190071B

Bandara D.R.K.W.M.S.D

In [2]:
```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
```

In [18]:
```
#Question1
s = open('templeSparseRing/templeSR_par.txt','r')
assert s is not None
n = int(s.readline())
#first img
l = s.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 img
l = s.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('templeSparseRing/'+ im1_fn, cv.IMREAD_COLOR)
im2 = cv.imread('templeSparseRing/'+ im2_fn, cv.IMREAD_COLOR)
assert im1 is not None
assert im2 is not None
```

In [37]:
```
#Question2
sift = cv.xfeatures2d.SIFT_create()
kp1,desc1 = sift.detectAndCompute(im1,None)
kp2,desc2 = sift.detectAndCompute(im2,None)
FLANN_INDEX_KDTREF = 1
index_params = dict(algorithm = FLANN_INDEX_KDTREF,trees = 5)
search_params = dict(checks=100)
flann = cv.FlannBasedMatcher(index_params,search_params)
matches = flann.knnMatch(desc1,desc2, 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)
E = K2.T @ f @ K1
print('F= ',f)
print('E= ',E)
```

```
F=  [[ 1.49034037e-06  1.44154168e-05 -2.53948320e-02]
 [-8.25788252e-06  8.67005344e-08  4.00767127e-03]
 [ 2.27526901e-02 -7.28270380e-03  1.00000000e+00]]
E=  [[ 3.44509489e+00  3.34434549e+01 -3.25145725e+01]
 [-1.91581088e+01  2.01870994e-01  2.33852108e+00]
 [ 3.21786978e+01 -4.43004055e+00 -6.22266684e-03]]
```

In [35]:
```
#Question3

retval,R,t,mask = cv.recoverPose(E,pts1,pts2,K1)
print('R= ',R)
print('t= ',t)
```

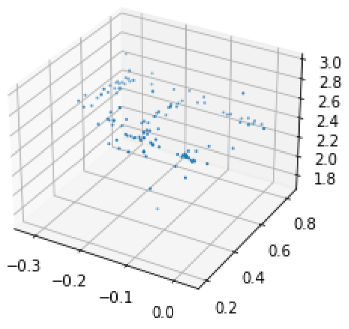
```
R=  [[ 0.99754052 -0.07006883  0.00181066]
 [ 0.06705657  0.9615392   0.2663565 ]
 [-0.02040431 -0.26557999  0.96387288]]
t=  [[ 0.01312307]
 [-0.86850153]
 [-0.49551275]]
```

In [36]:
```
#Question4

R_t_1 = np.concatenate((R1,t1),axis = 1)
R_t_2 = np.empty((3,4))
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
print('P2_= ',P2_)
```

```
P2_=  [[ 1.56140182e+02  1.53317827e+03 -1.67326558e+02 -9.66850517e+02]
 [ 1.53102041e+03 -1.25962559e+02 -1.71538765e+02  1.56694615e+02]
 [ 5.65837070e-02  8.28361136e-02 -9.94955508e-01  6.45008519e-01]]
```

In [30]:
```
#Question5
points4d = cv.triangulatePoints(P1,P2_,pts1.T,pts2.T)
points4d /= points4d[3,:]
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()
```


3.0
2.8
2.6
2.4
2.2
2.0
1.8
0.8
0.6
0.4
0.2
−0.3
−0.2
−0.1
0.0