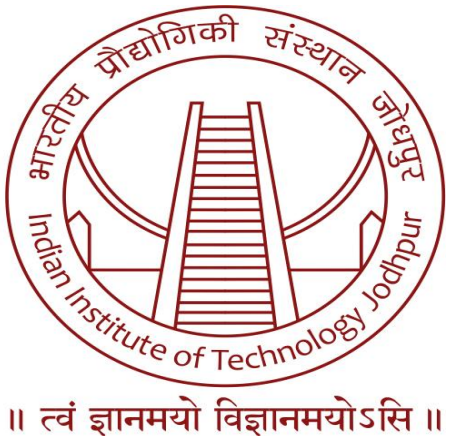
Autonomous System
Fractal 2, Programming Assignment-1



Submitted By:
Umang Barbhaya (M20CS017)

Course Instructor:
Dr. Rajendra Nagar
Indian Institute of Technology Jodhpur

Team Information:
Umang Barbhaya (M20CS017)

Colablink:
https://colab.research.google.com/drive/1Ab2IKLxCt0StD7tKWIo77h8tdy7xsELk#scrollTo=YJMaB3m_nGyc

1. **Consider two images I1 (“im1.jpg”) and I2 (“im2.jpg”) of a static scene captured from a single camera with the given intrinsic camera matrix K (“Intrinsic Matrix K.txt”).**
   The files im1.jpg and im2.jpg and the intrinsic camera matrix K are given below:

Figure: Image 1 (im1.jpg)

Figure: Image 2 (im2.jpg)

The Intrinsic Matrix K is given below:

1698.873755	0.000000	971.7497705
0.000000	1698.8796645	647.7488275
0.000000	0.000000	1.000000

- **Find a set of ground-truth correspondences {(pi, p0i)}ni=1 using any of the existing implementations. Ensure that there are at least n = 100 true correspondences.**

  Answer:

  We have used Scale Invariant Feature Transform (SIFT) technique for finding the set of ground-truth correspondences. It is a python library whose main function is to identify the key features of the images and match these features. By using this feature detection algorithm, each keypoints are described and matched across the images. A total of 1795 sets of ground-truth correspondences are found. The python snapshot output of the set of ground-truth correspondences is shown below:

```
(1160.608154296875, 438.9165344238281) (1157.55322265625, 468.7167053222656)
(1162.043701171875, 907.6554565429688) (1160.5177001953125, 933.6729736328125)
(1162.063232421875, 572.1692504882812) (1159.5213623046875, 601.7001953125)
(1162.063232421875, 572.1692504882812) (1159.5213623046875, 601.7001953125)
(1163.0789794921875, 406.87255859375) (1160.15087890625, 436.70751953125)
(1163.947998046875, 996.6685180664062) (1110.6142578125, 990.215087890625)
(1166.029296875, 461.345703125) (1163.070068359375, 490.5953369140625)
(1166.4503173828125, 1048.21826171875) (1137.800537109375, 1043.165771484375)
(1170.0849609375, 472.0318908691406) (1174.116943359375, 498.928466796875)
(1170.0880126953125, 997.1647338867188) (1123.802490234375, 1109.598388671875)
(1171.093994140625, 457.33843994140625) (1168.2716064453125, 485.7295837402344)
(1172.121337890625, 464.1701354980469) (955.6382446289062, 452.1195373535156)
(1173.8253173828125, 413.9115905761719) (1170.697021484375, 443.39111328125)
(1177.1370849609375, 470.4154968261719) (1180.146240234375, 502.0074768066406)
(1178.21630859375, 472.6647033691406) (782.3855590820312, 850.27392578125)
(1179.25732421875, 424.7236022949219) (1144.61279296875, 1007.5425415039062)
(1406.39111328125, 659.2801513671875) (1409.2625732421875, 698.8324584960938)
1795
```

Figure 1: Set of ground-truth correspondences

The plotting of ground-truth correspondences is given below:

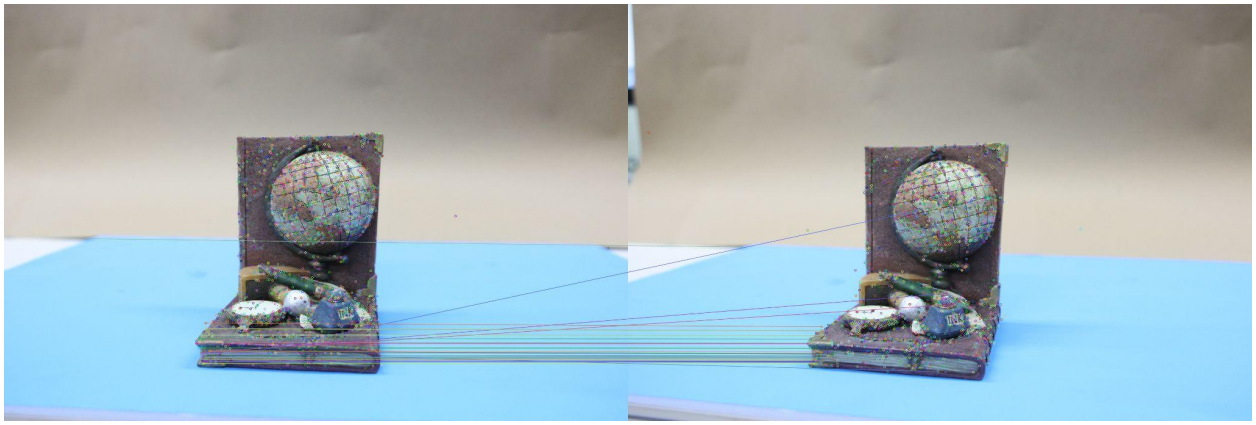


Figure: Plot of ground-truth correspondences

- **Assume that the world-coordinate system is aligned with the coordinate-system of the camera location. Implement the algorithm taught in the class to find the Essential matrix E.**
  Answer:
  The Intrinsic Matrix K is given below:

  1698.873755 0.000000 971.7497705
  0.000000 1698.8796645 647.7488275

  We can find out the K inverse by using numpy library i.e np.linalg.inv(K) in Python. Therefore, the K inverse calculated snippet is given below:

```
----------------------------K inverse-----------------------------------
[[ 5.88625257e-04  0.00000000e+00 -5.71996458e-01]
 [ 0.00000000e+00  5.88623209e-04 -3.81279994e-01]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e+00]]
```

Figure: Snapshot of output of K inverse

For calculating the Essential Matrix E, we need to find out all the normalized image coordinates xi and yi. These can be calculated using the below formula:

$$\mathbf{x}_i = \mathbf{K}^{-1}\hat{\mathbf{p}}_i \text{ and } \mathbf{y}_i = \mathbf{K}^{-1}\hat{\mathbf{p}}'_i$$

After using the above formula, the calculated normalized image coordinates xi and yi are shown below:

```
------------------------------xi-------------------------------------
[[-0.4213934   0.04794284  1.        ]
 [-0.21908907  0.23973179  1.        ]
 [-0.21556209  0.2478906   1.        ]
 ...
 [ 0.12153142 -0.1030586   1.        ]
 [ 0.12214419 -0.13127784  1.        ]
 [ 0.25584784  0.00679026  1.        ]]
------------------------------yi-------------------------------------
[[-0.36586553  0.06318988  1.        ]
 [-0.24224359  0.24057801  1.        ]
 [-0.23837301  0.24857134  1.        ]
 ...
 [-0.11146456  0.11921094  1.        ]
 [ 0.10175154  0.21178286  1.        ]
 [ 0.25753252  0.03006998  1.        ]]
```

Figure: Snapshot of output of calculated xi and yi points

Now, for estimating the Essential Matrix we need to calculate the matrix A from the below formula:

$$\mathbf{y}_i^\top \mathbf{E}\mathbf{x}_i \quad = \quad 0$$

Matrix A is :

$$\begin{bmatrix} x_{i1}y_{i1} & x_{i2}y_{i1} & y_{i1} & x_{i1}y_{i2} & x_{i2}y_{i2} & y_{i2} & x_{i1} & x_{i2} & 1 \end{bmatrix}$$

```
------------------------------A-------------------------------------
[[ 0.15417332 -0.01754063 -0.36586553 ... -0.4213934   0.04794284
   1.        ]
 [ 0.05307292 -0.05807349 -0.24224359 ... -0.21908907  0.23973179
   1.        ]
 [ 0.05138419 -0.05909043 -0.23837301 ... -0.21556209  0.2478906
   1.        ]
 ...
 [-0.01354645  0.01148738 -0.11146456 ...  0.12153142 -0.1030586
   1.        ]
 [ 0.01242836 -0.01335772  0.10175154 ...  0.12214419 -0.13127784
   1.        ]
 [ 0.06588914  0.00174871  0.25753252 ...  0.25584784  0.00679026
   1.        ]]
```

Figure: Snapshot of Computed Matrix A

After obtaining the matrix A, we will use np.linalg.svd(A) for calculating the essential matrix. The essential matrix obtained is given below:

```
----------------------Essential Matrix-------------------------------
[[-9.94799029e-01 -1.00345197e-01 -6.84498097e-03]
 [ 1.34601685e-02  2.47111650e-03  6.40576706e-03]
 [ 4.70694083e-03  4.61568872e-04 -4.25692063e-05]]
```

Figure: Essential Matrix E

- **Decompose the obtained Essential matrix E into the camera motion rotation matrix R and the translation vector t.**

