Homework 5

Mochen Linghu id:0025635488

October 22, 2016

1 RGB image segmentation using Otsu Algorithm

- First, we need to divide the RGB image into 3 channels which are Red, Green and Blue channels
- Then we apply Otsu algorithm on each of these 3 channels to get $mask_B$, $mask_G$ and $mask_R$
- In order to get the mask, we construct a 256-level array h so that $h[i] = n_i$, where n_i is the number of pixels which are in the i grayscale level
- Now we want to find a threshold k for the image so that we can recognize foreground and background. The way to determine k is to find a k that maxmize the variance between foreground class and background class:

$$\sigma_B = \omega_0\omega_1(\mu_0 - \mu_1)^2$$

where μ is the mean of each class:

$$\mu_0 = \Sigma_{i=0}^{k}\frac{iP_i}{\omega_0}$$
$$\mu_1 = \Sigma_{i=k+1}^{L}\frac{iP_i}{\omega_1}$$

L is the maximum grayscale level. P_i is the probability of the ith grayscale level in whole picture: $P_i = \frac{n_i}{N}$. And *omega* is the probability of each class:

$$\omega_0 = \Sigma_{i=1}^{k}P_i$$
$$\omega_1 = \Sigma_{i=k+1}^{L}P_i$$

- After k is determined, we regard it as the threshold to construct a mask that the pixels that have lower grayscale value than k in original channel become 0. This is mask represents the foreground in the channel
- We need to repeat this step for several times so we can get good result
- Finally we just combine the mask of 3 channels and logically "AND" them to get the final overall foreground

2 Texture based segmentation

We first convert the image into grayscale. Then we place a window of size $N \times N$ at each pixel and compute the variance of the pixels in the widow. And we do this for 3 different value of N, $N = 3, 5, 7$ as we scan the image. Then we just regard the 3 variances at each pixels as the RGB characterization in previous section and get the foreground.

3 Contour Extraction

The contour is the pixels of foreground that are adjacent to background. We check the 3×3 window for every foreground pixels. If there is at least 1 pixel with zero value in the window, then we consider it is a contour pixel.

4 Observations

The Otsu algorithm is mainly depended on the variance between background graylevel and foreground graylevel, which performs really bad in some situations like lake.jpg and leopard.jpg in this homework. Whenever the background has similar graylevel as the foreground does, it will become hard to get the accurate foreground. If we apply texture based segmentation, it helps get rid off this situation since the texture measure indicate the variance between every pixels and their surroundings. As a result, texture based segmentation prefer to recognize the pixels with high variances with surroundings as foreground. However, in lake.jpg, the foreground pixels have low variances which lead to recognization error.

