EE475 HW#5 Sezen Perçin

1.Gradient and Laplacian

Since Sobel the coefficients in sobel operators are equal to 0 when summed up, their response is expected to be near zero when there was no important change in their scope. Therefore, these operators are used to indicate where important change on pixel values happens such as the edges of the 1111 area in our case. These operators are accepted to yield gradient components gx and gy.

The Laplacian that is seen lastly is also a derivative operator, but it is a second-order one.

Gradient components calculated with Sobel operators:

```
gx =

     0     0     0     0     0     0     0
     0     1     3     4     4     4     4
     0     1     3     4     4     4     4
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
```

```
gy =

     0     0     0     0     0     0     0
     0     1     1     0     0     0     0
     0     3     3     0     0     0     0
     0     4     4     0     0     0     0
     0     4     4     0     0     0     0
     0     4     4     0     0     0     0
```

Calculated G=abs(gx)+abs(gy) and Laplacian:

```
G =

     0     0     0     0     0     0     0
     0     2     4     4     4     4     4
     0     4     6     4     4     4     4
     0     4     4     0     0     0     0
     0     4     4     0     0     0     0
     0     4     4     0     0     0     0
```

```
L =

     0     0     0     0     0     0     0     0
     0     0    -4    -4    -4    -4     0     0
     0    -4    -8    -2    -4    -4     0     0
     0    -4    -2     8     4     4     0     0
     0    -4    -4     4     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
```

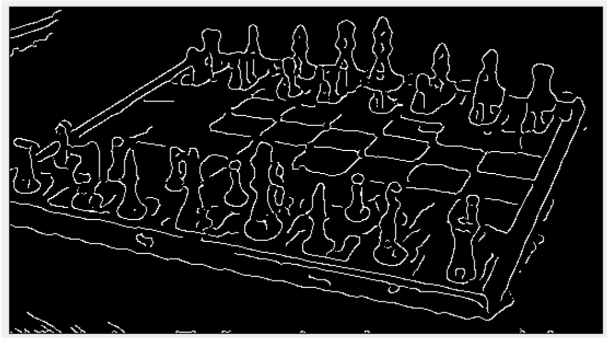
2. Hough Transform

For the Hough Transform, first the edges are determined using the Canny-Edge detector. Then those edges are put in the operation. For every lines that were present in the image, there is a line equation in the form y=mx+b that can be determined by the (x,y) coordinates of the pixels that the line passes through. In Hough transform a new parametrization is made and those m and b values are tried to be determined in a new space called Hough Space. In this space, the variables are rho and theta and the new equation for the point is determined by p=x*cos(theta)+y*sin(theta). For theta values ranging in [-90:5:90] we calculate the p values and save this record on an accumulation matrix with the corresponding theta and p indexing. Cells with the most records indicate the information of a shape (line in our case) in the image. Then the lines are determined as y=(p-xcos(theta))/sin(theta).

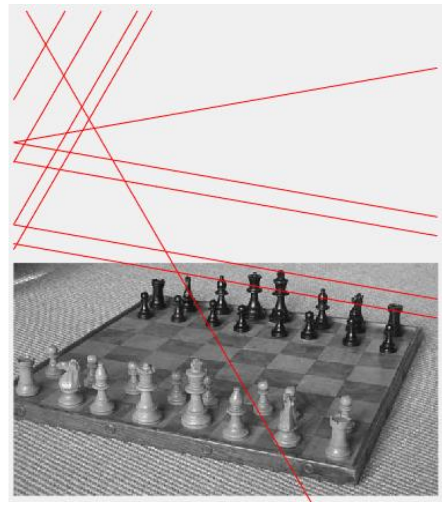
Since the image has this carpet pattern full of lines, first a Gaussian smoothing filter is used to decrease the effects of these patterns:

Unfortunately, I Could not find a way to intersect the hough lines I found with the original image.

Edges found by Canny-Edge Detector:



Hough Lines found:



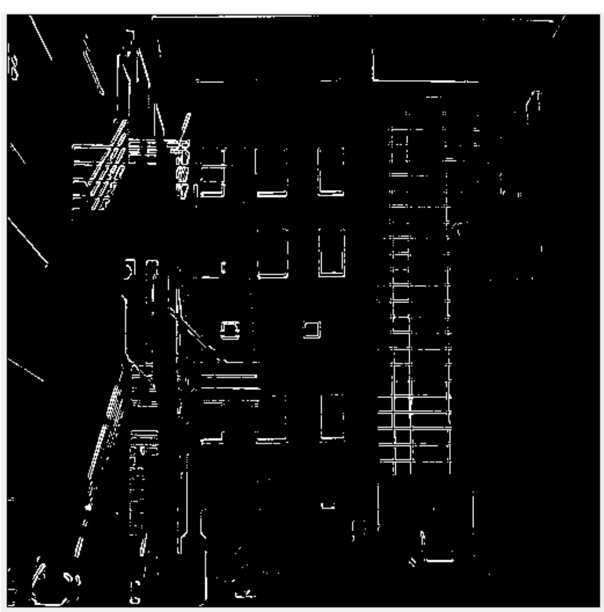
3.Edge Linking and Completion:

Normally the size of the Gaussian mask should be an odd number, greater or equal than 6*sigma, by trying I found that 2*ceil(2*sigma)+1 would work better for our case.

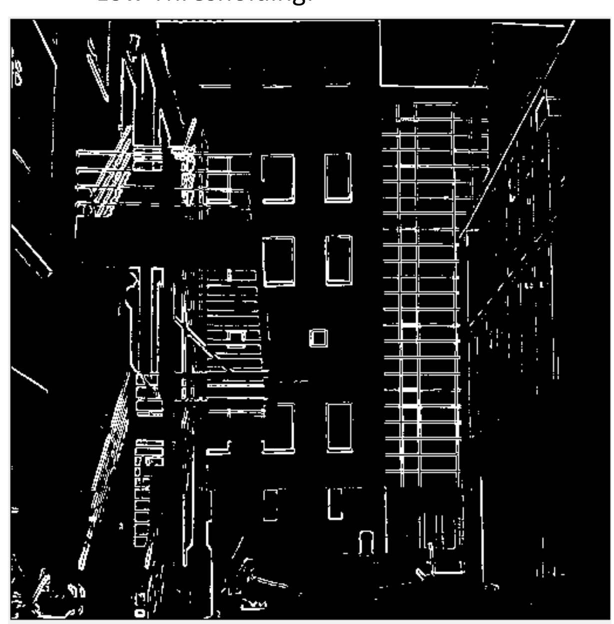
Since thresholds as 95,80 or 70 percent do not give meaningful result, as our professor mailed us, I took high threshold as 0.35 and low one as 0.25.

The aim was to determine significant edges by high thresholding and then filling the gaps between those with the help of information we gained from low thresholding. Then I look for the 8- neighborhood of every significant edge that we obtained from high thresholding. If there was an overlapping between the low thresholding results and the target pixel, this pixel was marked as a continuation of the edge. The other data points that low thresholding results contain are neglected.

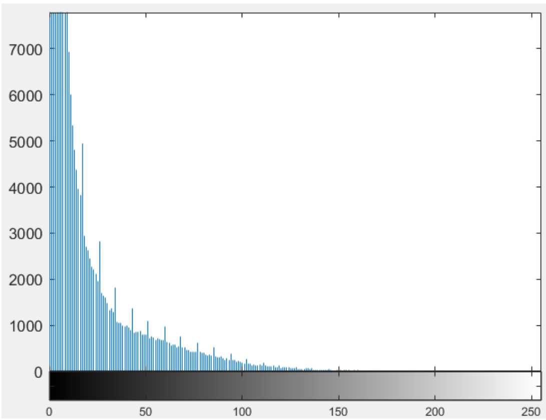
High Thresholding:



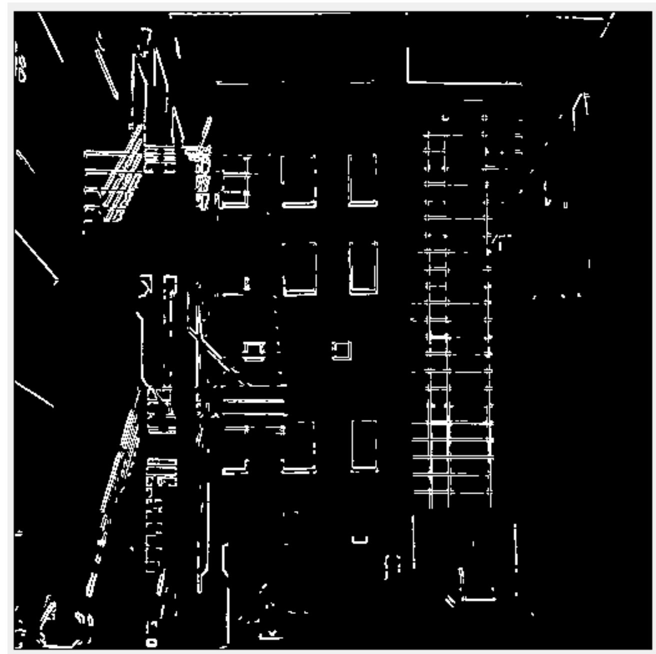
Low Thresholding:



Histogram of the gradient field:



Result:



4.SEGMENTATION by REGION GROWING

For this question I followed the steps below:

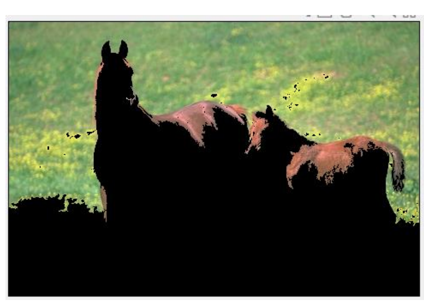
1)Random point are selected from the regions that were separated on the ground-truth boundaries. I selected more than one because otherwise the whole process was not so accurate and it was slow.

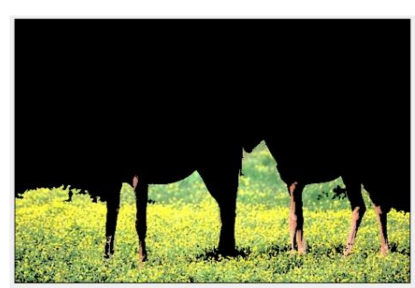
2)Than the sum of the RGB values of these regions with the number of pixels in the regions are calculated to obtain the mean RGB values of the region.

3)Until the until no pixels are added as new to the regions a while loop is operated. In this loop, for every pixel, the 8 neighborhood is checked if it contains a labeled pixel. Than if the square of the norm of the difference between labeled pixel and the target one is below the threshold value (in my case it was 28), then the target pixel is labeled with the same one as the mentioned pixel. Then the new mean value and the number of the pixels of the region is calculated to be considered on the next turn of the loop.

4)If there was no region change anymore, threshold value is increased by 10 and the whole operation on (3) is repeated until number of the pixels are above 150000. (The total number of the pixels were around 152000.)

For the image with the horses I chose 4 regions. Results:





Since the RGB values of the two horses are similar and the head of the first horse is darker than its value, pixel region of the second horse grew faster than the first one and contained the body of the first horse. More seeds could have been chosen from the body of the first horse. I chose mines near the head.

For dee picture I chose 3 regions. Since there were color differences on the deer and my points were in darker area, some of the background contained a piece from the body of the deer:

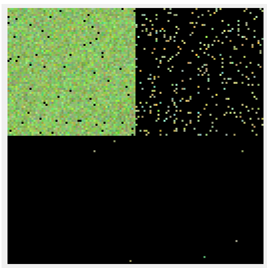
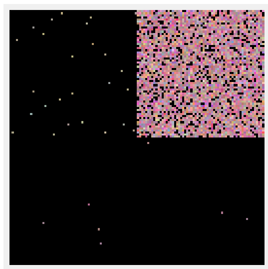
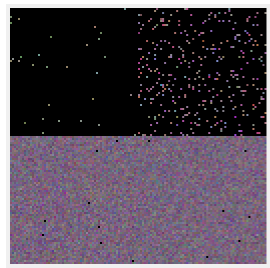
5. SEGMENTATION by CLUSTERING

For the last question I chose 3 seed points as:

```
%green x=30 y=32 r g b
C1=[111 205 77];
C11=[0 0 0];
%pink x=97 y=34
C2=[206 106 156];
C22=[0 0 0];
%purple x=58 y=93
C3=[125 103 121];
C33=[0 0 0];
```

Also, I chose the epsilon value that will be used as a factor to decide if there was a significant change in the new vectors with respect to its old from from the older steps of iteration. I chose it as 0.001 and if the change in vector was below that epsilon value for all the segments, the operation was meant to be stopped.

Results when displayed separately Part 1:

Since only the RGB values are used as the selection criteria, it can be seen that the segments are divided based on their colors.

There were problems with my part2 algorithm that yields more unaccurate results. But I know that in part 2 determining vector contains more pixel attributes. This means that the pixels have to be compared on more levels and for a pixel to be belong to a region, it has to satisfy more selection criteria. Therefore I would except the second part to deliver more accurate results.