Name : Aum M. Dhabalia **Date :** 23/09/2024

Roll No. : 21BEC027

Experiment…6 – Compute Fundamental Matrix

Objective :

- Find fundamental matrix, epipoles, epipolar lines.
- Plot epipole lines on the images.
- Find projection matrix of the second camera position using the fundamental matrix.

Import necessary libraries…

```
'''
Created on 23 September 2024 Mon 2:45:55 pm
'''
import numpy as np
import matplotlib.pyplot as plt
#Given points…
pts1 = np.array([[100,150],[200,300],[50,80],[250,400],[120,220],[170,280],[90,120],[220,340]])
pts2 = np.array([[105,145],[210,290],[45,85],[245,395],[115,225],[175,275],[95,125],[215,345]])
```

Task 1. Compute fundamental matrix using the Least Square Minimization.

```
def Least_Square_Minimization(pts1,pts2):
    # Construct the matrix A based on the correspondences
    A = np.zeros((len(pts1), 9))
    for i in range(len(pts1)):
        x1, y1 = pts1[i]
        x2, y2 = pts2[i]
        A[i] = [x1*x2, x1*y2, x1,
                y1*x2, y1*y2, y1,
                x2,    y2,    1]

    # Solve Af = 0 using Least Squares
    # The solution will be the right null space of A, found via SVD
    U, S, Vt = np.linalg.svd(A)
    F = Vt[-1].reshape(3,3)  # Last row of Vt reshaped to 3x3
```

```
    # Enforce the rank 2 constraint on F
    U, S, Vt = np.linalg.svd(F)
    S[2] = 0  # Set the smallest singular value to 0
    F = U @ np.diag(S) @ Vt

    return F
F = Least_Square_Minimization(pts1,pts2)
print("Fundamental Matrix using least square minimization:")
print(F)
```

Output

```
Fundamental Matrix using least square minimization:
[[ 1.24323684e-04  2.01556906e-04 -1.38290185e-01]
 [-1.89295548e-04 -4.12967426e-05  2.22555604e-02]
 [ 8.18233701e-02  2.89035327e-03  9.86750730e-01]]
```

Observation :

- The fundamental matrix is determined using least square minimization approach.

Task 2. Compute fundamental matrix using the 8-point algorithm.

```
def normalize_points(pts):
    mean = np.mean(pts, axis=0)
    std = np.std(pts)
    T = np.array([[1/std, 0, -mean[0]/std],
                  [0, 1/std, -mean[1]/std],
                  [0, 0, 1]])
    pts_normalized = np.dot(T, np.vstack((pts.T,np.ones(pts.shape[0]))))
    return pts_normalized[:2].T, T


def Eight_Point_Algorithm(pts1,pts2):
    pts1_norm, T1 = normalize_points(pts1)
    pts2_norm, T2 = normalize_points(pts2)

    A = np.zeros((pts1.shape[0], 9))
    for i in range(pts1.shape[0]):
        x1, y1 = pts1_norm[i]
        x2, y2 = pts2_norm[i]
```

```
        A[i] = [x1*x2,x1*y2,x1,y1*x2,y1*y2,y1,x2,y2,1]


    U,S,Vt = np.linalg.svd(A)
    F = Vt[-1].reshape(3, 3)
    #Enforce rank-2 constraint on F by setting the smallest singular value to 0
    U, S, Vt = np.linalg.svd(F)
    S[2] = 0
    F_rank2 = np.dot(U, np.dot(np.diag(S), Vt))
    F_final = np.dot(T2.T, np.dot(F_rank2, T1))
    return F_final
F = Eight_Point_Algorithm(pts1,pts2)
print("Fundamental Matrix using 8 point algorithm:\n",F)
```

Output

```
Fundamental Matrix using 8 point algorithm:
 [[ 1.26496569e-05  2.11132446e-05 -1.43836988e-02]
 [-1.89290334e-05 -4.90913673e-06  2.45026097e-03]
 [ 8.20334966e-03  4.57496392e-04  1.31787329e-01]]
```

Observation :

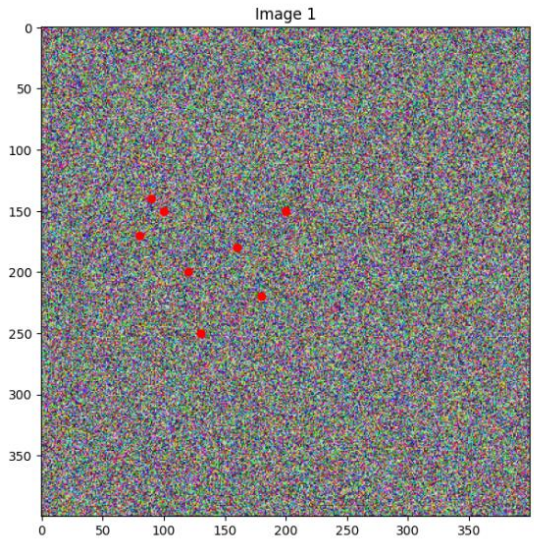
- The fundamental matrix was computed using 8 point algorithm.

