EN2550 _ Exercise 09

Index Number : 190328V

Name : KUMARA B.W.J.C.

In [ ]:

```
#Importing  all the Libraries
%matplotlib inline

import numpy as np
import sympy as sy
import matplotlib.pyplot as plt
import cv2 as cv
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
```

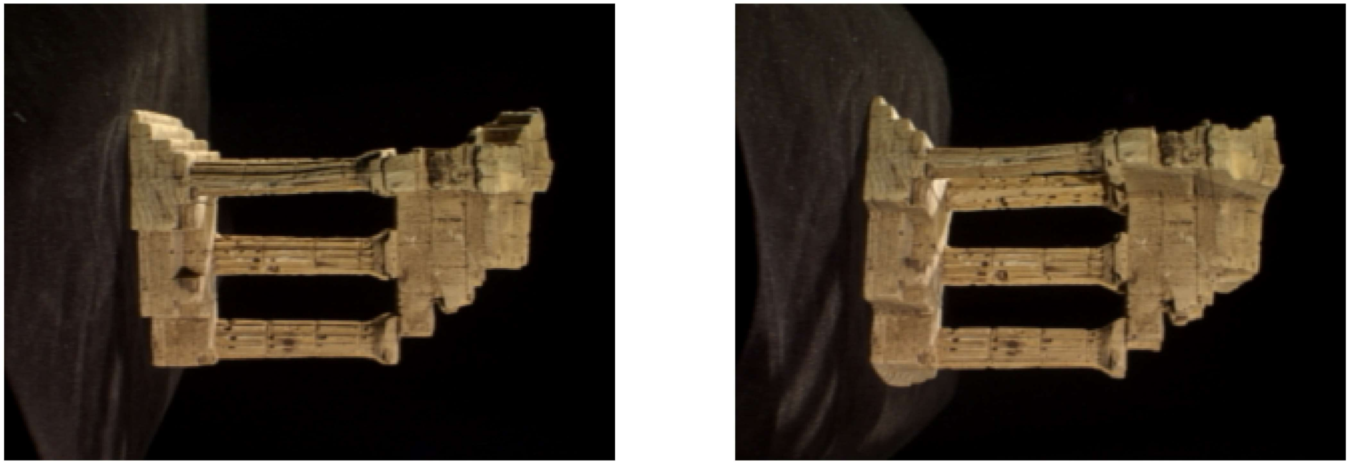
In [ ]:

```
# Computing K,R,t and Camera matrices for i = 1,2

img = open(r'Images/templeSparseRing/templeSR_par.txt','r')
assert img is not None
n = int(img.readline())
#Read information of first image
l = img.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))
#Read information of second image
l = img.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))
P1 = K1 @ np.hstack((R1,t1))
P2 = K2 @ np.hstack((R2,t2))

img1 = cv.imread('Images/templeSparseRing/templeSR0001.png',cv.IMREAD_COLOR)
assert img1 is not None
img2 = cv.imread('Images/templeSparseRing/templeSR0002.png',cv.IMREAD_COLOR)
```

```
assert img2 is not None
fig,ax = plt.subplots(1,2,figsize=(12,12))
i1 = cv.cvtColor(img1,cv.COLOR_BGR2RGB)
i2 = cv.cvtColor(img2,cv.COLOR_BGR2RGB)
ax[0].axis('off')
ax[1].axis('off')
ax[0].imshow(i1)
ax[1].imshow(i2)
```

Out[ ]:

```
<matplotlib.image.AxesImage at 0x24f8faa8610>
```



Q1. Find SIFT features in the two images

In [ ]:

```
sift = cv.SIFT_create()
kp1,decs1 = sift.detectAndCompute(img1,None)
kp2,decs2 = sift.detectAndCompute(img2,None)

FLANN_INDEX_KDTREE = 0
indexParams = dict(algorithm=FLANN_INDEX_KDTREE,trees=5)
searchParams = dict(checks=50)

flann = cv.FlannBasedMatcher(indexParams,searchParams)
matches = flann.knnMatch(decs1,decs2,k=2)

good = []
pts1 = []
pts2 = []
```

```
for i,(m,n) in enumerate(matches):
    if m.distance<0.7*n.distance:
        good.append(m)
        pts1.append(kp1[m.queryIdx].pt)
        pts2.append(kp2[m.trainIdx].pt)
pts1 = np.array(pts1)
pts2 = np.array(pts2)
```

Q2. Computing fundamental matrix F and essential matrix E.

