Aaron Feinberg
CMPSC-497
Lab 1a - Detecting Circular Objects
Materials: Matlab, Household Objects, Camera (Phone)

Discussion

In this Lab, I made of use of a few of MATLAB's image processing and computer vision features to determine the "roundness" of an object. We first take the raw image and convert it to gray scale and into a binary representation of the image. We then apply a few post processing techniques such as grain and noise removal (removing all groups of pixels below a certain threshold) to avoid having them counted as objects.

Roundness Algorithm

We find the size of each distinct object on the screen and then use the pixels at the boundary of each object to determine its shape and then we calculate its perimeter. We then super-impose a circle over the object. This circle has a radius such that the circle's perimeter would be equal to that of the object it is superimposed over. We then calculate the following ratio of these 2 figure's areas:

$$metric = \frac{a_{original}}{a_{circle}}$$

The close this metric is to 1, the more "round" a given object may be. To keep things interesting (and since i didn't really have too many distinct perfectly round objects), I set the minimum roundness threshold to 0.70.

Code

Since we'd be performing this operation 3 images, i created a function called `processImage()` which accepts 3 parameters.

- `im` - the image
- `roundness_threshold` - the minimum thershold for assuming something is "round" or not
- `graypoint` - a value for which the builtin `rgb2gray()` function should use as the graypoint (-1 for letting the function itself decide).

Code That Did Not Originate From the Lecture notes

Since the Image processing steps were the same for all images, I created a function called `processImage()` which returns the total number of objects and the total number of round objects (those which exceeded the threshold value given by the `roundness_threshold` parameter). The inputs to this function are

- `im` - the image file
- `roundness_threshold` - the value above which objects are round
- `graypoint` - this was something i had to tweak as different images had objects of different colors and slightly different lighting/angles. The gray point selected by the `rgb2gray()` function didn't do a very good job of choosing that value for this task.

In order to count the objects, i created a counter called `objects_exceed_threshold`. This counter would increment each time an object exceeded the roundness threshold and kept track of how many round objects there are in the photo.

```
% importing image of objects
im_1 = imread('/home/aaron/Pictures/circular_obj.jpg');
im_2 = imread('/home/aaron/Pictures/circular_obj_2.jpg');
im_3 = imread('/home/aaron/Pictures/circular_obj_3.jpg');

% processing 3 images - adjusting graypoint and roudness threshold
as needed
processImage(im_1, .70, .165)
processImage(im_2, .70, .35)
processImage(im_3, .85, -1)

% input: image, minimum roundness value, gray point (-1 lets algo
```

```
choose)
% returns total objects and number of round objects
function [total_objects, round_objects] = processImage(im,
roundness_threshold, graypoint)
    % display original image
   imtool(im);


    % converting to grayscale
    im_gray = rgb2gray(im);


    % binarizing the image (passing -1 lets us have the default
algorithm choose the gray point)
    if graypoint == -1
        graypoint = graythresh(im_gray);
    end

    im_bin_threshold = imbinarize(im_gray, graypoint); % removes
objects


    % removing noise/grains
    im_bin_denoise = bwareaopen(im_bin_threshold, 1000); % lots of
tweaking

    % filling in any holes in the image
    se = strel('disk', 15); % lots of tweaking
    im_bin_closed = imclose(im_bin_denoise, se);
    im_bin_filled = imfill(im_bin_closed, 'holes');


    % Boundary and Label Matrices
    [B, L] = bwboundaries(im_bin_filled, 'noholes')
    imshow(label2rgb(L, @jet, [.5 .5 .5])) % Displaying the distinct
objects



     % getting true and circular areas
     stats = regionprops(L, 'Area', 'Centroid');
```

```
     % counting round objects (those that are above threshold value)
     objects_exceed_threshold = 0;

      % appending more data to the graph
    hold on


    % drawing boundaries
    for k = 1 : length(B)
            boundary = B{k}
            plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)

        delta_sq = diff(boundary).^2;
        perimeter = sum(sqrt(sum(delta_sq,2)));

        area1 = perimeter^2/(4*pi);
        area2 = stats(k).Area;

        area_true = stats(k).Area;
        metric = area2/area1

        metric_string = sprintf("%2.2f",metric);

        % plot a circle about the centroid
        if metric > roundness_threshold
                centroid = stats(k).Centroid;
            objects_exceed_threshold = objects_exceed_threshold + 1;
                plot(centroid(1), centroid(2), 'ko');
        end

        % plot the roundness or ciruclarity metric near the object
        text(boundary(1,2)-100, boundary(1,1)+120, metric_string,
'color', 'y','FontSize', 14, 'FontWeight', 'bold');
    end

     hold off

    sprintf("Total Objects: %.f", length(B))
    sprintf("Objects Exceeding Roundness Threshold: %.f",
objects_exceed_threshold)
```

