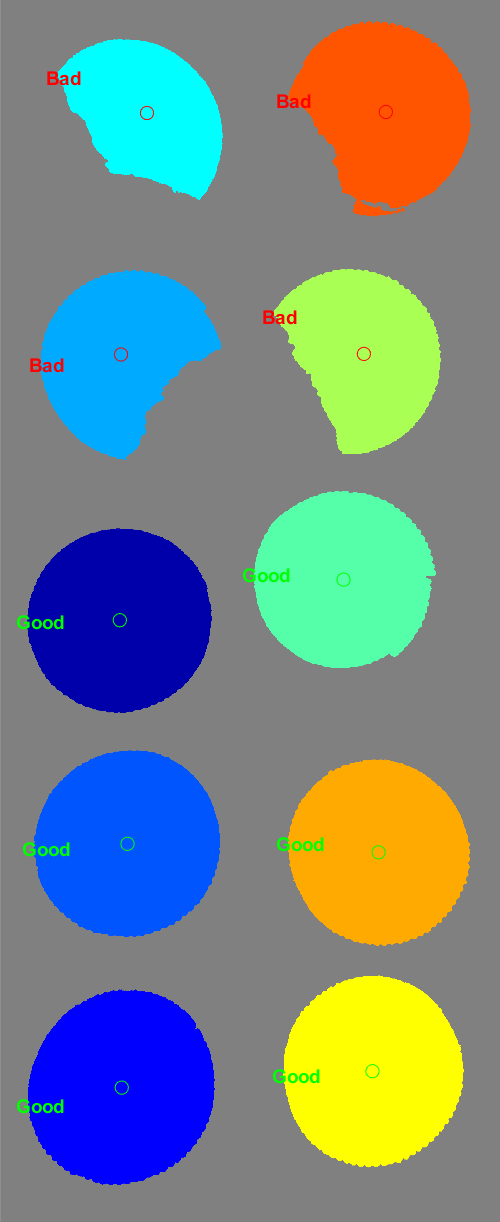
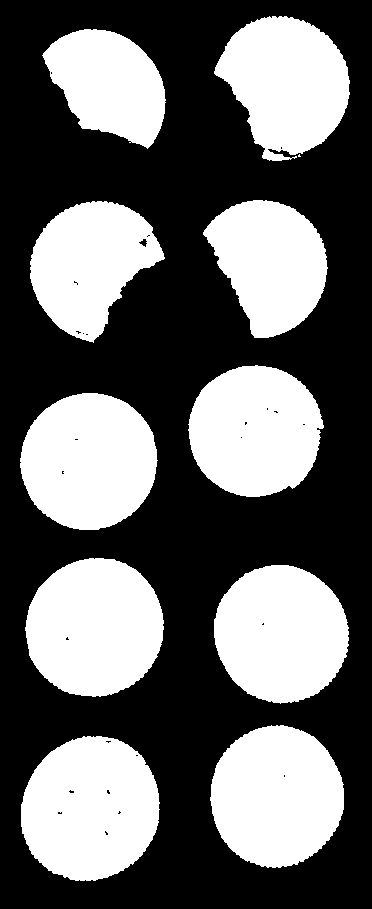
Katherine Banis  
CMPSC 497  
Lab#1B (part 2): Cookie Defect Detector  
09/12/2024

Objective: Design a cookie defect detector based on computer vision that would detect if a round cookie was defective (broken, etc.) based on roundness calculation. Test with actual cookies (or crackers) and determine what defects can be detected.

Materials: MATLAB, 5 good cookies, 5 bad cookies

Original Image and Figures:



Results:

Total number of cookies: 10  
Number of good cookies: 6  
Number of bad cookies: 4

Conclusion:

This assignment was done with five good cookies and five bad cookies. As you can see from the results, the program was able to correctly identify all ten cookies. However, it detected six good cookies and four bad cookies. The extra good cookie is due to the fact that the “bad cookie” is still pretty round. When the program fills in the cookie to then identify its roundness, the defective top half of the cookie is hidden, making it appear to be a good cookie. I experimented with the threshold. Originally when I made the threshold 0.9, the program classified all cookies as bad cookies. When I adjusted the threshold to 0.8, it classified only three cookies as good cookies. These three cookies were the ones I was directly above when I took the picture. The other two good cookies were not perfectly round because you could see a little from the sides and base of the cookie. However, at a threshold of 0.8, the cookie that ended up being classified as a good cookie despite being a bad cookie, was indeed classified as a bad cookie even though I was directly above it when taking the picture. Ultimately, I went with the threshold of 0.7 as I felt it better represented the cookies.

MATLAB Script

% access original rgb image

rgb = imread('cookie.jpg');

imshow(rgb);

% grayscale

gray = rgb2gray(rgb);

threshold = graythresh(gray);

figure, imshow(gray)

% binary image

bw = im2bw(gray, threshold);

figure, imshow(bw)

% remove all objects less than 200 px

bw1 = bwareaopen(bw, 200);

figure, imshow(bw1)

% fill gaps

se = strel('disk', 2);

bw2 = imclose(bw1, se);

figure, imshow(bw2);

% fill any holes

bw3 = imfill(bw2, 'holes');

figure, imshow(bw3)

% get px for boundaries of each object

[B,L] = bwboundaries(bw3, 'noholes');

% L = label matrix - try imtool(L) and imshow(L)

% display L and draw each boundary

imshow(label2rgb(L, @jet, [.5 .5 .5]))

hold on % allow graphics to be added to same plot

% find area in px and centroid (x,y) for each object in L

stats = regionprops(L, 'Area', 'Centroid');

% good and bad cookie values

threshold = 0.7;

badCookies = 0;

goodCookies = 0;

% loop over each object (each object has a boundary)

for k = 1 : length(B)

boundary = B{k};

delta\_sq = diff(boundary).^2;

perimeter = sum(sqrt(sum(delta\_sq, 2)));

area = stats(k).Area;

metric = (4 \* pi \* area) / perimeter^2;

centroid = stats(k).Centroid;

if metric > threshold

plot(centroid(1), centroid(2), 'go', 'MarkerSize', 10); % good cookies marked with green

goodCookies = goodCookies + 1;

text(boundary(1,2)-35, boundary(1,1)+13, 'Good', 'Color', 'green', 'FontSize', 14, 'FontWeight', 'bold');

else

plot(centroid(1), centroid(2), 'ro', 'MarkerSize', 10); % bad cookies marked with red

badCookies = badCookies + 1; % Corrected this line from goodCookies to badCookies

text(boundary(1,2)-35, boundary(1,1)+13, 'Bad', 'Color', 'red', 'FontSize', 14, 'FontWeight', 'bold');

end

End

% print results

fprintf('Total number of cookies: %d\n', length(B));

fprintf('Number of good cookies: %d\n', goodCookies);

fprintf('Number of bad cookies: %d\n', badCookies);