Name : Chathumini B.G.D.T.

Index Number : 190107T

Github Repo: https://github.com/dulmi-19/Image-Processing-and-Machine-Vision

In [ ]:

```
import numpy as np
import cv2 as cv

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

sift = cv.SIFT_create()
kp1, desc1 = sift.detectAndCompute(im1, None)
kp2, desc2 = sift.detectAndCompute(im2, None)

FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE,tree=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
retval,R,t,mask = cv.recoverPose(E,pts1,pts2,K1)

R_t_1 = np.concatenate((R1,t1),axis=1)
R2_ = R1 @ R
t2_ = R1 @ t
R_t_2= np.concatenate((R2_,t2_),axis=1)
```

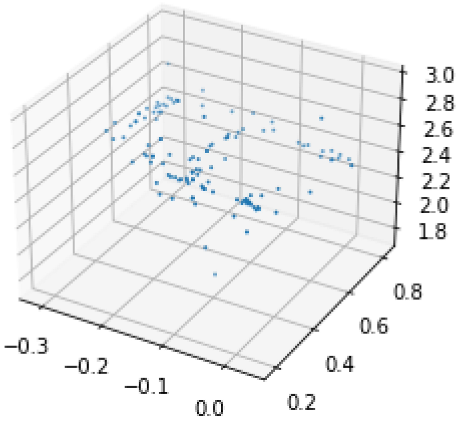
```
P1 = K1 @ np.hstack((R1,t1))
P2_ = K2 @ R_t_2
```

In [ ]:

```
import matplotlib.pyplot as plt
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()
```



In [ ]:

```
F
```

Out[ ]:

```
array([[ 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]])
```

In [ ]:

```
F_ = np.array([[ 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]])

F_/=F_[2,2]
F_
```

Out[ ]:

```
array([[ 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]])
```