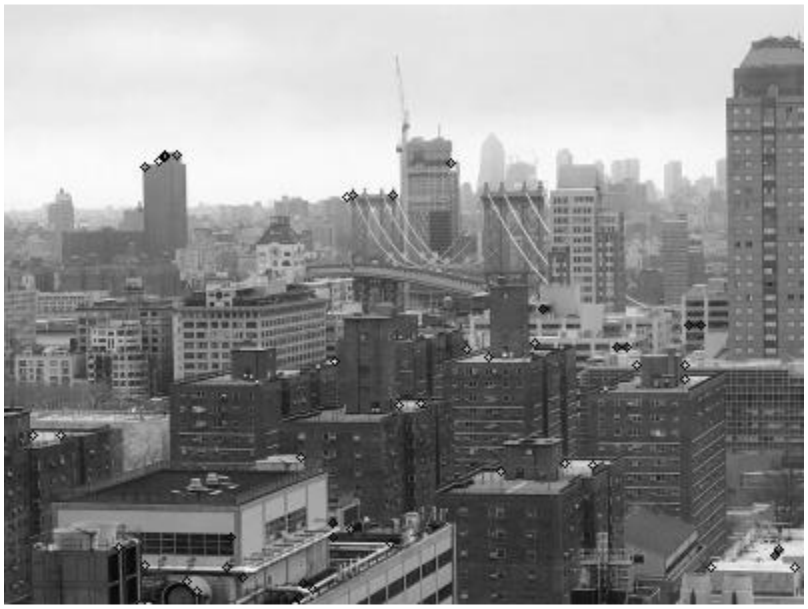
Exercise 1

A) Harris detector Implementation

```
img = np.float64(img)
n1,n2 = img.shape
#Gradients
Ix = signal.convolve2d(img, h1, boundary='symm', mode='same')
Iy = signal.convolve2d(img, h2, boundary='symm', mode='same')
Ixx = Ix*Ix
Iyy = Iy*Iy
Ixy = Ix*Iy
Ixx = signal.convolve2d(Ixx, h, boundary='symm', mode='same')
Iyy = signal.convolve2d(Iyy, h, boundary='symm', mode='same')
Ixy = signal.convolve2d(Ixy, h, boundary='symm', mode='same')
#Algorithm
A = np.zeros([2*n1,2*n2])
H = np.zeros([n1,n2])
final_image = np.zeros([n1,n2])
final_image1 = np.zeros([n1,n2])
temp = np.zeros([2,2])
temp1 = np.zeros([3,3])
alpha = 0.06
for i in range (0, n1):
    for j in range(0,n2):
        for k in range(0,2):
            for l in range(0,2):
                if k == 0 and l == 0:
                    A[2*i+k,2*j+l] = Ixx[i,j]
                    temp[k,l] = Ixx[i,j]
                elif k == 0 and l == 1:
                    A[2*i+k,2*j+l] = Ixy[i,j]
                    temp[k,l] = Ixy[i,j]
                elif k == 1 and l == 0:
                    A[2*i+k,2*j+l] = Ixy[i,j]
                    temp[k,l] = Ixy[i,j]
                elif k == 1 and l == 1:
                    A[2*i+k,2*j+l] = Iyy[i,j]
                    temp[k,l] = Iyy[i,j]
        H[i,j] = np.linalg.det(temp) - alpha*np.trace(temp*temp)
#end
max_im = np.max(H)
min_im = np.min(H)
H1 = np.round((H-min_im)*255/(max_im-min_im));
H = np.pad(H1,((1,1),(1,1)),'constant',constant_values=(0,0))
```

B) Sigma=1 and Features=50



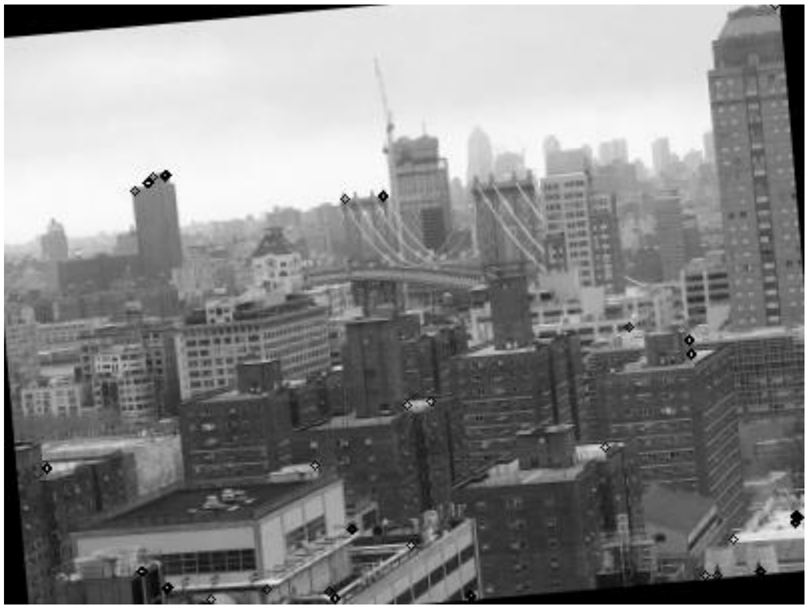
As corners are detected it makes sense.