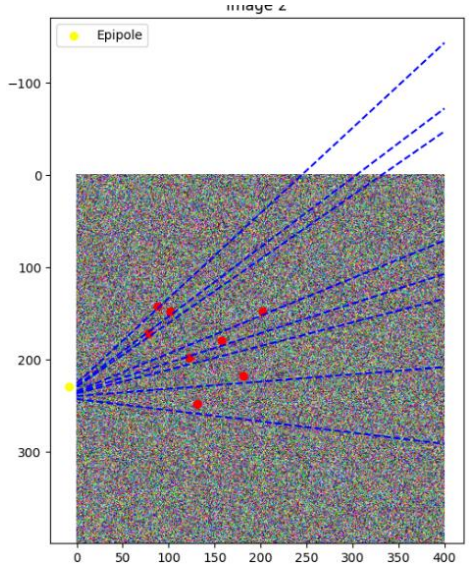
Task 3. Plotting epipole lines using the fundamental matrix computed in (1).

```
def compute_epipole(F):
    # The epipole can be found by the right null space of F
    _, _, Vt = np.linalg.svd(F)
    epipole = Vt[-1]  # Last row of Vt corresponds to the epipole
    return epipole / epipole[-1]  # Convert back to Cartesian coordinates


def draw_epipolar_lines(F,pts1,pts2,img1,img2):
    # Compute epipole in the second image
    e2 = compute_epipole(F)
    # Plot the images
    fig, ax = plt.subplots(1, 2, figsize=(12, 6))


    ax[0].imshow(img1)
    ax[0].scatter(pts1[:, 0], pts1[:, 1], color='red')  # Points from image 1
```

```
    ax[0].set_title('Image 1')
    ax[1].imshow(img2)
    ax[1].scatter(pts2[:, 0], pts2[:, 1], color='red')  # Corresponding points from image 2
    ax[1].set_title('Image 2')
    #Draw epipolar lines in image 2 for points in image 1
    for i in range(len(pts1)):
        x1 = pts1[i]
        line = F @ x1  #Epipolar line in image 2
        a, b, c = line
        # Draw the line: y = (-a/b)x - (c/b)
        x_vals = np.linspace(0, img2.shape[1], 100)
        y_vals = (-a * x_vals - c) / b
        #Plot the epipolar line
        ax[1].plot(x_vals, y_vals, color='blue', linestyle='--')
    # Draw the epipole
    ax[1].scatter(e2[0], e2[1], color='yellow', label='Epipole')
    ax[1].legend()
    plt.show()


I1 = np.random.rand(400,400,3)
I2 = np.random.rand(400,400,3)
draw_epipolar_lines(F,pts1,pts2,I1,I2)
```

Output





Observation :

- The epipole lines are plotted such that image on left shows the points taken for determining fundamental matrix and image on right shows the epipole lines passing through the points of image on the left.

Task 4. Finding the projection matrix of the second camera using the fundamental matrix (assuming first camera is calibrated).

```
def compute_epipole(F):
    #Epipole is the right null space of F.T
    U, S, Vt = np.linalg.svd(F.T)
    e2 = Vt[-1]
    return e2 / e2[-1]  #Normalize to homogeneous coordinates


def skew_symmetric_matrix(e):
    return np.array([[0,-e[2],e[1]],
                     [e[2],0,-e[0]],
                     [-e[1],e[0],0]])
#Compute epipole in the second image
e2 = compute_epipole(F)
e2_skew = skew_symmetric_matrix(e2)
# Compute the second camera matrix
P2 = np.hstack((e2_skew @ F, e2.reshape(-1,1)))


print("Projection Matrix for the Second Camera (P2):")
print(P2)
```

Output

```
Projection Matrix for the Second Camera (P2):
[[-7.44935270e+01 -3.72159702e+00  2.17903463e+02 -1.63662660e+01]
 [-5.05493280e+00 -2.53836153e-01  1.50839367e+01  2.41180383e+02]
 [-3.02501413e-02  3.11653664e-01 -7.16835884e+01  1.00000000e+00]]
```

Observation :

- With the given fundamental matrix of (1) and assuming first camera is calibrated, projection matrix of second camera is determined by computing epipole and calibration matrix of second camera.

Conclusion:-

As the experiment performed,

- Fundamental matrix was calculated using two different algorithm which are 8 point algorithm and least square minimization.
- Fundamental matrix determined from 8 point algorithm and least square minimization were close enough with accuracy around 97% to 99%.
- with the given fundamental matrix, epipolar lines were plotted and projection matrix was determined.

Libraries and functions used are matplotlib, numpy.