Istanbul Technical University
Faculty of Computer and Informatics
Computer Engineering Department

BLG 453E
Computer Vision
Homework IV

Uğur Uysal - 150140012

Dec 24th, 2018

Contents

1 Usage

The code depends on numpy, opencv and PyQt5. If all requirements are satisfied, then code can be executed.

```
$ python3 filename
```

2 Corner Detection

In first part of homework, Harris Corner Detection[1] algorithm is implemented. In brief, the algorithm checks the gradient changes in windows, if the change satisfies threshold, we mark this point as a corner. It is a very simple definition of the algorithm. Before the calculating gradients, I convoluted the image with a Gaussian filter which has kernel size is three and $\sigma = 1$, in order to remove noise from image. Then we need to calculate image gradients(I_x and I_y), then we can compute G matrix.

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

We choose a proper W (window size), and calculate eigenvalues of matrix, then λ_2 computed and compared to an empiric threshold. If the λ_2 is bigger than threshold point is marked as corner.

2.1 Implementation

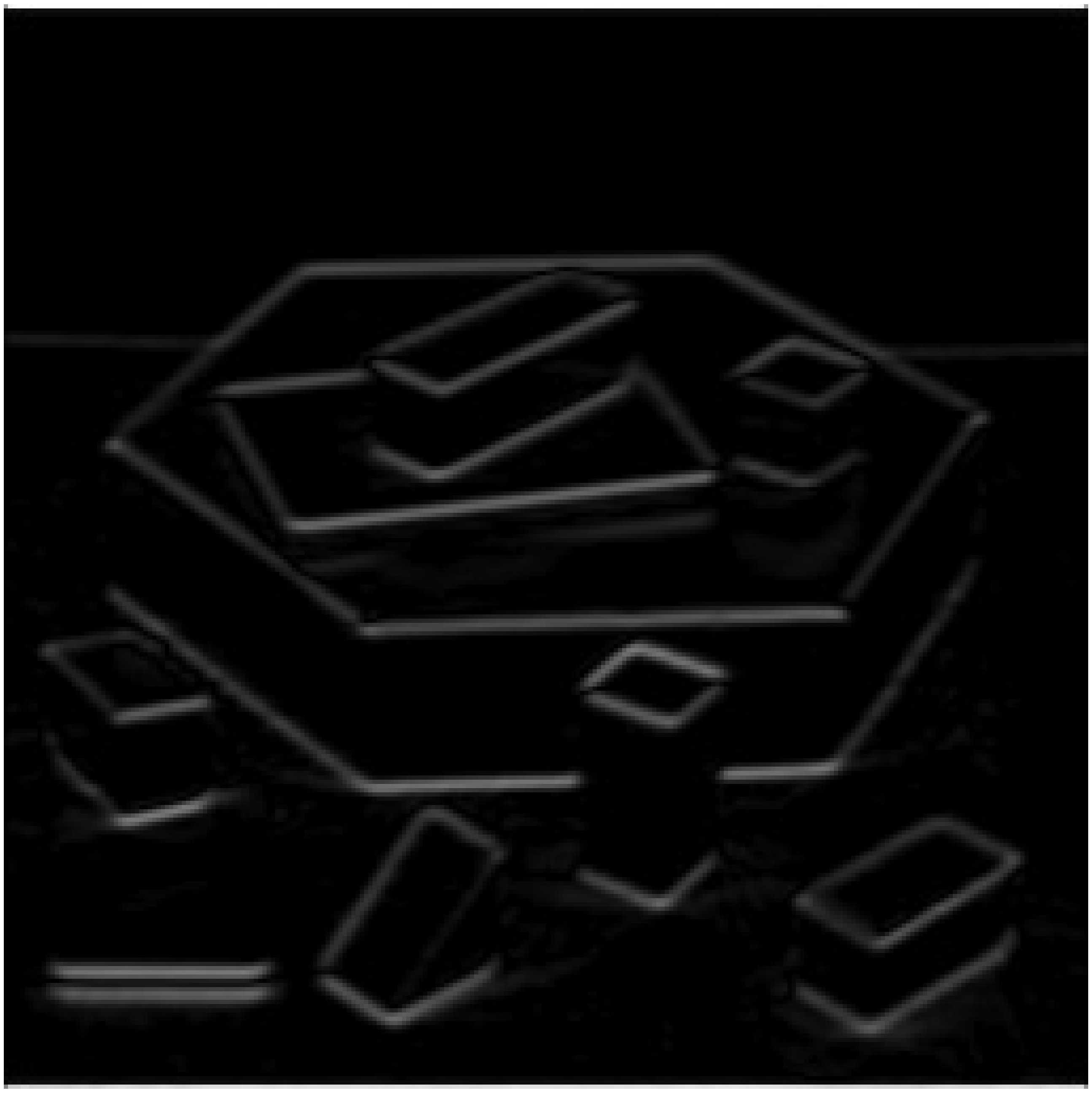
I calculated the gradients with cconvolution filters. I tried Sobel [2] filter and Prewitt [3] filter. I decided to use Prewitt after several tries.

