Assignment 5: Epipolar geometry and triangulation

Machine perception

2021/2022

Create a folder `assignment5` that you will use during this assignment. Unpack the content of the `assignment5.zip` that you can download from the course webpage to the folder. Save the solutions for the assignments as Python scripts to `assignment5` folder. In order to complete the assignment you have to present these files to the teaching assistant. Some assignments contain questions that require sketching, writing or manual calculation. Write these answers down and bring them to the presentation as well. The tasks that are marked with ★ are optional. Without completing them you can get at most 75 points for the assignment. The maximum total amount of points for each assignment is 100. Each optional task has the amount of additional points written next to it.

At the defense, the obligatory tasks **must** be implemented correctly and completely. If your implementation is incomplete, you will not be able to defend the assignment.

Introduction

In this assignment we will review several parts of the epipolar geometry [1] (chapter 10.1) and robust estimation [1] (page 346).

Exercise 1: Disparity

In this assignment we will focus on calculating disparity from a two-camera system. Our analysis will be based on a simplified stereo system, where two identical cameras are aligned with parallel optical axes and their image planes (CCD sensors) lie on the same plane (Image 1a).

In Image 1b we can observe that we can write the following relations using similar triangles:

$$\frac{x_1}{f} = \frac{p_x}{p_z} \quad , \quad \frac{-x_2}{f} = \frac{T - p_x}{p_z}. \tag{1}$$

(a) Using the equations (1) derive the expression for *disparity* which is defined as $d = x_1 - x_2$. What is the relation between the distance of the object (point $\mathbf{p}$ in Image 1) to camera and the disparity d? What happens to disparity when the object is close to the cameras and what when it is far away?

(b) Write a script that computes the disparity for a range of values of p_z. Plot the values to a figure and set the appropriate units to axes. Use the following parameters of the system: focal length is $f = 2.5mm$ and stereo system baseline is $T = 12cm$.

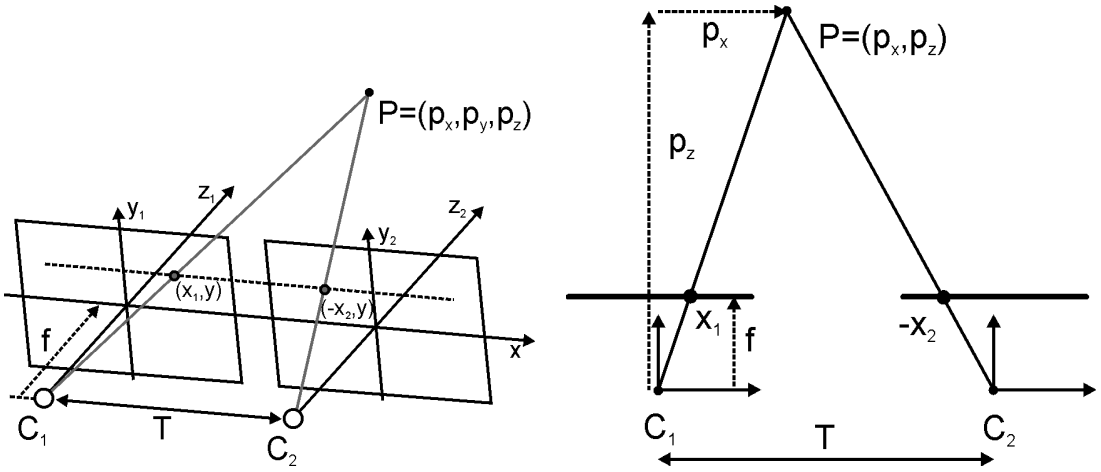


Figure 1: Left image shows the simplest stereo setup with two parallel cameras. Right image shows geometric relations when mapping of a point $\mathbf{p}$ in 3D space to axis x in image planes of the cameras.

(c) In order to get a better grasp on the idea of distance and disparity, you will calculate the numbers for a specific case. We will take the parameters from a specification of a commercial stereo camera *Bumblebee2* manufactured by the company PointGray: $f = 2.5mm$, $T = 12cm$, whose image sensor has a resolution of 648×488 pixels that are square and the width of an pixel is $7.4\mu m$. We assume that there is no empty space between pixels and that both cameras are completely parallel and equal. Lets say that we use this system to observe an (point) object that is detected at pixel 550 in x axis in the left camera and at the pixel 300 in the right camera. How far is the object (in meters) in this case? How far is the object if the object is detected at pixel 540 in the right camera? Solve this task analytically and bring your solution to the presentation of the exercise.

