EE 482 Computational Neuroscience

Homework 2

Cemal Güven Adal

21703986

Name: Cemal Güven Adal
ID: 21703986

Question 1

For question 1 the responses of LGN for visual images is given. Images that are represented in stim and count contains number of spikes in 15.6 time stamp. The file c2p3.mat is downloaded to access the data which is given in the homework. Python is used as the programming language and jupyter notebook is used ide for this homework

a) For part a STA is asked to be done for 10 time steps before each spike. Before this process I have used the c2p3.mat file and I have separated the data using headers in order to access the count and stim images from different arrays. Following code block shows importing necessary python libraries, loading c2p3.mat data and separating it to arrays:

```
#importing necessary libraries to python
import matplotlib.pyplot as plt
import numpy as np
import scipy.io as sio
#loading .mat file to python
data1 = sio.loadmat('c2p3.mat')
data1.keys()
#breaking down stim and counts from.mat file and checking their sizes
stim= data1['stim']
counts = data1['counts']
print(np.shape(stim))
print(np.shape(counts))
#
```

```
dict_keys(['__header__', '__version__', '__globals__', 'counts', 'stim'])
```

```
(16, 16, 32767)
(32767, 1)
```

$$x = \frac{1}{n} * \sum_{i=1}^{n} s_i * count_i \qquad (1)$$

As it can be seen in the formula 1 for calculating STA we need to sum up stim images. Then I have defined an STA function. Each image has its spike triggered average value. 10 time steps is chosen in the homework as the time interval to use before each spike. Count of spikes is multiplied with it's corresponding stim images. As it can be seen in the formula 1 after the multiplying process result is needed to be divided by the number of spikes in the time stamp. To write this function in python in the function STA I have taken the counts array which is the array that holds the spike counts for corresponding stim image and stim array which is the array that holds the 16x16 images. Because summing the 16x16 images wont change the resulting image size and there will be 10 images at the end I have defined an third dimensional average array with the 16x16x10 size. After defining the average array the last 10 stim images before each spike is multiplied with the spike count. This is done for each spike count. Than average array is divided by the sum of all counts in order to achieve the result of STA. Code can be seen below:

```
#STA function
def STA(counts,stim):
    average=np.zeros((16,16,10))
    for i in range (10,counts.shape[0]):
        average=average+stim[:,:,i-10:i]*counts[i,:]
    average=average/(np.sum(counts))
    return average
```

After defining the STA function I have used the given stim and given data in order to create the aver function which contains 10 16x16 images. Then I have printed the 10 images in a row using a for loop and imshow function. Imshow is used because code is written in python and not in matlab and it is the corresponding command of python to imagesc command. In the following code shape of the aver is checked and all 10 images printed with gray scale color map. Minimum and maximum of aver is given to control the grey scale. Each step before the spike is shown. Code can be seen below:

```
aver=STA(counts,stim)
print(np.size(aver))
for i in range(0,10):

    plt.figure()
    plt.imshow(aver[:,:, i], vmin=np.min(aver), vmax=np.max(aver), cmap='gray')
    plt.colorbar()
    plt.title(str(10-i)+'Steps Before Spike')
```

Each of the resulting figures can be seen below:

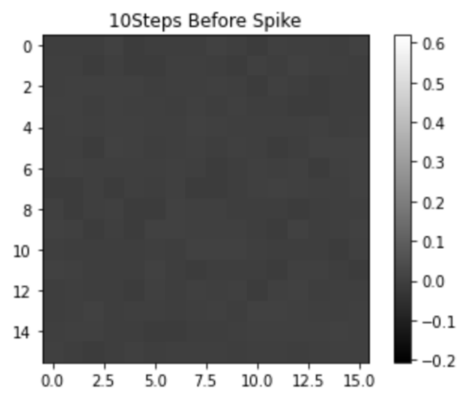

Fig.1:10 steps before the spike

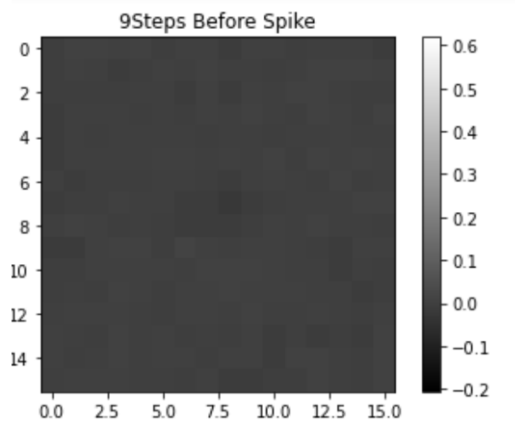

Fig.2: 9 steps before the spike

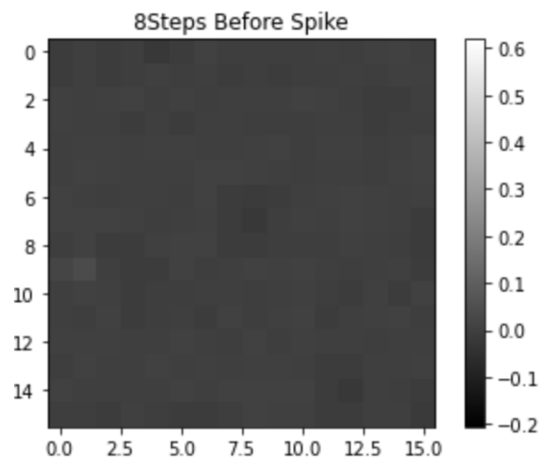

Fig.3: 8 steps before the spike

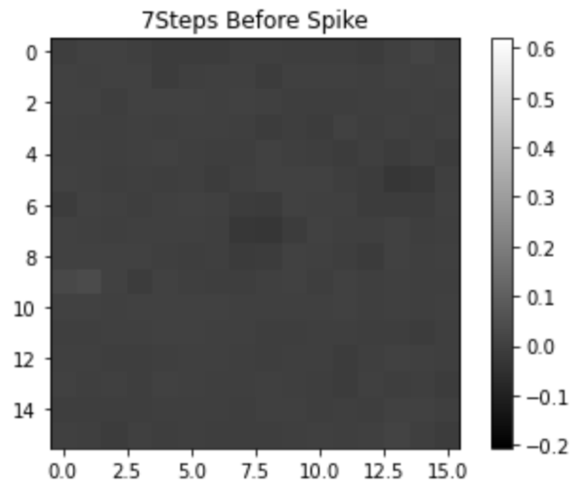

Fig.4: 7 steps before the spike

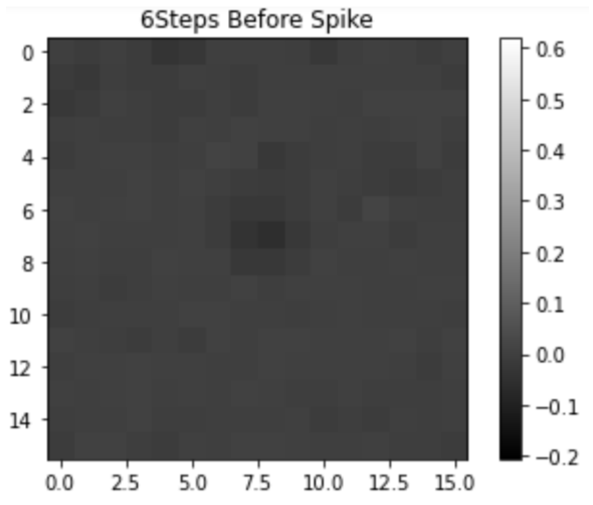

Fig.5: 6 steps before the spike

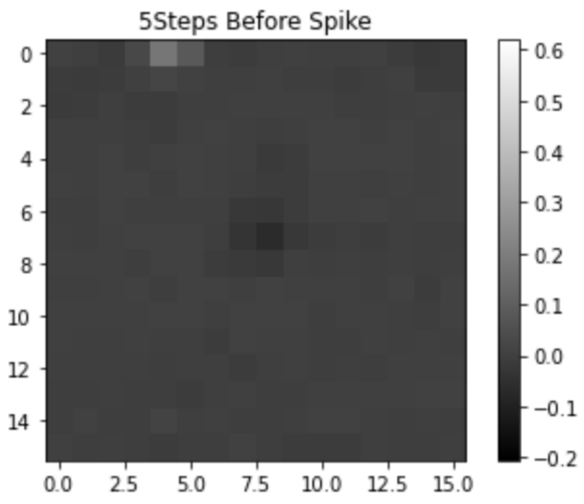

Fig.6: 5 steps before the spike

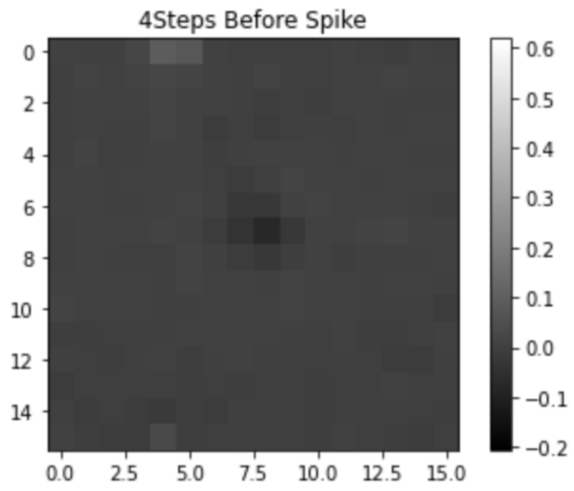

Fig.7: 4 steps before the spike

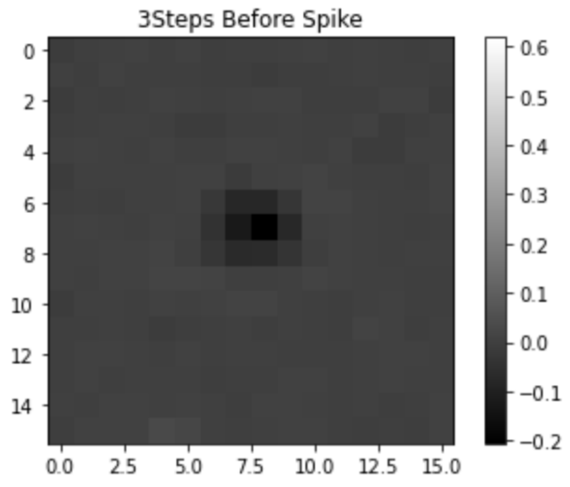

Fig.8: 3 steps before the spike

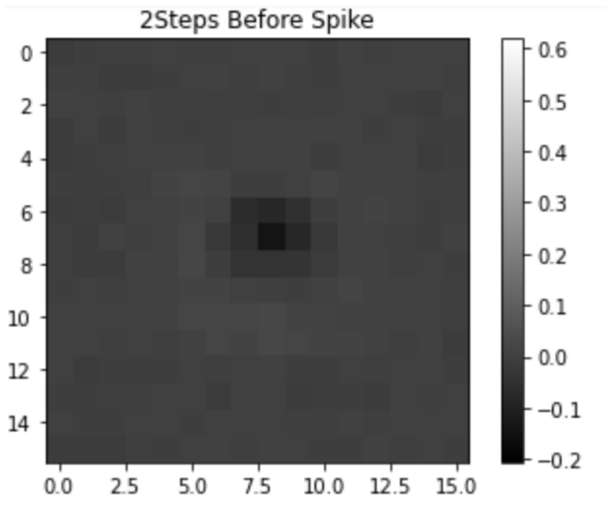

Fig.9: 2 steps before the spike

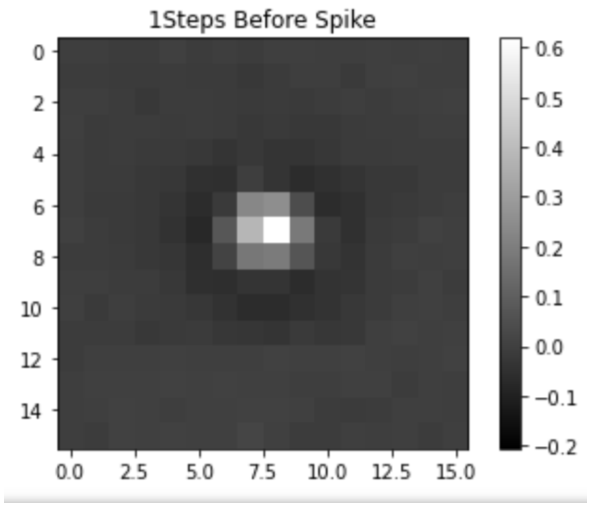

Fig.10: 1 steps before the spike

We can conclude from the images from STA derived filter that LGN cell is not selective for direction. It can be seen that as we approach to the spike middle of the images turn brighter. Thus we can conclude that spike fires when whiteness appears after blackness. Thus LGN cell id selective for luminous intensity.

b) As time moves forward to the firing of the spike STA images changes. Intervals between 5 to 12.5 in the columns and intervals between 5 to 11 in rows are the areas which the change could be seen most visibly. As the firing of the spike approaches these areas gets brighter and brighter and middle turns most bright just 1 step before the spike. In this part it has been asked to sum all the STA images over dimensions. Code for this can be seen below for summing row dimensions and column:

```
#partb
plt.figure()
averagerow=np.sum(aver,axis=0)
plt.title("Row Summed Averages")
plt.imshow(averagerow, cmap='gray')
plt.figure()
averagecol=np.sum(aver,axis=1)
plt.title("Coloumn Summed Averages")
plt.imshow(averagerow, cmap='gray')
```

Resulting images can be seen below:

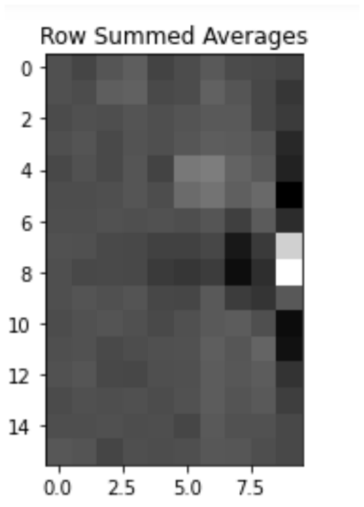

Fig.11: Row Summed STA

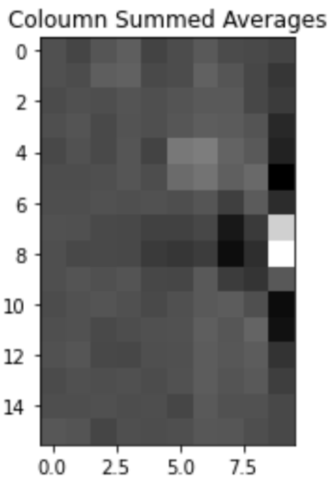

Fig.11: Column Summed STA

STA images summed over both row and column dimensions. Matrix is space time separable because RF profile doesn't change over time. Only the brightness change as it can be seen in the each figure in part a but general profile stays the same because relative position of each region stays the same. Thus because it can be described as a product of two function matrix is space time separable. Space time separable cells are not direction selective. They respond same to either direction change in the stimuli.

c) In part c it is been asked that finding the projection of each sample by using Frobenius inner product of STA image and stimulus image. Frobenius inner product is:

$$(A, B)_F = A_{11}B_{11} + A_{12}B_{12} \dots \dots + A_{nm}B_{nm} \quad (2)$$

In the code below I have taken the Frobenius inner product of the stimulus image and STA image:

```
#PARTC
#for all stimlus
aver1=aver[:,:,-1]#for takng the last one
hist1=np.zeros(32767)
for i in range(32767):
    hist1[i] = np.sum(aver1*stim[:,:,i])
plt.figure(figsize=(8,10))
plt.hist(hist1,density='True', bins=90,  rwidth=0.70)
plt.title('Stimulus Projection for all Spikes')
plt.show()
#for stimulus non 0's
hist2=np.array([])

for i in range(32767):
    if counts[i,:]!=0:
        sumvalue=np.sum(aver1*stim[:,:,i])
        hist2=np.append(hist2,sumvalue)
plt.figure(figsize=(8,10))
plt.hist(hist2,density='True', bins=90, rwidth=0.70)
plt.title('Stimulus Projection for non zero Spikes')
plt.show()
```

Histogram for all and just non zero spikes can be seen below:

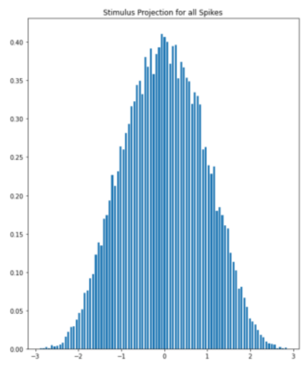

Fig.12: Stimulus Projection Histogram for all spikes

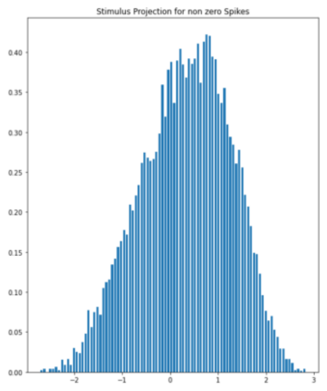

Fig.13: Stimulus Projection Histogram for non zero spikes

Name: Cemal Güven Adal
ID: 21703986

From observing both histograms we can conclude that all projections are gaussians and non zero stimulus resembles stimulus. From all projections most times it takes value 0 or close to 0 thus it has a mean close to 0. When stimulus projected when only spike count is not zero mean increases. Non linear function of the graph can be found from the bayes rule. Non linear function of the graph is the resulting PMF of bayes rule:

$$P(spike|\ \text{stimulus}) = \frac{\text{P(spike,stimulus)}}{\text{P(stimulus)}} \qquad (3)$$

STA somewhat discriminates the spike eliciting stimuli yet the difference between histograms could be more to discriminate better. Thus STA discriminate spike eliciting stimuli but it is not significantly.

Question 2

a) In the question its been asked to construct a difference of gaussians centred surround receptive field. Formula is given below:

$$D(x,y) = \frac{1}{2\pi\sigma^2{}_c} e^{-(\frac{x^2+y^2}{2\sigma^2{}_c})} -= \frac{1}{2\pi\sigma^2{}_s} e^{-(\frac{x^2+y^2}{2\sigma^2{}_s})} \qquad (4)$$

Central gaussian width is given as 2 pixels and surround gaussian width is given as 4 pixels. With this variables corresponding code is written to sample receptive field as 21x21 matrix. Calcul function is used for long calculation and Dog 21x21 matrix is created for displaying receptive field. Code can be seen below:

```
#Question 2
#part a
def Calcul(x,y):
    thetaS=4
    thetaC=2
    a=(1/(2*np.pi*thetaC*thetaC))*np.exp((-(x*x+y*y)/(2*thetaC*thetaC)))-
(1/(2*np.pi*thetaS*thetaS))*np.exp((-(x*x+y*y)/(2*thetaS*thetaS)))
    return a

Dog=np.zeros((21,21))
for x in range(-10,10):
    for y in range(-10,10):
        a=Calcul(x,y)
        Dog[x+10,y+10]=a
plt.figure()
plt.imshow(Dog)
plt.colorbar()
plt.title('DOG Receptive Field')
np.shape(Dog)
```

Display can be seen in the image below:

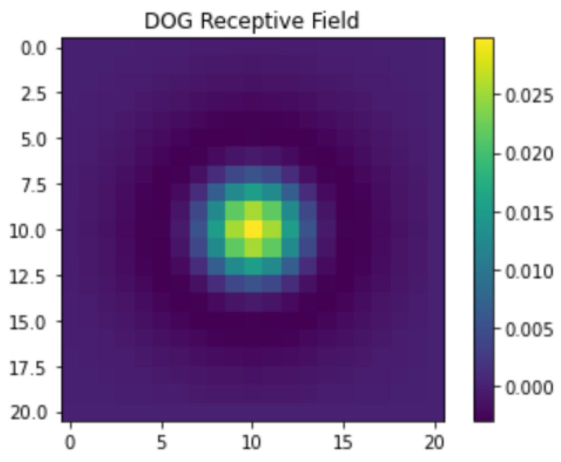

Fig.14: Dog Receptive Field

b)In part b first I have uploaded the image of monkey then I have used the receptive field I have created in part a for every pixel by centring the 21x21 matrix for each pixel. Image given in the homework is 480 to 512. Thus this method either gives error or creates artifacts on the edges after 470 or below 10 for row axis and edges after 502 or below 10 for column axis. This numbers come from the fact that while centring a 21x21 matrix edges are 10 to 11 distance away from the edges while doing the pixel by pixel centring. To handle this problem I have limited the receptive field centring in a way that it never goes out of bounds by keeping the indexes same after these boundaries in the for loop. This way edges were protected. Code can be seen below:

```
#part b
monk=plt.imread("hw2_image.bmp")
plt.figure()
plt.imshow(monk)
plt.title('original image')
np.shape(monk)
monk2=np.zeros((480,512))
print(np.shape(monk))
for i in range(0,480):
    for k in range(0,512):
        i1=i
        k1=k
        if i1<=10:
            i1=10
        if i1>=470:
            i1=469
        if k1<=10:
            k1=10
        if k1>=502:
            k1=501
        h=monk[i1-10:i1+11,k1-10:k1+11,0]*Dog
        monk2[i,k]=np.sum(h)
plt.figure()
plt.imshow(monk2,cmap='gray')
plt.title('Neural Activity')
plt.colorbar()
```

Resulting figures can be seen below:

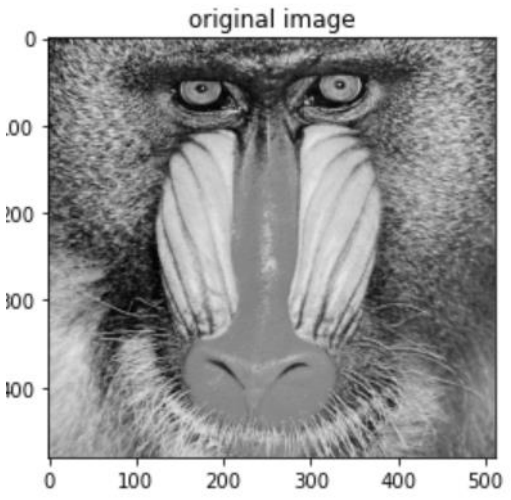

Fig.15: Original Image Given in the Homework

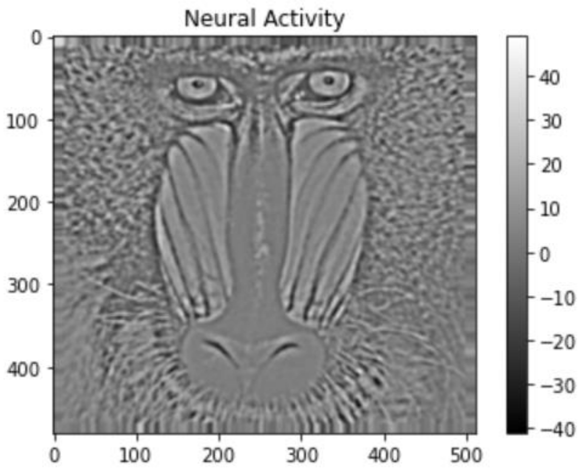

Fig.16: DOG Filtered

Comparing two images it can be seen that edges are more clear after applying the DOG filter. Thus we can say that LGN cells are used for detecting the edges of visual stimuli.

c)Edge detector is created as it is asked in the part c. Different threshold values has been tested for finding the optimal one. Resulting images made clear that observations in part b is true clearly because with the help of the images that has been passed from the edge detector it is clear that edges are more clear in the filtered version. Code for edge detector and trial of different threshold values can be seen below:

```
#partc
def edgedetec(h,threshold):
    a=np.zeros((480,512))

    for i in range(0,480):
        for k in range(0,512):
            if h[i,k]<threshold:
                a[i,k]=0
            else:
                a[i,k]=1
    return a

edgedetected=edgedetec(monk2,2)
plt.figure()
plt.imshow(edgedetected,cmap='gray')
plt.title("threshhold=2")
edgedetected=edgedetec(monk2,10)
plt.figure()
plt.imshow(edgedetected,cmap='gray')
plt.title("threshhold=10")
edgedetected=edgedetec(monk2,1)
plt.figure()
plt.imshow(edgedetected,cmap='gray')
plt.title("threshhold=1")
edgedetected=edgedetec(monk2,0)
plt.figure()
plt.imshow(edgedetected,cmap='gray')
plt.title("threshhold=0")
edgedetected=edgedetec(monk2,-1)
plt.figure()
plt.imshow(edgedetected,cmap='gray')
plt.title("threshhold= -1")
edgedetected=edgedetec(monk2,-2)
plt.figure()
plt.imshow(edgedetected,cmap='gray')
plt.title("threshhold=-2")
```

