CMPT 412 Assignment 2: Tennis Ball Locator

Alec Pershick

301190478

Strategy

In order to locate tennis balls from an input image, my initial idea was to convert the image into black and white pixels so I could more easily differentiate the gradients in the image. This seemed to work with various edge detection algorithms but the Canny Edge Detection algorithm seemed to get the best results from thorough testing.

The next step was to generate a sample disk and mask to compare across the thresholded image. I made up a mask initialized with black pixels (-1s) and then filled the inside of the disk structure with whites (1s). This would allow me to compare regions of the image to the shape of the mask (circular) by convolving the mask with the image.

Using the resulted convolved image, I could then find “candidate” disks throughout the image that passed a condition (good enough to be a tennis ball). From there, shrinking the connected points to a single central point is how I could find the central coordinates of each tennis ball.

Problems / Solutions

This is where the tinkering began, as there are multiple variables or parameters to functions that needed a multitude of testing in order to find consistent results throughout multiple images.

The first parameter I needed to set was for the Canny Edge Detector function. This function takes two parameters, a threshold and a sigma value. Changing these actually had major impacts on my results, as setting a lower sigma value would detect many more smaller less prominent edges. This is not what I wanted, as I really only care about the circular edges of the tennis balls in the image. I ended up setting the threshold to 0.2 and sigma to 50.

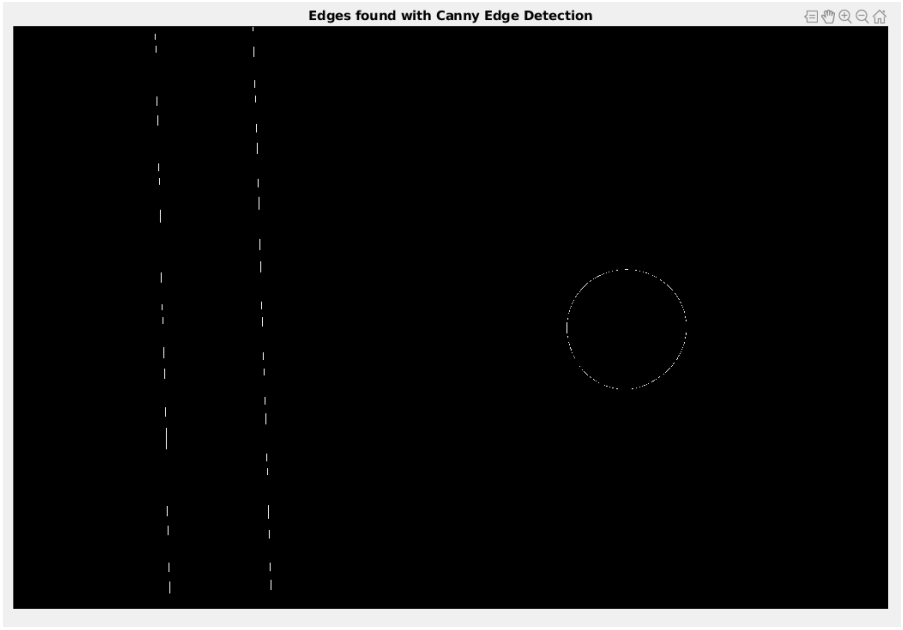
Next I needed to figure out how large to make the mask/disk structure that I would compare the regions of the image to. This was a tough choice, as making the disk's radius bigger meant that pictures with smaller tennis balls in it might get overlooked. I ended up settling my value to 300.5, which still does not work for many of the images with small tennis balls, but it was the best I could get. One thing I was thinking about implementing was a function to either switch between two different values for this based on the image, or have a for-loop testing each value until it finds the best option (usually the one with between 1-3 circles found). Unfortunately I never got around to actually figuring out a proper implementation.

The last variable in my code that needed tweaking based on the image was my "y" value. This value holds the points that pass the condition passed into the "find" function. I ended up setting my parameter to (c > 1500) which is a very high value. I needed this value to be high enough that all of the edges except for the perfectly circular ones were thrown out.