C)

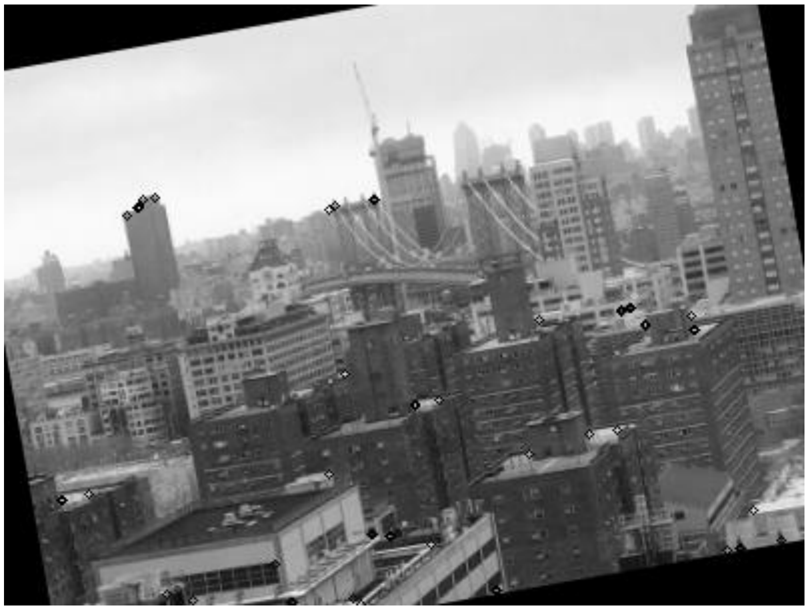
```
img_resize = cv2.resize(img, (0,0), fx=0.5, fy=0.5)

rows,cols = img.shape
M = cv2.getRotationMatrix2D((cols/2,rows/2),1,1)  #5 degree
dst = cv2.warpAffine(img,M,(cols,rows))
img_tilt=dst
```

D)

i) Rotate 5 degree

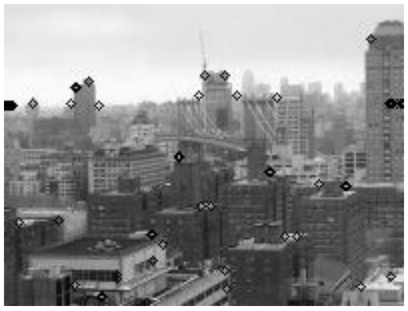


ii) Rotate 10 degree

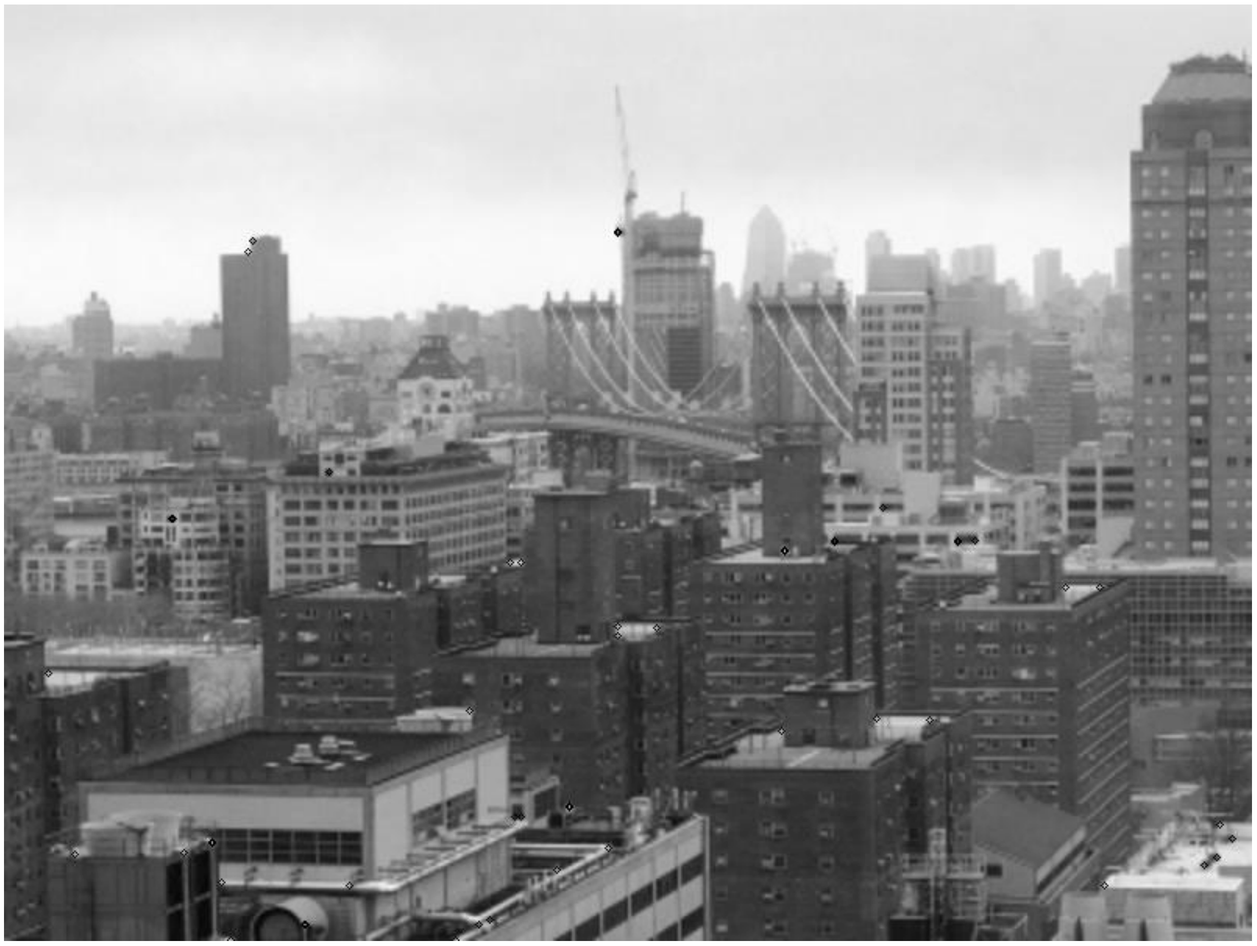


As the rotation increases the feature points decreases.

iii) Scale by half



iv) Scale doubled



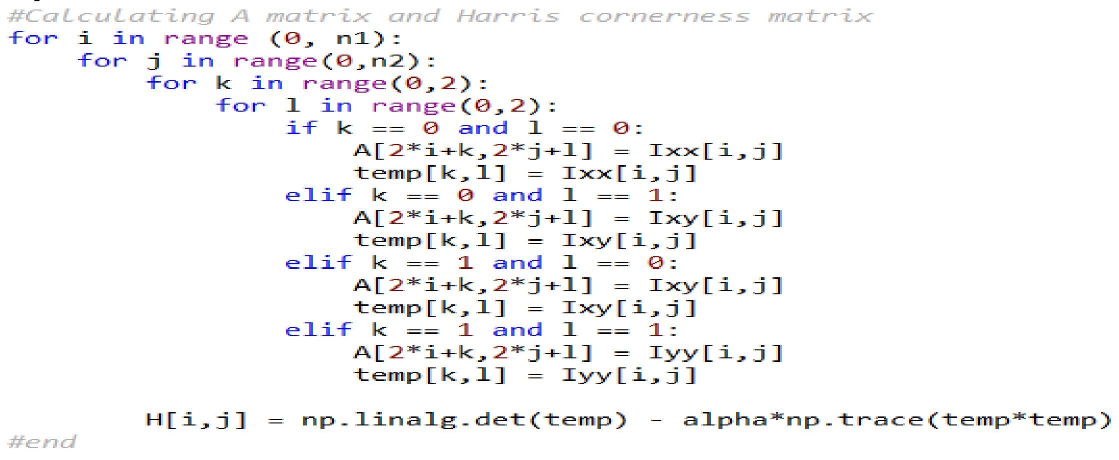
The Harris detector is invariant to scaling many feature points are lost

Exercise 2

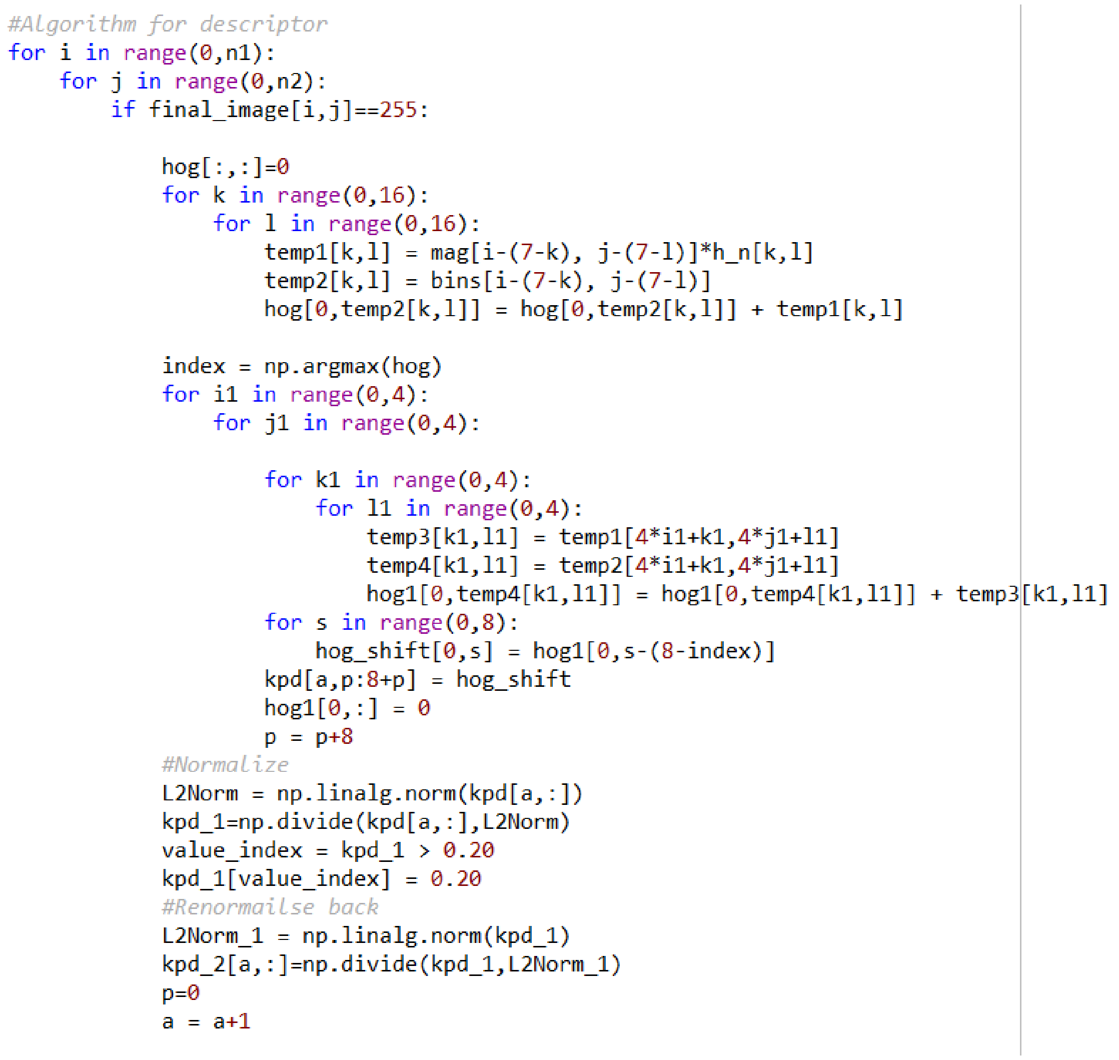
A) and B)

```python
#h= matlab_style_gauss2D(shape=(21,21),sigma=2)
h1=der_gauss(shape=(5,5),sigma=1)
h2=der_gauss_y(shape=(5,5),sigma=1)


img = cv2.imread('BK_left.JPG',0)
img = np.float64(img)
n1,n2 = img.shape

N = 50  #50 features
q = 45  # b bins of 45 degree

Ix = signal.convolve2d(img, h1, boundary='symm', mode='same')
Iy = signal.convolve2d(img, h2, boundary='symm', mode='same')

mag = np.zeros([n1,n2])
orient = np.zeros([n1,n2])
bins = np.zeros([n1,n2])
for i in range(0,n1):
    for j in range(0,n2):
        mag[i,j] = math.sqrt(Ix[i,j]**2 + Iy[i,j]**2)
        if Ix[i,j]==0:
            orient[i,j] = 90
        else:
            orient[i,j] = math.degrees(math.atan2(Iy[i,j],Ix[i,j]))

        orient[i,j] = (orient[i,j] + 360) % 360
        bins[i,j] = np.floor((orient[i,j]+(q/2))/q)
        if bins[i,j]==8:
            bins[i,j]=0
```

