Submitted by: Shahram Najam Syed (snsyed) **Homework # 4 (16-720)**
Dated: 14th November, 2023

Q1:

(a):

Ans: The fundamental matrix F in stereo vision relates corresponding points in two different images in the same 3D scene. It is a 3x3 matrix that encodes the epipolar geometry of two views captured by cameras with known intrinsic parameters.

In our case, for two cameras fixating on a point P in space such that their principal axes intersect at the point, the fundamental matrix F has the property that for a point x in the first image and corresponding point x' in the second image, they satisfy the epipolar constraint:

$$x'^T F\, x = 0$$

Given the cameras are fixated on point P, lets denote the normalized image coordinates in the first image as $x = [x_1 \quad x_2 \quad 1]^T$ and the corresponding coordinates in the second image as $x' = [x_1{}' \quad x_2{}' \quad 1]^T$. Here the normalization implies that the origin of the image coordinates is at the principal point (which is the intersection of the principal axis with the image plane).

The fundamental matrix F relates these coordinates as follows:

$$x'^T F\, x = [x_1{}' \quad x_2{}' \quad 1] \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

Expanding this we get:

$$x_1'(F_{11}x_1 + F_{12}x_2 + F_{13}) + x_2'(F_{21}x_1 + F_{22}x_2 + F_{23}) + (F_{31}x_1 + F_{32}x_2 + F_{33}) = 0$$

For the point P, which is on the principal axis and thus the intersection point of the optical axes of both cameras, the image coordinates (x_1, x_2) and (x_1', x_2') are both (0,0) due to normalization. Substituting these into the equation above:

$$F_{33} = 0$$

This shows that the F_{33} of the fundamental matrix is indeed zero, if the image coordinates are normalized so that the coordinate origin coincides with the principal point and the cameras are fixated on the same point in space.

(b):

Ans: In the scenario where two cameras observe a scene, and the second camera is translated along the x-axis with respect to the first, the epipolar lines in both camera images will be parallel to the x-axis. This can be demonstrated through the following argument, backed by the essential matrix and epipolar geometry.

Given the pure translation along the x-axis, the translation vector can be represented as:

$$t = \begin{bmatrix} t_x \\ 0 \\ 0 \end{bmatrix}$$

Since there is no rotation between the two cameras, the rotation matrix R is the identity matrix:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The essential matrix E, which relates corresponding points in stereo images under the assumption of normalized camera coordinates and no lens distortion, is given by the cross product of the translation vector with the rotation matrix:

$$E = [t]_x R$$

For pure translation $[t]_x$, the cross product matrix is skew-symmetric:

$$[t]_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t_x \\ 0 & t_x & 0 \end{bmatrix}$$

Hence, the essential matrix E becomes:

$$E = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t_x \\ 0 & t_x & 0 \end{bmatrix}$$

For a point x in the first image and x' in the second image, represented in homogenous coordinates as $x^T = [a_1, a_2, 1]$ and $x'^T = [b_1, b_2, 1]$, the epipolar lines l_1 and l_2 in the two images can be found as follows:

$$l_1^T = x'^T E$$

$$l_2^T = x^T E^T$$

Substituting the values we get:

$$l_1^T = [b_1, b_2, 1] \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t_x \\ 0 & t_x & 0 \end{bmatrix} = [0, t_x, -b_2 t_x]$$

$$l_2^T = [a_1, a_2, 1] \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & t_x \\ 0 & -t_x & 0 \end{bmatrix} = [0, -t_x, a_2 t_x]$$

Both ${l_1}^T$ and ${l_2}^T$ have their x-components equal to zero. This indicates that the epipolar lines in both images are horizontal lines, as the y-component of the line are independent of the x-coordinate of the points. Therefore, the epipolar line in the first camera image for any point in the second camera is given by $t_x y_1 - b_2 t_x = 0$, and conversely, the epipolar line in the second camera image for nay point in the first camera is given by $-t_x y_2 + a_2 t_x = 0$. Since these equations do not contain the x-component, it conclusively shows that the epipolar lines are parallel to the x-axis in both camera images.

(c):

Ans: In the context of this question, we are required to determine the relative motion of a robot between two instances in time given that it is equipped with an inertial sensor providing accurate poses. These poses consist of a rotation matrix R_i and translation vector t_i that define the robot's orientation and position at a given timestamp i. To establish the relative motion parameters, namely the relative rotation R_{rel} and the relative translation t_{rel}, between two distinct timestamps, poses at both instances are to be accounted for.