Conclusion

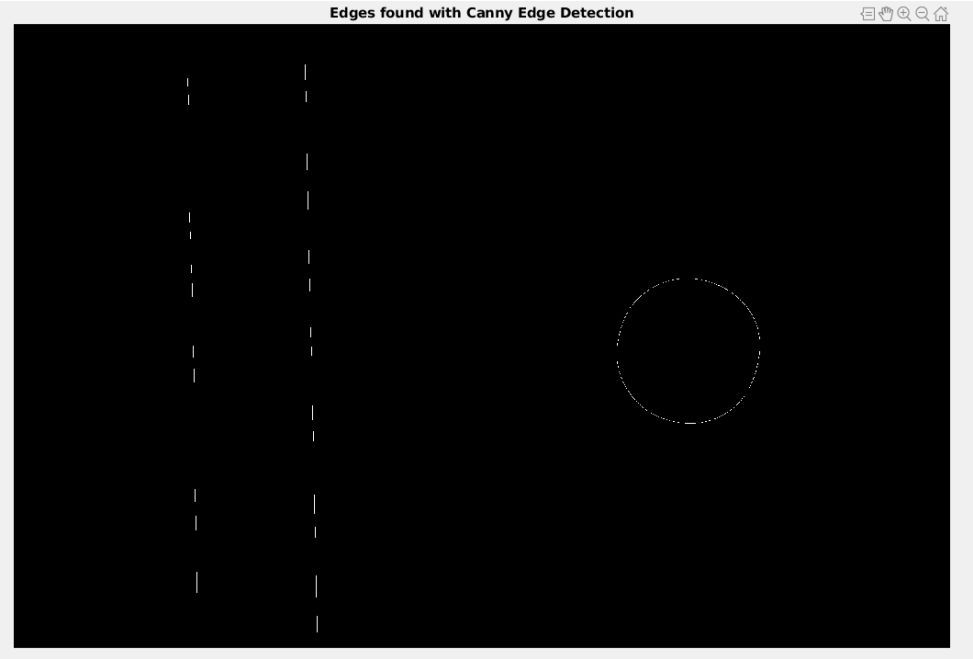
With the tweaks above, I managed to get consistent results from 4-5 of the sample images keeping everything the same. With tweaking parameters, I can get almost any of the images to give me the coordinates of the tennis balls, but that is impractical. The tests were for the most part successful in locating the tennis balls. As shown in **Test #1**, the simple case of a single tennis ball works flawlessly. The white dot indicates the center of the tennis ball in the resulting image, and is quite accurate. For a bit more complicated task, **Test #2** shows that even with lettering on the tennis ball, the filtering does a good job of distinguishing that it is still one solid object and not multiple edges. In **Test #3** we can see that even having two tennis balls practically touching each other, my algorithm separates them, along with their shadows. My solution is not perfect, and I could probably improve upon it with additional time, but it does a solid job when it comes to locating tennis balls within input images.

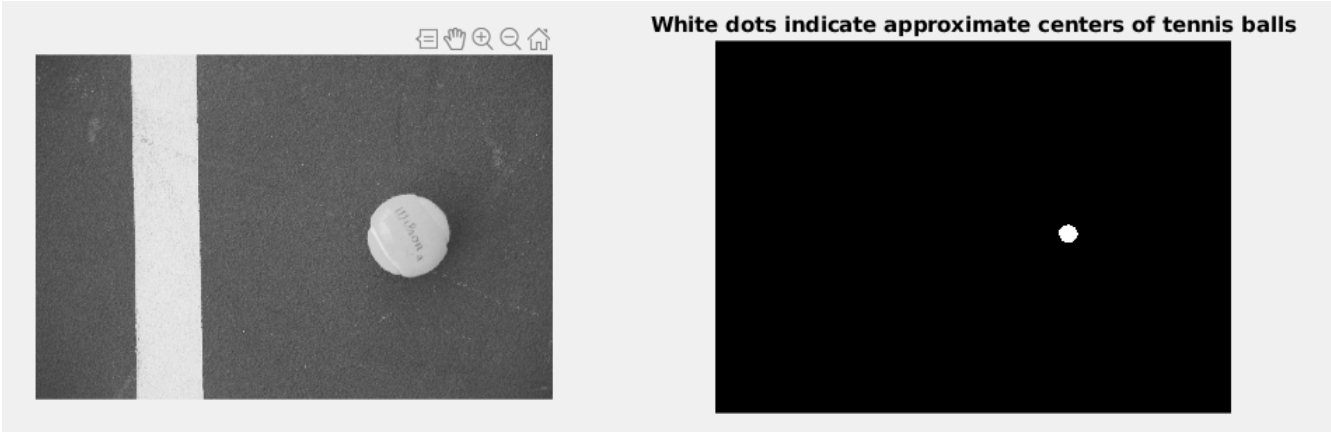
Test #1




```
ans =

    'Number of circles found is: '

ans =

     1

ans =

    'Circle locations are: '

ans =

        1362        2449
```