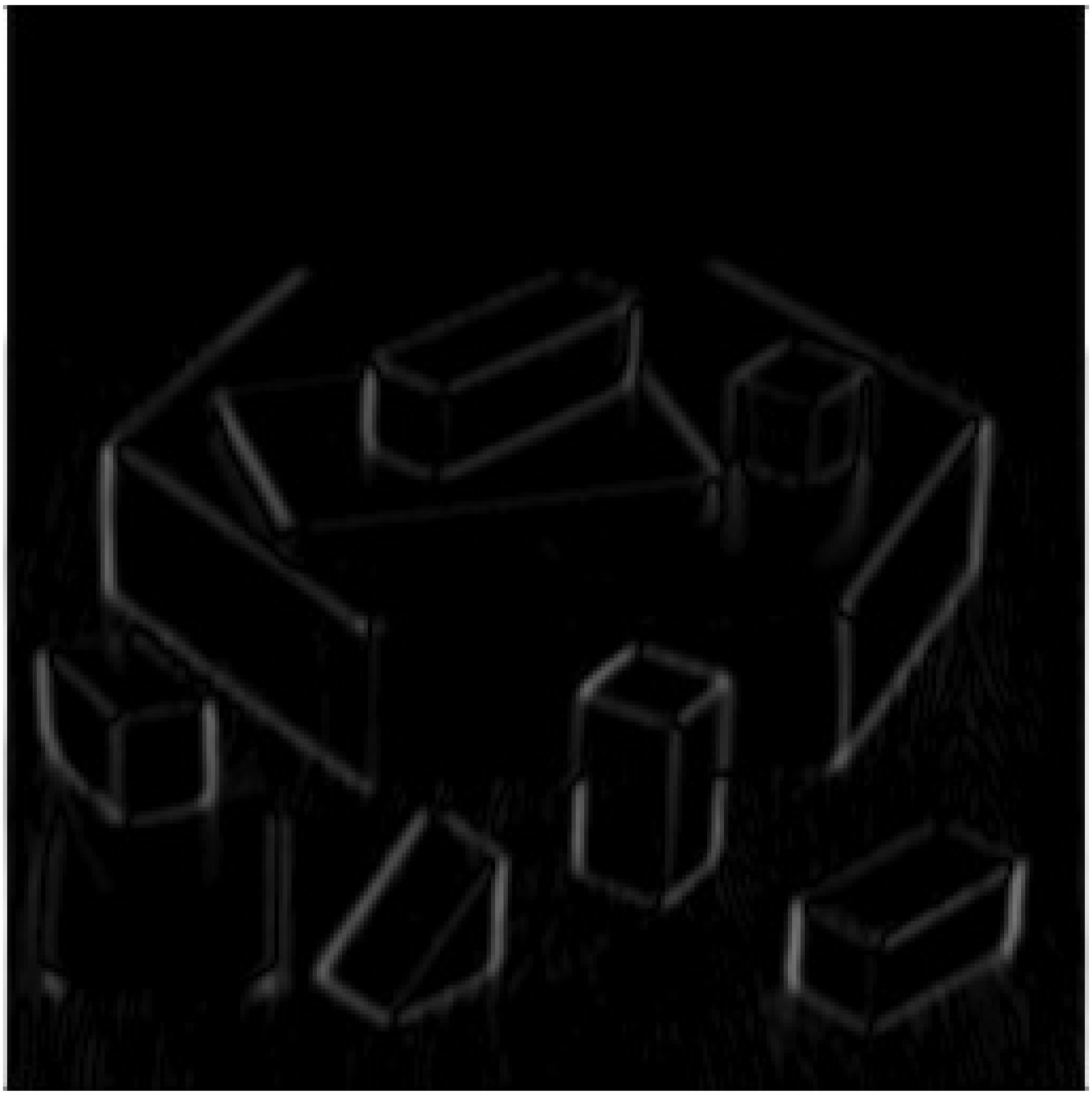
```
def imgConv(img, kernel, threshold=10):
    C = img.shape[2]
    res = np.ones(img.shape)
    for i in range(C):
        res[:, :, i] = signal.convolve2d(img[:, :, i], kernel, mode="same")
    res = np.abs(res)
    res[res < 15] = 0
    res = res.astype(np.float)
    res = 255 * (res-np.min(res)) / (np.max(res) - np.min(res))
    return res
```