C)



```
#Calculating A matrix and Harris cornerness matrix
for i in range (0, n1):
    for j in range(0,n2):
        for k in range(0,2):
            for l in range(0,2):
                if k == 0 and l == 0:
                    A[2*i+k,2*j+l] = Ixx[i,j]
                    temp[k,l] = Ixx[i,j]
                elif k == 0 and l == 1:
                    A[2*i+k,2*j+l] = Ixy[i,j]
                    temp[k,l] = Ixy[i,j]
                elif k == 1 and l == 0:
                    A[2*i+k,2*j+l] = Ixy[i,j]
                    temp[k,l] = Ixy[i,j]
                elif k == 1 and l == 1:
                    A[2*i+k,2*j+l] = Iyy[i,j]
                    temp[k,l] = Iyy[i,j]

        H[i,j] = np.linalg.det(temp) - alpha*np.trace(temp*temp)
#end
```



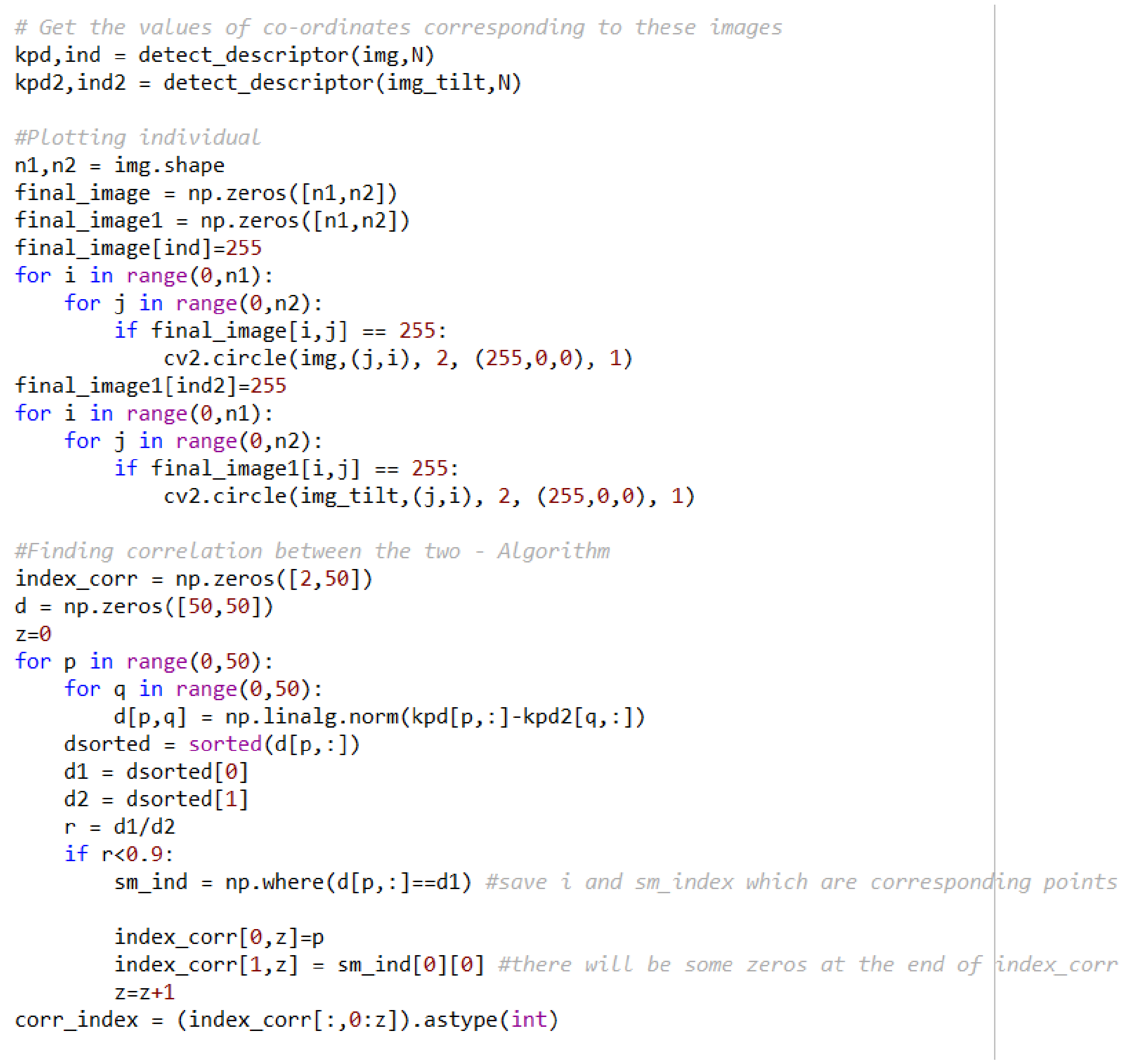
```
#Algorithm for descriptor
for i in range(0,n1):
    for j in range(0,n2):
        if final_image[i,j]==255:

            hog[:,:]=0
            for k in range(0,16):
                for l in range(0,16):
                    temp1[k,l] = mag[i-(7-k), j-(7-l)]*h_n[k,l]
                    temp2[k,l] = bins[i-(7-k), j-(7-l)]
                    hog[0,temp2[k,l]] = hog[0,temp2[k,l]] + temp1[k,l]

            index = np.argmax(hog)
            for i1 in range(0,4):
                for j1 in range(0,4):

                    for k1 in range(0,4):
                        for l1 in range(0,4):
                            temp3[k1,l1] = temp1[4*i1+k1,4*j1+l1]
                            temp4[k1,l1] = temp2[4*i1+k1,4*j1+l1]
                            hog1[0,temp4[k1,l1]] = hog1[0,temp4[k1,l1]] + temp3[k1,l1]
                    for s in range(0,8):
                        hog_shift[0,s] = hog1[0,s-(8-index)]
                    kpd[a,p:8+p] = hog_shift
                    hog1[0,:] = 0
                    p = p+8
            #Normalize
            L2Norm = np.linalg.norm(kpd[a,:])
            kpd_1=np.divide(kpd[a,:],L2Norm)
            value_index = kpd_1 > 0.20
            kpd_1[value_index] = 0.20
            #Renormailse back
            L2Norm_1 = np.linalg.norm(kpd_1)
            kpd_2[a,:]=np.divide(kpd_1,L2Norm_1)
            p=0
            a = a+1
```

Exercise 3

Using the results from exercise 1 and 2 and implementing the algorithm

Results:

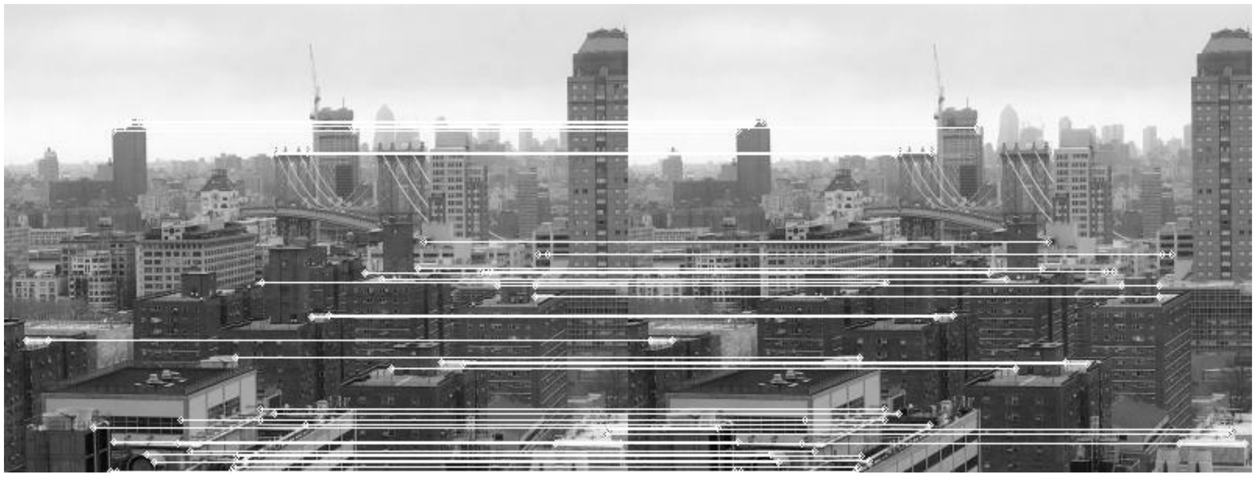
A) And B)



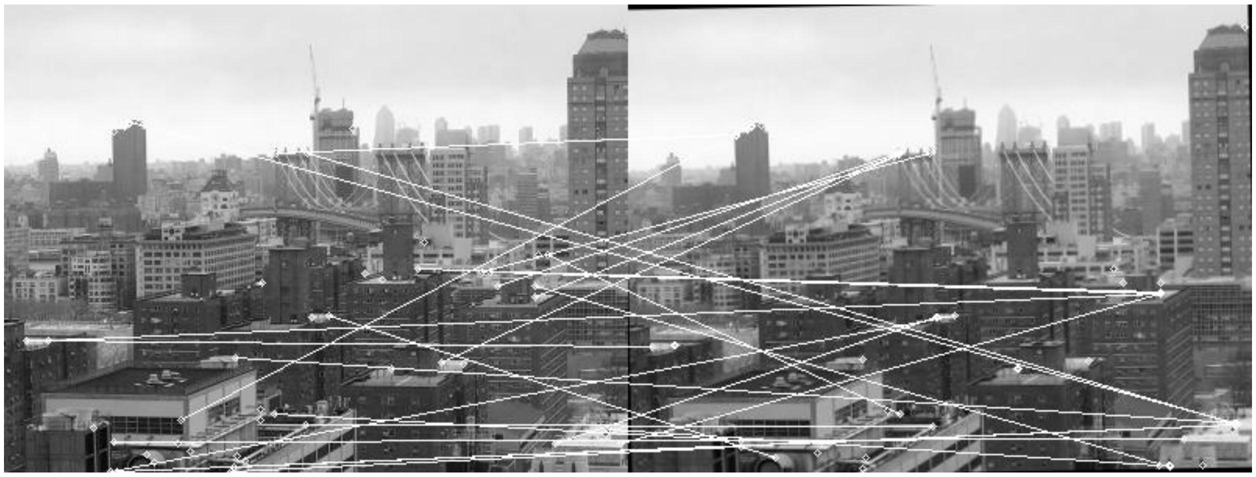
```python
# Get the values of co-ordinates corresponding to these images
kpd,ind = detect_descriptor(img,N)
kpd2,ind2 = detect_descriptor(img_tilt,N)

#Plotting individual
n1,n2 = img.shape
final_image = np.zeros([n1,n2])
final_image1 = np.zeros([n1,n2])
final_image[ind]=255
for i in range(0,n1):
    for j in range(0,n2):
        if final_image[i,j] == 255:
            cv2.circle(img,(j,i), 2, (255,0,0), 1)
final_image1[ind2]=255
for i in range(0,n1):
    for j in range(0,n2):
        if final_image1[i,j] == 255:
            cv2.circle(img_tilt,(j,i), 2, (255,0,0), 1)

#Finding correlation between the two - Algorithm
index_corr = np.zeros([2,50])
d = np.zeros([50,50])
z=0
for p in range(0,50):
    for q in range(0,50):
        d[p,q] = np.linalg.norm(kpd[p,:]-kpd2[q,:])
    dsorted = sorted(d[p,:])
    d1 = dsorted[0]
    d2 = dsorted[1]
    r = d1/d2
    if r<0.9:
        sm_ind = np.where(d[p,:]==d1) #save i and sm_index which are corresponding points

        index_corr[0,z]=p
        index_corr[1,z] = sm_ind[0][0] #there will be some zeros at the end of index_corr
        z=z+1
corr_index = (index_corr[:,0:z]).astype(int)
```

C) As the *degree* increases the mappings are *not satisfactory*

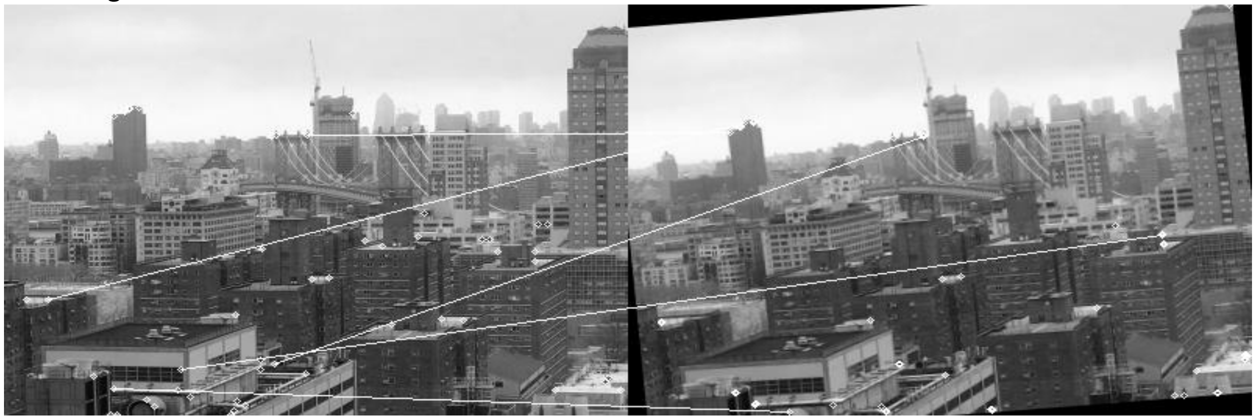
Zero degree r=0.9



One degree r=0.9



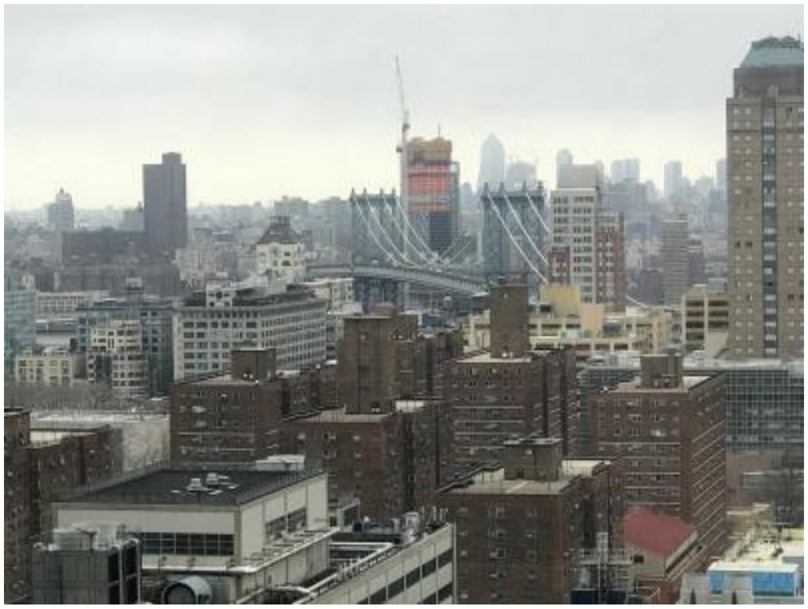
Five degree r=0.5



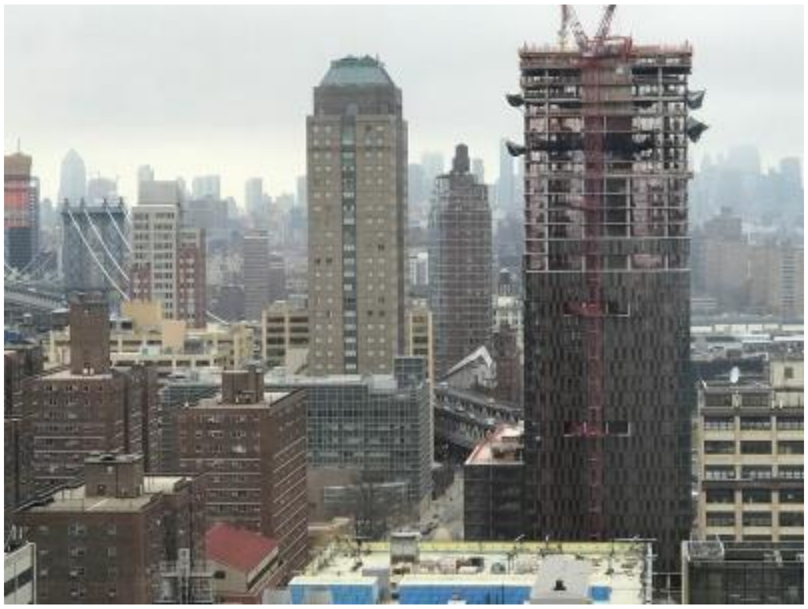
As the *degree* increases the mappings are *not satisfactory*. So it does not look reasonable.

Exercise 4

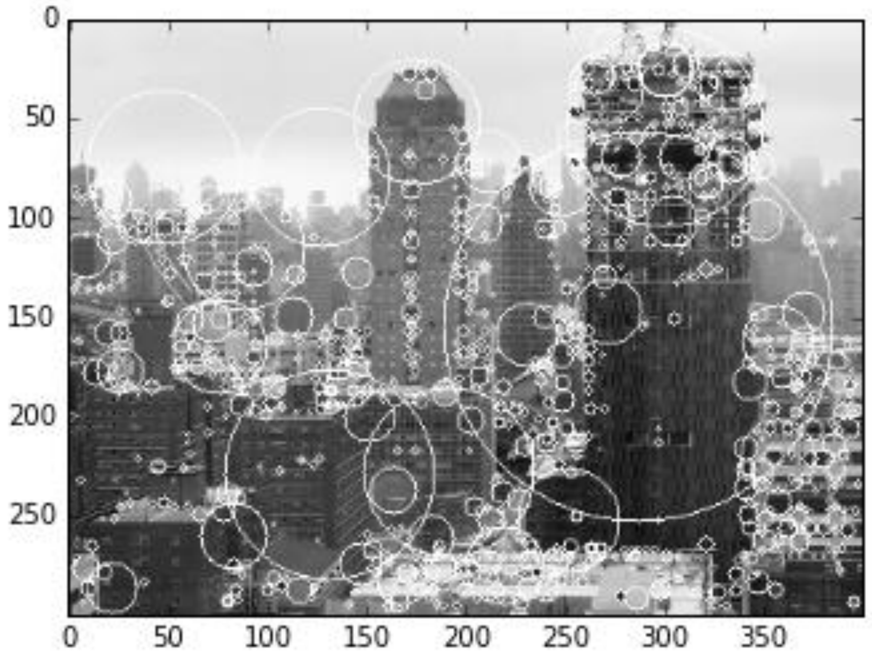
Left Image:



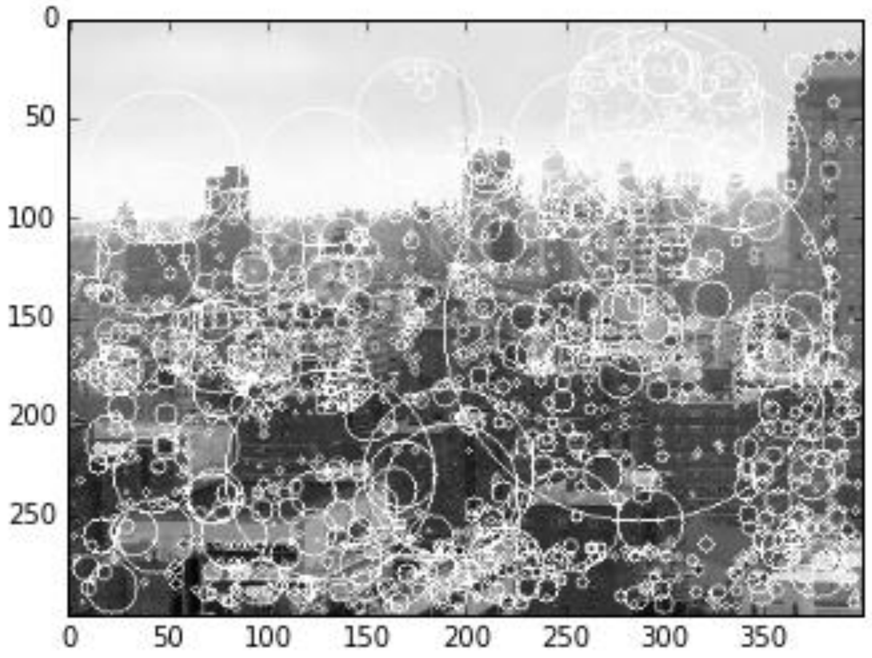
Right Image:



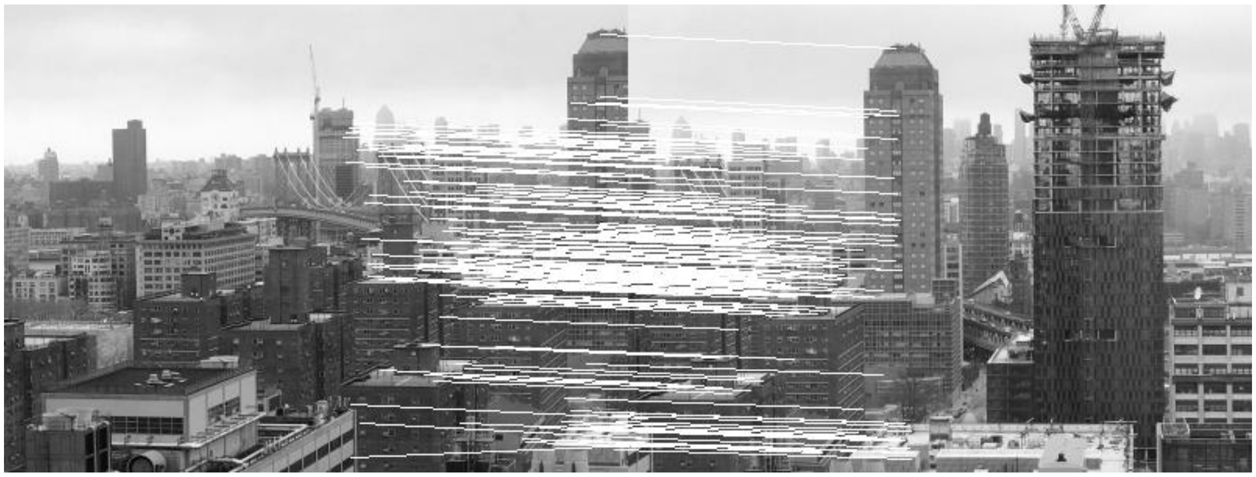
Right Image with SIFT points with scale as radius:



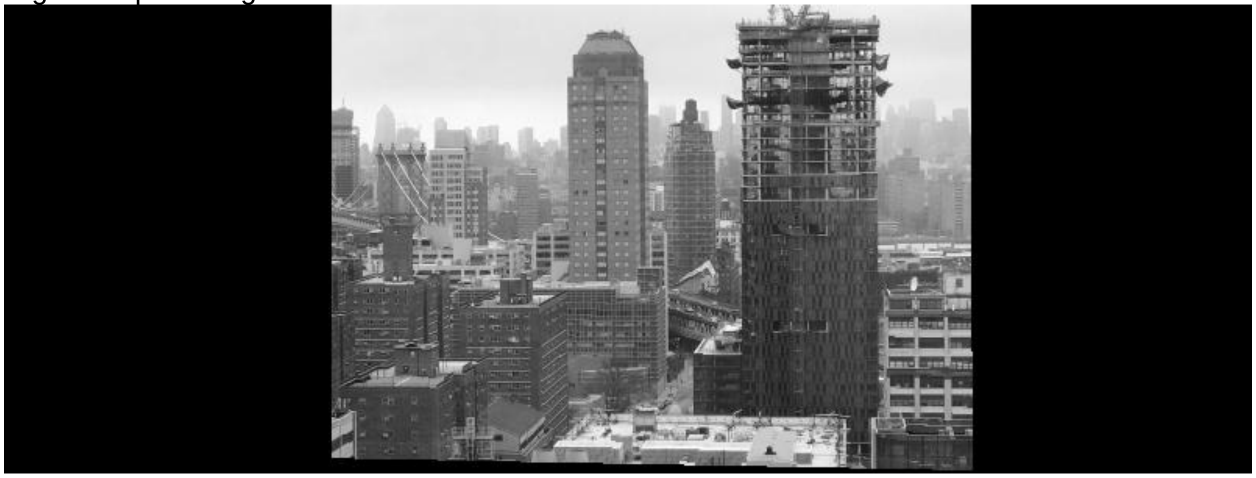
Left Image with SIFT points with scale as radius:



Correspondence between images:



Right Warped Image:



Stitched Image: