Coin Localization and Classification

Mikayla Biggs & Alexander Powers
Prof. Reinhard Beichel
ECE:5480 Digital Image Processing
December 9th, 2020

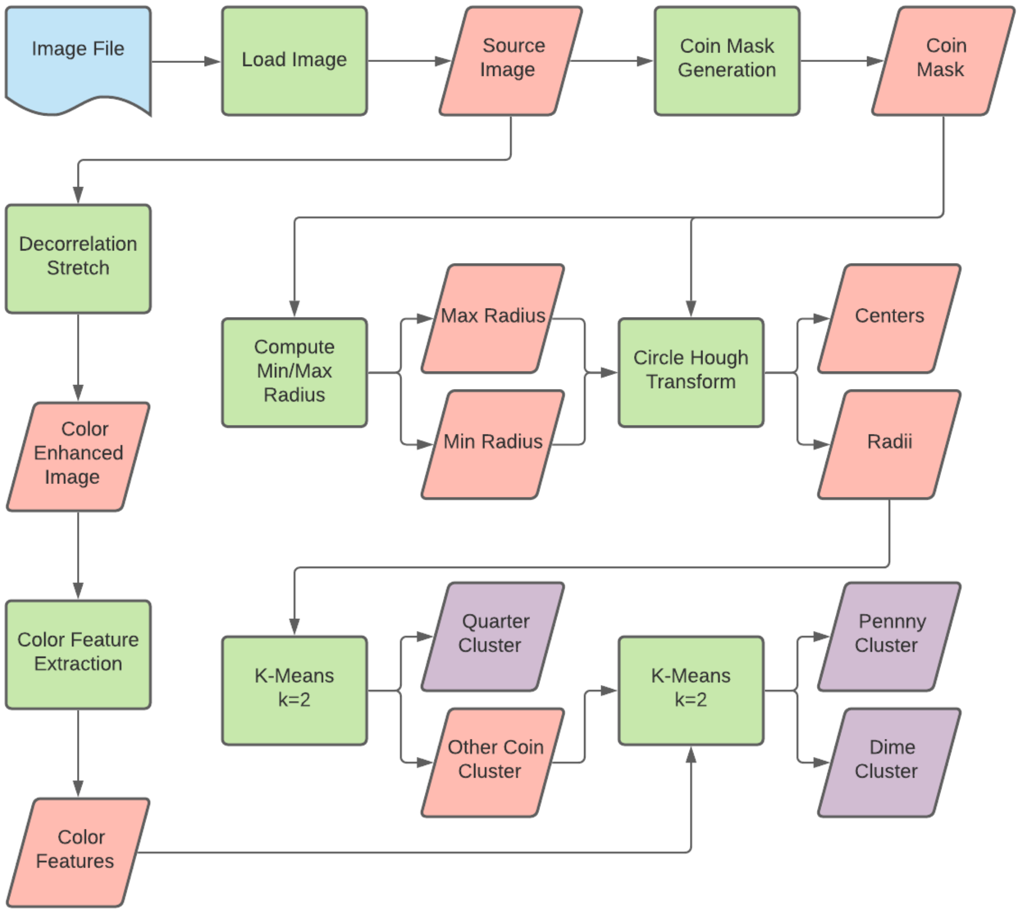
Table of Contents

1. Introduction

This project, broken into two parts, tasked us with the challenge of object localization and classification of different coins on paper. The first part, consisting of only pennies and dimes, posed non-homogenous lighting and colored line background noise issues that needed to be overcome to detect and classify the coins. For classification of the detected coins, image-derived features were to be used, such as shape and color features. The second portion of this project aimed to generalize the algorithm developed in part one and extend it to a multi-class and multi-scale problem. Meaning, the algorithm for detection and classification includes quarters and must work on both simple and complex backgrounds, all at different resolutions. Our group acquired the image dataset utilized in this portion of the project with four different challenging and simple backgrounds resampled at different resolutions to satisfy the multi-scale requirement.

2. Methods

This section details the methods used to address the proposed challenges defined above for both parts of this project.



2.1 Image Acquisition

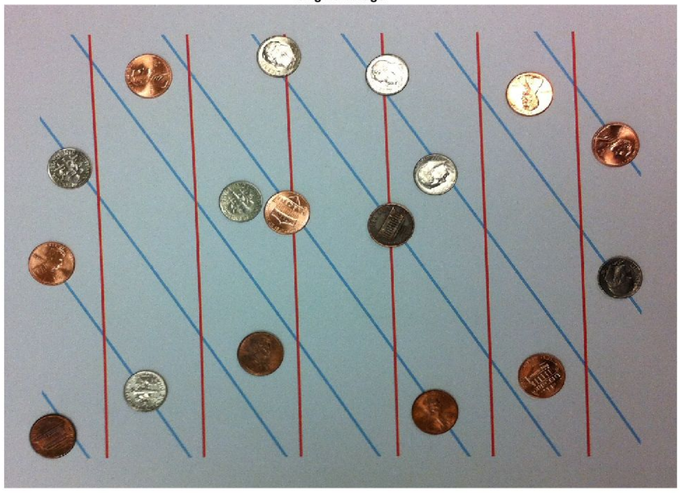
The second portion of this project required the acquisition of six new images. These images should be more challenging and have different resolutions for analysis. The images utilized in this algorithm development are three plain white backgrounds and four complex background images, all with varying arrangements of coins. These seven images were then resampled at four different resolutions to satisfy the multi-scale requirements. After this processing step, the dataset consists of 28 images, all with 15 coins present, five of each coin type for ease of analysis. The complex backgrounds include red and blue writing with a border from the countertop below the paper.

2.2 Segmentation

2.2.1 Part 1 Segmentation

The challenges posed in part 1 of this project are illustrated in the image below.

Original Image



As one can see, there is non-homogenous lighting across the whole image, causing a gradient and shine on the coins as well as red and blue lines in the background that intersect the coins. The first problem we aimed to solve was removing the background from the image and mask out the coins. To do this, we decided to take advantage of the fact that the coins will have a strong edge in all three color channels (RGB) while the red and blue lines will only present in the red and blue channels, respectively. Our method in doing so is shown in the code snippet below.

```
for channel=1:3
    I_edge = edge(I(:,:,channel), 'sobel');
    I_close = imclose(I_edge, disk5);
    morph_data(:,:,channel)=I_close;
end
intersected_edge_masks = morph_data(:, :, 1) & morph_data(:, :, 2) &
morph_data(:, :, 3);
```

The two main methods used in this step are the Sobel edge detector and the morphological closing operation. The Sobel edge detector performs a 2-D spatial gradient on each color channel to emphasize regions of *high* spatial frequency, i.e., areas of the most significant change in pixel intensity, which defines an "edge." The operator uses a pair of 3x3 convolution kernels that respond maximally to horizontal and vertical edges due to the perpendicular orientations. However, due to this fact and the coins having curved edges in all directions, we will have disconnected coin edges. To connect these edges, we applied the morphological closing operation.

Morphological operators utilize a structuring element to operate on some defined neighborhood. This structuring element establishes the shape and size of the neighborhood to apply the operation within, i.e., closing or opening. The goal of these operations is to help define solid objects within an image. A closing operation is applying a dilation and erosion operation on the neighborhood where the purpose of dilation is to "swell" the region and erosion just the opposite. The closing operation fills strongly connected components, connecting any gaps. The closing operation is useful to connect any gaps in the edge images of our coins. After the edge images of each channel are obtained, we apply an AND operation to keep only the edge signals that appear in all three color channels. This effectively removes the red and blue lines as they only appear in the red and blue channels, respectively.

