BLG 453E Project 4 Report

Computer Vision

Tuğba Özkal

150120053

17.12.2017

BLG 453E Project 4 Report

Harris Corner Detection

Compiling: $ python3 Q1.py

In this section, corners of blocks image are detected by using Harris Corner Detection method. The algorithm is given below:

```
def harris(I):
    row, col = I.shape

    sig = 0.3
    M = gaussian_filter(I, sigma=sig)
    dy, dx = np.gradient(M)
    Ixx = dx**2
    Ixy = dy*dx
    Iyy = dy**2

    thresh = 50000000000
    newImg = I.copy()
    color_img = cv2.cvtColor(newImg, cv2.COLOR_GRAY2RGB)
    window = 3
    k = 0.06
    for y in range(window, row-window):
        for x in range(window, col-window):
            windowIxx = Ixx[y-window:y+window+1, x-window:x+window+1]
            windowIxy = Ixy[y-window:y+window+1, x-window:x+window+1]
            windowIyy = Iyy[y-window:y+window+1, x-window:x+window+1]
            Gxx = windowIxx.sum()
            Gxy = windowIxy.sum()
            Gyy = windowIyy.sum()

            det = (Gxx * Gyy) - (Gxy**2)
            trace = Gxx + Gyy
            r = det - k*(trace**2)

            if r > thresh:
                color_img.itemset((y, x, 0), 255)
                color_img.itemset((y, x, 1), 0)
                color_img.itemset((y, x, 2), 0)
    return color_img
```

Here, grayscale image is smoothed by gaussian filtering. Then, gradients of the image in x and y directions are calculated. A threshold value is determined. It is used for corner detection.
A window with 3 size is defined. It walks around the image and detects corners.
Color image is created. While the window walks around the image, changing are calculated as a matrix.

$$G = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

The determinant of matrix is calculated and then, trace value is found. r value is found by using trace and determinant.

```
det = (Gxx * Gyy) - (Gxy**2)
trace = Gxx + Gyy
r = det - k*(trace**2)
```

k: 0.04 - 0.06

If r value is bigger than threshold, that point is corner, else that point is edge or flat.

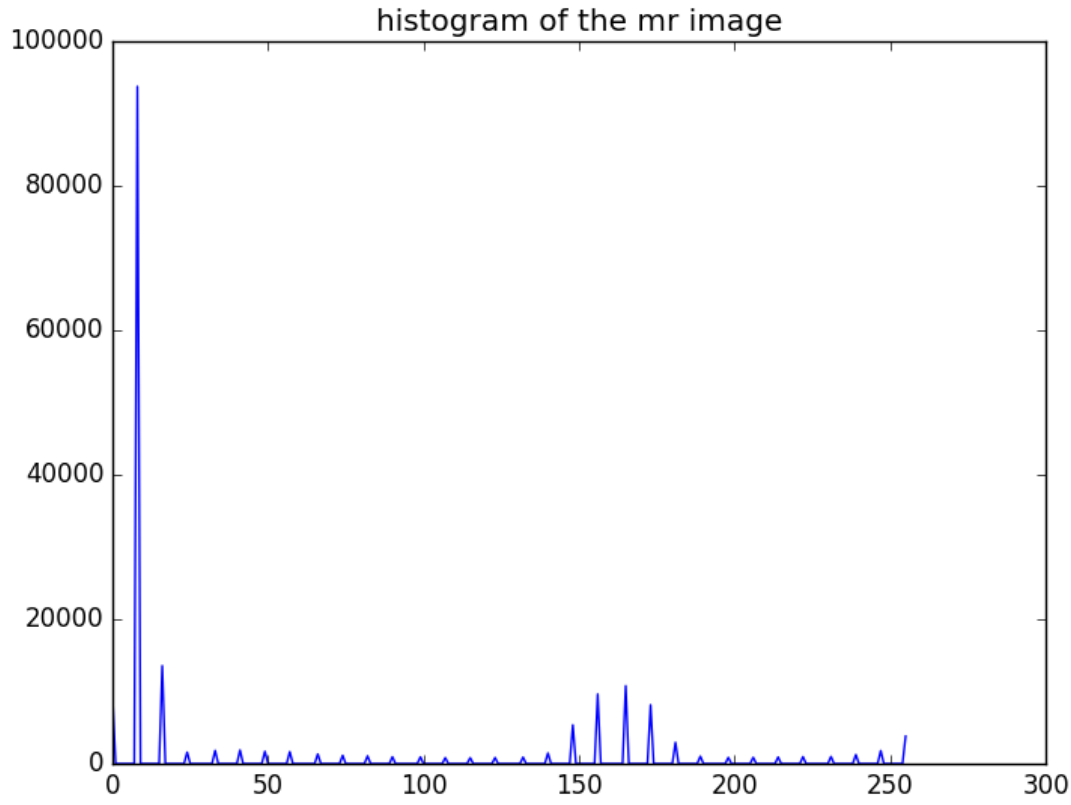
My algorithm is work for chessboard but it does not give a good solution for blocks image.