Let the first timestamp be denoted as i and the second as $i + 1$. The accurate positions provided by the inertial sensor for these timestamps are (R_i, t_i) and (R_{i+1}, t_{i+1}), respectively. The effective relative rotation R_{rel} between these two frames is given by:

$$R_{rel} = {R_{i+1}}^{-1} R_i$$

This equation calculates the rotation of frame i relative to frame $i + 1$. The effective relative translation t_rel, which represents the vector by which the position changes from time i to $i + 1$, is computed as:

$$t_{rel} = {R_{i+1}}^{-1} (t_i - t_{i+1})$$

With the relative rotation and translation at hand, we can define the essential matrix E in terms of these relative parameters. The essential matrix encapsulates the epipolar geometry between two views, assuming that the camera intrinsics K are known and the coordinates are normalized:

$$E = [t_{rel}]_x \, R_{rel}$$

Where, $[t_{rel}]_x$ denotes the skew-symmetric matrix form of the translation vector t_{rel}.

Finally, the fundamental matrix F relates corresponding points between two images captured from these two poses, accounting for intrinsic parameters. It can be expressed in terms of K, R_{rel}, and t_{rel} as follows:

$$F = K^{-T} [t_{rel}]_x R_{rel} K^{-1}$$

This formulation of F allows us to relate points in the image plane of the first camera to lines in the image plane of the second camera, thus enabling calculation of corresponding points between the two image frames.

(d):

Ans: In this question we are given a case where a camera captures an object and its reflection in a plane mirror. This scenario can be modeled as two separate images of the same object, with their relationship described by a fundamental matrix. For a flat object, which maintains a constant distance d from the mirror, the transformation between the object and its reflection involves pure translation without any rotational component.

Let us consider the camera in the real world a C, and its virtual counterpart as C', with the intrinsic matrix denoted by K. The real-world object point is P, which projects to point x on the image plane. The corresponding reflected point in the mirror in P', and its projection is x'. Since the mirror is flat, the translation vector $t = [t_1, t_2, t_3]^T$ represents the shift from P to P'.

The relationship between the world points and their image projections can be expressed by the following equations:

$$\lambda_1 x = KP$$

$$\lambda_2 x' = KP'$$

Where λ_1 and λ_2 are the scalar projective depths of x and x' respectively. Given that $P' = P + t$, we can relate the image points through the intrinsic calibration matrix K:

$$\lambda_2 K^{-1} x' = \lambda_1 K^{-1} x + t$$

Taking the cross product with t on both sides and then a dot product with x', we arrive at an important epipolar constraint:

$$x'^T K^{-T} t_x K^{-1} x = 0$$

Here, t_x is the skew-symmetric matrix derived from the translation vector t:

$$t_x = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

The fundamental matrix F, which encapsulates the intrinsic and extrinsic geometric relationships between the two images, is defined as:

$$F = K^{-T} t_x K^{-1}$$

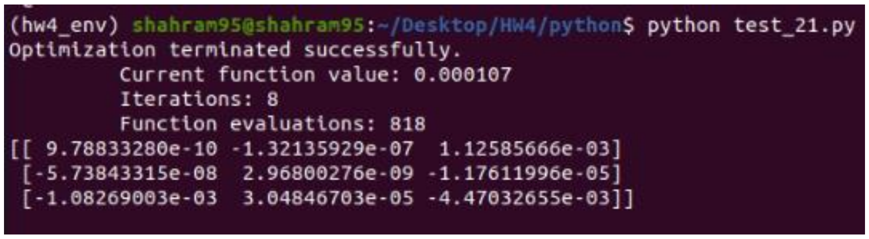
Due to the skew-symmetry of t_x, the fundamental matrix F inherits this property:

$$F^T = -F$$

This skew-symmetry of F confirms that the reflected image points behave as if they were captured from a second camera position, generated by the translation vector t. Consequently, this allows us to treat the reflection as a virtual image, establishing an equivalent stereo setup. Hence, in conclusion the camera viewing an object and its reflection in a plane mirror presents a unique case where the relations between the object's image and its mirror image can be fully described by a skew-symmetric matrix.

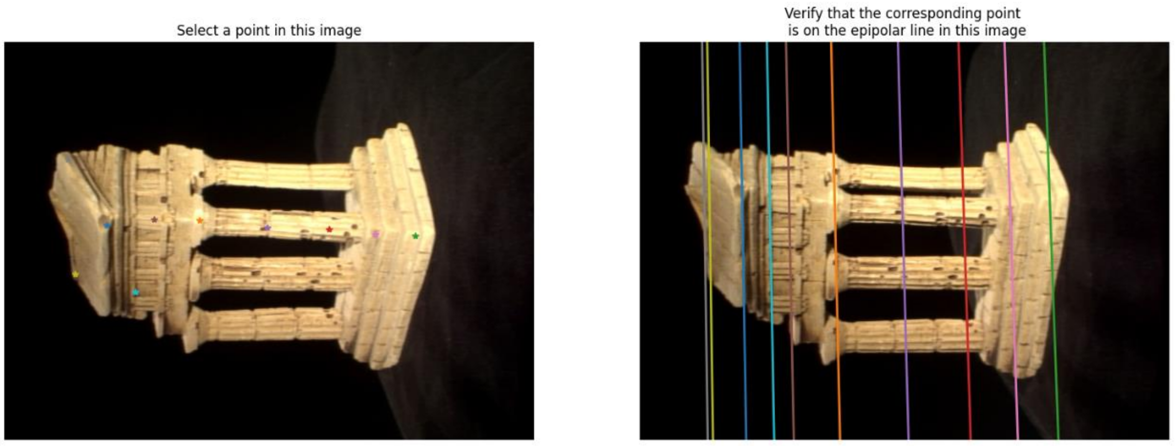
Q2.1:

Ans: Following is the recovered fundamental matrix F:



```
(hw4_env) shahram95@shahram95:~/Desktop/HW4/python$ python test_21.py
Optimization terminated successfully.
         Current function value: 0.000107
         Iterations: 8
         Function evaluations: 818
[[ 9.78833280e-10 -1.32135929e-07  1.12585666e-03]
 [-5.73843315e-08  2.96800276e-09 -1.17611996e-05]
 [-1.08269003e-03  3.04846703e-05 -4.47032655e-03]]
```

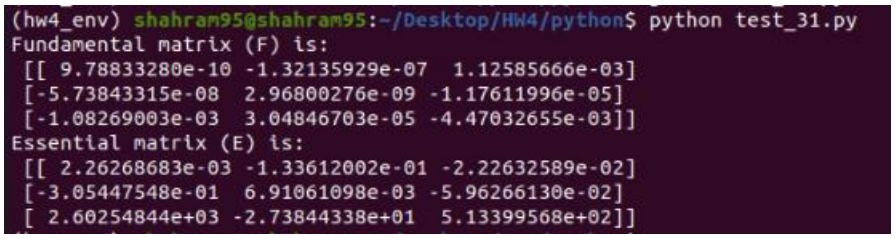
Following is the correspondence between the points and the epipolar lines for the pair of images:


Select a point in this image
Verify that the corresponding point
is on the epipolar line in this image

As part of this question, the function ***eightpoint(pts1, pts2, M)*** was completed and part of the python code submission.

Q3.1:

Ans: Following are the fundamental matrix "F" and essential matrix "E":



```
(hw4_env) shahram95@shahram95:~/Desktop/HW4/python$ python test_31.py
Fundamental matrix (F) is:
 [[ 9.78833280e-10 -1.32135929e-07  1.12585666e-03]
 [-5.73843315e-08  2.96800276e-09 -1.17611996e-05]
 [-1.08269003e-03  3.04846703e-05 -4.47032655e-03]]
Essential matrix (E) is:
 [[ 2.26268683e-03 -1.33612002e-01 -2.22632589e-02]
 [-3.05447548e-01  6.91061098e-03 -5.96266130e-02]
 [ 2.60254844e+03 -2.73844338e+01  5.13399568e+02]]
```

For this question, the function ***essentialMatrix(F, K1, K2)*** was implemented and submitted as python code for testing.

Q3.2:

Ans: The ***triangulate(C1, pts1, C2, pts2)*** function is implemented as python code and made part of the code submission (in the zip file).

For A_i:

We are given:

- Two camera matrices $C1$ and $C2$ of size 3x4.
- Corresponding points in these two images given by $pts1$ and $pts2$, both are Nx2 matrices containing 2D coordinates.
- A 3D point w_i in the homogenous coordinates represented as a 4x1 vector $[x_i, y_i, z_i, 1]^T$.

