University of Girona

Visual Perception

Lab 3

Compute the Fundamental Matrix from two cameras

Author:
Di MENG

Supervisor:
Pro. Joaquim Salvi

April 16, 2017

1 Introduction

The objective of this lab is to compute fundamental matrix from two simulated cameras. Based on given parameters of cameras, we construct the transformation matrices of two cameras. The projected points on two camera planes can be computed from a set of 3d points by transformation matrices respectively. The fundamental matrix can be obtained by analytically, the 8-points method with Least Squares, 8-points method with Eigen Analysis. By increasing the noise of projection points, the epipolar geometries related to fundamental matrices from two method are compared.

2 Part 1

2.1 Step 1-2: Define Camera 1 and Camera 2 with given parameters

```
%% Parameter preparation
au1 = 100; av1 = 120; uo1 = 128; vo1 = 128;  % Camera 1

au2 = 90; av2 = 110; uo2 = 128; vo2 = 128;   % Camera 2
ax = 0.1; by = pi/4; cz = 0.2;
tx = -1000; ty = 190; tz = 230;
```

2.2 Step 3: Get the intrinsic transformation matrices of both cameras, and the rotation and translation between both cameras

The world coordinate is set to the coordinate system of camera 1 which means the rotation matrix of camera 1 is identity and translation is zero with respect to the world origin. The transformation matrices of two cameras are computed.

```
%% Intrinsic matrices and Transformation
wRc1 = [1, 0, 0; 0, 1, 0; 0, 0, 1];
wTc1 = [0; 0; 0];
c1Kw = [wRc1, wTc1];          % Camera 1 with respect to world coordinate
in_mat_c1 = [au1, 0, uo1; 0, av1, vo1; 0, 0, 1]; % Cam1 intrinsic matrix
in_mat_c1_corre = [au1, 0, uo1, 0; 0, av1, vo1, 0; 0, 0, 1, 0];
mat_0 = [0, 0, 0];
wKc1 = c1Kw;                                    % Cam1 extrinsic matrix
ex_mat_c1_corre = [wKc1; mat_0, 1];

Rot_x = [1, 0, 0; 0, cos(ax), -sin(ax); 0, sin(ax), cos(ax)];
Rot_y = [cos(by), 0, sin(by); 0, 1, 0; -sin(by), 0, cos(by)];
Rot_z = [cos(cz), -sin(cz), 0; sin(cz), cos(cz), 0; 0, 0, 1];

c1Rc2 = Rot_x * Rot_y * Rot_z;
c1Tc2 = [tx; ty; tz];
c2Kw = [c1Rc2, c1Tc2];         % Camera 2 with respect to camera 1 system
in_mat_c2 = [au2, 0, uo2; 0, av2, vo2; 0, 0, 1]; % Cam2 intrinsic matrix
in_mat_c2_corre = [au2, 0, uo2, 0; 0, av2, vo2, 0; 0, 0, 1, 0];
wKc2 = [c1Rc2', -c1Rc2'*c1Tc2];                 % Cam2 extrinsic matrix
ex_mat_c2_corre = [wKc2; mat_0, 1];
```

2.3 Step 4: Get the Fundamental matrix analytically as the product of matrices defined in step 3

According to the lecture, fundamental matrix can be computed analytically by the following equations:

$$F = A'^{-t} R^t [t]_x A^{-1}$$

The translation vector t is converted to the form of its antisymmetric matrix $[t]_x$.

```
%% Fundamental matrix
c1Tc2_x = [0, -tz, ty; tz, 0, -tx; -ty, tx, 0]; % antisymmetric matrix
F_mat = inv(in_mat_c2)'*c1Rc2'*c1Tc2_x*inv(in_mat_c1);
F_mat_norm = F_mat/F_mat(end);                  % Normalized Fundamental matrix
disp('Normalized Fundamental matrix');
disp(F_mat_norm);
```

Normalized fundamental matrix is $\begin{pmatrix} 3.2235e-05 & 7.2676e-05 & -0.0066178 \\ 2.4597e-05 & -2.3923e-05 & 0.012979 \\ -0.0095802 & -0.017343 & 1 \end{pmatrix}$

