Exercise 1
2D-3D geometry in homogeneous coordinates and camera projection

Goals

1. Practise with homogeneous coordinates to code 2D and 3D geometrical entities
2. Practise with perspective projection of basic geometrical entities.

Data

1. Two monocular RGB images from a sequence in the Scannet dataset, with partial common field of view:

image 1

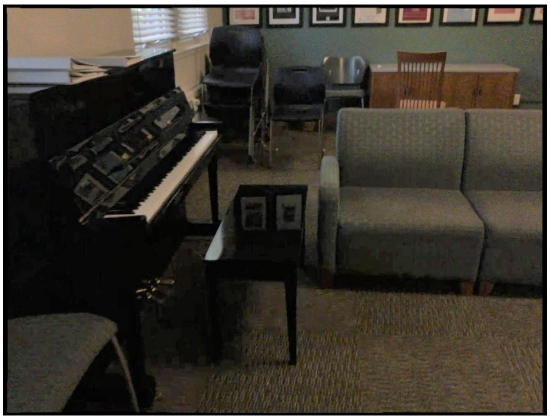
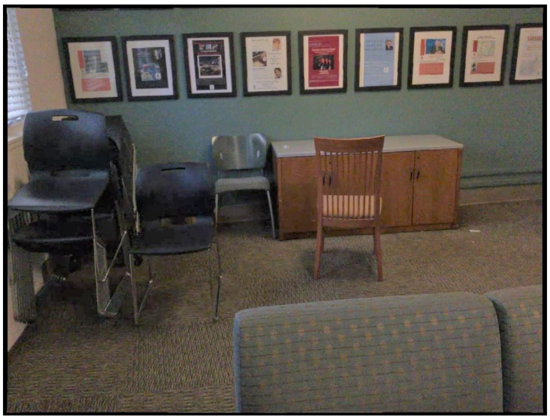


image 2



2. Each camera pose R_wc1, t_wc1, R_wc1, t_wc1, and calibration parameters α_x , α_y , x_0 , y_0

Calibration parameters:	
α_x = 1165.723022 px α_y = 1165.738037 px x_0 = 649.094971 px y_0 = 484.765015 px	R_w_c1 = [-0.107841 0.3888 -0.91495 0.99396 0.23439 -0.1072 -0.2024 -0.921 -0.3.891] t_w_c1 = [6.1377 1.3981 1.4762]' m R_w_c2 = [0.107366 0.306900 -0.945666 0.993780 -0.061402 0.092902 -0.029555 -0.949759 -0.311584] t_w_c2 = [4.4462 1.41698 1.5726]' m

3. 3D coordinates of some selected points in the scene (in meters).
   A: [3.44,0.80,0.825]
   B: [4.20,0.80,0.815]
   C: [4.20,0.60,0.820]
   D: [3.55,0.60,0.820]
   E: [-0.01, 2.6, 1.21]
4. Data can be downloaded from:
   https://drive.google.com/drive/folders/1PCWvw5IGSvNtogwUHtrbNYqgQezzs6PY?usp=sharing

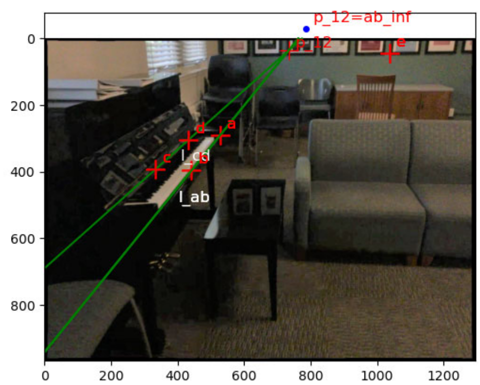
Software

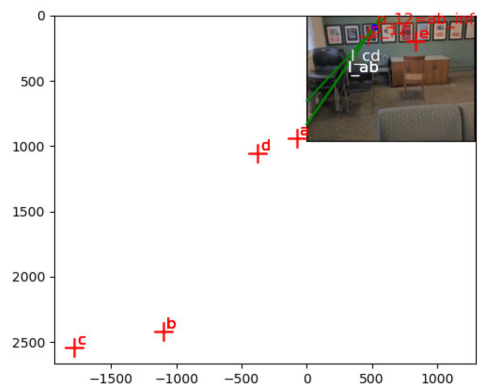
Recommended libraries (Python3): Numpy, OpenCV and matplotlib

Exercises

For both images:

1. Compute the projection matrices P1 and P2.
2. Compute and plot all the projections of the 3D points A,B,C,D,E in both images.
3. Compute and plot in each image the line l_ab defined by a, and b (projections on A and B).
4. Compute and plot in the image the line l_cd, defined by c and d (projections on C and D) in the image.
5. Computep_12 the intersection point of l_ab and l_cd
6. Compute the 3D infinite point corresponding to the 3D direction defined by points A and B, AB_inf.
7. Project the point AB_inf with matrix P to obtain the corresponding vanishing point ab_inf.
8. Next two figures display the solution to the previous exercises





9. Compute the equation of the 3D plane π defined by the points A,B,C,D.
   Solution: [-0.0056 -0.0017 -0.7615 0.6482]

10. Compute the distance of each of the 3D points A,B,C,D to the plane π
    Solution:
    d_A = -0.002 m
    d_B = 0.002 m
    d_C = -0.002 m
    d_D = 0.003 m
    d_E = -0.365 m

Exercise 2
Two view geometry. Homography and fundamental matrix

Goals

1. Compute F and H from assuming scene and two camera geometry.
2. Usage of F and H to guide matching between two views.
3. Compute F and H from noise-free image matches.

Data

1. Two monocular images from EuRoC dataset, with common field of view and a significant parallax.

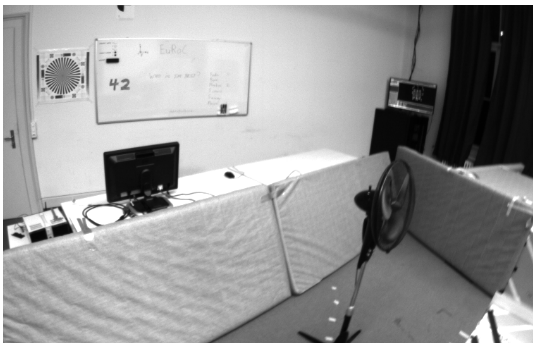



image 1, original, distorted image   Image 2, original, distorted image.


image 1, undistorted image   Image 2, undistorted image

2. Intrinsic parameters of the camera after undistortion are (both images had been captured with the same camera). α_x = 458.654 px, α_y = 457.296 px, x_0 = 367.215 px, y_0 = 248.375 px
3. Extrinsic parameters for both cameras:

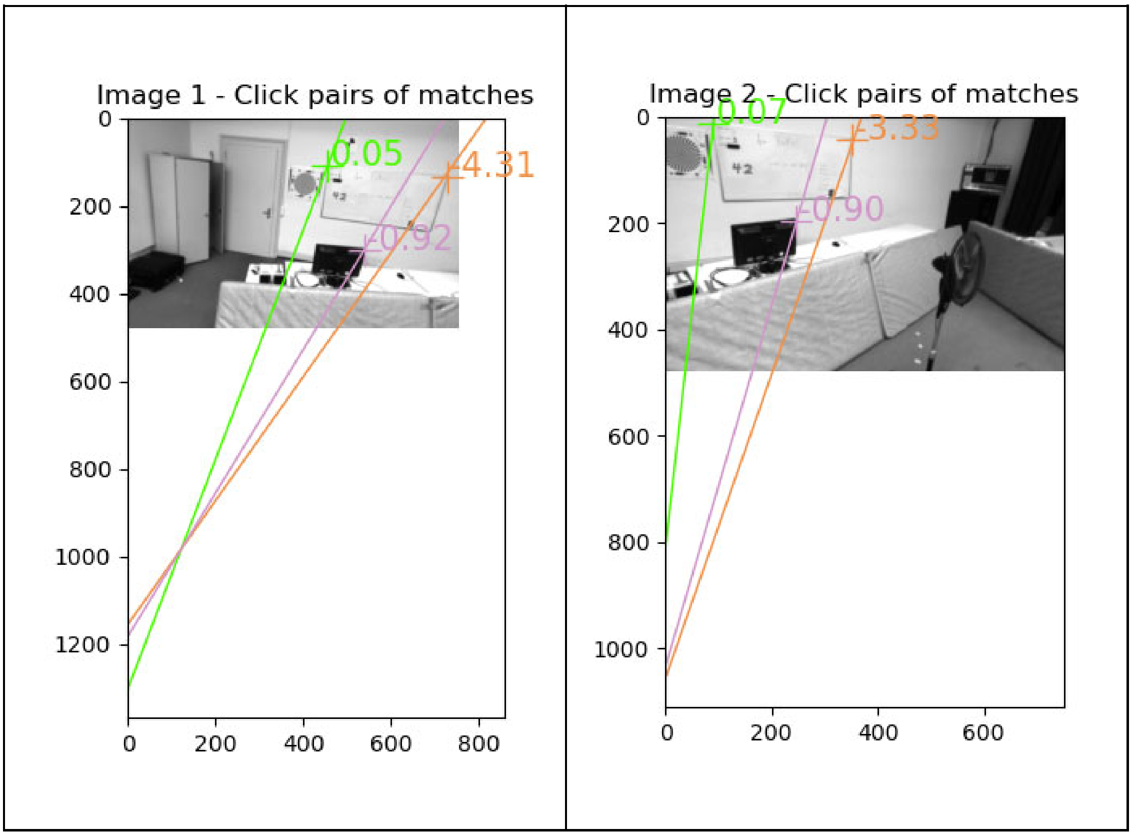
R_w_c1 = [-0.9762 -0.1841 0.1143
-0.1588 0.2488 -0.9554
0.1475 -0.9509 -0.2721]
t_w_c1 = [0.5091 0.3592 1.8899] m

