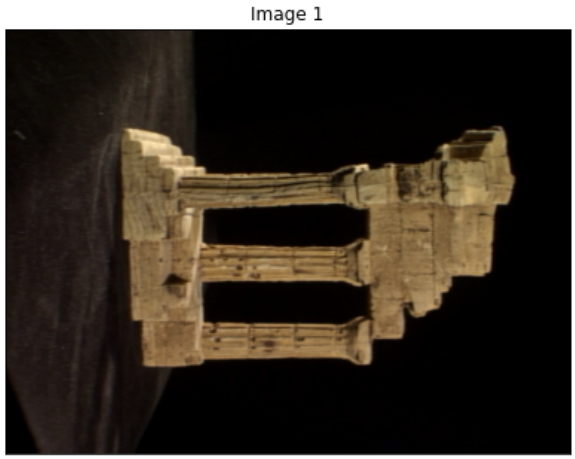
Name : Fernando I.A.M.D.

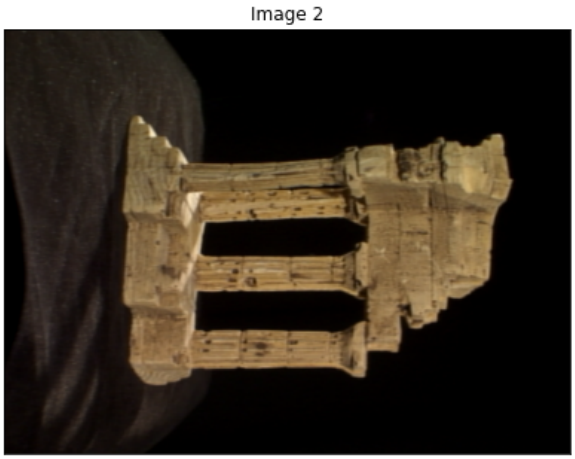
Index No.: 190172K

In [ ]:
```
import cv2
import numpy as np
import sympy
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
from plyfile import PlyData,PlyElement
%matplotlib inline
```

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 =  cv2.imread(r'./templeSparseRing/'+im1_fn , cv2.IMREAD_COLOR)
im2 =  cv2.imread(r'./templeSparseRing/'+ im2_fn , cv2.IMREAD_COLOR)
assert im1 is not None
assert im2 is not None
fig , ax =  plt.subplots(1,2,figsize=(15,15))
ax[0].imshow(cv2.cvtColor(im1, cv2.COLOR_BGR2RGB))
ax[0].set_title('Image 1')
ax[0].set_xticks([]), ax[0].set_yticks([])

ax[1].imshow(cv2.cvtColor(im2, cv2.COLOR_BGR2RGB))
ax[1].set_title('Image 2')
ax[1].set_xticks([]), ax[1].set_yticks([])

plt.plot()
```

Out[ ]:
```
[]
```





Q1

In [ ]:
```
sift = cv2.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 = cv2.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)
```

Q2

In [ ]:
```
#Fundamental matrix
F, mask = cv2.findFundamentalMat(pts1,pts2,cv2.FM_RANSAC)

#Essential matrix
E = K2.T@F@K1
```

Q3

In [ ]:
```
retval,R,t,mask = cv2.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))
```

Q4

In [ ]:

```
P2_=K2@R_t_2
```

Q5

In [ ]:

```
points4d = cv2.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()
```