5 Result

5.1 lake.jpg

Figure 1: Original image of lake.jpg

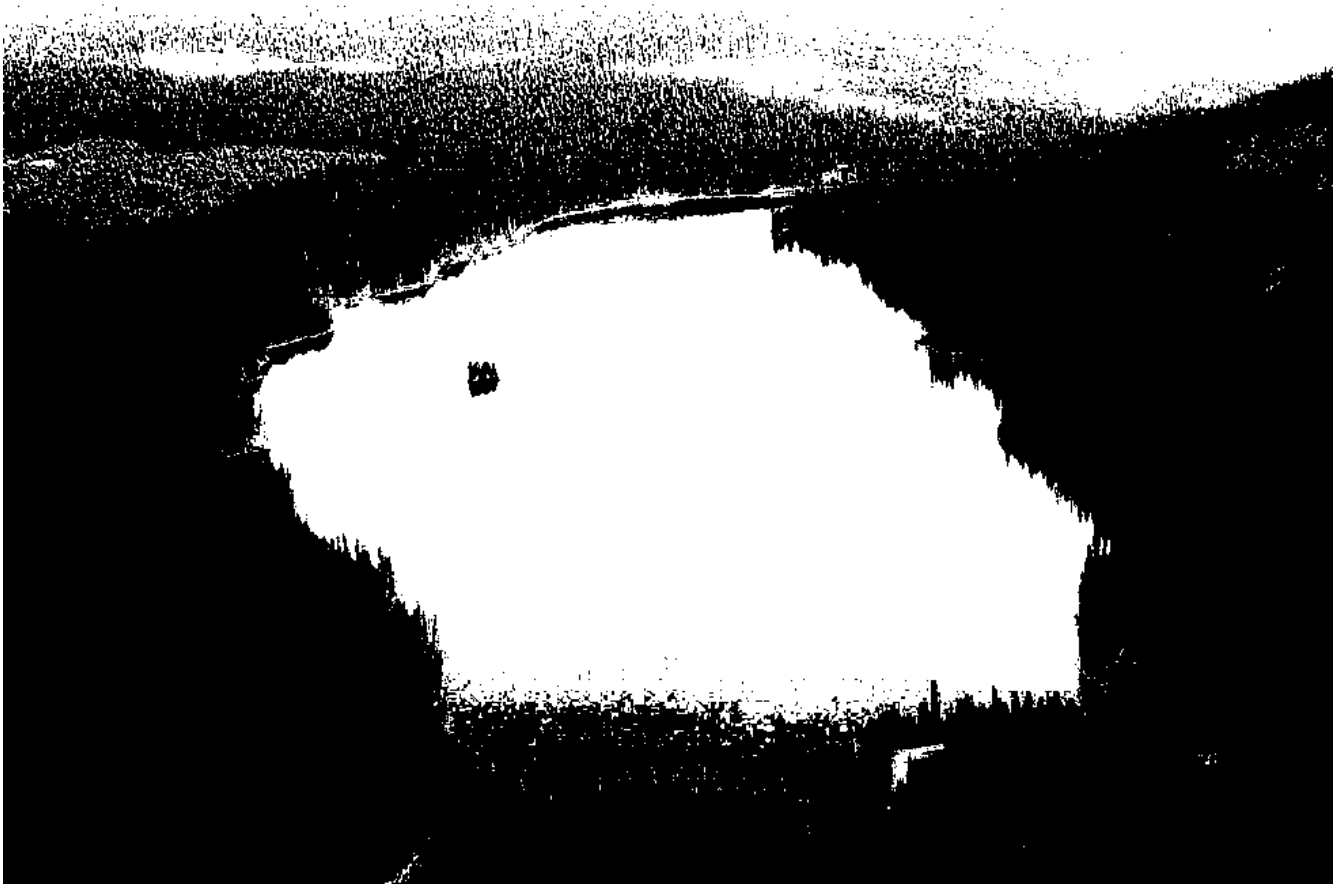

Figure 2: Blue channel mask for lake.jpg

Figure 3: Green channel mask for lake.jpg

Figure 4: Red channel mask for lake.jpg

Figure 5: Foreground with only Otsu algorithm for lake.jpg

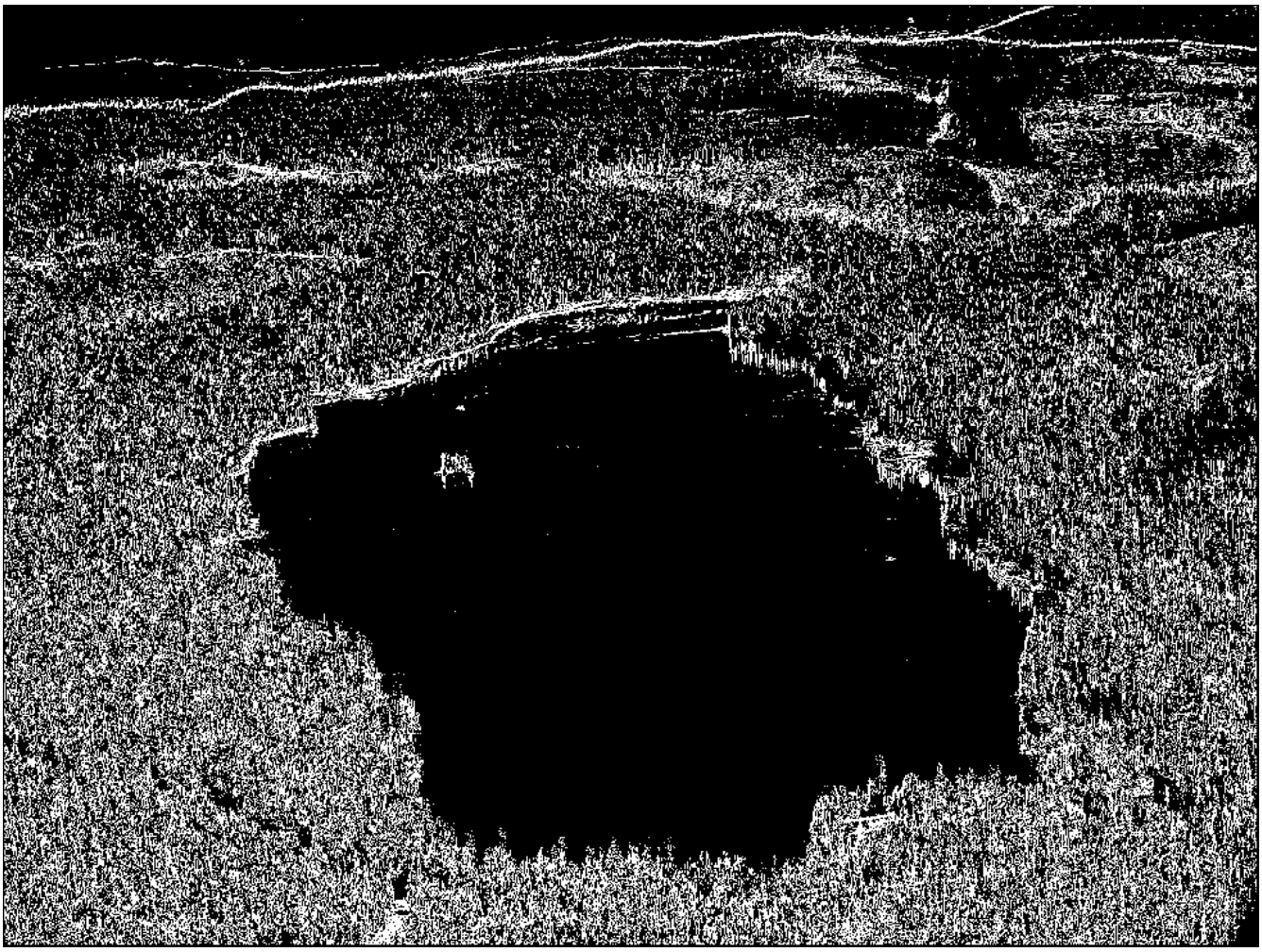

Figure 6: Texture with N=3 for lake.jpg

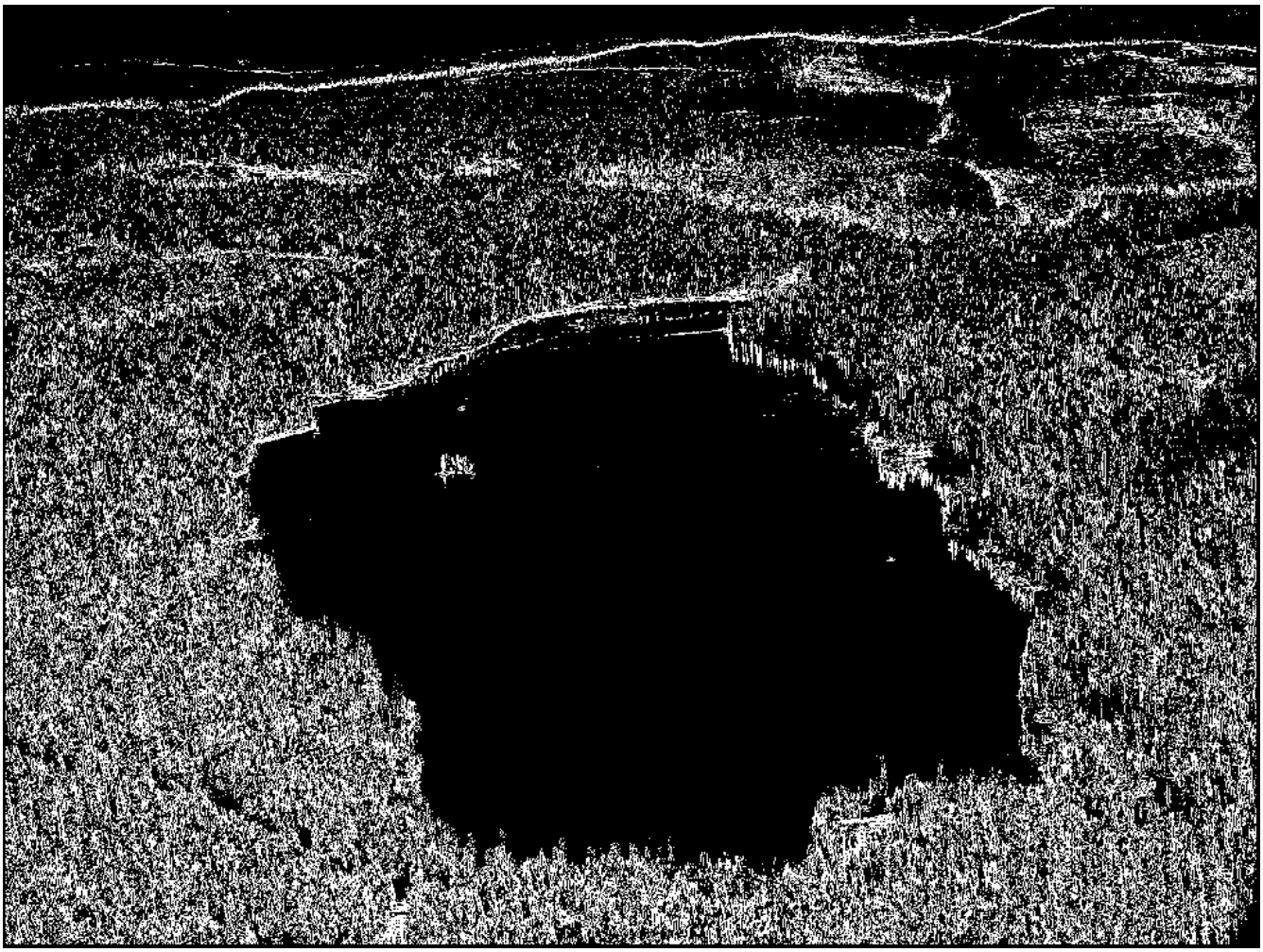

Figure 7: Texture with N=5 for lake.jpg

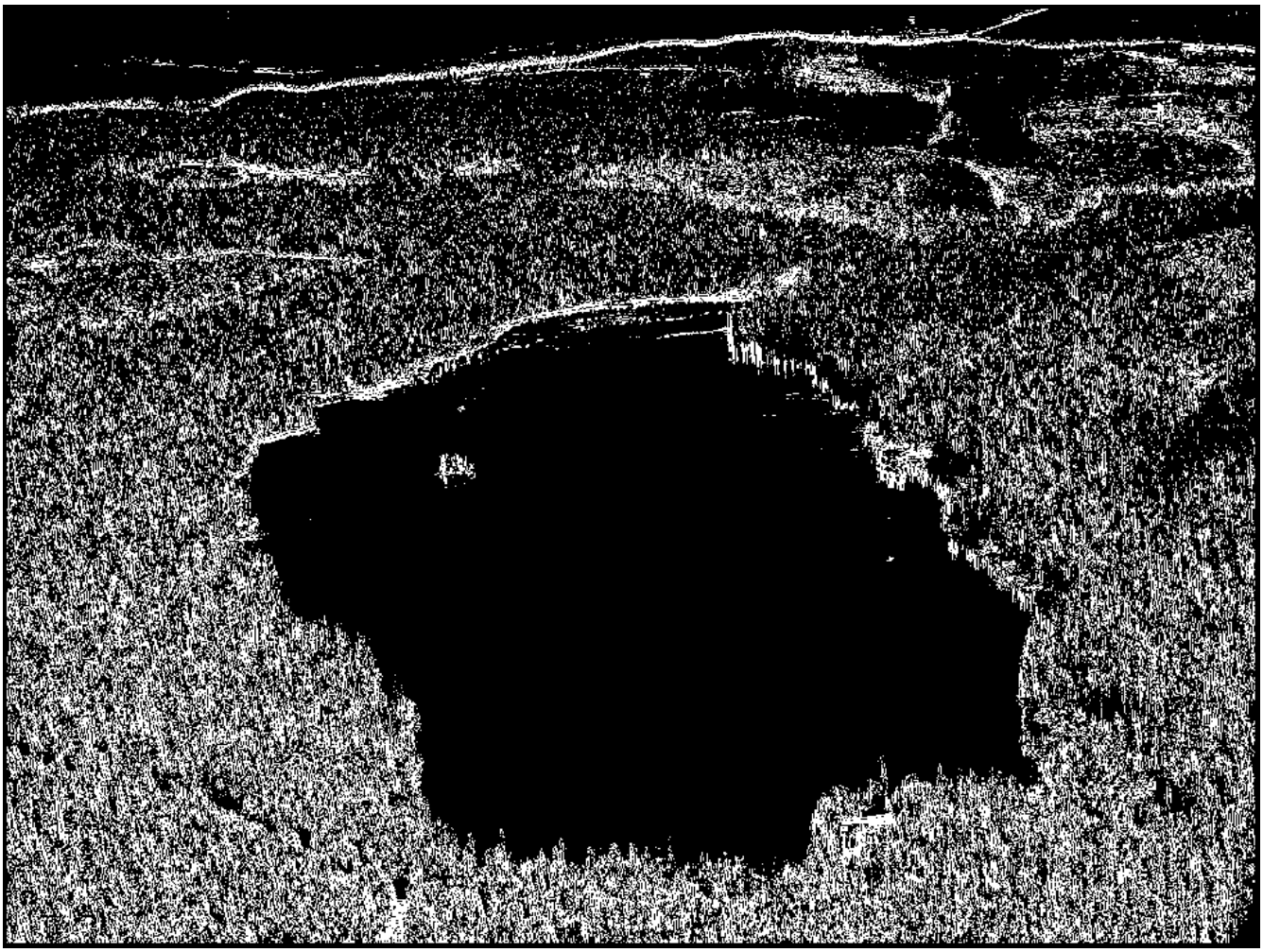

Figure 8: Texture with N=7 for lake.jpg

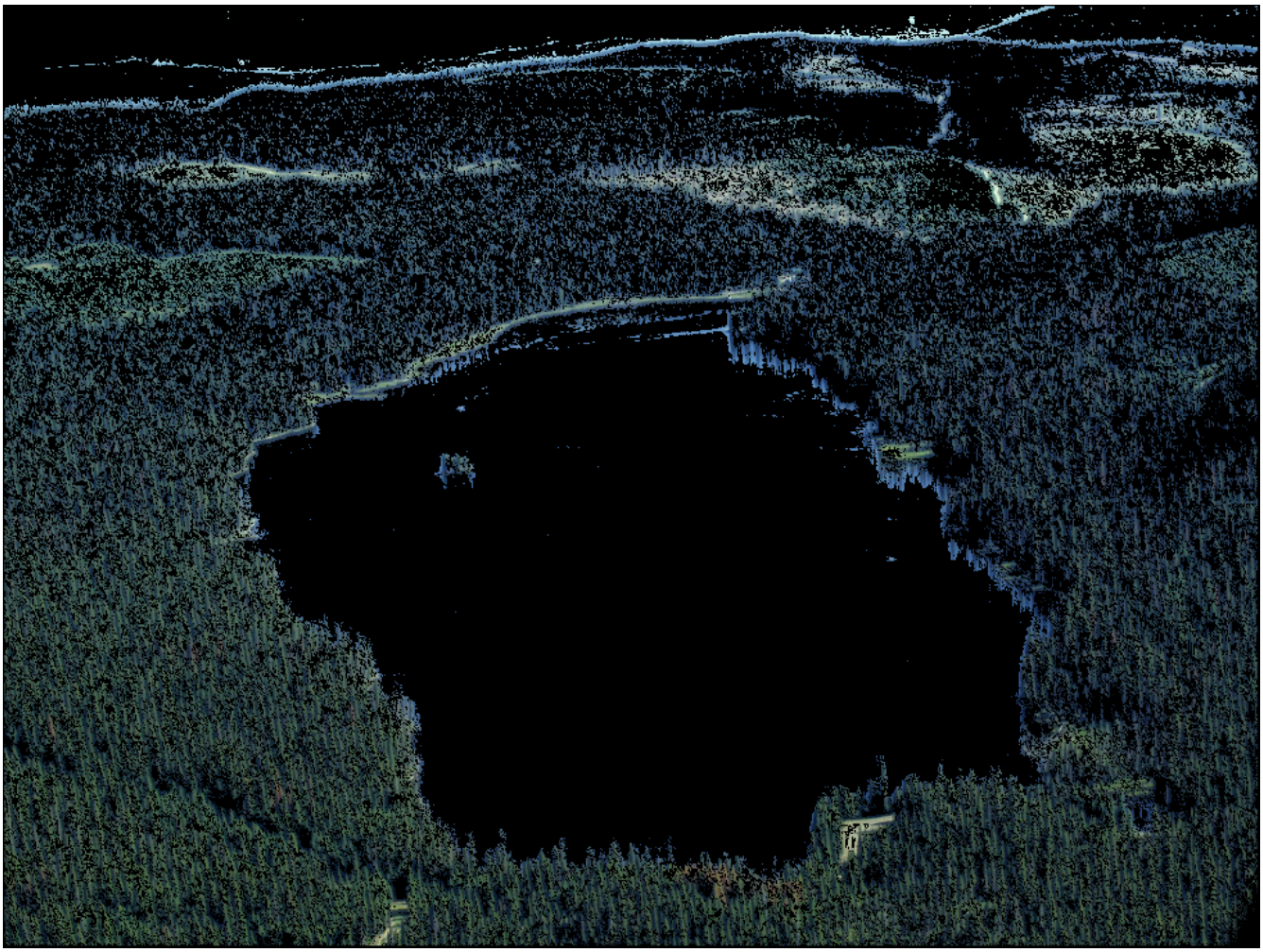

Figure 9: Foreground with texture for lake.jpg

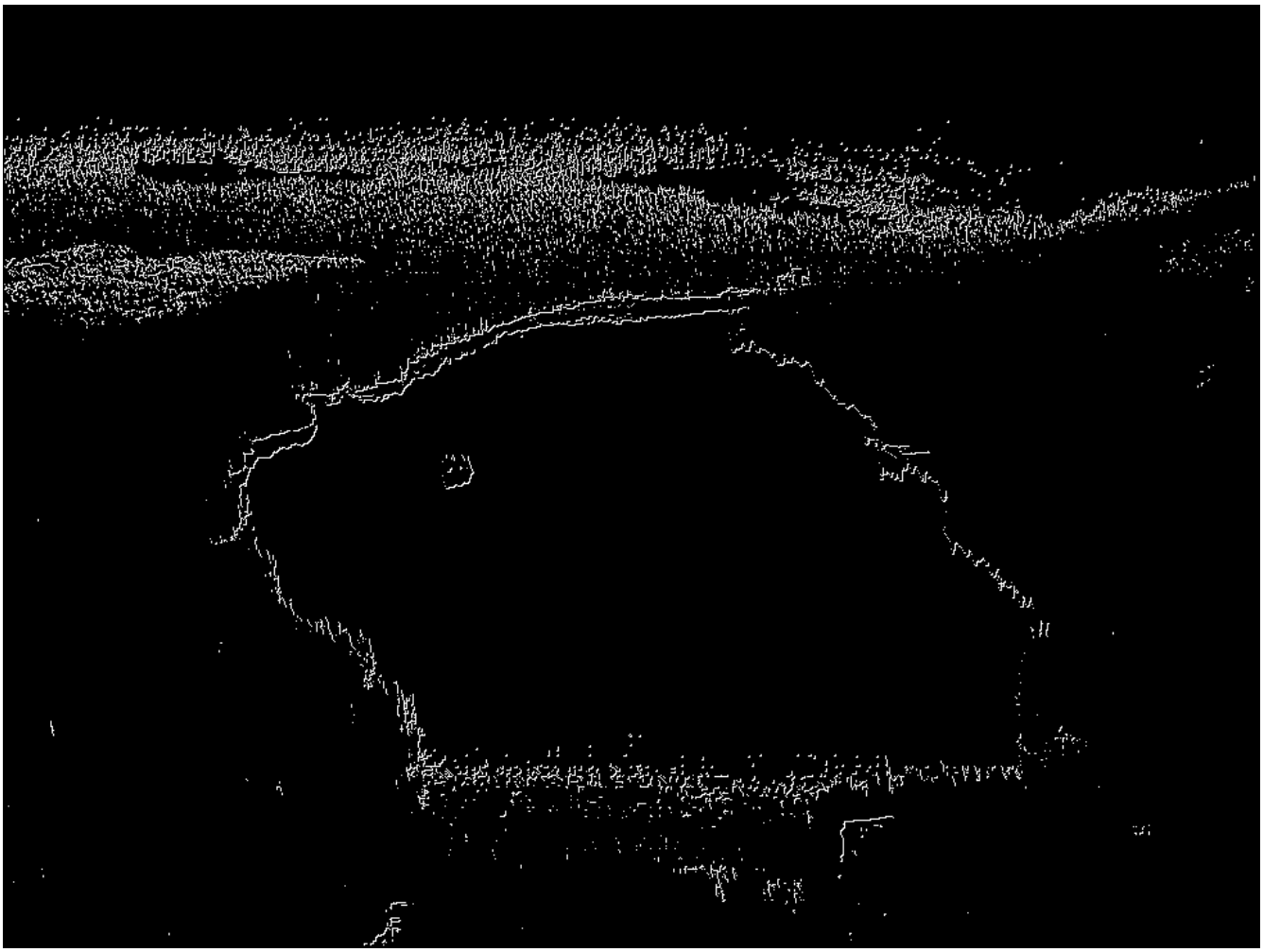

Figure 10: Contour extraction for lake.jpg

5.2 leopard.jpg

Figure 11: Original image of leopard.jpg

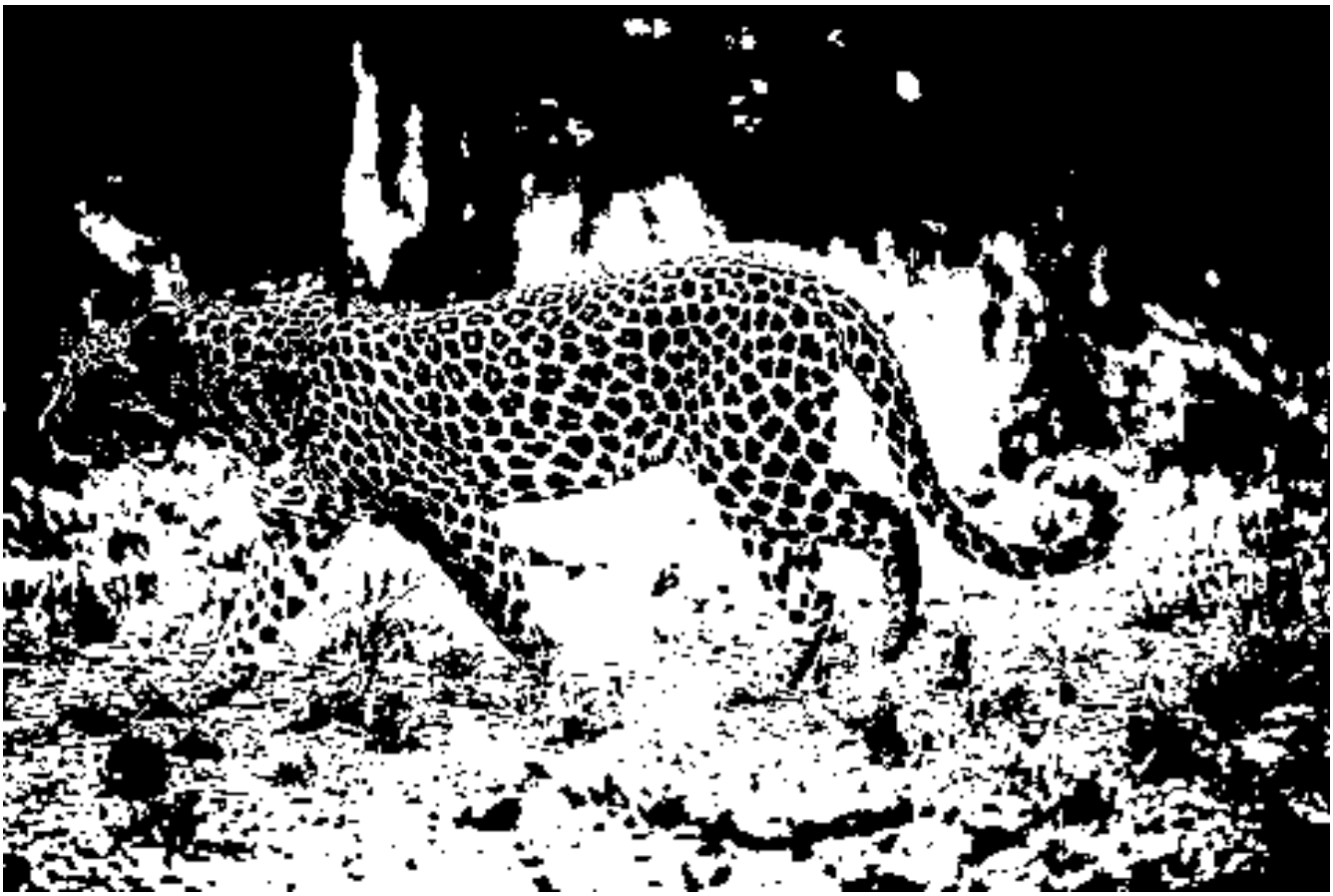

Figure 12: Blue channel mask for leopard.jpg

Figure 13: Green channel mask for leopard.jpg

Figure 14: Red channel mask for leopard.jpg

Figure 15: Foreground with only Otsu algorithm for leopard.jpg

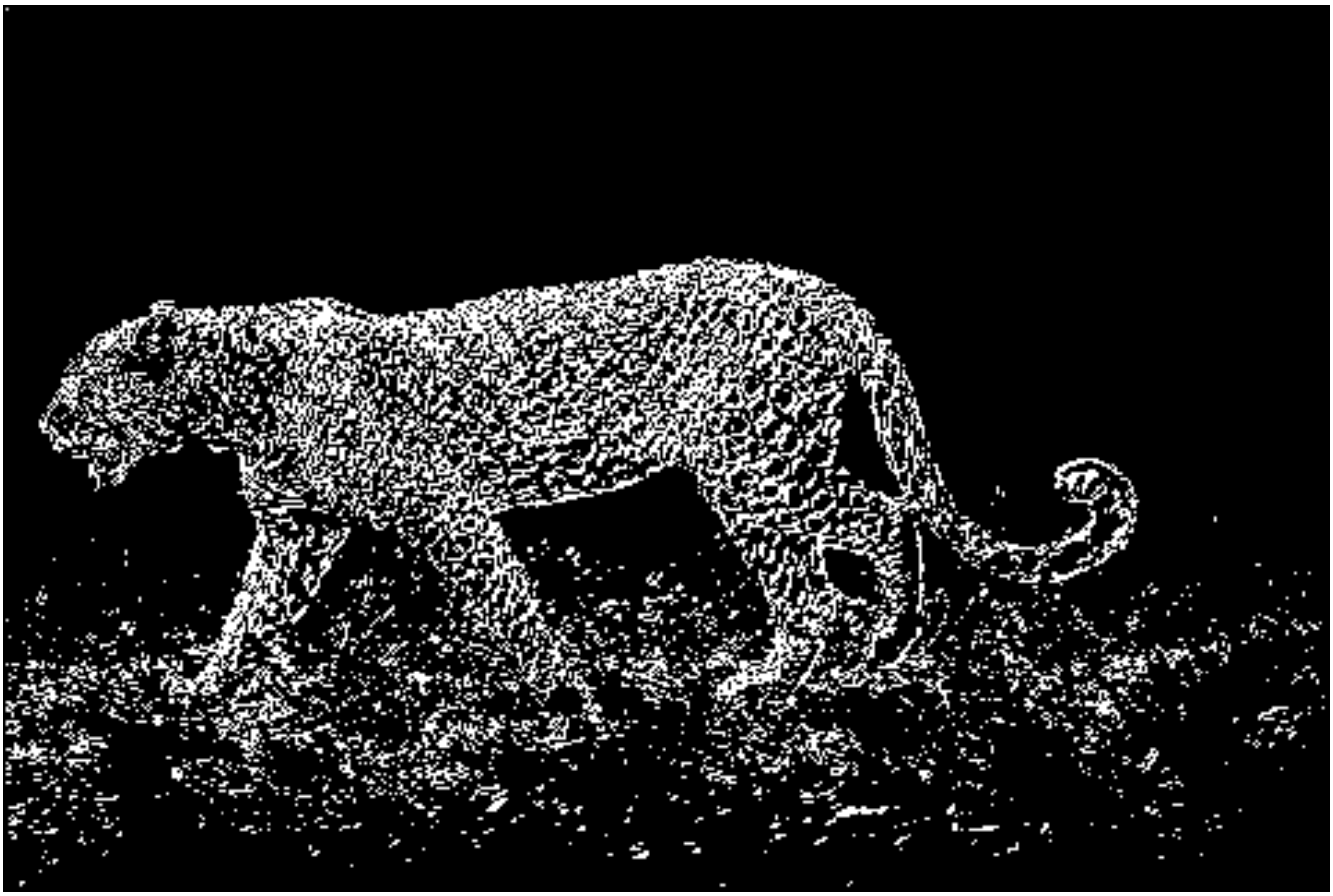

Figure 16: Texture with N=3 for leopard.jpg

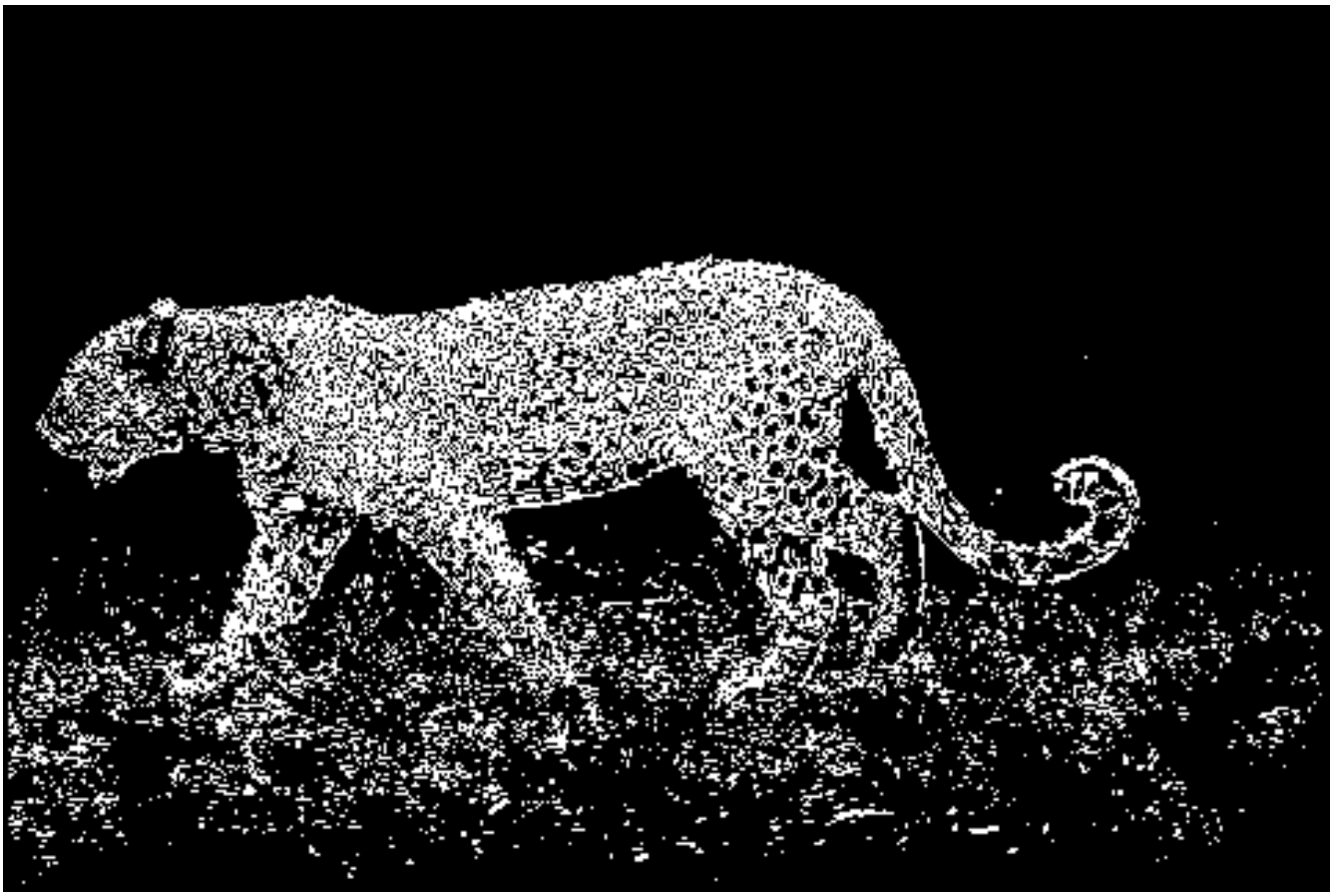