R_w_c2 = [0.7319   0.2086 0.6487
-0.3265   -0.7282 0.6025
0.5981   -0.6528 -0.4649]
t_w_c2 = [0.6822 0.1198 0.9345] m

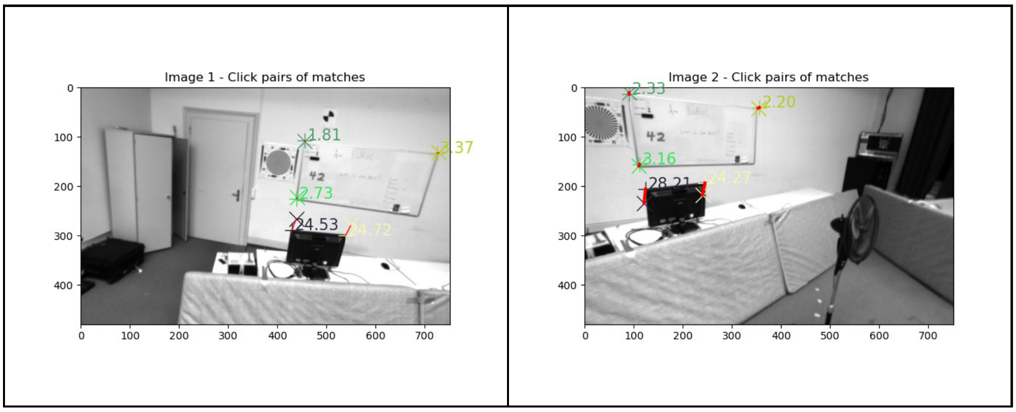
4. Whiteboard plane equation in the C1 reference is
0.243360266797104X -0.738377227649819Y -0.440791457369051Z+ 0.448639879349884=0

Exercises

1. Compute the fundamental matrix F from the camera poses and camera calibration parameters.
2. Verify the epipolar geometry:
   a. Click a point on image1, plot the corresponding epipolar line in the image2, verify that the corresponding point is close to the epipolar line.
   b. Click a point on image2, plot the corresponding in the image1, verify that the corresponding point is close to the epipolar line.
   c. Click two corresponding points in image1 and image2. Plot the corresponding epipolar lines. Plot in the images the distance between the point and the corresponding epipolar line (in pixels)



3. Compute the homography H defined by the whiteboard plane and the two images.
4. Verify the homography:
    a. Click a point on image1, plot the transferred point according to the homography in image2.
    b. Click a point on image2, plot the transferred point according to the homography in image1.
    c. Click two corresponding points in image1 and image2. Plot the transferred points. Plot in the images the distance between the corresponding point and the transferred points, in pixels.