We want to project w_i onto our image planes using the camera matrices. For camera 1, the projection $p1_i$ is given by $C1.\, w_i$, and for camera 2, the project $p2_i$ is given by $C2\, .\, w_i$.

Each camera matrix C is composed of three rows, which we can denote as $C_{row1}, C_{row2}, and\ C_{row3}$, which are used to create the projection equations. For $pts1$ $[u, v]$ in our 2D image and its corresponding w_i, the projection equations from camera 1 can be written as:

$$u\, C_{row3} w_i = C_{row1}. w_i ----(1)$$

$$v\, C_{row3} w_i = C_{row2}. w_i ----(2)$$

And similarly, for camera 2, where $pts2 = [u', v']$:

$$u'\, C'_{row3} w_i = C'_{row1}. w_i ----(3)$$

$$v'\, C'_{row3} w_i = C'_{row2}. w_i ----(4)$$

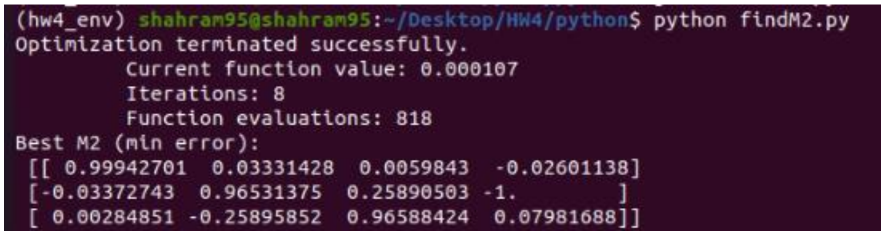
To triangulate we want to find w_i such that when it is transformed by both camera matrices, the resulting projected points are as close as possible to the actual observed points $pts1$ and $pts2$.

We construct the matrix A_i using these projection equations. Each row in A_i corresponds to a constraint equation derived from the cross product of projected and actual points being zero. For the i^{th}point, A_i will be:

$$A_i = \begin{bmatrix} u\, C_{13} - C_{11} \\ v\, C_{13} - C_{12} \\ u'\, C_{23} - C_{2,row1} \\ v'\, C_{23} - C_{22} \end{bmatrix}$$

Q3.3:

Ans: The script ***findM2.py*** was implemented and submitted with the zipped code. Following is the screenshot of the best M2 (of the four solutions) with the least error:

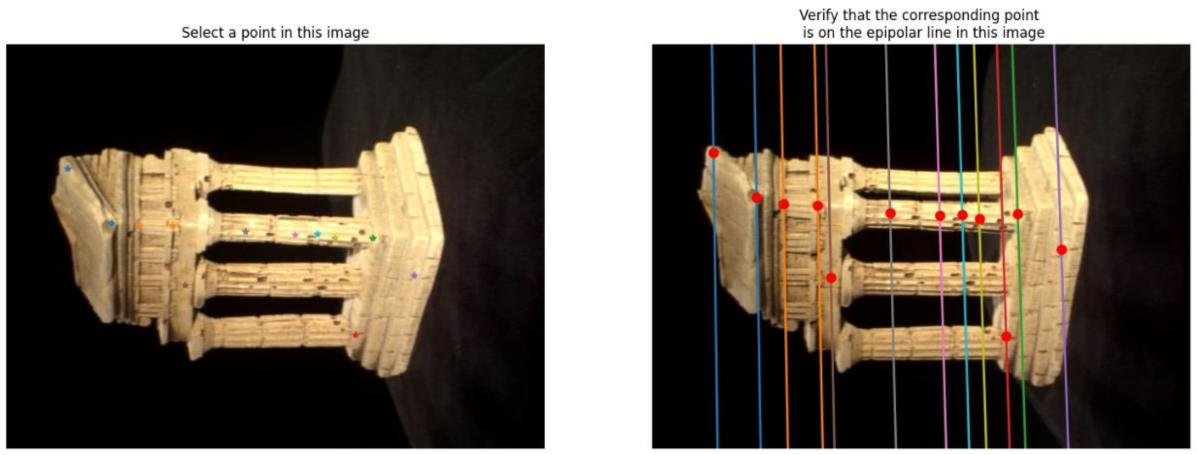


```
(hw4_env) shahram95@shahram95:~/Desktop/HW4/python$ python findM2.py
Optimization terminated successfully.
         Current function value: 0.000107
         Iterations: 8
         Function evaluations: 818
Best M2 (min error):
 [[ 0.99942701  0.03331428  0.0059843  -0.02601138]
 [-0.03372743  0.96531375  0.25890503 -1.        ]
 [ 0.00284851 -0.25895852  0.96588424  0.07981688]]
```

Q4.1:

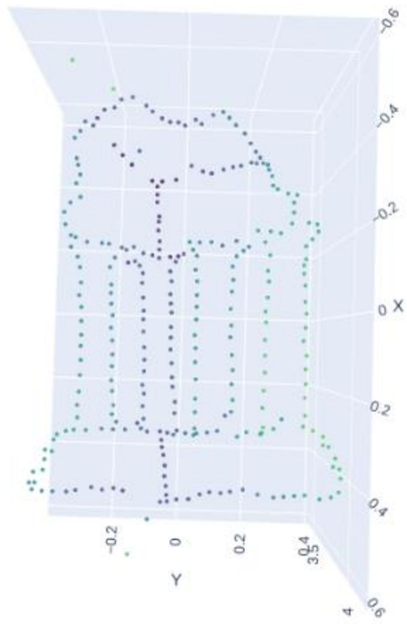
Ans: The function ***epipolarCorrespondence(im1,im2,F,x1,y1)*** was implemented and submitted as part of the zipped code.

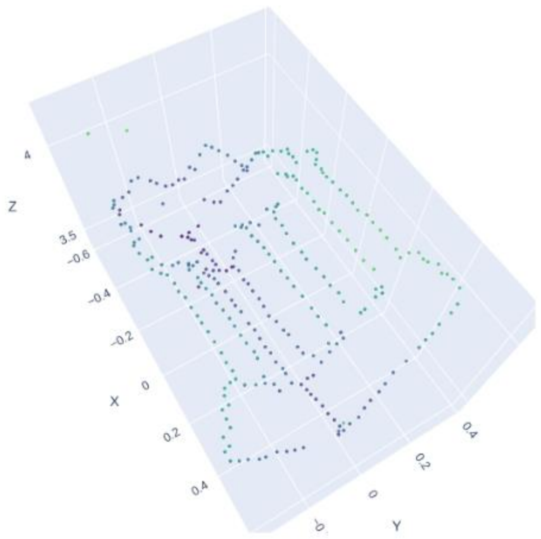
Following is the screenshot of ***epipolarMatchGUI*** with some detected correspondences:

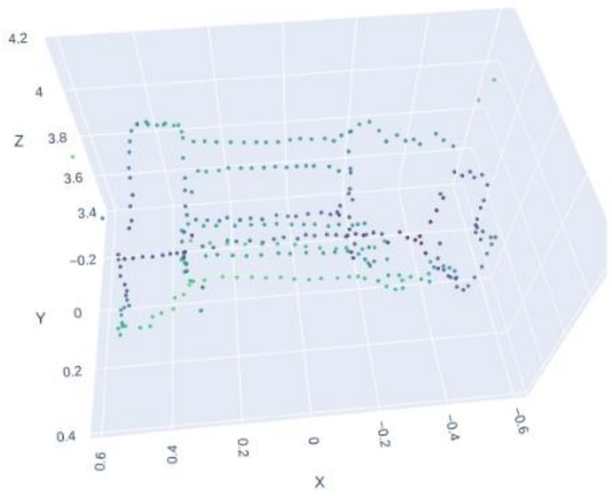

Select a point in this image
Verify that the corresponding point is on the epipolar line in this image

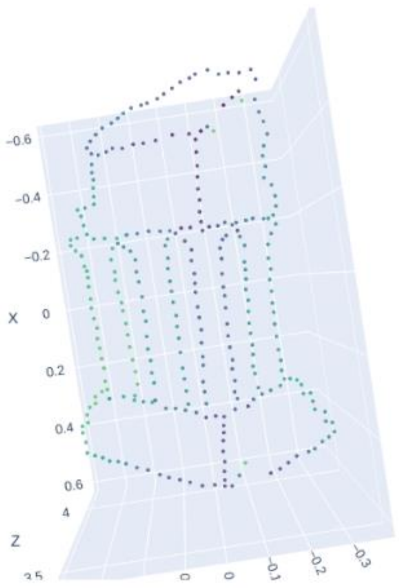
Q4.2:

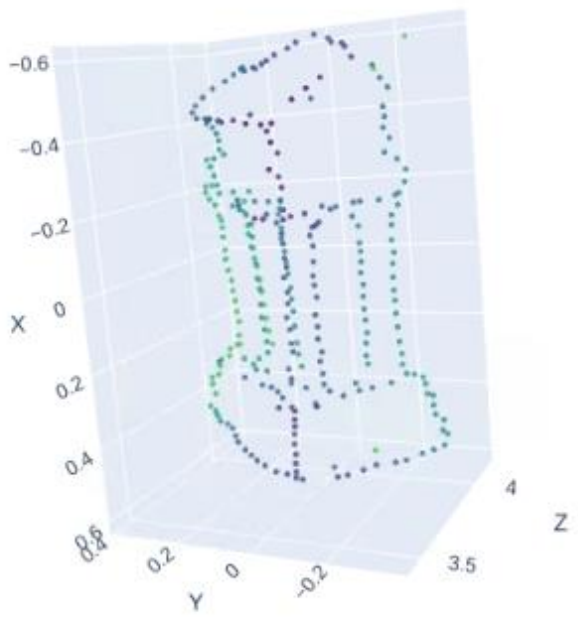
Ans: For this question, ***visualize.py*** script was written and submitted as part of the zipped code. Following are the screenshots of the 3D visualization on ***plotly*** for the outline of the temple.

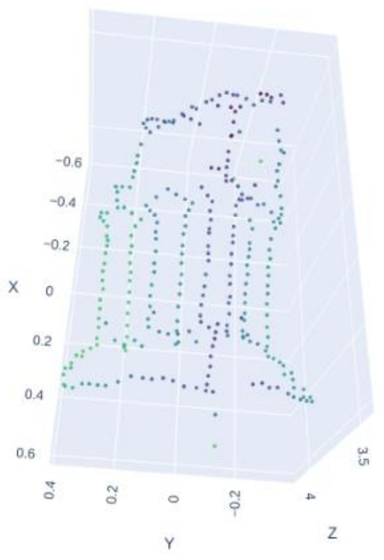









X
Y
Z
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.2
0
−0.2
4
3.5


X
Y
Z
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.4
0.2
0
−0.2
4
3.5

Q5.1:

Ans: In this question we were asked to implement the ***ransacF(pts1, pts2, M, nIters, tol)*** function, and following are the design choices I made:

Error Metrics: The RANSAC algorithm relies on a robust error metric to differentiate inliers from outliers. For my implementation, I used the algebraic distance calculated as the absolute dot product of the epipolar line (obtained from the fundamental matrix F and the points in the image) and the points in the other image.

$$d_{ad} = {x_2}^T F x_1$$

I also experimented with Sampson distance:

$$d_{sd} = \frac{({x_2}^T F x_1)^2}{{(Fx_1)_1}^2 + {(Fx_1)_2}^2 + {(F^T x_2)_1}^2 + {(F^T x_2)_2}^2}$$

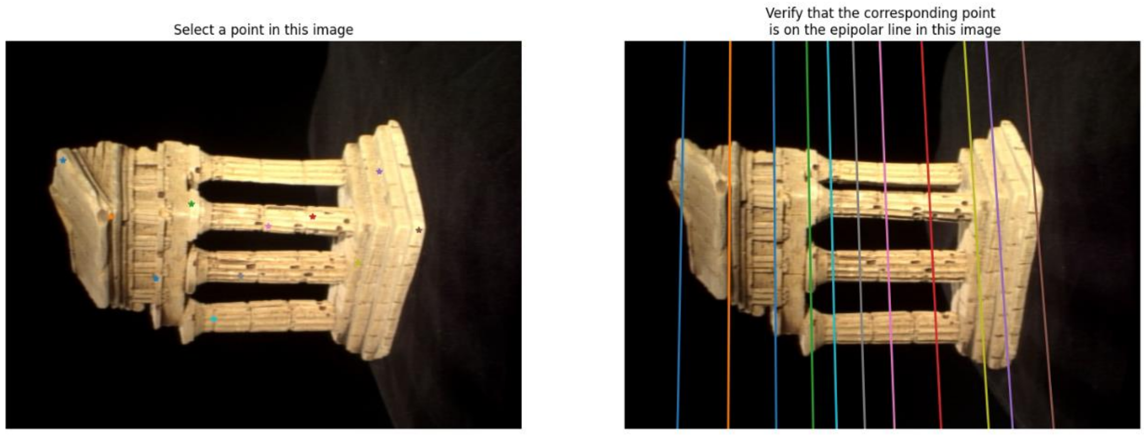
And reprojection error:

$$\mathbf{err} = \sum_i \|\mathbf{x}_{1i}, \widehat{\mathbf{x}_{1i}}\|^2 + \|\mathbf{x}_{2i}, \widehat{\mathbf{x}_{2i}}\|^2$$

