CS 682: COMPUTER VISION

Abhishek Bodas

G01160204

HW3

Website: http://mason.gmu.edu/~abodas/vision/

Username: CS682

Password: abodas682

Q1)

Gray level and color edges

- Load all the images from the directory and blur the image.
- For each image, compute image gradients using cv2.sobel() operator with kernel size=3.
- Compute the magnitude using cartToPolar() of x and y gradients and discard negative edges and display the magnitude
- For color compute all x,y gradients of all 3 BGR color channels

Gray:

```
import cv2
import glob #To import glob module
import numpy as np
from matplotlib import pyplot as plt

grayhist=[]
print("Grayscale")
for eachimage in glob.glob('ST2main/*.jpg'): #Location of unziped image folder
    print(eachimage)
    grayimage = cv2.imread(eachimage,0)
    grayimage=cv2.blur(grayimage,(5,5))

    grayx=cv2.Sobel(grayimage,cv2.CV_64F,1,0,ksize=3) #To compute image gradients
    grayy=cv2.Sobel(grayimage,cv2.CV_64F,0,1,ksize=3)

    graymagnitude,grayangle = cv2.cartToPolar(grayx,grayy, angleInDegrees=True)
    cv2.imshow("Gray edges",(np.uint8(graymagnitude)))
```

Color:

```
colorimage = cv2.imread(eachimage)
colorimage=cv2.blur(colorimage,(5,5))

bluex = cv2.Sobel(colorimage[:,:,0],cv2.CV_64F,1,0,ksize=3) #To compute image
gradients x,y of each colored channels
bluey = cv2.Sobel(colorimage[:,:,0],cv2.CV_64F,0,1,ksize=3)
greenx = cv2.Sobel(colorimage[:,:,1],cv2.CV_64F,1,0,ksize=3)
greeny = cv2.Sobel(colorimage[:,:,1],cv2.CV_64F,0,1,ksize=3)
redx = cv2.Sobel(colorimage[:,:,2],cv2.CV_64F,1,0,ksize=3)
redy = cv2.Sobel(colorimage[:,:,2],cv2.CV_64F,0,1,ksize=3)
colormagnitude,colorangle =
cv2.cartToPolar(bluex+greenx+redx,bluey+greeny+redy,angleInDegrees=True)
cv2.imshow("Color edges",(np.uint8(colormagnitude)))
```

Q2)

Compute direction from cartToPolar() round it off and plot histogram for gray

```
    grayangle = np.rint(np.divide(grayangle,10))
    ghist,gbin = np.histogram(grayangle,36,[0,36],weights=graymagnitude)  #36bins
    grayhist.append(ghist)
    plt.plot(ghist)
    plt.text(0,0,"Gray histograms", bbox=dict(facecolor='yellow', alpha=0.5))
    plt.show()
```

Q3)

Compute direction from cartToPolar() round it off and plot histogram for gray

```
    colorangle = np.rint(np.divide(colorangle,10))
    chist,cbin = np.histogram(colorangle,36,[0,36],weights=colormagnitude)  #36bins
    colorhist.append(chist)
    plt.plot(chist)
    plt.text(0,0,"Color histograms", bbox=dict(facecolor='yellow', alpha=0.5))
    plt.show()
```

Q4)

Histogram comparison functions, intersection and chi-square

```
def hintersection(h1,h2): #histogram comparison function by calculating intersection
    hmin = np.sum(np.minimum(h1, h2))
    hmax = np.sum(np.maximum(h1, h2))
    return float(hmin/hmax)

def hchisquared(h1,h2): #histogram comparison function by calculating chi square
    hchi=0
    for i in range(0,len(h1)):
```

```
        if (h1[i]+h2[i])>5:
            hchi+=(((h1[i]- h2[i])**2)/float(h1[i]+h2[i]))
    return hchi
```

Q5)

Using above 2 functions and plotting heatmaps for both color and gray histograms

Color:

```
#Using above functions to compare image pairs
intmatrix = np.zeros((99,99),dtype='uint8')
for i1 in range(0,99):
    for i2 in range(0,99):
        intmatrix[i1][i2] = (1-(hintersection(colorhist[i1],colorhist[i2])))*255
plt.imshow(intmatrix, cmap='hot')
plt.colorbar()
plt.text(0,0,"Intersection", bbox=dict(facecolor='yellow', alpha=0.5)) #To display
text
plt.show()

chimatrix = np.zeros((99,99), dtype='float64')
for i1 in range(0,99):
    for i2 in range(0,99):
        chimatrix[i1][i2]=hchisquared(colorhist[i1],colorhist[i2])
plt.imshow(chimatrix,cmap='hot')
plt.colorbar()
plt.text(0,0,"Chi squared", bbox=dict(facecolor='yellow', alpha=0.5)) #To display
text
plt.show()
#Display in sequential order, Close each window for the next window, press q to exit
```

Gray:

```
#Using above functions to compare image pairs
intmatrix = np.zeros((99,99),dtype='uint8')
for i1 in range(0,99):
    for i2 in range(0,99):
        intmatrix[i1][i2] = (1-(hintersection(grayhist[i1],grayhist[i2])))*255
plt.imshow(intmatrix, cmap='hot')
plt.colorbar()
plt.text(0,0,"Intersection", bbox=dict(facecolor='yellow', alpha=0.5)) #To display
text
plt.show()

chimatrix = np.zeros((99,99), dtype='float64')
for i1 in range(0,99):
    for i2 in range(0,99):
        chimatrix[i1][i2]=hchisquared(grayhist[i1],grayhist[i2])
plt.imshow(chimatrix,cmap='hot')
plt.colorbar()
plt.text(0,0,"Chi squared", bbox=dict(facecolor='yellow', alpha=0.5)) #To display
text
```

```
plt.show()
#Display in sequential order, Close each window for the next window, press q to quit
```

Extra Credit Q1)

Create a meshgrid and plot orientations using quiver()

Gray:

```
    axis1, axis2 = np.meshgrid(np.arange(1600), np.arange(1200)) #Plotting
    plt.subplot()
    plt.quiver(axis1[::50,::50], axis2[::50,::50], np.uint8(grayx[::50,::50]),
np.uint8(grayy[::50,::50]), units='width') #To display orientations
    plt.text(0,0,"Gray orientations", bbox=dict(facecolor='gray', alpha=0.5))
    plt.show()
```

Color:

```
    axis1, axis2 = np.meshgrid(np.arange(1600), np.arange(1200)) #Plotting
    plt.subplot()
    plt.quiver(axis1[::50,::50], axis2[::50,::50], np.uint8(bluex[::50,::50]),
np.uint8(bluey[::50,::50]), units='width') #To display orientations of channels
    plt.text(0,0,"Blue orientation", bbox=dict(facecolor='blue', alpha=0.5))
    plt.show()
    plt.quiver(axis1[::50,::50], axis2[::50,::50], np.uint8(greenx[::50,::50]),
np.uint8(greeny[::50,::50]), units='width')
    plt.title('green orientations')
    plt.text(0,0,"Green orientations", bbox=dict(facecolor='green', alpha=0.5))
    plt.show()
    plt.quiver(axis1[::50,::50], axis2[::50,::50], np.uint8(redx[::50,::50]),
np.uint8(redy[::50,::50]), units='width')
    plt.text(0,0,"Red orientations", bbox=dict(facecolor='red', alpha=0.5))
    plt.show()
```

Extra Credit Q2) and Q3)

