Students:
Mir Mohibullah Sazid [sazidarnob@gmail.com]
Syma Afsha [symaafsha.eece@gmail.com]

Lab 3 : Epipolar Geometry and Stereo

Introduction

Epipolar geometry forms the basis of understanding the correlation among multiple views of a three-dimensional (3D) space that are observed with distinct cameras. The geometrical relations among the 2D projections of the 3D points in a scene onto the image planes of the camera and the actual 3D points, this is what the algorithm deals with. In a stereo vision setup, every 3D point in a scene lies on its particular epipolar plane that passes through both camera centres. This plane, penetrating for each camera's picture plane, creates epipolar lines. The location of a point on a picture plane determines the position where its counterpart point should occur along the epipolar line on the other picture.
Essential for understanding epipolar geometry is knowing the epipoles, or the points on the image planes where the lines connecting the camera centers intersect. Each camera epipole (where the other one is projected into) is its image plane.
The main principle behind epipolar geometry is an 3X3 matrix, known as the fundamental matrix. This matrix links corresponding points on different images, and it collects each and every intrinsic projection information that holds two cameras together in a stereo vision system. It is crucial for tasks such as stereo matching and 3D reconstruction. The basic matrix is usually constructed using image correspondences, namely pairs of points which represent a single three-dimensional point. The matrix is a rank 2 matrix even though its computation becomes very complex and often involving the 8-point algorithm but its determinant value being zero directed that its is not Full Rank.
Noise significantly affects the accuracy of epipolar geometry in stereo vision, including the computation of epipolar lines and epipoles. In stereo vision systems, noise usually appear as errors in the image point coordinate detection process. Because they are utilized to calculate the fundamental matrix, these points are essential to determining the epipolar geometry. The epipolar lines, which are necessary for accurately mapping corresponding spots across stereo images, are defined in large part by this matrix. Incorrect epipolar lines result from the direct impact of noise on these image points on the fundamental matrix's correctness. Such errors in the epipolar lines can seriously impair activities such as stereo matching, where the ability to precisely correspond two images is important.Additionally, it also affects the computation of epipoles, which are the points at which the line that connects the camera centers and the image planes intersects. The understanding of the scene geometry by the stereo vision system can be further affected by inaccurately estimated epipoles resulting from noisy data.
Furthermore, the impact of noise extends beyond the distortion of epipolar lines and epipoles, it

also permeates further procedures such as 3D reconstruction. Reconstructing the 3D structure of a scene from its 2D projections requires triangulation, which is logically inaccurate when the epipolar geometry is gone. Robust statistical techniques like RANSAC (Random Sample Consensus) are frequently used to address these issues. By recognizing and ignoring data outliers, these techniques reduce the influence of noise on the basic matrix computation. To put it briefly, noise management is critical to preserving the integrity of epipolar geometry and providing consistent results in stereo vision applications.
All things considered, epipolar geometry and the fundamental matrix offer an organized method of comprehending how 3D points in a scene are projected onto 2D image planes and how these projections can be correlated between various viewpoints to accomplish various tasks like motion tracking, object recognition in a multi-camera setup, and 3D reconstruction.

Objectives

In this lab exercise, we have understood the concept of Epipolar Geometry through implementing the concepts in MATLAB. Our contributions for this lab are the followings:

1. We have constructed a Fundamental matrix analytically from two simulated cameras and also their transformation matrices.

2. We have computed the Fundamental matrix by using the 8-point method and compared both fundamental matrices.

3. We have drawn the epipolar geometry in both image planes (points, epipoles and epipolar lines).

4. We have added and increased the noise in 2D points and repeated the computation.

Finally, we have checked the consistency of the epipolar geometry obtained.

Implementation

The task was completed successfully in a series of steps, which are listed below in chronological order.

Step 1

camera 1 is defined with the following parameters and set the world coordinate system to the coordinate system of camera 1 (i.e., Rotation = Identity and Translation = 0). The figure 1 Shows the code of step 1.

Step 2

Camera 2 is defined with respect to camera 1. The figure shows the step 2

```
%% Step 1
% Camera 1
au1 = 100; av1 = 120; uo1 = 128; vo1 = 128;
imageSize = [256 256];
R_c1 = eye(3); % 3x3 Identity matrix
t_c1 = zeros(3,1); % Translation vector [0; 0; 0]
```

Figure 1: MatLAB Code for Step 1

```
%% Step 2
% Camera 2
au2 = 90; av2 = 110; uo2 = 128; vo2 = 128;
ax = 0.1; by = pi/4; cz = 0.2; % XYZ EULER
tx = -1000; ty = 190; tz = 230;
t_c2 = [tx; ty; tz]; % Translation vector for camera 2
% Construct rotation matrix for camera 2 using XYZ Euler angles
Rx = [1, 0, 0; 0, cos(ax), -sin(ax); 0, sin(ax), cos(ax)]; % Rotation around x-axis
Ry = [cos(by), 0, sin(by); 0, 1, 0; -sin(by), 0, cos(by)]; % Rotation around y-axis
Rz = [cos(cz), -sin(cz), 0; sin(cz), cos(cz), 0; 0, 0, 1]; % Rotation around z-axis
% The rotation matrix for camera 2 is the product of the rotations
R_c2 = Rx * Ry * Rz;
```

Figure 2: Code for step 2

```
%% STEP 3
% Compute intrinsic matrices and projection matrices

K1 = [au1 0 uo1; 0 av1 vo1; 0 0 1]; % Intrisics matrix for camera 1
wR1c = eye(3);   % rotation of camera 1, from the camera to the world coordinate frame
wt1c = [0 0 0]'; % translation of camera 1, from the camera to the world coordinate frame

% Note: ************** You have to add your own code from here onward ************
K2 = [au2, 0, uo2; 0, av2, vo2; 0, 0, 1]
T_c1w = [R_c1, t_c1; 0, 0, 0, 1]
% Transformation matrix from camera 2 to camera 1
T_c2c1 = [R_c2, t_c2; 0, 0, 0, 1];

% Transformation matrix from camera 2 to world
T_c2w = T_c1w * T_c2c1;

% The inverse of transformation from camera to world is the transformation from world to camera
T_w2c = inv(T_c2w);

% Projection matrix P for camera 1 (since camera 1 is aligned with the world frame)
P1 = K1 * [eye(3), zeros(3,1)];

% Projection matrix P for camera 2
P2 = K2 * T_w2c(1:3,:);
```

Figure 3: Camera Two Rotation and Translation (Step 3)

```
K1 =

   100     0   128
     0   120   128
     0     0     1

>> K2

K2 =

    90     0   128
     0   110   128
     0     0     1

>> P1

P1 =

   100     0   128     0
     0   120   128     0
     0     0     1     0

>> P2

P2 =

   1.0e+05 *

    0.0015    0.0001    0.0003    1.4318
    0.0008    0.0010    0.0012    0.2996
    0.0000   -0.0000    0.0000    0.0056
```

Figure 4: Intrinsic Matrix and Projection Matrix for Camera 1 and 2

Step 3

The intrinsic matrices of both cameras, and the rotation and translation between both cameras and the two camera projection matrices P1 and P2 is computed. Figure 3 shows the code The coordinate system of the first camera is similar to the world and for this, the rotation is an identity matrix and translation is 0. On the other hand, for camera two, the coordinate is rotated and translated from camera one. The following code snippet (Figure 4) shows the corresponding rotation and translation matrix.

Step 4

After that, we have computed the fundamental matrix as per the instruction of the lab guide. (refer to Figure **??**).

```
%% STEP 4


% First, calculate the skew-symmetric matrix of t_c2
t_x = [0, -tz, ty; tz, 0, -tx; -ty, tx, 0];

% Then calculate the Fundamental matrix F1
F1 = inv(K2).' * R_c2.' * t_x * inv(K1);
F1 = F1./F1(3,3);
% Display the Fundamental matrix
disp('Fundamental matrix F:');
disp(F1);
```

(a) Calculation of Fundamental matrix

```
Fundamental matrix F:
    0.0000    0.0001   -0.0066
    0.0000   -0.0000    0.0130
   -0.0096   -0.0173    1.0000
```

(b) Output of fundamental matrix

Figure 5: Step 4

The fundamental matrix suggests a minor translation and rotation between two camera views with the majority of the transformation encapsulated in the scale factor of 1.0000. Small non-zero off-diagonal elements hint at a stereo setup.

Step 5

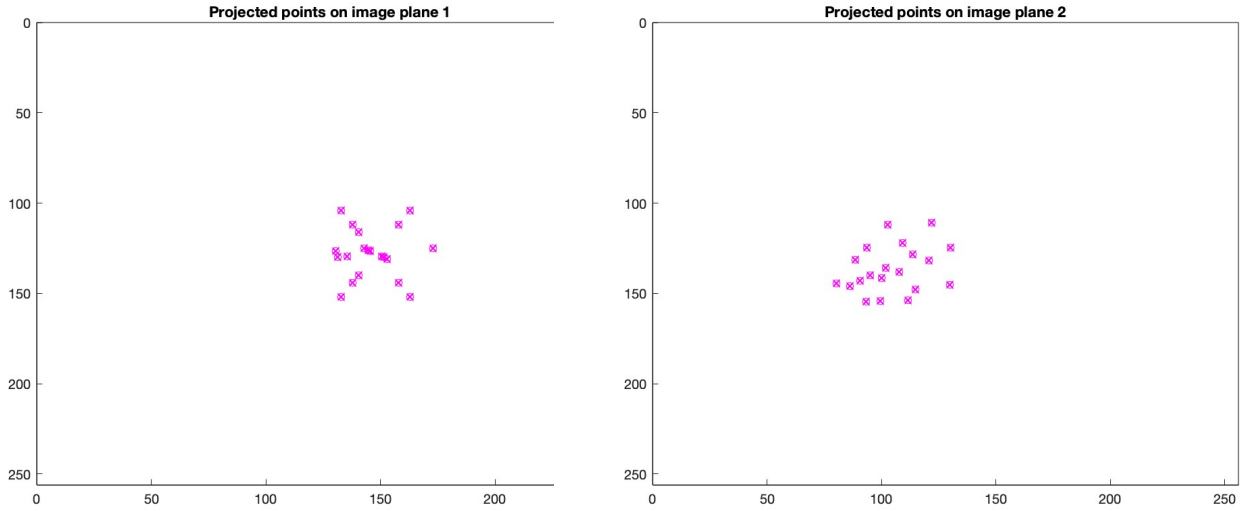
For this step, a set of 20 3D points have been defined with respect to the world coordinate system.

Step 6

With the help of mvg-projectPointToImagePlane function we have passed the 3D points with the projection matrices P1 and P2 to compute the corresponding 2D projections.

Step 7

The computed 2D projection points are plotted in Figure **??**



(a) 2D projected points (b) 2D projected points

Figure 6: Step 7 Graph

Step 8

From the lecture notes, we have retrieved the pseudo code of 8 point method and SVD to calculate the fundamental matrix. (as depicted in Figure **??**)

```
%step 8

[~,numcol] = size(V);

% Create U_new using a more compact matrix operation
U_new = [cam2_p2d(1,:)'.*cam1_p2d(1,:)' cam2_p2d(1,:)'.*cam1_p2d(2,:)' ...
    cam2_p2d(1,:)' cam2_p2d(2,:)'.*cam1_p2d(1,:)' cam2_p2d(2,:)'.*cam1_p2d(2,
    cam2_p2d(2,:)' cam1_p2d(1,:)' cam1_p2d(2,:)' ones(numcol,1)];


% Compute SVD of U_new
[~, ~, V_eight] = svd(U_new);

% Reshape the last column of V_eight to form F1
F= reshape(V_eight(:,9), 3,3)';
% Compute SVD of F1
[U_one, D_one, V_one] = svd(F);
% Enforce rank-2 constraint on F1
F = U_one*diag([D_one(1,1) D_one(2,2) 0])*V_one';
```

(a) Calculation of 8-point method

```
F =

    0.0000    0.0001   -0.0066
    0.0000   -0.0000    0.0130
   -0.0096   -0.0173    0.9997
```

(b) Output

Figure 7: Step 8

Step 9

We have compared the fundamental matrix and the 8 point fundamental matrix using the sum of absolute difference between each value of the matrix in this step. And the error is depicted in Figure In the context of epipolar geometry, the Sum of Absolute Differences (SAD) is often used to measure the dissimilarity between corresponding pixels in stereo images. The SAD is typically used in stereo matching algorithms to find corresponding points in left and right images. A lower

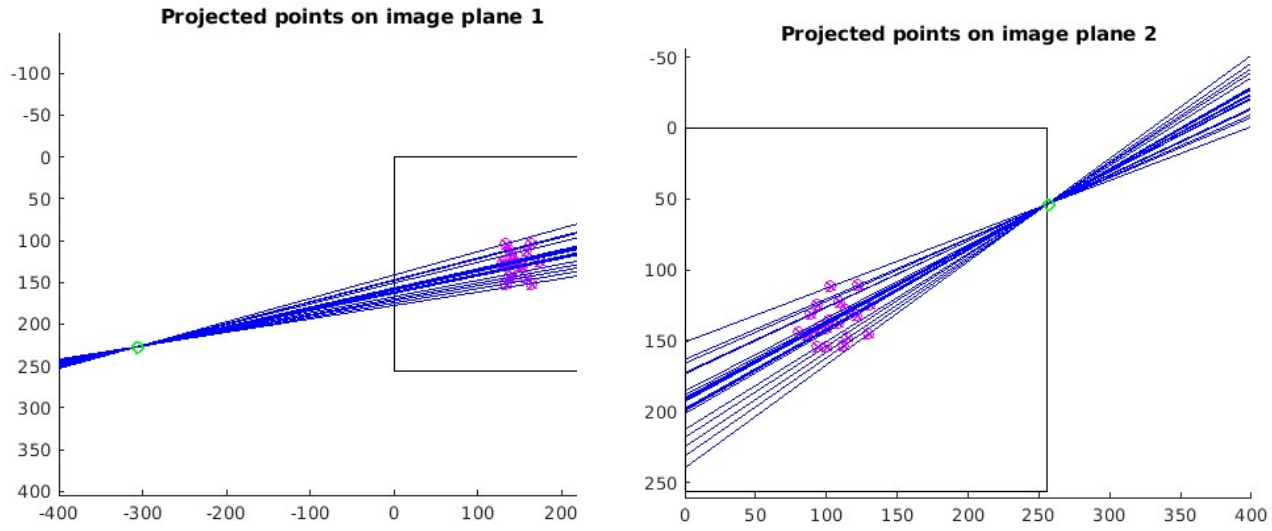
```
Sum of Absolute Differences: 0.00031638
```

Figure 8: Fundamental Matrix Difference

SAD is generally preferable because it indicates a smaller difference between corresponding pixels. The idea is to find the best match, and a lower SAD suggests a better match. In this step we got a SAD value of 0.00031638.

Step 10

In this step, we have computed the epipolar lines and epipoles (as depicted in Figure **??**) using the fundamental matrix.



(a) Epipolar Lines and Epipole in image plane 1 (b) Epipolar Lines and Epipole in image plane 2

Figure 9: Epipolar Lines and Epipole

In Figure **??**, the epipolar lines appear to converge at a point on the left side of the image plane, indicating the position of the epipole. Similarly, in Figure **??**, the epipolar lines converge on the right, indicating the epipole's position for the second image plane. The setup of these lines and points is instrumental in triangulating the position of the original 3D point that the two cameras observed.