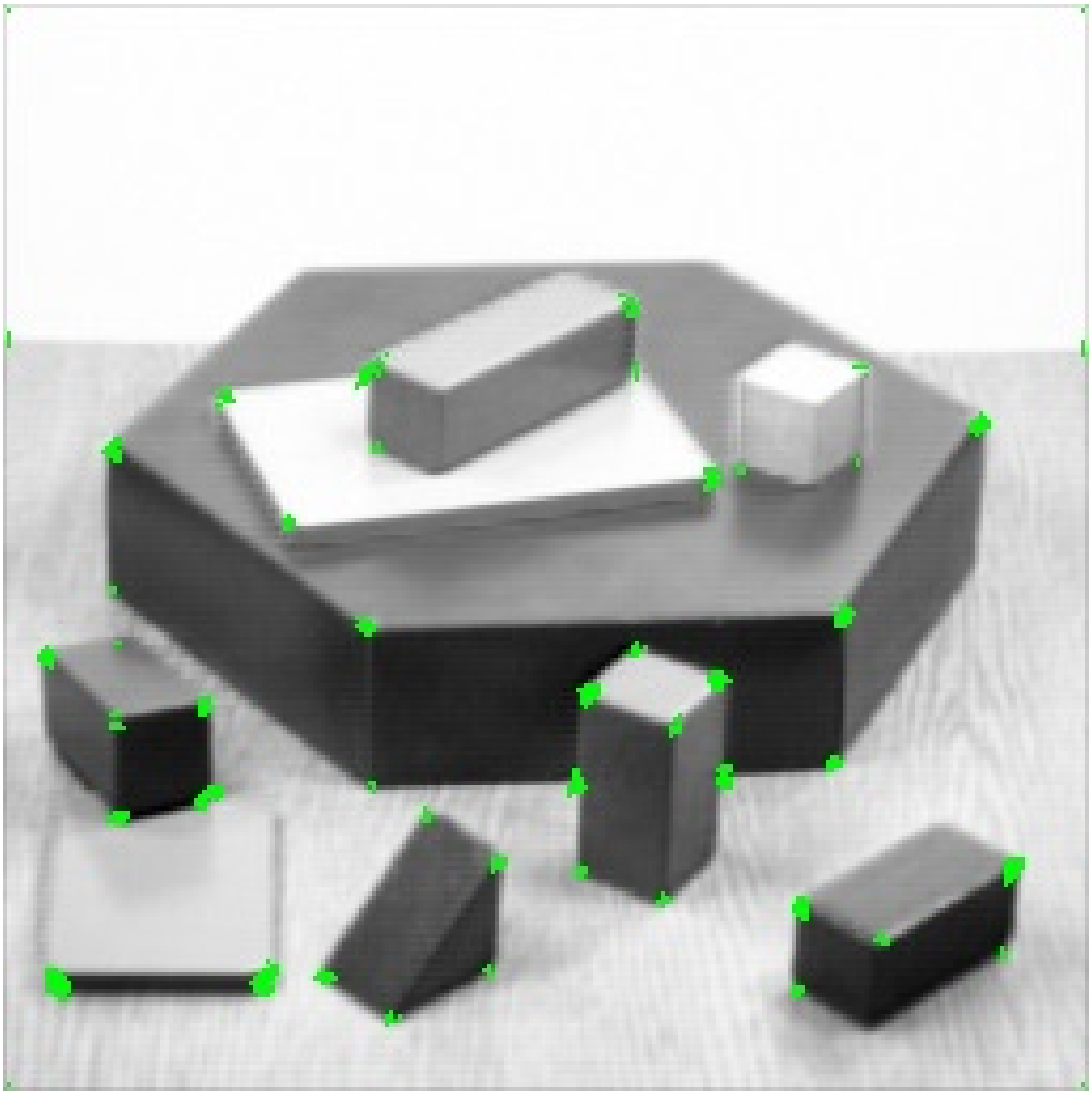
```
smoothImage = cv2.GaussianBlur(img, (ksize, ksize), sigma)
Ix = imgConv(smoothImage, filter_x)
Iy = imgConv(smoothImage, filter_y)

# Calculate Ix2 and Iy2 and Ixy
Ixx = Ix**2
Iyy = Iy**2
Ixy = Iy*Ix
```

