COMPUTER VISION
Harris Corner Detector Algorithm
Assignment 1

Contribution by Azka rehman:

Wrote code for question 1

Suggested logic for question 2

Wrote code for Q3

Compiled report 30 percent

Contribution by Mushkbar Fatima:

Wrote code for question 2

Found trends in threshold and repeatability

Compiled 70 percent report

Submitted By:

Azka Rehman

arehman.ee38ceme

174227

Mushkbar Fatima

fmushkabr.ee38ceme

175540

TABLE OF CONTENTS

Used Language: *Python*
Used Libraries:

i. *numpy*
ii. *cv2*
iii. *matplotlib*
iv. *math*
v. *PIL*
vi. *imutils*

PROBLEM 1:

Description:

Implement Harris Detector

Discussion:

Program starts by importing libraries and images. Then all the functions are defined first. These defined functions include the main functions addressing the problems as well as side functions supporting the main functions.
In the first problem, Harris function takes an image as input and performs following function:

- Applying horizontal and vertical Sobel detector on image.
- Sobel detector gave Ix and Iy as output.

$$M = \sum w(x,y) \begin{bmatrix} {I_x}^2 & I_x I_y \\ I_x I_y & {I_y}^2 \end{bmatrix}$$

-
- Using numpy library Ixx=Ix*Ix, Iyy= Iy*Iy and Ixy=Ix*Iy were computed.
- Aggregation/averaging is done by applying gaussian filter on each variable separately i.e. Ixx,Iyy,Ixy were averaged.
- The results after averaging were then used to compute R denoted by C in code. Formula to calculate R is as follows:

$$R = \det M - k(trace\, M)^2$$

$$\det M = \lambda_1 \lambda_2$$

$$trace\, M = \lambda_1 + \lambda_2$$

-
- The function then prints image with [x] marks indicating the corner points.
- The function returns the Harris response of image.

NMS

- Non-maximum suppression using a 3x3 mask is then applied to the interest point strength and a threshold is used to select strong interest points.

- After adjusting threshold to optimum value, values of corner points greater than threshold are taken only in an array C. This array will give detected feature points in an image.
- The element of an array C is compared with its neighborhood elements, if this specific element is greater than neighbor elements then value 1 is assigned to this specific element otherwise zero.
- This process gives precise result of detected feature points mainly on corners.
- But detected feature points majorly depend on threshold. If threshold is much smaller, more weaker feature points would be detecting or if we increase it further weaker feature point will eliminate, and only stronger feature points would detect at high threshold value.
- So, we must use hit and trial method for choosing threshold value.

Problems Faced:

Problems were faced in predicting the right threshold which will give optimum results.

CODE LISTING:

Below is Harris Detector Function.

```
#defining Harris detector function
def Harris(img):
        gray1 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        gray = np.float32(gray1)
        ix = cv2.Sobel(gray,cv2.CV_64F,1,0,ksize=3)
        iy = cv2.Sobel(gray,cv2.CV_64F,0,1,ksize=3)
        ixx=np.multiply(ix,ix)
        iyy=np.multiply(iy,iy)
        ixy=np.multiply(ix,iy)
        average_ixx = cv2.GaussianBlur(ixx,(3,3),0)
        average_iyy=cv2.GaussianBlur(iyy,(3,3),0)
        average_ixy=cv2.GaussianBlur(ixy,(3,3),0)
        det=average_ixx*average_iyy-(average_ixy)**2
        trace=average_ixx+average_iyy
        C3=det-0.04*(trace)**2
        smallest = np.amin(C3)
        biggest = np.amax(C3)
        C=abs(C3)
        #threshold=((np.absolute(smallest)+biggest)/90)
        #threshold=1554478150
        threshold=(0.01* biggest)
        #checked above thresholds
        #results are shit
        C_thresh=C>threshold
        for k in range(0,len(C)):
            for l in range (0,len(C[0])):
                if C[k][l]< threshold:
                    C[k][l]=0
```

```
A=np.zeros((len(C),len(C[0])),dtype=int)
        for m in range(3,len(C)-4):
            for n in range (3,len(C[0])-4):
                if C[m][n]>=threshold:
                    N=np.array([[C[m-2][n-2],C[m-2][n-1],C[m-2][n],C[m-2][n+1],C[m-2][n+2]],
                                [C[m-1][n-2],C[m-1][n-1],C[m-1][n],C[m-1][n+1],C[m-1][n+2]],
                                [C[m][n-2],C[m][n-1],0,C[m][n+1],C[m][n+2]],
                                [C[m+1][n-2],C[m+1][n-1],C[m+1][n],C[m+1][n+1],C[m+1][n+2]],
                                [C[m+2][n-2],C[m+2][n-1],C[m+2][n],C[m+2][n+1],C[m+2][n+2]]])

                    max_value=np.amax(N)
                    if C[m][n]>=max_value:
                        A[m][n]=1
                    else:
                        A[m][n]=0
        m1=np.count_nonzero(A)
        print("Number of Corner points",m1)
        A1 = np.float32(A)
        img2 = np.zeros_like(img)
        img2[:,:,0] = gray
        img2[:,:,1] = gray
        img2[:,:,2] = gray

        plt.imshow(gray,cmap='gray')
        for i in range (0,len(A)):
            for j in range (0,len(A[0])):
                if A[i][j]==True :
                    plt.scatter(j,i,s=1, c='red', marker='x') #for gray image
                    #img2[i][j]=[255,0,0] #for colored image
                    #this method is used before nms to show precise points on image


        plt.show()
        #plt.imshow(img2)
        return A1
```