Now, to obtain the Essential matrix E into the camera motion rotation matrix R and the translation vector t, we will implement the Linear Eight-Point Algorithm.
We will update the essential matrix as

$$\mathbf{E} = \mathbf{U}\text{diag}\left(\begin{bmatrix}\sigma & \sigma & 0\end{bmatrix}\right)\mathbf{V}^\top \text{ with } \sigma = \frac{\sigma_1 + \sigma_2}{2}.$$

Then, decompose E as [t] x R as:

$$\begin{aligned}\mathbf{R} &= \mathbf{U}\mathbf{R}_z^\top\left(\pm\frac{\pi}{2}\right)\mathbf{V}^\top \\ [\mathbf{t}]_\times &= \mathbf{U}\mathbf{R}_z\left(\pm\frac{\pi}{2}\right)\Sigma\mathbf{U}^\top.\end{aligned}$$

Therefore, the above formulae is written as following:

```
W=np.array([[0,-1,0],
            [1,0,0],
            [0,0,1]])

U, S, V = np.linalg.svd(Essential_Matrix)
R1, R2, T1, T2 = U @ W @ V, U @ W.transpose() @ V, U[::, 2], -U[::, 2]
```

Figure: snapshot of the formulae for decomposition of Essential matrix

After this step, we are able to find the rotation matrix R and translation vector t. They are shown below in the figure:

```
------------------------------R1-----------------------------------------
[[ 0.01010485 -0.16785126 -0.98576054]
 [-0.9962524  -0.08637678  0.00449548]
 [-0.0859014   0.98202088 -0.16809505]]
------------------------------R2-----------------------------------------
[[-0.01106044  0.1774401   0.98406945]
 [ 0.99393324  0.10964833 -0.00859967]
 [-0.10942749  0.97800422 -0.17757637]]
------------------------------T1-----------------------------------------
[0.0048918  0.01187211 0.99991756]
------------------------------T2-----------------------------------------
[-0.0048918  -0.01187211 -0.99991756]
```

Figure: Snapshot output of camera motion rotation matrix R1, R2 and translation vector t

- **Let Pi be the corresponding 3D point for the pixel pair (pi, p0i). Find Pi, $\forall$ i $\in$ {1, 2, . . . , n} using the triangulation approach learned in the class.**
  For triangulation approach, we need to find out the projection matrix for both the camera 1 and camera 2

```
Projection matrix for camera-1:
[[1.69887376e+03 0.00000000e+00 9.71749770e+02 0.00000000e+00]
 [0.00000000e+00 1.69887966e+03 6.47748827e+02 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00 0.00000000e+00]]
Projection matrix for camera-2:
[[-6.63077926e+01  6.69120456e+02 -1.83802904e+03  9.79980201e+02]
 [-1.74815547e+03  4.89359112e+02 -1.01246089e+02  6.67864705e+02]
 [-8.59013966e-02  9.82020878e-01 -1.68095049e-01  9.99917558e-01]]
[[-0.02806493 -0.15868133 -0.21213601]]
```

Figure: Snapshot output of projection matrix for camera1(i.e M) and camera 2 (i.e N)

Now, using the below formula, we are able to find Pi for i = 1 to n using triangulation approach.

$$\begin{bmatrix} [\hat{\mathbf{p}}_1]_\times \mathbf{M} \\ [\hat{\mathbf{p}}_2]_\times \mathbf{N} \end{bmatrix} \hat{\mathbf{P}} = \mathbf{0} \Rightarrow \mathbf{A}\hat{\mathbf{P}} = \mathbf{0}.$$

The snapshot of calculated Pi is given below:

```
[[ 0.06885794  0.26299302  0.23078098]
 [-0.11729772 -0.1707492   0.22255901]
 [-0.12346449 -0.1658155   0.22126707]
 ...
 [ 0.0604032  -0.06831293  0.08362789]
 [ 0.10214701  0.05963612 -0.09419094]
 [-0.02806493 -0.15868133 -0.21213601]]
```

Figure: Snapshot output of Pi using triangulation approach

- **Plot the obtained Pi, $\forall i \in \{1, 2, \ldots, n\}$ and the camera center t.**
  The plot figure of Pi and camera center t is shown below:

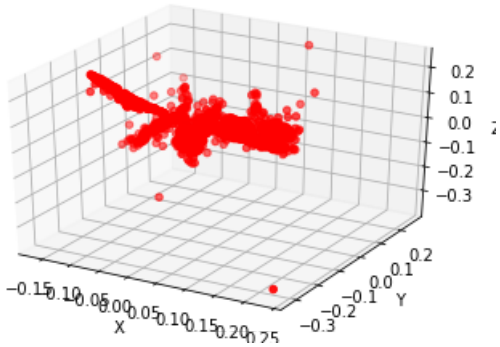


Figure: Snapshot of the plot of Pi, $\forall i \in \{1, 2, \ldots, n\}$