EN2550 - Fundementals of Image Processing and Machine Vision

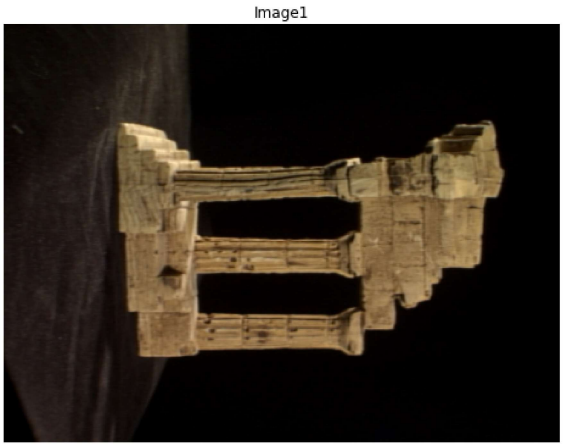
Name : R.G.S.M. RANATUNGA

Index No. : 190504H

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

In [ ]:
```
f = open(r'./templeSparseRing/templeSR_par.txt', 'r')
assert f is not None

n = int(f.readline())

# Reading the information on the first image
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))

# Reading the information on the 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 images and show
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, 18))

ax[0].set_title('Image1')
ax[0].imshow(cv.cvtColor(im1, cv.COLOR_BGR2RGB))
ax[0].axis('off')
ax[1].set_title('Image2')
ax[1].imshow(cv.cvtColor(im2, cv.COLOR_BGR2RGB))
ax[1].axis('off')
```

Out[ ]:
```
(-0.5, 639.5, 479.5, -0.5)
```



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)

E = K2.T @ F @ K1 #Essential matrix
retval, R, t, mask = cv.recoverPose(E, pts1, pts2, K1)

R_t_1 = np.concatenate((R1, t1), axis=1) # 3 x 4
R_t_2 = np.empty((3,4))
R2_ = R1 @ R
t2_ = R1 @ t
R_t_2 = np.concatenate((R2_,t2_), axis=1) # 3 x 4

P1 = K1 @ np.hstack((R1, t1)) #First camera matrix from data read file
P2_ = K2 @ R_t_2 # Second camera matrix estimated from the fundemental matrix computed
```

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(1)
ax = fig.add_subplot(111, projection = '3d')

ax.scatter(X, Y, Z, s=1, cmap='gray')

plt.show()
```