PROBLEM 2:

Description:

Prove Rotation Invariant property.

Discussion:

Repeatability after rotation was plotted using python code.
Program implemented to solve this problem includes following functions:

Repeatability

- In repeatability function, a rotated angle, image array and its detected feature points array after Harris detector is passed
- First, an original image array is passed through Harris function which results in detected feature points array and then it is passed through rotation. After this, first array is compared with another second array which was first passed through rotation and then Harris function.
- First array consists of few key points while the second array consists of many key points because second array was a result of rotation and then Harris detector. Rotated image usually consists of more key points because in place of real corners, black area comes due to sudden transition of intensity, many more feature points detect as a result.

- After this, both array's key points are compared in a Matching point function where Euclidian distance was also measured between the same features of both arrays.
- Matching feature points were then placed in an array M.
- Rep was measured by dividing M by N. M was matched feature points and N was possibility of matching.

$$N = \frac{correspondence}{possible\ correspondence/\min(both\ arrays)}$$

- Then repeatability against rotation was plotted.

Rotation_invariant_property()(main function):
It rotates images from 15 degrees to 360 degrees and perform Harris detection on them iteratively. It also computes corresponding rotated points of original images. Finally, it checks whether the points of original image and rotated image matches or not. It returns M, N and repeatability arrays as outputs.

matching_points():
It gives number of matching points of two images passed.
It takes two images as input, save their true points in four arrays i.e. X1, Y1, X2, Y2 and compute their distance. If the distance is less than 8 pixels it adds one to the M which indicates number of matching points. It then returns the number of matching points M, number of feature points in original image N, repeatability array rep=M/N and array of angles.

Rotation():
Rotates an image on certain angle.

Plot():
This function plots the repeatability.

Problems Faced:

Problems were faced in rotation of Harris key points of original image. Rotation function did not give the exact rotated coordinates where Harris key points were. It's not much efficient.

CODE LISTING:

Rotation_invariant_property()(main function):

```
def Rotation_invariant_property(img,orgimg):
    M=np.zeros(24)
    rep=np.zeros(24)
    N=np.zeros(24)
    a=np.zeros(24)
    l=0
    i=0
    for k in range (15,361,15):
        org=rotation(orgimg,k)
        N2=np.count_nonzero(org)
        rot=rotation(img,k)
        (H_R)=Harris(rot)
        N3=np.count_nonzero(H_R)
        p=matching_points(org,H_R)
        M[l]=p
        N[l]=min(N2,N3)
        rep[l]=M[l]/N[l]
        l+=1
        if(k<361):
            a[i]=k
            i+=1
    return M,rep,N,a
```