Now that we have obtained an edge image for the coins, the next step is to generate a coin mask to be passed into the Hough transform. To do so, we simply used more morphological operators to fill in the regions within the edges of our coins, as shown in the coin snippet below.

```
i_dil = imdilate(intersected_edge_masks, disk10);
i_fil = imfill(i_dil, 'holes');
i_err = imerode(i_fil, disk10);
i_open = imopen(i_err, disk15);
```

The dilation thickens our edges. The fill operation fills in all the pixels within a region with a connectivity of 4. The erosion fills any remaining holes while reducing the coin boundary that we originally thickened to preserve the coins' true size. This mask can now be passed into a Circle Hough Transform using Matlab's imfindcircles() function. A Circle Hough Transform is a variant of the Hough Transform used to detect circles in either a two or three-dimensional parameter space depending on whether the radius is known. The Hough Transform, in general, works by solving a dual problem in a separate parameter space, sometimes called the Hough domain or dual plane. For circle detection on circles with an unknown radius, we have to search three-dimensional parameter space defined by: (a, b, r). Where (a, b, r) corresponds to the circle of the equation: $(x-a)^2 + (y-b)^2 = r$. Because our radius is unknown, we have to search a three-dimensional parameter space (a, b, r) rather than the two-dimensional subspace (a, b) that we would if we knew the radius of the coins. The actual algorithm for computing the circles amounts to iterating through the image space points and computing circles of various radii in the Hough space. The Hough space coordinates that have the greatest activation represent the centers and radii of the circles in the image space.

The imfindcircles() function provides the centers and radii for all detected circles in our coin mask. These two metrics allowed for visualization of our object localization and created a label mask for each coin in the image. These metrics would also prove useful for the coin classification portion of this project, to be discussed later in this report.

2.2.2 Part 2 Segmentation

For the second part of this project, two stages of algorithm development will be discussed in detail. The first was the algorithm designed to handle any complex background image input coupled with a second algorithm to handle images with simple backgrounds. However, it was later clarified that utilizing two different algorithms should not be the final solution. Hence, the second algorithm that will be discussed is the final algorithm developed to handle both simple and complex. The original intention was to add a background classification step to branch into either algorithm based on the classification. Still, it was challenging to find a metric that would be independent of scale, so that path of development was abandoned.

The simple background algorithm is relatively simple. First, the RGB image is converted to HSV. We found that the saturation channel of the HSV images contained the strongest representation of the coins on the paper. The edge image is then taken of only the saturation channel. A similar methodology used in part 1 for dilating the edges, filling in the connected components, and finally eroding the added pixels is applied to the edge image. This generates the coin mask that is later passed into the Circular Hough Transform detailed in 2.1.1. An additional step of using bwperim(), which creates another edge image, was added to handle cases where two coins are very close or even touching. This function can separate the two, and then another filling operation is applied to generate the final simple coin mask. The full method is defined in the code snippet below.

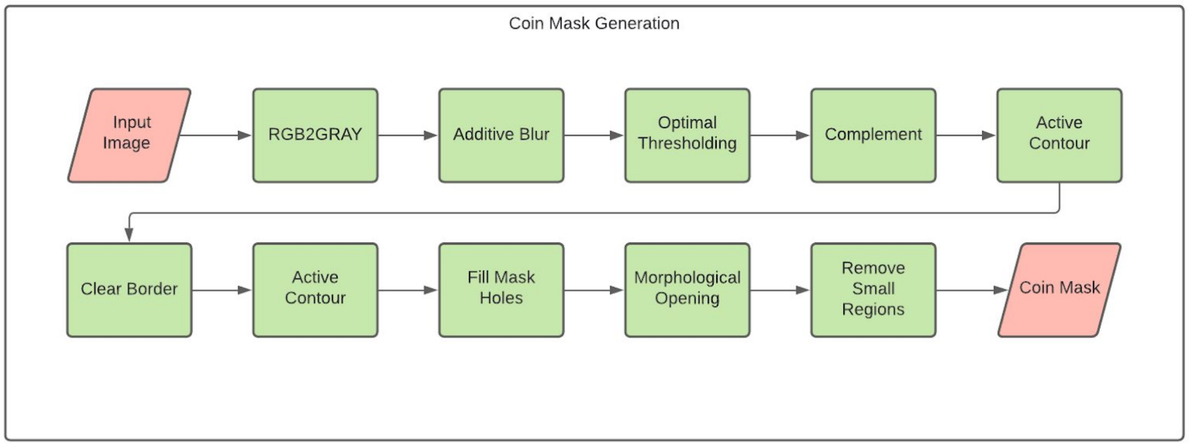
```
I_hsv = rgb2hsv(I);
edge_image = edge(I_hsv(:,:,2));
strelly = strel('disk', 5);
edge_image_dil = imdilate(edge_image, strelly);
edge_image_fill = imfill(edge_image_dil, 'holes');
edge_image_err = imerode(edge_image_fill, strelly);

perim = bwperim(edge_image_err, 8);
coin_mask = imfill(perim,'holes');
```

The complex background algorithm is slightly more complicated. First, the image is converted to HSV and grayscale. We found that the hue channel contained all of the background noise without the coins for the HSV image, while the saturation channel contained both. To leverage this fact, the hue channel is then binarized to later assist in background removal, and a large median filter is applied to the grayscale image to blur out as much background noise as possible. The pixels where the background signal is found in the hue channel then replaces those same pixels in the original unfiltered grayscale image. This removes the harsh lines of the background while preserving the sharp edges of the coin objects. To address the challenge of having the counter-top in the images, the function imclearborder() is applied to the binarized grayscale image. After a morphological filling operation is applied, the coin mask is mostly complete. However, the steps above are still sensitive to small background objects. A two cluster K-Means clustering algorithm is applied to delineate small background objects and the larger coins to address this. The MATLAB function regionprops was utilized to find the area metrics for all detected objects, which are then used as the features that are passed into K-Means. While this identifies which objects are similar, the background objects still need to be removed from the coin mask. To accomplish this, the larger cluster's mean and standard deviation are used to generate upper and lower area thresholds for objects of interest. The thresholds are expanded in either direction by two standard deviations to account for both small and large coins and passed into the bwareafilt() function along with the coin mask to remove all objects that do not fall within those bounds. This finalizes the methodology for complex image segmentation that will then be passed into the Circular Hough Transform for object detection. The code snippet below details this method.

```
I_hsv = rgb2hsv(I);
  I_gray = rgb2gray(I);
  I_hue = imbinarize(I_hsv(:,:,1));
  I_filt = medfilt2(I_gray, [250 250]);
  I_gray(I_hue) = I_filt(I_hue);
  I_bin = imclearborder(imcomplement(imbinarize(I_gray)));
  I_morph = imfill(I_bin, 'holes');
  obj_areas = struct2array(regionprops(I_morph, 'area'))';
  obj_classes = kmeans(obj_areas,2);

  c2_mean = mean(obj_areas(obj_classes==2));
  c2_std = std(obj_areas(obj_classes==2));
  c2_UT = c2_mean + 2*c2_std;
  c2_LT = c2_mean - 2*c2_std;

  coin_mask = bwareafilt(I_morph, [c2_LT c2_UT]);
```

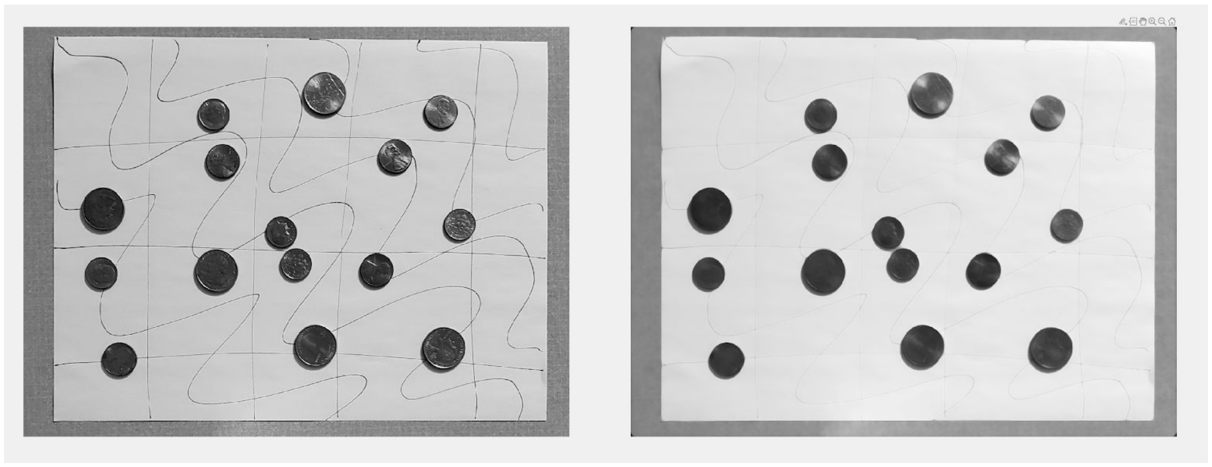
2.2.3.2 Final Segmentation Pipeline

Rather than attempting to compute a meaningful metric for all image resolutions to branch between complex and simple backgrounds, as discussed above, we decided to create a single, more generalized approach that worked for both background types. This pipeline leverages the fact that the coins are generally the largest circular objects in the image by utilizing a series of blurring levels to remove thinner components that were not relevant to the detection problem.