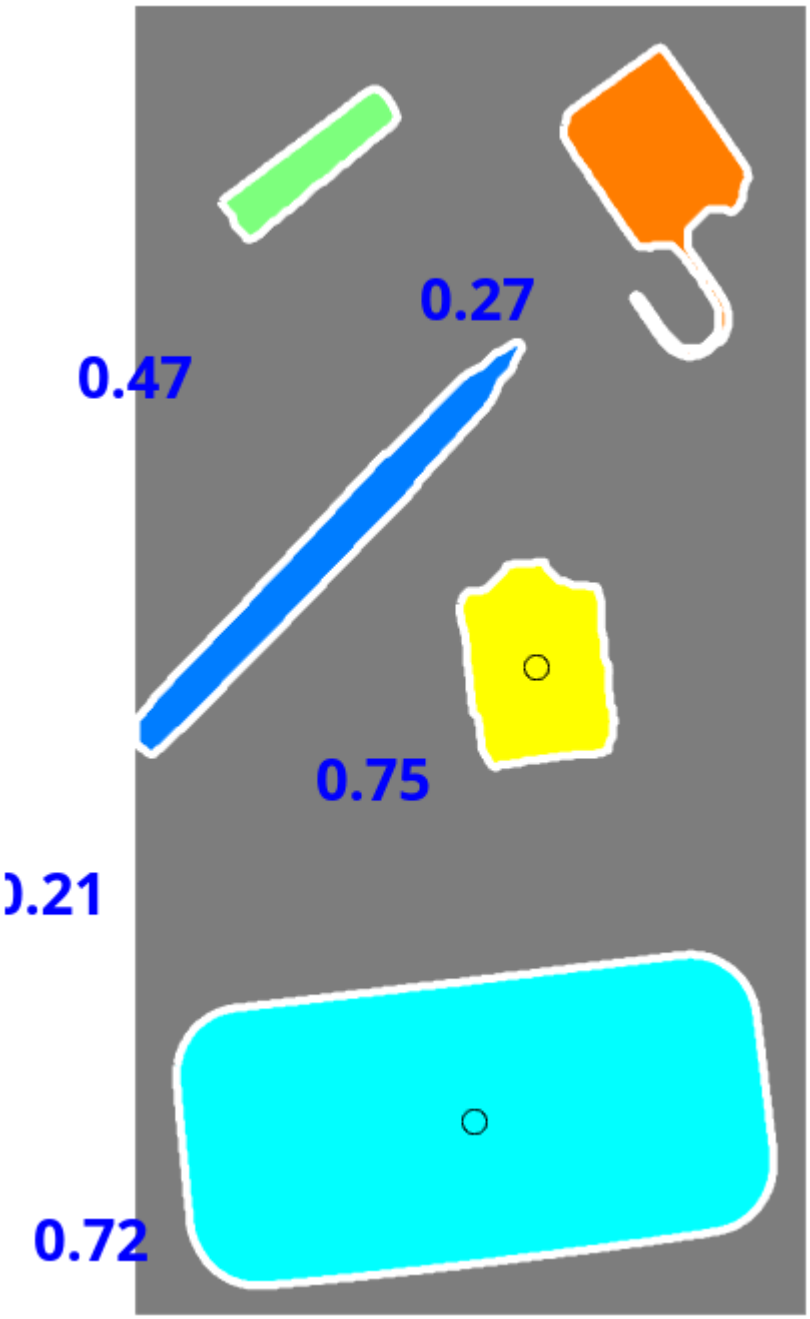
```
    total_objects = length(B);
    round_objects = objects_exceed_threshold;

end
```

Results

Image 1


Image Tool 1 - im
dry erase
remooble.
Pixel info: (X, Y) Pixel Value



ans = "Total Objects: 5"

ans = "Objects Exceeding Roundness Threshold: 2"

Image 2

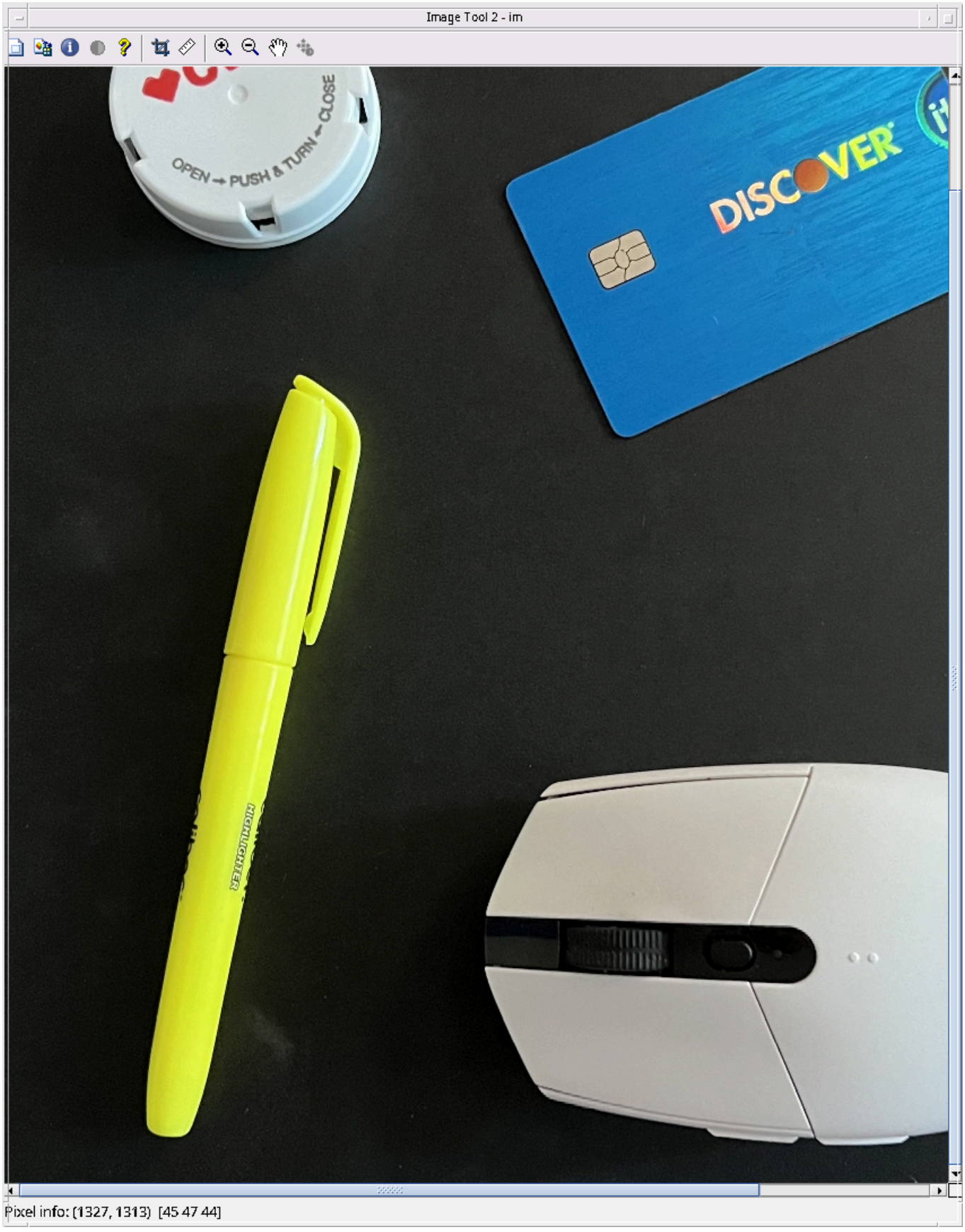
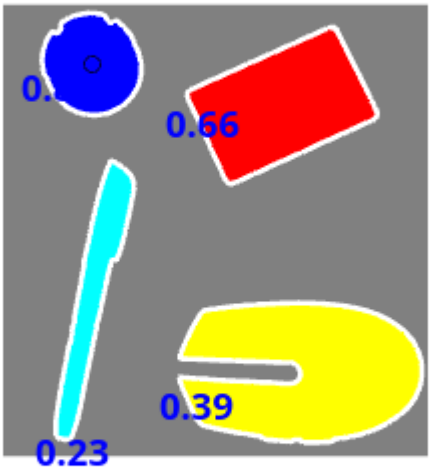


Image Tool 2 - im
OPEN → PUSH & TURN ← CLOSE
DISCOVER
HIGHLIGHTER
Pixel info: (1327, 1313) [45 47 44]

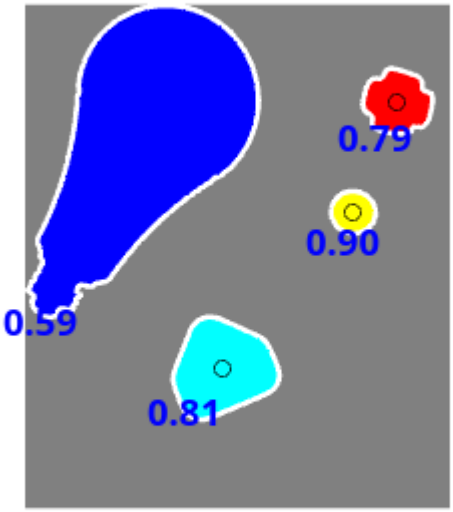


ans = "Total Objects: 4"

ans = "Objects Exceeding Roundness Threshold: 1"

Image 3


Image Tool 3 - im
Pixel info: (X, Y) Pixel Value



ans = "Total Objects: 4"

ans = "Objects Exceeding Roundness Threshold: 3"

Problems

One of the main problems I encountered was the issue of the default gray point. I found that in order to avoid doing a lot of post processing in MATLAB, I needed to find objects with a lot of contrast between the background. The default gray point that was chosen was often sensible as an average brightness value but that was actually detrimental in most cases for this lab. It was also clear that any sort of glare or uneven lighting showed up as an object when I binarized the image. Also, objects like the little cologne bottle in the first image had a black logo in the center which when binarized created a hole in the center of the bottle. This required me to play with the imclose() and imfill() functions quite a bit (as well as for other objects).

In summary, selecting a high contrasting gray point, allowed you to set a black point such that you wouldn't lose any objects nor pickup a lot of noise and

grain.

Question: Can this algorithm be used to count number of M&Ms?

Yes, Since M&Ms are uniformly sized then it seems reasonable. The only difficulty would be to have enough contrast between M&Ms of a darker color and those of a lighter color. It would also be important to fine tune parameters like the bwfill() values are large enough to ensure that the M inside the M&M gets filled but small enough that the entire M&M isn't removed. This would be made easier with consistent lighting and a high contrast, uniform background.