After calculating the gradients, sliding window is implemented. I choose threshold as 700. Output example is added.

```
    for y in range(half, h-half):
        for x in range(half, w-half):
            # Normalize sums respect number of elements in window
            Sumxx = Ixx[y-half:y+half+1, x -
                        half:x+half+1].sum() / wsize**2
            Sumxy = Ixy[y-half:y+half+1, x -
                        half:x+half+1].sum() / wsize**2
            Sumyy = Iyy[y-half:y+half+1, x -
                        half:x+half+1].sum() / wsize**2

            G = np.array([[Sumxx, Sumxy], [Sumxy, Sumyy]])

            lamdas, vector = np.linalg.eig(G)
            r = lamdas.min()

            if r > thresh:
                corner_x.append(x)
                corner_y.append(y)

    color_img[corner_y, corner_x] = (0, 255, 0)
    cv2.imwrite("corners.jpg", color_img.astype(np.uint8))
    return color_img
```

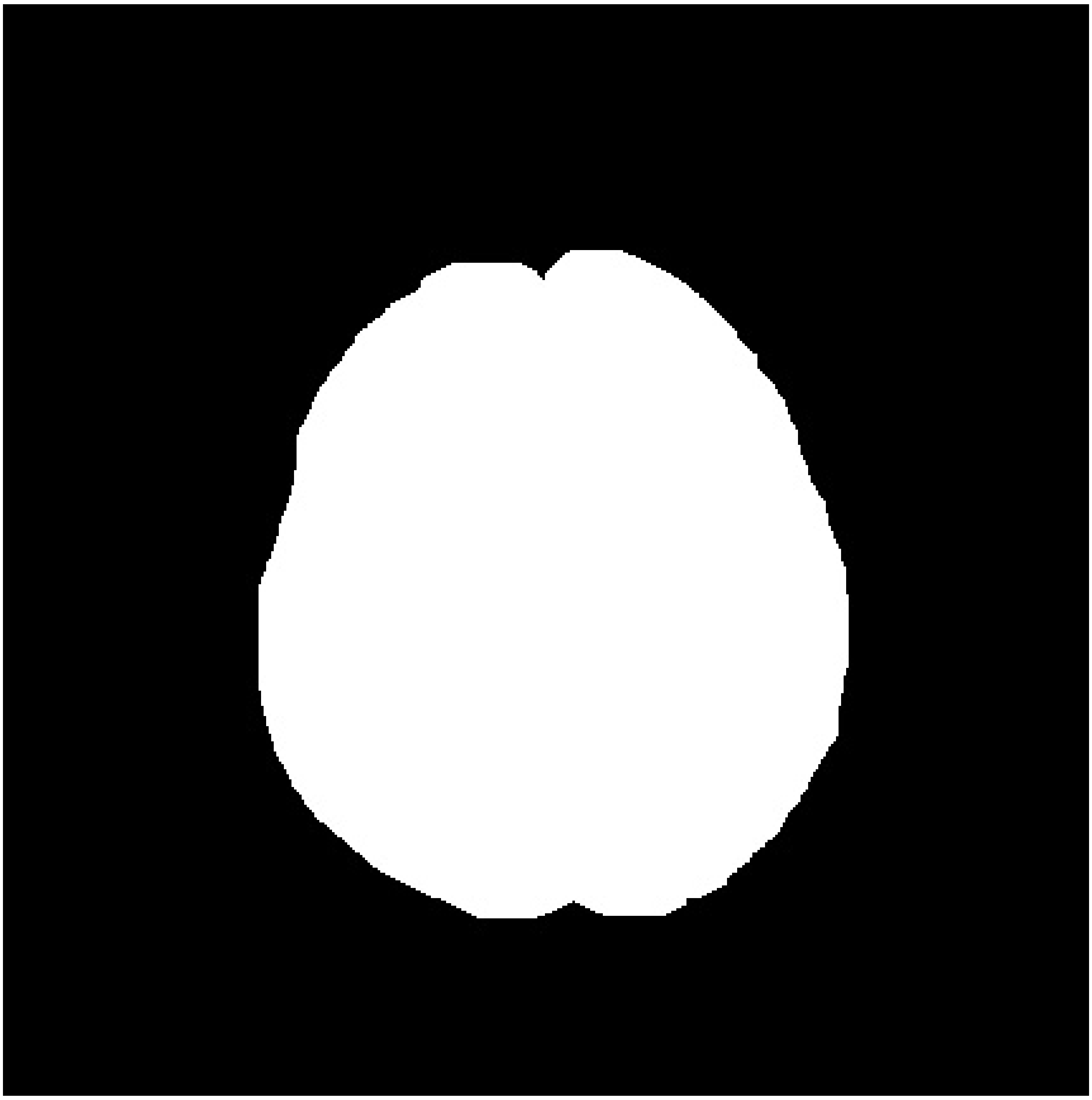






3 Segmentation

In this part homework, we implemented K-means [4] algorithm to MR image of Brain. Objective was detecting the tumor area and finding its boundaries. First I converted image to gray scale using opencv method. I used 55 as the threshold. Then I used "morphologyEx" method in opencv with two different shaped kernel in order get the mask given in the homework PDF.

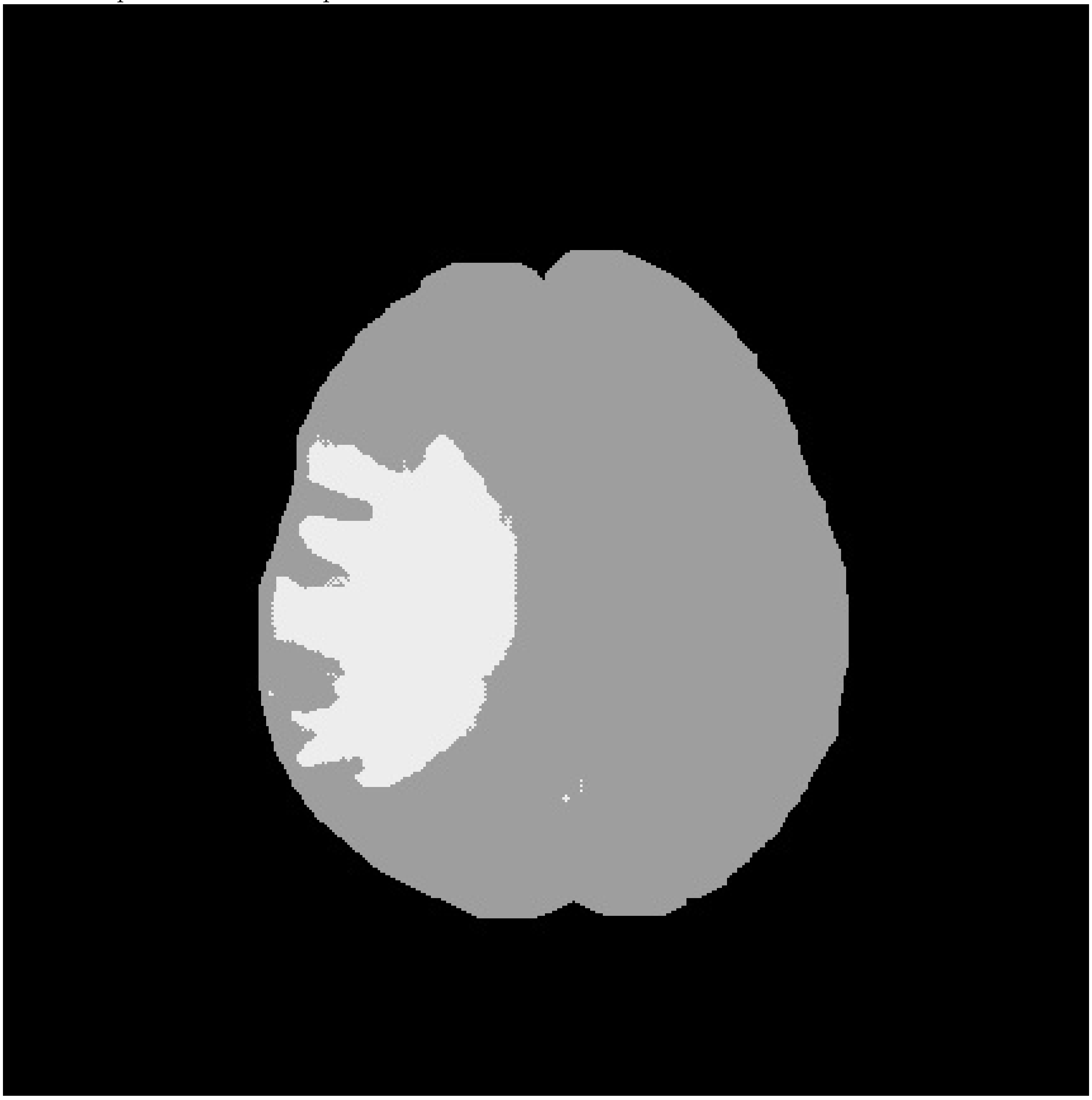


3.1 Segmentation with K means

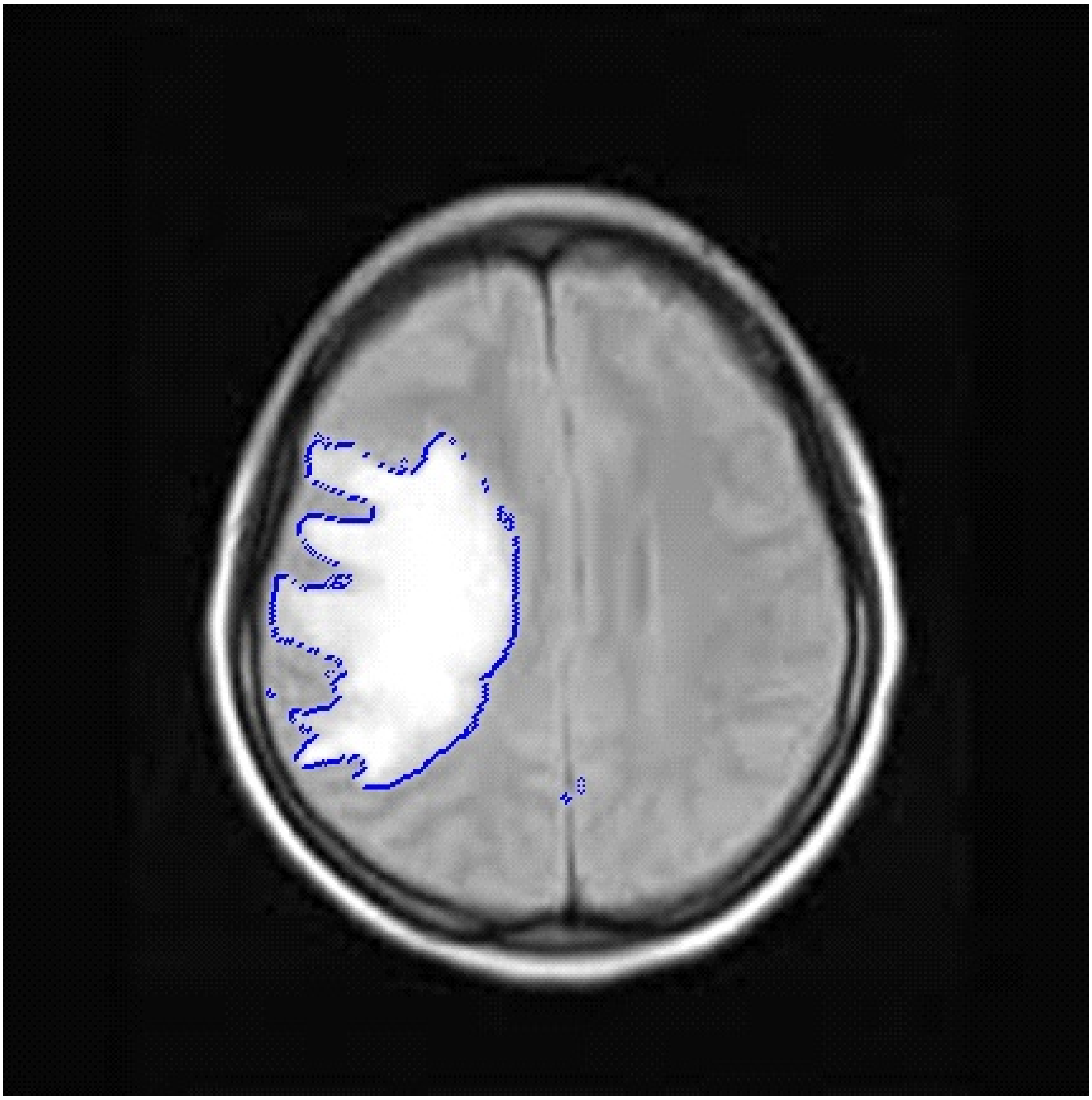
I select non-zero intensity valued points to cluster.

```
def Kmeans(img, K=2, n_iters=20):
np.random.seed(1)
shape = img.shape
x, y = np.where(img > 0)
centers = [randPoint(img, x, y) for i in range(K)]
clusters = np.zeros(shape)
# Converge logic should be different
for i in range(n_iters):

    for i in range(len(x)):
        clusters[x[i], y[i]] = closest(
            img[x[i], y[i]], (x[i], y[i]), centers)
    for i in range(K):
        mean = np.sum(img[clusters == i+1]) / \
            len(np.where(clusters == i+1)[0])
        centers[i] = mean

clusters[clusters == 1] = round(centers[0])
clusters[clusters == 2] = round(centers[1])
```

Then I painted clustered pixels to same color.



In order to get boundaries, I made an assumption that number of tumor pixels are lesser than number of normal brain pixels. Then I painted normal brain pixels to black, I get gradients from this image and used it as mask. Then I get the pixels that greater than zero to make them blue in original image.



References

[1] Haris Corner Detection

[2] Sobel Filter

[3] Prewitt Filter

[4] K Means