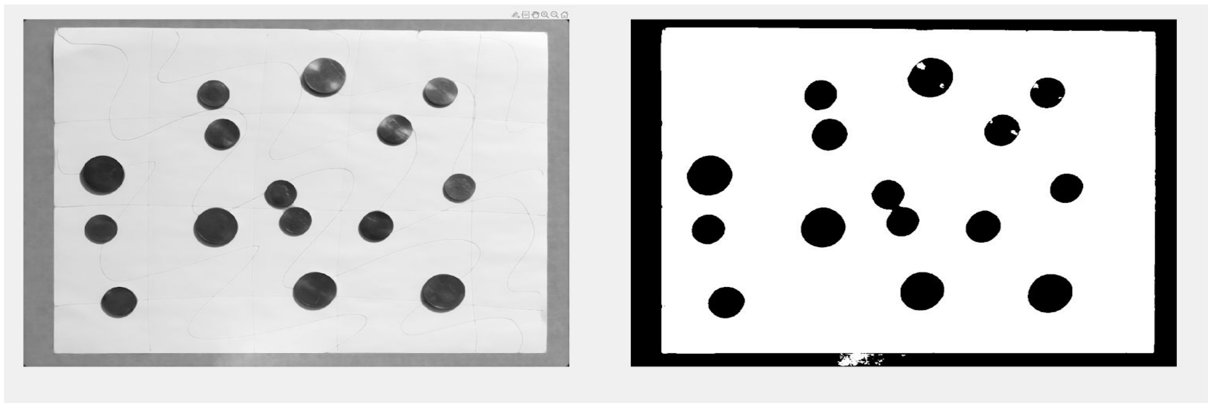


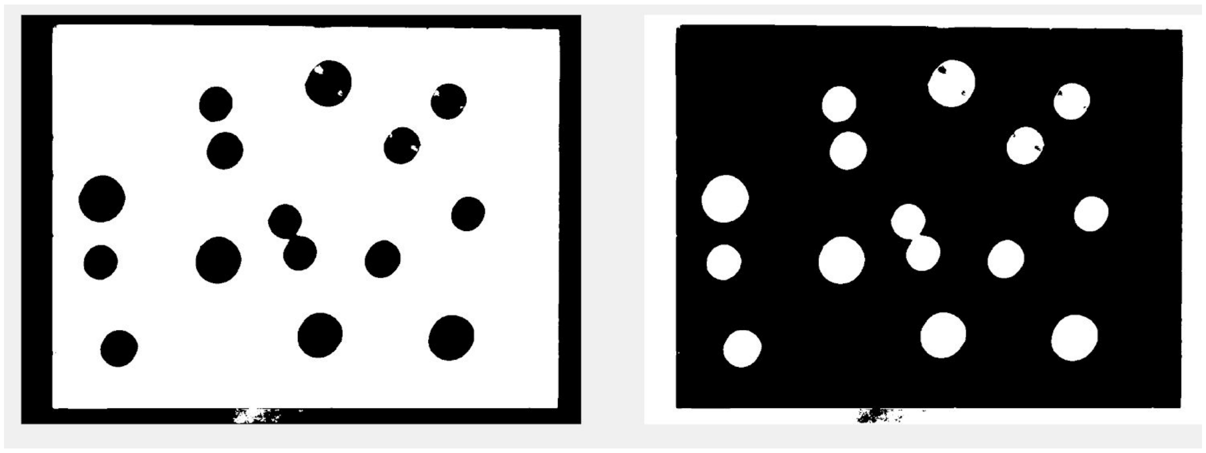
The pipeline above outlines how each coin mask was generated. This series of steps solved a variety of problems that we ran to throughout the development process. This section will discuss each process in this pipeline and the motivation for its inclusion.

This first step in this pipeline is creating a blurred image from the grayscale version of the source image by successive addition of median filtered images with larger and larger kernel sizes. By combining images with different levels of sharpness, we retain some of the information for smaller blurring levels while still removing lots of the background, which isn't represented in most of the additions. The result of this process is pictured below.

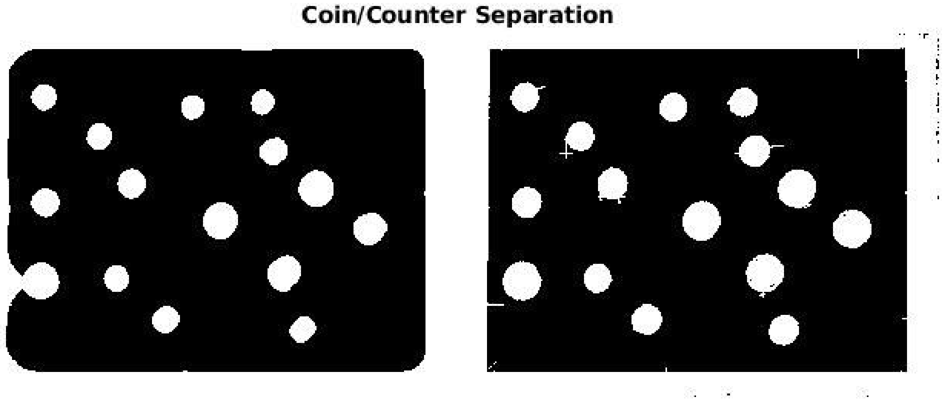


Next, we perform the same optimal thresholding that was discussed and implemented in homework nine. This step generates a segmentation of the paper, which we then invert to create a mask representing only the countertop and the coins. These two steps are shown below.





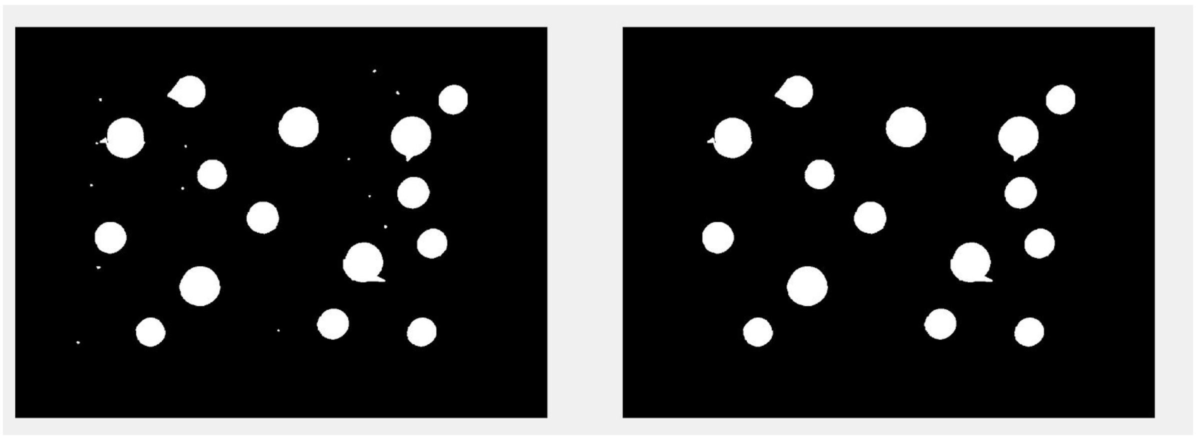
At this point, we detected a common problem in some images. We saw that coins close to the paper's edge were connected to the countertop when the image was inverted. To separate these coins from the countertop mask, we utilized active contour models. Active contour works by minimizing the energy of a spline representation of the masks. In application to our problematic images, it had the following effect.