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

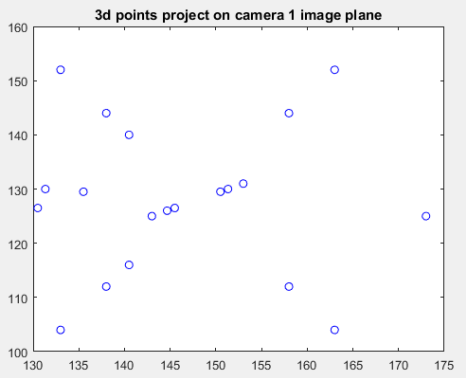
Twenty 3d points are defined with respect to the world coordinate saved in a 4x20 dimensional matrix(the code for this part is not displayed in order not to fill up the page). The variable "pt_num" is assigned the number of points.

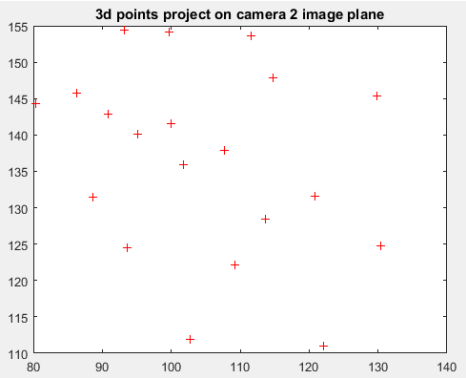
2.5 Step 6: Compute the couples of image points in both image planes by using the matrices of step 3

The corresponding intrinsic and extrinsic matrices are used to projects points on image planes.

```
%% Project 3d points onto both the two image planes
V_2d_c1 = [];                % Projection of 3d points on camera 1 plane

for n = 1:pt_num
    V_2d_c1(:,n) = in_mat_c1_corre*ex_mat_c1_corre*V_3d(:,n);
    V_2d_c1(1,n) = V_2d_c1(1,n)/V_2d_c1(3,n);
    V_2d_c1(2,n) = V_2d_c1(2,n)/V_2d_c1(3,n);
end
V_2d_c1(3,:) = [];
%disp(V_2d_c1);

V_2d_c2 = [];                % Projection of 3d points on camera 2 plane 2
for n = 1:pt_num
    V_2d_c2(:,n) = in_mat_c2_corre*ex_mat_c2_corre*V_3d(:,n);
    V_2d_c2(1,n) = V_2d_c2(1,n)/V_2d_c2(3,n);
    V_2d_c2(2,n) = V_2d_c2(2,n)/V_2d_c2(3,n);
end
V_2d_c2(3,:) = [];
%disp(V_2d_c2);
```

2.6 Step 7: Open two windows in matlab, which will be used as both image planes, and draw the 2D points obtained in step 6





2.7 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

The equation of system is following the form:

$$m'^{t} F m = 0$$

Where m' and m are the projection on image planes of two cameras respectively. After a set of operating, the fundamental matrix can be deduced:

$$F = -(U_n^t U_n)^{-1} U_n^t 1_n$$

The result of fundamental matrix from Least Square is:

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

```
%% Fundamental matrix computed by least-square method
U = [];
for n = 1:pt_num
    U(n,1) = V_2d_c1(1,n)*V_2d_c2(1,n);
    U(n,2) = V_2d_c1(2,n)*V_2d_c2(1,n);
    U(n,3) = V_2d_c2(1,n);
    U(n,4) = V_2d_c1(1,n)*V_2d_c2(2,n);
    U(n,5) = V_2d_c1(2,n)*V_2d_c2(2,n);
    U(n,6) = V_2d_c2(2,n);
    U(n,7) = V_2d_c1(1,n);
    U(n,8) = V_2d_c1(2,n);
end
%disp(U);
vector_ones = repelem(1,pt_num)';
F_leastS = [-inv(U'*U)*U'*vector_ones;1];
F_leastS = reshape(F_leastS,[3,3])';
disp('Fundamental matrix from least square method');
disp(F_leastS);
```

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

The fundamental matrices obtained analytically and from 8-point method with Least Square are exactly the same. Because the projections on image planes are computed by analytical fundamental matrix, and the fundamental matrix from Least Square method is computed by using these 2d points which is like backward operating.

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

Drawing epipolar geometry is written in a function:

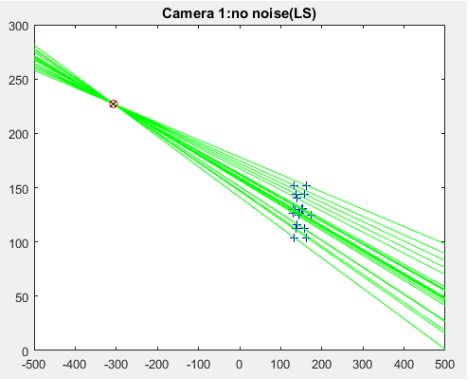
```
%% Draw epipolar geometry(LS)
draw_ep (V_2d_c1, V_2d_c2, F_leastS, c1Rc2, c1Tc2, in_mat_c1, in_mat_c2, 'no noise(LS)');
```

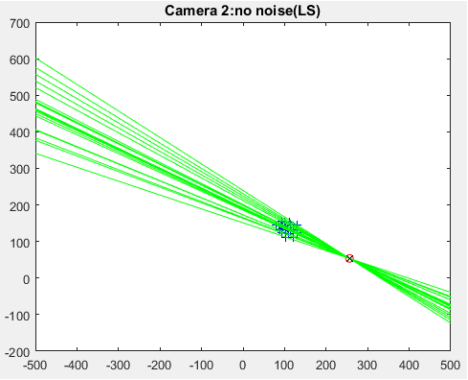
Epipolar line is defined as $l = Fm$. The slope and intersection with axis y can be extracted from this equation. Twenty epipolar lines are drawn in

each plane in this case.(Only the code of drawing epipolar geometry in image plane of camera 2 is shown here)

```
function draw_ep (V_2d_c1, V_2d_c2, F, c1Rc2, c1Tc2, in_mat_c1, in_mat_c2, str)

% Camera 2
figure, plot(V_2d_c2(1,:), V_2d_c2(2,:), 'b+');      % Draw the projections
title(strcat('Camera 2: ', str));
hold on
x = [-500,500];
for i = 1:size(V_2d_c1, 2)
    lm_2 = F * [V_2d_c1(:,i);1];
    y = -lm_2(1) / lm_2(2) * x - lm_2(3) / lm_2(2);  % epipolar line equation
    plot(x, y, 'g');

    if i == 1
        y_prev = y;
    end
    if i == 2
        [xi,yi] = polyxpoly(x,y_prev,x,y);
        plot(xi, yi, 'ro');               % epipole: epipolar lines intersection
    end
end
e_2 = in_mat_c2 * [c1Rc2', -c1Rc2'*c1Tc2] * [0;0;0;1];
plot(e_2(1)/e_2(3), e_2(2)/e_2(3), 'kx'); % epipole: projection of cam2 origin

hold off
```

The following figure shows the epipolar lines in green, the projections in blue, the epipole obtained from intersection of lines in red circle, the epipole from projection of the other camera in black cross. As we observed after zooming in, the points are perfectly lying on the epipolar lines.





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

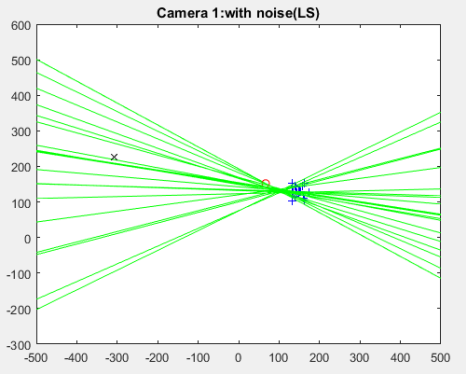
The range of noise is flexible here by changing the value of variable.

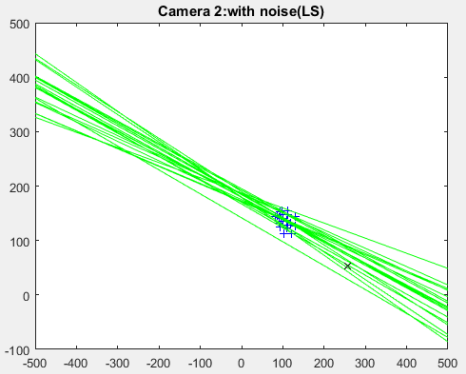
```
%% Add noise to 2d points
range_noi = 2;    % range of Gaussain noise
noise1 = range_noi*randn(2,pt_num)*0.5;
noise2 = range_noi*randn(2,pt_num)*0.5;
V_2d_c1_noi = V_2d_c1 + noise1;
V_2d_c2_noi = V_2d_c2 + noise2;
```

2.11 Step 12: Again repeat step 8 up to 10 with the noisy 2D points. Compare the epipolar geometry obtained

The fundamental matrix is recomputed using noisy 2d points by the same method as step 8. The result is:

$$F\_leastS\_noi = \begin{pmatrix} 2.3357e-05 & -9.064e-07 & -0.0023354 \\ 3.2026e-05 & 4.7671e-06 & -0.0035402 \\ -0.0072666 & -0.0021821 & 1 \end{pmatrix}$$



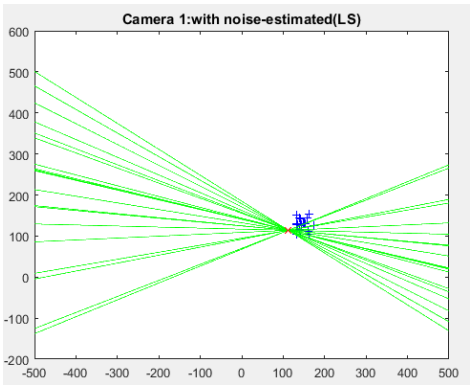


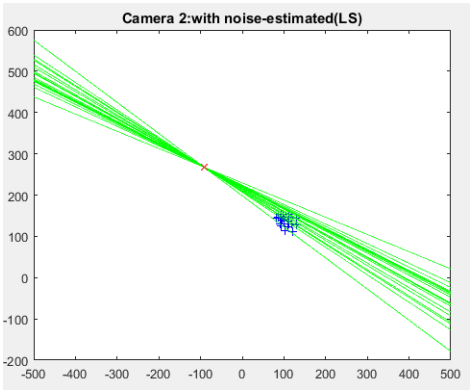
The epipolar lines do not cross at a single point and the epipole is not unique seen from the figure above. That is because the rank of fundamental matrix is

changed to 3 which contains the eigenvalue of noise. So, in order to draw the epipolar geometry reasonably, the fundamental matrix is reduced to rank-2 in the drawing process and epipole is recomputed by using fundamental matrix. The function below is used to draw the calibrated epipolar geometry.

```
draw_ep_calibrated (V_2d_c1_noi, V_2d_c2_noi, F_leastS_noi, 'with noise-estimated(LS)');
```

As we can see in the figure, the epipole is unique and the epipolar lines cross at a single point. But after zooming in we observed the noisy points do not lie on the epipolar lines.





2.12 Step 13: Increase the Gaussian noise of step 11 (now in the range [-2,+2] for the 95% of points) and repeat step 8-12

The fundamental matrix from [-2,+2] noisy points is:

$$F\_leastS\_noi = \begin{pmatrix} 1.758e-05 & 1.22e-06 & -0.0021752 \\ 2.9245e-05 & 1.203e-05 & -0.0044817 \\ -0.0059825 & -0.0028741 & 1 \end{pmatrix}$$

The epipolar geometry looks similar with the one added [-1,+1] noise but shows more noisy when increasing the range of noise.

3 Part 2

3.1 Step 14: Compute the fundamental matrix by using the 8-point 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

Based on the equation $U_n f = 0$, f lies on the nullspace of $U_n = UDV^T$. Hence, by using SVD, f corresponds to a multiple of the column of V that belongs to the unique singular value of D equal to 0.

```
%% Fundamental matrix computed by SVD method
U_2 = [U ones(pt_num,1)];
[u,d,v] = svd(U_2);
F_svd = reshape(v(:,end),[3,3])';
F_svd = F_svd/F_svd(end);
disp('Fundamental matrix from SVD method');
disp(F_svd);
```

The result is: $\begin{pmatrix} 3.2235e-05 & 7.2676e-05 & -0.0066178 \\ 2.4597e-05 & -2.3923e-05 & 0.012979 \\ -0.0095802 & -0.017343 & 1 \end{pmatrix}$

3.2 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

The fundamental matrix with noise in range [-1,+1]:

$$\begin{pmatrix} 1.9011e-05 & 4.8546e-06 & -0.0019121 \\ 3.2585e-05 & 8.2891e-06 & -0.0028018 \\ -0.0068961 & -0.0043955 & 1 \end{pmatrix}$$

The fundamental matrix with noise in range [-2,+2]:

$$\begin{pmatrix} 1.9914e-05 & -7.9934e-06 & -0.0019291 \\ 2.9001e-05 & 1.4508e-05 & -0.0058123 \\ -0.0059711 & -0.0012892 & 1 \end{pmatrix}$$

The epipolar geometry from SVD method shows similar result with the one from Least Square which can be seen in the matlab code. It is not displayed here in order not to fill up the page.

3.3 Step 16: Compare the epipolar geometry obtained in step 15 (using SVD) with the one obtained in steps 11 and 13 (using LS).

The distances from noisy points to their corresponding epipolar lines are computed and the maximum and mean values are extracted for comparing the accuracy of two methods when adding different ranges of noise. The comparison is shown below regarding to two cameras.

Table 1: Comparison of Camera 1

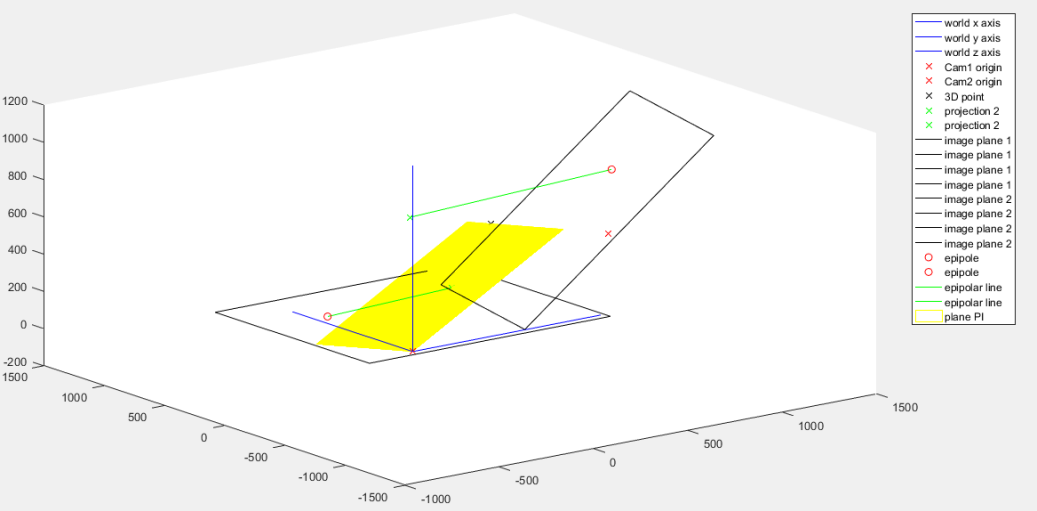
Camera 1	[-1,+1]		[-2,+2]	
	max	mean	max	mean
LS	10.513	4.3896	32.634	13.676
SVD	10.513	4.3893	32.637	13.676

Table 2: Comparison of Camera 2

Camera 2	[-1,+1]		[-2,+2]	
	max	mean	max	mean
LS	24.025	10.497	76.740	20.966
SVD	24.025	10.497	76.724	20.967

As we can see from the table, the difference between Least Square and SVD methods is very tiny which can be ignored. We can say the accuracy of results using these two methods is almost the same. However, the difference appears significantly when increasing the range of noise. The points deviate epipolar lines more as noise raising.

4 Part 3


world x axis
world y axis
world z axis
Cam1 origin
Cam2 origin
3D point
projection 2
projection 2
image plane 1
image plane 1
image plane 1
image plane 1
image plane 2
image plane 2
image plane 2
image plane 2
epipole
epipole
epipolar line
epipolar line
plane PI
1200
1000
800
600
400
200
0
-200
1500
1000
500
0
-500
-1000
-1500
-1000
-500
0
500
1000
1500