matching_points():

```
def matching_points(img,r):

        M=0
        X1=np.zeros((len(r),len(r[0])))
        D=np.zeros((len(r),len(r[0])))
        Y1=np.zeros((len(r),len(r[0])))
        X2=np.zeros((len(r),len(r[0])))      # rotated
        Y2=np.zeros((len(r),len(r[0])))
        for i in range (0,len(img)) :
            for j in range (0,len(img[0])) :
                if img[i][j]>0:
                    X1[i][j]=i
                    Y1[i][j]=j
                else:
                    X1[i][j]=0
                    Y1[i][j]=0
        for i in range (0,len(r)) :
            for j in range (0,len(r[0])) :
                if r[i][j]>0:
                    X2[i][j]=i
                    Y2[i][j]=j
                else:
                    X2[i][j]=0
                    Y2[i][j]=0
        for i in range (0,len(r)) :
            for j in range (0,len(r[0])) :
                if X1[i][j]!=0 or Y1[i][j]!=0 or X2[i][j]!=0 or Y2[i][j]!=0:
                    f=norm(X1[i][j],Y1[i][j],X2[i][j],Y2[i][j])
                    if f<8:
                        D[i][j]=1
                        M+=1
                    else:
                        D[i][j]=0
        M1=np.count_nonzero(D)

        return M1
```

Rotation function:

```
def rotation(img,ang):
    rows = len(img)
    cols = len(img[0])
    M = cv2.getRotationMatrix2D((cols/2,rows/2),ang,1)
    dst = cv2.warpAffine(img,M,(cols,rows))
    return dst

 def norm(x01,y01,x02,y02):
    n = math.sqrt((abs(x02 - x01))**2 + (abs(y02 - y01))**2)
    return n

def plot(M,rep,N,a):
    print(M)
    print(rep)
    print(N)
    plt.plot(a, rep, color='yellow', linestyle='dashed', linewidth = 2,
         marker='o', markerfacecolor='blue', markersize=10)
    plt.xlabel('Angle or Scale')
    plt.ylabel('Repeatibility')
    plt.title('Graph')
    plt.show()
```

PROBLEM 3:

Description:

Prove Scale variant property.

Discussion:

Repeatability after scaling was plotted using python code.
Program implemented to solve this problem includes following functions:

scale_variant_property()(main function):

It rotates images from m^0 to m^8 iteratively and perform Harris detection on them. Finally, it checks whether the points of original image and rotated image matches or not. It returns M, N and repeatability arrays as outputs.

scaling():

It scales an image by a certain factor.

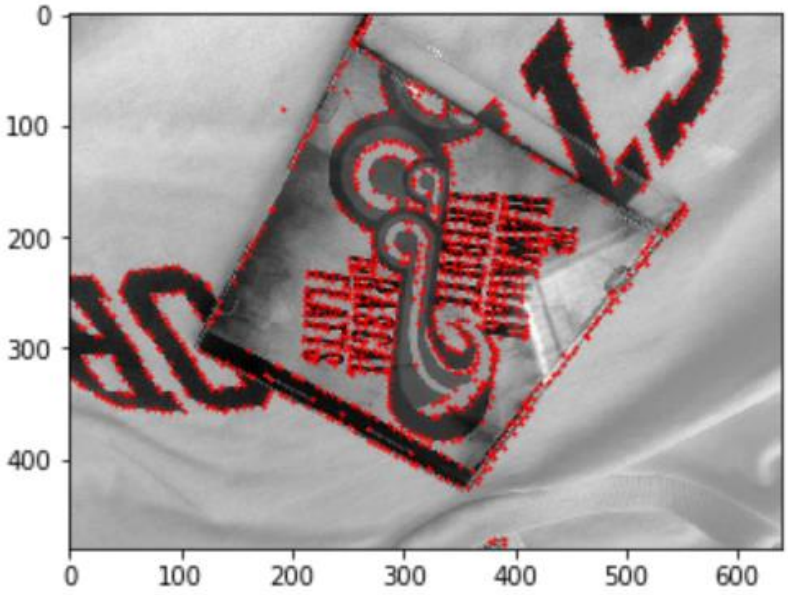
CODE LISTING:

```
def scaling(image,f):
    scl_factor = f
    w = int(image.shape[1] * scl_factor )
    h = int(image.shape[0] * scl_factor )
    dim = (w, h)
    rimg=resized = cv2.resize(image, dim, interpolation = cv2.INTER_CUBIC)
    return rimg

  def scale_variant_property(image):

    l=0
    M=np.zeros(9)
    rep=np.zeros(9)
    fac=np.zeros(9)
    N=np.zeros(9)
    for i in range (0,9):
        factor=1.2**(i)
        fac[l]=factor
        si=scaling(image,factor)
        sr=Harris(si)
        N3=np.count_nonzero(sr)
        if i==0:
            N2=np.count_nonzero(sr)
            himg=sr

        M[l]=matching_points(himg,sr)
        N[l]=min(N2,N3)
        rep[l]=M[l]/N[l]
        l+=1
    return M,N,rep,fac
```