5. Compute the homography from 4 or more correspondences you select by clicking. Verify the computed homography.
6. Compute the F from 8 or more correspondences you select by clicking. Verify the computed epipolar geometry.

Exercise 3
3D point triangulation

Goals

1. Triangulate 3D points from matches in two localized and calibrated cameras.

Data

1. Two monocular images from EuRoC dataset, with common field of view and a significant parallax.

image 1, undistorted image Image 2, undistorted image

2. Intrinsic parameters of the camera (both images had been captured with the same camera). α_x = 458.654 px, α_y = 457.296 px, x_0 = 367.215 px, y_0 = 248.375 px
3. Extrinsic parameters for both cameras, pose of the two cameras:

```
T_WC1 = [[-0.6118  -0.275    0.7417  1.4202]
         [-0.7909  0.2285  -0.5677  2.4423]
         [-0.0134  -0.9339 -0.3573  1.2341]
         [ 0.        0.        0.        1.      ]]

T_WC2 = [[-0.512  -0.2779  0.8128  0.5236]
         [-0.8586  0.194  -0.4745  1.9537]
         [-0.0258 -0.9408 -0.338   1.2876]
         [ 0.       0.        0.       1.   ]]
```

4. The point matches of each image are given in files (1 line per point, two columns defining the point coordinates in pixels):
   a. PointsImage1.txt
   b. PointsImage2.txt

Exercises

1. Compute the projection matrices for C1 and C2.
2. Triangulate the 3D pose, in the W reference, for each of the matched points.

Exercise 4
Structure from Motion

Goals

1. Compute the camera motion from the fundamental matrix.
2. Compute the camera motion and the scene structure from a fundamental matrix and point matches.

Data

1. Two monocular images from EuRoC dataset, with common field of view and a significant parallax.

image 1, undistorted image

Image 2, undistorted image

2. Intrinsic parameters of the camera (both images had been captured with the same camera). α_x = 458.654 px, α_y = 457.296 px, x_0 = 367.215 px, y_0 = 248.375 px
3. Extrinsic parameters for both cameras, pose of the two cameras:

```
T_WC1 = [[-0.6118  -0.275     0.7417  1.4202]
         [-0.7909  0.2285  -0.5677  2.4423]
         [-0.0134  -0.9339 -0.3573  1.2341]
         [ 0.          0.          0.         1.        ]]

T_WC2 = [[-0.512  -0.2779  0.8128  0.5236]
         [-0.8586  0.194  -0.4745  1.9537]
         [-0.0258 -0.9408 -0.338   1.2876]
         [ 0.        0.          0.        1.    ]]
```

4. The fundamental matrix that gives the epipolar line in image 2 for a point in image 1.

```
F=[[-0.          0.          0.0012]
   [-0.          0.         -0.0205]
   [-0.0008  0.0133   1.]]
```

5. The point matches of each image are given in files (1 line per point, two columns defining the point coordinates in pixels):
   a. PointsImage1.txt
   b. PointsImage2.txt

Exercises

1. From the fundamental matrix F, compute the 4 solutions for the up to scale camera motion.
2. Select which of the 4 previous motions corresponds to the actual camera motion.
3. Compute the scene structure.

Exercise 5
Dense motion estimation

Goal

Compute motion in an image pixel

Data

Two images from the MPI-Sintel dataset

I_0	I_1
	

You can download them from:
https://drive.google.com/drive/folders/1rijGbrqh2LbkbCEDozYh-NKQgVbRkvnT?usp=sharing

If you want to take a look to the whole MPI-Sintel dataset you can reach it at:
https://drive.google.com/drive/folders/1dcoMpYSnLoAqGav2QyIF77EtGpN7_y7Z?usp=sharing

Exercises

For the pixel [584 92] in I_0:

1. Compute the motion with pixel accuracy by SSD by an exhaustive search in a 101x101 window around [584 92] in I_1. Use a 11x11 centered window.
2. Compute the motion with pixel accuracy by SSD by an exhaustive search in a the whole I_1. Select the window size.
3. Compute the motion with pixel accuracy by NCC by an exhaustive search in all the pixels in I_1
4. Compute the motion with pixel accuracy by NCC by searching in a 101x101 window around [584 92].

