Multiple View Geometry

Lab3 – Epipolar Geometry and Stereo

Lab Sessions: 2 (4 hours)
Total Workload: 10 hours
Programming Language: Python or MATLAB

Nov 2025 (v1)

1. Objective

The objective of these two lab sessions is to gain further understanding on the concepts behind the simplest multicamera set-up – the stereo camera pair.

In this exercise you will:

1. Construct a Fundamental matrix analytically from two simulated cameras. Construct their projection matrices.
2. Define a set of 3D points and get their corresponding pairs of point projections on the images.
3. Compute the Fundamental matrix by using the 8-point method. Compare both fundamental matrices.
4. Draw the epipolar geometry in both image planes (points, epipoles and epipolar lines).
5. Increase the noise in 2D points and repeat the computation. Check the consistency of the epipolar geometry obtained.

2. Implementation

Part 1

Step 1.
Define camera 1 with the following parameters and set the world coordinate system to the coordinate system of camera 1 (i.e., Rotation = Identity and Translation = 0):

$au_1 = 100; \quad av_1 = 120; \gamma = 0; uo_1 = 128; \quad vo_1 = 128;$
$Image\ size: 256x256$ pixels

Step 2.
Define camera 2 with respect to camera 1 with the following parameters, where the angles follow the XYZ Euler angle convention:

$au_2 = 90, \quad av_2 = 110, \gamma = 0, \ uo_2 = 128, \quad vo_2 = 128,$
$a_x = 0.1rad, \quad b_y = \frac{\pi}{4}rad, \quad c_z = 0.2rad, \#XYZ\ EULER$
$t_x = -1000mm, \quad t_y = 190mm, \quad t_z = 230mm$
$Image\ size: 256x256$ pixels

Note that these rotation and translation parameters define the position of camera 2 with respect to camera 1. Since camera 1 is oriented in the world coordinate system, then these parameters will be for camera 1:

${}^{w}R_{c1} = I_3 \quad {}^{w}t_{c1} = [0\ 0\ 0]^T$

and camera 2:
${}^{w}R_{c2} = \mathrm{ROT}_x(a_x)\ \mathrm{ROT}_y(b_y)\ \mathrm{ROT}_z(c_z) \quad {}^{w}t_{c2} = [t_x\ t_y\ t_z\ ]^T$

Step 3.
Compute the intrinsic matrices of both cameras, and the rotation and translation between both cameras.
Compute also the two camera projection matrices P_1 and P_2. Remember that

$$\mathrm{K}=\begin{bmatrix}\alpha_u & \gamma & u_0\\ 0 & \alpha_v & v_0\\ 0 & 0 & 1\end{bmatrix}$$

$${}^{c}T_{w}=\begin{bmatrix} {}^{c}R_{w} & {}^{c}t_{w}\\ 0\ \ 0\ \ 0 & 1\end{bmatrix}$$

$$P = K \cdot \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix} \cdot {}^{c}T_{w}$$

Note that

$${}^{c}T_{w}={}^{w}T_{c}{}^{-1}=\begin{bmatrix} {}^{w}R_{c} & {}^{w}t_{c}\\ 0\ \ 0\ \ 0 & 1\end{bmatrix}^{-1}=\begin{bmatrix} {}^{w}R_{c}{}^{T} & -{}^{w}R_{c}{}^{T}\cdot {}^{w}t_{c}\\ 0\ \ 0\ \ 0 & 1\end{bmatrix}$$

Step 4.
Get the Fundamental matrix analytically as the product of matrices defined in step 3, as follows. Remember to represent the translation vector in the form of its skew-symmetric matrix (see lecture notes).

$${}^{2}F_{1} = {K_2}^{-T} \cdot \ {}^{1}R_{2}{}^{T} \cdot \left[{}^{1}t_{2}\right]_{X} \cdot {K_1}^{-1}$$

The above formulation is slightly different from the one presented in the lectures notes, but it is equivalent. You can check this by computing the transpose of ${}^{2}F_{1}$, and comparing it to the lecture notes.