Step 11

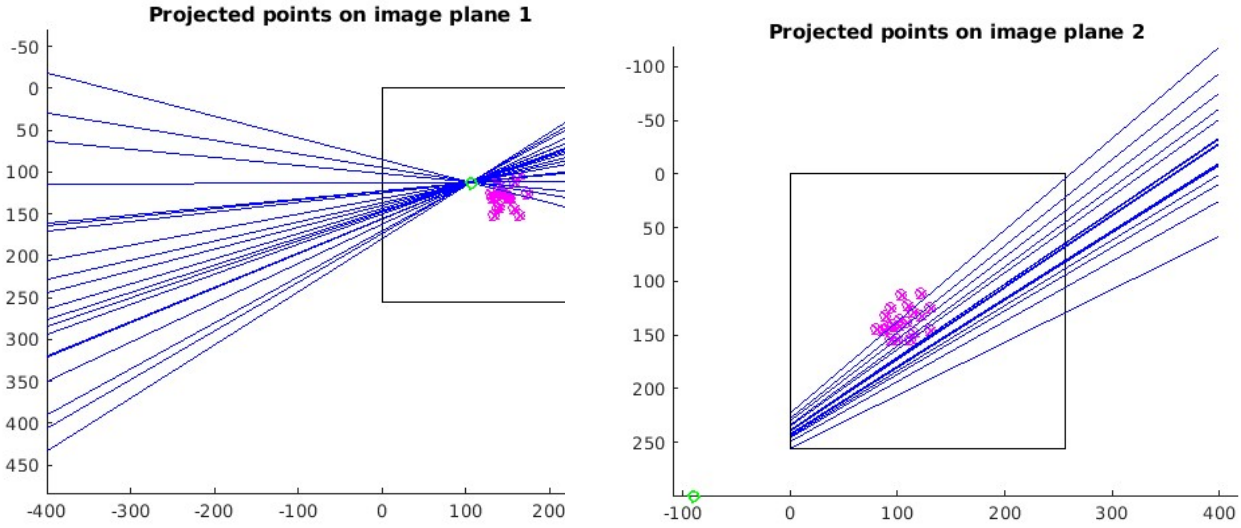
In this step, we have added Zero Mean Gaussian noise to the points such that 95 percentage of points are in the range [-1, +1]. The code of this step is shown in Figure. 10

```
%step 11
proj_size = size(cam1_p2d);
r_one = normrnd(0,0.5,proj_size);
r_two = normrnd(0,0.5,proj_size);
cam1_p2d_noise = cam1_p2d + r_one;
cam2_p2d_noise = cam2_p2d + r_two;
```

Figure 10: Code for adding Gaussian noise

Step 12

After adding noise we have recomputed the epipole and epipolar lines. There is a significant amount of change after adding noise which is illustrated in Figure **??**



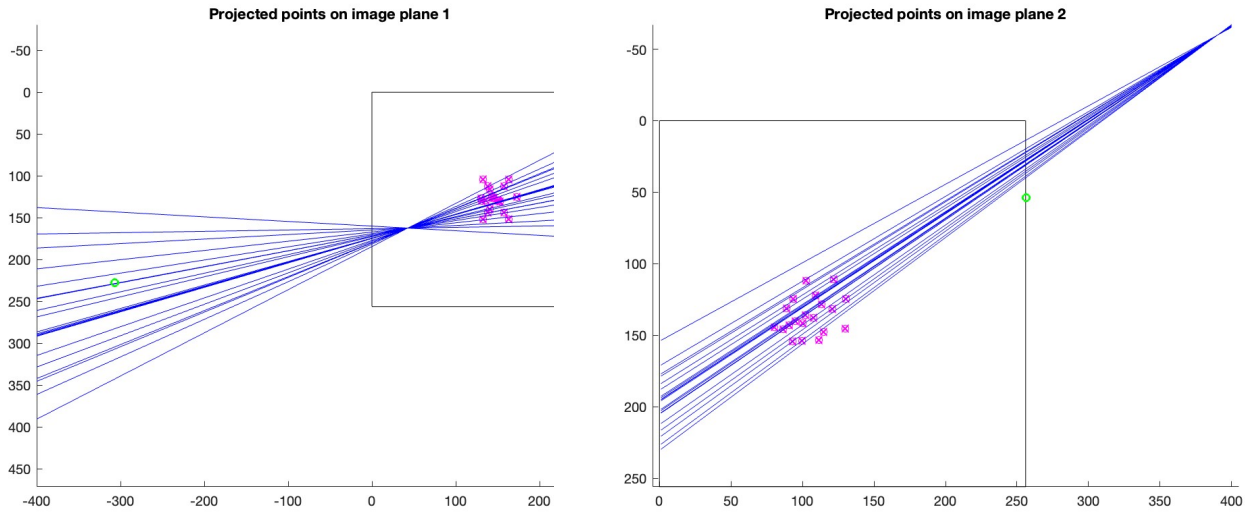
(a) Epipolar Lines and Epipole in image plane 1 (b) Epipolar Lines and Epipole in image plane 2