Figure 17: Texture with N=5 for leopard.jpg

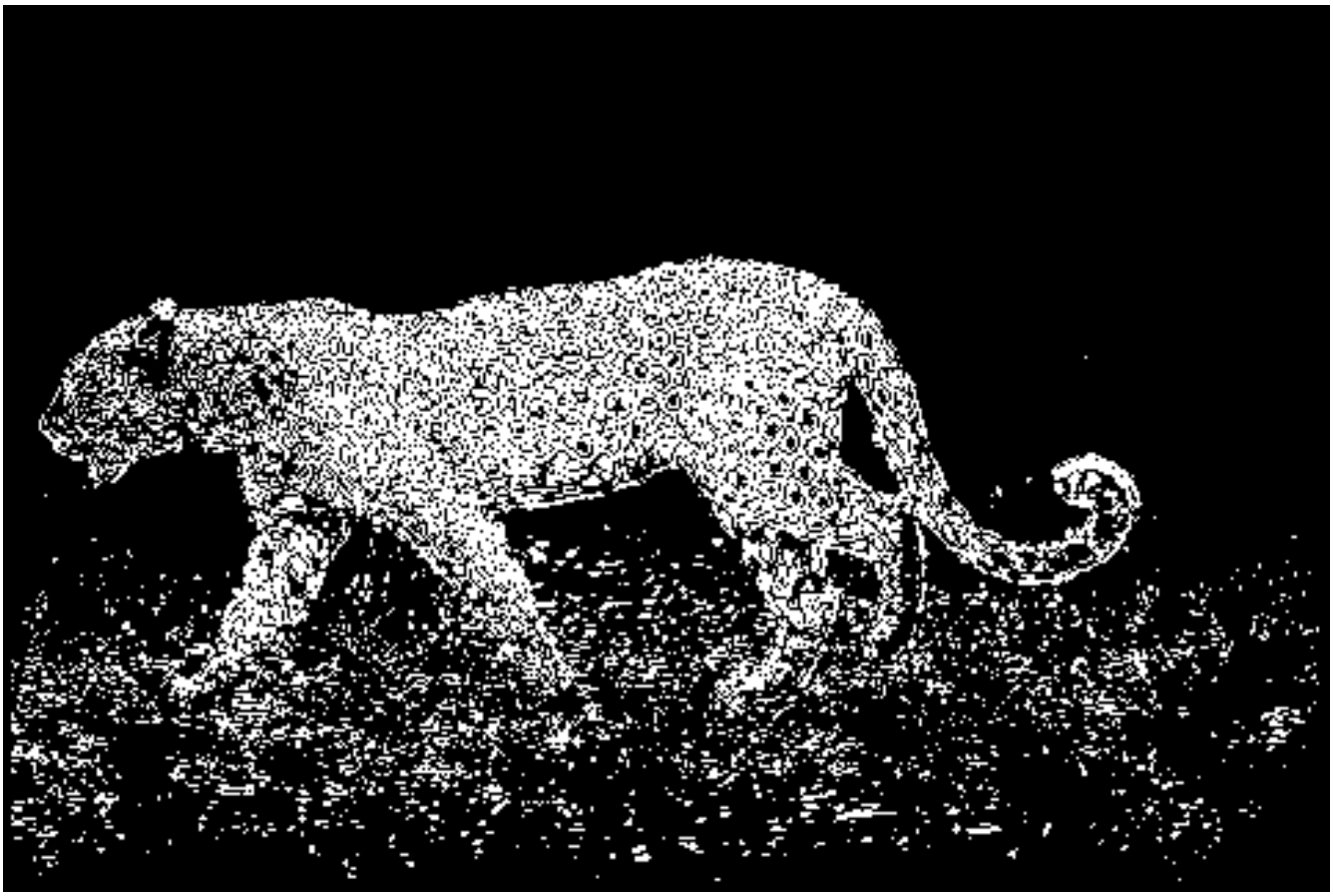

Figure 18: Texture with N=7 for leopard.jpg

Figure 19: Foreground with texture for leopard.jpg

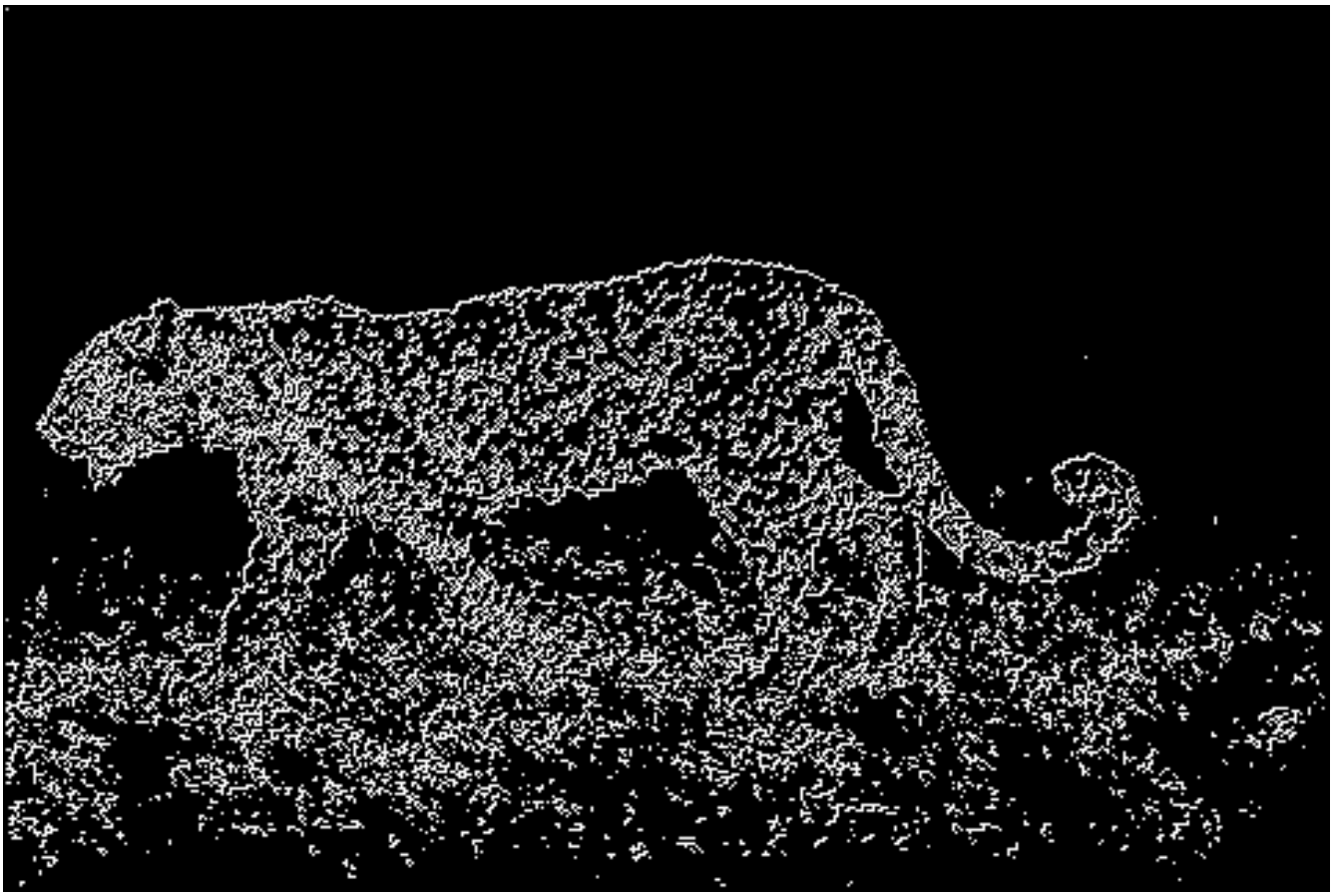

Figure 20: Contour extraction for leopard.jpg

5.3 brain.jpg

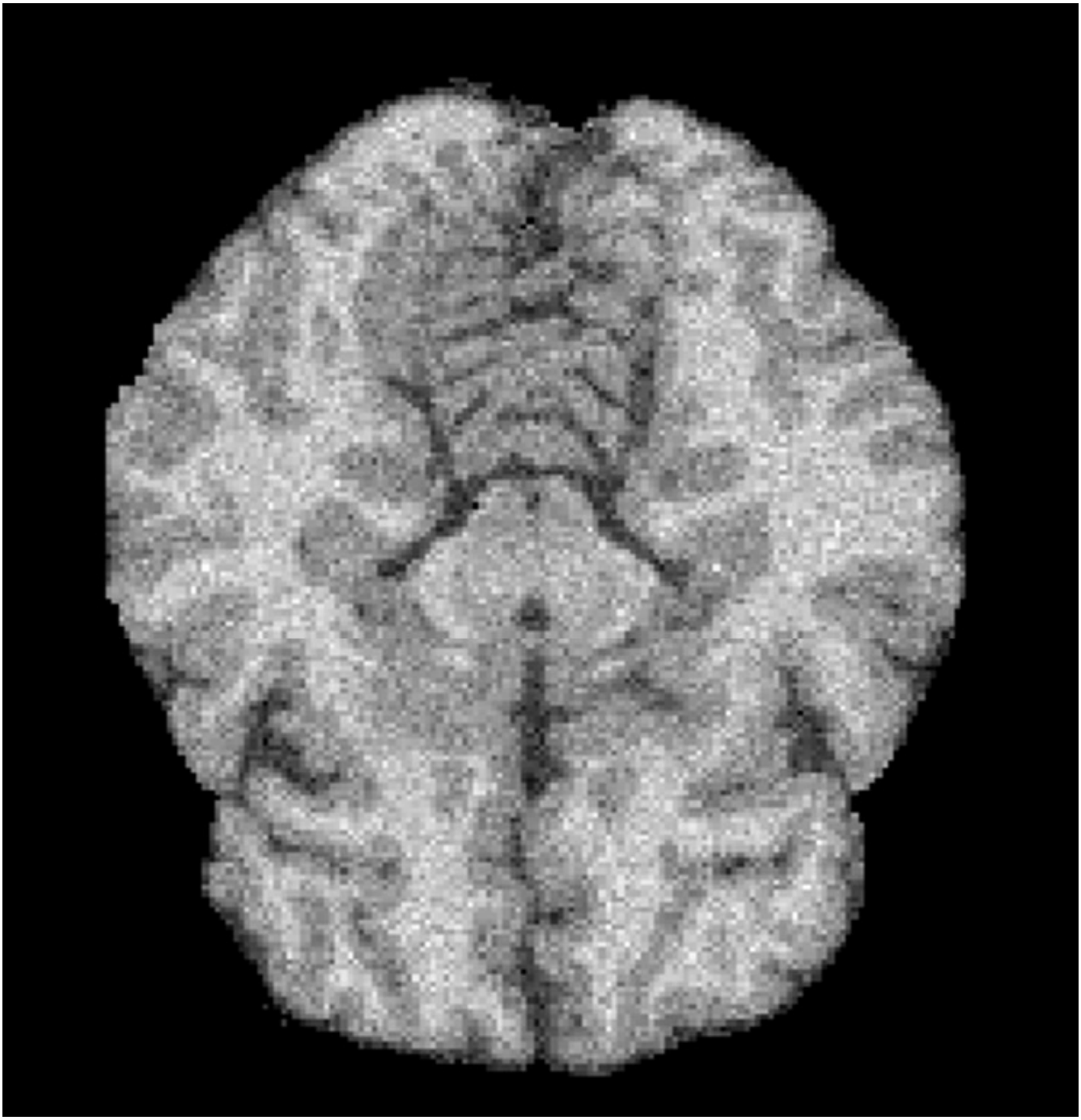

Figure 21: Original image of brain.jpg

Figure 22: Blue channel mask for brain.jpg

Figure 23: Green channel mask for brain.jpg

Figure 24: Red channel mask for brain.jpg

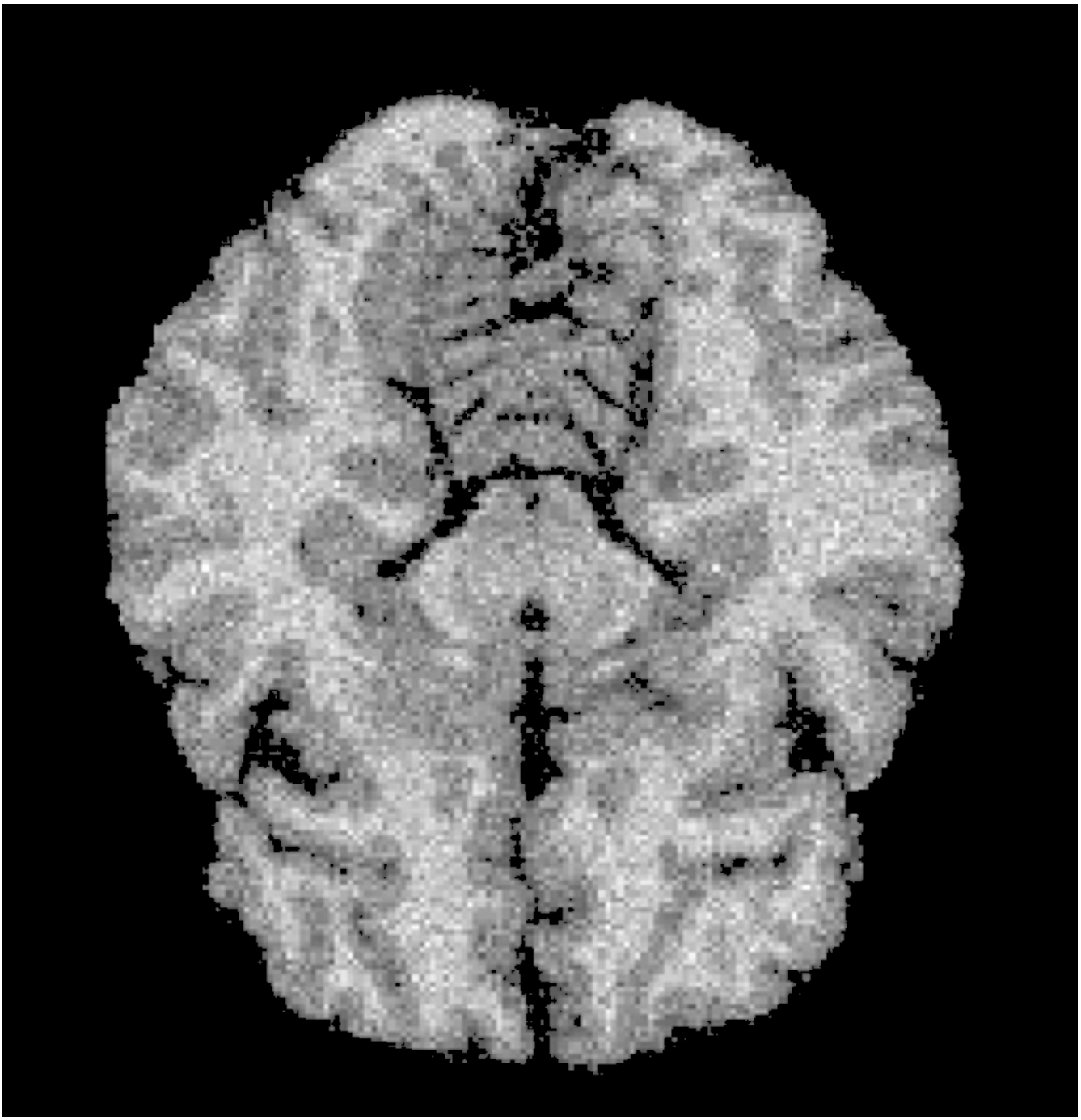

Figure 25: Foreground with only Otsu algorithm for brain.jpg

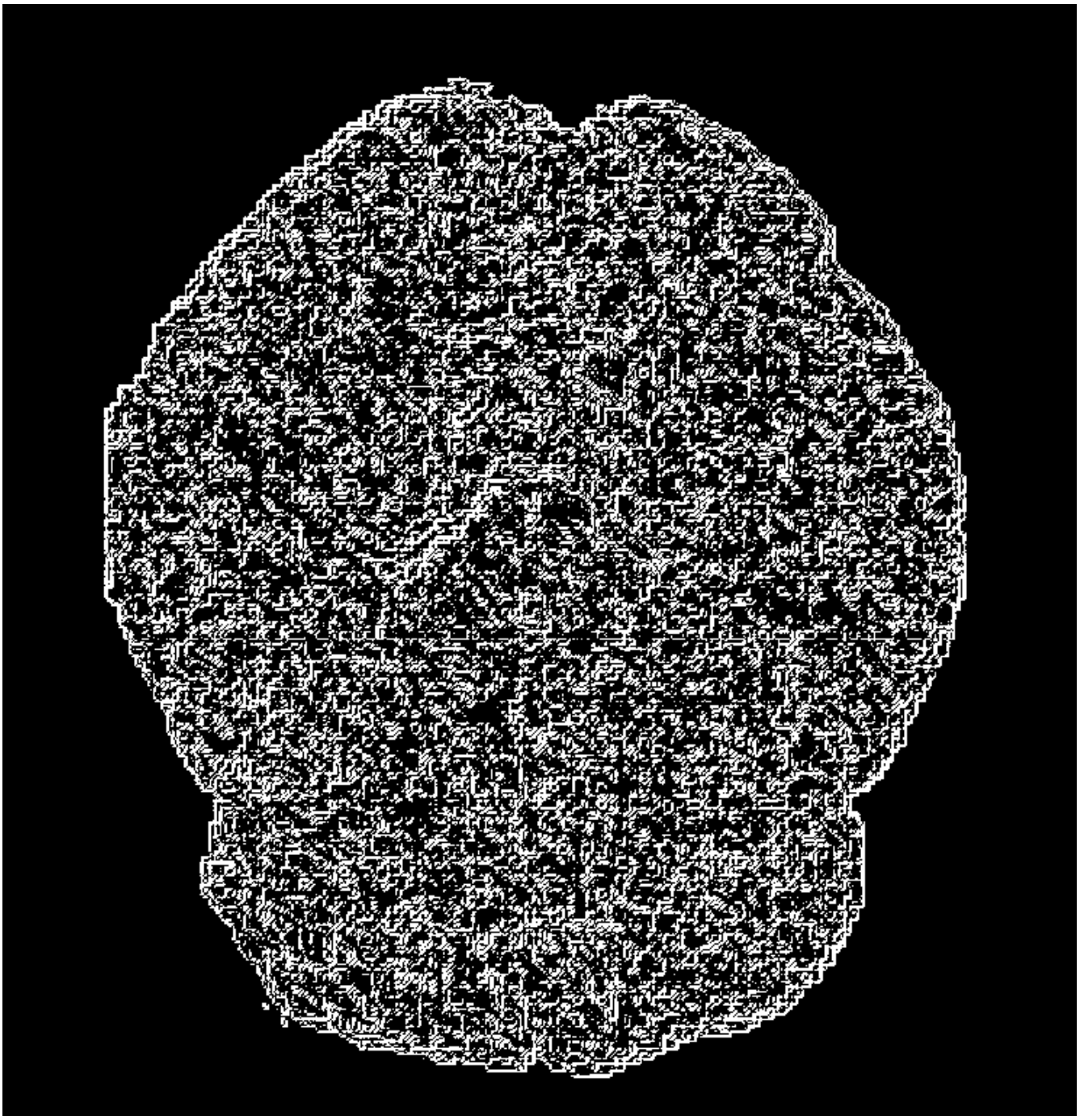

Figure 26: Texture with N=3 for brain.jpg

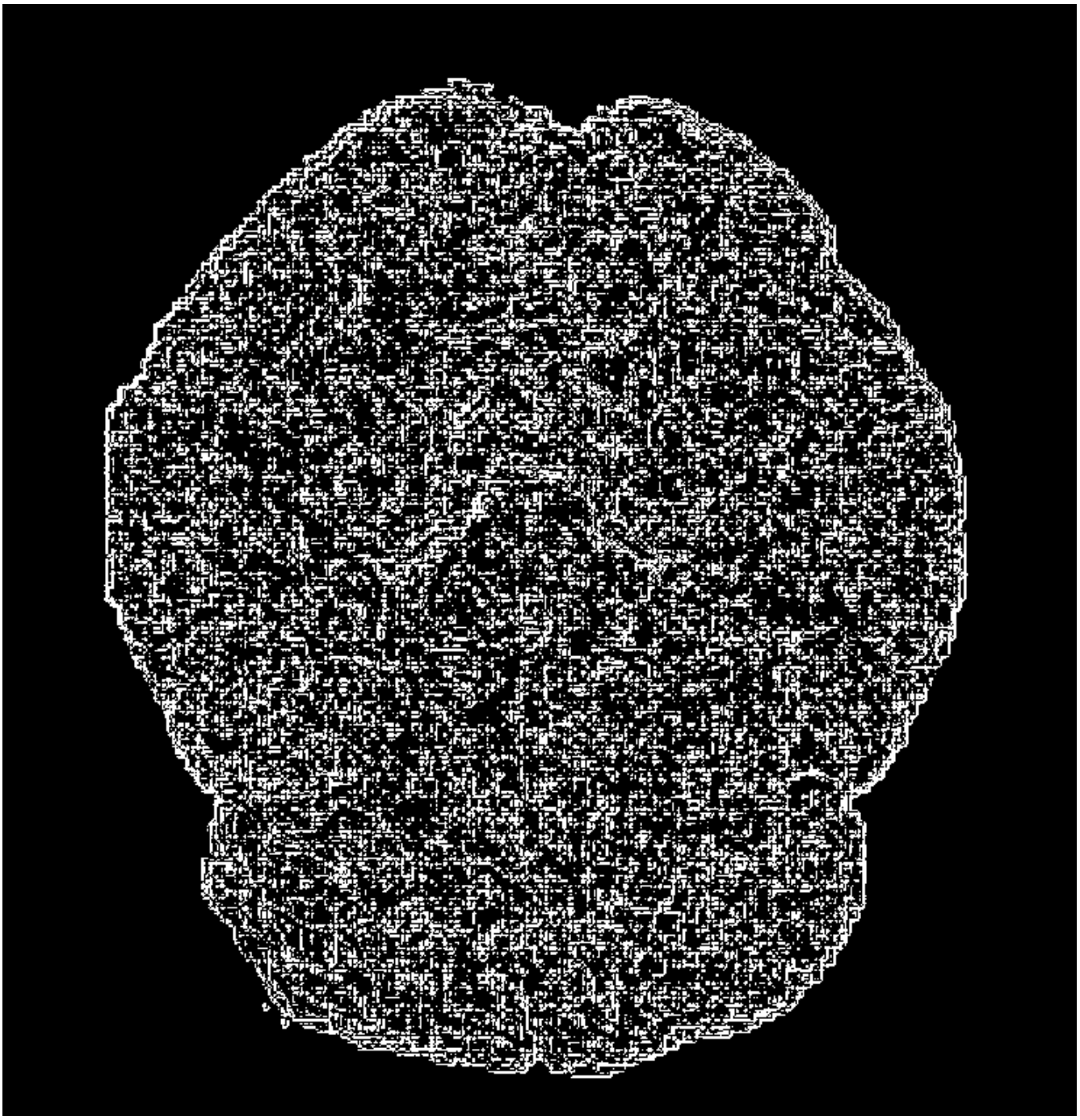

Figure 27: Texture with N=5 for brain.jpg

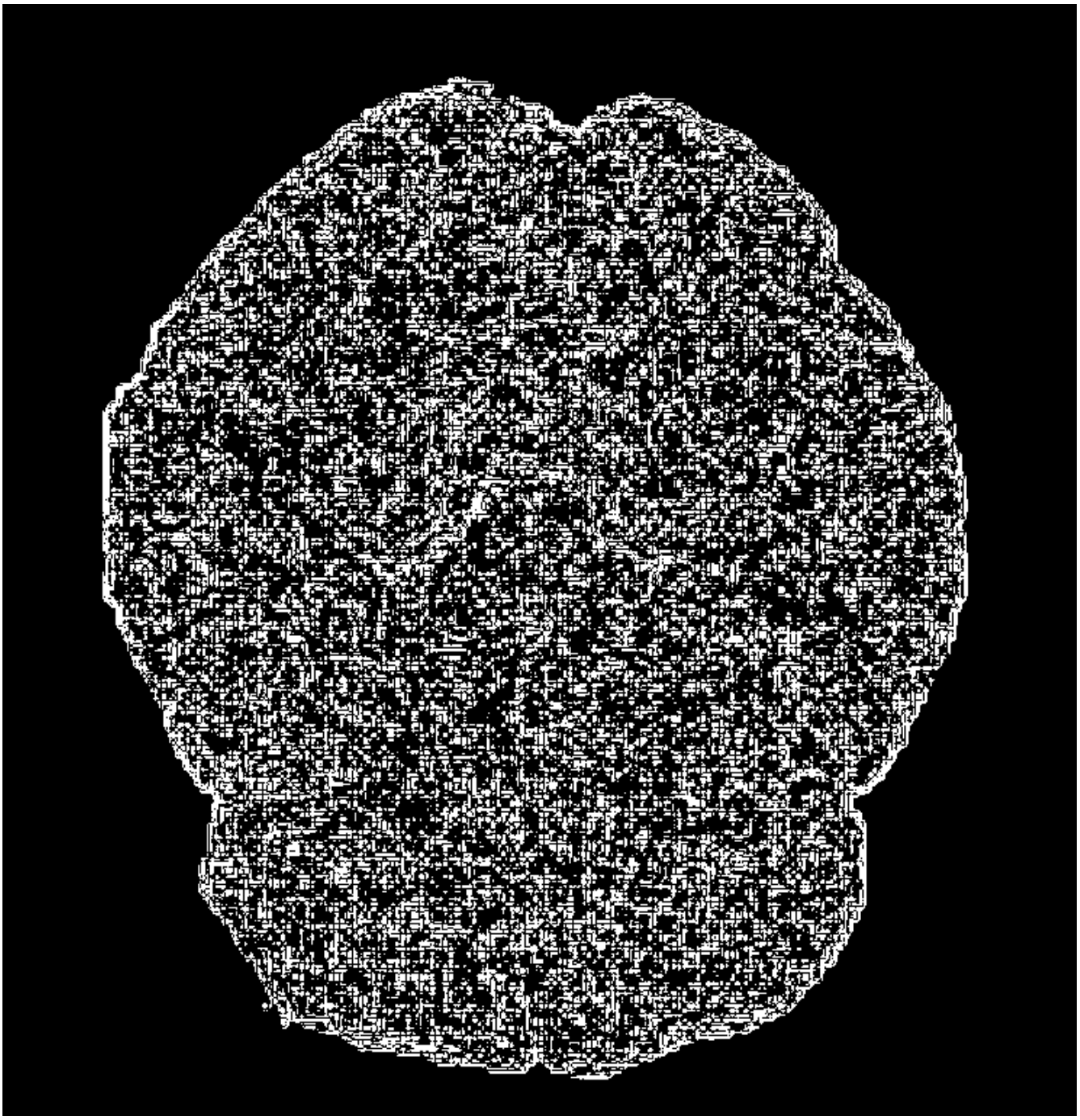

Figure 28: Texture with N=7 for brain.jpg

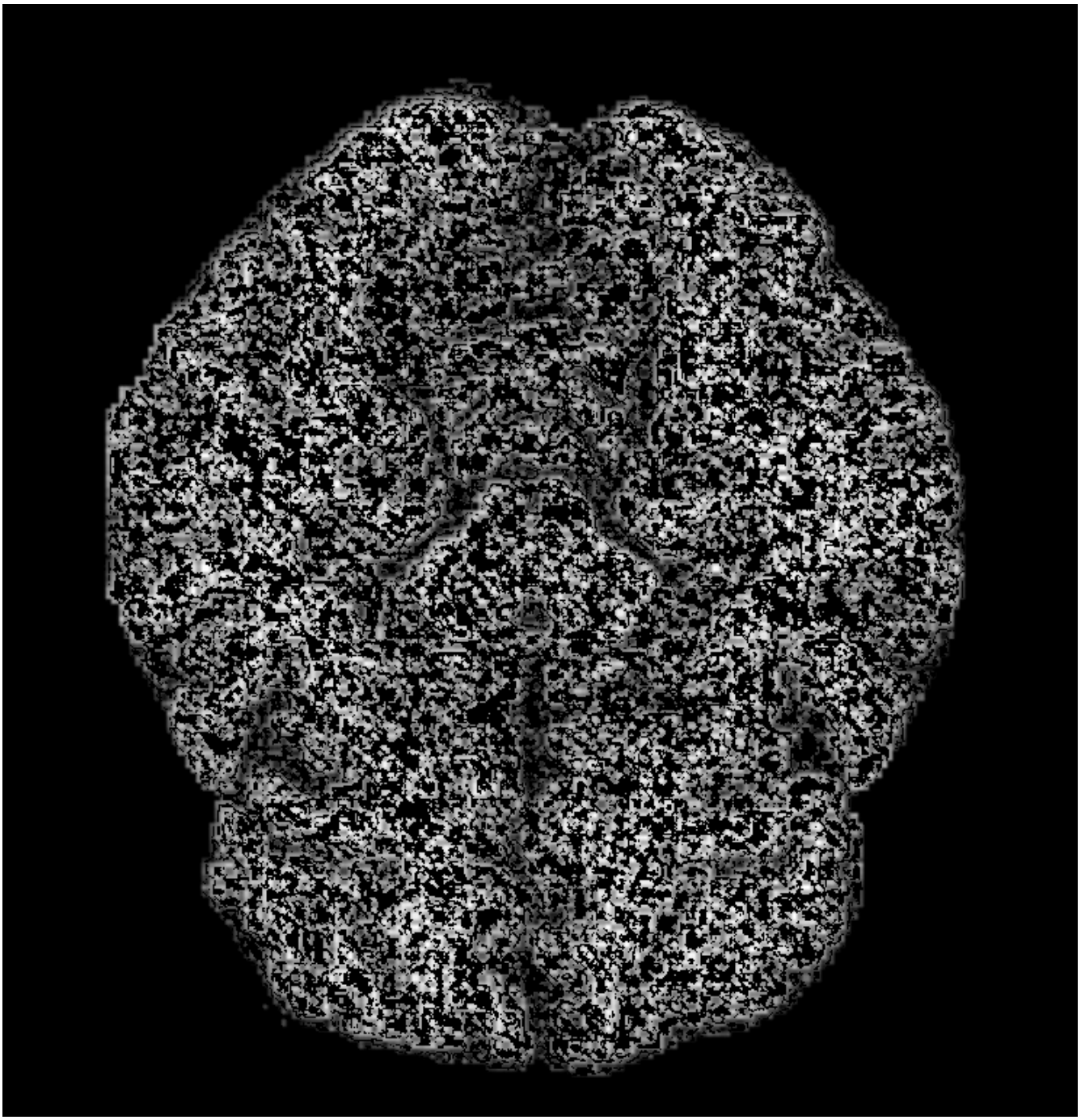

Figure 29: Foreground with texture for brain.jpg

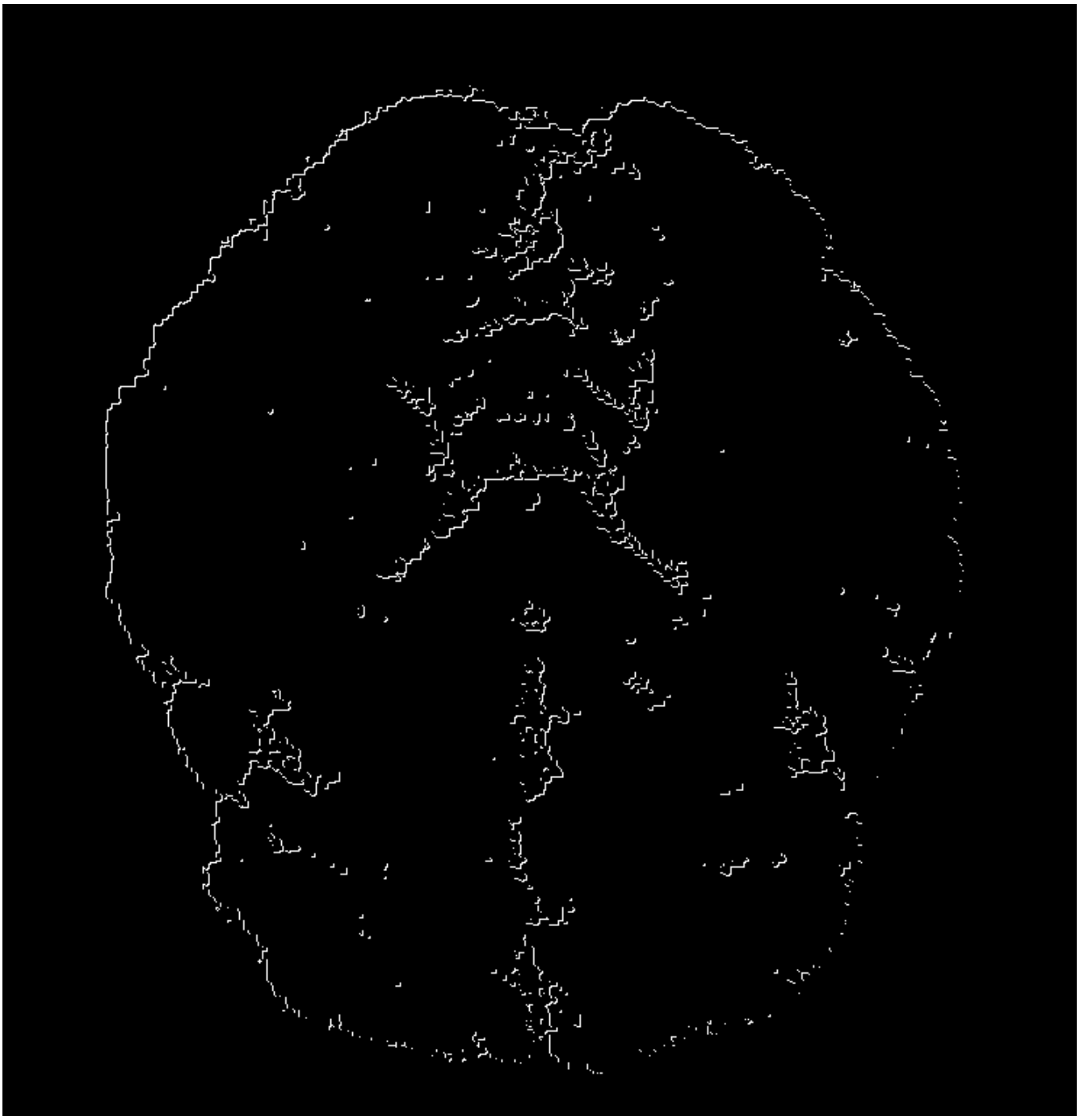

Figure 30: Contour extraction for brain.jpg

6 Source Code

```
import cv2
import numpy as np
import math
```

```
#Sum from n to m terms in array###################################