Resulting figures can be seen below:

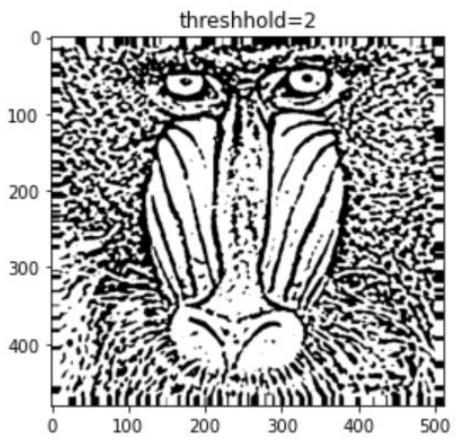

Fig.17: Threshold =2 Edge Detector

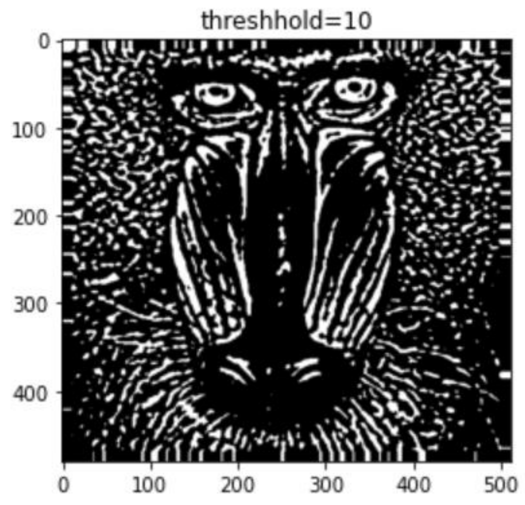

Fig.18: Threshold =10 Edge Detector

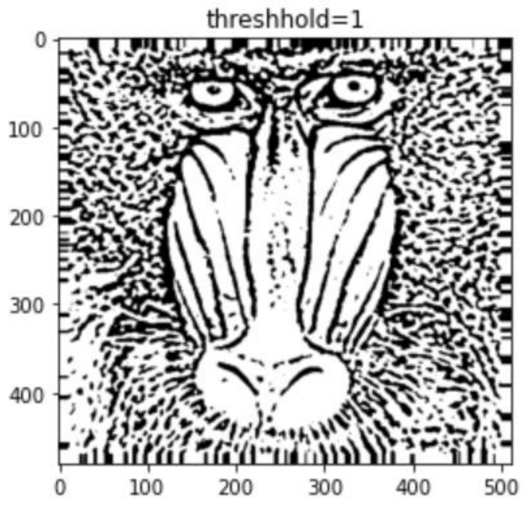

Fig.19: Threshold =1 Edge Detector

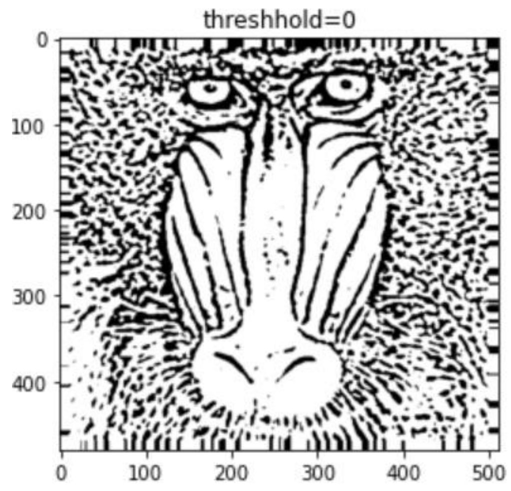

Fig.20: Threshold =0 Edge Detector

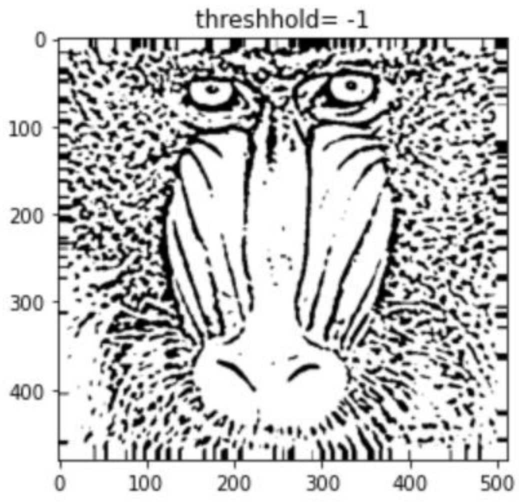

Fig.21: Threshold =-1 Edge Detector

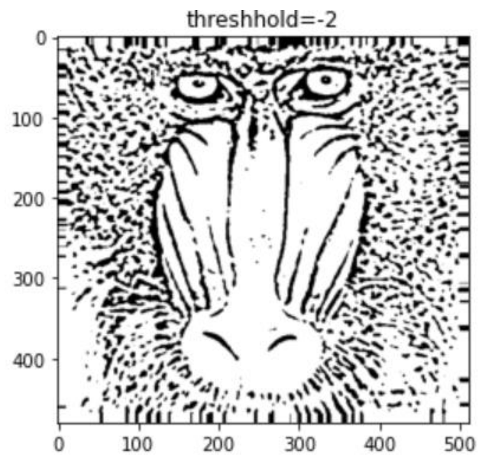

Fig.22: Threshold =-2 Edge Detector

From the resulting images it can be concluded that edge detection gives the best perfonmanca around the values 1 and 0. When it increases performance highly suffers. Best threshold value can be concluded as 0.

d) Like in part a we are asked to create the Gabor filter. Equation of Gabor filter is given as:

$$\exp\left(-\frac{(k(\theta).x)^2}{2\sigma_l{}^2} - \frac{(k_{ort}(\theta).x)^2}{2\sigma_w{}^2}\right)\cos\left(\frac{2\pi k_{ort}(\theta).x}{\lambda} + \varphi\right) \qquad (5)$$

Unit vector with orientation θ can be defined as:

$$k(\theta) = cos(\theta) + sin(\theta)$$

Because cos^2 and sin^2 addition equals to 1 and orientation angle can be represented as trigonometric functions this display is chosen. Dot product of two vectors that are orthogonal is 0. From this I have defined an orthogonal function as:

$$k_{ort}(\theta) = -sin(\theta) + cos(\theta)$$

Resulting dot product is 0 thus two unit vectors are orthogonal. Then I have written a function for Gabor with given values and displayed the generated receptive field as 21x21 matrix. Code can be seen below:

```
#part d
def Gabor(x,y,o):
    sigmaL=3
    sigmaw=3
    theta1=0
    landa=6
    k=np.array([np.cos(o),np.sin(o)])
    kOrt=np.array([-np.sin(o),np.cos(o)])
    c=[x,y]
    var1=np.dot(k,c)*np.dot(k,c)/(2*sigmaL*sigmaL)
    var2=np.dot(kOrt,c)*np.dot(kOrt,c)/(2*sigmaw*sigmaw)
    var3=np.cos(((2*np.pi*(np.dot(kOrt,c)))/landa)+theta1)
    b=np.exp(-var1-var2)*var3
    return b

res=np.zeros((21,21))
for x in range(-10,10):
    for y in range(-10,10):
        b=Gabor(x,y,np.pi/2)
        res[x+10,y+10]=b
plt.figure()
plt.imshow(res)
plt.title('Receptive Gabor Field')
plt.colorbar()
np.shape(res)
```