RESULTS (Image 1):

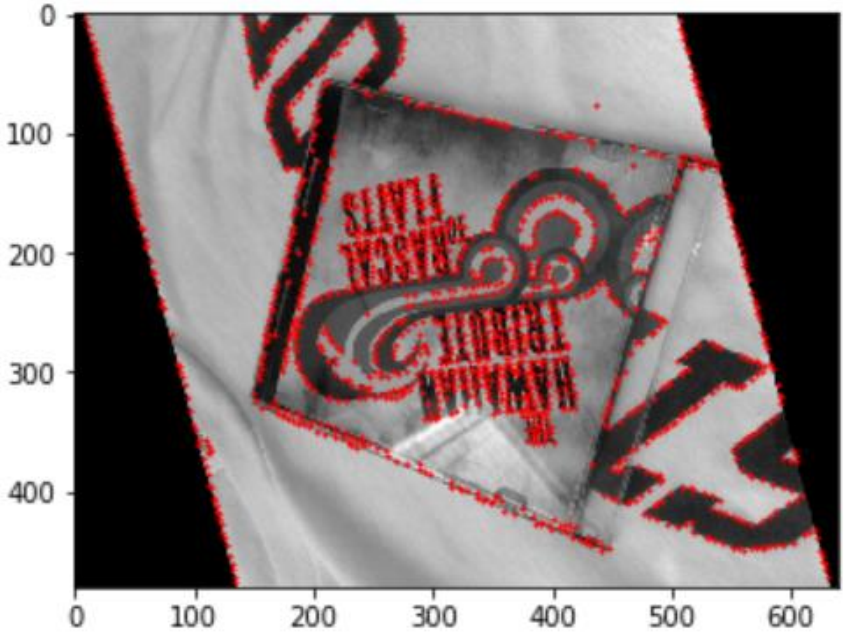
- ***Harris Response:***

```
Harris Response of Image 1
Number of Corner points 1212
```



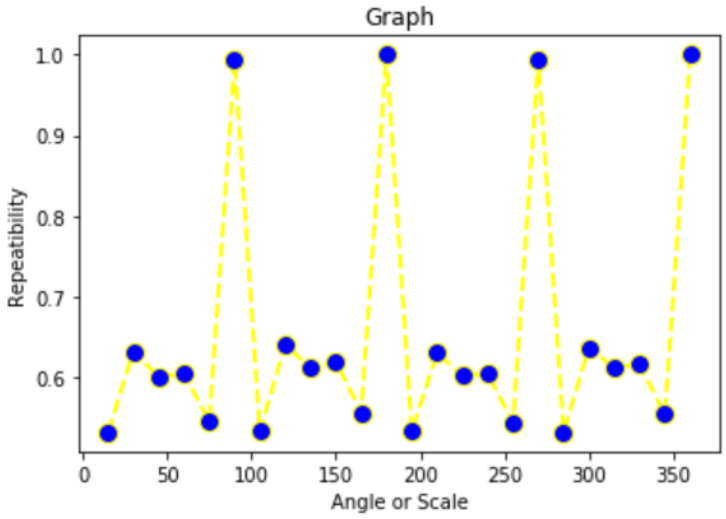
- ***Rotation performed(sample):***

```
Number of Corner points 1416
```



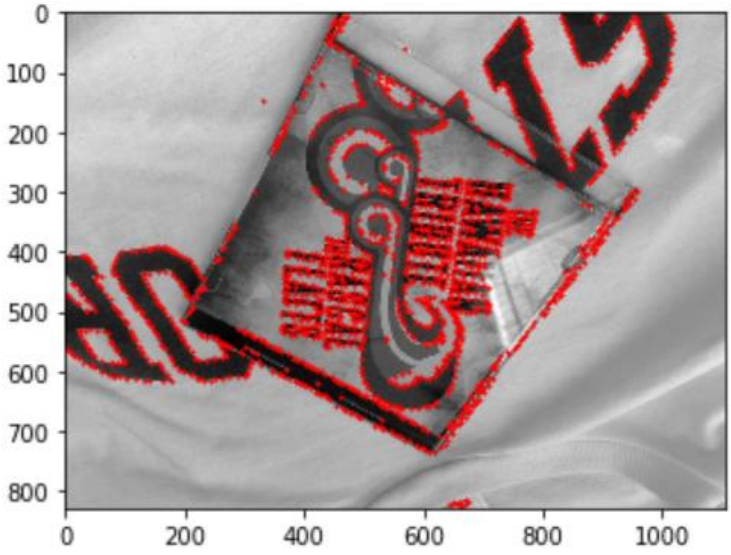
- ***Repeatability plot for rotation Image 1:***

```
[ 754.  712.  603.  582.  733. 1104.  753.  748.  715.  712.  821. 1207.
  752.  690.  598.  583.  732. 1104.  754.  735.  715.  703.  820. 1212.]
[0.53248588 0.63232682 0.60119641 0.6049896  0.545793   0.99280576
 0.53366407 0.64150943 0.61163388 0.62020906 0.55698779 1.
 0.53371185 0.63071298 0.60282258 0.60539979 0.54464286 0.99369937
 0.53248588 0.63746748 0.61215753 0.61720808 0.55480379 1.        ]
[1416. 1126. 1003.  962. 1343. 1112. 1411. 1166. 1169. 1148. 1474. 1207.
 1409. 1094.  992.  963. 1344. 1111. 1416. 1153. 1168. 1139. 1478. 1212.]
```



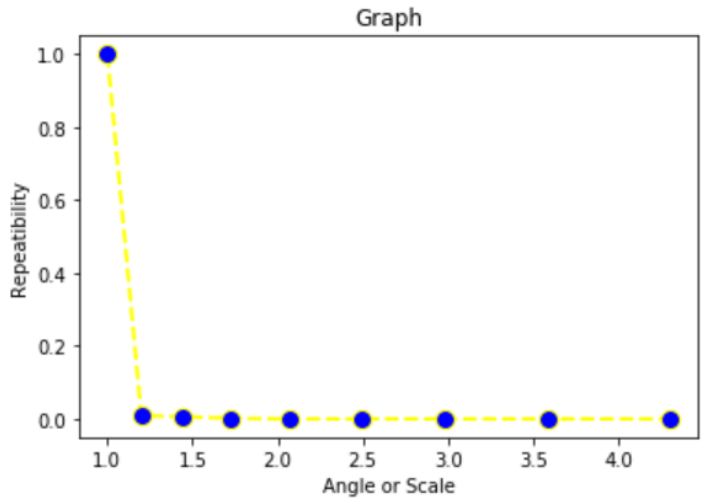
- ***Scaling performed(sample):***

```
Number of Corner points 1923
```



- ***Repeatability plot after Scaling image1:***

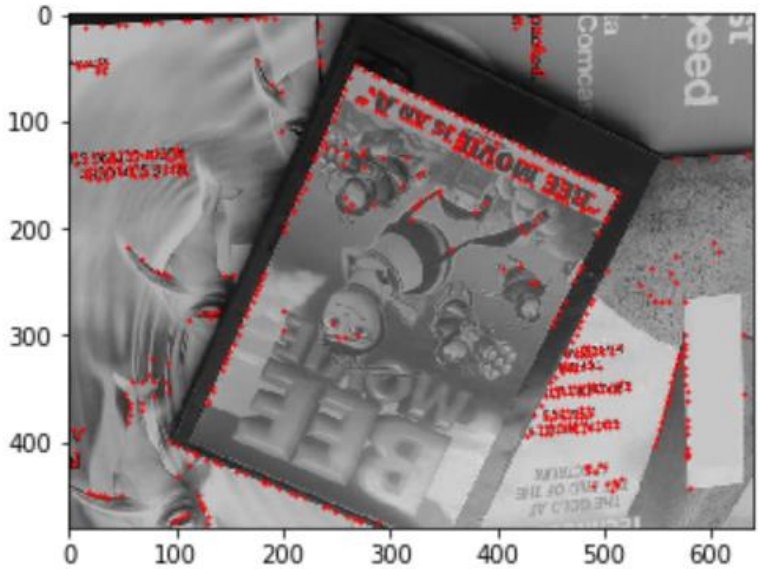
```
[1212.   11.    7.    2.    0.    0.    0.    0.    0.]
[1.         0.00907591 0.00577558 0.00165017 0.         0.
 0.         0.         0.        ]
[1212. 1212. 1212. 1212. 1212. 1212. 1212. 1212. 1212.]
```



RESULTS (Image 2):

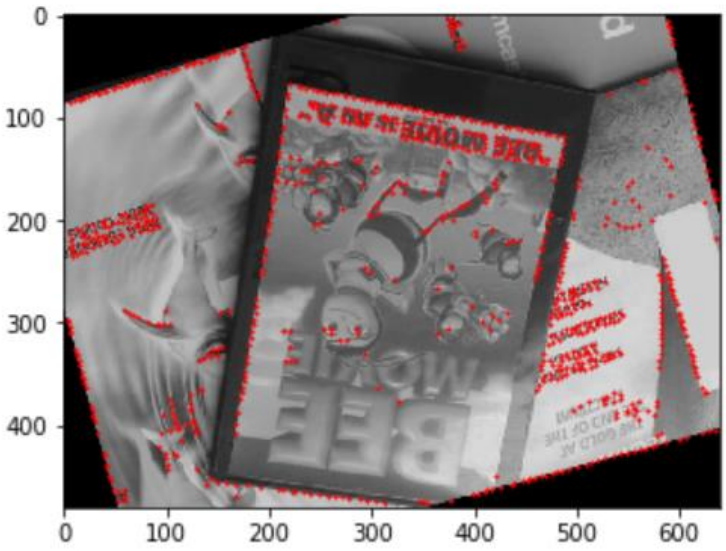
- ***Harris Response:***

```
Harris Response of Image 2
Number of Corner points 624
```



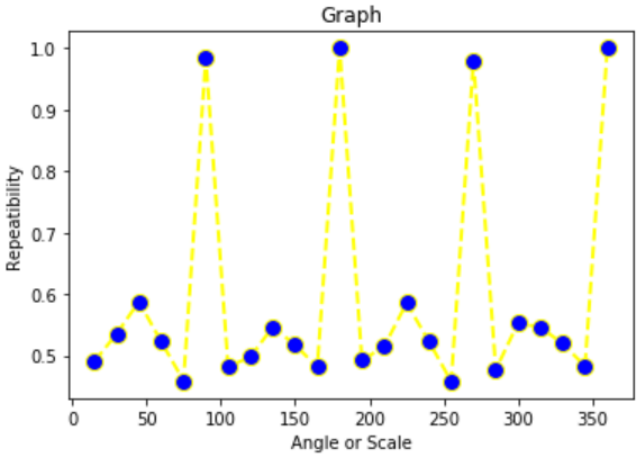
- ***Rotation performed(sample):***

```
Rotation Invariant Property >> Image2
Number of Corner points 893
```



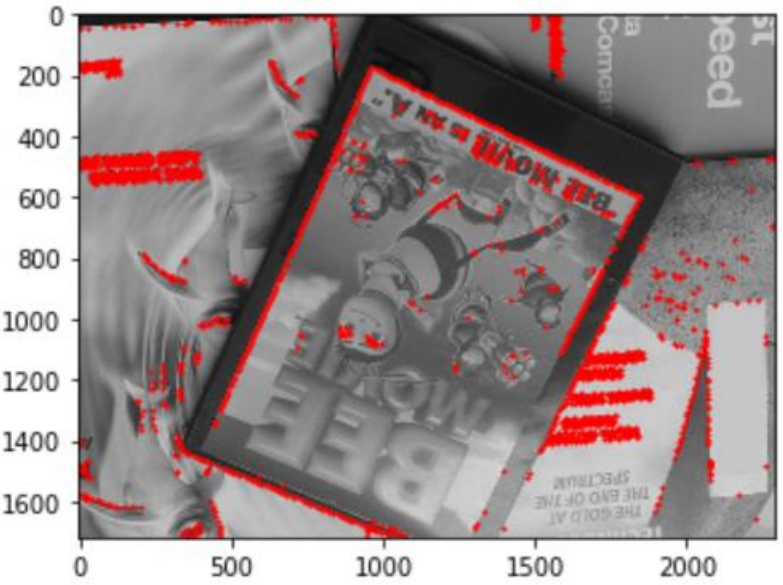
- ***Repeatability plot after Rotation of image2:***

```
[438. 394. 412. 402. 388. 482. 379. 346. 364. 396. 436. 613. 439. 395.
 411. 403. 388. 481. 374. 343. 363. 398. 434. 624.]
[0.49048152 0.53532609 0.58773181 0.52275683 0.4591716  0.98367347
 0.4821883  0.49784173 0.54654655 0.51764706 0.4812362  1.
 0.49215247 0.51499348 0.58547009 0.52337662 0.45808737 0.9796334
 0.47704082 0.55322581 0.54586466 0.51958225 0.48275862 1.        ]
[893. 736. 701. 769. 845. 490. 786. 695. 666. 765. 906. 613. 892. 767.
 702. 770. 847. 491. 784. 620. 665. 766. 899. 624.]
```



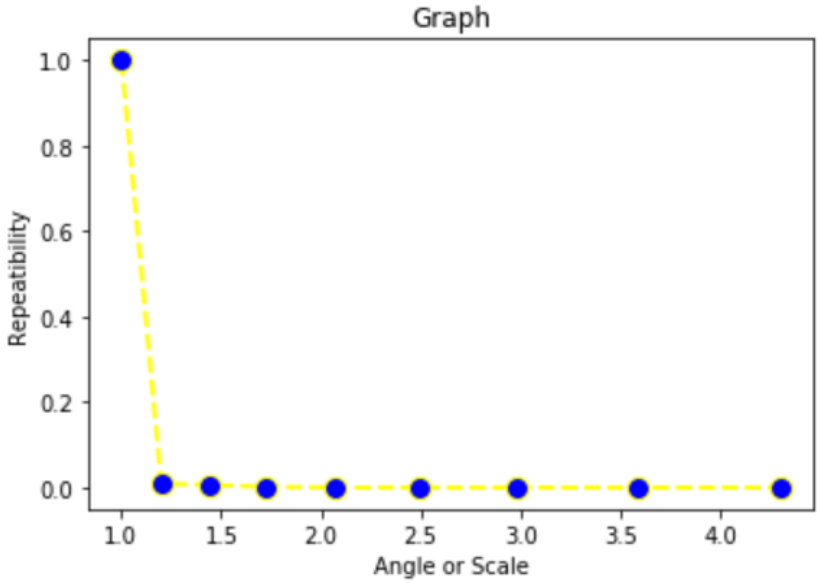
- ***Harris response after Scaling of image 2(sample):***

```
Number of Corner points 2228
```



- ***Repeatability plot after Scaling image2:***

```
[1212.   11.    7.    2.    0.    0.    0.    0.    0.]
[1.          0.00907591 0.00577558 0.00165017 0.          0.
 0.          0.         0.        ]
1212
```



OBSERVATIONS AND CONCLUSIONS:

As we decrease threshold, feature points after NMS increases and then matching points also increases at same level. It is good that matching points increases but increased feature points after NMS is not good. **So, there exist a tradeoff between these two values.** If we keep decreasing threshold, NMS will increase with a greater speed but at some limit our matching points will stop increasing. That's why we must adjust value of threshold by analyzing both above mentioned values.

- ***Robustness:***

  Harris detector is robust to angle variation of image. It is visible from the repeatability plot of graph that most feature points remain same after rotation.

  Harris detector is not robust to scale variation. This is because coordinates of Harris points change as we increase the size of image. It ca be seen from the graph that corner points decrease gradually as we increase the scale.