Figure 11: Epipolar Lines and Epipole with Gaussian Noise

The magenta points are the projections of the 3D points onto the image planes, now perturbed by the added noise. The points were initially at precise locations, the noise cause them to scatter, but most would stay within the specified range from their original positions. The blue lines are the epipolar lines, which should theoretically intersect at the corresponding points in the other image. However, due to the noise, the corresponding points is not lie exactly on the epipolar lines, and the lines are not intersect perfectly at a single point.

Question 1

Are points on the epipolar lines?
Solution: From Figure **??**it is clearly seen that the points are not on the epipolar lines.



(a) Epipolar Lines and Epipole in image plane 1 (b) Epipolar Lines and Epipole in image plane 2

Figure 12: Epipolar Lines and Epipole with Gaussian Noise

Question 2

Are the epipoles a unique point?
Solution: No, the epipoles are not unique. The reason behind this is, the epipolar lines are not passing through the same point.

Question 3

What condition should the F matrix fulfill so that all epipolar lines cross at the same point?
Solution: If the rank 2 constraint is imposed on the fundamental matrix(F), then all the epipolar lines cross at the same point.

Question 4

Impose that condition and Verify if all epipolar lines now cross at single same point on each image.
Solution: Figure **??** shows that, after imposing the rank 2 constraint, the epipolar lines cross at the same point.

Question 5

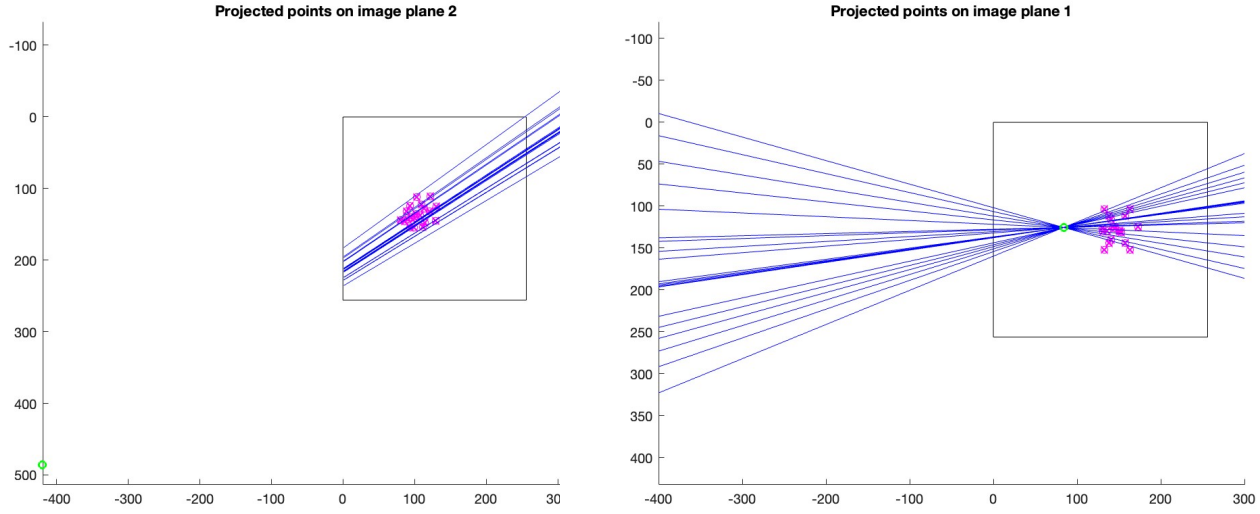
Investigate how the location of both epipoles can be computed directly from the F matrix, using the SVD, and describe the procedure.
Solution: The following code snippet(Figure15) shows the location of both epipoles can be computed directly from the fundamental matrix.

Step 13

In this step, we have increased the Zero Mean Gaussian noise to the points such that 95 percent of points are in the range [-2, +2]. Figure 2.13 shows the outcome.

```
%
% % Draw epipoles
[U,~,V] = svd(F);
ep_1 = V(:,end); % The epipole in the first image is the last column of V
ep_2 = U(:,end); % The epipole in the second image is the last column of U
%
% % Normalize the epipoles (to make the third coordinate 1)
ep_1 = ep_1 / ep_1(3);
ep_2 = ep_2 / ep_2(3);
```

Figure 13: Epipole Location Computation Directly from F Matrix



(a) Epipolar Lines and Epipole in image plane 1 (b) Epipolar Lines and Epipole in image plane 2

Figure 14: Epipolar Lines and Epipole with Gaussian Noise

The points were initially at little scattered locations, the increased noise cause them to more scatter.Due to the increased noise, the corresponding points is not lie exactly on the epipolar lines, and the lines are not intersect perfectly at a single point.

Step 14

In this step we have implemented the coordinate normalization method and Check the condition number of U-n of the 8-point algorithm that we get with and without coordinate normalization. The coordinates of the image points are scaled so that the average distance from the origin is about the square root of 2. This is done to have a consistent scale across different images and to reduce numerical errors during calculations. The centroid of the points (the average of all points) is moved to the origin of the coordinates system. This helps in centering the distribution of points and, again, aids in reducing computational errors. After these steps are applied, the fundamental matrix is recalculated with the normalized coordinates, which should lead to more accurate computation of the epipolar lines.

```
% Normalize points from camera 1
mean_cam1_p2d = mean(cam1_p2d(1:2,:), 2);
std_cam1_p2d = std(cam1_p2d(1:2,:), 0, 2);
T1 = [sqrt(2)/std_cam1_p2d(1) 0 -sqrt(2)/std_cam1_p2d(1)*mean_cam1_p2d(1);
      0 sqrt(2)/std_cam1_p2d(2) -sqrt(2)/std_cam1_p2d(2)*mean_cam1_p2d(2);
      0 0 1];
cam1_p2d_normalized = T1 * [cam1_p2d(1:2,:); ones(1, size(cam1_p2d, 2))];

% Normalize points from camera 2
mean_cam2_p2d = mean(cam2_p2d(1:2,:), 2);
std_cam2_p2d = std(cam2_p2d(1:2,:), 0, 2);
T2 = [sqrt(2)/std_cam2_p2d(1) 0 -sqrt(2)/std_cam2_p2d(1)*mean_cam2_p2d(1);
      0 sqrt(2)/std_cam2_p2d(2) -sqrt(2)/std_cam2_p2d(2)*mean_cam2_p2d(2);
      0 0 1];
cam2_p2d_normalized = T2 * [cam2_p2d(1:2,:); ones(1, size(cam2_p2d, 2))];

% Ensure that points are in homogeneous coordinates
cam1_p2d_normalized = cam1_p2d_normalized ./ cam1_p2d_normalized(3,:);
cam2_p2d_normalized = cam2_p2d_normalized ./ cam2_p2d_normalized(3,:);

%for step 8:
U_new_normalized = [cam2_p2d_normalized(1,:)'.*cam1_p2d_normalized(1,:)' cam2_p2d_normalized(1,:)'.*cam1_p2d_normalized(2,:)' ...
    cam2_p2d_normalized(1,:)' cam2_p2d_normalized(2,:)'.*cam1_p2d_normalized(1,:)' cam2_p2d_normalized(2,:)'.*cam1_p2d_normalized(2,:)' ...
    cam2_p2d_normalized(2,:)' cam1_p2d_normalized(1,:)' cam1_p2d_normalized(2,:)' ones(numcol,1)];

% Compute SVD of U_new_normalized
[~, ~, V_eight_normalized] = svd(U_new_normalized);

% Reshape the last column of V_eight_normalized to form F_normalized
F_normalized = reshape(V_eight_normalized(:,9), 3,3)';

% Enforce the rank-2 constraint on F_normalized
[U_normalized, D_normalized, V_normalized] = svd(F_normalized);
F_normalized = U_normalized * diag([D_normalized(1,1) D_normalized(2,2) 0]) * V_normalized';

% Denormalize the F matrix
F_D = T2' * F_normalized * T1;
```

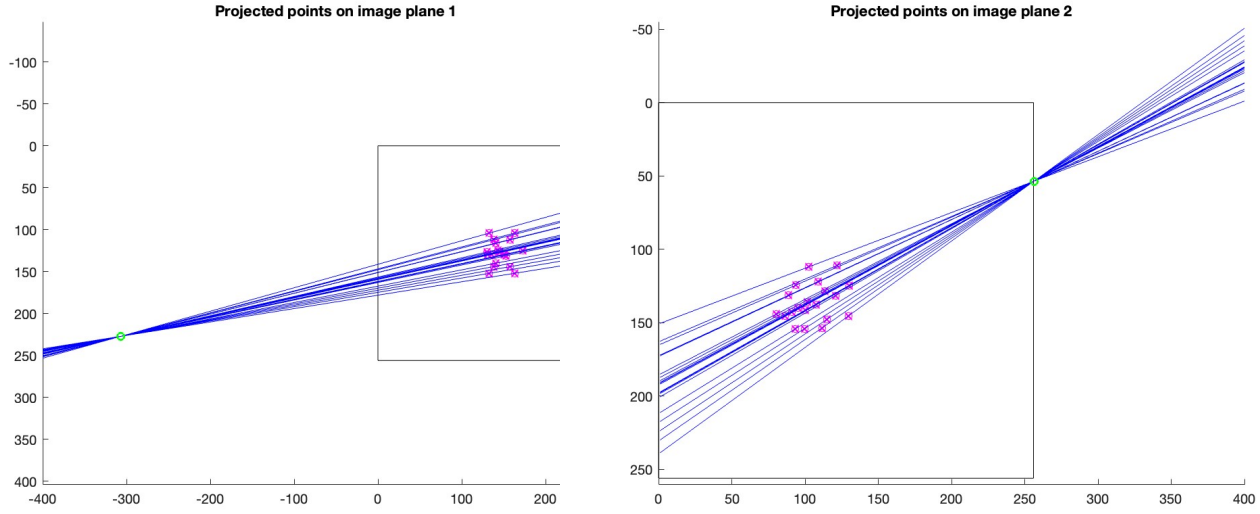
Figure 15: Enter Caption

```
Condition number with normalization: 3633686782628192
Condition number without normalization: 8297746.2557
Sum of Absolute Differences: 6.3769
```

Figure 16: SAD and Condition Number

Question 6

Are there improvements in terms of the sum of absolute differences (Step 9) and on the location of the epipolar lines (Step 10)?
Solution: No. With the normalization, sum of absolute differences has increased a lot compare to step 9 and the locaton of eipolar lines have improved. The output is shown in figure



(a) Epipolar Lines and Epipole in image plane 1 (b) Epipolar Lines and Epipole in image plane 2

Figure 17: Epipolar Lines and Epipole with Normalization

The resulting epipolar lines have intersect more consistently at corresponding points between

the two image planes, leading to improved stereo matching and 3D reconstruction accuracy. The figures show the epipolar lines and points post-normalization, which should now be in a normalized coordinate system that enhances the precision of stereo vision algorithms.

Conclusion

The preceding experiments have provided us with fundamental insights into the implementation of epipolar geometry and the consequences of adjusting parameters. Initially, we successfully derived the Fundamental Matrix, a central component of epipolar geometry. Subsequently, we calculated the epipoles and epipolar lines, plotting their projections. Introducing noise had a notable impact, causing the epipole to be non-unique, given that the lines did not intersect at a single point. To address this, we imposed a rank 2 constraint and directly computed the epipoles from the F matrix. Finally, we intensified the noise to observe the effects on the behavior of epipolar geometry.