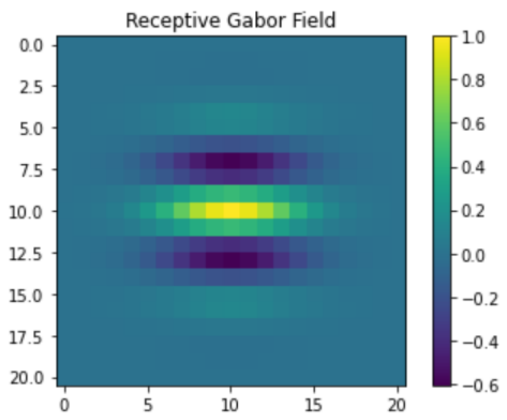

Fig.23: Receptive Gabor Field

e) Similar to part b neural responses has been centred to each corresponding receptive fields. Edges problems are avoided in the same method as in the part b. Code for the process and displaying the image can be seen below:

```
#part e
monke=plt.imread("hw2_image.bmp")
plt.figure()
plt.imshow(monke)
plt.title('original image')
np.shape(monke)
monk3=np.zeros((480,512))
print(np.shape(monke))
for i in range(0,480):
    for k in range(0,512):
        i1=i
        k1=k
        if i1<=10:
            i1=10
        if i1>=470:
            i1=469
        if k1<=10:
            k1=10
        if k1>=502:
            k1=501
        h1=monke[i1-10:i1+11,k1-10:k1+11,0]*res
        monk3[i,k]=np.sum(h1)
plt.figure()
plt.imshow(monk3,cmap='gray')
plt.title('Gabor Filtred')
plt.colorbar()
```

Resulting images can be seen below:

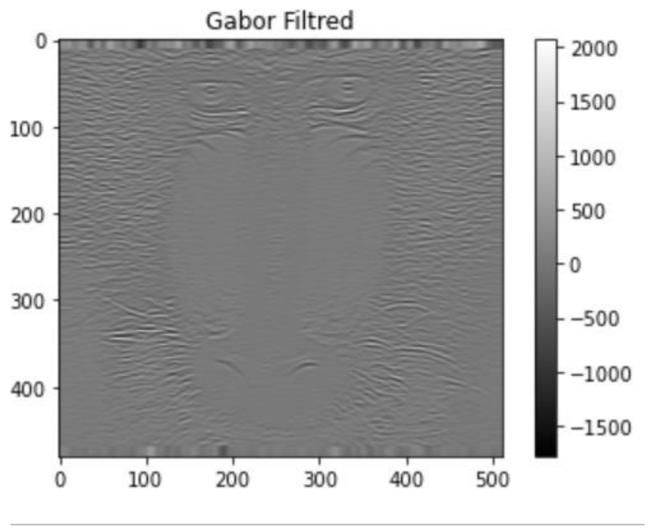

Fig.24: Gabor Filtered Image

Gabor’s main function is to highlight features of the image with given orientations with given theta. As we can see with pi/2 horizontal features of the image more focused and detected.

Name: Cemal Güven Adal
ID: 21703986

f) As it is asked in the question Gabor receptive fields with different orientations are used in the question. It is discussed in the previous part Gabor filter orientation focuses on the image lines with that orientation. In the code both Gabor receptive field arrays and corresponding images are displayed. In 0 orientation vertical values are focused and with each change to the Gabor focus changes as it can be seen from the images. Code is given below:

```
#part f
#0
resf1=np.zeros((21,21))
for x in range(-10,10):
    for y in range(-10,10):
        b=Gabor(x,y,0)
        resf1[x+10,y+10]=b
plt.figure()
plt.title('Gabor Receptive Field theta=0')
plt.imshow(resf1)
monkf1=np.zeros((480,512))
for i in range(0,480):
    for k in range(0,512):
        i1=i
        k1=k
        if i1<=10:
            i1=10
        if i1>=470:
            i1=469
        if k1<=10:
            k1=10
        if k1>=502:
            k1=501
        hf1=monke[i1-10:i1+11,k1-10:k1+11,0]*resf1
        monkf1[i,k]=np.sum(hf1)
plt.figure()
plt.imshow(monkf1,cmap='gray')
plt.title('Gabor Filtered Image theta=0')
plt.colorbar()
#pi/6

resf2=np.zeros((21,21))
for x in range(-10,10):
    for y in range(-10,10):
        b=Gabor(x,y,np.pi/6)
        resf2[x+10,y+10]=b
plt.figure()
plt.imshow(resf2)
plt.title('Gabor Receptive Field theta=pi/6')

monkf2=np.zeros((480,512))
for i in range(0,480):
    for k in range(0,512):
        i1=i
        k1=k
        if i1<=10:
            i1=10
        if i1>=470:
            i1=469
        if k1<=10:
            k1=10
        if k1>=502:
            k1=501
        hf2=monke[i1-10:i1+11,k1-10:k1+11,0]*resf2
        monkf2[i,k]=np.sum(hf2)
plt.figure()
plt.imshow(monkf2,cmap='gray')
plt.title('Gabor Filtered Image theta=pi/6')

plt.colorbar()

#pi/3

resf3=np.zeros((21,21))
for x in range(-10,10):
    for y in range(-10,10):
        b=Gabor(x,y,np.pi/3)
        resf3[x+10,y+10]=b
plt.figure()
plt.imshow(resf3)
plt.title('Gabor Receptive Field theta=pi/3')

monkf3=np.zeros((480,512))
for i in range(0,480):
    for k in range(0,512):
        i1=i
        k1=k
        if i1<=10:
            i1=10
        if i1>=470:
            i1=469
        if k1<=10:
            k1=10
        if k1>=502:
            k1=501
        hf3=monke[i1-10:i1+11,k1-10:k1+11,0]*resf3
        monkf3[i,k]=np.sum(hf3)
plt.figure()
plt.imshow(monkf3,cmap='gray')
plt.title('Gabor Filtered Image theta=pi/3')

plt.colorbar()

#pi/2
```

```
resf4=np.zeros((21,21))
for x in range(-10,10):
    for y in range(-10,10):
        b=Gabor(x,y,np.pi/2)
        resf4[x+10,y+10]=b
plt.figure()
plt.imshow(resf4)
plt.title('Gabor Receptive Field theta=pi/2')

monkf4=np.zeros((480,512))
for i in range(0,480):
    for k in range(0,512):
        i1=i
        k1=k
        if i1<=10:
            i1=10
        if i1>=470:
            i1=469
        if k1<=10:
            k1=10
        if k1>=502:
            k1=501
        hf4=monke[i1-10:i1+11,k1-10:k1+11,0]*resf4
        monkf4[i,k]=np.sum(hf4)
plt.figure()
plt.imshow(monkf4,cmap='gray')
plt.title('Gabor Filtered Image theta=pi/2')
plt.colorbar()
#sum of all
allmonkey=monkf1+monkf2+monkf3+monkf4
plt.figure()
plt.imshow(allmonkey,cmap='gray')
plt.title('Gabor Filtered Sum')
plt.colorbar()
```

Resulted figures can be seen below:

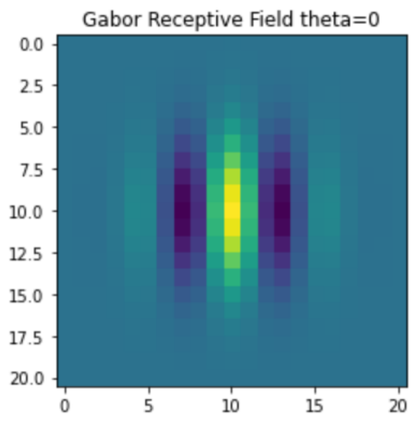

Fig.25: Gabor Receptive Field theta=0

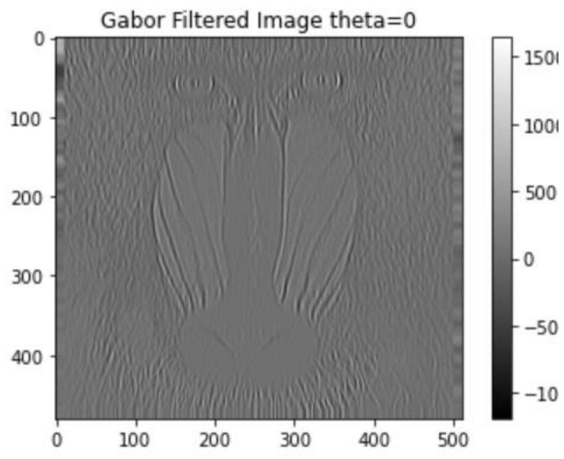

Fig.26: Gabor Filtered Image theta=0

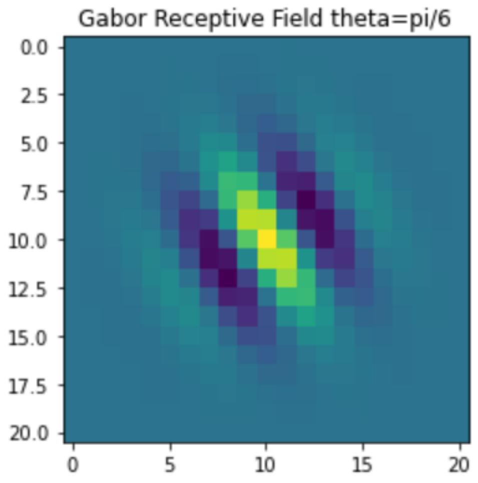

Fig.27: Gabor Receptive Field theta=pi/6

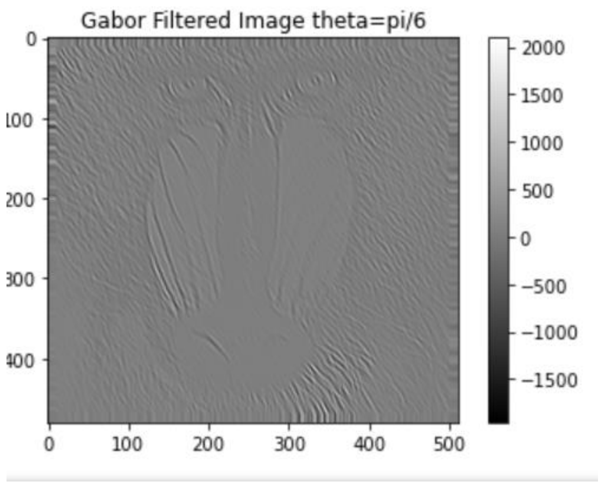

Fig.26: Gabor Filtered Image theta=pi/6

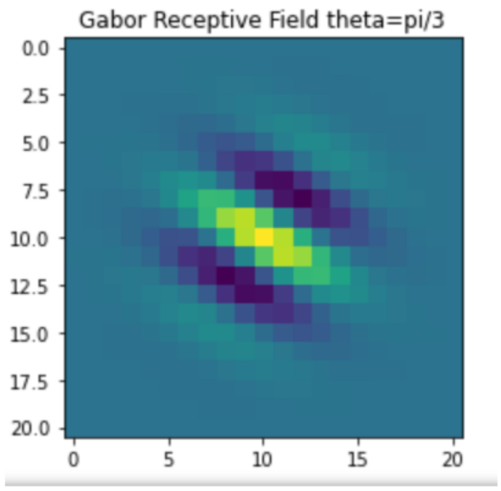

Fig.27: Gabor Receptive Field theta=pi/3

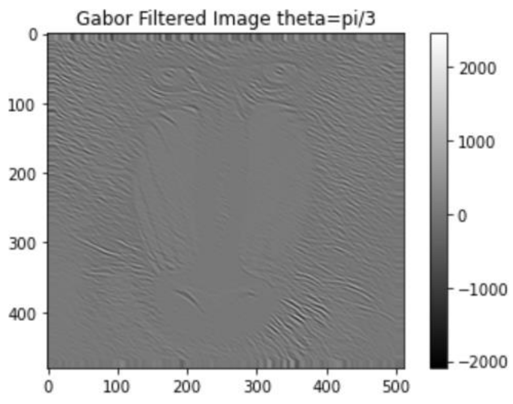

Fig.28: Gabor Filtered Image theta=pi/3

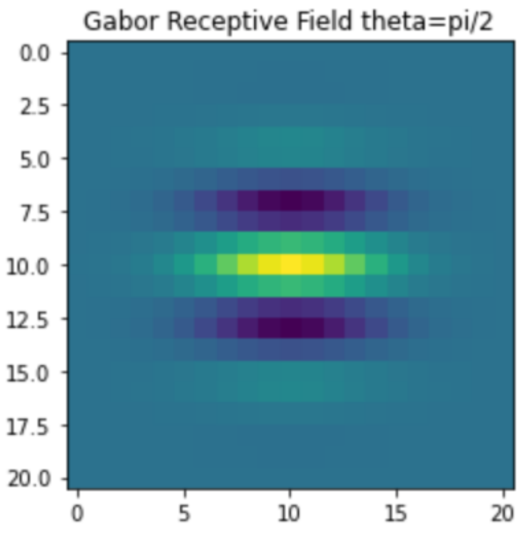

Fig.29: Gabor Receptive Field theta=pi/2

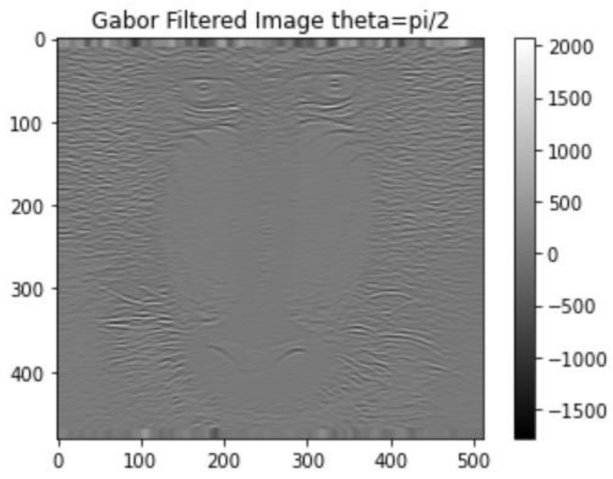

Fig.30: Gabor Filtered Image theta=pi/2

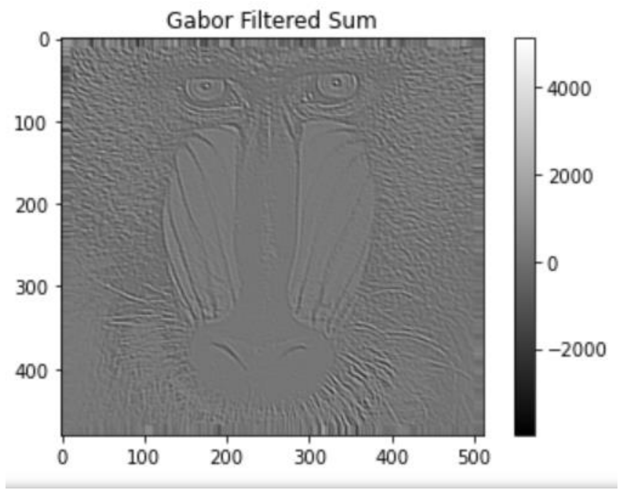

Fig.31: Gabor Filtered Image Summed

As it can be seen from the figures orientation of Gabor determines which lines of the image is shown more apparently. Theta determines which angle lines are apparent for example if theta pi/2 horizonal lines are more apparent. With this analysis when all images with different Gabor thetas are summed over all lines will be more detectable. The last image is clearly more focused for each line. It is because various lines with different angles are focused compared to other figures which only some lines with corresponding angles are focused.