Now that we had isolated the coins, we removed the component of the mask representing the countertop by clearing all mask components connected to the edge of the image by an eight neighborhood operator. We then ran active contouring one more time for good measure to ensure that we had dilated to masks to represent the entire area of the coin effectively.

This algorithm's last steps were primarily focused on the cleanup of smaller bits of mask that were caught by the optimal thresholding algorithm. First, we filled all of the holes in our masks so that the following morphological opening operation would not accidentally open up an edge of a coin and make it more difficult to be detected by the hough transform. Lastly, we removed some smaller flecks from our image using an area-based k-means clustering.

The use of clustering to remove smaller 'flecks' or unwanted masks was based on some issues we were having in the larger resolution images. We remove dots of a certain size based on the clusters that we generate. Our method clusters all mask regions based on their area into two groups, a large area cluster and a small area cluster, which is then used to compute a minimum pixel area. We then take the difference between the smallest area in the large group and the largest area in the small group and use that as a cutoff for a mask's acceptable size. All regions smaller than this size are removed from the image. This algorithm is specifically designed to preserve coins, as long as they are clustered into the 'large area' cluster. The result of this last cleaning step is pictured below:



2.3 Classification

This section details the methods used for the image-based feature extraction and classification for both parts of this project.

2.3.1 Part 1 Classification

2.3.1.1 Image-based Feature Extraction

The segmentation portion of this algorithm resulted in a list of radii that were the output from the Circular Hough Transform function imfindcircles(). These radii describe the relative size of the coins on the paper, and it is reasonable to assume that for our two classes, pennies and dimes, the radii for each class will be similar, if not equal. With this assumption, we chose to use an unsupervised clustering algorithm where the radii obtained from the Hough transform are the image features.

2.3.1.2 Coin Classification

The unsupervised clustering algorithm we chose to use was K-Means clustering. K-Means is an iterative algorithm that aims to partition the image features into k distinct groupings while minimizing the intra-cluster variation. First, k cluster centroids are initialized using a random seed. Each instance is assigned to the cluster with the "closest" mean, allowing the algorithm to cluster similar instances together around each centroid. For our problem, we used two clusters, one for each class. The coin radii were then grouped based on the similarity of their radii, and the cluster label is used as our class labels, illustrated in the code snippet below.

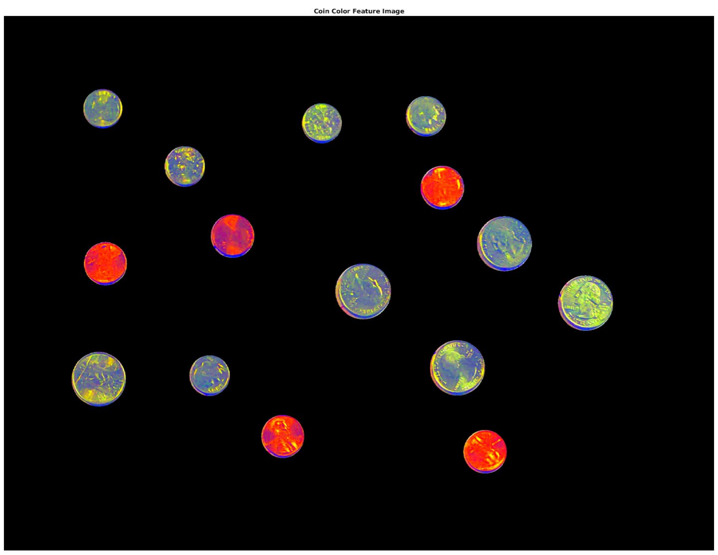
```
k = 2;
thresh_labels =  kmeans(radii, k);

class_1 = radii(thresh_labels==1);
class_2 = radii(thresh_labels==2);

mu_1 = mean(class_1);
mu_2 = mean(class_2);

if mu_1 > mu_2
    class1_name=sprintf('pennies');
    class2_name=sprintf('dimes');
else
    class1_name=sprintf('dimes');
    class2_name=sprintf('pennies');
end
```

Using the means of the radii for each class, we could then determine which cluster contained pennies or dimes based on which cluster had the smaller mean radii (dimes).

2.3.2 Part 2 Classification

2.3.2.1 Image-based Feature Extraction

The added challenge of the addition of a larger coin is that clustering on the coin radii only would result in higher misclassification between pennies and dimes. For this reason, color-based image features are extracted for both RGB and HSV images in addition to the radii feature. The coin radii features are found in the same manner as described in part 1 by using the radii values returned from the Circular Hough Transform; however, in this method, the radii are normalized before being added to the feature space to improve the feature discrimination between the copper pennies and silver dimes, decorrelation stretching is applied to the input image. Decorrelation stretching enhances the color separation of an image, exaggerating the color bands (i.e., RGB). Suppose each of the bands of pixels in the original image that defines a color is treated as a coordinate in space. In that case, decorrelation stretching moves those points in space farther apart, making it easier to see a difference between them. For example, if the original image's red and green channels are similar, the decorrelation stretching will make them appear more dissimilar and exaggerate their differences. This allowed for the exaggeration of the red pixels in the copper coins to better distinguish them from dimes, as shown in the image below.



The image above is then used to generate an additional HSV image. The coin mask generated during segmentation is then applied to the color images. The feature calculations are only on the coins themselves, excluding background colors, like in the image above. Then, for *each* coin region, the average color channel intensity is calculated for all R, G, B, H, S, and V channels resulting in a seven-dimensional feature space when radii are added. The code snippet below details the feature extraction steps described.

```
I = decorrstretch(I);
I_hsv = rgb2hsv(I);
n_coins = size(mask,3);
features = ones(n_coins,7);
features(:,7) = radii ./ max(radii);

for coin_idx=1:n_coins
    % rgb features
    color_coin_mask = I .* uint8(mask(:,:,coin_idx));
    for channel_idx=1:3
        channel = color_coin_mask(:,:,channel_idx);
        channel_sum = sum(channel, 'all');
        channel_elem_count = sum(uint8(mask(:,:,coin_idx)),'all');
        features(coin_idx, channel_idx) = (channel_sum / channel_elem_count)/256;
    end

     % hsv features
    color_coin_mask = I_hsv .* double(mask(:,:,coin_idx));
    offset = 3;
    for channel_idx=1:3
        channel = color_coin_mask(:,:,channel_idx);
        channel_sum = sum(channel, 'all');
        channel_elem_count = sum(uint8(mask(:,:,coin_idx)),'all');
        features(coin_idx, channel_idx+offset) = (channel_sum / channel_elem_count);
    end
end
```

2.3.2.2 Coin Classification

The final classification pipeline to be detailed is 2-fold: classify quarters and “others” with radii features, classify pennies and dimes with additional color features. Once again, an unsupervised K-Means clustering algorithm is used to group the coin features for classification, as detailed in section 2.2.1.2.

The first stage is to identify the larger quarters from the smaller pennies and dimes. A two cluster K-Means algorithm is used to accomplish this, and the cluster with the largest mean is said to be the quarter cluster. With the quarter objects identified, they can be removed from the coin mask so that the second stage of classification only considers pennies and dimes.