In [ ]:

```
F,mask = cv.findFundamentalMat(pts1,pts2,cv.FM_RANSAC)
E = K2.T @ F @ K1
```

Q3. Recovering the pose of the second camera with respect to the first using recoverPose method

In [ ]:

```
retval,R,t,mask = cv.recoverPose(E,pts1,pts2,K1)
```

Q4. Computing the camera Matrix of P_2

In [ ]:

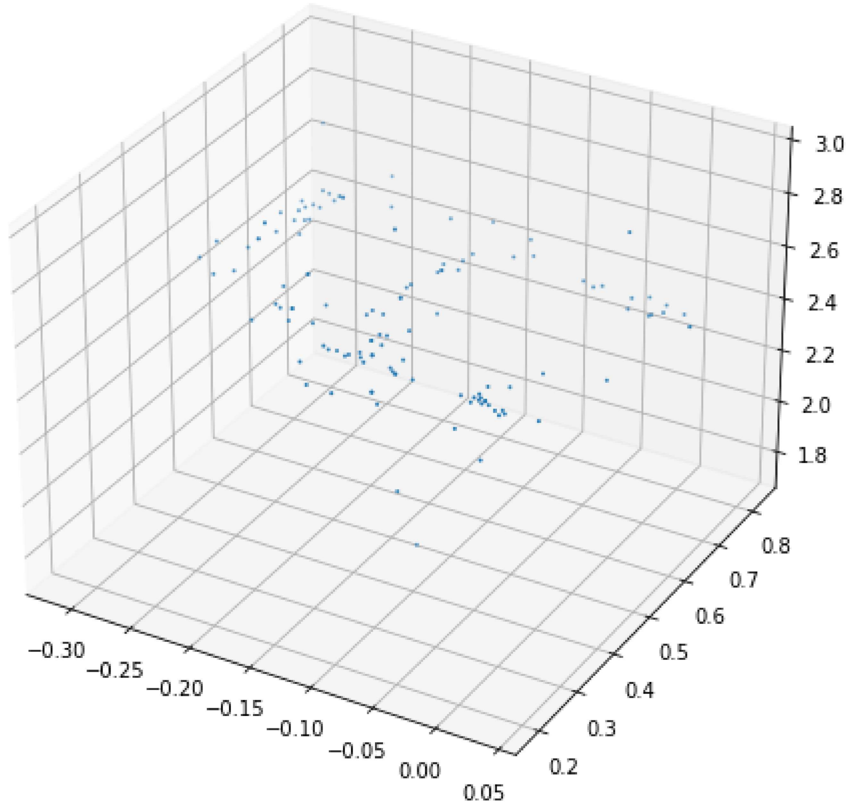
```
R_t_1 = np.concatenate((R1,t1),axis=1) # 3 x 4 matrix
R_t_2 = np.empty((3,4))

R2_= R1 @ R
t2_ = R1 @ t

R_t_2 = np.concatenate((R2_,t2_),axis=1)

p2_ = K2 @ R_t_2
```

Q5. Finding the 3-D point locations using triangulatePoints method and Plotting them.

In [ ]:

```
points4D = cv.triangulatePoints(P1,p2_,pts1.T,pts2.T)
points4D /= points4D[3,:]

X = points4D[0,:]
Y = points4D[1,:]
Z = points4D[2,:]
```

```
fig=plt.figure(figsize=(8,8))

ax = fig.add_subplot(111,projection='3d')
ax.scatter(X,Y,Z,s=1,cmap='gray')

plt.show()
```


3.0
2.8
2.6
2.4
2.2
2.0
1.8
0.8
0.7
0.6
0.5
0.4
0.3
0.2
−0.30
−0.25
−0.20
−0.15
−0.10
−0.05
0.00
0.05