Edge detector again is used and different threshold values are tested. Code can be seen below:

```
edgedetectedmonkey=edgedetec(allmonkey,0)
plt.figure()
plt.imshow(edgedetectedmonkey,cmap='gray')
plt.title('Threshold=0')
edgedetectedmonkey=edgedetec(allmonkey,400)
plt.figure()
plt.imshow(edgedetectedmonkey,cmap='gray')
plt.title('Threshold=400')
```

```
edgedetectedmonkey=edgedetec(allmonkey,1000)
plt.figure()
plt.imshow(edgedetectedmonkey,cmap='gray')
plt.title('Threshold=1000')
edgedetectedmonkey=edgedetec(allmonkey,-10)
plt.figure()
plt.imshow(edgedetectedmonkey,cmap='gray')
plt.title('Threshold=-10')
edgedetectedmonkey=edgedetec(allmonkey,100)
plt.figure()
plt.imshow(edgedetectedmonkey,cmap='gray')
plt.title('Threshold=100')
```

Figures can be seen below:

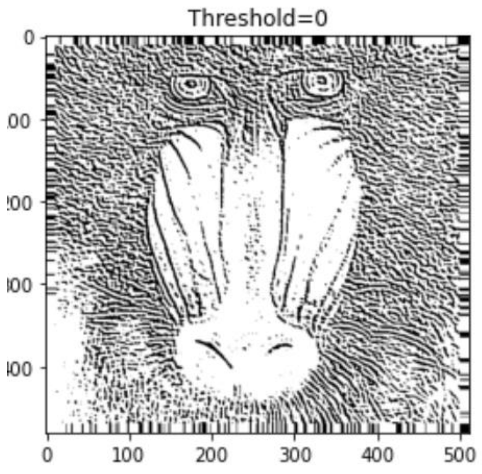

Fig.32: Threshold=0

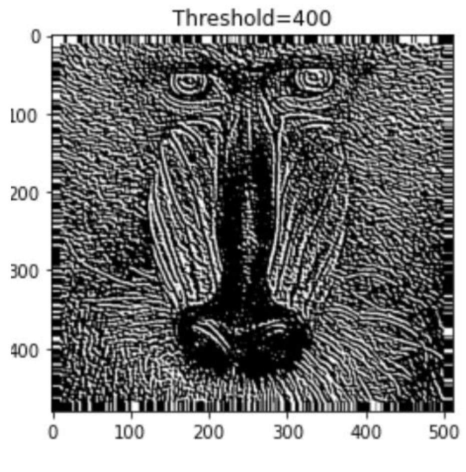

Fig.33: Threshold=400

Name: Cemal Güven Adal
ID: 21703986

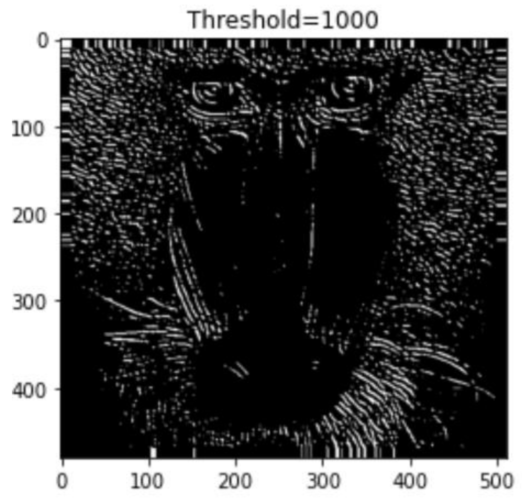

Fig.34: Threshold=1000

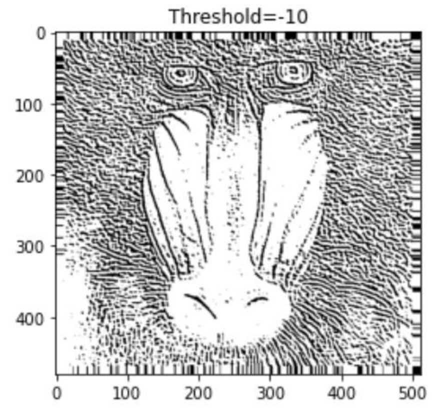

Fig.35: Threshold=-10

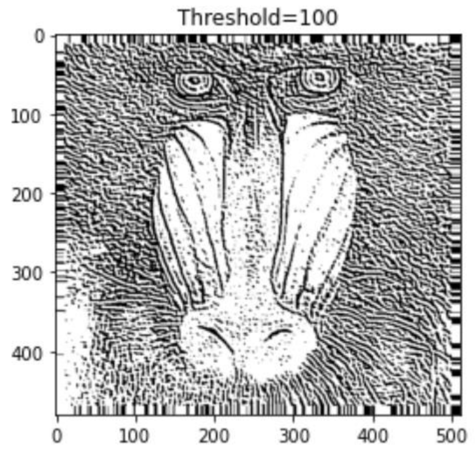

Fig.36: Threshold=100

From the images it can be seen that threshold values gives appropriate values till 100 and performance decreases after values increases to 400s so but between -10 and 100 gives the best results as it can be seen from the figures.

Name: Cemal Güven Adal
ID: 21703986

References

Greenberg, M. D. (1988). *Advanced engineering mathematics*. Englewood Cliffs, NJ: Prentice-Hall Internat.

Space-time separable RF. [Online]. Available: http://ohzawa-lab.bpe.es.osakau.ac.jp/ohzawa-lab/teaching/RF/XTseparable.html#:~:text=Space%2Dtime%20separable%20receptive%20field%20of%20a%20simple%20cell&text=as%20the%20product%20of%20two,to%20be%20space%2Dtime%20separable. [Accessed: 15-Mar-2021].

Appendices

Code

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.io as sio
import sys
# In[2]:
def CemalGuven_Adal_21703986_hw2(question):
    if question == '1':
        # --------------------------------------------------------------------------Question1-------------------------------------------------------------------
        # Question 1 A-----------------------------------------
        # Defining array A which is given in the Homework question 1
        # In[2]:

        # loading .mat file to python
        data1 = sio.loadmat('c2p3.mat')
        data1.keys()



        # breaking down stim and counts from.mat file and checking their sizes
        stim = data1['stim']
        counts = data1['counts']
        print(np.shape(stim))
        print(np.shape(counts))






        # STA function
        def STA(counts, stim):
            average = np.zeros((16, 16, 10))
            for i in range(10, counts.shape[0]):
```

```
        average = average + stim[:, :, i - 10:i] * counts[i, :]
    average = average / (np.sum(counts))

    return average



aver = STA(counts, stim)
print(np.size(aver))
for i in range(0, 10):
    plt.figure()
    plt.imshow(aver[:, :, i], vmin=np.min(aver), vmax=np.max(aver), cmap='gray')
    plt.colorbar()
    plt.title(str(10 - i) + 'Steps Before Spike')



# partb
plt.figure()
averagerow = np.sum(aver, axis=0)
plt.title("Row Summed Averages")
plt.imshow(averagerow, cmap='gray')

plt.figure()
averagecol = np.sum(aver, axis=1)
plt.title("Coloumn Summed Averages")
plt.imshow(averagerow, cmap='gray')



# PARTC
# for all stimlus
aver1 = aver[:, :, -1]  # for takng the last one
hist1 = np.zeros(32767)
for i in range(32767):
    hist1[i] = np.sum(aver1 * stim[:, :, i])
plt.figure(figsize=(8, 10))
plt.hist(hist1, density='True', bins=90, rwidth=0.70)
plt.title('Stimulus Projection for all Spikes')
plt.show()
# for stimulus non 0's
hist2 = np.array([])

for i in range(32767):
    if counts[i, :] != 0:
        sumvalue = np.sum(aver1 * stim[:, :, i])
        hist2 = np.append(hist2, sumvalue)
plt.figure(figsize=(8, 10))
plt.hist(hist2, density='True', bins=90, rwidth=0.70)
plt.title('Stimulus Projection for non zero Spikes')
```

```
        plt.show()




    elif question == '2':
        # part a
        def Calcul(x, y):
            thetaS = 4
            thetaC = 2
            a = (1 / (2 * np.pi * thetaC * thetaC)) * np.exp((-(x * x + y * y) / (2 * thetaC *
thetaC))) - (
                        1 / (2 * np.pi * thetaS * thetaS)) * np.exp((-(x * x + y * y) / (2 * thetaS *
thetaS)))
            return a

        Dog = np.zeros((21, 21))
        for x in range(-10, 10):
            for y in range(-10, 10):
                a = Calcul(x, y)
                Dog[x + 10, y + 10] = a
        plt.figure()
        plt.imshow(Dog)
        plt.colorbar()
        plt.title('DOG Receptive Field')
        np.shape(Dog)



        # part b
        monk = plt.imread("hw2_image.bmp")
        plt.figure()
        plt.imshow(monk)
        plt.title('original image')
        np.shape(monk)
        monk2 = np.zeros((480, 512))
        print(np.shape(monk))
        for i in range(0, 480):
            for k in range(0, 512):
                i1 = i
                k1 = k
                if i1 <= 10:
                    i1 = 10
                if i1 >= 470:
                    i1 = 469
                if k1 <= 10:
                    k1 = 10
                if k1 >= 502:
                    k1 = 501
                h = monk[i1 - 10:i1 + 11, k1 - 10:k1 + 11, 0] * Dog
        monk2[i, k] = np.sum(h)
plt.figure()
plt.imshow(monk2, cmap='gray')
plt.title('Neural Activity')
plt.colorbar()