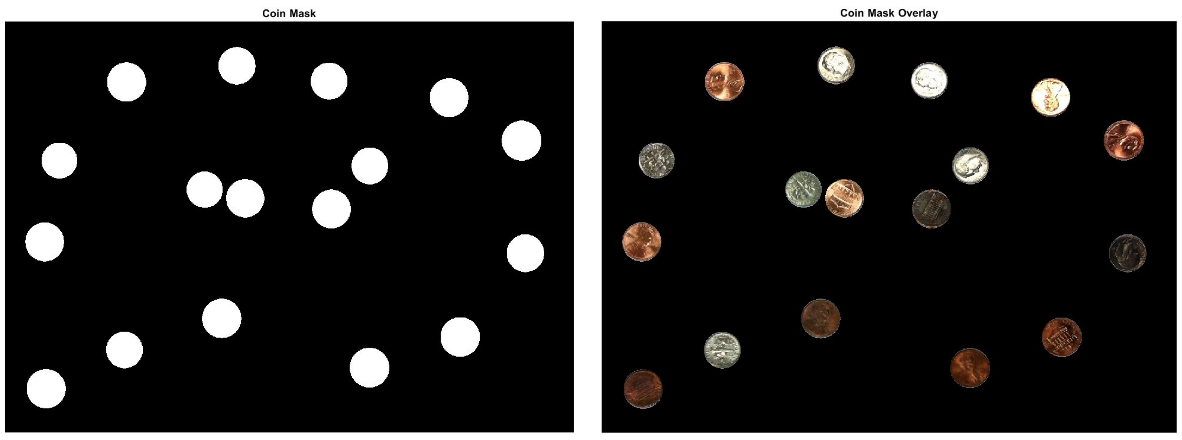
The second stage extracts the color features described above for all of the pennies and dimes detected in the input image and adds them to the feature space. Then, another two cluster K-Means algorithm is used to group the instances. The cluster with the smallest mean is said to be dimes and the other to be pennies.

3. Results

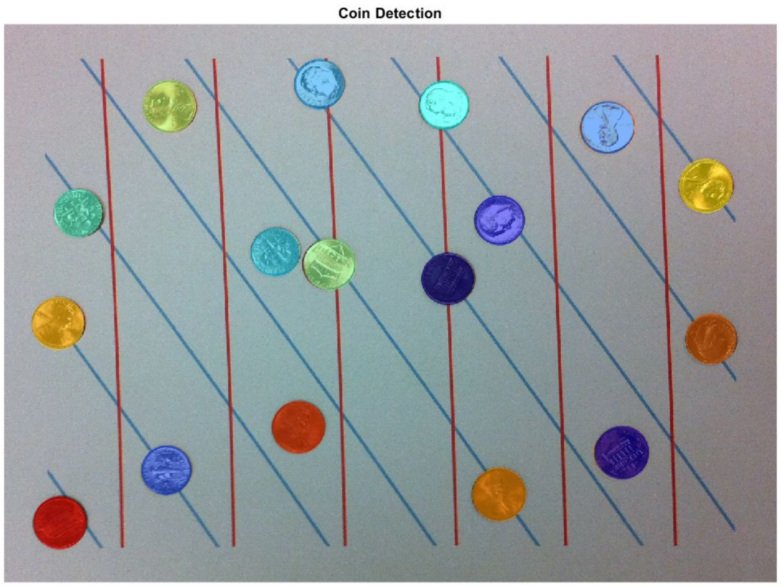
3.1 Segmentation

3.1.1 Part 1 Object Localization

The segmentation methodologies described in section 2.1.1 aimed to remove the background of the image and create a mask for all of the coins on the page. The results of these methods on the challenge image provided in Part 1 are shown below.

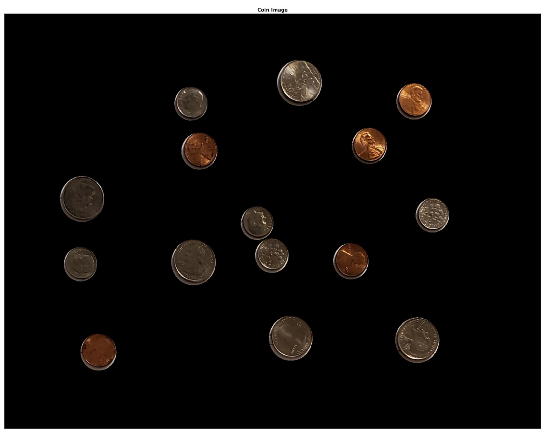


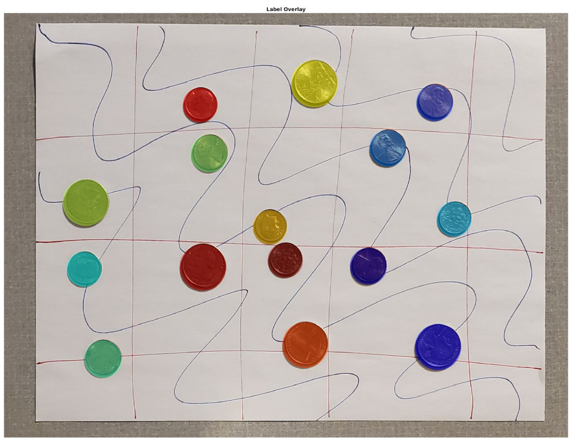
The methods were able to successfully detect all of the coins on the page without gaping around edges or lost coin area. The mask was then used to generate a label mapping of all of the individual coins detected where one unique label is given to each coin. This mapping is shown below, where the unique label is the color of the coin.



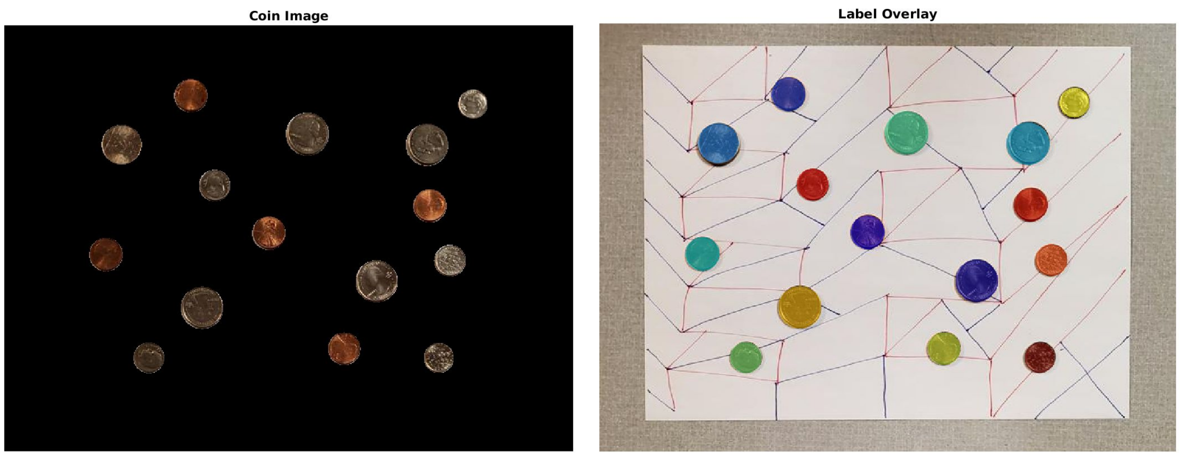
3.1.2 Part 2 Object Localization

Our method successfully detects 419/420 coins across four different resolutions of all seven images. The object localization is successful with both simple and complex backgrounds at all resolutions. The images below show six different images, three complex backgrounds, and three simple, all at different resolutions.