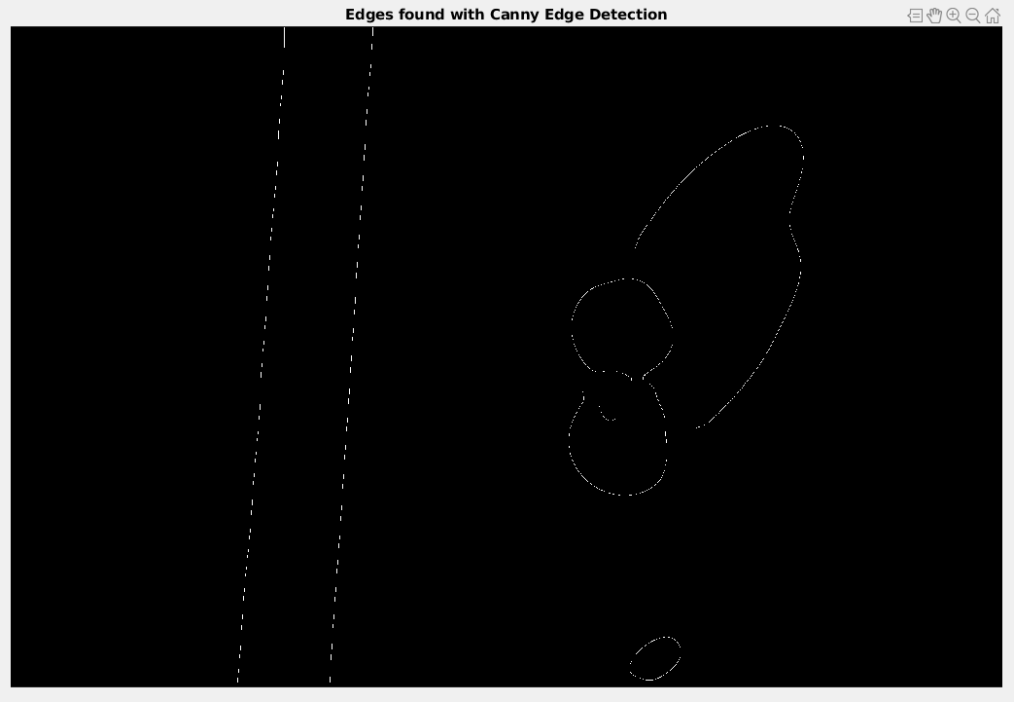
Test #2

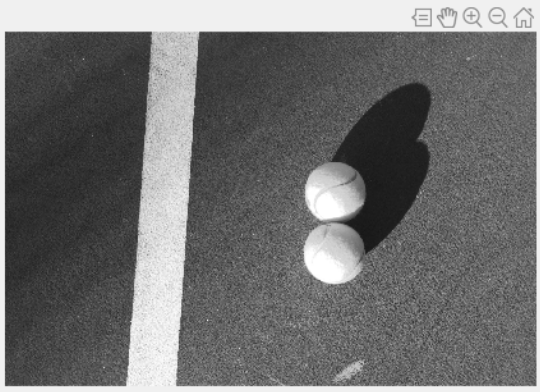


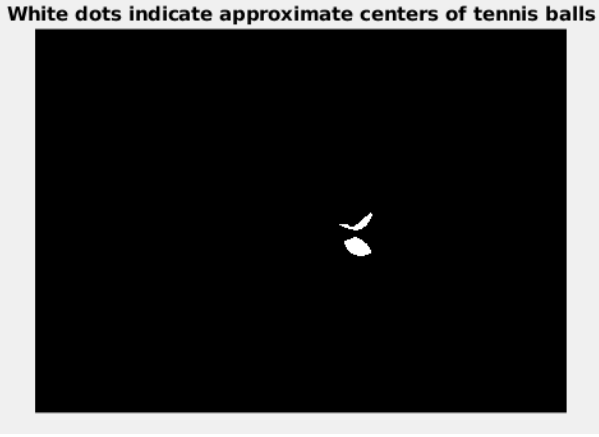



```
ans =

    'Number of circles found is: '

ans =

     1

ans =

    'Circle locations are: '

ans =

        1369        2508
```

Test #3








```
ans =

    'Number of circles found is: '


ans =

     2


ans =

    'Circle locations are: '


ans =

        1366        2216
        1498        2223
```

The Codes

```
% Assignment 2: Tennis Ball Locator
% CMPT 412
% Alec Pershick
% 301190478

close all; % Clears previous figures each run
image=imread('images/OneBallLetteringVerticalLarge.jpg');

% Converting the image to black and white in order to more easily detect edges
bw=(double(image(:,:,1))+double(image(:,:,2))+double(image(:,:,3)))/(3*255);

% Canny edge detection with threshold of 0.1 and sigma of 60
ec4 = edge(bw,'canny', .2, 50);
multi=cat(4,bw,ec4);
figure; imshow(ec4);
title('Edges found with Canny Edge Detection');

cs=300.5;           % Generate disks of radius 300.5 (must always be something + 1/2)
border=2;           % Border to go around the disk
ms=2*(cs+border);   % Mask size for the disk
msh=floor(ms/2)+1;  % Midpoint of the disk
mask=-ones(ms,ms);  % Initialize mask with a bunch of (-1)s

% Fill inside the disk with 1s (white)
for i=1:ms
    for j=1:ms
        if (i-msh)^2+(j-msh)^2<=cs^2
            mask(i,j)=1;
        end
    end
end

c=conv2(mask,ec4);  % Convolve mask with thresholded image to
                    % measure how disk-like each region is

y=find(c>1500);     % Locations of candidate circular disks are
                    % those with high values after the convolution
```

```
res=zeros(size(c)); % Important here that 'c' is in fact bigger
                    % than 'bw' because of convolution boundaries.

res(y)=1;           % Mark all candidate locations in res

% Display the original image vs the approximate tennis ball locations
figure;
subplot(1,2,1);
imshow(bw);
subplot(1,2,2);
imshow(res);
title('White dots indicate approximate centers of tennis balls');

result=bwmorph(res,'shrink',Inf);  %shrink connected objects to single points

[xcenter, ycenter]=find(result>0);

'Number of circles found is: '

length(xcenter)

'Circle locations are: '

[xcenter ycenter]
```