Segmentation of Tumor Region From a MR Image

Compiling: $ python3 Q2.py

Firstly, after read the image histogram is calculated.

```
for i in range(row):
   for j in range(col):
      hist[arr[i][j]] += 1
```

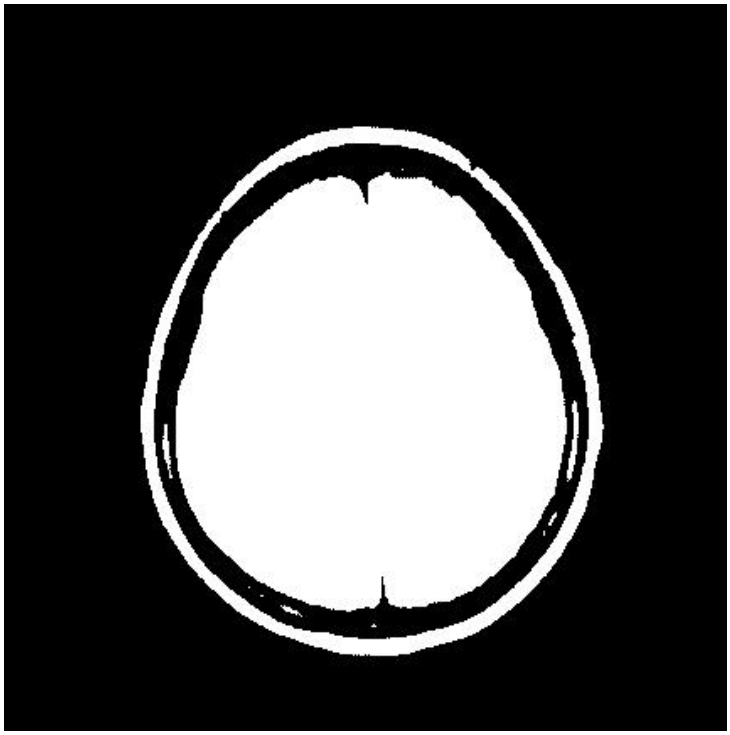


According to histogram, threshold value is determined as 100.

```
thresh = 100

for i in range(row):
   for j in range(col):
      if(arr[i][j] < thresh):
         arr[i][j] = 0
      else:
         arr[i][j] = 255
```

If the pixel value is greater than threshold value, it is set to 255, else it is set to zero. The occurred image is given below.



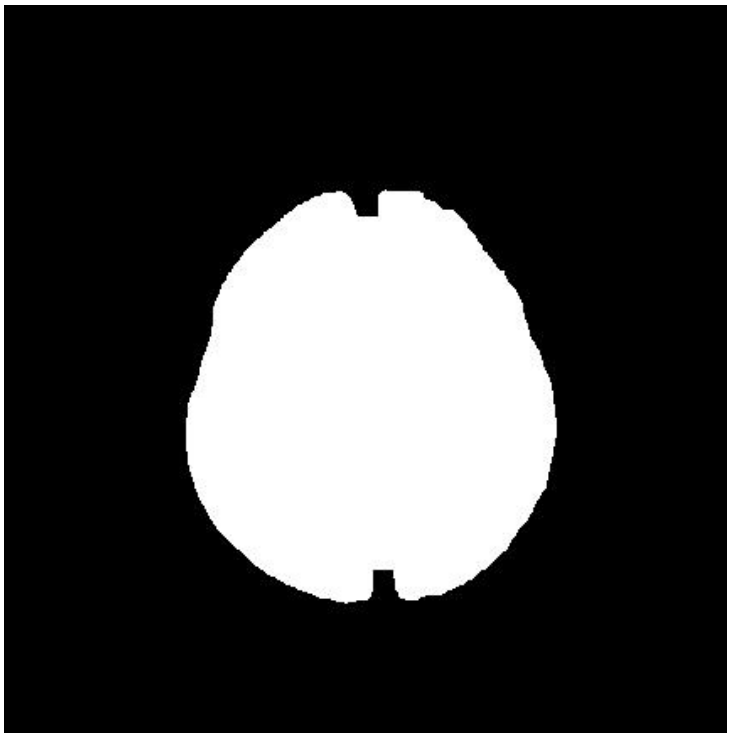
A kernel is created which is 12x12 sizes. Its all elements are equal to 1.

```
kernel = np.ones((12,12),np.uint8)
```

