Index number : 190026T

Name : AHAMED M.I.I

In [ ]:

```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
f = open(r'E:\Aca\aca sem 4\Image Processing & Machine vision\exercises\exercise_09\
assert f is not None

n = int(f.readline())
l = f.readline().split()
im1_fn = l[0]
#for first image
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))

#for 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 image sand show
im1 =  cv.imread(r'E:\Aca\aca sem 4\Image Processing & Machine vision\exercises\exer
assert im1 is not None

im2 =  cv.imread(r'E:\Aca\aca sem 4\Image Processing & Machine vision\exercises\exer
assert im2 is not None
fig , ax =  plt.subplots(1,2,figsize=(15,15))
ax[0].imshow(cv.cvtColor(im1, cv.COLOR_BGR2RGB))
ax[0].set_title('Image 1')
ax[0].set_xticks([]), ax[0].set_yticks([])

ax[1].imshow(cv.cvtColor(im2, cv.COLOR_BGR2RGB))
ax[1].set_title('Image 2')
ax[1].set_xticks([]), ax[1].set_yticks([]);

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

In [ ]:

```
#1)

sift = cv.SIFT_create()
```

```
kp1, des1 = sift.detectAndCompute(im1,None)
kp2, des2 = 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(des1, des2, k = 2)
pts1 = []
pts2 = []


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

pts1 = np.array(pts1)
pts2 = np.array(pts2)
```

In [ ]:

```
#2)


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

print(F)
print(E)
```

```
[[ 1.19353197e-06  1.48128487e-05 -2.65668422e-02]
 [-8.37167541e-06  6.34793204e-07  2.04080864e-03]
 [ 2.41439516e-02 -5.73622910e-03  1.00000000e+00]]
[[ 2.75898779e+00  3.43654884e+01 -3.42837514e+01]
 [-1.94221058e+01  1.47803397e+00 -5.08742503e-01]
 [ 3.41148335e+01 -1.68046954e+00 -1.62748485e-02]]
```

In [ ]:

```
#3)

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))
```

In [ ]:

```
#4)

P2_ = K2@R_t_2

print(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 [ ]:

```
#5)

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, figsize=(5, 5))
ax = fig.add_subplot(111,projection = '3d')
ax.scatter(X, Y, Z, s = 1, cmap = 'gray')
ax.set_title("points")
plt.show()
```