In this lab, the goal was to implement a color tracking algorithm to distinguish between red and non red objects on the screen. The alogrithm will place a centroid in each red object the screen as well as a bounding box around the object

Algorithm

the algorithm works in the following way: first we separate the image into color intensity matrices for each pixel value color. This lets us compare color intensities for each pixel color (Red, Green, Blue) for each pixel in the original image. We then apply the following heuristic:
`red = (r>2*g) & (r>2*b) & (r>30);`

This ensures that the pixel currently being observed is at least twice as bright as the other values and at least 30 units above 0 in brightness. These conditions filter out non-red light, or white light which, although it has a high red intensity, also has a high intensity of the other pixel values.

Custom Code

in this function i implemented the code to do two things:

1. Plot a '+' in the centroid of each object in some given image
2. draw a bounding box around an object

```
function [] = processIm(im_loc)

    im = imread(im_loc)

    r=im(:,:,1);
    g=im(:,:,2);
    b=im(:,:,3);

    % hueristic function for determining red only
    red = (r>2*g) & (r>2*b) & (r>30);
    % group all red objects within 5 pixels of each other as one
object (this one should be tweaked)
    se = strel('disk', 35);
    red = imclose(red,se);

    % remove all objects smaller than 35 pixels in area (adjust this
threshold as needed)
    red = bwareaopen(red, 20);

        % Centroid

    % separate red and NOT red
    s = regionprops(bwlabel(red), 'centroid');
    S = vertcat(s.Centroid);

    % plotting...
    figure
    imshow(im)
    hold on
    plot(S(:,1), S(:,2), '+') % identify red objects with +
    zoom on


        % Bounding Box
    % get all red objects (appear as white blobs from intensity
graph)
    stats = regionprops(red, 'BoundingBox', 'Centroid') % passing in
red intensity binarized image

    hold on
    % This is a loop to bound the red objects in a rectangular box.
    for object = 1:length(stats)
        bb = stats(object).BoundingBox;
        bc = stats(object).Centroid;
        rectangle('Position', bb, 'EdgeColor', 'r', 'LineWidth', 2)
% draw bounding box
        plot(bc(1), bc(2), '-m+') % mark centroid (x, y) with '+'
        a=text(bc(1)+15,bc(2), strcat('X: ', num2str(round(bc(1))),
' Y: ', num2str(round(bc(2)))));
        set(a, 'FontName', 'Arial', 'FontWeight', 'bold',
'FontSize', 12, 'Color', 'yellow');
```

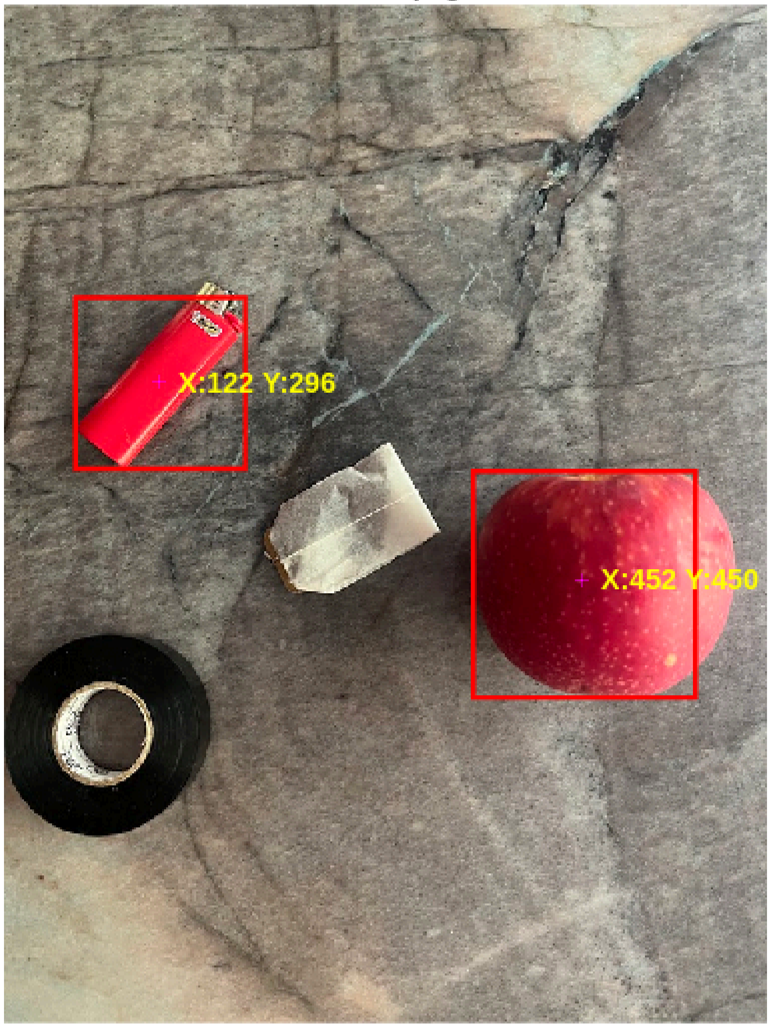
```
        end
    end
```

Results

stats = 2×1 struct

Fields	Centroid	BoundingBox
1	[121.6175,295.6192]	[56.5000,229.5000,133,134]
2	[452.1122,450.3538]	[367.5000,365.5000,174,177]

