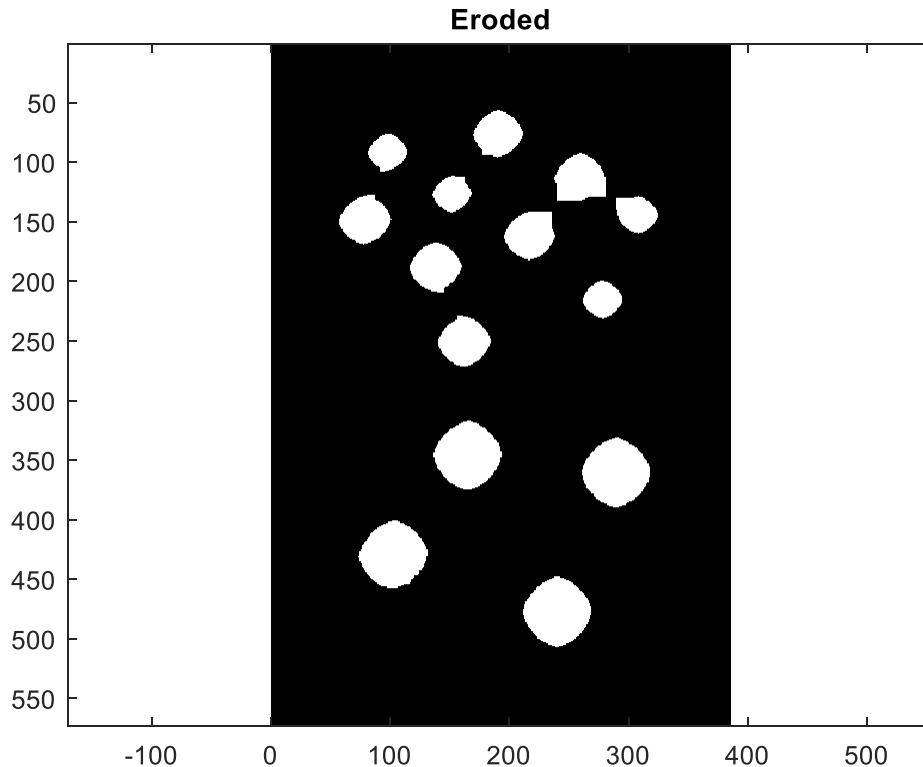


Figure 7 Erosion filtering on the dilated image

b. Measuring features for the coins

Once we have synthesized the image according to our needs, the next step revolves around obtaining the connected components of the final image and extracting some important information for those components so that we can use them in our analysis later.

Using the `bwconncomp` function on the image, we obtain a struct that contains each connected component. A connected component is the region of foreground pixels that is separated from other components with background pixels. For example, in the image above we may observe 14 connected components.

Once the connected components are found, we pass each component to the `regionprops` function which provides a set of features for each component. The features of prime interest to us are the centroid location of the coins and the coin areas i.e., the total number of pixels in each coin component.

Next, we make matching filters for each coin type. This is done through a custom `MakeCircleMatchingFilter` function which takes a certain filter size and creates a square matrix of zeroes with size $W \times W$ where W is the filter size defined in the start of the code. Each filter is then a set of background pixels with a circle foreground that has a diameter which is input in the function for dimes, nickels, and quarters separately. Hence, we obtain three different filters.

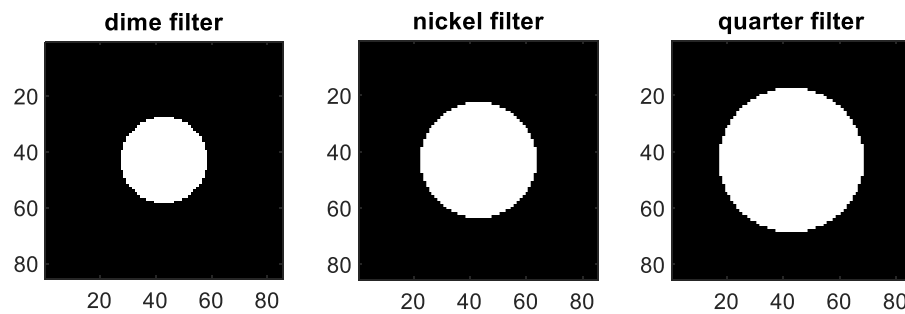


Figure 8 Matching filters created

The last process in this step is to perform a correlation test to obtain the Pearson's correlation coefficient for each connected component with each filter and store the value in a matrix. Hence, for the purpose of this image, we obtain a 14 by 3 matrix with the first column showing results of the centroids matched with the dime, the 2nd column and 3rd column representing the centroids correlation coefficient with the nickel and quarter respectively. Illustratively, the position (2,1) will contain the correlation coefficient of the second component with the dime filter to show how much these match.

The correlation technique is quite similar to convolution that we studied in this course wherein the filter is moved a certain position each time to check if the pixel on the filter is equal to the corresponding filter on the actual image.

c. Coin segmentation

The last step is based on performing a k-means clustering test to obtain classes of clusters in the image. This provides for an unsupervised learning classifier, a type of machine learning algorithm that is used to create labels of the clusters/ components found in the image based on some patterns. In this case, the labels are created through the average size of the components that we stored earlier using the `regionprops` function.

The `kmeans` function, initially creates random classes from 1 to 3 since it finds 3 different types of components in the image. We then relabel the classes so that 1 corresponds to dimes, 2 to nickels and 3 to quarters based on coin size.

In the last step, we visualize the obtained results by plotting different colored circles around each coin type and adding the corresponding coin value to the total count of the money. This is then output as the final image:

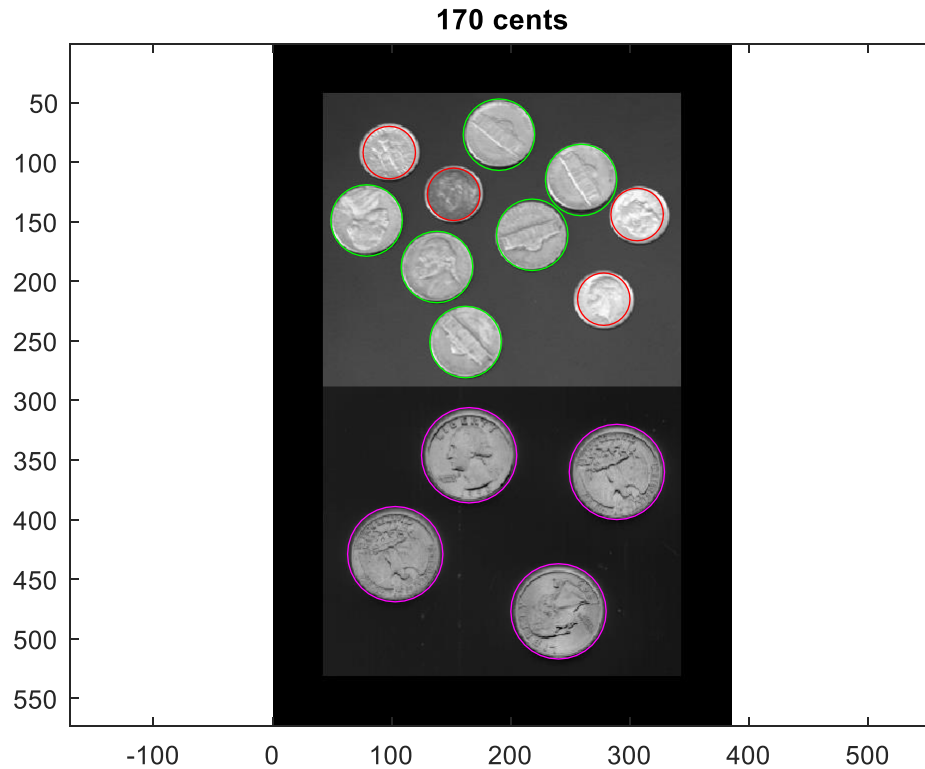

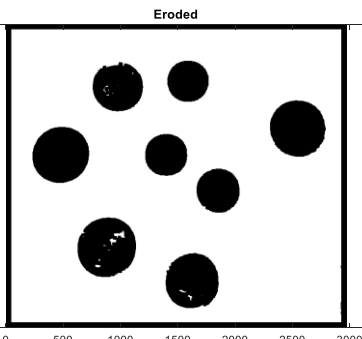

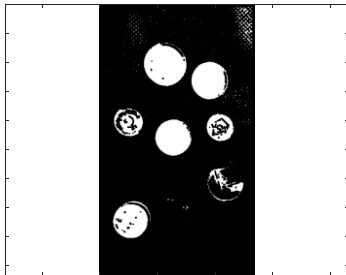

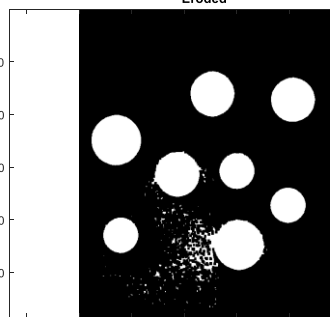


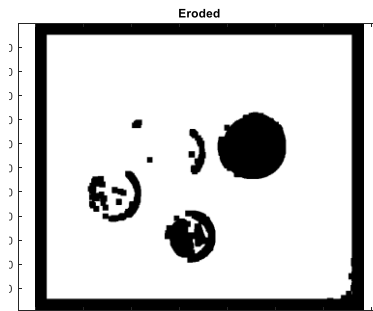
Figure 9 The final output

IV. Expected Outcome

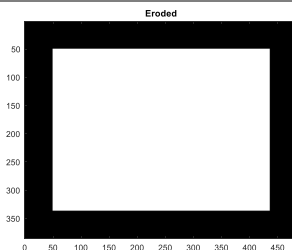
Originally, we had proposed to make a coin classifier for Pakistani coins. However, we were not able to achieve the proposed outcome in the limited timeframe of our project. After running the same algorithm on pictures that we took of the Pakistani coins (some of which you may find below), we came to the conclusion that this algorithm has certain drawbacks such as not being able to classify the image if all noise isn't effectively removed in the first part of the process.

The following table in figure 10 contains some images that we tried the algorithm on and the conclusions that we derived through it:

Original Image	Synthesized image	Conclusion
		<p>The first image was taken on a plain white paper background.</p> <p>The result is the switching of the background and foreground. However, we should have used a contrasting background so the thresholding can take place effectively</p>
		<p>The background, although contrasting was quite grainy, and it has an opposite effect to the expected noise suppression.</p>
		<p>With a contrasting maroon background, we were able to mitigate the noise to quite an extent however some false positives can still be observed.</p>



Subsequently, we made an attempt with paper as the background. However, the synthesized result is qualitatively worse than the last backgrounds.



Lastly, on the suggestion of our instructor we also tried obtaining images from Google to match the built-in images that we used from the MATLAB library. However, those were not fitting for our purpose since Pakistani coins are not well documented.

Figure 10 Different trials on Pakistani coins

V. Final Result

We took our picture of Pakistani coins on a dark green colored paper (as shown in Figure 11) and the code finally showed accurate results for an array of different pictures taken with the same background. The output for the coins was also correct, displaying a total of Rs. 41 in the test image. Hence, the conclusion we derived was that the image synthesis techniques used in the project require a dark plain background with a flash of the camera so that the light reflects off the surface to create a contrasting image with defined a background and foreground.

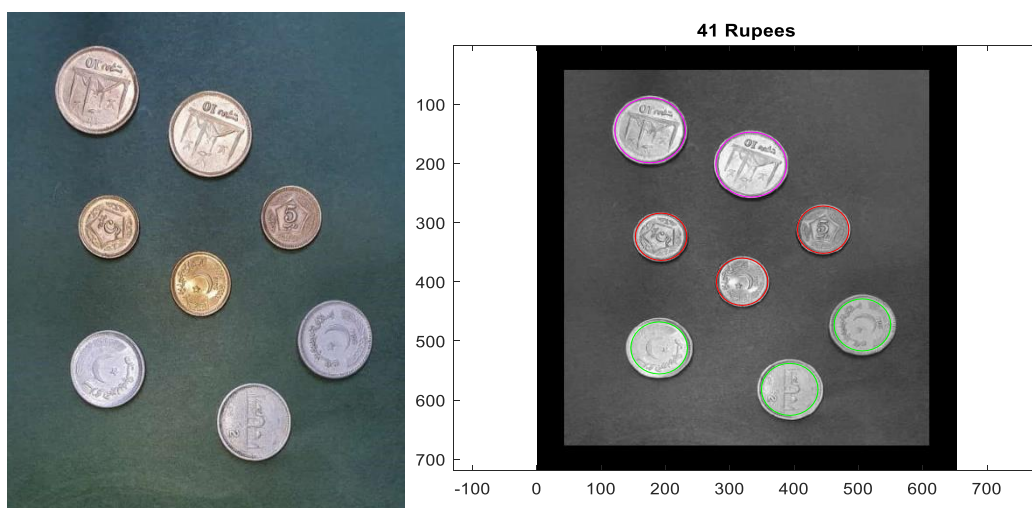


Figure 11 Final outcome of the project

VI. References

- [1] Vanderbilt University, "Introduction to Data, Signal, and Image Analysis with MATLAB," [Online]. Available: <https://www.coursera.org/learn/matlab-image-processing/home/welcome>.