Reconstruction from two views

Daniel Barmaimon
Visual Perception - Vibot Master Degree
University of Girona

April 13, 2014

Introduction

The main purpose of this lab is to understand how to reconstruct a 3D point from the projections of two different cameras. Differences between methods and the influence of noise over them would be also studied.

Part 1 - Eight points method and least squared

Step 1 - Define camera 1 with intrinsic and extrinsic parameters

The coordinate system for the first camera will be set as the world reference, simplifying the computation.

```
au1 = 100;  % Relation: f. length and pixel area
av1 = 120;  % Relation: f. length and pixel area
uo1 = 128;  % Principal p. (x, i. plane)
vo1 = 128;  % Principal p. (y, i. plane)
%Image size: 256 x 256
% Defining intrinsic matrix
A1 = [au1 0 uo1;...
      0 av1 vo1;...
      0 0 1]
```

Step 2 - Define camera 2 with respect to camera 1

```
au2 = 90;   % Relation: f. length and pixel area
av2 = 110;  % Relation: f. length and pixel area
uo2 = 128;  % Principal p. (x, i. plane)
vo2 = 128;  % Principal p. (y, i. plane)
ax = 0.1    % Rot. around x axis in rad;
by = pi/4.0 % Rot. angle around y axis in rad;
cz = 0.2    % Rot. angle around z axis in rad;
%XYZ EULER
tx = -1000; %Distances in mm
ty = 190;   %Distances in mm
tz = 230;   %Distances in mm
%Image size: 256 x 256
% Defining intrinsic matrix
A2 = [au2 0 uo2;...
      0 av2 vo2;...
      0 0 1]
```

Step 3 - Define extrinsic matrices for both cameras, and translation and rotation between both

The camera two will be translated and rotated with respect to the world (camera 1 coordinate system) and it will be reflected in the definition of the camera with respect to the world.

```
% Defining extrinsic matrix camera 1
```

```
E1 = [1 0 0;...      % Camera 1 is set as the
      0 1 0;...      % reference for the world:
      0 0 1]         % E1 = I

% Defining extrinsic matrix camera 2
R1 = [1 0 0; 0 cos(ax) -sin(ax); 0 sin(ax) cos(ax)];
R2 = [cos(by) 0 sin(by); 0 1 0; -sin(by) 0 cos(by)];
R3 = [cos(cz) -sin(cz) 0; sin(cz) cos(cz) 0; 0 0 1];

R = R1*R2*R3;        % Rotation Matrix
t = [tx ty tz]';     % Translation

T = [0 -tz ty;...
     tz 0 -tx;...
     -ty tx 0]       % Antisymmetric Matrix
```

Step 4 - Calculate the fundamental matrix analytically

The fundamental matrix will contain the intrinsic parameters of both cameras and the rigid transformation of one of them respect to the other.

$$F = A_2^{-t} R^t T A_1^{-1} \tag{1}$$

```
F = (inv(A2))'*R'*T*inv(A1);
F = F/F(3,3)
```

The normalization of the matrix would be useful to compare with other results that will be obtained later.

```
--------------------------
STEP 4

F =

    0.0000    0.0001   -0.0066
    0.0000   -0.0000    0.0130
   -0.0096   -0.0173    1.0000

--------------------------
```

Figure 1: Analytic result for F

Step 5 - Define the following set of object points with respect to the world coordinate system

```
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];
```

Be aware that 3D points are represented in homogeneous coordinates.

Step 6 - Compute the couples of image points in both image planes by using the matrices of Step 3

```
ex1 = [0 0 1]'  % Extension for comp. reason
c1Kw = [E1,ex1] % Extrinsic m. camera 1
ex2 = -(R'*t)   % Rigid transformation
c2Kw = [R',ex2] % Extrinsic m. camera 2

M1 = A1*c1Kw;   % Camera m. for camera 1
M2 = A2*c2Kw;   % Camera m. for camera 2
M1 = M1/M1(3,4);% Normalization
M2 = M2/M2(3,4);% Normalization

Vc1 = projectingPoints(V', M1);
Vc2 = projectingPoints(V', M2);
for i=1:size(Vc1,2)
    Vc1(:,i) = Vc1(:,i)/Vc1(3,i);
    Vc2(:,i) = Vc2(:,i)/Vc2(3,i);
end
```

Step 7 - Draw the 2D points obtained in step 6

In the following figure is possible to distinguish the projection of the 3D points over the two image planes.

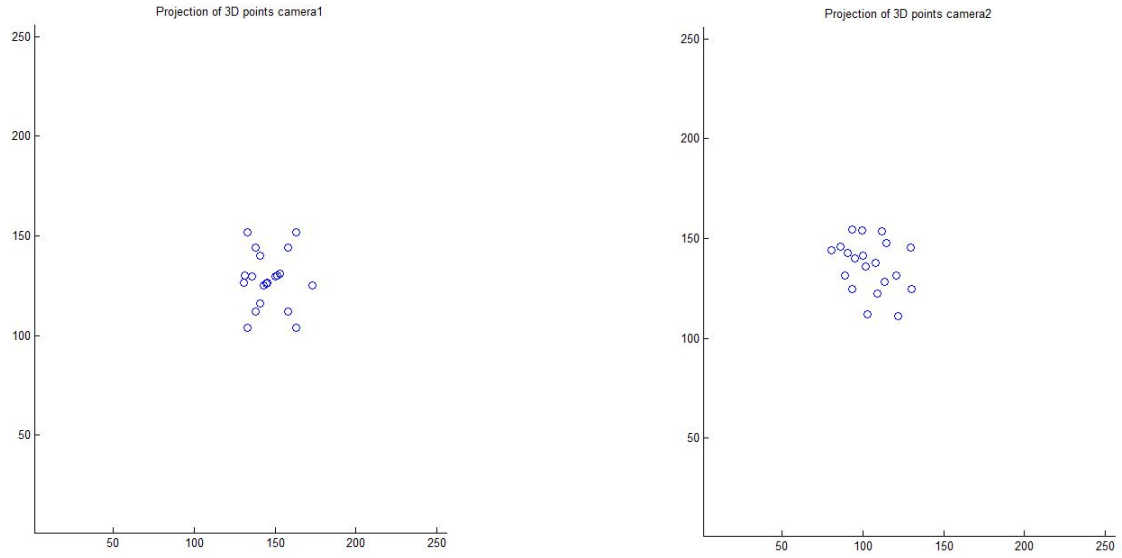

Figure 2: Projection of 3D points over image planes

Step 8 - Compute the fundamental matrix by using the 8-point method and least-squares by means of the 2D points obtained in step 6

A function to compute the fundamental matrix using this method was implemented.

```
function [ F ] = fundamental( points, pointsC1, pointsC2 )
%   This function will return the fundamental matrix given
%   a number of projected points in the two cameras
%   This number should be at least 8

number = size(points, 2);

% The parameter f(3,3) will be fixed to 1
f = zeros(1,8);
u = zeros(number,8);
o = ones(number,1);
```

```
12
13 % Composition of matrix u.
14 % Warning: our matrix is in the form m'F'm
15 for i=1:number
16     u(i,1)=pointsC1(1,i)*pointsC2(1,i);
17     u(i,2)=pointsC1(2,i)*pointsC2(1,i);
18     u(i,3)=pointsC2(1,i);
19     u(i,4)=pointsC1(1,i)*pointsC2(2,i);
20     u(i,5)=pointsC1(2,i)*pointsC2(2,i);
21     u(i,6)=pointsC2(2,i);
22     u(i,7)=pointsC1(1,i);
23     u(i,8)=pointsC1(2,i);
24 end
25 f = (-pinv(u)*o)';
26 f = [f, 1];
27 F = reshape(f, 3,3);
28 F = F';
29 end
```

Step 9 - Compare the step 8 matrix with the one obtained in step 4

The comparison of the two matrices and the error computed for the subtraction of them is collected in Fig.3

```
--------------------------
STEP 9
This is fundamental matrix with 8 points method and least squares

F1 =

    0.0000    0.0001   -0.0066
    0.0000   -0.0000    0.0129
   -0.0096   -0.0173    1.0000

This if fundamental matrix with intrinsec and extrinsec matrices

F =

    0.0000    0.0001   -0.0066
    0.0000   -0.0000    0.0130
   -0.0096   -0.0173    1.0000

The error between the two matrices is error = F - F1

error1 =

   1.0e-04 *

    0.0005    0.0031    0.2591
    0.0006    0.0009    0.5676
    0.0364    0.5161         0

--------------------------
```

Figure 3: Comparison of the results for F

Step 10 - Draw in the windows of step 7 all epipolar geometry

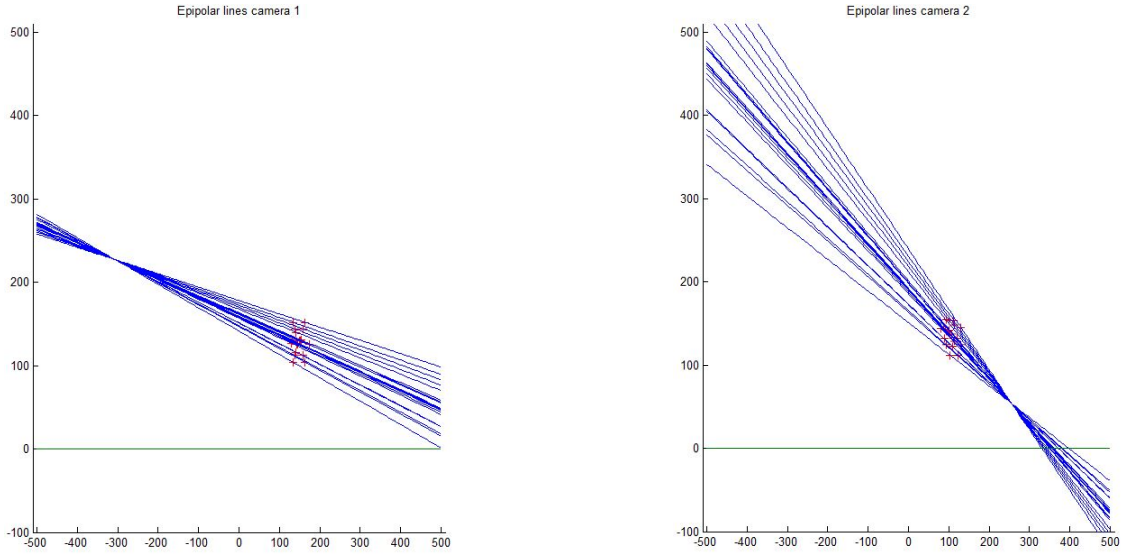

Figure 4: Projected points and epipole lines using 8 point method with least squares method and points without noise

To appreciate that all the points are crossed by an epipole line the Fig.5 is shown.

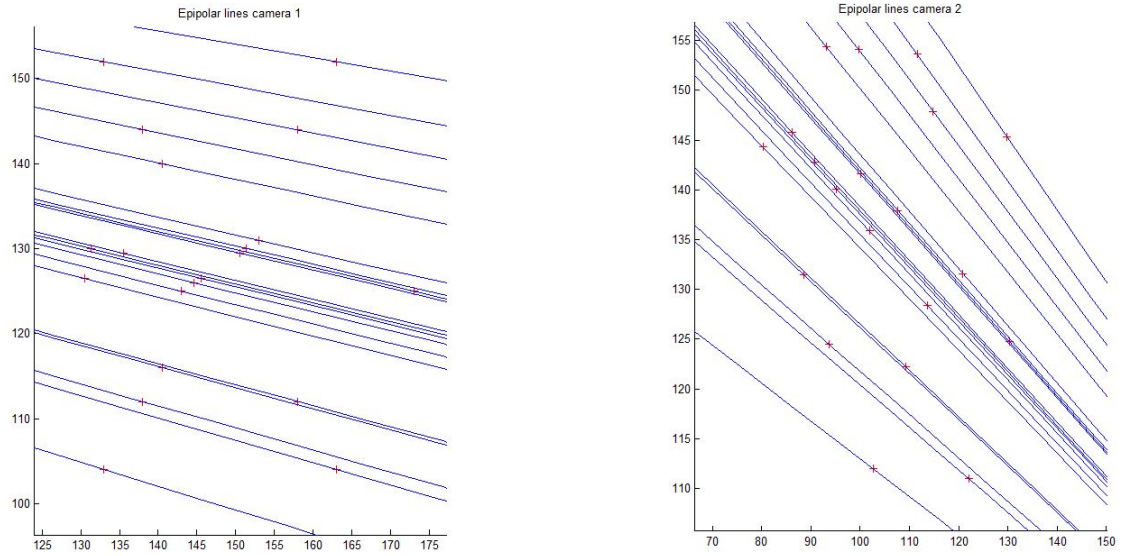

Figure 5: Projected points and epipole lines using 8 point method with least squares method and points without noise