Ground truth for the motion is [-2.24 -0.56] pixels

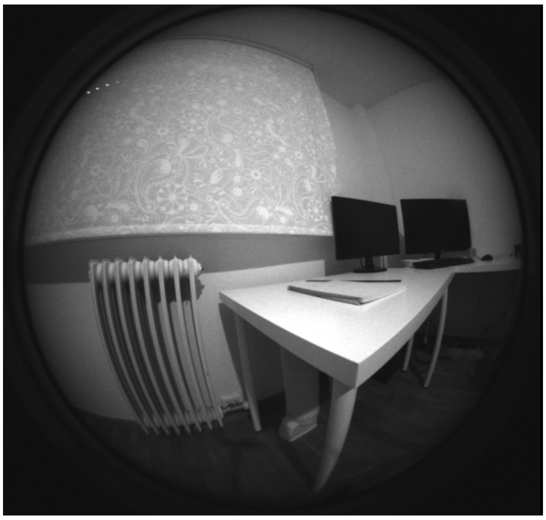
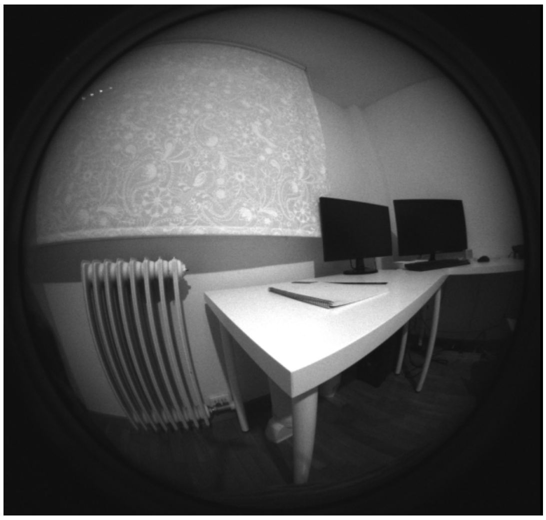
Exercise 6
Omnidirectional vision

Goals

1. Compute and understand the non-linear Kannala-Brandt projection model
2. Compute and understand the non-linear Kannala-Brandt unprojection model.
3. Triangulate 3D points from a fish-eye stereo using planes.

Provided data

1. Two fisheye images from the stereo system Realsense T265 tracking camera.

2. The intrinsic calibration of both cameras: `K_1.txt`, `K_2.txt`, `D1_k_array.txt`, `D2_k_array.txt;`
K matrix for the focal length and principal point and D for the polynomial coefficients such that [k1,k2,k3,k4] = D[0:4] and $d(\theta) = \theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9$.
3. The example points **X** are projected on the pixels **u** in fisheye 1 (left).

$$\mathbf{X}_1 = \begin{pmatrix} 3 \\ 2 \\ 10 \\ 1 \end{pmatrix}, \quad \mathbf{X}_2 = \begin{pmatrix} -5 \\ 6 \\ 7 \\ 1 \end{pmatrix}, \quad \mathbf{X}_3 = \begin{pmatrix} 1 \\ 5 \\ 14 \\ 1 \end{pmatrix}$$

$$\mathbf{u}_1 = \begin{pmatrix} 503.387 \\ 450.1594 \\ 1 \end{pmatrix}, \quad \mathbf{u}_2 = \begin{pmatrix} 267.9465 \\ 580.4671 \\ 1 \end{pmatrix}, \quad \mathbf{u}_3 = \begin{pmatrix} 441.0609 \\ 493.0671 \\ 1 \end{pmatrix}$$

4. The extrinsic calibration of the stereo system: i.e. the rotation and the translation between cameras. `T_12.txt`, `T_wc1.txt`, `T_wc2.txt`
5. The point matches of each image are given in files (1 column per point, 3 rows defining the point coordinates in pixels): `x1.txt, x2.txt`

Exercises

1. Compute the projection model of Kannala-Brandt in the given points.
2. Compute the unprojection model of Kannala-Brandt for the given image projection
3. Compute the pair of planes corresponding to each direction vector.
4. Triangulate the rays using 4 planes for each 3D point.