Compute eigen values and eigen vectors using formula given in ColorGradients.pptx and using numpy methods for faster computation

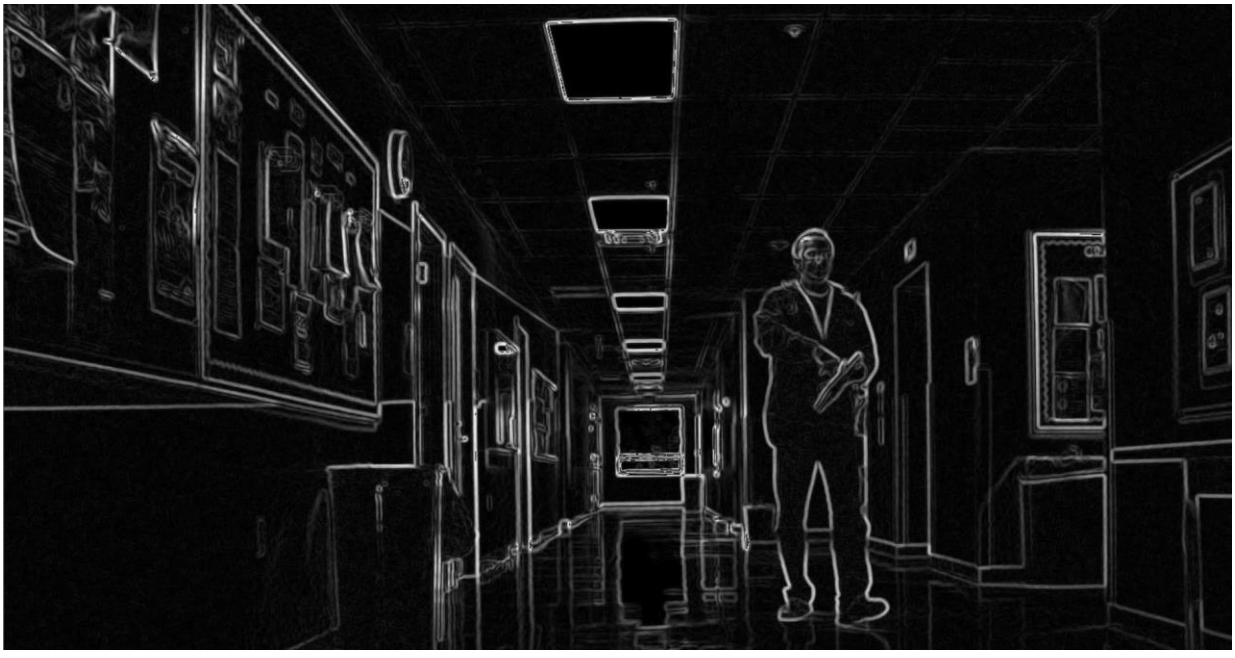
```
#Script to display edges of image using eigen values
import cv2
import glob #To import glob module
import numpy as np
from matplotlib import pyplot as plt

eigenhist=[]
print("Eigen")
```

```
for eachimage in glob.glob('ST2main/*.jpg'): #Location of unziped image folder
    print(eachimage)
    colorimage = cv2.imread(eachimage)
    colorimage=cv2.blur(colorimage,(5,5))

    bluex = cv2.Sobel(colorimage[:,:,0],cv2.CV_64F,1,0,ksize=3) #To compute image
gradients x,y of each colored channels
    bluey = cv2.Sobel(colorimage[:,:,0],cv2.CV_64F,0,1,ksize=3)
    greenx = cv2.Sobel(colorimage[:,:,1],cv2.CV_64F,1,0,ksize=3)
    greeny = cv2.Sobel(colorimage[:,:,1],cv2.CV_64F,0,1,ksize=3)
    redx = cv2.Sobel(colorimage[:,:,2],cv2.CV_64F,1,0,ksize=3)
    redy = cv2.Sobel(colorimage[:,:,2],cv2.CV_64F,0,1,ksize=3)

    a = np.square(bluex) + np.square(greenx) + np.square(redx)
#Matrix S= |a b|
    b = np.multiply(bluex,bluey) + np.multiply(greenx,greeny) +
np.multiply(redx,redy)   # |b c|
    c = np.square(bluey) + np.square(greeny) + np.square(redy)

    elambda=0.5*((a+c) + np.sqrt(np.square(a+c)-4*(np.multiply(a,c)-np.square(b))))
#Formula for lambda
    emagnitude = np.sqrt(elambda)
    cv2.imshow("eigen edges",(np.uint8(emagnitude)))

    x = np.divide(-b,a-elambda,where=a - elambda != 0)    #x=(-b/a-lambda)
    e1 = np.divide(x,np.sqrt(np.square(x)+1))
    e2 = np.divide(1,np.sqrt(np.square(x)+1))
    eangle = cv2.phase(e1,e2,angleInDegrees=True) #Direction in degrees
    eangle = np.rint(np.divide(eangle,5))

    ehist,ebin = np.histogram(eangle,36,[0,36])  #36bins
    eigenhist.append(ehist)
    plt.plot(ehist)
    plt.text(0,0,"Eigen histograms", bbox=dict(facecolor='yellow', alpha=0.5))
    plt.show()

def hintersection(h1,h2): #histogram comparison function by calculating intersection
    hmin = np.sum(np.minimum(h1, h2))
    hmax = np.sum(np.maximum(h1, h2))
    return float(hmin/hmax)

def hchisquared(h1,h2): #histogram comparison function by calculating chi square
    hchi=0
    for i in range(0,len(h1)):
        if (h1[i]+h2[i])>5:
            hchi+=(((h1[i]- h2[i])**2)/float(h1[i]+h2[i]))
    return hchi

#Using above functions to compare image pairs
intmatrix = np.zeros((99,99),dtype='uint8')
for i1 in range(0,99):
    for i2 in range(0,99):
        intmatrix[i1][i2] = (1-(hintersection(eigenhist[i1],eigenhist[i2])))*255
plt.imshow(intmatrix, cmap='hot')
plt.colorbar()
```

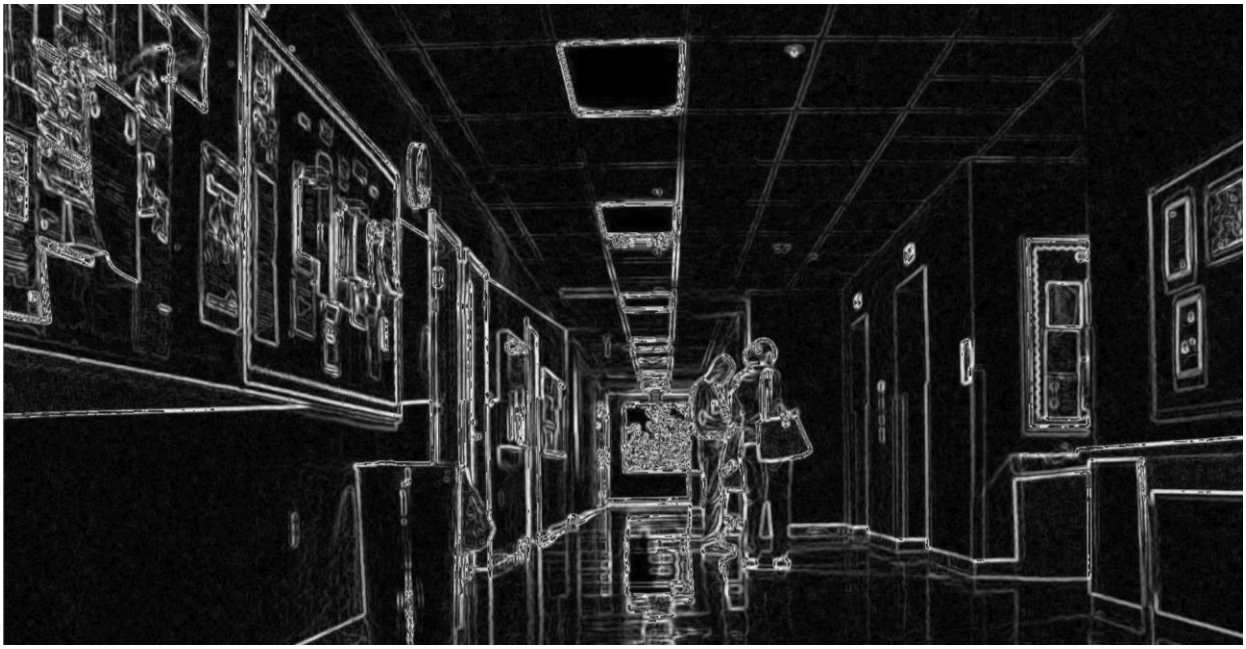
```
plt.text(0,0,"Intersection", bbox=dict(facecolor='yellow', alpha=0.5)) #To display
text
plt.show()

chimatrix = np.zeros((99,99), dtype='float64')
for i1 in range(0,99):
    for i2 in range(0,99):
        chimatrix[i1][i2]=hchisquared(eigenhist[i1],eigenhist[i2])
plt.imshow(chimatrix,cmap='hot')
plt.colorbar()
plt.text(0,0,"Chi squared", bbox=dict(facecolor='yellow', alpha=0.5)) #To display
text
plt.show()
#Display in sequential order, Close each window for the next window,press q to exit
```

Output images:

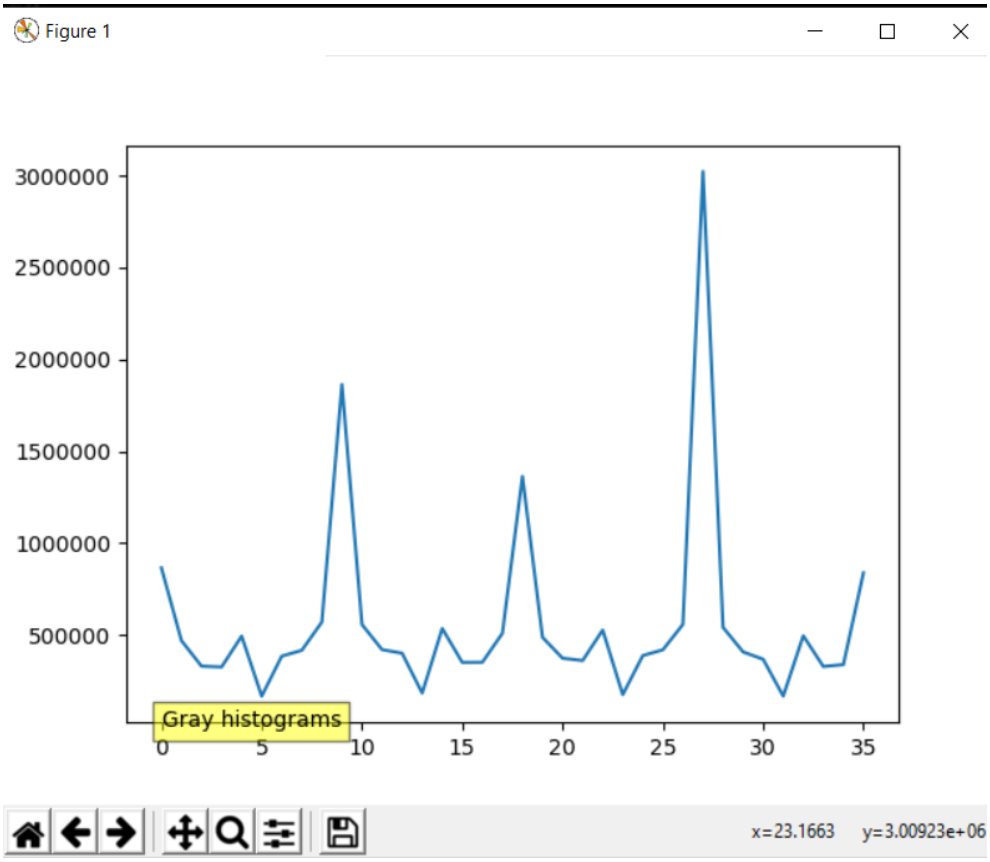
Gray Edges

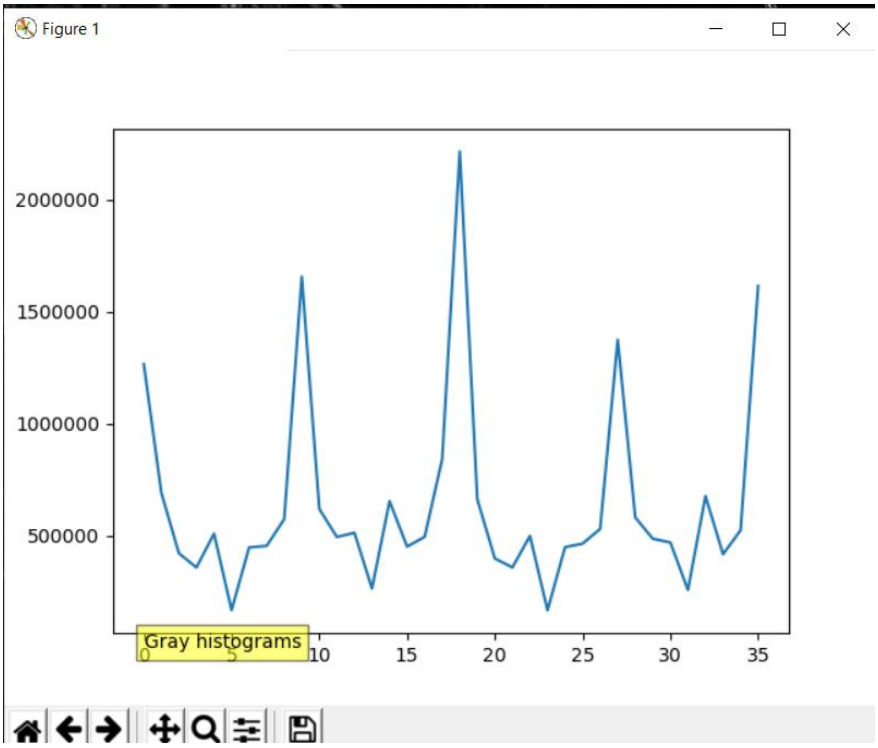


Color edges:

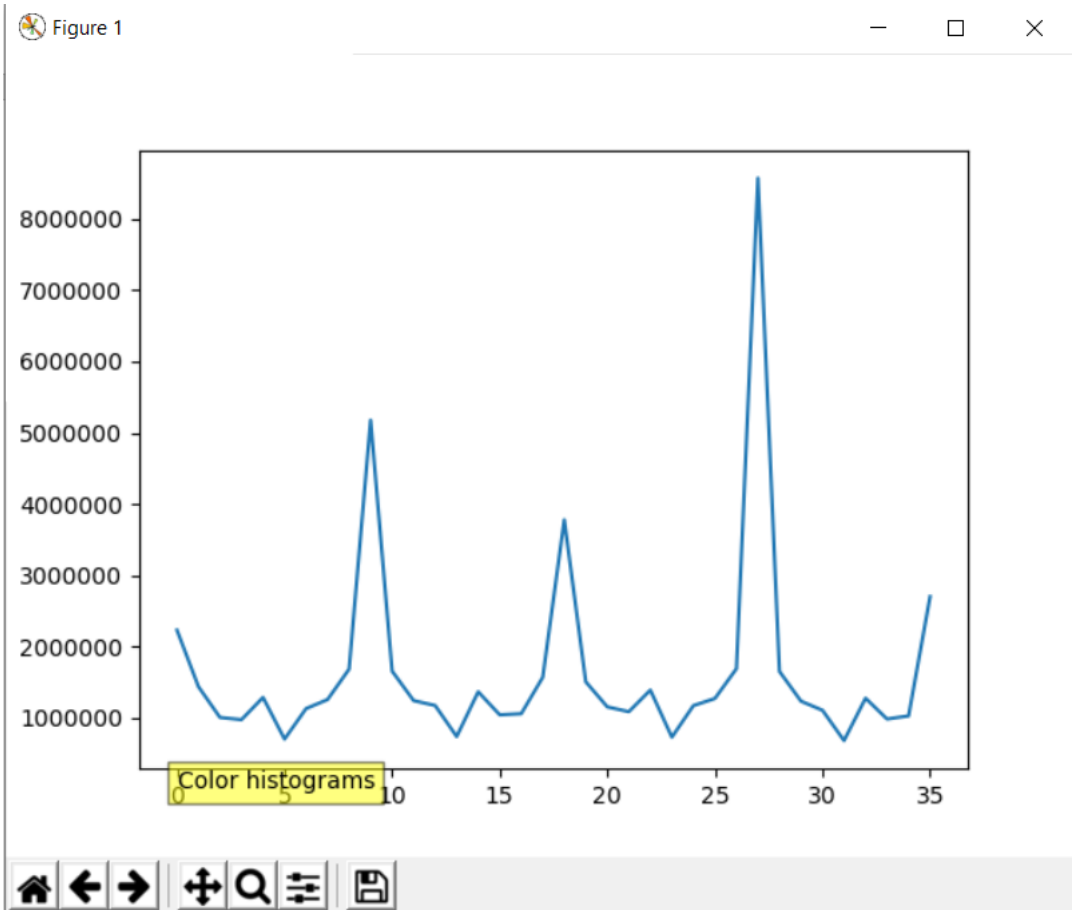


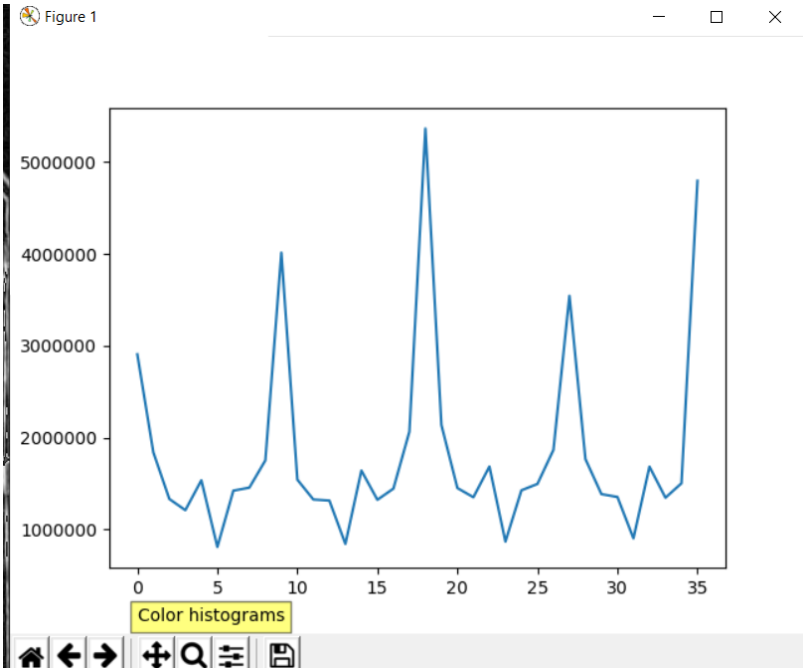
Gray histogram:



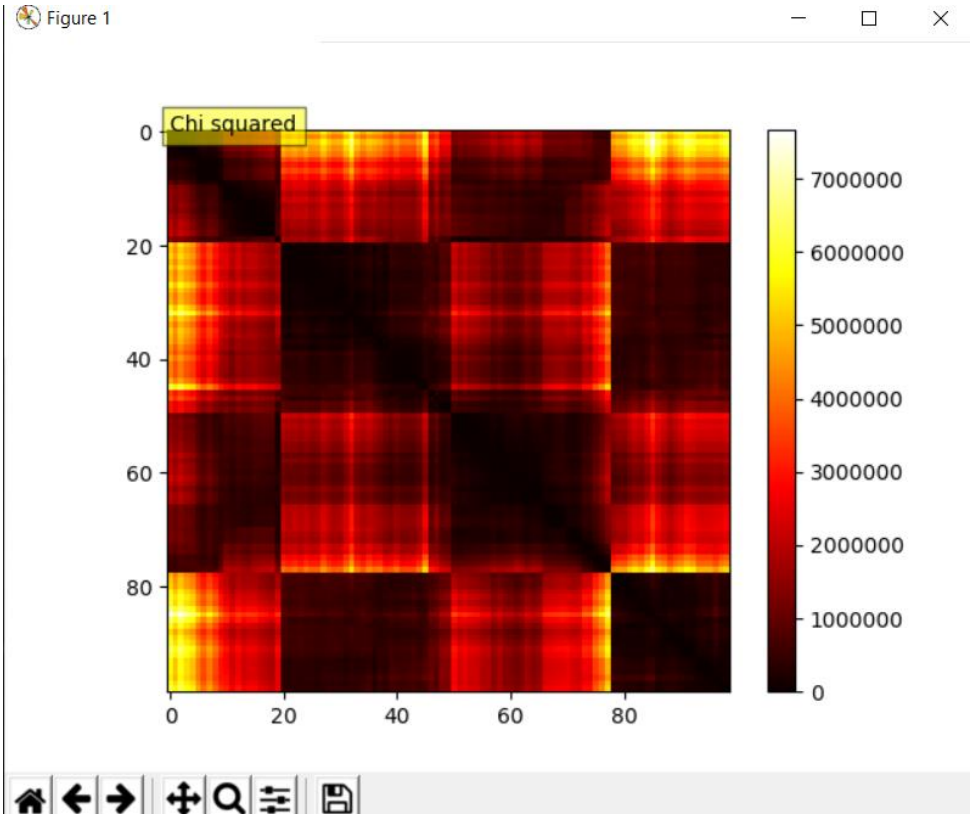


Color histogram:

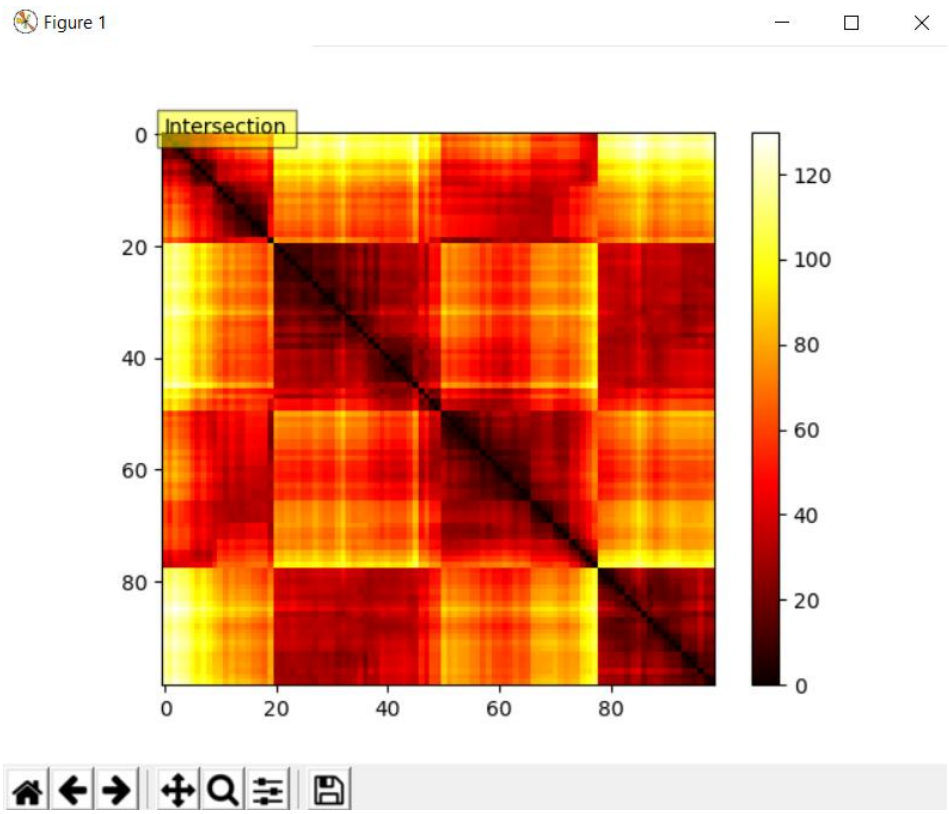




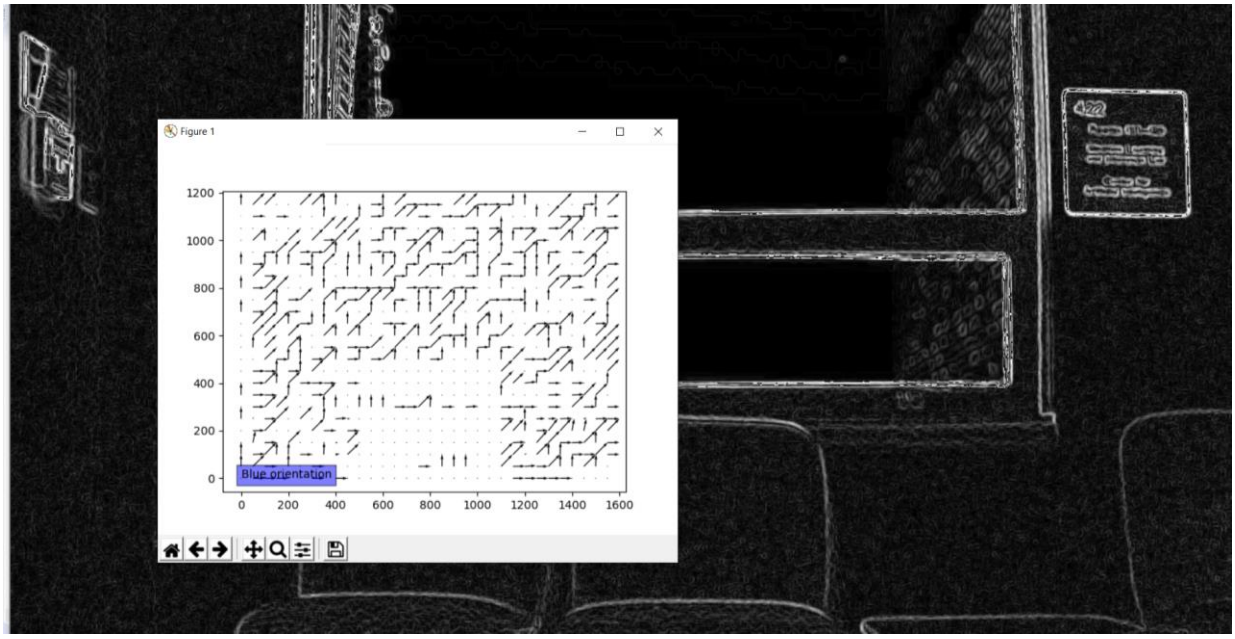
Gray chisquare:

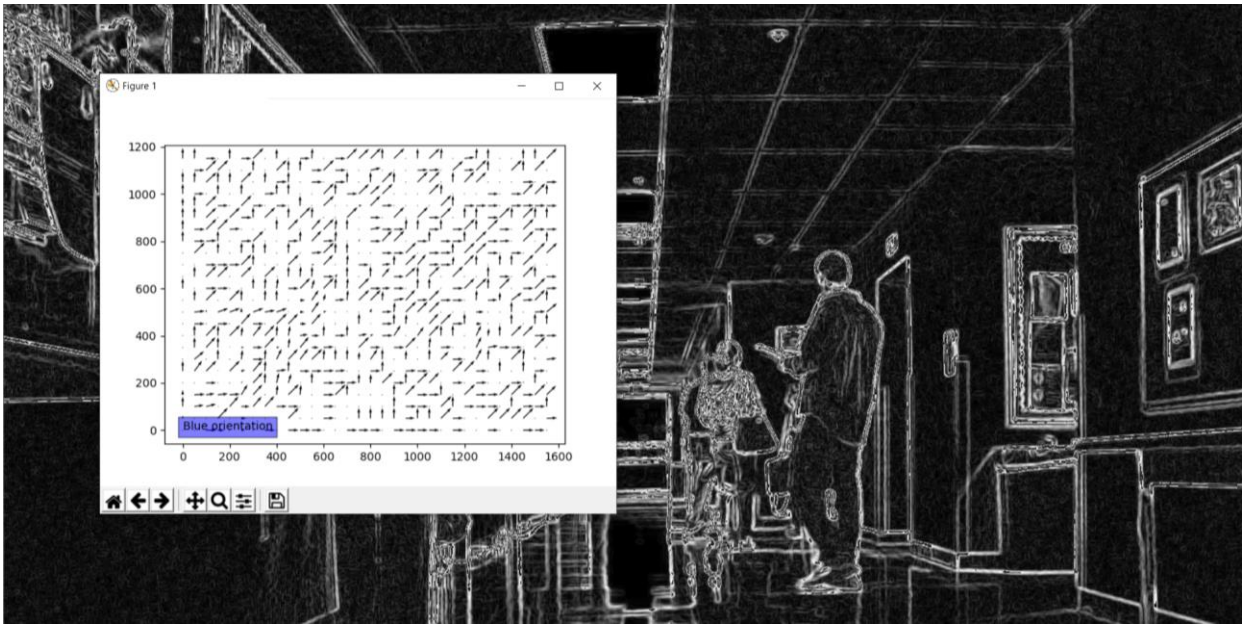


Gray intersection:

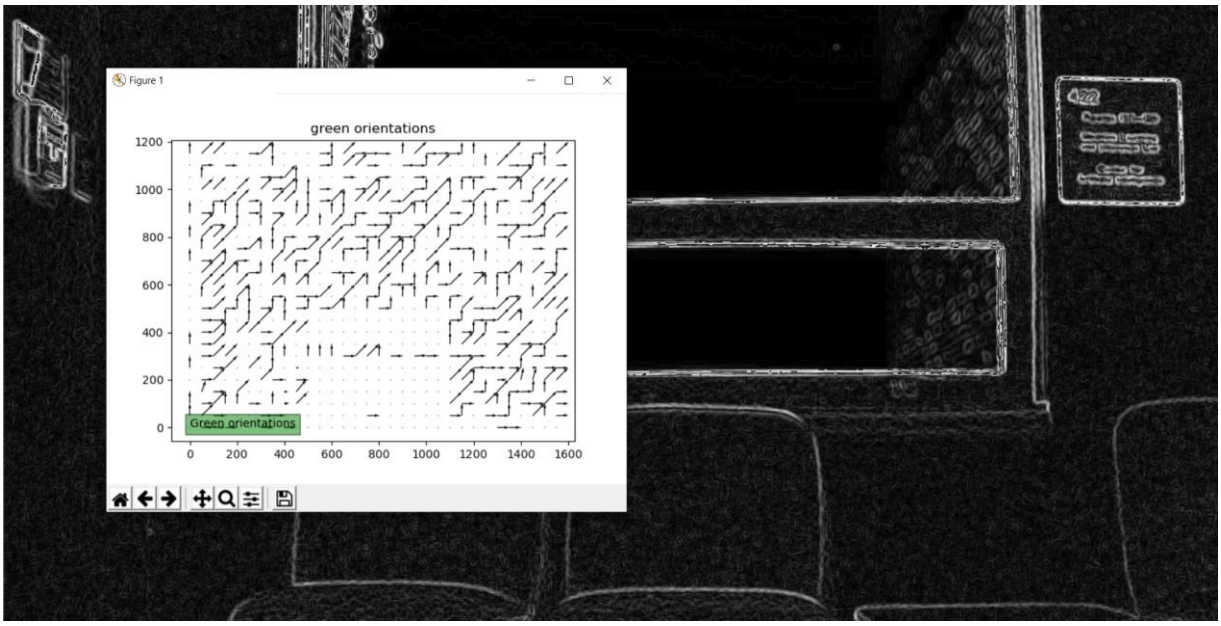


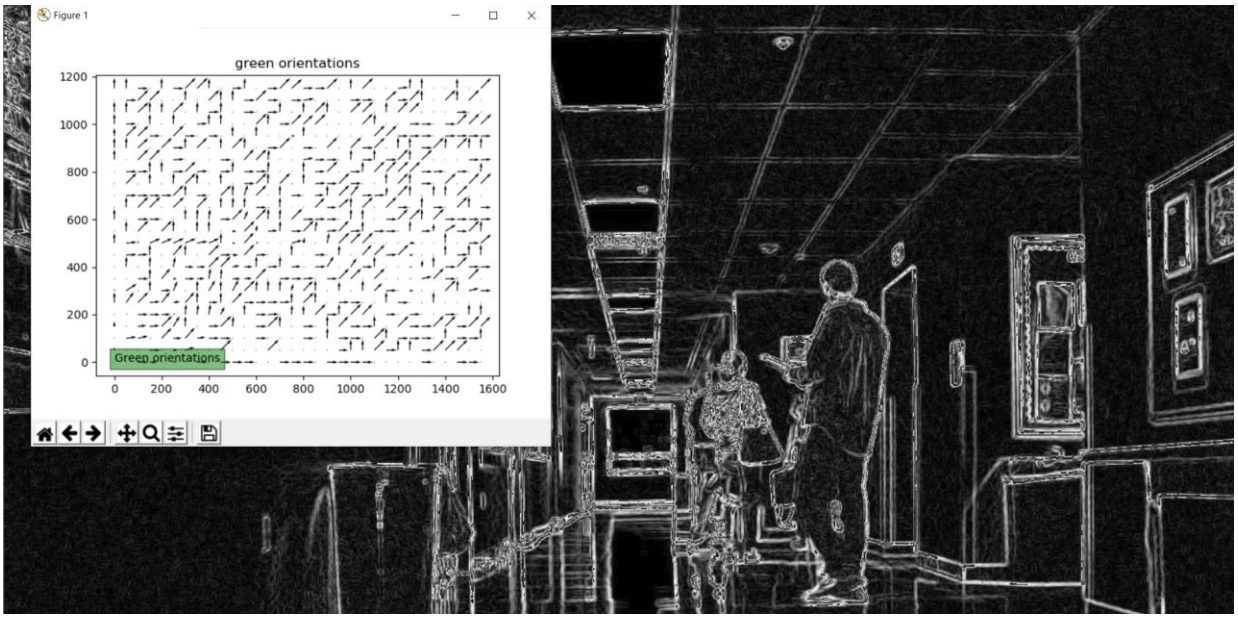
Blue orientation:



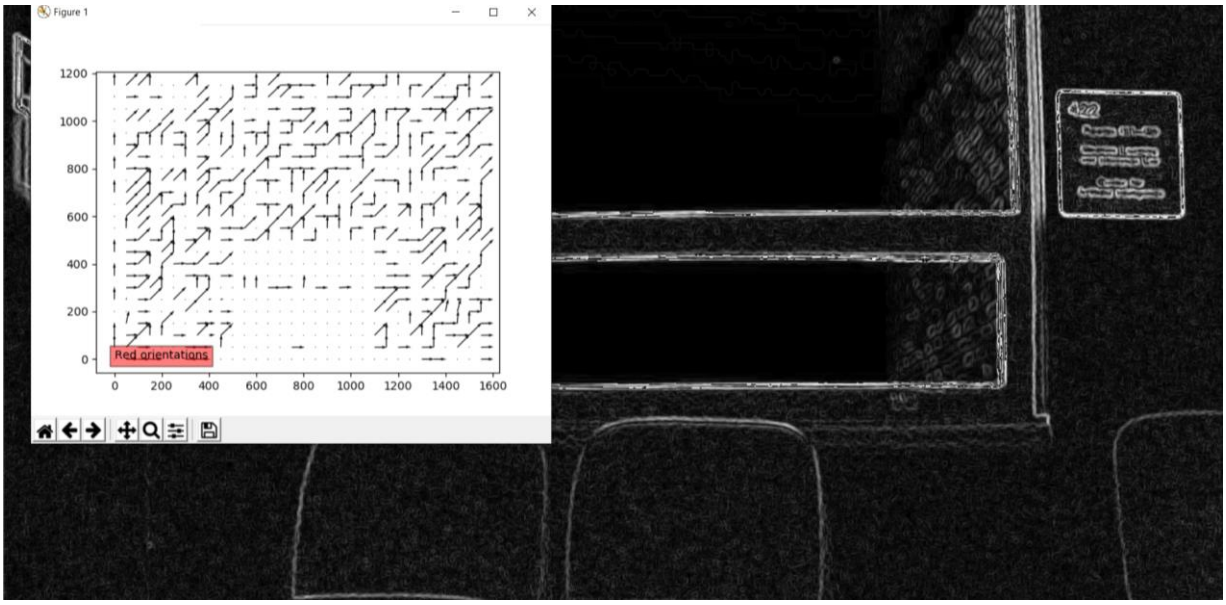


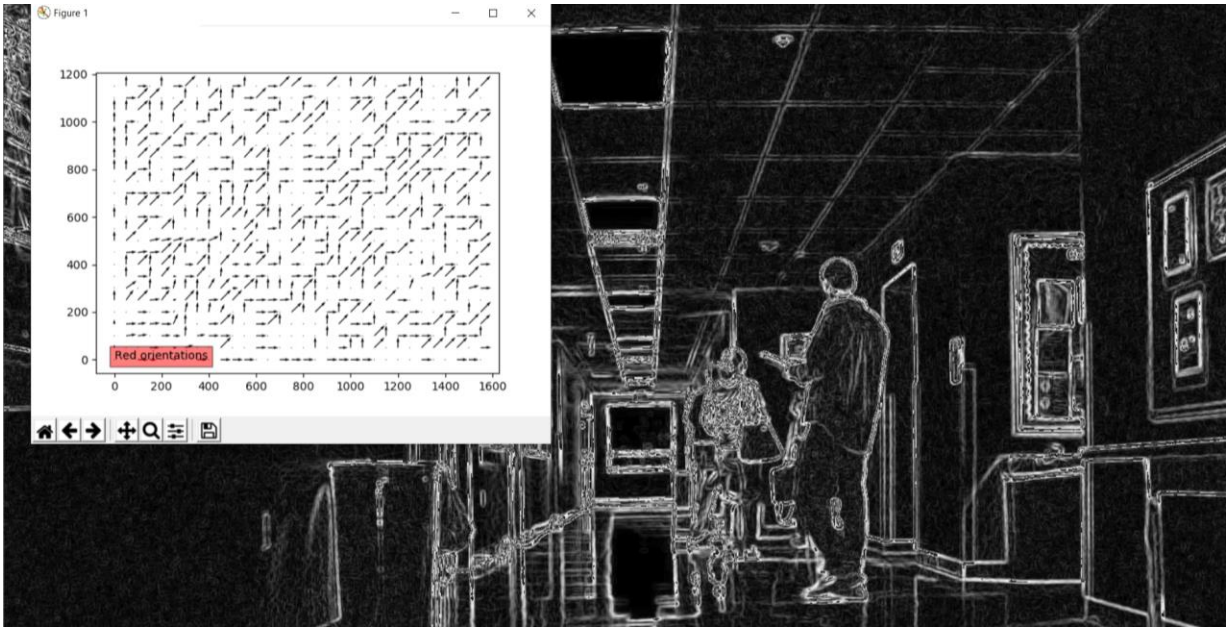
Green orientation:



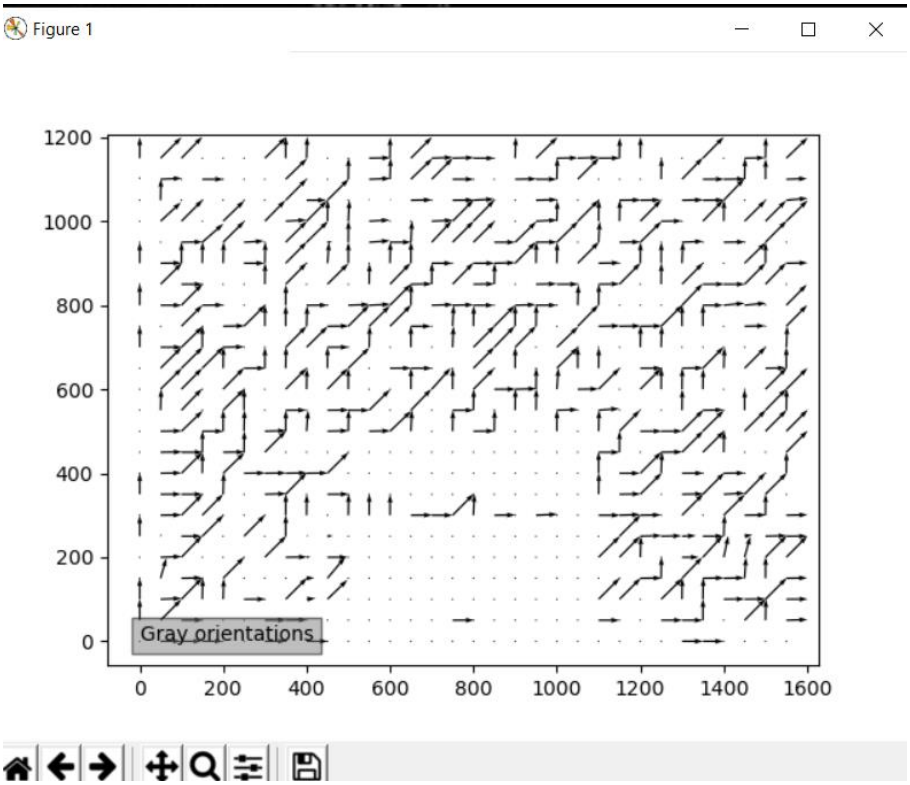


Red orientation:

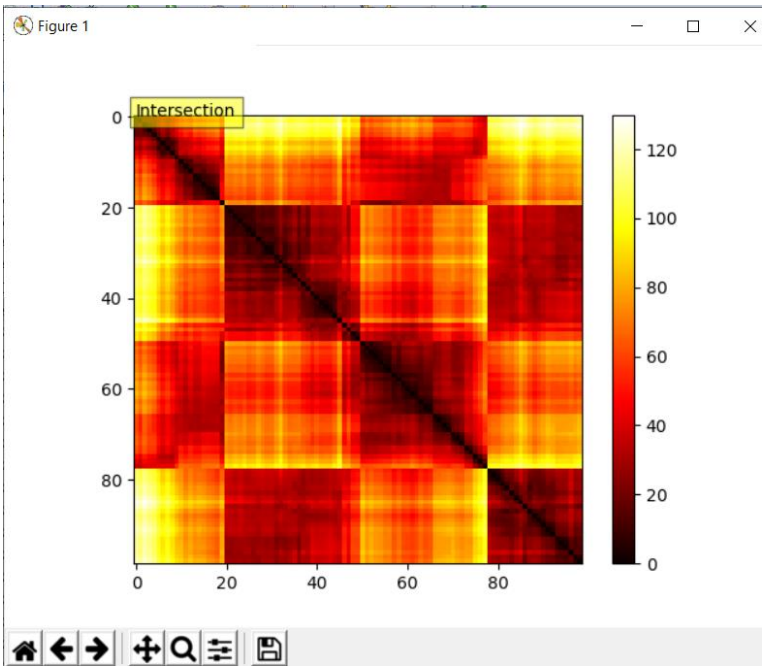


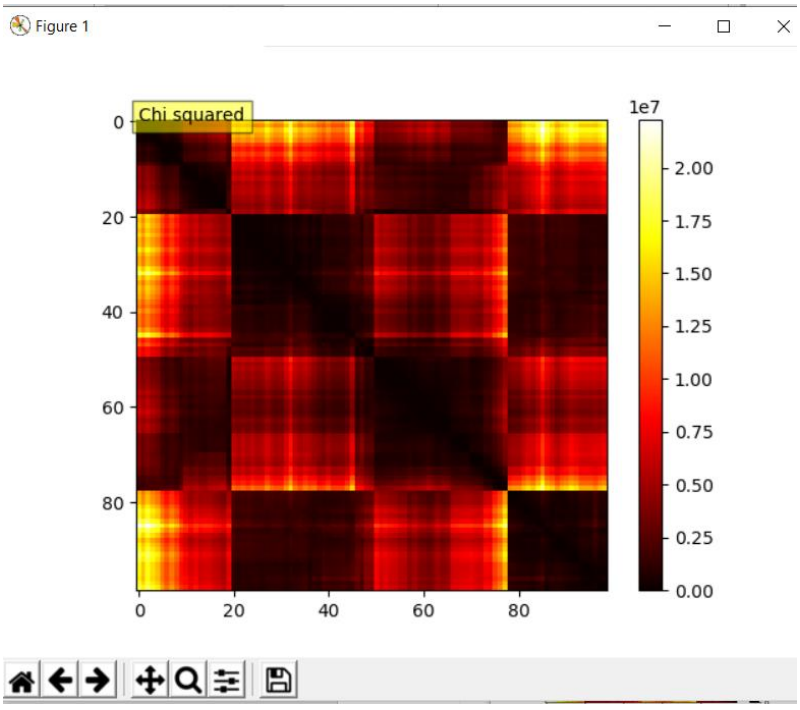


Gray orientation

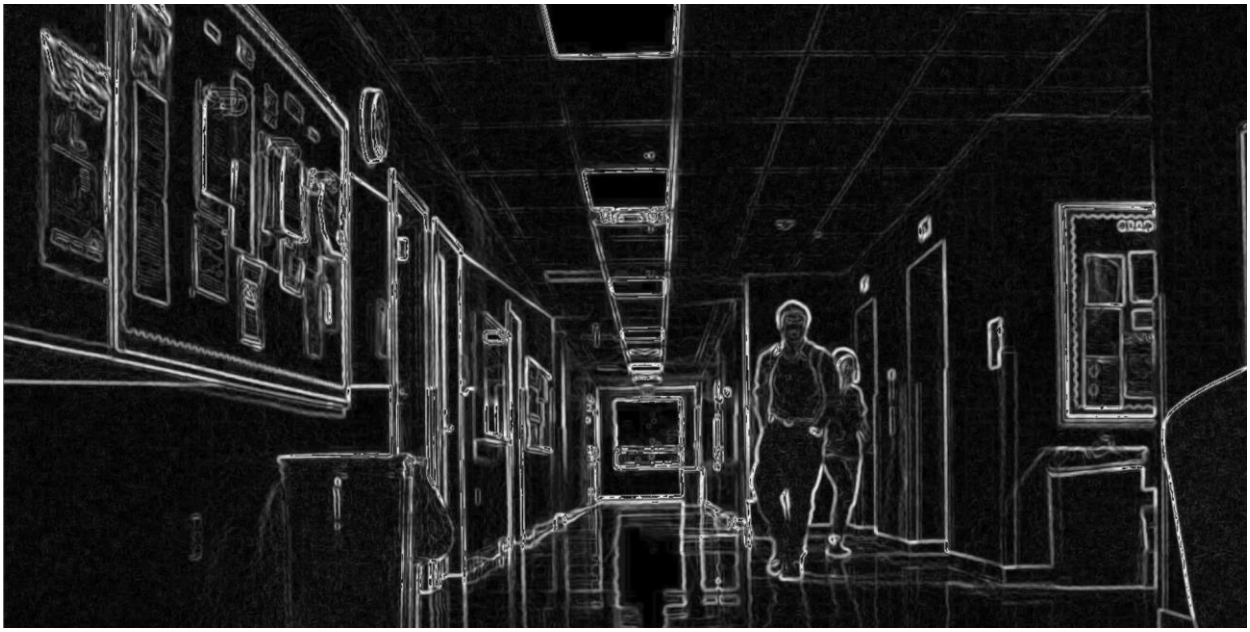


Color intersection and chi squared:

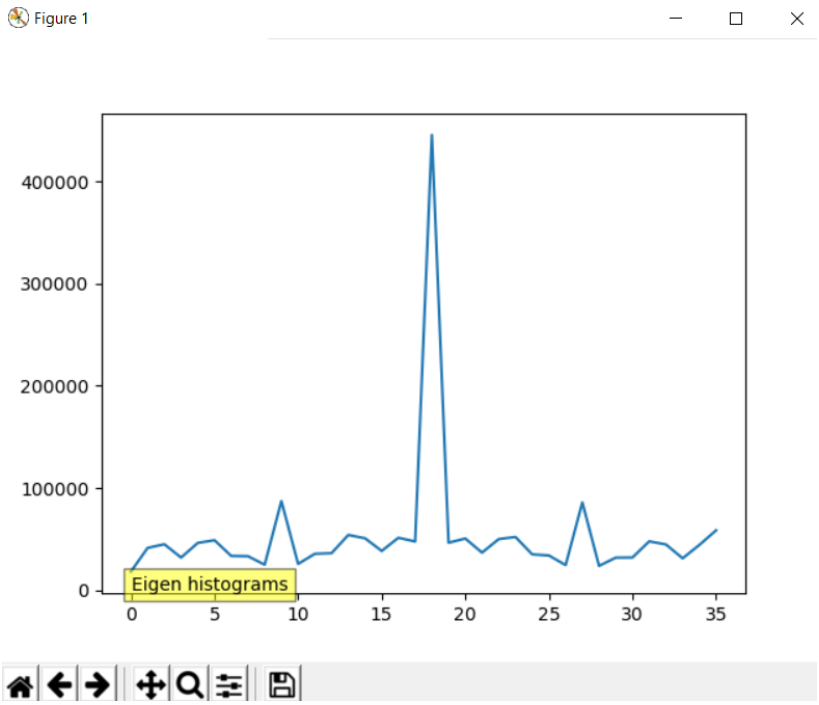


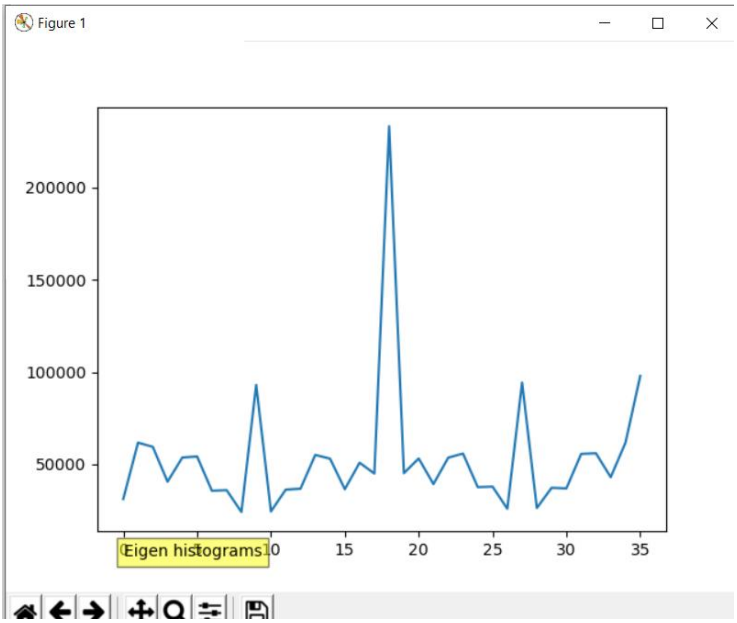


Eigen edges:



Eigen histogram:





Eigen intersection and chisquared