Then, erosion is applied to new image. In this way, skull of the brain is removed.

```
erosion = cv2.erode(arr,kernel,iterations = 1)
erosion = np.array(erosion, dtype=np.uint8)
t = Image.fromarray(erosion)
t.save('skull.jpg')
```

The new image is given below.



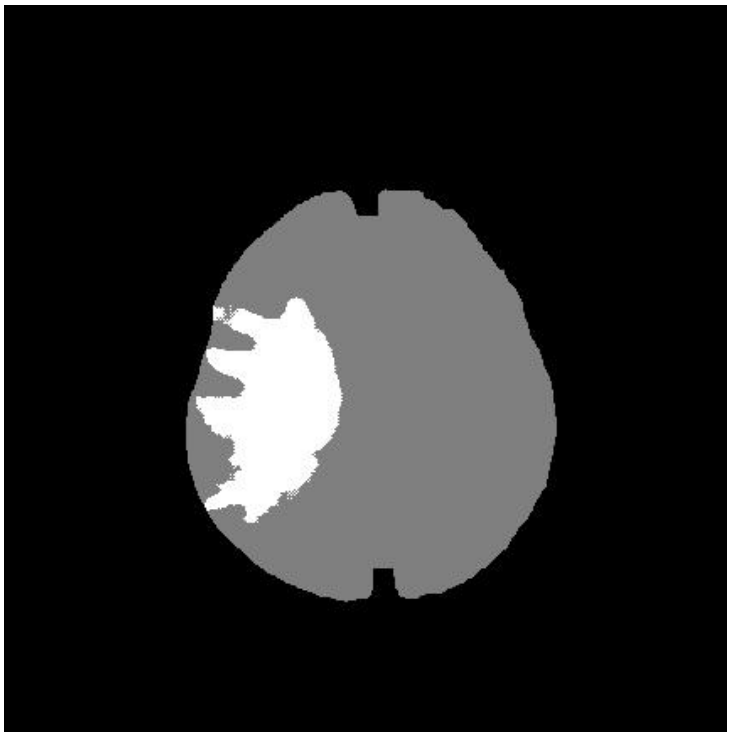
According to histogram, another threshold value is determined to 220. By using this threshold value, tumor is detected.

```
thresh_skull = 220
for i in range(row):
   for j in range(col):
      if(skull[i][j] < thresh):
         skull[i][j] = 0
      elif(skull[i][j] > thresh and skull[i][j] < thresh_skull):
         skull[i][j] = 127
      else:
         skull[i][j] = 255

      if(skull[i][j] > erosion[i][j]):
         skull[i][j] = 0
tumor = np.array(skull, dtype=np.uint8)
s = Image.fromarray(tumor)
s.save('tumor.jpg')
```

If the value of the point is lower than first threshold (100), it is set to 0. It means that the zeros are outside of the brain. If the point value is between in two threshold values (100-220), it is set to 127. This area means the brain. If the point value greater than second threshold (220), it is set to 255. That means the

tutor region. But in this way we cannot obtain the brain and tutor without skull. So, point value and erosion image point value are compared and according to result, skull is removed again.