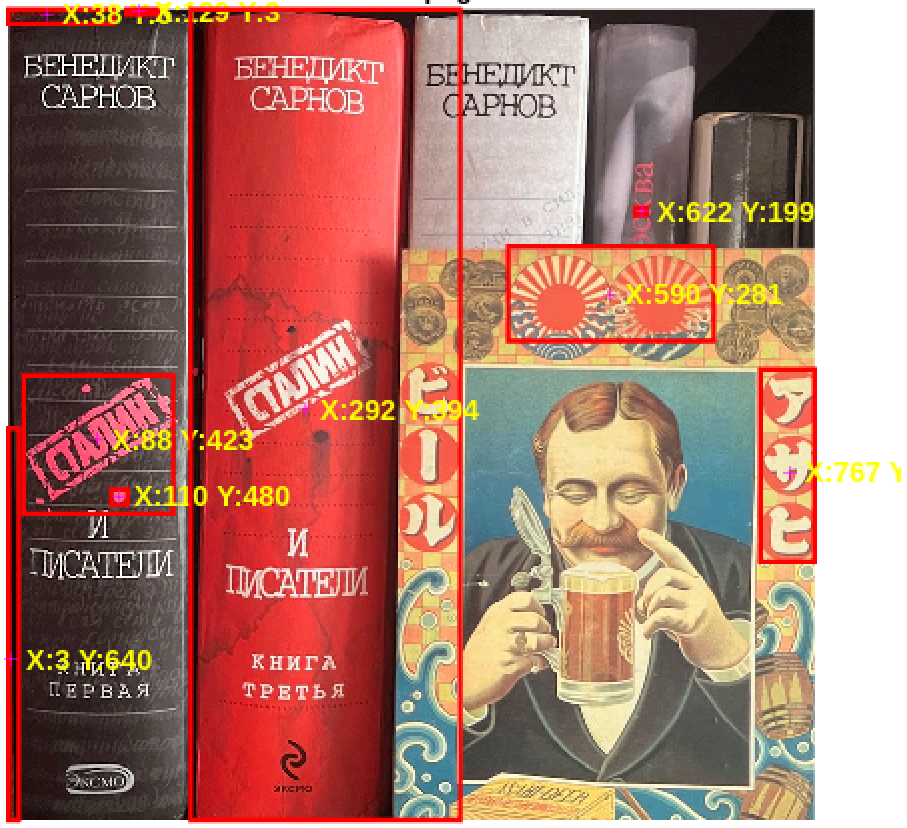
two red.png



stats = 9×1 struct

Fields	Centroid	BoundingBox
1	[38.0948,5.2599]	[0.5000,0.5000,103,14]
2	[3.0913,640.0782]	[0.5000,412.5000,10,385]
3	[87.7016,423.3643]	[15.5000,360.5000,147,136]
4	[109.6354,479.5635]	[101.5000,472.5000,16,15]
5	[128.5962,2.6442]	[116.5000,0.5000,31,5]
6	[291.8070,393.8673]	[179.5000,0.5000,264,797]
7	[590.3688,280.6616]	[490.5000,233.5000,202,93]
8	[621.8158,199.2632]	[615.5000,195.5000,11,7]
9	[766.9234,455.8050]	[738.5000,354.5000,53,189]

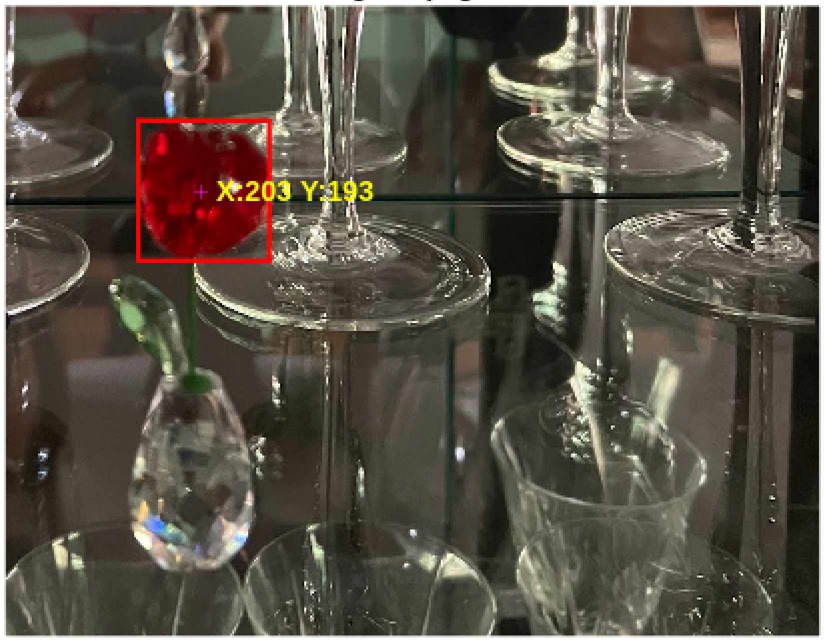
book.png



stats = struct with fields:

Centroid: [203.3797 193.3496]

BoundingBox: [138.5000 119.5000 136 146] glass.png



Conclusion

This lab was a good first introduction into color recognition but the task gets a bit more difficult when we have images with both large and small red objects. By setting a static structuring element size, we may be overlooking or under filtering red objects. Also, as is clear from the `book.png` photo, its interesting to note that the bounding box combined adjacent red portions of the image (in the Japanese text) with the red book. I played with the `strel` size and got varying results to bound the red images .