Visual Perception
Lab 3 Report: Reconstruction From Two Views

Mohammad Rami Koujan
M.Sc. VIBOT
University of Girona

April 24, 2016

1 Introduction and Problem Definition

The objective of this lab practice is to become familiar with the problem of reconstruction from two views by describing two simulated cameras, obtaining their fundamental matrices analytically, defining a set of 3D points and getting their corresponding couples of projecting points, computing the Fundamental matrix by using the 8-point method, comparing both fundamental matrices, drawing the epipolar geometry in both images planes (points, epipoles and epipolar lines), and increasing the noise in 2D points and repeating the computations. There are, in fact, many techniques that are based on the principle of reconstruction from two views. For example, Depth from Focus/Defocus and Depth from Zooming, Shape from Structured Light and Shape from Photometric Stereo, Shape from Shading, Shape from Texture and Shape from Geometric Constraints.

2 Analysis of the followed steps

This section presents the different steps that were followed in this assignment in details. In the first step, it was necessary to define the intrinsic parameters of the first camera that are used in the following steps. Those parameters are defined as follow:

- au1 = 100; av1 = 120; uo1 = 128; vo1 = 128;

where the first image size is considered to be 256*256 and the world coordinates system is the same as this camera system.

The second step is about defining camera 2 with respect to camera 1 as follows:

- au2 = 90; av2 = 110; uo2 = 128; vo2 = 128;
- ax = 0.1; by = pi/4; cz = 0.2 ;
- tx = -1000; ty = 190; tz = 230;
- Rotx=[1 0 0;0 cos(ax) -sin(ax);0 sin(ax) cos(ax)];
- Roty=[cos(by) 0 sin(by) ;0 1 0;-sin(by) 0 cos(by)];
- Rotz=[cos(cz) -sin(cz) 0;sin(cz) cos(cz) 0;0 0 1];
- T=[tx,ty,tz];
- R=Rotx*Roty*Rotz;
- % output

$$T = \begin{bmatrix} -1000 & 190 & 230 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.6930 & -0.1405 & 0.7071 \\ 0.2669 & 0.9611 & -0.0706 \\ -0.6697 & 0.2376 & 0.7036 \end{bmatrix}$$

where T and R represent the translation vector and the rotation matrix of camera 2 with respect to camera 1.

In the next step, the intrinsic transformation matrices of both cameras, and the rotation and translation between both cameras were obtained from the previously defined parameters.

- p1=[eye(3) [0 0 0]'];
- p2=[R T'];
- A1=[au1 0 uo1;0 av1 vo1;0 0 1];
- A2=[au2 0 uo2;0 av2 vo2;0 0 1];
- % output

$$p1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$p2 = \begin{bmatrix} 0.69301 & -0.14048 & 0.70711 & -1000 \\ 0.26686 & 0.96115 & -0.070593 & 190 \\ -0.66972 & 0.23762 & 0.70357 & 230 \end{bmatrix}$$

$$A1 = \begin{bmatrix} 100 & 0 & 128 \\ 0 & 120 & 128 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A2 = \begin{bmatrix} 90 & 0 & 128 \\ 0 & 110 & 128 \\ 0 & 0 & 1 \end{bmatrix}$$

where p1 represents the transformation matrix from camera 1 to the world. Since they are the same, p1 is just a zero translation with identity rotation. p2 represents the transformation matrix from camera 2 to camera 1, and A1 and A2 are the 3*3 intrinsic matrices of both cameras

In the fourth step, the Fundamental matrix was computed analytically as the product of matrices defined in step 3, as follows:

- antiSym=[0 -T(3) T(2); T(3) 0 -T(1); -T(2) T(1) 0];
- F=inv(A2')*R'*antiSym*inv(A1);
- % output

$$antiSym = \begin{bmatrix} 0 & -230 & 190 \\ 230 & 0 & 1000 \\ -190 & -1000 & 0 \end{bmatrix}$$

$$F = \begin{bmatrix} 0.020958 & 0.047252 & -4.3028 \\ 0.015992 & -0.015554 & 8.4389 \\ -6.2288 & -11.276 & 650.18 \end{bmatrix}$$

A set of object points (20) with respect to the world coordinate system (or camera 1 coordinate system) was defined in step five below:

- V(:,1) = [100;-400;2000;1];
- V(:,2) = [300;-400;3000;1];
- V(:,3) = [500;-400;4000;1];
- V(:,4) = [700;-400;2000;1];
- V(:,5) = [900;-400;3000;1];
- V(:,6) = [100;-50;4000;1];
- V(:,7) = [300;-50;2000;1];
- V(:,8) = [500;-50;3000;1];

- V(:,9) = [700;-50;4000;1];
- V(:,10) = [900;-50;2000;1];
- V(:,11) = [100;50;3000;1];
- V(:,12) = [300;50;4000;1];
- V(:,13) = [500;50;2000;1];
- V(:,14) = [700;50;3000;1];
- V(:,15) = [900;50;4000;1];
- V(:,16) = [100;400;2000;1];
- V(:,17) = [300;400;3000;1];
- V(:,18) = [500;400;4000;1];
- V(:,19) = [700;400;2000;1];
- V(:,20) = [900;400;3000;1];

Those 3D points are defined in the homogeneous coordinates.

Step 6 is about calculating the couples of image points in both image planes by using the matrices of step 3. In order to do that, the corresponding intrinsic matrices with dimensions 3x4 and the corresponding extrinsic matrices with dimensions 4x4 were used to project the 3D points onto both image planes and get both projections in pixels. Since the first camera is located at the origin of the world coordinates system, Ext1=[I 0] while camera 2 is located at [R t] and thus $Ext2 = [R^{\mathrm{t}} - R^{\mathrm{t}}t]$.

- Int1=[A1,[0 0 0]'];
- Int2=[A2,[0 0 0]'];
- ext1=[p1;[0 0 0 1]];
- ext2=[R' -R'*T';0 0 0 1];
- points1=Int1*ext1*V;
- points2=Int2*ext2*V;
- for i=1:size(V,2)
- points1(1:3,i)=points1(1:3,i)/points1(3,i);
- points2(1:3,i)=points2(1:3,i)/points2(3,i);
- end
- % output

$$ext1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ext2 = \begin{bmatrix} 0.69301 & 0.26686 & -0.66972 & 796.34 \\ -0.14048 & 0.96115 & 0.23762 & -377.75 \\ 0.70711 & -0.070593 & 0.70357 & 558.7 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

the "for" loop was used to normalize the x and y coordinates of the resulting 3D points by the third component of resultant points vectors.

The aim of step 7 is to draw the 2D points obtained in step 6 by opening two windows in matlab, which were used as both image planes.

- scatter(points1(1,:),points1(2,:));

- figure;scatter(points2(1,:),points2(2,:));hold on;

where the function "scatter" was used to plot those points which are shown below:

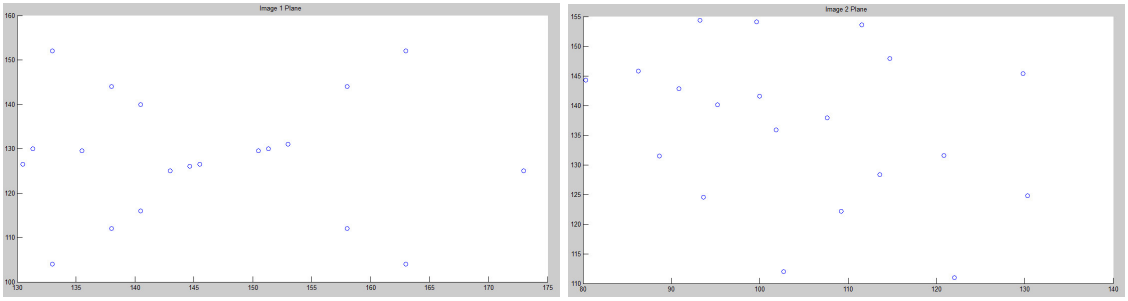

Figure 1: Projected 2D points on both cameras' planes

Next, the Fundamental matrix was computed by using the 8-point method and least-squares by means of the 2D points obtained in step 6. The following equation was used:

$$(x', y', 1)F\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$

- u=zeros(size(V,2),8);
- for i=1:size(V,2)
- u(i,:)=[points1(1,i)*points2(1,i),points1(2,i)*points2(1,i),
- points2(1,i),points1(1,i)*points2(2,i),points1(2,i)*points2(2,i),points2(2,i),points1(1,i),points1(2,i)];
- end
- F_least=-u\ones(size(V,2),1);
- F_least=[F_least ;1];
- F_least=[F_least(1:3)';F_least(4:6)';F_least(7:9)'];
- % output

$$F\_least = \begin{bmatrix} 3.2235e-05 & 7.2676e-05 & -0.0066178 \\ 2.4597e-05 & -2.3923e-05 & 0.012979 \\ -0.0095802 & -0.017343 & 1 \end{bmatrix}$$

Where after constructing the 'u' matrix in a "for" loop from the previously defined 3D points, the least square method is used to find F_least by using the backslash operator in Matlab. Then, the ninth element which is assumed to be one was added to the final solution.

Then, in step 9, the step 8 matrix was compared with the one obtained in step 4 after normalizing it by the ninth element of this matrix.

- F=F(:,:)/F(3,3);
- F_diff=F-F_least;
- % output

$$F\_diff = \begin{bmatrix} -5.5565e-19 & -4.7298e-18 & 3.3567e-16 \\ 4.8789e-19 & 1.9821e-18 & -9.6971e-16 \\ 9.8879e-17 & 8.6042e-16 & 0 \end{bmatrix}$$

As it is shown by the difference matrix, the two matrices are almost identical.

Step 10 is dedicated for drawing the epipolar geometry for both cameras, i.e. epipoles and epipolar lines by using the Fundamental matrix obtained in step 8. This was done in two steps and repeated for both cameras. Firstly, the epipole line corresponding to each 3D point was computed by mainly using the following equation l'm=Fm=[u1,u2,u3]^T (for the second camera) and lm'=F'm'=[u1,u2,u3]^T (for the first camera). Then, from the preceding equation, y=mx+d was extracted so that m and d were obtained from the components of the epipolar line. Hence, fixing x in both boundaries of the image plane, gives the corresponding y components. Secondly, the epipole was computed in three different ways. The first one is by using the projection method where the origins of the first and second cameras were projected on each other by using the intrinsic and extrinsic matrices of the second and first cameras respectively. The second method is to use the intersection point of all the epipolar lines on one image after calculating all of them. This was done, indeed, by solving the system of these equations using the backslash operator in Matlab. The third method is to calculate the singular value decomposition of the Fundamental matrix (F' for the second camera and F for the first one). Then, the epipole is a multiple of the column of V (for the first image) or U (for the second image) that belongs to the zero singular value of F. The following Matlab code shows the previously explained computations for both cameras:

```
- % epipole geometry for the second image
- L_prime=F_least*points1;
- x1=-500;
- x2=500;
- m=zeros(size(V,2),1);
- d=zeros(size(V,2),1);
- for i=1:size(V,2)
- m(i)=-L_prime(1,i)/L_prime(2,i);
- d(i)=-L_prime(3,i)/L_prime(2,i);
- y1=m(i)*x1+d(i);
- y2=m(i)*x2+d(i);
- plot([x1 x2],[y1 y2]);hold on;
- end

- % computing the epipole by projection
- e_prime=Int2*ext2*[0;0;0;1];
- e_prime=e_prime(1:2,1)/e_prime(3,1);
- scatter(e_prime(1),e_prime(2),'fill');
- text(e_prime(1),e_prime(2),'Epipole_prime');
- title('Second image plane');

- % computing the epipole from the crossing of the epipolar lines
- mat=[-m,ones(size(V,2),1)];
- e_prime_cross=mat\ d;
- e_diff_cross=e_prime-e_prime_cross;

- % computing the epipole from the Fundamental matrix
- [∼, ∼, v_mat] = svd(F');
```

- e_prime_Fund=v_mat(1:2,end)/v_mat(end,end);
- e_diff_Fund=e_prime-e_prime_Fund;
- % eipipole geometry for first image plane
- L=F_least'*points2;
- figure;scatter(points1(1,:),points1(2,:));hold on;
- m2=zeros(size(V,2),1);
- d2=zeros(size(V,2),1);
- for i=1:size(V,2)
- m2(i)=-L(1,i)/L(2,i);
- d2(i)=-L(3,i)/L(2,i);
- y1=m2(i)*x1+d2(i);
- y2=m2(i)*x2+d2(i);
- plot([x1 x2],[y1 y2]);
- end

- % computing the epipole by projection
- e=Int1*ext1*[T,1]';
- e=e(1:2)/e(3);
- scatter(e(1),e(2),'fill');
- text(e(1),e(2),' Epipole');
- title('First image plane');

- % computing the epipole from the crossing of the epipolar lines
- mat2=[-m2,ones(size(V,2),1)];
- e_cross=mat2\ d2;
- e_diff2_cross=e-e_cross;

- % computing the epipole from the Fundamental matrix
- $[\sim, \sim, v\_mat]$ = svd(F);
- e_Fund=v_mat(1:2,end)/v_mat(end,end);
- e_diff2_Fund=e-e_Fund;
- % output

$$e\_diff\_cross = \begin{bmatrix} 2.6148e-12 \\ -1.485e-12 \end{bmatrix}$$

$$e\_diff\_Fund = \begin{bmatrix} -2.4443e-12 \\ -2.0677e-12 \end{bmatrix}$$

$$e\_diff2\_cross = \begin{bmatrix} 2.4386e-11 \\ -4.2633e-12 \end{bmatrix}$$

$$e\_diff2\_Fund = \begin{bmatrix} -5.9686e-12 \\ 2.0748e-12 \end{bmatrix}$$

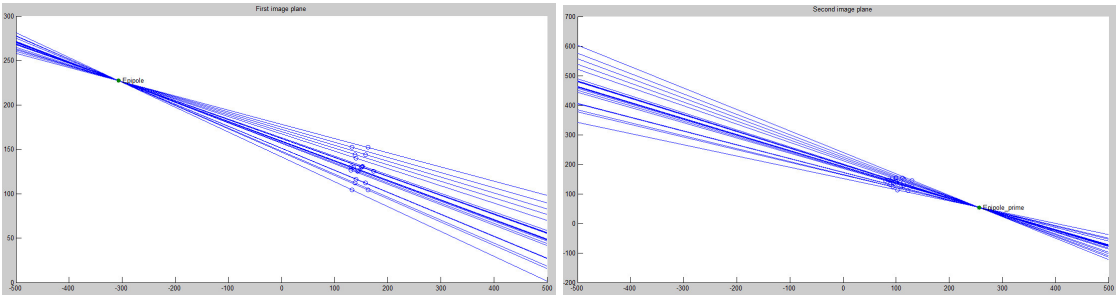

Figure 2: Epipoles and epipolar lines for both cameras

It is evident from the images of both cameras that the epipolar lines intersect at the epipole and the differences between the epipoles calculated using the three mentioned methods are very small which means that they are almost the same.

In step 11, Gaussian noise was added to the 2D points producing discrepancies between the range [-1,+1] pixels for 95% of points (which means sigma is equal to 0.5).

- sigma=0.5;
- Noise=sigma*randn(2,size(points1,2));
- Noise2=sigma*randn(2,size(points2,2));
- points1_n=points1(1:2,:)+Noise;points1_n(3,:)=1;
- points2_n=points2(1:2,:)+Noise2;points2_n(3,:)=1;

In step 12 it is required to repeat step 8 to 10 with the noisy 2D points and compare the epipole geometry. The same previous code was used except for the 2D points which were substituted by the noisy ones. Additionally, the mean and standard deviation for the set of differences between the noisy points and the corresponding epipolar lines were calculated for both images. However, the final results are shown at the end of this report for distinct number of points and values of sigma.

- u_n=zeros(size(V,2),8);
- for i=1:size(V,2)
- u_n(i,:)=[points1_n(1,i)*points2_n(1,i),points1_n(2,i)*points2_n(1,i),points2_n(1,i), points1_n(1,i)*points2_n(2,i) ,points1_n(2,i)*points2_n(2,i), points2_n(2,i),points1_n(1,i),points1_n(2,i)];
- end
- F_n=-u_n \ones(size(V,2),1);
- F_n=[F_n ;1];
- %step 12-9
- F_n=[F_n(1:3)';F_n(4:6)';F_n(7:9)'];
- F_diff_n=F_n-F;
- %step 12-10
- % epipole geometry for the second image
- L_prime_n=F_n*points1_n;
- figure;scatter(points2_n(1,:),points2_n(2,:));hold on;

- m_n=zeros(size(V,2),1);
- d_n=zeros(size(V,2),1);
- mean_dis=zeros(size(V,2),1);
- for i=1:size(V,2)
- m_n(i)=-L_prime_n(1,i)/L_prime_n(2,i);
- d_n(i)=-L_prime_n(3,i)/L_prime_n(2,i);
- y1=m_n(i)*x1+d_n(i);
- y2=m_n(i)*x2+d_n(i);
- mean_dis(i)=abs(L_prime_n(1,i)*points2_n(1,i)+L_prime_n(2,i)*points2_n(2,i)+L_prime_n(3,i)) sqrt(L_prime_n(1,i)^2+L_prime_n(2,i)^2);
- plot([x1 x2],[y1 y2]);hold on;
- end
- ave=mean(mean_dis);
- s_dev=std(mean_dis);
- % computing the epipole by projection
- % the same as without noise
- scatter(e_prime(1),e_prime(2),'fill');
- text(e_prime(1),e_prime(2),' Epipole_prime');
- title('Second image plane (noisy)');
- % computing the epipole from the crossing of the epipolar lines
- mat_n=[-m_n,ones(size(V,2),1)];
- e_prime_cross_n=mat_n\ d_n;
- e_diff_cross_n=e_prime(1:2)-e_prime_cross_n;
- scatter(e_prime_cross_n(1),e_prime_cross_n(2),'fill');
- text(e_prime_cross_n(1),e_prime_cross_n(2),' Epipole_prime_cross_noisy');
- % computing the epipole from the fundamental matrix
- $[\sim, \sim, v_m at_n]$ = svd(F_n');
- e_prime_Fund_n=v_mat_n(1:2,end)/v_mat_n(end,end);
- e_diff_Fund_n=e_prime-e_prime_Fund_n;
- scatter(e_prime_Fund_n(1),e_prime_Fund_n(2),'fill');
- text(e_prime_Fund_n(1),e_prime_Fund_n(2),' Epipole_prime_F_noisy');
- % eipipole geometry for first image plane
- L_n=F_n'*points2_n;
- figure;scatter(points1_n(1,:),points1_n(2,:));hold on;
- m2_n=zeros(size(V,2),1);
- d2_n=zeros(size(V,2),1);
- mean_dis2=zeros(size(V,2),1);

- for i=1:size(V,2)
- m2_n(i)=-L_n(1,i)/L_n(2,i);
- d2_n(i)=-L_n(3,i)/L_n(2,i);
- y1=m2_n(i)*x1+d2_n(i);
- y2=m2_n(i)*x2+d2_n(i);
- mean_dis2(i)=abs(L_n(1,i)*points1_n(1,i)+L_n(2,i)*points1_n(2,i)+L_n(3,i))/sqrt(L_n(1,i)^2+L_n(2,i)^2);
- plot([x1 x2],[y1 y2]);
- end
- ave2=mean(mean_dis2);
- s_dev2=std(mean_dis2);
- ave_LS=(ave+ave2)/2
- s_dev_LS=(s_dev+s_dev2)/2
- % computing the epipole by projection
- % the same as without noise
- title('First image plane (noisy)');
- scatter(e(1),e(2),'fill');
- text(e(1),e(2),' Epipole');
- % computing the epipole from the crossing of the epipolar lines
- mat2_n=[-m2_n,ones(size(V,2),1)];
- e_cross_n=mat2_n\ d2_n;
- e_diff2_n=e(1:2)-e_cross_n;
- scatter(e_cross_n(1),e_cross_n(2),'fill');
- text(e_cross_n(1),e_cross_n(2),' Epipole_cross_noisy');
- % computing the epipole from the fundamental matrix
- $[\sim, \sim, v_m at_n]$ = svd(F_n);
- e_Fund_n=v_mat_n(1:2,end)/v_mat_n(end,end);
- e_diff_Fund_n2=e-e_Fund_n;
- scatter(e_Fund_n(1),e_Fund_n(2),'fill');
- text(e_Fund_n(1),e_Fund_n(2),' Epipole_F_noisy');
- % checking the rank of the noisy Fundamental matrix
- rank(F_n)
- % output

$$F\_n = \begin{bmatrix} 1.8351e-05 & 1.1836e-05 & -0.0026121 \\ 2.9106e-05 & 7.0494e-06 & -0.0021701 \\ -0.0064325 & -0.0049618 & 1 \end{bmatrix}$$

$$F\_diff\_n = \begin{bmatrix} -1.3884e-05 & -6.084e-05 & 0.0040057 \\ 4.5094e-06 & 3.0972e-05 & -0.01515 \\ 0.0031477 & 0.012381 & 0 \end{bmatrix}$$

$$e\_diff\_cross\_n = \begin{bmatrix} -174.51 \\ 89.446 \end{bmatrix}$$

$$e\_diff\_Fund\_n = \begin{bmatrix} -166.55 \\ 101.77 \end{bmatrix}$$

$$e\_diff2\_n = \begin{bmatrix} -347.42 \\ 79.189 \end{bmatrix}$$

$$e\_diff\_Fund\_n2 = \begin{bmatrix} -344.95 \\ 75.071 \end{bmatrix}$$

- rank(F_n)=3

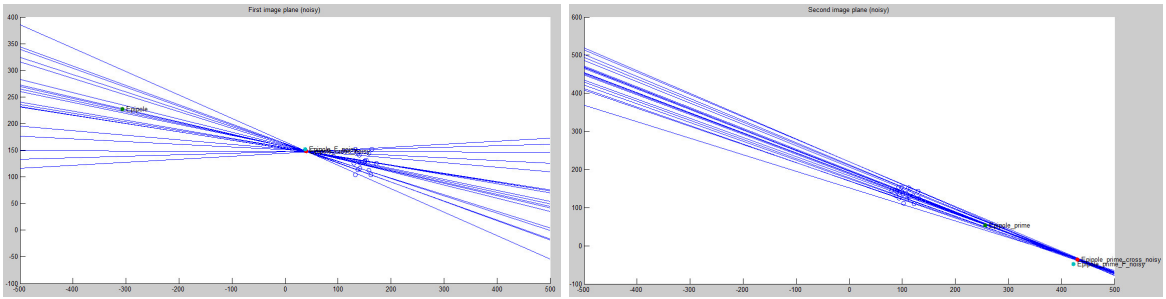

Figure 3: Epipoles and epipolar lines for both cameras when the 2D points are noisy($\sigma = 0.5$)

The noisy Fundamental matrix shows that it is affected by the added noise because it has different values from the original one and it becomes a rank-3 matrix instead of 2. Moreover, the epipoles computed from three different methods are also not identical anymore since they get the results using non similar methods which are affected by noise differently. The previous two images demonstrate that the epipolar lines are not intersecting any longer and the epipoles computed from the Fundamental matrix and from the intersection are not the same as the one computed form the projection.

In the second part of this step, the Fundamental matrix was forced to be rank-2 again which causes the epipolar lines to intersect at one point. This was done by first computing the SVD (F=UDV$^{\mathrm{T}}$), setting the smallest eigenvalue of the D matrix to zero, and then calculating the F matrix again. The following Matlab code shows the explained method in details:

- $[U, S, v\_mat]$ = svd(F_n);
- $[\sim, i]$=min([S(1,1),S(2,2),S(3,3)]);
- S(i,i)=0;
- F_n=U*S*v_mat';
- rank(F_n)
- %output
  rank(F_n)=2

$$F\_n = \begin{bmatrix} 2.7904e-05 & 6.8442e-06 & -0.0030782 \\ 2.4975e-05 & 3.5993e-06 & -0.0023965 \\ -0.0068547 & -0.0033989 & 1 \end{bmatrix}$$

The rest of the code is identical to the previous case except for the F_n where the updated one was used. The following two images show the new results.

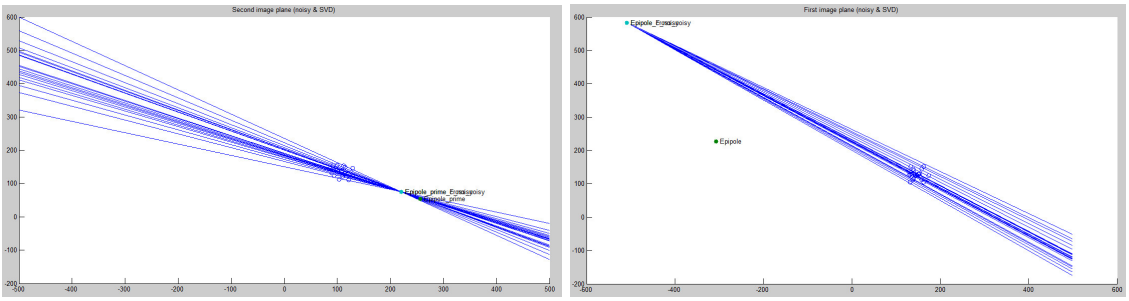

Figure 4: Epipoles and epipolar lines for both cameras when forcing the noisy Fundamental matrix to be rank-2 again

It is clearly shown that the epipolar lines in both images' planes are intersecting at one point as a result of forcing the Fundamental matrix to be rank 2 as it was before adding the noise.

Next, The purpose of step 13 is just to increase the range of the added noise to [-2,+2] which means increasing sigma to 1. Therefore, the preceding code was run again but for sigma equals to 1. Below are the results:

$$F\_n = \begin{bmatrix} 1.8858e-05 & -1.3058e-05 & -0.0016999 \\ 2.9186e-05 & 1.6532e-05 & -0.0071563 \\ -0.0056645 & -4.4885e-05 & 1 \end{bmatrix}$$

$$F\_diff\_n = \begin{bmatrix} -1.3377e-05 & -8.5734e-05 & 0.0049179 \\ 4.5892e-06 & 4.0455e-05 & -0.020136 \\ 0.0039158 & 0.017298 & 0 \end{bmatrix}$$

$$e\_diff\_cross\_n = \begin{bmatrix} 121.69 \\ -54.843 \end{bmatrix}$$

$$e\_diff\_Fund\_n = \begin{bmatrix} 123.03 \\ -54.459 \end{bmatrix}$$

$$e\_diff2\_n = \begin{bmatrix} -482.73 \\ 105.28 \end{bmatrix}$$

$$e\_diff\_Fund\_n2 = \begin{bmatrix} -482.35 \\ 104.03 \end{bmatrix}$$

And the results after forcing the Fundamental matrix to be rank-2 in this case are:

$$F\_n = \begin{bmatrix} 1.8845e-05 & -1.3067e-05 & -0.0016999 \\ 2.9176e-05 & 1.6525e-05 & -0.0071563 \\ -0.0056645 & -4.4886e-05 & 1 \end{bmatrix}$$

$$e\_diff\_cross\_n = \begin{bmatrix} 123.03 \\ -54.459 \end{bmatrix}$$

$$e\_diff\_Fund\_n = \begin{bmatrix} 123.03 \\ -54.459 \end{bmatrix}$$

$$e\_diff2\_n = \begin{bmatrix} -482.35 \\ 104.03 \end{bmatrix}$$

$$e\_diff\_Fund\_n2 = \begin{bmatrix} -482.35 \\ 104.03 \end{bmatrix}$$

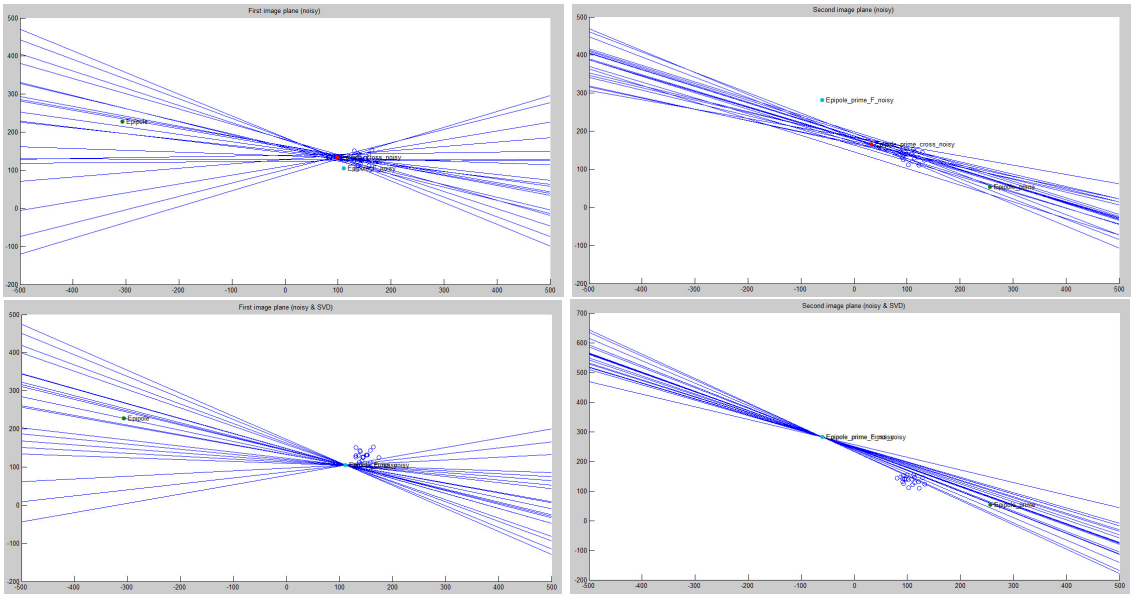

Figure 5: Epipoles and epipolar lines for both cameras when when the 2D points are noisy($\sigma = 1$), in the first row, and when forcing the noisy Fundamental matrix to be rank-2 again, in the second row

The previous results show that increasing the range of the added noise has the effect of raising the discrepancies which is shown in the F_diff_n matrix. After forcing the Fundamental matrix to be rank-2, the epipolar lines are intersecting again at one point

The fundamental matrix was computed, in step 14, by using the 8-point method and SVD from the 2D points obtained in step 6 without noise where the solution corresponds to a multiple of the column of V that belongs to the zero singular value of u_SVD as show below.

- u_SVD=zeros(size(V,2),9);
- for i=1:size(V,2)
- u_SVD(i,:)=[points1(1,i)*points2(1,i),points1(2,i)*points2(1,i),
- points2(1,i),points1(1,i)*points2(2,i),points1(2,i)*points2(2,i),points2(2,i),points1(1,i),points1(2,i),1];
- end
- [$\sim, S, v\_mat$]=svd(u_SVD);
- [$\sim, i$]=min([S(1,1),S(2,2),S(3,3),S(4,4),S(5,5),S(6,6),S(7,7),S(8,8),S(9,9)]);
- F_SVD=v_mat(:,i);
- F_SVD=F_SVD/F_SVD(end);
- F_SVD=[F_SVD(1:3)';F_SVD(4:6)';F_SVD(7:9)'];
- F_diff=F_SVD-F_least;
- % output

$$F\_SVD = \begin{bmatrix} 3.2235e-05 & 7.2676e-05 & -0.0066178 \\ 2.4597e-05 & -2.3923e-05 & 0.012979 \\ -0.0095802 & -0.017343 & 1 \end{bmatrix}$$

$$F\_diff = \begin{bmatrix} -1.6263e-19 & -3.483e-18 & 1.7347e-16 \\ 6.6746e-19 & 8.8769e-19 & -6.9215e-16 \\ 1.3878e-17 & 7.4246e-16 & 0 \end{bmatrix}$$

It is clear from the difference matrix that the least square and SVD methods give fairly the same results.

Steps 10 to 13 were repeated, in step 15, but this time for the Fundamental matrix that is obtained from the previous step. Hence, the same code was used again but for F_SVD, which is computed in the last step, instead of F_least. The following are the results.

$$e\_diff\_cross = \begin{bmatrix} 1.6485e-12 \\ -1.3074e-12 \end{bmatrix}$$

$$e\_diff\_Fund = \begin{bmatrix} -5.7412e-12 \\ -4.6754e-12 \end{bmatrix}$$

$$e\_diff2\_cross = \begin{bmatrix} -1.08e-12 \\ 5.6843e-13 \end{bmatrix}$$

$$e\_diff2\_Fund = \begin{bmatrix} -7.0486e-12 \\ 5.6843e-13 \end{bmatrix}$$

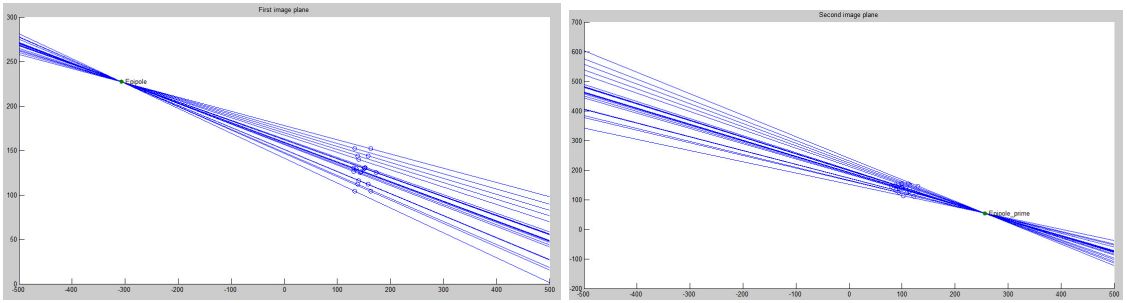

Figure 6: Epipoles and epipolar lines for both cameras when using SVD method

Those results are rather identical to the ones in step 10 (for F_least). Thereafter, a Gaussian noise was added to produce discrepancies between the range [-1,+1] pixels for the 95% of points (the same noisy points defined previously were used in this step). Thus, the code was run again but this time for the noisy points to get the Fundamental matrix using the SVD method and then drawing the epipole geometry of this noisy configuration as follows:

$$F\_SVD\_n = \begin{bmatrix} 2.072e-05 & -6.2838e-06 & -0.0021156 \\ 2.848e-05 & 1.3366e-05 & -0.0054021 \\ -0.0060778 & -0.0014886 & 1 \end{bmatrix}$$

$$F\_diff\_n = \begin{bmatrix} -1.1515e-05 & -7.896e-05 & 0.0045023 \\ 3.8831e-06 & 3.7288e-05 & -0.018382 \\ 0.0035024 & 0.015854 & 0 \end{bmatrix}$$

$$e\_diff\_cross\_n = \begin{bmatrix} 162.47 \\ -87.406 \end{bmatrix}$$

$$e\_diff\_Fund\_n = \begin{bmatrix} 171.26 \\ -98.191 \end{bmatrix}$$

$$e\_diff2\_n = \begin{bmatrix} -439.61 \\ 103.53 \end{bmatrix}$$

$$e\_diff\_Fund\_n2 = \begin{bmatrix} -443.53 \\ 113.68 \end{bmatrix}$$

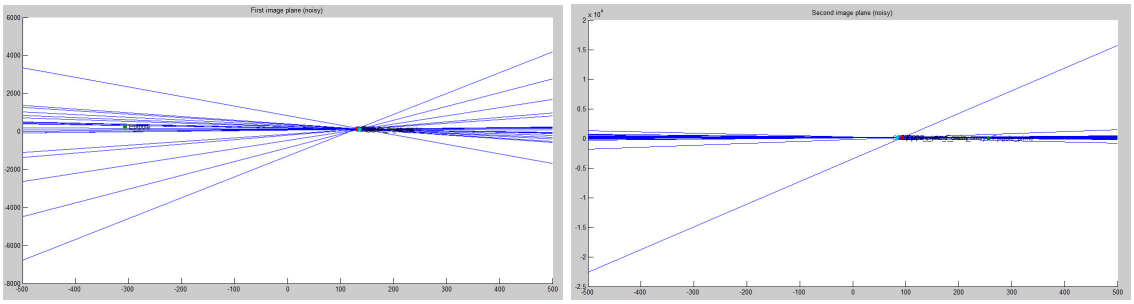

Figure 7: Epipoles and epipolar lines for both cameras when the 2D points are noisy ($\sigma = 0.5$ and SVD method)

The F_diff_n matrix shows the difference between the noisy and non noisy Fundamental matrices where it is clear that the noise has induced some discrepancies. Furthermore, the epipolar lines are neither passing through the 2D points nor intersecting at the epipole for both cameras. Now, forcing F_SVD_n to be rank-2 instead of rank-3, gives the following results:

$$F\_SVD\_n = \begin{bmatrix} 2.2487e-05 & 6.2363e-06 & -0.0025083 \\ 2.459e-05 & 8.7863e-06 & -0.003003 \\ -0.0062446 & -0.0040266 & 1 \end{bmatrix}$$

$$e\_diff\_cross\_n = \begin{bmatrix} 1254.6 \\ -1113.2 \end{bmatrix}$$

$$e\_diff\_Fund\_n = \begin{bmatrix} 1254.6 \\ -1113.2 \end{bmatrix}$$

$$e\_diff2\_n = \begin{bmatrix} -381.65 \\ 94.897 \end{bmatrix}$$

$$e\_diff\_Fund\_n2 = \begin{bmatrix} -381.65 \\ 94.897 \end{bmatrix}$$

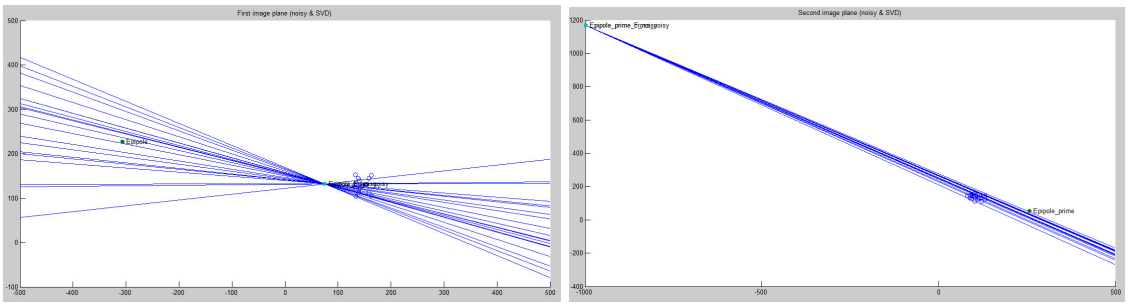

Figure 8: Epipoles and epipolar lines for both cameras when forcing the noisy SVD Fundamental matrix to be rank-2 again

where the epipolar lines are intersecting at one point again.
Now, repeating again steps 10 to 13 but for sigma equals to 1 ([-2,2] range) returns the following:

$$F\_SVD\_n = \begin{bmatrix} 2.2429e-05 & -7.8939e-06 & -0.0022967 \\ 2.7493e-05 & 1.3617e-05 & -0.0055154 \\ -0.0060626 & -0.0011562 & 1 \end{bmatrix}$$

$$F\_diff\_n = \begin{bmatrix} -9.8058e-06 & -8.057e-05 & 0.0043212 \\ 2.8958e-06 & 3.754e-05 & -0.018495 \\ 0.0035176 & 0.016186 & 0 \end{bmatrix}$$

$$e\_diff\_cross\_n = \begin{bmatrix} 178.93 \\ -91.156 \end{bmatrix}$$

$$e\_diff\_Fund\_n = \begin{bmatrix} 158.95 \\ -87.157 \end{bmatrix}$$

$$e\_diff2\_n = \begin{bmatrix} -450.51 \\ 126.08 \end{bmatrix}$$

$$e\_diff\_Fund\_n2 = \begin{bmatrix} -449.63 \\ 111.28 \end{bmatrix}$$

And the results after forcing the Fundamental matrix to be rank-2 in this case are:

$$F\_SVD\_n = \begin{bmatrix} 2.2459e-05 & -7.8692e-06 & -0.0022967 \\ 2.7537e-05 & 1.3653e-05 & -0.0055154 \\ -0.0060626 & -0.0011562 & 1 \end{bmatrix}$$

$$e\_diff\_cross\_n = \begin{bmatrix} 158.95 \\ -87.157 \end{bmatrix}$$

$$e\_diff\_Fund\_n = \begin{bmatrix} 158.95 \\ -87.157 \end{bmatrix}$$

$$e\_diff2\_n = \begin{bmatrix} -449.63 \\ 111.28 \end{bmatrix}$$

$$e\_diff\_Fund\_n2 = \begin{bmatrix} -449.63 \\ 111.28 \end{bmatrix}$$

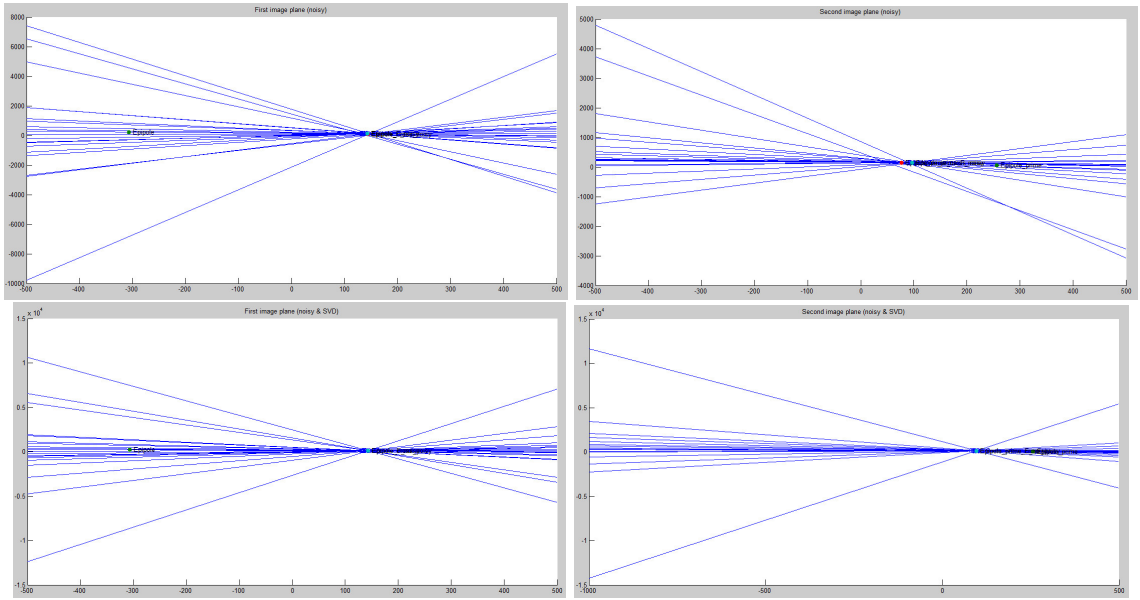

Figure 9: Epipoles and epipolar lines for both cameras when when the 2D points are noisy($\sigma = 1$), in the first row, and when forcing the noisy Fundamental matrix to be rank-2 again, in the second row, for SVD method

This case, in fact, shows slightly higher discrepancies due to the higher value of sigma.

The following tables compare the two methods for calculating the Fundamental matrix (using SVD and Least-square) when they are affected by noise. The first table presents the values of the mean and standard deviation of the set of differences between the noisy points and their corresponding epipolar lines when the used method is LMS for different sigmas and number of points. In fact, the shown results are the average of the obtained numbers from both cameras. The second table shows the same parameters but for the SVD method. The third table presents the difference between the first two in order to compare them easily and to know which of both fundamental matrices minimizes the distance between points and epipolar

lines.

#points	10		20		50		100	
Sigma	Mean	SD	Mean	SD	Mean	SD	Mean	SD
0	0	0	0	0	0	0	0	0
0.05	0.13057	0.10711	0.051514	0.030174	0.047046	0.031811	0.05655	0.037453
0.1	0.2822	0.31121	0.053656	0.033352	0.085017	0.069984	0.1159	0.087278
0.5	0.71678	0.96626	0.70221	0.56938	1.2609	1.4305	0.89993	0.89607
1	1.9336	1.7245	1.9665	1.7148	3.5097	5.2031	3.3004	5.0591

Table 1: Results of the Least Square method

#points	10		20		50		100	
Sigma	Mean	SD	Mean	SD	Mean	SD	Mean	SD
0	0	0	0	0	0	0	0	0
0.05	0.1301	0.10653	0.051512	0.030173	0.047044	0.03181	0.056547	0.037452
0.1	0.28219	0.31116	0.053654	0.033352	0.08501	0.069978	0.11589	0.087263
0.5	0.7167	0.96622	0.70219	0.56937	1.2609	1.4305	0.89992	0.89606
1	1.9333	1.7244	1.9664	1.7148	3.5097	5.2028	3.3004	5.0589

Table 2: Results of the SVD method

#points	10		20		50		100	
Sigma	Mean	SD	Mean	SD	Mean	SD	Mean	SD
0	0	0	0	0	0	0	0	0
0.05	0.00047462	0.00058004	1.3606e-06	8.3737e-07	2.2423e-06	1.6271e-06	2.863e-06	1.8661e-06
0.1	1.413e-05	5.355e-05	1.6064e-06	5.7011e-07	7.2122e-06	6.464e-06	1.1624e-05	1.4983e-05
0.5	8.5319e-05	3.2321e-05	1.7187e-05	1.4335e-05	6.3914e-07	7.5942e-07	4.6827e-06	1.0019e-05
1	0.00033532	0.00014806	1.2265e-05	1.5074e-05	1.5666e-05	0.00029252	0.0004	0.0002

Table 3: Difference between the first and second method (LS-SVD)

The previous tables reveal the following facts:

1. For the same number of points, the mean and SD increase for higher values of sigma.
2. Increasing the number of points has the effect of increasing the accuracy and consequently decreasing the mean and SD. However, it is possible that the mean and SD will increase even with more points since there is still a small probability of getting high noise values outside the expected range, which happened in some of the presented values.
3. The table of difference shows that the difference is always positive, even it is small, which indicates that SVD method is slightly better than LS method in terms of minimizing the distance between points and epipolar lines.

3 Optional Part

In this part, the entire geometry of both cameras, 3D points, π plane, and the coordinates systems are drawn in Matlab. The detailed code for this part is included in the .m file submitted with this report. The following images demonstrate the results of running this code:

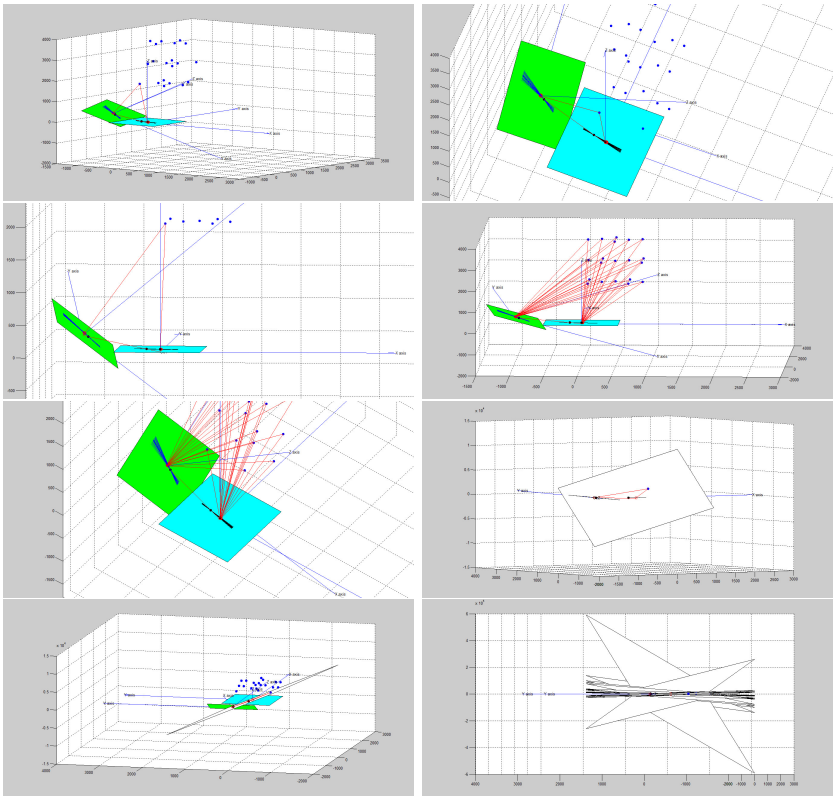

Figure 10: The entire simulated environment of the two cameras

4 Conclusion

In this lab assignment, two cameras are simulated successfully starting from a set of 3D points, obtaining their projections on both cameras, calculating the Fundamental matrices using two methods (LS & SVD), studying the effect of noise on those matrices and consequently on the entire environment. Moreover, the effect of the noise on these methods is inspected by finding the accuracy of the Fundamental matrix before and after adding the Gaussian noise, and a number of comparisons are drawn. Indeed, considerable attention should be paid to the structure of the implemented code and the different parameters names so that nothing will be overwritten.