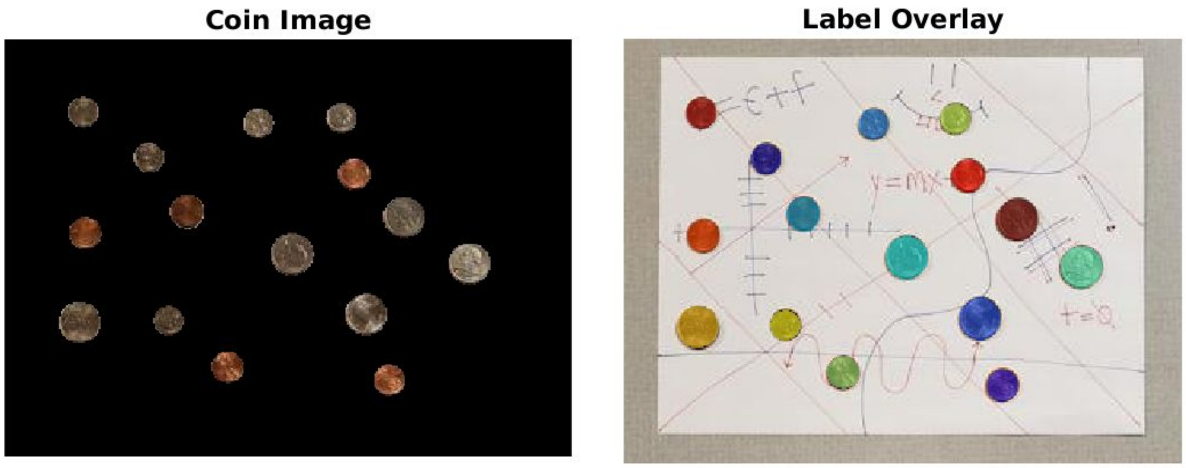




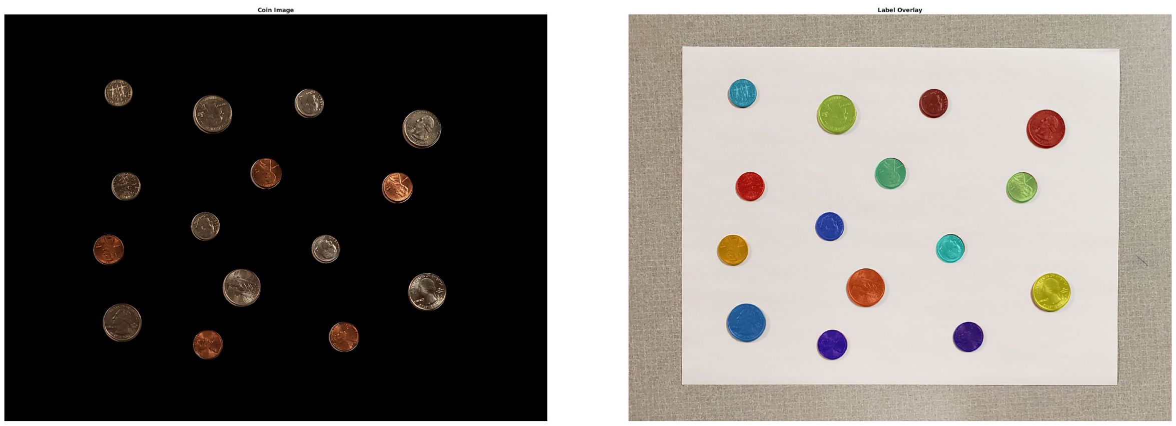
1-complex high-resolution complex background



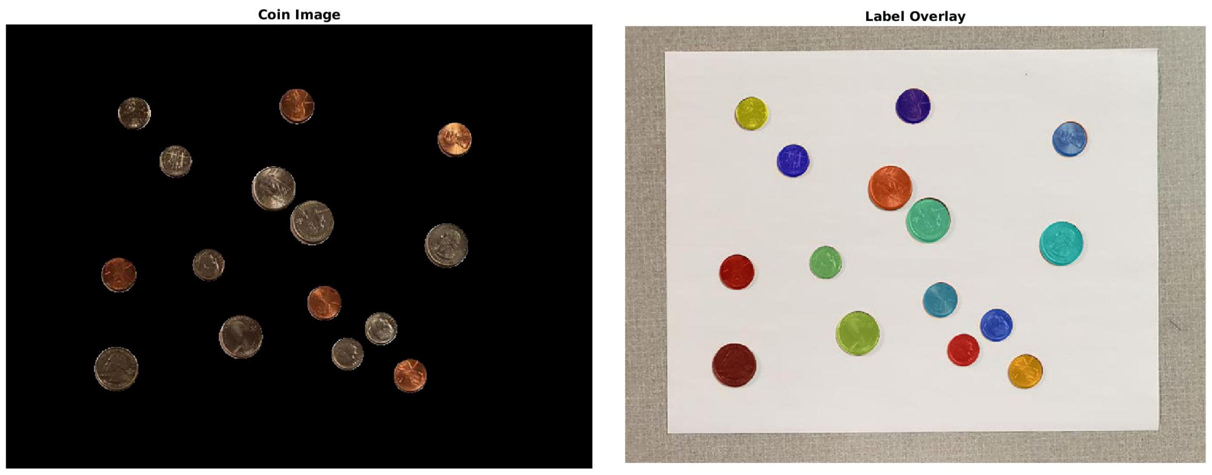
2-complex medium resolution complex background



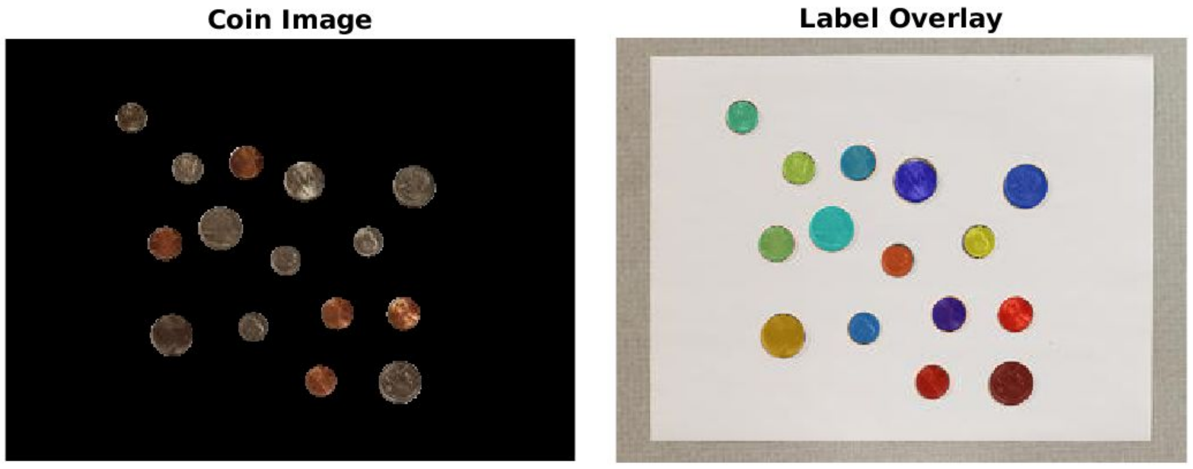
3-complex low resolution complex background



1-simple high-resolution simple background



2-simple medium resolution simple background

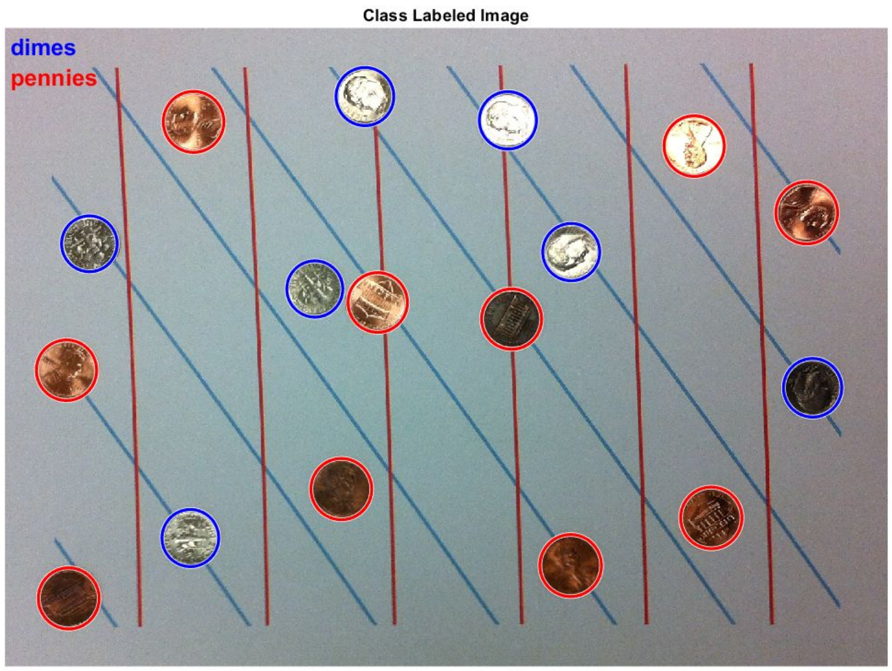


3-simple low-resolution simple background

3.2 Classification

3.2.1 Part 1 Object Classification

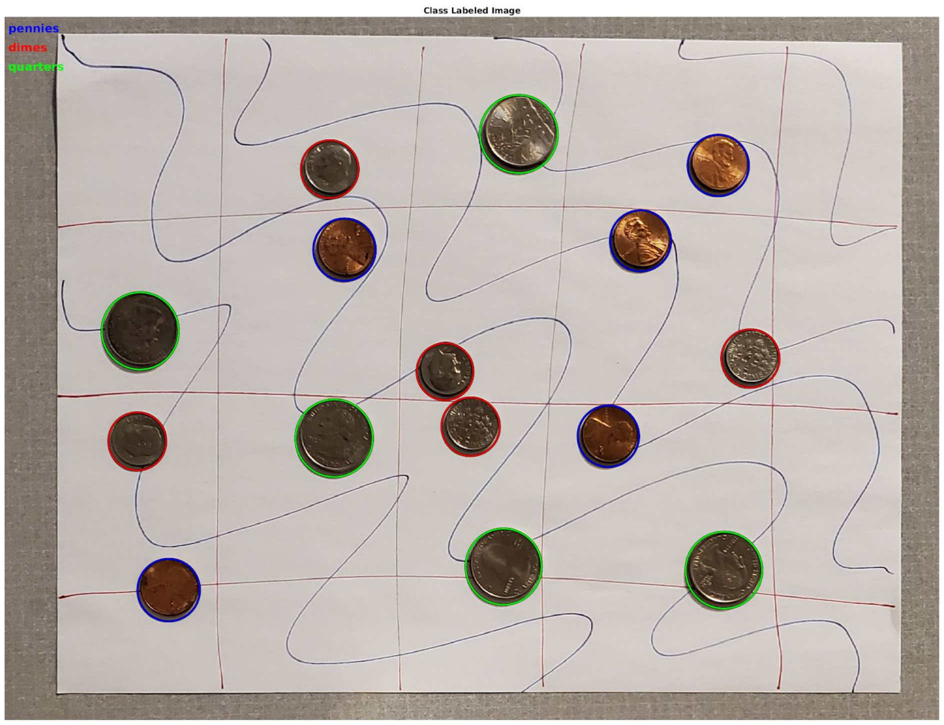
The KMeans classification method described in section 2.1.2 using the coin radii found in the Hough transform function results in the classification shown below.



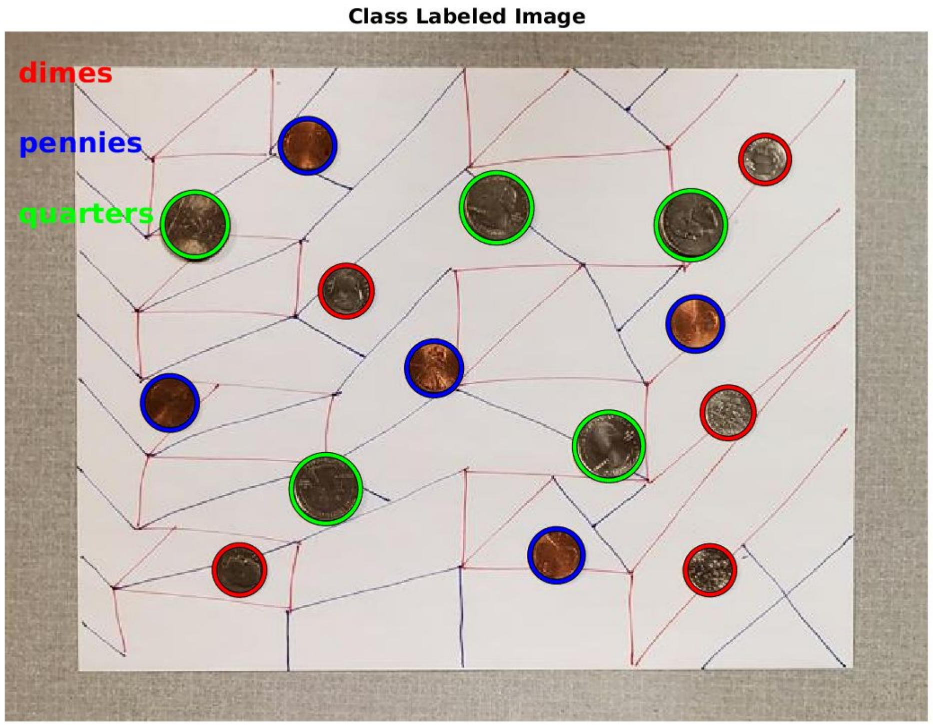
As one can see, the algorithm is able to detect and classify all coins in the test image with 100% accuracy.

3.2.2 Part 2 Object Classification

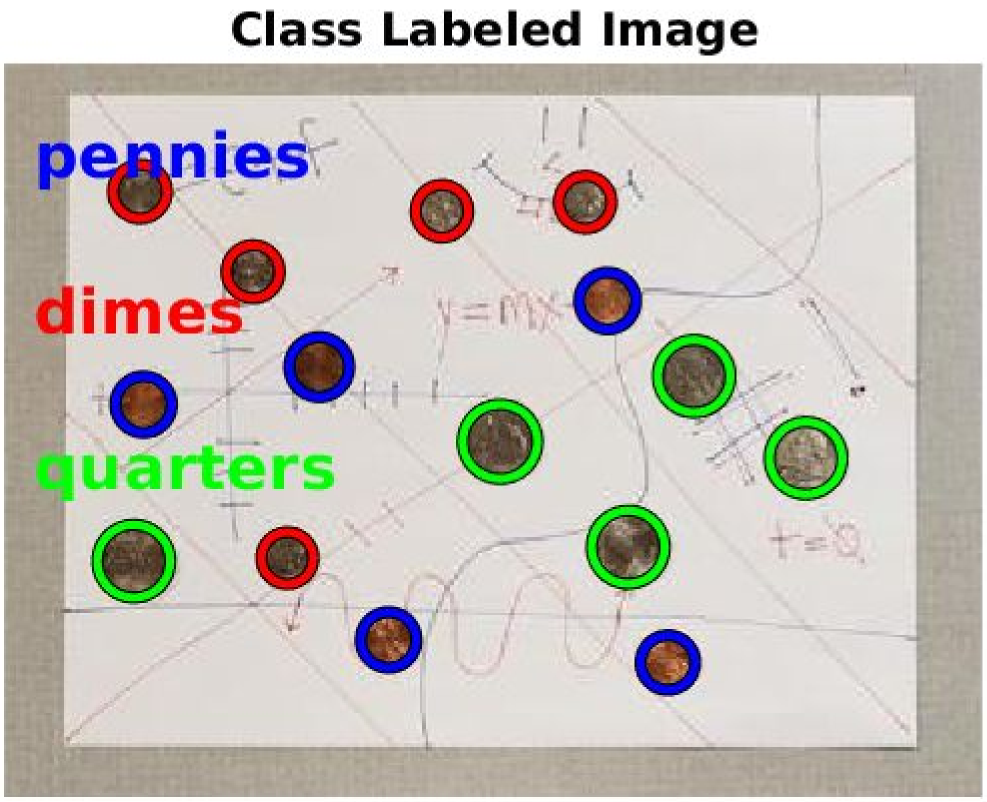
Our two-stage k-means method for classification was able to successfully all 419/419 detected coins across four different resolutions of all seven images. This algorithm was successful regardless of the inter-image lighting variance and inter-resolution size variance. The images below will show six different images, three complex backgrounds, and three simple, all at various resolutions with their correct classifications of each image overlayed.



