Name : Aum M. Dhabalia **Date :** 21/10/2024

Roll No. : 21BEC027

Experiment…8 – SIFT feature descriptor

Objective :

- To understand the concept of SIFT algorithm.
- To find key points and descriptors.

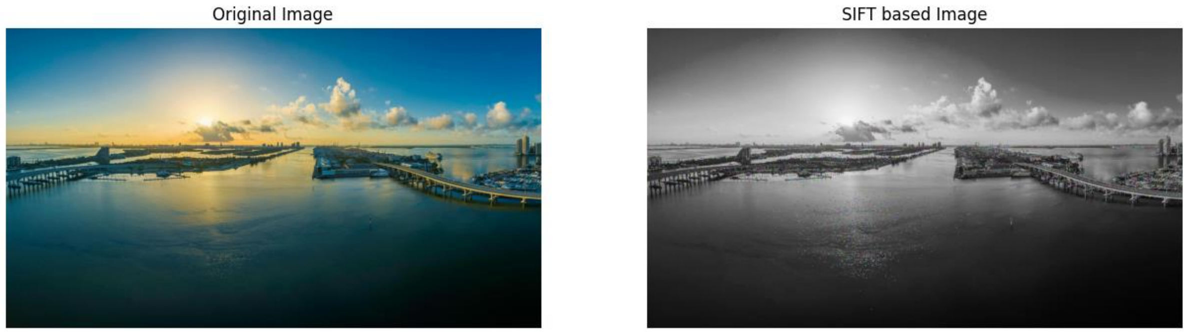
Import necessary libraries…

```
'''
Created on 21 October 2024 Mon 2:09:15 pm
'''
import numpy as np
import cv2
import matplotlib.pyplot as plt
import math
```

Task 1. Write a program to compute the SIFT feature descriptors of the image.

```
I = cv2.imread(r"D:\Nirma Files\Computer Vision\Experiments\Panorama sample1.jpg")
Ig = cv2.cvtColor(I,cv2.COLOR_BGR2GRAY)
sift = cv2.SIFT_create()
kp = sift.detect(Ig,None)

I1 = cv2.drawKeypoints(Ig,kp,I)

plt.subplot(121),plt.imshow(cv2.cvtColor(cv2.imread(r"D:\Nirma Files\Computer Vision\Experiments\Panorama sample1.jpg"),cv2.COLOR_BGR2RGB))
plt.title('Original Image')
plt.axis("off")
plt.subplot(122),plt.imshow(I1)
plt.title('SIFT based Image')
plt.axis("off")
plt.show()
```

Output



Observation :

- Built in SIFT algorithm was applied to detect feature points.
- Red and blue dots are detected feature points

Task 2. Write a program to generate panorama image using SIFT feature descriptor.

```
I1 = cv2.imread(r"D:\Nirma Files\Computer Vision\Experiments\Panorama sample1.jpg")

I2 = cv2.imread(r"D:\Nirma Files\Computer Vision\Experiments\Panorama sample2.jpg")

Ig1 = cv2.cvtColor(I1,cv2.COLOR_BGR2GRAY)

Ig2 = cv2.cvtColor(I2,cv2.COLOR_BGR2GRAY)


kp1, dsc1 = sift.detectAndCompute(Ig1,None)

kp2, dsc2 = sift.detectAndCompute(Ig2,None)

kI1 = cv2.drawKeypoints(Ig1,kp1,I1)

kI2 = cv2.drawKeypoints(Ig2,kp2,I2)


plt.subplot(121),plt.imshow(kI1),plt.title('Image 1'),plt.axis("off")

plt.subplot(122),plt.imshow(kI2),plt.title('Image 2'),plt.axis("off")

plt.show()


bf = cv2.BFMatcher()

matches = bf.knnMatch(dsc1,dsc2,k=2)

goodmatches = []
```

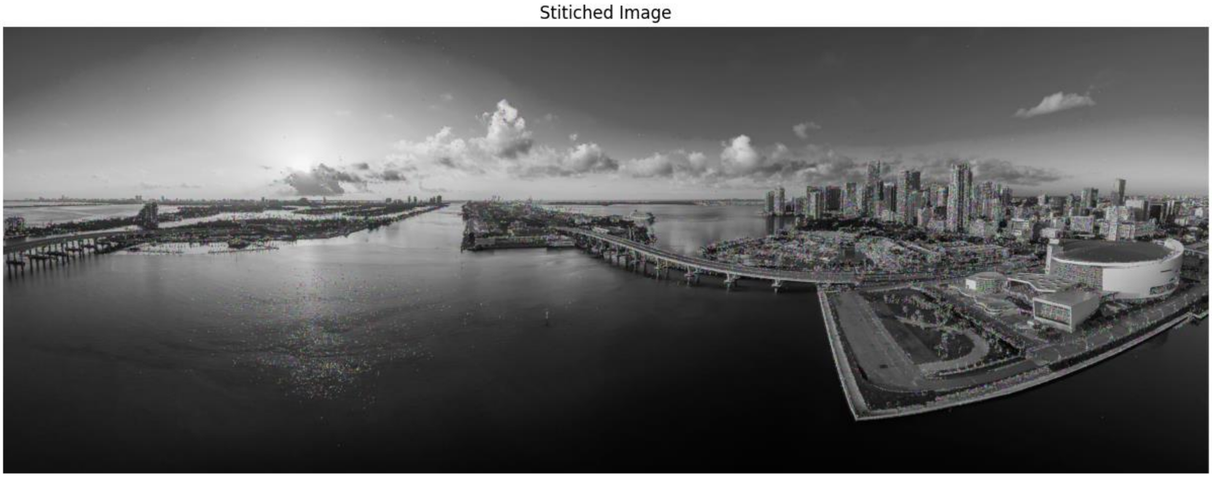
```
for m,n in matches:
    if m.distance < 0.75*n.distance : goodmatches.append(m)

matchImg = cv2.drawMatches(I1,kp1,I2,kp2,goodmatches,np.array([]),(0,255,255),flags=2)
plt.imshow(matchImg),plt.title('Match point Image'),plt.axis("off")
plt.show()

src_pts = np.float32([kp1[m.queryIdx].pt for m in goodmatches]).reshape(-1, 1, 2)
dst_pts = np.float32([kp2[m.trainIdx].pt for m in goodmatches]).reshape(-1, 1, 2)

#Finding Homography Matrix and mask
H,mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
print("Homograpy Matrix")
print(H)

matchesMask = mask.ravel().tolist()
h, w = I1.shape[:2]
pts = np.float32([[0,0],[0,h-1],[w-1,h-1],[w-1,0]]).reshape(-1, 1, 2)

matchIndex = []
for i in range(len(matchesMask)):
        if (matchesMask[i]) : matchIndex.append(i)

matchArray = []
for i in matchIndex:
        matchArray.append(goodmatches[i])

#Finding 10 random matches using inliers
randomMatch = np.random.choice(matchArray,10,replace=False)
draw_params = dict(matchColor=(200,255,158),singlePointColor=None,flags=2)

matchImage = cv2.drawMatches(I1,kp1,I2,kp2,randomMatch,None,**draw_params)
plt.imshow(matchImage),plt.title("Matches"),plt.axis("off")
```

```
plt.show()


h1, w1 = I2.shape[:2]
h2, w2 = I1.shape[:2]
pts1 = np.float32([[0, 0],[0, h1],[w1, h1],[w1, 0]]).reshape(-1, 1, 2)
pts2 = np.float32([[0, 0],[0, h2],[w2, h2],[w2, 0]]).reshape(-1, 1, 2)
pts2_ = cv2.perspectiveTransform(pts2, H)
pts = np.concatenate((pts1, pts2_),axis=0)

#Finding the minimum and maximum coordinates
[xmin, ymin] = np.int32(pts.min(axis=0).ravel() - 0.5)
[xmax, ymax] = np.int32(pts.max(axis=0).ravel() + 0.5)
t = [-xmin, -ymin]

Ht = np.array([[1, 0, t[0]], [0, 1, t[1]], [0, 0, 1]])
#Warping the first image on the second image using Homography Matrix
result = cv2.warpPerspective(I1, Ht.dot(H), (xmax-xmin, ymax-ymin))
result[t[1]:h1+t[1],t[0]:w1+t[0]] = I2

plt.imshow(result),plt.title("Stitiched Image"),plt.axis("off")
plt.show()
```

Output

```
Homograpy Matrix
[[ 1.00001414e+00 -1.03749955e-06 -3.35305415e+03]
 [ 1.11209322e-05  9.99980524e-01 -1.01256301e+00]
 [ 5.88329971e-09 -1.40808821e-08  1.00000000e+00]]
```

Stitiched Image



Observation :

- *Panorama sample1.jpg* and *Panorama sample2.jpg* were stitiched to form original *Panorama* image.
- With SIFT keypoints and descriptors were identified and with those keypoints homography matrix was calculated.
- The homography matrix was then used to warp *Panorama sample1.jpg* and then concatenated with *Panorama sample2.jpg*

Conclusion:-

As the experiment performed,

- the concept of feature point detection was familiarized and discussed.
- Scale Invariant Feature Transform (SIFT) algorithm was familiarized and implemented for feature point detection.
- using SIFT algorithm, a panorama image was stitched.

Libraries and functions used are matplotlib, numpy, OpenCV and math.