Andrew Kozempel

Professor Avanzato

CMPSC 497

8 September 2023

**Lab #1A: Round (Circular) Object Detection**

**Problem Statement:**

The objective of this lab is to design and implement a computer vision algorithm to detect total objects and round objects in a given image using MATLAB. The algorithm is aiming to analyze images to identify and detect distinct objects and then classify them based on their degree of roundness.

**MATLAB Script:**

| **%{ Andrew Kozempel CMPSC 497 Fall 2023 LAB #1A Finding Round Objects %}  images = {'test1.png', 'test2.png', 'test3.png'};  for i = 1:length(images)   % Access original RGB image  RGB = imread(images{i});  figure, imshow(RGB);    % Create grayscale image  gray = rgb2gray(RGB);  threshold = graythresh(gray);  figure, imshow(gray);    % Create bw (binary image)  bw = im2bw(gray, threshold);  figure, imshow(bw);    % Remove all object containing fewer than 30 pixels  bw1 = bwareaopen(bw, 30);  figure, imshow(bw1);    % Fill a gap in the pen's cap  se = strel('disk', 2); % Structuring element; "paintbrush"  bw2 = imclose(bw1, se); % We will cover later in course  figure, imshow(bw2);    % Fill any holes, so that regionprops can be used to estimate the area   % enclosed by each of the boundaries  bw3 = imfill(bw2, 'holes');  figure, imshow(bw3);    % Get pixels for boundaries of each object  [B,L] = bwboundaries(bw3, 'noholes');    % Display the label matrix and draw each boundary  imshow(label2rgb(L, @jet, [.5 .5 .5]));    % Allow graphics to be added to same plot  hold on;    % length(B) is number of objects  for k = 1:length(B)  % B is "cell" data type (set)  boundary = B{k};  plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2);  end    % Find area(in pixels) and centroid (x, y) for each object in label matrix  stats = regionprops(L, 'Area', 'Centroid');     % Arbitrary value (change as needed)  threshold = 0.9;     % Initialize round object count  round\_count = 0;    % Loop over the boundaries (each object has a boundary)  for k = 1:length(B)    % Obtain (X,Y) boundary coordinates corresponding to label 'k'  boundary = B{k};    % Compute a simple estimate of the object's perimeter    % Find 2-col. array of (x2-x1)^2 and (y2-y1)^2  delta\_sq = diff(boundary).^2;     % Sum (row) and take sqrt to find dist.  % Then sum all distances to find perimeter  perimeter = sum(sqrt(sum(delta\_sq,2)));     area1 = perimeter^2/(4\*pi);    % Obtain the area calculation corresponding to label 'k'  area2 = stats(k).Area; % So, we calculated the area 2 different ways    % Compute the roundness metric (compare 2 methods)  metric = area2/area1; % Circular objects have metric close to 1    % Display the results  metric\_string = sprintf('%2.2f', metric);    % Mark objects above the threshold with a small black circle in the center of the object  if metric > threshold  centroid = stats(k).Centroid;  plot(centroid(1), centroid(2), 'ko');    % Increment the round object count  round\_count = round\_count + 1;   end    text(boundary(1,2)-35, boundary(1,1)+13, metric\_string, 'Color', 'y', 'FontSize', 14, 'FontWeight', 'bold');  end    % Print test case number  fprintf('\nTest Case %d:\n', i)   % Print total number of objects (calculated @ line 36)  fprintf('Total Number of objects: %d\n', length(B));    %Print total number of round objects (calculated @ line 92)  fprintf('Total Number of round objects: %d\n', round\_count); end** |
| --- |

**Results:**

| **Test Case** | **Starting Image** | **Final Image** |
| --- | --- | --- |
| **1** |  |  |
| **2** |  |  |
| **3** |  |  |

| **MATLAB OUTPUT** | |
| --- | --- |
| **Copy and Paste** | **Screenshot** |
| >> Lab\_1A  Test Case 1:  Total Number of objects: 6  Total Number of round objects: 1  Test Case 2:  Total Number of objects: 6  Total Number of round objects: 2  Test Case 3:  Total Number of objects: 6  Total Number of round objects: 1 |  |

**Conclusion:**

Using the script we created in class as a base, I was able to make a few modifications to create a loop to test 3 different images. All images consisted of six objects. Test cases 1 and 3 had one round object each, while test case 2 had two round objects. Given the resulting final images and output, I was able to successfully identify the amount of total objects, as well as the amount of round objects within each image.

The main issue that I had trouble with was the lighting of the pictures. I had to retake pictures a couple of times because the algorithm had issues identifying distinct boundaries of certain objects. Some objects would be split into more than one object because of the poor lighting. To fix this, I had to ensure consistent lighting or I had to modify the position and orientation of certain objects.

Another issue was creating proper boundaries of the objects, but it did not affect the overall accuracy of the outcome. I also lowered the threshold to 0.9 and the lowest number (roundness metric) belonging to a round object was 0.91.

**Discussion (Can this algorithm be used to count number of M&Ms?) :**

The conditions would have to be ideal for the algorithm to work properly. The M&Ms must not be overlapping or that will cause issues creating boundaries of distinct objects. Secondly, there must not be any misshapen pieces or else it will not identify it as a round object (M&M). However, if there are only M&Ms present, it does not necessarily need to identify round objects, so it could still potentially work.