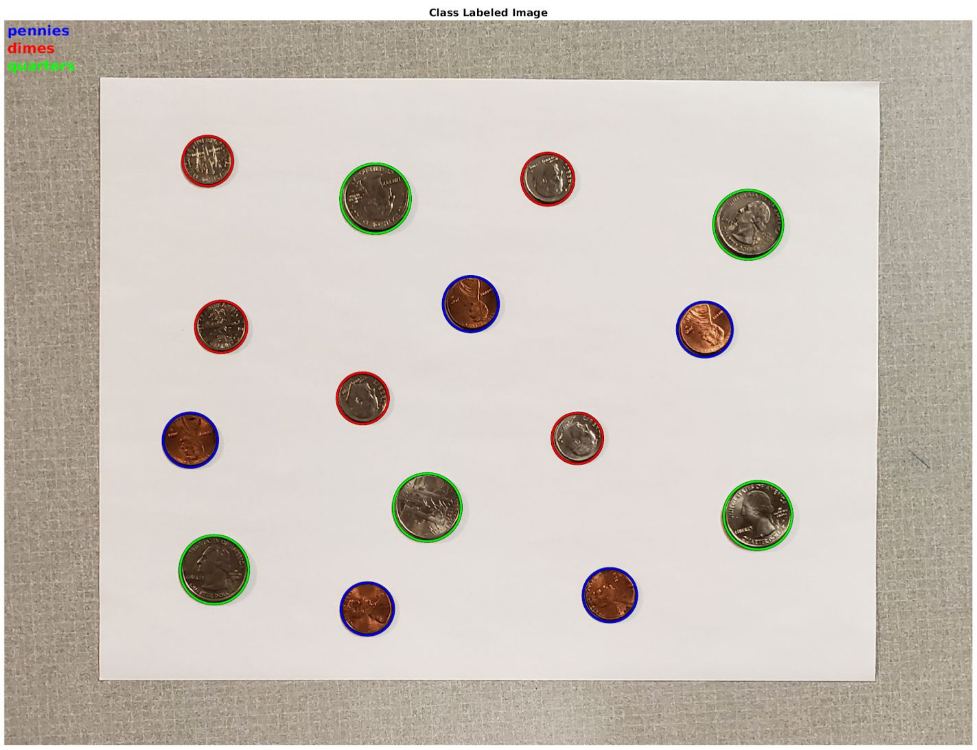
1-complex high resolution complex background



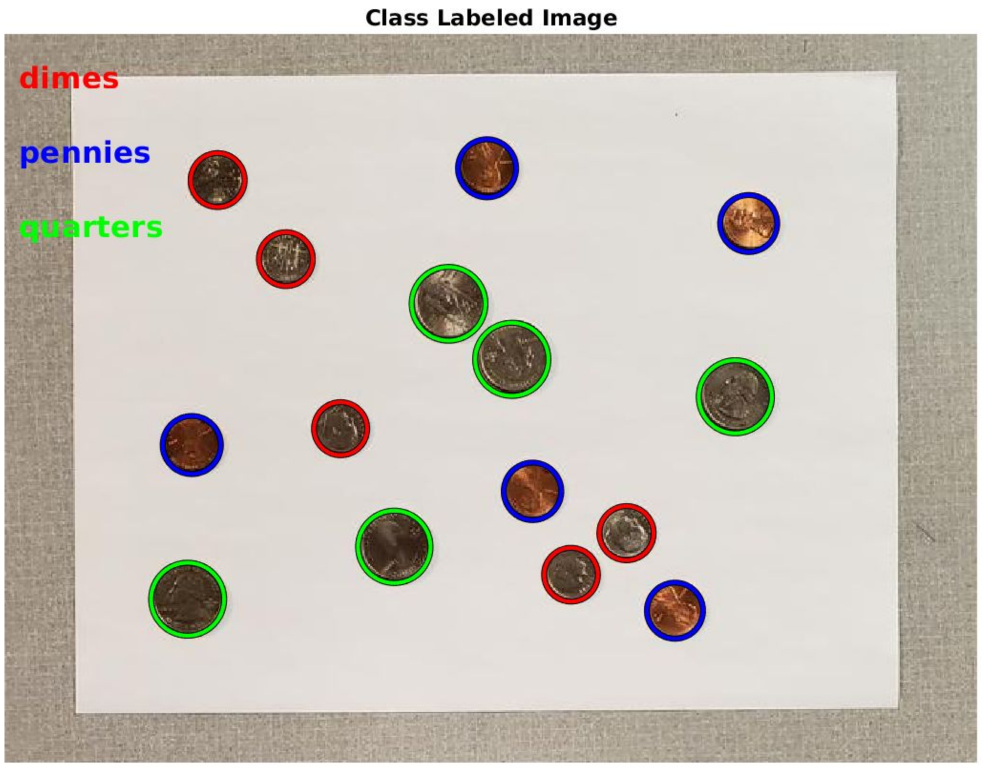
2-complex medium resolution complex background



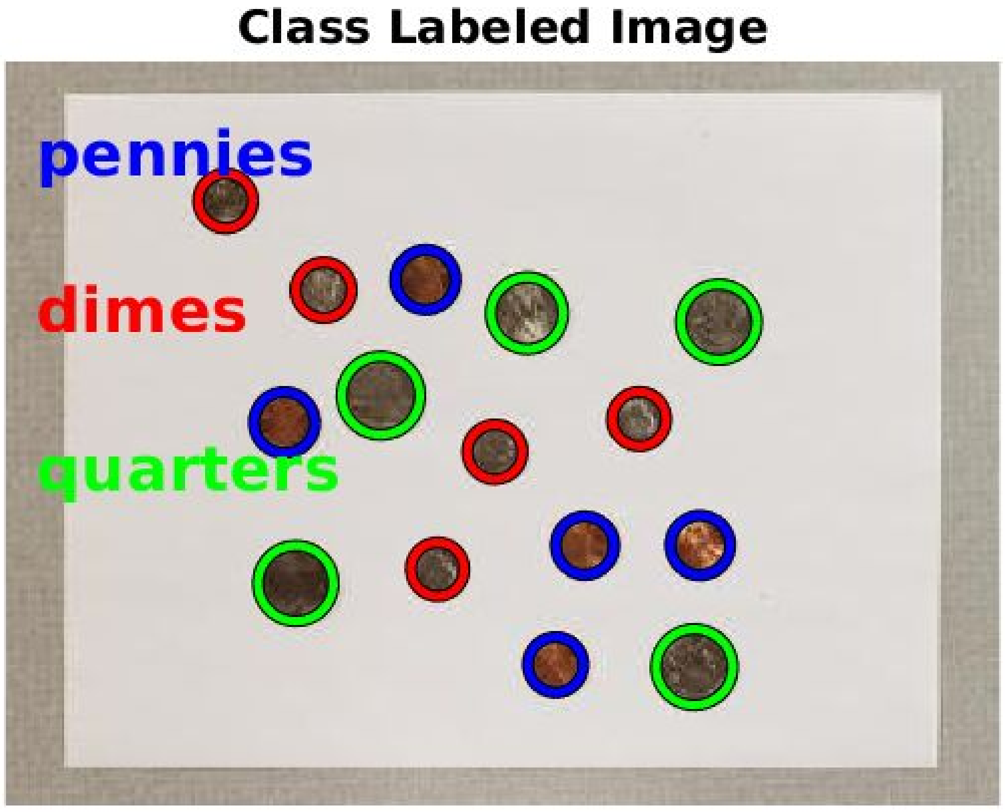
3-complex low resolution complex background



1-simple high resolution simple background



2-simple medium resolution simple background



3-simple low resolution simple background

4. Conclusion

Throughout this project, we learned about a variety of new techniques by simply encountering and attempting to find elegant solutions to bugs as they arose. We explored contrast enhancement, lighting correction, LAB color space, and many other components of the world of digital image processing that weren't even included in our final solution. We learned alot about thresholding and object detection techniques, and how the hough transform works in application. One of the biggest takeaways of this project is that designing generalizable image processing algorithms that will work effectively in all use cases is quite tricky, as it is typically possible to create an input image that will cause the process to fail.