ECSE 4530 Image Processing

HW 6

Mar 26, 2018
Daniel Southwick
661542908

```
%% ===================================6.1====================================
```

Main Function

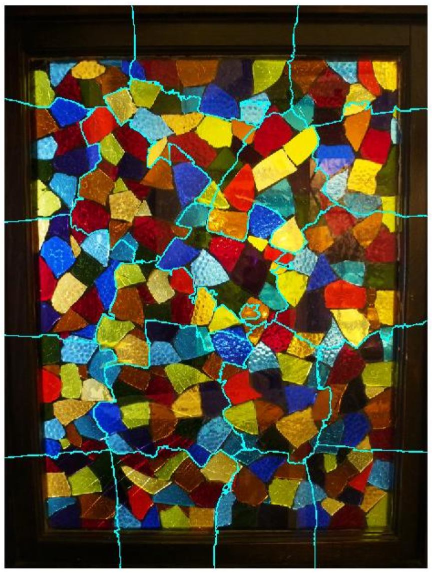
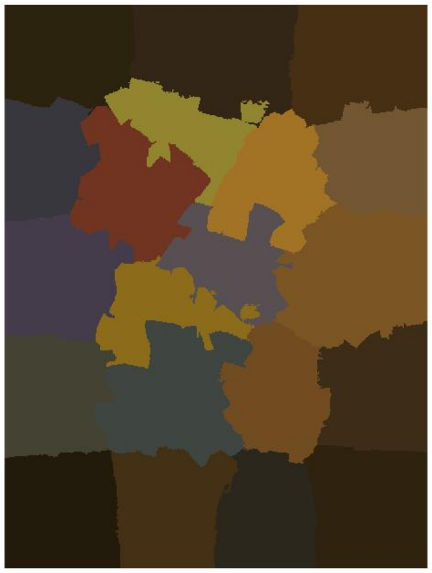
```
%% 1
glass = imread('glass.png');
mysuperpix(glass,50);
mysuperpix(glass,200);
mysuperpix(glass,500);
```

Mysuperpix.m

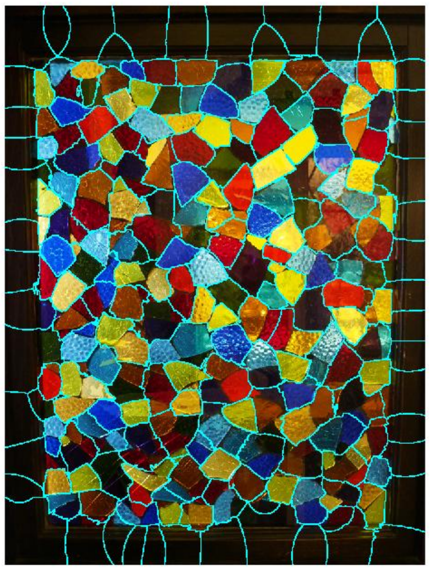
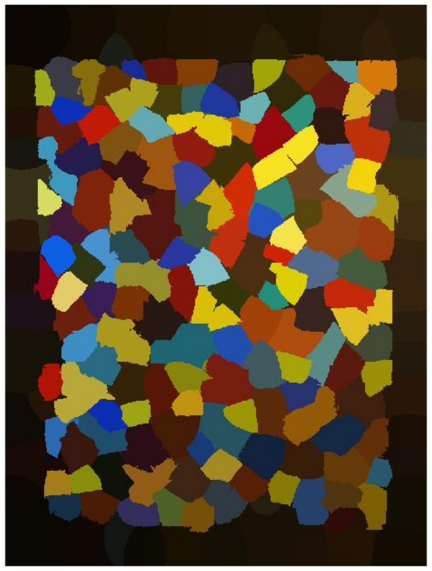
```
function mysuperpix( im, k )
[L, N] = superpixels(im, k);
figure
BW =boundarymask(L);
imshow(imoverlay(im,BW,'cyan'),'InitialMagnification',67);

outputImage = zeros(size(im),'like',im);
idx = label2idx(L);
numRows = size(im,1);
numCols = size(im,2);
for labelVal = 1:N
    redIdx = idx{labelVal};
    greenIdx = idx{labelVal}+numRows*numCols;
    blueIdx = idx{labelVal}+2*numRows*numCols;
    outputImage(redIdx) = mean(im(redIdx));
    outputImage(greenIdx) = mean(im(greenIdx));
    outputImage(blueIdx) = mean(im(blueIdx));
end
figure
imshow(outputImage,'InitialMagnification',67)
end
```

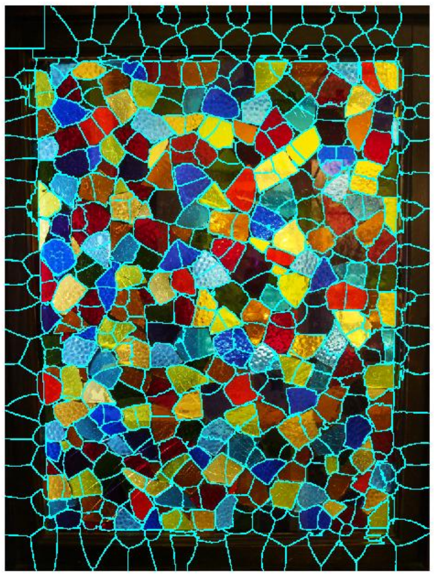
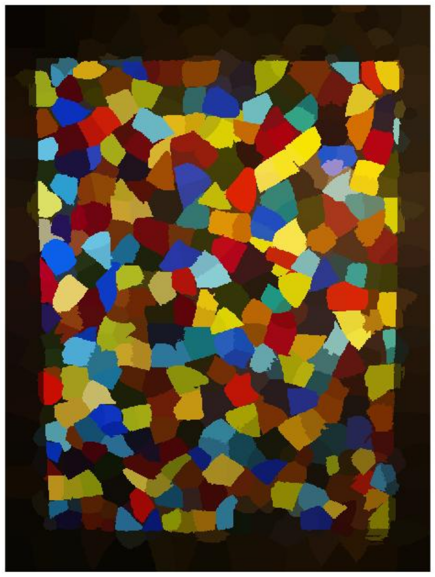
Matlab Result

k = 50 (Too few super pixels)

k = 200 (About right super pixels)

k = 500 (Too many super pixels)

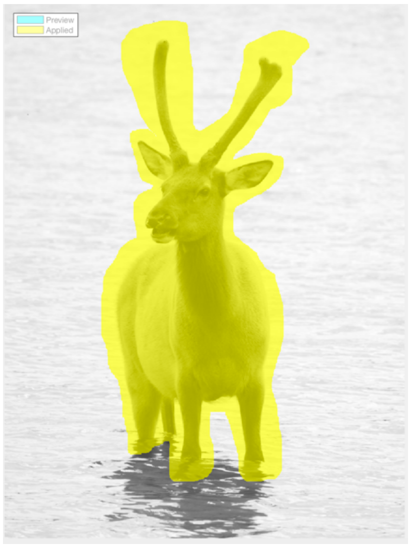
After adjusting k values, we found k=50 would represent “too few” super pixels, the stained glass created by the mean color could not represent the original glass. When k=200, the stained glass would represent the original glass, most of the color trunk was detected by super pixel. Then, we adjusted k to 500, created the “too many” super pixels, some of the same color had been divided into two parts.

```
%% ===================================6.2====================================
```

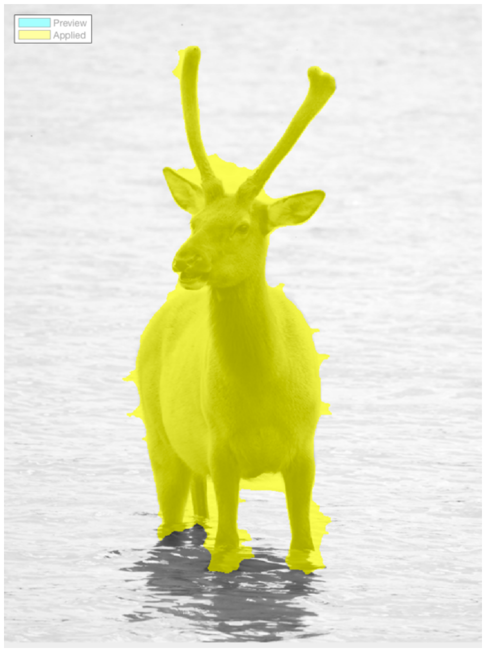
Main Function

```
%% 2
elk = imread('elk.png');
fish = imread('fish.png');
imageSegmenter;
```

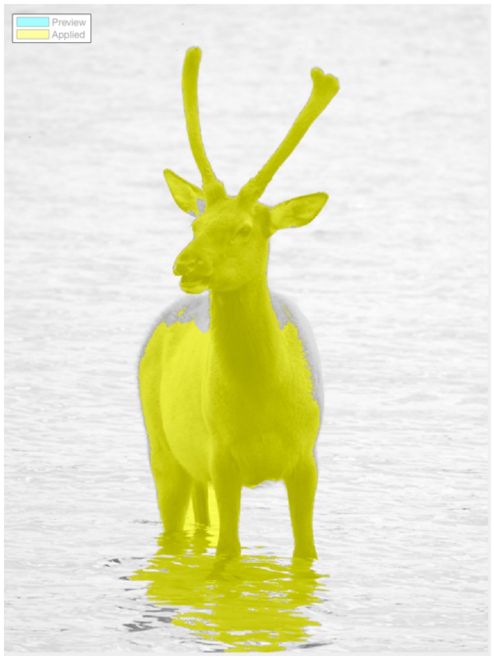
Matlab Result



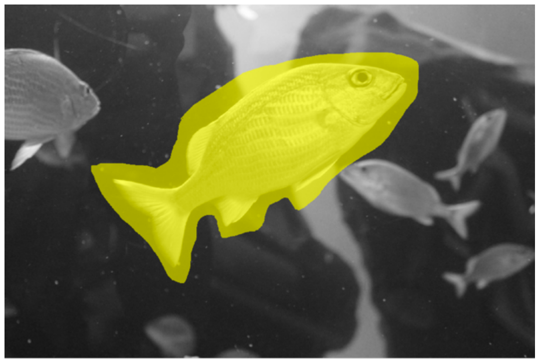
Freehand initial contour

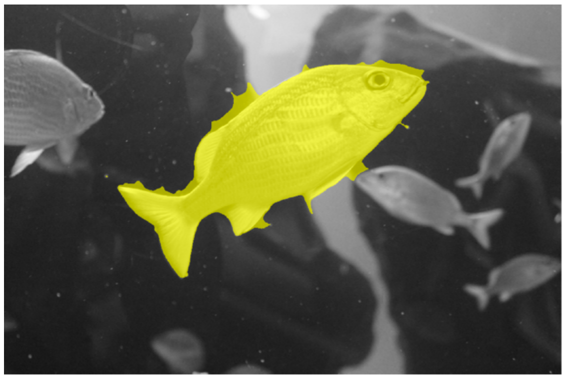


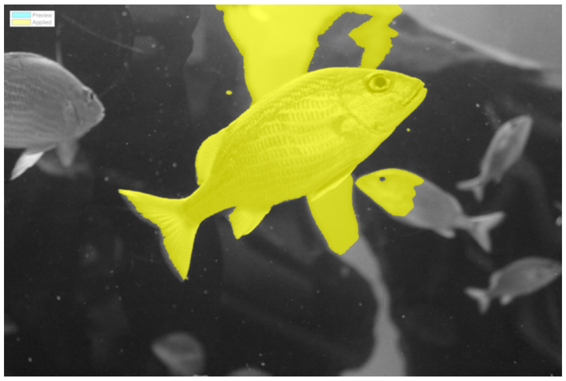
Edge-based 600 iterations



Region-based 300 iterations


Freehand initial contour


Edge-based 600 iterations


Region-based 300 iterations

In this question, we first draw freehand initial contours around the elk and the finish, then we apply edge-based segmentation algorithm with 600 iterations and region-based segmentation algorithm with 300 iterations to the freehand contours. Results has been shown above. Overall speed wise, the region-based is much faster than the edge-based, since the region-based algorithm uses Chan-Vese algorithm, which is a much simple calculation comparing to the edge-based. However, though region-based segmentation is faster, it does not work as well as the edge-based in the above cases. For Elk, edge-base picked up a few water waves where elk' feet stands, and region-base did not pick up the whole elk, it excluded the lighter skin where really close to the color of the water. For fish, edge-based also picked up some of the edges in the background, and region-base included too much background where that part of the water color is close to the fish skin color.

```
%% ===================================6.3=====================================
```

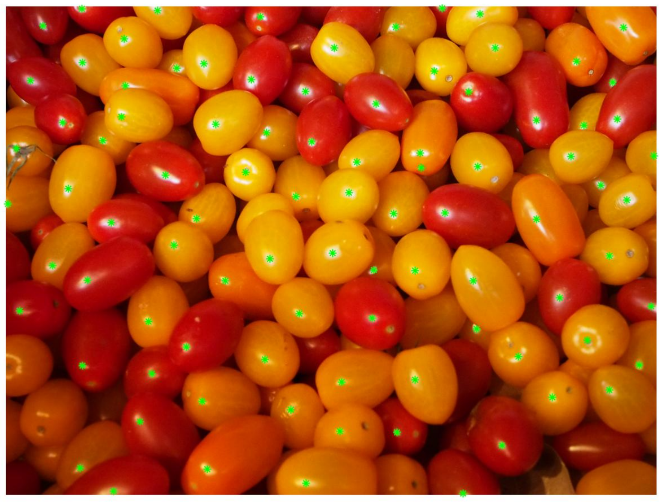
Main Function

```
tomatoes = imread('tomatoes.png');
Rt = 195;
Gt = 75;
Bt = 25;
BW = false(size(tomatoes,1),size(tomatoes,2),size(tomatoes,3));
BW(tomatoes(:,:,1) > Rt & tomatoes(:,:,2) > Gt & tomatoes(:,:,3) > Bt) =
true;

BW1 = imopen(BW, strel('diamond',4));
BW2 = imclose(BW1, strel('diamond',9));
stats = regionprops(BW2);
centroids = cat(1,stats.Centroid);

imshow(tomatoes);
hold on;
plot(centroids(:,1),centroids(:,2),'g*');
hold off;
```

Matlab Result



98 Tomatoes Detected

We first used our data cursor to look through the color values of the white pixels and the tomatoes, then we came up a threshold that later adjusted, threshold red channel to 195, green channel to 75 and the blue channel to 25, we found this to be the most accurate presentation of one white dot per tomato.

Next, we created BW with all false in all three channels, which created a black image, then we write over this black image wherever over the threshold. Then we created a diamond structuring element using strel with two resolution 4 and 9 and used imopen and imclose to isolate out those shapes are closed to a diamond.

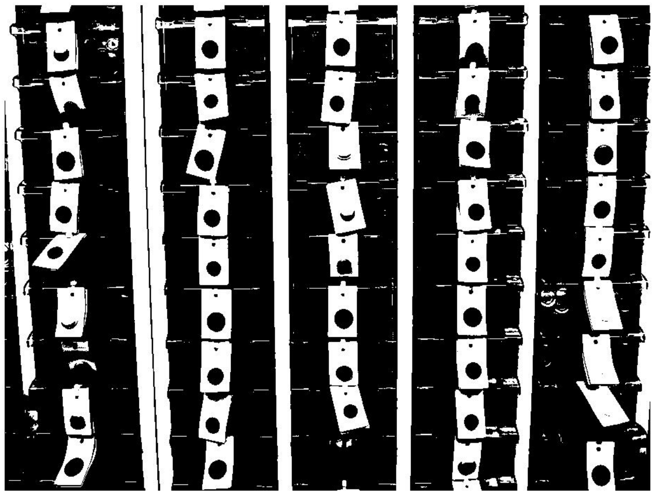
Lastly, we use regionprops to obtain the info about these potential tomato light and find the centroids and plot the centroid locations with green dots ('g') on top of the original tomatoes.*

```
%% ===================================6.4======================================
```

Main Function

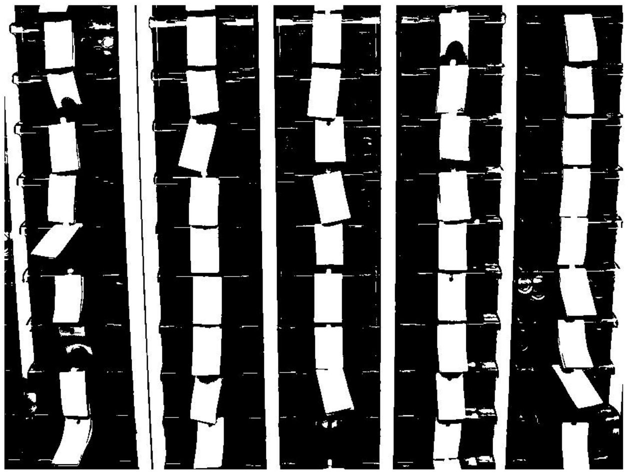
```
% a
buttons = imread('buttons.png');
BW1 = im2bw(buttons,180/256.);
% b
BW2 = bwfill(BW1,'holes');
% c
BW3 = im2bw(BW2 - BW1);
% d
stats =
regionprops('table',BW3,'Centroid','MajorAxisLength','MinorAxisLength');
centers = stats.Centroid;
diameters = mean([stats.MajorAxisLength stats.MinorAxisLength],2);
radii = diameters/2;
indicies = find(radii < 8);
radii(indicies) = NaN;
centers(indicies) = NaN;
imshow(BW3);
hold on
viscircles(centers,radii);
hold off
```

Matlab Result



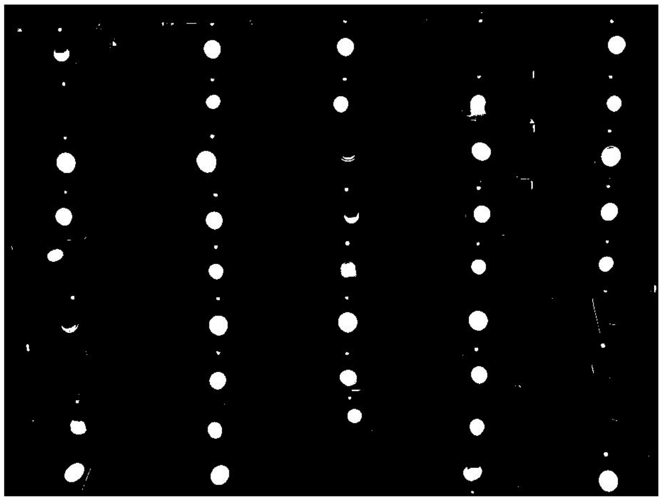
BW1-RGB above high threshold

We first picked out pixels that have over 180 in values in all three RGB channels, this step picked up the white shelves in between racks of buttons and also ignored one or two button racks that button color is closed to the background.



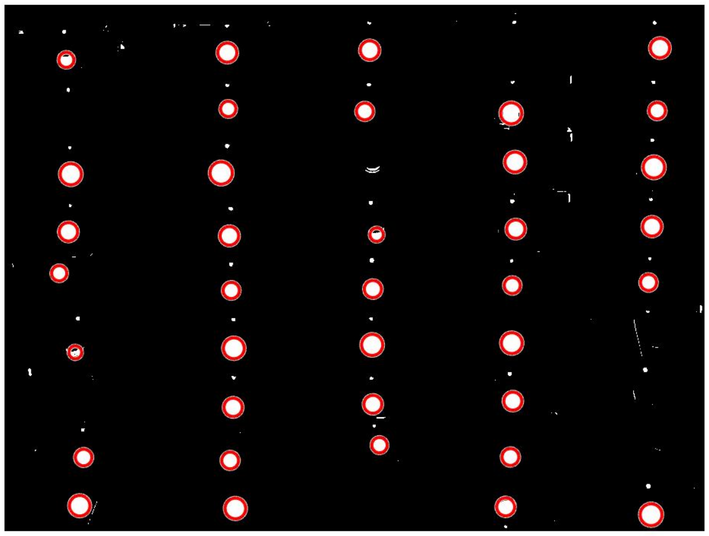
BW2-Holes filled in using bwfill

We used the default bwfill and filled in all the holes in the image, which are the locations of the buttons. One or two holes were not filled, since some buttons are not closed from BW1 and open area cannot be filled by bwfill.



BW3-Subtraction of precious two images

Then, we subtracted BW1 out of BW2 then convert the double back to binary using im2bw, the result returned us the detected buttons location with small areas of extra junk.



Isolate out the buttons

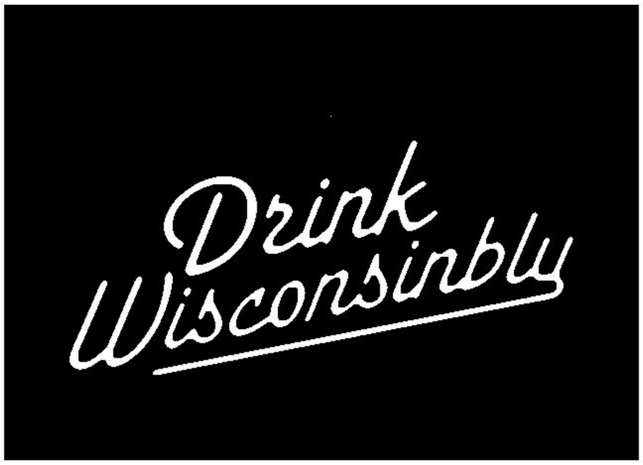
Lastly, we use region props and returned the stats of BW3, which included the centroids, major and minor axis length. In our algorithm, we first locate all the centroids, then find the half of the average of the major and minor axis length of each centroid and consider this as the button's radius. Now the radius has been stored in a list, we threshold those have a radius above 8, this effectively eliminated the small extra junk. We plotted these circles against BW3 and they do seems like the buttons we want to find!

```
%% ===================================6.5====================================
```

Main Function

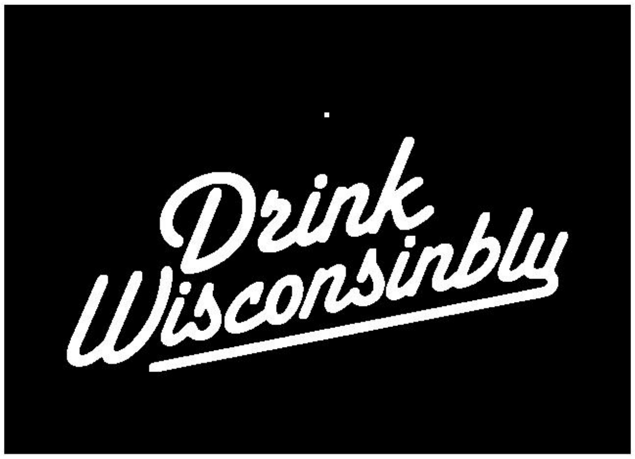
```
%% 5
% a
d3 = strel('disk',3);
% b
drink = imread('drink.png');
drink2 = imerode(drink,d3);
% c
drink3 = imdilate(drink2,d3);
% d
drink4 = imerode(imdilate(drink,d3),d3);
```

Matlab Result



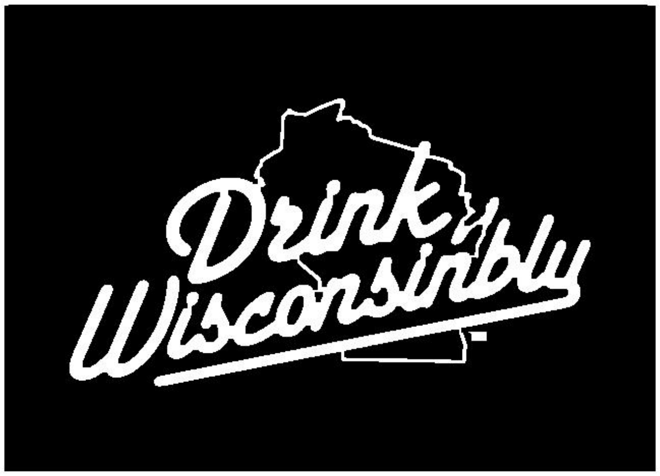
drink-Erode Image

The Wisconsin state logo was eliminated by our imerode, imerode with a disk structure of radius 3 closed the gap of anything with width smaller than 3. The Wisconsin state logo is much narrower than drink wisconsinbly. So only the drink wisconsinbly remains.



drink2-Erode then dilate

Next, we dilated the eroded image, drink wisconsinbly went back to its original width since dilate will make existing gap bigger. A not important small noise was also amplified once dilated.



drink3-Dilate then erode

Now the order is reversed, dilate then erode produced a different picture then erode then dilate, morphological closing instead of morphological opening. This operation made some of the originally isolated character connected, like the ‘s’ and the ‘c’. The difference between the two result would be one is for closing gaps and another is for connecting gaps.