# partc
def edgedetec(h, threshold):
    a = np.zeros((480, 512))

    for i in range(0, 480):
        for k in range(0, 512):
            if h[i, k] < threshold:
                a[i, k] = 0
            else:
                a[i, k] = 1
    return a

edgedetected = edgedetec(monk2, 2)
plt.figure()
plt.imshow(edgedetected, cmap='gray')
plt.title("threshhold=2")
edgedetected = edgedetec(monk2, 10)
plt.figure()
plt.imshow(edgedetected, cmap='gray')
plt.title("threshhold=10")
edgedetected = edgedetec(monk2, 1)
plt.figure()
plt.imshow(edgedetected, cmap='gray')
plt.title("threshhold=1")
edgedetected = edgedetec(monk2, 0)
plt.figure()
plt.imshow(edgedetected, cmap='gray')
plt.title("threshhold=0")
edgedetected = edgedetec(monk2, -1)
plt.figure()
plt.imshow(edgedetected, cmap='gray')
plt.title("threshhold= -1")
edgedetected = edgedetec(monk2, -2)
plt.figure()
plt.imshow(edgedetected, cmap='gray')
plt.title("threshhold=-2")



# part d
def Gabor(x, y, o):
    sigmaL = 3
    sigmaw = 3
    theta1 = 0
    landa = 6
    k = np.array([np.cos(o), np.sin(o)])
    kOrt = np.array([-np.sin(o), np.cos(o)])
    c = [x, y]
    var1 = np.dot(k, c) * np.dot(k, c) / (2 * sigmaL * sigmaL)
    var2 = np.dot(kOrt, c) * np.dot(kOrt, c) / (2 * sigmaw * sigmaw)
    var3 = np.cos(((2 * np.pi * (np.dot(kOrt, c))) / landa) + theta1)
    b = np.exp(-var1 - var2) * var3
    return b

res = np.zeros((21, 21))
for x in range(-10, 10):
    for y in range(-10, 10):
        b = Gabor(x, y, np.pi / 2)
        res[x + 10, y + 10] = b
plt.figure()
plt.imshow(res)
plt.title('Receptive Gabor Field')
plt.colorbar()
np.shape(res)



# part e
monke = plt.imread("hw2_image.bmp")
plt.figure()
plt.imshow(monke)
plt.title('original image')
np.shape(monke)
monk3 = np.zeros((480, 512))
print(np.shape(monke))
for i in range(0, 480):
    for k in range(0, 512):
        i1 = i
        k1 = k
        if i1 <= 10:
            i1 = 10
        if i1 >= 470:
            i1 = 469
        if k1 <= 10:
            k1 = 10
        if k1 >= 502:
            k1 = 501
        h1 = monke[i1 - 10:i1 + 11, k1 - 10:k1 + 11, 0] * res
        monk3[i, k] = np.sum(h1)
plt.figure()
plt.imshow(monk3, cmap='gray')
plt.title('Gabor Filtred')
plt.colorbar()



# part f
# 0
resf1 = np.zeros((21, 21))
for x in range(-10, 10):
    for y in range(-10, 10):
        b = Gabor(x, y, 0)
        resf1[x + 10, y + 10] = b
plt.figure()
plt.title('Gabor Receptive Field theta=0')
plt.imshow(resf1)
monkf1 = np.zeros((480, 512))
for i in range(0, 480):
    for k in range(0, 512):
        i1 = i
        k1 = k
        if i1 <= 10:
            i1 = 10
        if i1 >= 470:
            i1 = 469
        if k1 <= 10:
            k1 = 10
        if k1 >= 502:
            k1 = 501
        hf1 = monke[i1 - 10:i1 + 11, k1 - 10:k1 + 11, 0] * resf1
        monkf1[i, k] = np.sum(hf1)
plt.figure()
plt.imshow(monkf1, cmap='gray')
plt.title('Gabor Filtered Image theta=0')
plt.colorbar()
# pi/6

resf2 = np.zeros((21, 21))
for x in range(-10, 10):
    for y in range(-10, 10):
        b = Gabor(x, y, np.pi / 6)
        resf2[x + 10, y + 10] = b
plt.figure()
plt.imshow(resf2)
plt.title('Gabor Receptive Field theta=pi/6')

monkf2 = np.zeros((480, 512))
for i in range(0, 480):
    for k in range(0, 512):
        i1 = i
        k1 = k
        if i1 <= 10:
            i1 = 10
        if i1 >= 470:
            i1 = 469
        if k1 <= 10:
            k1 = 10
        if k1 >= 502:
            k1 = 501
        hf2 = monke[i1 - 10:i1 + 11, k1 - 10:k1 + 11, 0] * resf2
        monkf2[i, k] = np.sum(hf2)
plt.figure()
plt.imshow(monkf2, cmap='gray')
plt.title('Gabor Filtered Image theta=pi/6')

plt.colorbar()

# pi/3

resf3 = np.zeros((21, 21))
for x in range(-10, 10):
    for y in range(-10, 10):
        b = Gabor(x, y, np.pi / 3)
        resf3[x + 10, y + 10] = b
plt.figure()
plt.imshow(resf3)
plt.title('Gabor Receptive Field theta=pi/3')

monkf3 = np.zeros((480, 512))
for i in range(0, 480):
    for k in range(0, 512):
        i1 = i
        k1 = k
        if i1 <= 10:
            i1 = 10
        if i1 >= 470:
            i1 = 469
        if k1 <= 10:
            k1 = 10
        if k1 >= 502:
            k1 = 501
        hf3 = monke[i1 - 10:i1 + 11, k1 - 10:k1 + 11, 0] * resf3
        monkf3[i, k] = np.sum(hf3)
plt.figure()
plt.imshow(monkf3, cmap='gray')
plt.title('Gabor Filtered Image theta=pi/3')

plt.colorbar()

# pi/2

resf4 = np.zeros((21, 21))
```

```
for x in range(-10, 10):
    for y in range(-10, 10):
        b = Gabor(x, y, np.pi / 2)
        resf4[x + 10, y + 10] = b
plt.figure()
plt.imshow(resf4)
plt.title('Gabor Receptive Field theta=pi/2')

monkf4 = np.zeros((480, 512))
for i in range(0, 480):
    for k in range(0, 512):
        i1 = i
        k1 = k
        if i1 <= 10:
            i1 = 10
        if i1 >= 470:
            i1 = 469
        if k1 <= 10:
            k1 = 10
        if k1 >= 502:
            k1 = 501
        hf4 = monke[i1 - 10:i1 + 11, k1 - 10:k1 + 11, 0] * resf4
        monkf4[i, k] = np.sum(hf4)
plt.figure()
plt.imshow(monkf4, cmap='gray')
plt.title('Gabor Filtered Image theta=pi/2')
plt.colorbar()
# sum of all
allmonkey = monkf1 + monkf2 + monkf3 + monkf4
plt.figure()
plt.imshow(allmonkey, cmap='gray')
plt.title('Gabor Filtered Sum')
plt.colorbar()



edgedetectedmonkey = edgedetec(allmonkey, 0)
plt.figure()
plt.imshow(edgedetectedmonkey, cmap='gray')
plt.title('Threshold=0')
edgedetectedmonkey = edgedetec(allmonkey, 400)
plt.figure()
plt.imshow(edgedetectedmonkey, cmap='gray')
plt.title('Threshold=400')
edgedetectedmonkey = edgedetec(allmonkey, 1000)
plt.figure()
plt.imshow(edgedetectedmonkey, cmap='gray')
plt.title('Threshold=1000')
edgedetectedmonkey = edgedetec(allmonkey, -10)
plt.figure()
```

```
        plt.imshow(edgedetectedmonkey, cmap='gray')
        plt.title('Threshold=-10')
        edgedetectedmonkey = edgedetec(allmonkey, 100)
        plt.figure()
        plt.imshow(edgedetectedmonkey, cmap='gray')
        plt.title('Threshold=100')

        # In[ ]:

        # -------------------QUESION2------------------------------------------------------------------
------------------------------




question = input("enter question number")
CemalGuven_Adal_21703986_hw2(question)
```