Step 11 - Add some Gaussian noise to , pixels the 2D points producing discrepancies between the range[-1,+1] pixels for the 95% of points

For this section for future ones, a function to add noise has been created. The inputs parameters are the projected points and the values for mean and standard deviation of the normal distribution (noise) to be added.

```
function [ noisyPoints ] = addNoise(points, mean, sigma)
% This function will return the points introduced with a gaussian noise
% over them
% Inputs: points = Normalized points in images plane
%         mean   = Mean value for the gaussian function to generate the noise
```

```
%         sigma  = Sigma value for the gaussian function to generate the noise
% Outputs: noisyPoints = original image plane points with the noise added

noise = normrnd(mean, sigma,[2,size(points,2)]); % 2*sigma = 1 (we want 95% of data)
noise = [noise; zeros(1,size(points,2))];
noisyPoints = points + noise;
end
```

The results of the implementation of the eight point method over noisy points will lead to a non-unique epipole as it can be seen in Fig. 6

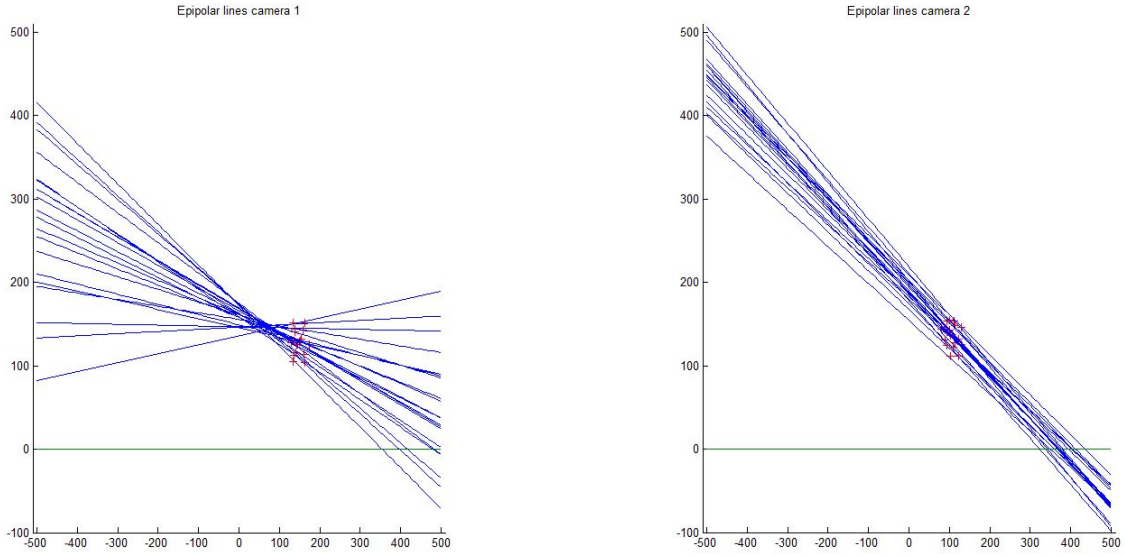

Figure 6: Projected points and epipole lines using 8 point method with least squares method and points with gaussian noise 1

Step 12 - Compare the epipolar geometry obtained

In this case the epipolar lines don't cross all the points. This is the effect of the noise and the cause of no having a unique epipole. A zoom view of the last images is shown in Fig. 7 The results of comparing the matrices obtained could be check in

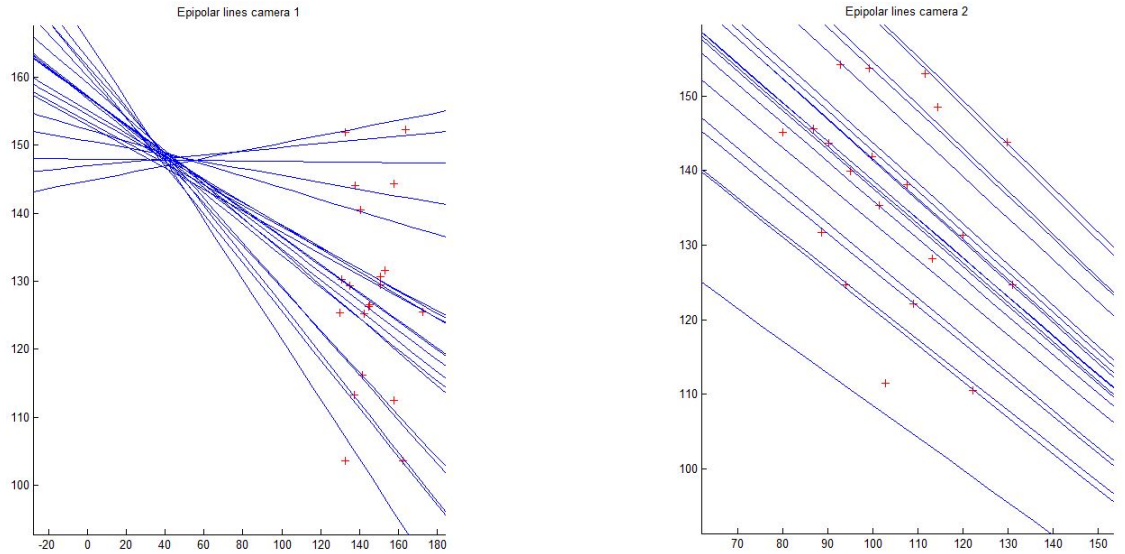

Figure 7: Detail of projected points and epipole lines using 8 point method with least squares method and points with Gaussian noise ($\mu = 0, \sigma = 0.5$)

the next lines. The result of this approximation could be observed in Fig. 9

```
--------------------------
STEP 12
Calculating fundamental matrix with 10 noisy points
This is fundamental matrix with 8 points method and 20 noisy points

F2 =

    0.0000    0.0000   -0.0041
    0.0000    0.0000    0.0011
   -0.0058   -0.0087    1.0000

This if fundamental matrix with intrinsec and extrinsec matrices

F =

    0.0000    0.0001   -0.0066
    0.0000   -0.0000    0.0130
   -0.0096   -0.0173    1.0000

The error between the two matrices is error = F - F2

error2 =

    0.0000    0.0000    0.0026
    0.0000    0.0000    0.0119
    0.0038    0.0087         0
```

Figure 8: Comparison of the results for F

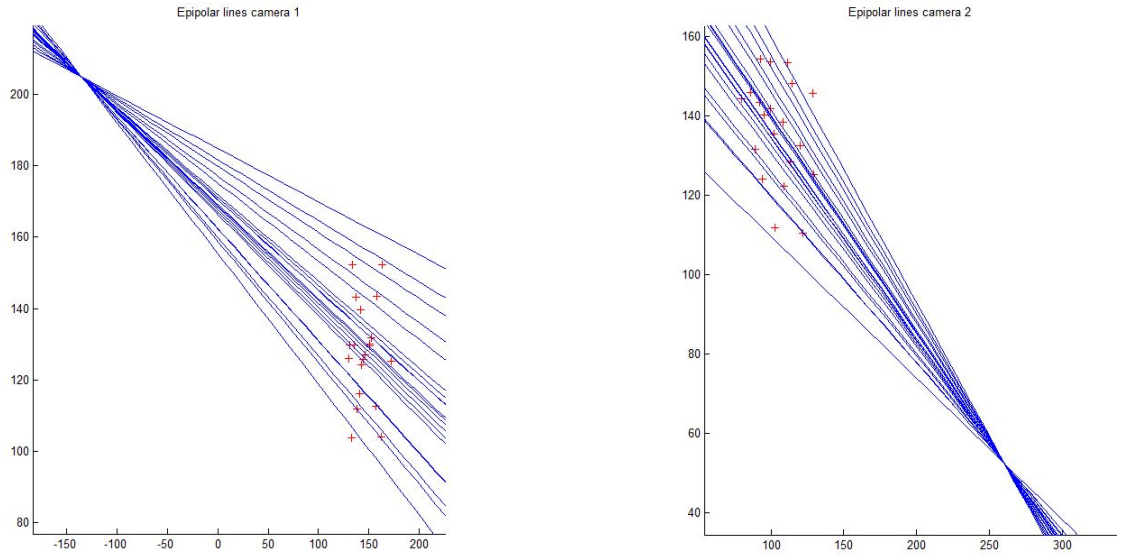

Figure 9: Approximation to a unique epipole using 8 point method with least squares method and points with Gaussian noise ($\mu = 0, \sigma = 0.5$)

It is possible to analyze what will happen if an approximation is made. For getting a unique epipole, a SVD decomposition will be made over the fundamental matrix calculated. The smallest eigen value will be set to zero, reducing the rank of the matrix to 2. The epipole will be unique this time.

Step 13 - Add more noise, Gaussian with range $[-2, +2]$ for 95 % of points and repeat the experiment

With more noise is expected a bigger difference in the values of the fundamental matrices and a bigger distance between the projected points and the epipolar lines. The results of the fundamental matrix obtained and the comparison with the one

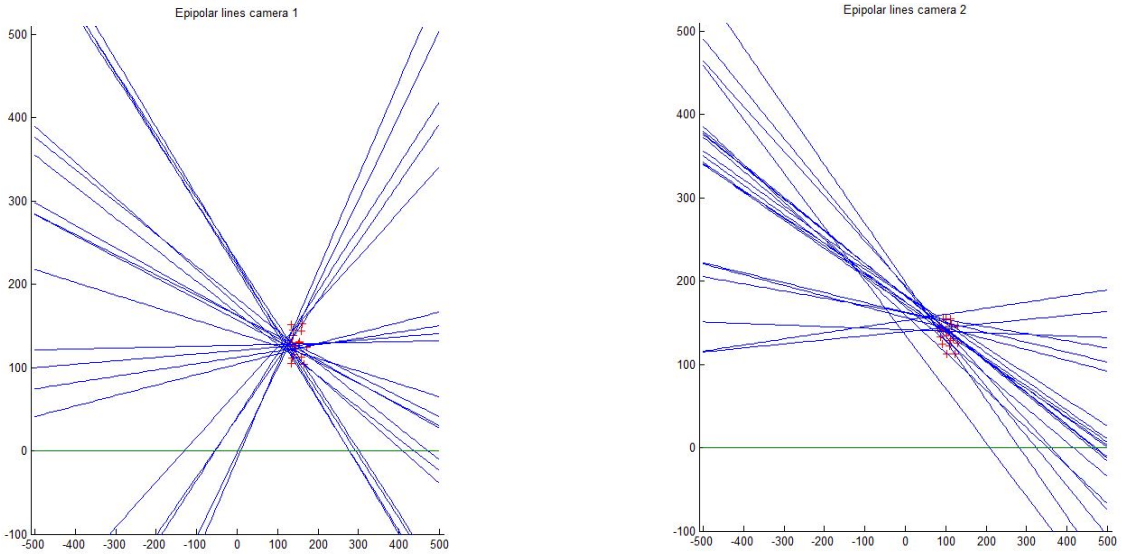

Figure 10: Projected points and epipole lines using 8 point method with least squares method and points with gaussian noise 2

without noise is shown below in Fig.11 The results of comparing the matrices obtained could be check in the next lines. A

```
--------------------------
STEP 13
Calculating fundamental matrix with 20 noisy points
This is fundamental matrix with 8 points method and 20 noisy points

F4 =

    0.0000   -0.0000   -0.0019
    0.0000    0.0000   -0.0058
   -0.0060   -0.0013    1.0000

This if fundamental matrix with intrinsec and extrinsec matrices

F =

    0.0000    0.0001   -0.0066
    0.0000   -0.0000    0.0130
   -0.0096   -0.0173    1.0000

The error between the two matrices is error = F - F4

error4 =

    0.0000    0.0001    0.0047
    0.0000    0.0000    0.0188
    0.0036    0.0160         0
```

Figure 11: Comparison of the results for F

preview of the results in Fig.10 allows to distinguish a greater amplitude in the range for the angle of the epipolar lines around the epipoles area, what denote the great influence of the noise over this method. For a closer view of the points and the lines Fig. 12 will be shown.

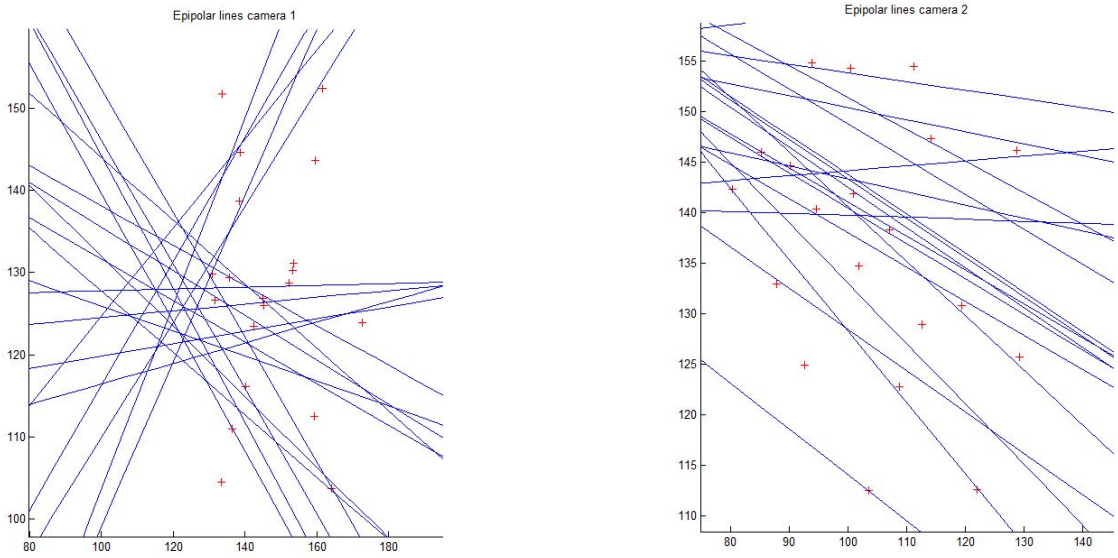

Figure 12: Detail of projected points and epipole lines using 8 point method with least squares method and points with Gaussian noise ($\mu = 0, \sigma = 1$)

Part 2 - Eight points method and SVD

In this section another linear method to solve the problem will be used. In this case is a variation of least squares with 8 points and is also called *orthogonal least squares method*. In this case the distance between the points and the epipolar lines are considered perpendicular distances to the lines and not in the *y-axis* direction.

Step 14 - Compute the fundamental matrix by using the 8-points method and SVD from the points 2D obtained in step 6 and without noise. Compare the obtained matrix with the one obtained in step 8.

In this case the way to calculate F is a little different. A nine column matrix will be used, setting a column of ones in the last position. The matrix F will be calculated by analyzing the matrix U^tU. The decomposition of this matrix in its SVD form will allow to get F as the eigen-vector related with the smallest eigen-value. This is the same as getting F that minimizes the orthogonal distances from the epipolar lines to the projected points. To get the matrix F a function in Matlab has been implemented as follows.

```
function [ Fsvd ] = svdMethod( points, pointsC1, pointsC2 )
%   This function will return the fundamental matrix given
%   a number of projected points in the two cameras
%   This number should be at least 8 using the SVD method

number = size(points, 2);

% The parameter f(3,3) will be fixed to 1 avoiding a zero vector solution
f = zeros(1,8);
u = zeros(number,9);
o = ones(number,1);

% Composition of matrix u. Warning: our matrix is in the form m'F'm
for i=1:number
    u(i,1)=pointsC1(1,i)*pointsC2(1,i);
    u(i,2)=pointsC1(2,i)*pointsC2(1,i);
    u(i,3)=pointsC2(1,i);
    u(i,4)=pointsC1(1,i)*pointsC2(2,i);
    u(i,5)=pointsC1(2,i)*pointsC2(2,i);
    u(i,6)=pointsC2(2,i);
    u(i,7)=pointsC1(1,i);
    u(i,8)=pointsC1(2,i);
    u(i,9)=1;
end
A = u'*u;
[S, V, D]=svd(A);
Fsvd = S(:,9);              % Eigenvector of smaller eigenvalue is F
Fsvd = reshape(Fsvd, 3,3);  % It is needed to reshape
Fsvd= Fsvd';                % And as matlab reshape in column form, it
end                         % is needed to transpose after reshape.
```

The fundamental matrix obtained and the comparison with the one got with the least squares method is shown below in Fig.13. They are quite similar but it was a result without noise. The representation of the points and the epipolar lines for this method without noise could be appreciated in Fig.14

```
--------------------------
STEP 14
This is fundamental matrix with 8 points method and least squares

F1 =

    0.0000    0.0001   -0.0066
    0.0000   -0.0000    0.0129
   -0.0096   -0.0173    1.0000

This if fundamental matrix with 8 points method and SVD

Fsvd =

    0.0000    0.0001   -0.0066
    0.0000   -0.0000    0.0129
   -0.0096   -0.0173    0.9997
```

Figure 13: Comparison of the results for F

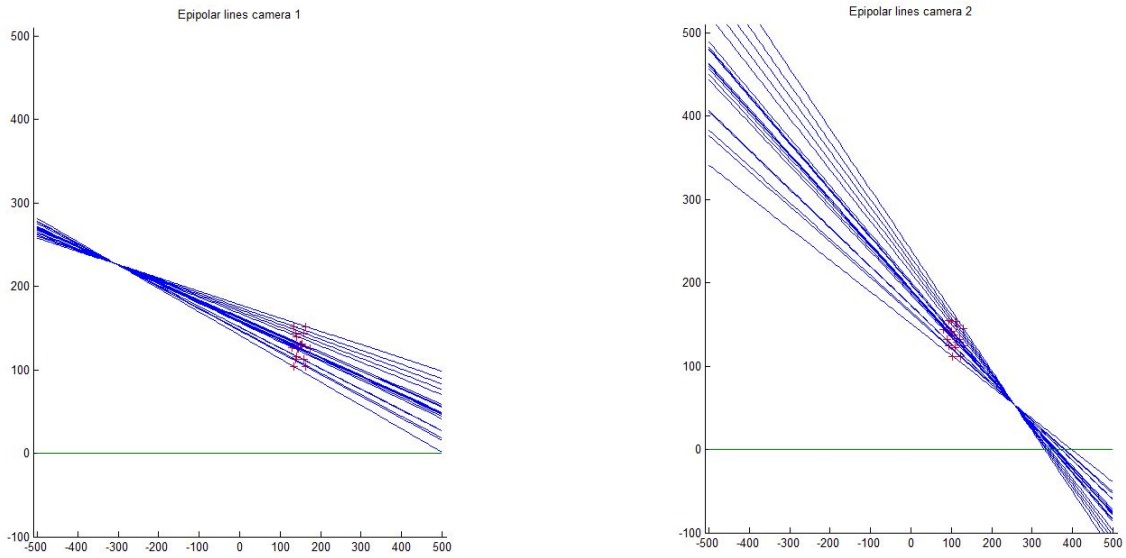

Figure 14: Projected points and epipole lines using 8 point method with SVD method and points without noise

Step 15 - Repeat step 10 up to 13 (with the matrix of step 14 instead of step 8) for some Gaussian noise first in the rang [-1, 1] and then in the rang [-2, 2] for the 95% of points.

Gaussian noise 1

In this case a Gaussian noise with $\mu = 0, \sigma = 0.5$ will be used to compute the result of SVD method. The values for the

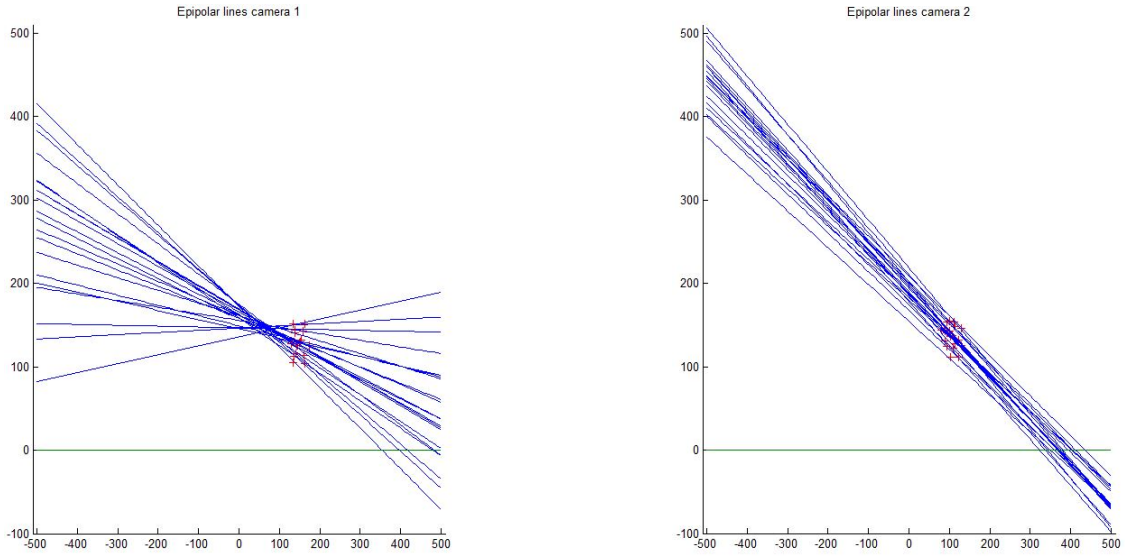

Figure 15: Projected points and epipole lines using 8 point method with SVD method and points with gaussian noise 1

matrix F in this case are shown below. To check the detail of the possible intersection of these lines and the points the Fig.17

```
---------------------------
STEP 15
Gaussian noise, sigma = 0.5
Using the same noisy points of step 10 (in step 15)
Calculating fundamental matrix with 10 noisy points for svd method
This is fundamental matrix with 8 points and least squares method, and 10 noisy points

F2 =

    0.0000   -0.0000   -0.0021
    0.0000    0.0000   -0.0054
   -0.0061   -0.0015    1.0000

This if fundamental matrix with 8 points and svd method, and 10 noisy points

FsvdN1 =

    0.0000   -0.0000   -0.0021
    0.0000    0.0000   -0.0054
   -0.0061   -0.0015    1.0000
```

Figure 16: Comparison of the results for F

is plotted.

After getting the decomposition of F with SVD values and setting to zero the last eigen-value, a unique epipole if obtained (Fig.18)

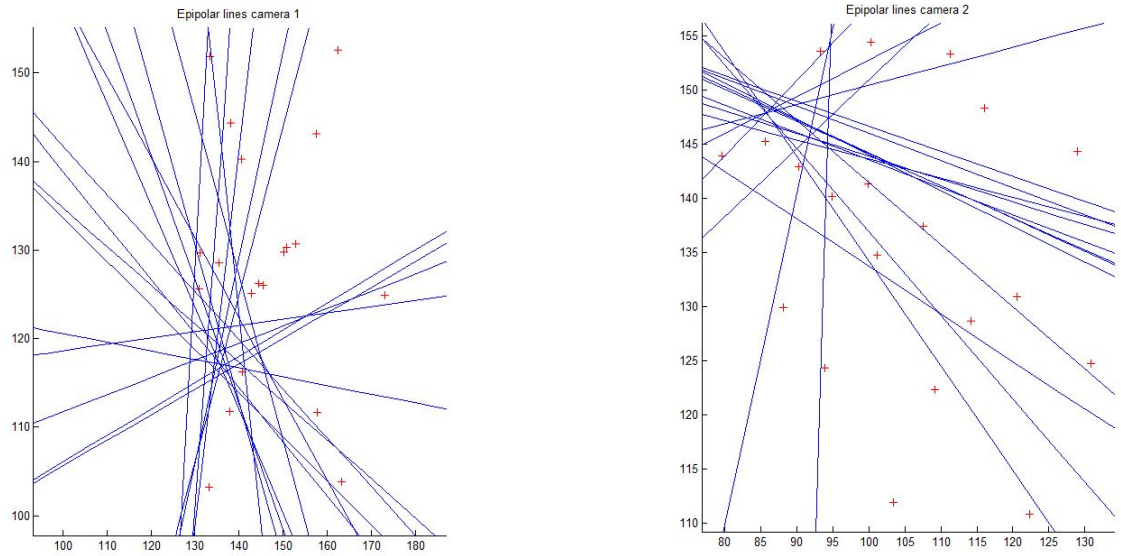

Figure 17: Detail of projected points and epipole lines using 8 point method with SVD method and points with Gaussian noise ($\mu = 0, \sigma = 0.5$)

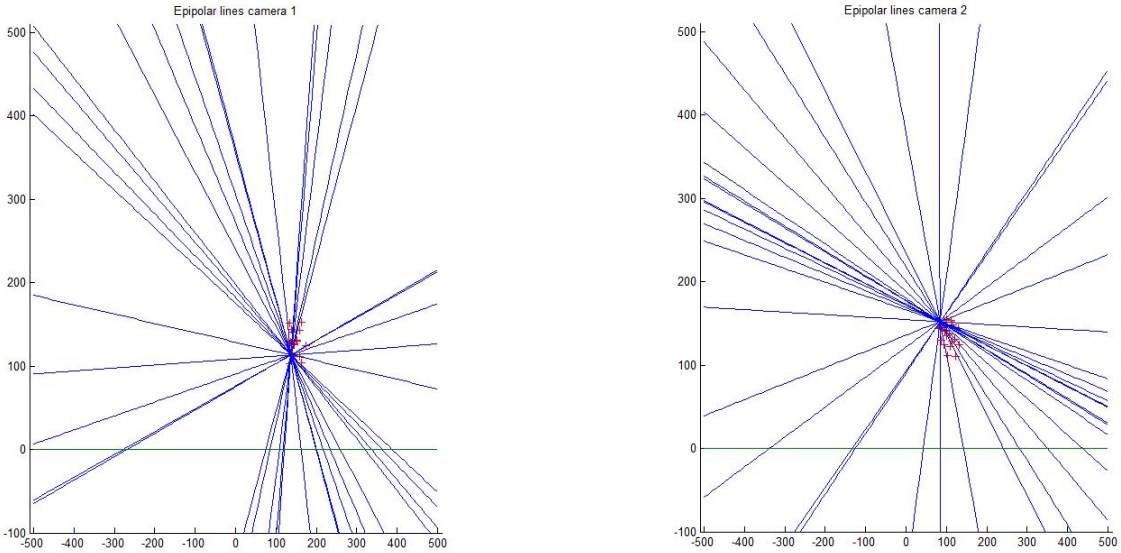

Figure 18: Approximation to a unique epipole using 8 point method with SVD method and points with Gaussian noise ($\mu = 0, \sigma = 0.5$)

Gaussian noise 2

If now the noise applied has double the range of previous one the results will vary in the following way. The epipolar lines and points are shown for this case in Fig.19 To check the detail of the possible intersection of these lines and the points the

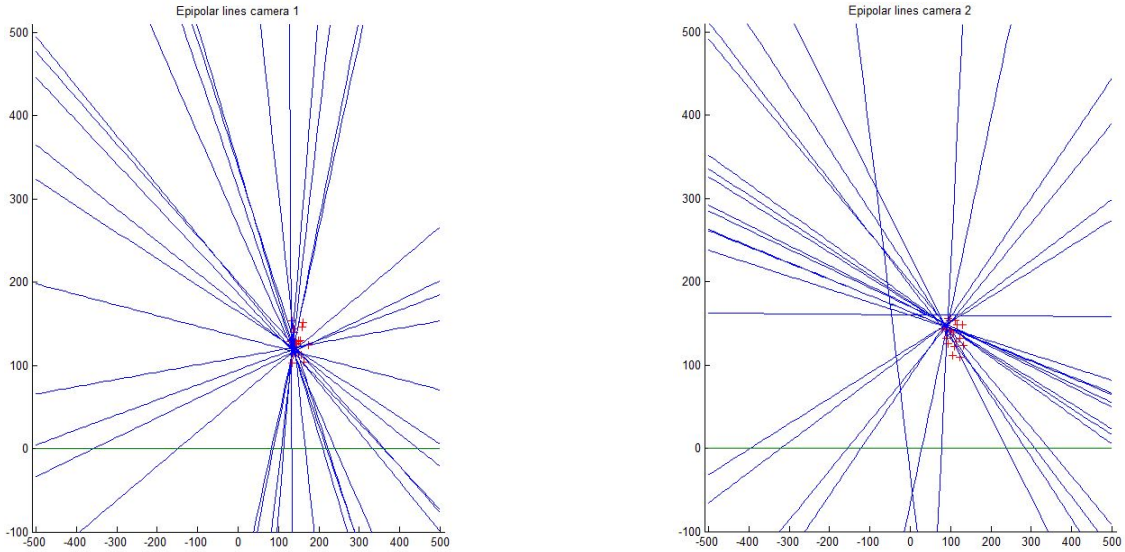

Figure 19: Projected points and epipole lines using 8 point method with SVD method and points with gaussian noise 2

Fig.20 is plotted. The fundamental matrix obtained in this occasion is represented in Fig.21 After getting the decomposition of F with SVD values and setting to zero the last eigen-value, a unique epipole if obtained (Fig.22)

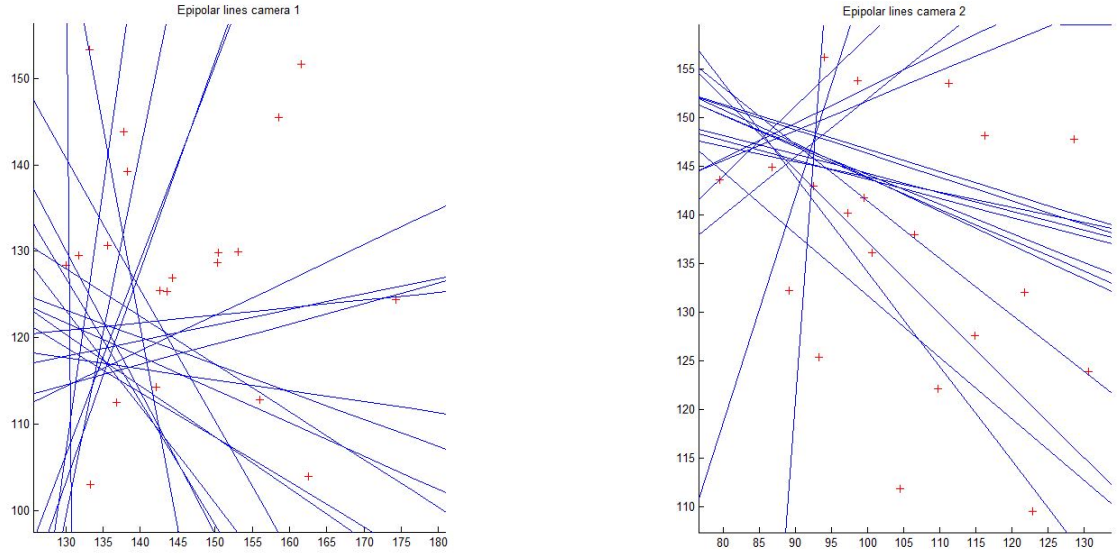

Figure 20: Detail of projected points and epipole lines using 8 point method with SVD method and points with Gaussian noise ($\mu = 0, \sigma = 1$)

```
--------------------------
Gaussian noise, sigma = 1
Using the same noisy points of step 11 (in step 15)
Calculating fundamental matrix with 20 noisy points for svd method
This is fundamental matrix with 8 points and least squares method, and 10 noisy points

F4 =

    0.0000   -0.0000   -0.0022
    0.0000    0.0000   -0.0053
   -0.0062   -0.0014    1.0000

This if fundamental matrix with 8 points and svd method, and 10 noisy points

FsvdN2 =

    0.0000   -0.0000   -0.0022
    0.0000    0.0000   -0.0053
   -0.0062   -0.0014    1.0000

The error between the two matrices is error = FsvdN2 - F4

error7 =

   1.0e-04 *

    0.0000    0.0000    0.0023
    0.0000    0.0000    0.0004
    0.0008    0.0025    0.3637
```

Figure 21: Comparison of the results for F

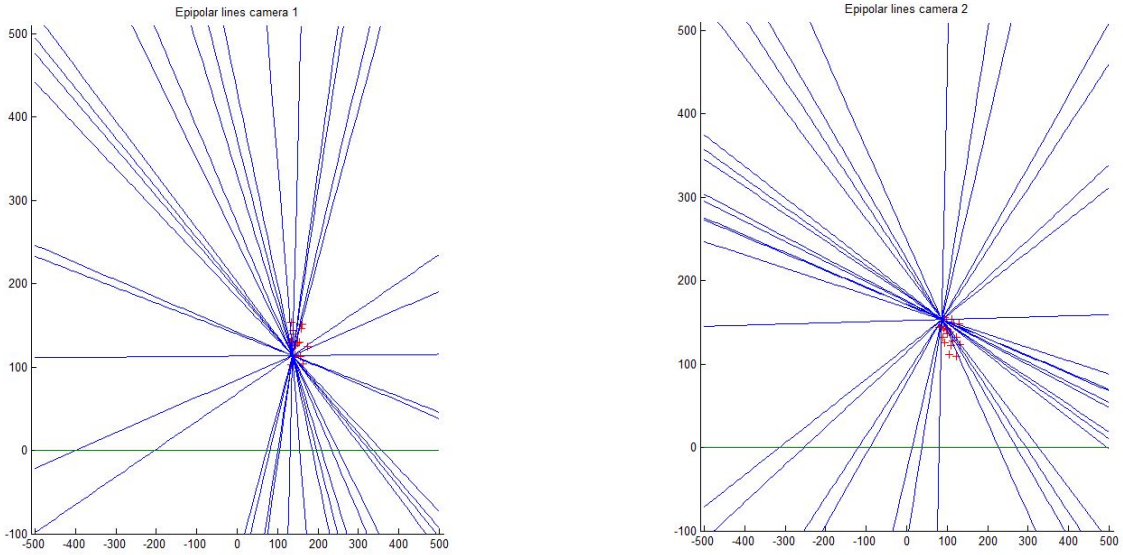

Figure 22: Approximation to a unique epipole using 8 point method with SVD method and points with Gaussian noise ($\mu = 0, \sigma = 1$)

Compare the epipolar geometry obtained in step 15 (using SVD) with the one obtained in steps 11 and 13 (using LS). Which of both fundamental matrices minimizes the distance between points and epipolar lines?

For this case two functions in Matlab had been implemented. First one will compute the minimum distance between the epipolar line and the projection of the 3D point. Second one will add all the distances for each of the cases, allowing the comparison of these distances.

```
function distance = dist2line( x,y, line2D)
% This function will return the minimum distance between a point and a line
% considering the line in general form, and 2D point as P(x,y)
distance = abs(line2D(1)*x+line2D(2)*y+line2D(3))/sqrt((line2D(1))^2+(line2D(2))^2);
end
```

```
function [mean, stdDev] = computeDistances( lines2D, points )
% This function will return the mean of total distance for all lines with respect
% the points that should be contained by the line itself
mean = 0;
stdDev = 0;
for i=1:size(lines2D,2)
    mean = mean + dist2line(points(1,i),points(2,i),...
        lines2D(:,i));
end
mean = mean /size(lines2D,2);
for i=1:size(lines2D,2)
    stdDev = stdDev +(mean - dist2line(points(1,i),points(2,i),lines2D(:,i)))^2;
end
stdDev = sqrt(stdDev/size(lines2D,2));
end
```

After using these two functions, the values obtained for each of the cases is shown in the following tables.

Case	Method	Noise	Mean Distance (mm)
1	Least squares	-	$1.8796 * 10^{-12}$
2	Least squares	Gaussian 1	1.1761
3	Least squares	Gaussian 1 (1 epipole)	2.6700
4	Least squares	Gaussian 2	2.7202
5	Least squares	Gaussian 2 (1 epipole)	26.1022
6	SVD	-	$1.0240 * 10^{-11}$
7	SVD	Gaussian 1	1.1760
8	SVD	Gaussian 1 (1 epipole)	2.6692
9	SVD	Gaussian 2	1.8428
10	SVD	Gaussian 2 (1 epipole)	26.1008

Case	Method	Noise	Std Deviation Distance (mm)
1	Least squares	-	$5.9041 * 10^{-13}$
2	Least squares	Gaussian 1	0.9649
3	Least squares	Gaussian 1 (1 epipole)	1.6945
4	Least squares	Gaussian 2	2.8586
5	Least squares	Gaussian 2 (1 epipole)	8.0474
6	SVD	-	$1.0240 * 10^{-11}$
7	SVD	Gaussian 1	1.6955
8	SVD	Gaussian 1 (1 epipole)	1.1434
9	SVD	Gaussian 2	1.8428
10	SVD	Gaussian 2 (1 epipole)	8.0471

It was expected a much better results for the SVD method as the ones that could be found in the ref.[1].

General comments

It was a nice exercise to understand all the maths that is behind the 3D reconstruction using two cameras. It was quite challenging and the workload was adequate for the time given. The only thing missed, was a 3D implementation of the model (tried but not achieved), with and without noise for the different methods.

References

[1] Xavier Armangu, Joaquim Salvi, "Overall view regarding fundamental matrix estimation", University of Girona, 24 October 2002