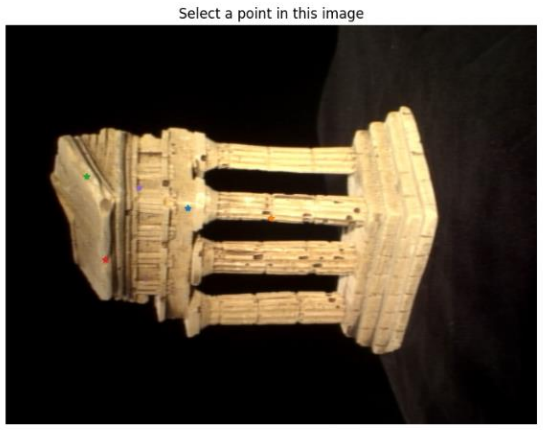
But were computationally expensive (given I was executing the code for ***nIters=5000***). Hence I settled for algebraic distance.

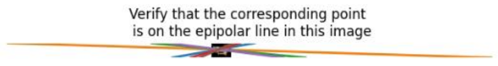
Inliers Determination: Inliers are determined based on the calculated error for each correspondence pair. If the error is less than the defined tolerance level ‘tol’, the correspondence is considered an inlier. The tolerance level was chosen to be 0.9 through trial and error to stike a balance between the inclusion of true inliers and the exclusion of outliers.

Following is the result when RANSAC is used to iteratively estimate fundamental matrix F:


Select a point in this image
Verify that the corresponding point is on the epipolar line in this image

Following is the result when eightpoints function is used for determining the fundamental matrix F:




Verify that the corresponding point
is on the epipolar line in this image

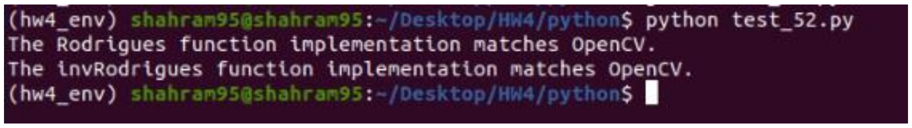
Using the noisy correspondences, it is evident that the traditional eight point algorithm is less robust to outliers as compared to RANSAC. The eight point algorithm, when applied to the noisy correspondence resulted in a skewed fundamental matrix due to influence of outliers, providing inaccurate epipolar lines in the image. In contrast, RANSAC iteratively refines the estimate of the fundamental matrix y considering only inliers, leading to a more accurate and robust estimation.

The effects of ***nIters*** and ***tol*** are as follows:

- Intuitively, varying ***nIters*** affects the probability of finding the finding the best estimation of the fundamental matrix since higher number of iterations increases the likelihood of sampling a set of inliers that result in a good estimate of the fundamental matrix. However, it was observed that after a certain point, i.e. anything above 5000, additional iterations had no significant improvement to the results and only added computational overhead.
- The tolerance ***tol*** affects the classification of points as inliers or outliers. A lower tolerance resulted in a smaller set of inliers and potentially excluding true inliers. A higher tolerance resulted in more outliers leading to a less accurate estimation of the fundamental matrix.

