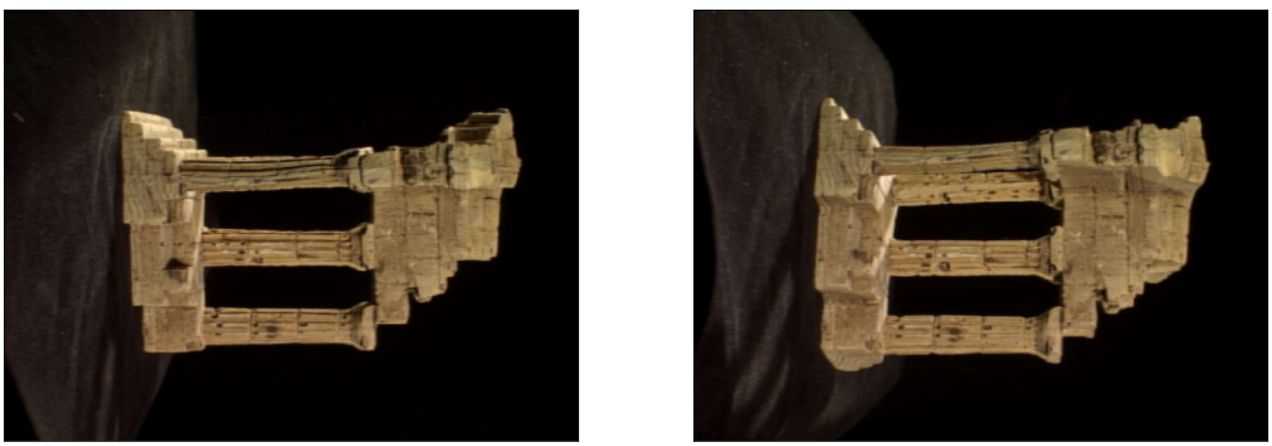
Name : Dilshan J.V.A.P

Index number : 190144D

Exercise 9

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv
```

In [ ]:
```
f = open(r'templeSparseRing/templeSR_par.txt', 'r')
assert f is not None
n= int(f.readline())
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))
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))
im1= cv.imread(r'templeSparseRing/' + im1_fn,cv.IMREAD_COLOR)
im2= cv.imread(r'templeSparseRing/' + im2_fn,cv.IMREAD_COLOR)
assert im1 is not None
assert im2 is not None
fig, ax = plt.subplots( 1, 2, figsize = (18, 8))
ax[0].imshow(cv.cvtColor(im1, cv.COLOR_BGR2RGB))
ax[1].imshow(cv.cvtColor(im2, cv.COLOR_BGR2RGB))
for i in range(2):
    ax [i] . set_xticks ([]) , ax [i] . set_yticks ([])
plt.show()
```



In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
sift = cv.xfeatures2d.SIFT_create()
kp1, decs1 = sift.detectAndCompute(im1, None)
kp2, decs2 = sift.detectAndCompute(im2, None)
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm =FLANN_INDEX_KDTREE, trees = 5 )
search_params = dict(checks=100)
flann = cv.FlannBasedMatcher(index_params, search_params)
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)
F,mask = cv.findFundamentalMat(pts1, pts2, cv.FM_RANSAC)
print ("F:\n",F)
E = k2.T @ F @ k1
print ("E:\n",E)
retval, R, t, mask = cv.recoverPose(E, pts1, pts2, k1)
R_t_1 = np.concatenate((R1, t1), axis =1) # 3 x 4
R2_ = R1 @ R
t2_ = R1 @ t
R_t_2 = np.concatenate((R2_, t2_), axis =1)
P1 = k1 @ np.hstack((R1, t1))
P2_ = k2 @ R_t_2
```

```
F:
 [[-4.44450786e-07  1.79763466e-06 -2.19348716e-02]
 [ 8.40214781e-06 -6.31040656e-07 -2.55145252e-03]
 [ 1.99925227e-02 -2.06679022e-03  1.00000000e+00]]
E:
 [[ -1.02739962   4.17047351 -32.87934309]
 [ 19.49280109  -1.46929664  -0.2549782 ]
 [ 33.34601275  -2.56216054  -0.04514887]]
```

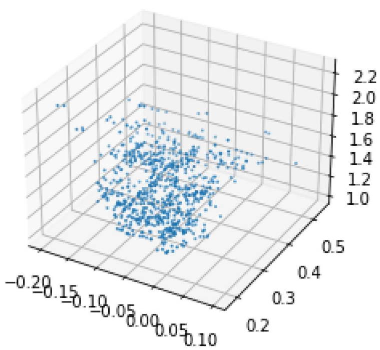
In [ ]:

```
sift = cv.xfeatures2d.SIFT_create()
kp1, decs1 = sift.detectAndCompute(im1, None)
kp2, decs2 = sift.detectAndCompute(im2, None)
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm =FLANN_INDEX_KDTREE, trees = 5 )
search_params = dict(checks=100)
flann = cv.FlannBasedMatcher(index_params, search_params)
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)
F,mask = cv.findFundamentalMat(pts1, pts2, cv.FM_RANSAC)
print ("F:\n",F)
E = k2.T @ F @ k1
print ("E:\n",E)
retval, R, t, mask = cv.recoverPose(E, pts1, pts2, k1)
R_t_1 = np.concatenate((R1, t1), axis =1) # 3 x 4
R2_ = R1 @ R
t2_ = R1 @ t
R_t_2 = np.concatenate((R2_, t2_), axis =1)
P1 = k1 @ np.hstack((R1, t1))
P2_ = k2 @ R_t_2
```

```
F:
 [[ 1.50846035e-06  2.83419586e-05 -1.79198610e-02]
 [-2.24355041e-05  1.59328003e-07  1.00190107e-02]
 [ 1.50687146e-02 -1.28138045e-02  1.00000000e+00]]
E:
 [[ 3.48698131e+00  6.57527306e+01 -1.59140935e+01]
 [-5.20498840e+01  3.70974672e-01  4.99830279e+00]
 [ 1.51828656e+01 -6.41813414e+00  3.64910387e-02]]
```

In [ ]:

```
points4d = cv.triangulatePoints(P1, P2_, pts1.T, pts2.T)
points4d /= points4d[3, :]
import matplotlib.pyplot as plt
X = points4d[0, :]
Y = points4d[1, :]
Z = points4d[2, :]
fig = plt.figure(1)
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X, Y, Z, s=1, cmap='gray')
plt.show()
```


2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.5
0.4
0.3
0.2
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10