HW - 3 (Hough Circles)

This lab employed the Hough transform to detect circles within in an image.

Hough Transform

The Hough transform solves the problem of inability to detect vertical lines by parameterizing the equations into an angle and a perpendicular line (called the $\rho\Theta$ space)

We define a line normal to ρ with the following equation:

$$xcos\Theta + ysin\Theta = \rho$$

in the $\rho\Theta$ space, the equivalent set of lines in a Cartesian plane appear as a pair of sinusoidal lines. The intersection of these two sinusoids indicates that points are co linear at that point in (x,y) space. We generate a point for each point in that space and then we count the intersections.

Code

Custom Function

I created a custom function which took in the following parameters

- `im_loc` - the image location
- `rmin, rmax` - the minimum and maximum radii for the circles in the image
- `object_polarity` - whether the object will be dark or bright with respect to its background
- `sensitivity` - how "sensitive" the algorithm is to detecting circles

```
function [] = findCircles(im_loc, rmin, rmax, object_polarity,
sensitivity)
    im = imread(im_loc);
    imshow(im);

        % image data
    disp(im_loc)
    disp(size(im)) % row, column, color values

    % get the center and radius for circles in the image
    [center, radius] = imfindcircles(im,
[rmin,rmax],'ObjectPolarity',object_polarity,'Sensitivity',
sensitivity)

    % display a in the center and red line about the
circumference
    viscircles(center, radius)
    hold on; plot(center(:,1), center(:,2),'yx','LineWidth',2);
    hold off;
end
```

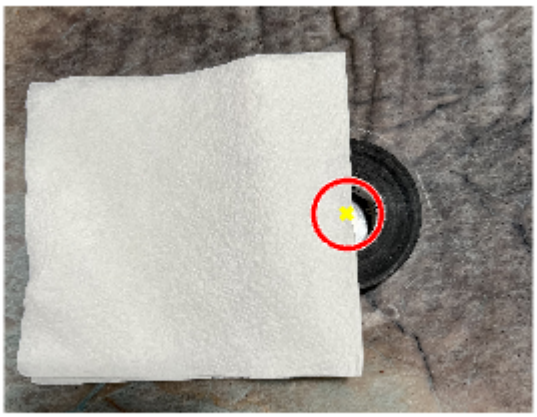
Output

week_4/hw3/tape.png
2304 2999 3

center = 1×2
103 ×
1.9476 1.1807

radius = 192.1332

ans =



week_4/hw3/coffee.png
3005 2692 3

center = 1×2
103 ×
1.3767 2.0405

radius = 300.0317

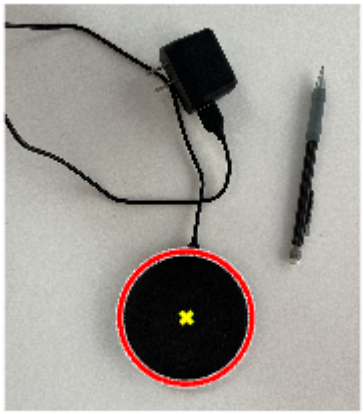
ans =

week_4/hw3/charger.png
3400 2972 3

center = 1×2
103 ×
1.5150 2.6233

radius = 553.5450

ans =



Difficulties

this algorithm was not as easy to work with as the previous line detection ones. It took quite a bit of tweaking to get the sensitivity value for the circle detection function correct. I had many instances of either no circles being found, too many circles being found, or circles being found in the wrong places. Given that this algorithm should be effective over the original image, I needed to demonstrate its efficacy but I feel that binarizing and denoising the image a bit would have helped greatly.