The ${}^{2}F_{1}$ matrix relates the points in the two images as

$$[u_2\ v_2\ 1]\ {}^{2}F_{1}\begin{bmatrix}u_1\\ v_1\\ 1\end{bmatrix}=0$$

Step 5.
Define the following set of object points with respect to the world coordinate system (same as the camera 1 coordinate system)

```
V = np.array([[100, -400, 2000],
              [300, -400, 3000],
              [500, -400, 4000],
              [700, -400, 2000],
```

```
[900, -400, 3000],
[100, -40, 4000],
[300, -40, 2000],
[500, -40, 3000],
[700, -40, 4000],
[900, -40, 2000],
[100, 40, 3000],
[300, 40, 4000],
[500, 40, 2000],
[700, 40, 3000],
[900, 40, 4000],
[100, 400, 2000],
[300, 400, 3000],
[500, 400, 4000],
[700, 400, 2000],
[900, 400, 3000]])
```

Step 6.
Compute the pairs of image points in both image planes by using the camera projection matrices of step 3.
Open two windows, which will be used as both image planes, and draw the 2D points obtained in step 6.

Step 7.
Compute the fundamental matrix with the 8-point method and SVD using of the 2D points obtained in step 6, *without imposing the rank constraint*.

Step 8.
Compare the matrix of step 7 with the one obtained in step 4. To obtain a value that is representative of the difference, you can use the sum of absolute differences or any other indicator that you consider adequate.

Step 9.
Draw in the windows of step 6 all the epipolar geometry, i.e., epipoles and epipolar lines by using the matrix obtained in step 7. You can represent the location of the epipoles with a cross ('+');
Enlarge the windows if necessary to view the epipoles properly.

[Hint] To draw an epipolar line in image 2 corresponding to a point $[u_1\ v_1\ 1]$ in image 1, you can compute

$$l'_m = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} = F \cdot \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}$$

and note that all points of the epipolar line must satisfy

$$[u_2, v_2, 1][l_1, l_2, l_3]^T = 0.$$

Re-write this equation in the form of and equation of a line $v = mu + d$ so that you obtain *m* and *d* from the components of the epipolar line. Hence, fixing u in both boundaries of the image plane, you can obtain the v component and draw the epipolar line by using the Plot function.

You can compute the epipoles by projecting the focal point of each camera to the image plane of the other. Moreover, the epipoles can be computed by intersecting two or more epipolar lines as all the epipolar lines cross at the epipole *in the absence of noise*.

Step 10.
Add zero-mean Gaussian noise to the 2D points such that 95% of that noise will be the range $[-1, +1]$ pixels.

Step 11.
Repeat step 7 up to 9 with the noisy 2D points. Compare the epipolar geometry obtained.

Questions 1

- Are points on the epipolar lines?
- Are the epipolar lines all converging on a single point?
- What condition should the F matrix fulfill so that all epipolar lines cross at the same point?

Impose that condition and

- Verify if all epipolar lines now cross at single same point on each image.
- Investigate how the location of both epipoles can be computed directly from the F matrix, using the SVD, and describe the procedure.

Step 12.
Increase the Gaussian noise of step 10 (now in the range $[-2, +2]$ for 95% of points) and repeat steps 7-11.

Step 13.
Implement the coordinate normalization method described in the classes slides. The transformations (T and T' in the slides) are homografies that modify the original projected points in each image so that they become centered at the origin and have standard deviation equal to 1.

Repeat steps 7-11 with the Gaussian noise in the range $[-2, +2]$ for 95% of points.

Check the condition number of U_n of the 8-point algorithm that you get with and without coordinate normalization. You can compute the condition number with the command *linalg.cond* of *numpy*.

Questions 2

- Are there improvements in terms of the sum of absolute differences (Step 8) and on the location of the epipolar lines (Step 9)?

Step 14.
Structure your code so that you can repeat step 13 for 100 times. In each of these iterations, save the location of the epipoles computed with and without coordinate normalization.
Create a scatter plot of the location of all the epipoles computed with and without coordinate normalization.
Comment on the results.

NOTE :

For this report write a short introduction to the topic of epipolar geometry and the fundamental matrix, where you explain, ***in your own words***[1], the fundamental ideas behind this topic.

This introduction should be placed at the beginning of your report and should take up to a full page.

3. Deliverable

This Lab exercise is to be done in groups of 2 students.

Your deliverable will consist of one report in PDF format, and one or more .py files with your code. You will have to upload these files as a single ZIP file to Moodle before the deadline.

The deadline is one week after the last lab session of this exercise.

The PDF should include the names of both members of the group and:

- **Should detail how you have solved every step**
- Include the .py code that solve each step
- Include the obtained results per step
- **Discuss, when necessary, the results obtained**.

The Python code should easy to understand and well commented. There is no need for extensive comments, but it should be clearly stated what each part of the code is doing.

Your code should run as is. You are encouraged to create your own functions to better structure the code. The main execution script should be clearly identified.

[1] Do not use an AI tool, which is easy to detect.