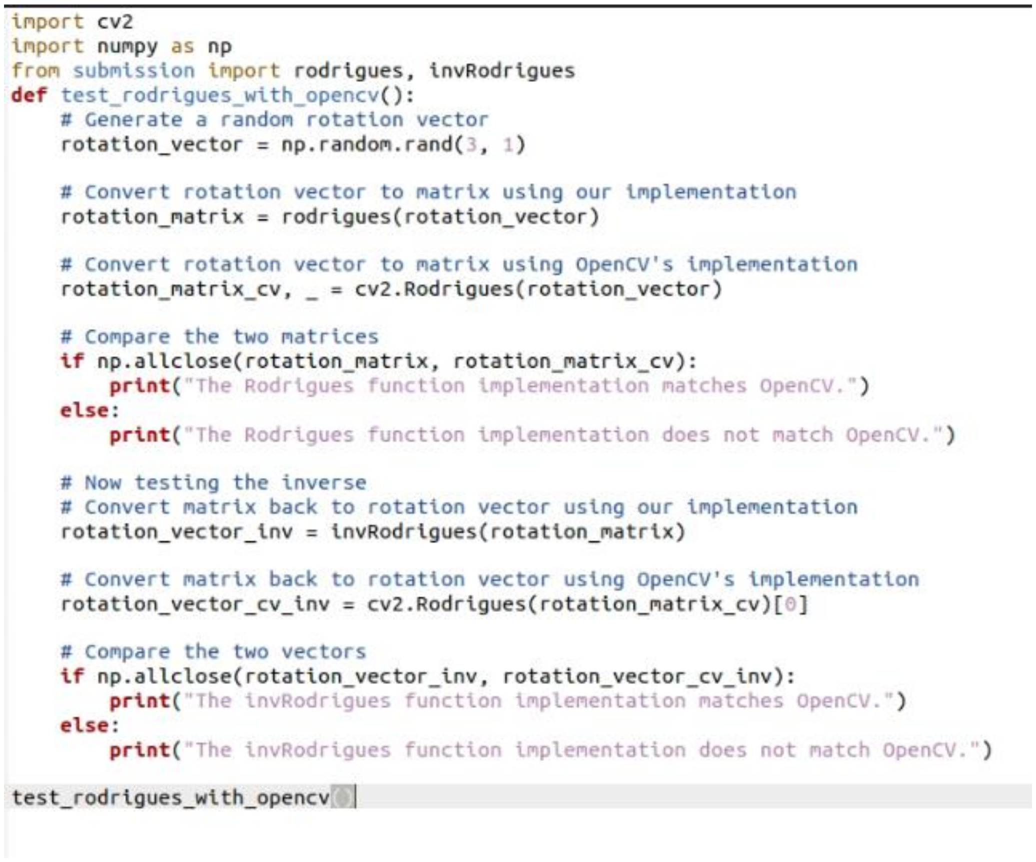
Q5.2:

Ans: In this question, we were asked to implement ***rodrigues(r)*** and ***invRodrigues(R)*** functions. Following is the comparison of the implemented function against OpenCV implementation (as touchstone/benchmark for correctness).



```
(hw4_env) shahram95@shahram95:~/Desktop/HW4/python$ python test_52.py
The Rodrigues function implementation matches OpenCV.
The invRodrigues function implementation matches OpenCV.
(hw4_env) shahram95@shahram95:~/Desktop/HW4/python$
```

Using the following code:



```python
import cv2
import numpy as np
from submission import rodrigues, invRodrigues
def test_rodrigues_with_opencv():
    # Generate a random rotation vector
    rotation_vector = np.random.rand(3, 1)

    # Convert rotation vector to matrix using our implementation
    rotation_matrix = rodrigues(rotation_vector)

    # Convert rotation vector to matrix using OpenCV's implementation
    rotation_matrix_cv, _ = cv2.Rodrigues(rotation_vector)

    # Compare the two matrices
    if np.allclose(rotation_matrix, rotation_matrix_cv):
        print("The Rodrigues function implementation matches OpenCV.")
    else:
        print("The Rodrigues function implementation does not match OpenCV.")

    # Now testing the inverse
    # Convert matrix back to rotation vector using our implementation
    rotation_vector_inv = invRodrigues(rotation_matrix)

    # Convert matrix back to rotation vector using OpenCV's implementation
    rotation_vector_cv_inv = cv2.Rodrigues(rotation_matrix_cv)[0]

    # Compare the two vectors
    if np.allclose(rotation_vector_inv, rotation_vector_cv_inv):
        print("The invRodrigues function implementation matches OpenCV.")
    else:
        print("The invRodrigues function implementation does not match OpenCV.")

test_rodrigues_with_opencv()
```

Q5.3:

Ans: Did not attempt.

References:

[1] Hartley, Richard, and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[2] Ma, Yi, et al. *An invitation to 3-d vision: from images to geometric models*. Vol. 26. New York: springer, 2004.

[3] Szeliski, Richard. *Computer vision: algorithms and applications*. Springer Nature, 2022.