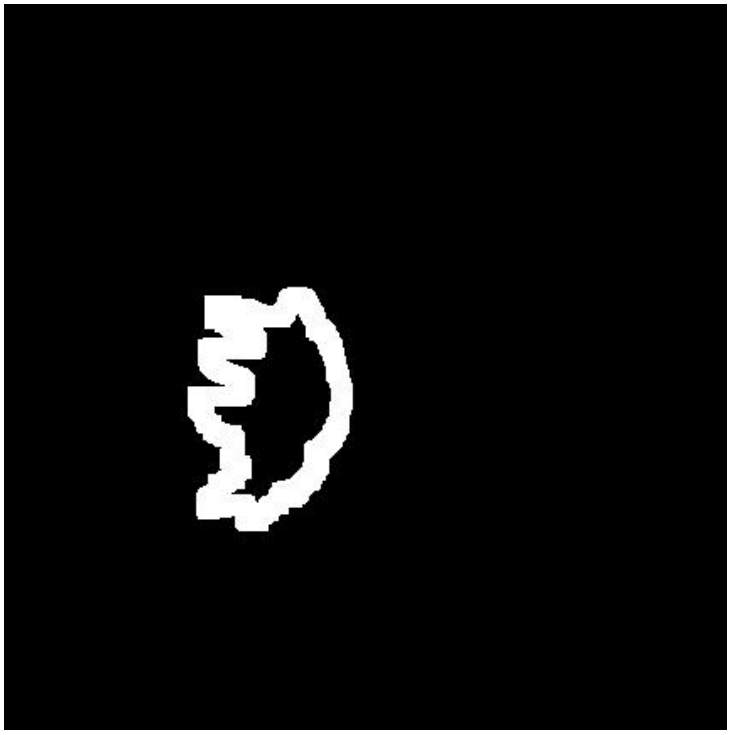


Detected tumor region image is given below.

Then, by using algorithm which is given below, region of tumor is detected. Here, morphologyEx function is used.

```
for i in range(row):
   for j in range(col):
      if(tumor[i][j] < 255):
         tumor[i][j] = 0
      else:
         tumor[i][j] = 255

boundary = cv2.morphologyEx(tumor, cv2.MORPH_GRADIENT, kernel)
boundary = np.array(boundary, dtype=np.uint8)
s = Image.fromarray(boundary)
s.save('boundary.jpg')
```

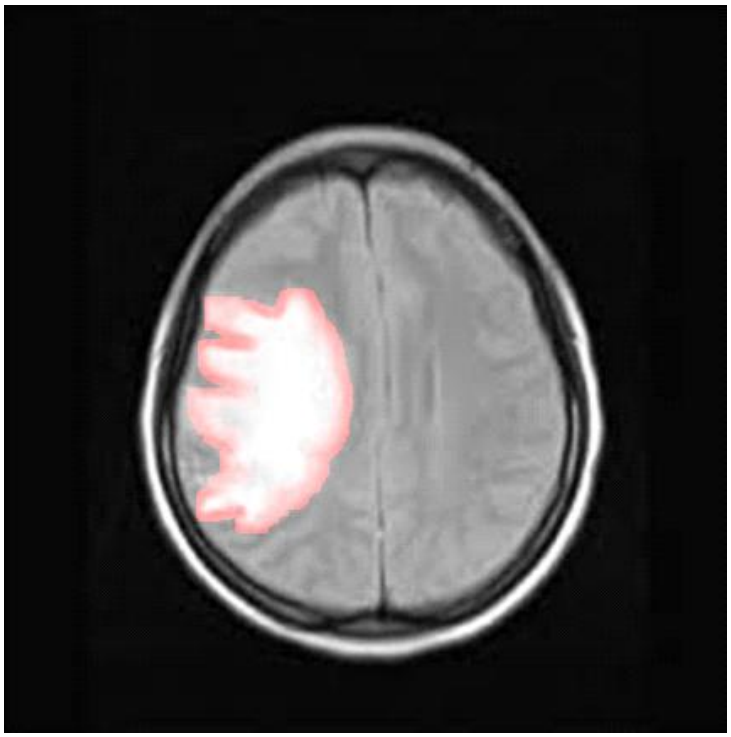


Boundary of the tumor is given below.

Then, the boundary pixels are implemented by red colour on the original image.

```
newImg = img.copy()
color_img = cv2.cvtColor(newImg, cv2.COLOR_GRAY2RGB)
for i in range(row):
   for j in range(col):
      if(boundary[i][j] == 255):
         color_img[i][j][0] = 0
         color_img[i][j][0] = 0
         color_img[i][j][0] = 255


final = np.array(color_img, dtype=np.uint8)
s = Image.fromarray(final)
s.save('final.jpg')
```



Boundary on the original image: