190622R_excercise_08

April 4, 2022

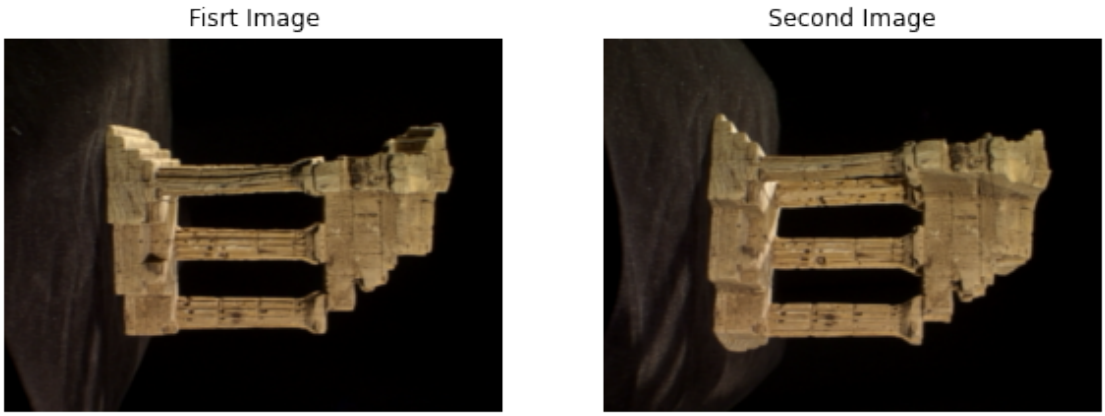
Name : Tharindu O.K.D

Index No. : 190622R

Question 01

[ ]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
f = open(r'templeSR_par.txt', 'r')
assert f is not None
n = int(f.readline())

# first image
l = f.readline().split()
im1_fn = l[0]
K1 = np.array([float(i) for i in l[1:10]]).reshape((3,3))
R1 = np.array([float(i) for i in l[10:19]]).reshape((3,3))
t1 = np.array([float(i) for i in l[19:22]]).reshape((3,1))

# second image
l = f.readline().split()
im2_fn = l[0]
K2 = np.array([float(i) for i in l[1:10]]).reshape((3,3))
R2 = np.array([float(i) for i in l[10:19]]).reshape((3,3))
t2 = np.array([float(i) for i in l[19:22]]).reshape((3,1))

# read the two images
img1 = cv.imread(im1_fn, cv.IMREAD_COLOR)
img2 = cv.imread(im2_fn, cv.IMREAD_COLOR)
assert img1 is not None
assert img2 is not None
fig, ax = plt.subplots(1, 2, figsize=(10, 10))
ax[0].imshow(cv.cvtColor(img1, cv.COLOR_BGR2RGB))
ax[0].axis('off')
ax[0].set_title("Fisrt Image")
ax[1].imshow(cv.cvtColor(img2, cv.COLOR_BGR2RGB))
ax[1].axis('off')
```

```
ax[1].set_title("Second Image")
plt.show()
```



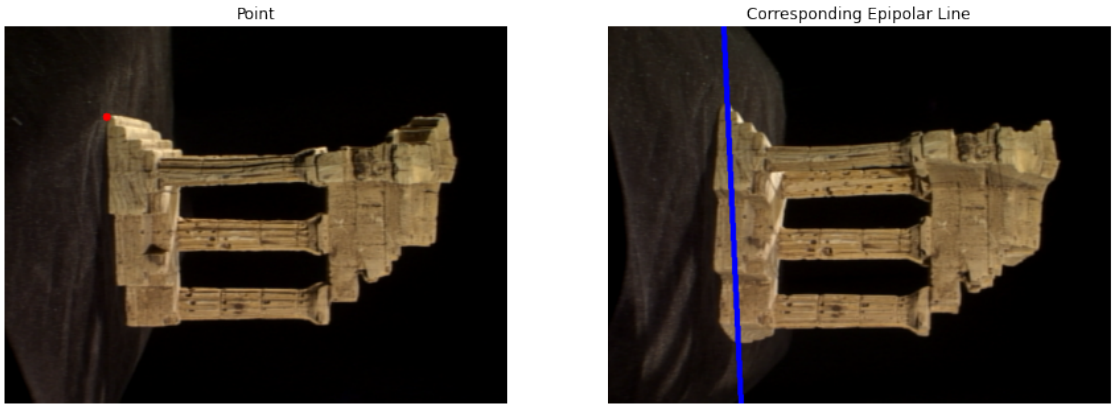
Question 02

[ ]:
```
# Compute the camera matrices

P1 = K1 @ np.hstack((R1, t1))
P2 = K2 @ np.hstack((R2, t2))
```

Question 03

[ ]:
```
from scipy.linalg import null_space

def skew(x):
    x = x.ravel()
    return np.array([[0, -x[2], x[1]], [x[2], 0, -x[0]], [-x[1], x[0], 0]])

C = null_space(P1)
C = C * np.sign(C[0, 0])
e2 = P2 @ C

e2x = skew(e2)

F = e2x @ P2 @ np.linalg.pinv(P1)

x = np.array([130, 115, 1])
cv.circle(img1, (x[0], x[1]), 5, (0, 0, 255), -1)
l2 = F @ x.T
p1 = np.array([0, (l2[0]*0 + l2[2])/l2[1]]).astype(int)
p2 = np.array([500, (l2[0]*500 + l2[2])/l2[1]]).astype(int)
```

```
cv.line(img2, (p1[0], p1[1]), (p2[0], p2[1]), (255, 0, 0), 5)

fig, ax = plt.subplots(1, 2, figsize=(15, 15))
ax[0].imshow(cv.cvtColor(img1, cv.COLOR_BGR2RGB))
ax[0].set_title("Point")
ax[0].axis("off")
ax[1].imshow(cv.cvtColor(img2, cv.COLOR_BGR2RGB))
ax[1].set_title("Corresponding Epipolar Line")
ax[1].axis("off")
plt.show()
```

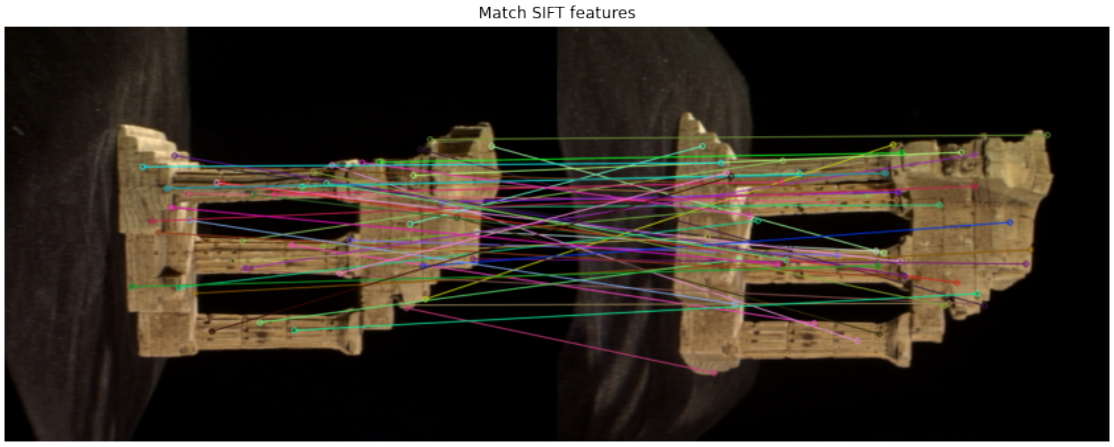


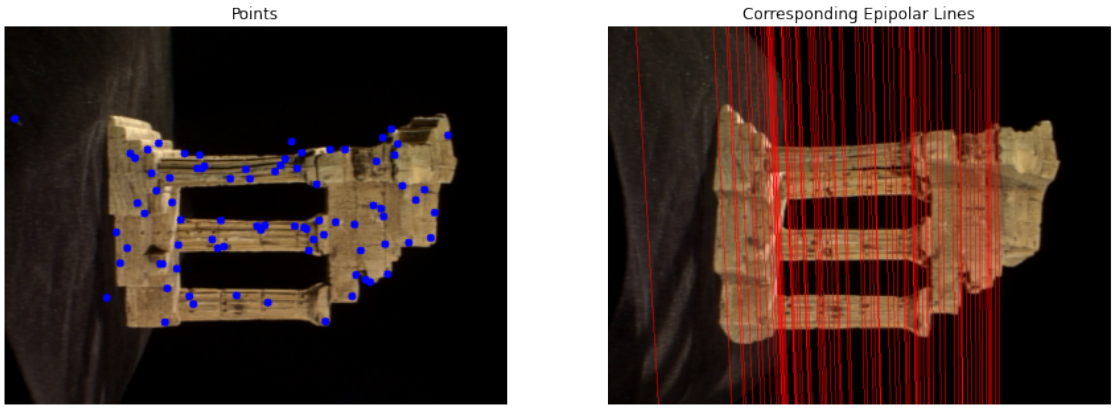
Question 04

Using above calculated Fundamental Matrix (F)

[ ]:
```
img1 = cv.imread(im1_fn)
img1 = cv.cvtColor(img1, cv.COLOR_BGR2RGB)
img2 = cv.imread(im2_fn)
img2 = cv.cvtColor(img2, cv.COLOR_BGR2RGB)


sift = cv.xfeatures2d.SIFT_create()

keypoints_1, descriptors_1 = sift.detectAndCompute(img1,None)
keypoints_2, descriptors_2 = sift.detectAndCompute(img2,None)
keypoints_1 = keypoints_1[::10]
keypoints_2 = keypoints_2[::10]
descriptors_1 = descriptors_1[::10]
descriptors_2 = descriptors_2[::10]


bf = cv.BFMatcher(cv.NORM_L1, crossCheck=True)
```

```
matches = bf.match(descriptors_1,descriptors_2)
matches = sorted(matches, key = lambda x:x.distance)

img3 = cv.drawMatches(img1, keypoints_1, img2, keypoints_2, matches[:150],␣
 ↪img2, flags=2)

fig, ax = plt.subplots(figsize=(15,15))
ax.imshow(img3)
ax.set_title("Match SIFT features")
plt.axis("off")
plt.show()

points = np.array(cv.KeyPoint_convert(keypoints_1))
ones = np.ones((points.shape[0], 1))
points = np.concatenate((points, ones), axis=1).astype(int)


for point in points:
    cv.circle(img1, (point[0], point[1]), 5, (0, 0, 255), -1)
    l2 = F @ point.T
    p1 = np.array([0, (l2[0]*0 + l2[2])/l2[1]]).astype(int)
    p2 = np.array([500, (l2[0]*500 + l2[2])/l2[1]]).astype(int)
    cv.line(img2, (p1[0], p1[1]), (p2[0], p2[1]), (255, 0, 0), 1)

fig, ax = plt.subplots(1, 2, figsize=(15, 15))
ax[0].imshow(img1)
ax[0].set_title("Points")
ax[0].axis("off")
ax[1].imshow(img2)
ax[1].set_title("Corresponding Epipolar Lines")
ax[1].axis("off")
plt.show()
```





Using FLANN based matcher and ratio test. (OpenCV documentation)

[ ]:
```
img1 = cv.imread(im1_fn,0)
img2 = cv.imread(im2_fn,0)
sift = cv.xfeatures2d.SIFT_create()

kp1, des1 = sift.detectAndCompute(img1,None)
kp2, des2 = sift.detectAndCompute(img2,None)

FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
search_params = dict(checks=50)
flann = cv.FlannBasedMatcher(index_params,search_params)
matches = flann.knnMatch(des1,des2,k=2)
pts1 = []
pts2 = []

for i,(m,n) in enumerate(matches):
    if m.distance < 0.8*n.distance:
        pts2.append(kp2[m.trainIdx].pt)
        pts1.append(kp1[m.queryIdx].pt)

pts1 = np.int32(pts1)
pts2 = np.int32(pts2)
F, mask = cv.findFundamentalMat(pts1,pts2,cv.FM_LMEDS)

pts1 = pts1[mask.ravel()==1]
pts2 = pts2[mask.ravel()==1]

def drawlines(img1,img2,lines,pts1,pts2):
    r,c = img1.shape
```

```
    img1 = cv.cvtColor(img1,cv.COLOR_GRAY2BGR)
    img2 = cv.cvtColor(img2,cv.COLOR_GRAY2BGR)
    for r,pt1,pt2 in zip(lines,pts1,pts2):
        color = tuple(np.random.randint(0,255,3).tolist())
        x0,y0 = map(int, [0, -r[2]/r[1] ])
        x1,y1 = map(int, [c, -(r[2]+r[0]*c)/r[1] ])
        img1 = cv.line(img1, (x0,y0), (x1,y1), color,1)
        img1 = cv.circle(img1,tuple(pt1),5,color,-1)
        img2 = cv.circle(img2,tuple(pt2),5,color,-1)
    return img1,img2


lines1 = cv.computeCorrespondEpilines(pts2.reshape(-1,1,2), 2,F)
lines1 = lines1.reshape(-1,3)
img5,img6 = drawlines(img1,img2,lines1,pts1,pts2)

lines2 = cv.computeCorrespondEpilines(pts1.reshape(-1,1,2), 1,F)
lines2 = lines2.reshape(-1,3)
img3,img4 = drawlines(img2,img1,lines2,pts2,pts1)
fig, ax = plt.subplots(1, 2, figsize=(15, 15))
ax[0].imshow(img5)
ax[0].set_title("Points")
ax[0].axis("off")
ax[1].imshow(img3)
ax[1].set_title("Corresponding Epipolar Lines")
ax[1].axis("off")
plt.show()
```