def sum_nm(s,n,m):
    sum = 0
    for i in range(n,m):
        sum += s[i]
    return sum
#End#################################################################

#Sum from n to m terms in array###################################
def sum_inm(s,n,m):
    sum = 0
    for i in range(n,m):
        sum += s[i]*i
    return sum
#End#################################################################


#Otsu algorithm for 1 channel#####################################################
def otsu_1cha(img,iterations):
    print('Otsu algorithm')
    mask = np.zeros((np.shape(img)[0],np.shape(img)[1]))
    mask.fill(255)

    hist = np.zeros(256,int)

    N = np.shape(img)[0] * np.shape(img)[1]

    imgcopy = np.matrix.copy(img)

    #Iterations
    for i in range(0,iterations):
        #Get the number of pixels in each level
        for y in range(0,np.shape(img)[0]):
            for x in range(0,np.shape(img)[1]):
                hist[int(imgcopy[y,x])] += 1

        threshold = 0
        max_bv = 0


        for k in range(0,256):
            w0 = sum_nm(hist,0,k) / N
            w1 = sum_nm(hist,k,256) / N
            if w0 != 0 and w1 != 0:
                u0 = sum_inm(hist,0,k) / N / w0
                u1 = sum_inm(hist,k,256) / N / w1

                bv = w0*w1*math.pow((u0-u1),2)
                if bv > max_bv:
                    max_bv = bv
                    threshold = k

        #apply mask
        for y in range(0,np.shape(img)[0]):
```

```
            for x in range(0,np.shape(img)[1]):
                if imgcopy[y,x] < threshold:
                    mask[y,x] = 0
    return mask

#End Otsu#############################################################

#Generate final foreground#############################################
def gfore(img,mb,mg,mr):
    result = np.matrix.copy(img)
    for y in range(0,np.shape(img)[0]):
        for x in range(0,np.shape(img)[1]):
            if mb[y,x] == 0 and mg[y,x] == 0 and mr[y,x] == 0:
                result[y,x] = 0

    return result
#End###################################################################


#Get the texture#######################################################
def gettexture(img,N):
    #convert to gray
    gray = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
    result = np.zeros((np.shape(img)[0],np.shape(img)[1]))

    hs = math.floor(N/2)
    for y in range(hs,np.shape(img)[0] - hs - 1):
        for x in range(hs,np.shape(img)[1] - hs - 1):
            window = gray[y-hs:y+hs,x-hs:x+hs]
            result[y,x] = int(math.sqrt(math.pow(gray[y,x] - window.mean(),2)))

    return result
######################################################################

#Contour Extraction####################################################
def contour(img):
    print('Contour extraction')
    result = np.zeros((np.shape(img)[0],np.shape(img)[1]))
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    hs = math.floor(3/2)
    for y in range(hs,np.shape(img)[0] - hs - 1):
        for x in range(hs,np.shape(img)[1] - hs - 1):
            if gray[y,x] != 0:
                window = gray[y-hs:y+hs,x-hs:x+hs]
                if (0 in window):
                    result[y,x] = (255)
    return result
#End###################################################################

img1 = cv2.imread('lake.jpg')
img2 = cv2.imread('leopard.jpg')
img3 = cv2.imread('brain.jpg')
```

```
#Lake.jpg#########################################
print('Start processing image Lake.jpg')
b,g,r = cv2.split(img1)
maskb = otsu_1cha(b,4)
cv2.imwrite('Lake_fB1.png',maskb)
maskg = otsu_1cha(g,4)
cv2.imwrite('Lake_fG1.png',maskg)
maskr = otsu_1cha(r,4)
cv2.imwrite('Lake_fR1.png',maskr)
foreground = gfore(img1,maskb,maskg,maskr)
cv2.imwrite('Lake_otsu.png',foreground)
cont = contour(foreground)
cv2.imwrite('Lake_cont.png',cont)

texture3 = gettexture(img1,3)
mask3 = otsu_1cha(texture3,4)
cv2.imwrite('Lake_3N.png',mask3)
texture5 = gettexture(img1,5)
mask5 = otsu_1cha(texture5,4)
cv2.imwrite('Lake_5N.png',mask5)
texture7 = gettexture(img1,7)
mask7 = otsu_1cha(texture7,4)
cv2.imwrite('Lake_7N.png',mask7)
foreground = gfore(img1,mask3,mask5,mask7)
cv2.imwrite('Lake_texture.png',foreground)

#Leopard.jpg#######################################
print('Start processing image Leopard.jpg')
b,g,r = cv2.split(img2)
maskb = otsu_1cha(b,4)
cv2.imwrite('Leo_fB1.png',maskb)
maskg = otsu_1cha(g,4)
cv2.imwrite('Leo_fG1.png',maskg)
maskr = otsu_1cha(r,4)
cv2.imwrite('Leo_fR1.png',maskr)
foreground = gfore(img2,maskb,maskg,maskr)
cv2.imwrite('Leo_otsu.png',foreground)

texture3 = gettexture(img2,3)
mask3 = otsu_1cha(texture3,4)
cv2.imwrite('Leo_3N.png',mask3)
texture5 = gettexture(img2,5)
mask5 = otsu_1cha(texture5,4)
cv2.imwrite('Leo_5N.png',mask5)
texture7 = gettexture(img2,7)
mask7 = otsu_1cha(texture7,4)
cv2.imwrite('Leo_7N.png',mask7)
foreground = gfore(img2,mask3,mask5,mask7)
cv2.imwrite('Leo_texture.png',foreground)

cont = contour(foreground)
cv2.imwrite('Leo_cont.png',cont)
#Brain.jpp#########################################
```

```
print('Start processing image Braind.jpg')
b,g,r = cv2.split(img3)
maskb = otsu_1cha(b,4)
cv2.imwrite('Brain_fB1.png',maskb)
maskg = otsu_1cha(g,4)
cv2.imwrite('Brain_fG1.png',maskg)
maskr = otsu_1cha(r,4)
cv2.imwrite('Brain_fR1.png',maskr)
foreground = gfore(img3,maskb,maskg,maskr)
cv2.imwrite('Brain_otsu.png',foreground)
cont = contour(foreground)
cv2.imwrite('Brain_cont.png',cont)

texture3 = gettexture(img3,3)
mask3 = otsu_1cha(texture3,4)
cv2.imwrite('Brain_3N.png',mask3)
texture5 = gettexture(img3,5)
mask5 = otsu_1cha(texture5,4)
cv2.imwrite('Brain_5N.png',mask5)
texture7 = gettexture(img3,7)
mask7 = otsu_1cha(texture7,4)
cv2.imwrite('Brain_7N.png',mask7)
foreground = gfore(img3,mask3,mask5,mask7)
cv2.imwrite('Brain_texture.png',foreground)
```