(d) ★ (10 points) Write a script that calculates the disparity for an image pair. Use the images in the directory `disparity`. Since the images were pre-processed we can limit the search for the most similar pixel to the same row in the other image. Since just the image intensity carries too little information, we will instead compare small image patches. A simple way of finding a matching patch is to use normalized cross correlation. NCC for matrices X and Y of equal size is defined as

$$NCC(\mathbf{X}, \mathbf{Y}) = \frac{1}{N} \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{(\sqrt{\frac{1}{N}\sum (x_i - \bar{x})^2})(\sqrt{\frac{1}{N}\sum (y_i - \bar{y})^2})}. \quad (2)$$

where x_i denotes a specific cell in matrix $\mathbf{X}$ and y_i a specific cell in matrix $\mathbf{Y}$. A patch from the second image is considered a match if it has the highest NCC value. The difference in x axis between the point from the first image and the matching point from the second image is the disparity estimate in terms of pixels[1]. The disparity search is performed in a single direction and is usually limited with a maximum value.

[1]Converting this estimate to distance requires information about the camera, which we do not have for the given image pairs.

Perform the search for the most similar patch in both images, so you obtain two disparity matrices. You can then merge them in some way you choose. Smooth the results using median filter and choose an appropriate search window to obtain good results.

Note: You cannot merge the results element-wise. The images were taken from different viewpoints and the results would be incorrect.

Question: Is the disparity estimated well for all pixels? If not, what are the characteristics of pixels with more consistent disparity estimation?

(e) ★ (5 points) Improve the noisy disparity estimation. You can implement a symmetric matching technique where the left image is compared to the right image and vice versa. The result can then be merged in some way that you choose. Smooth the results using median filter and choose an appropriate search window to obtain good results. Very noisy results will not be graded.

Exercise 2: Fundamental matrix, epipoles, epipolar lines

In the previous exercise we were dealing with a special stereo setup where the image planes of two cameras were aligned. In that case the projection of a 3D point has the same y coordinate in both cameras and the difference in x coordinate can tell us the distance of the point to the camera system. Such a setup ensures that the *epipolar lines in the cameras are parallel to the lines in the sensor*. This simplifies the search for correspondences (i.e. the projection of the same 3D point to both cameras). Generally the epipolar lines are not aligned with rows and in order to establish the relation between two image planes, the *fundamental matrix* needs to be computed.

In this exercise you will implement a simple version of a *eight-point algorithm* [1] that can be used to estimate a fundamental matrix between two cameras. We will first revisit the theory that will be used for the task. If we assume a list of perfect correspondence pairs of points in left $\mathbf{x} = [u, v, 1]^T$ and right $\mathbf{x}' = [u', v', 1]^T$ image in homogeneous coordinates. The fundamental matrix rule states that each correspondence be a valid solution for the following equation:

$$\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0 \quad ; \quad \mathbf{F} = \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix}, \tag{3}$$

where $\mathbf{F}$ denotes a fundamental matrix. Similarly that we have done for the estimation of homography in the previous exercise, we can write a relation between a single pair of

correspondence points as

$$\begin{bmatrix} uu' & uv' & u & vu' & vv' & v & u' & v' & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = 0. \tag{4}$$

If we combine $N \geq 8$ of these equations in a matrix $\mathbf{A}$ we get a system of linear equations:

$$\mathbf{Af} = \mathbf{0}$$

$$\begin{bmatrix} u_1u'_1 & u_1v'_1 & u_1 & v_1u'_1 & v_1v'_1 & v_1 & u'_1 & v'_1 & 1 \\ u_2u'_2 & u_2v'_2 & u_2 & v_2u'_2 & v_2v'_2 & v_2 & u'_2 & v'_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_Nu'_N & u_Nv'_N & u_N & v_Nu'_N & v_Nv'_N & v_N & u'_N & v'_N & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ \vdots \\ F_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{5}$$

We can solve the linear system above in a least-squares way by using the singular value decomposition method (SVD). Matrix $\mathbf{A}$ is decomposed to $\mathbf{A} = \mathbf{UDV}^T$. The solution according to least squares corresponds to the eigenvector $\mathbf{v}_n$ with the lowers eigenvalue, e.t. the last column of the matrix[2] $\mathbf{V}$.

Recall that the *epipole* of a camera is a point where all the epipolar lines (for that camera) intersect. This requires the rank of the matrix $\mathbf{F}$ to be 2, however, this is usually not true when dealing with noisy data – the fundamental matrix estimated using the approach above will not have a rank 2. In practice this means that the epipolar lines will not intersect in a single point but rather in a small neighborhood of such a point. To stabilize the system we have to subsequently limit the rank of the fundamental matrix. This can be done by performing a SVD decomposition to $\mathbf{F} = \mathbf{UDV}^T$, set the lowest eigenvalue (D_{33} in matrix $\mathbf{D}$) to 0, and then reconstruct back the corrected matrix $\mathbf{F}$ by multiplying $\mathbf{UDV}^T$. A newly created fundamental matrix will satisfy the rank condition rank $= 2$ and can be used to compute two epipoles (one for each camera)

$$\mathbf{Fe}_1 = 0 \quad \text{in} \quad \mathbf{F}^T\mathbf{e}_2 = 0, \tag{6}$$

by decomposing the matrix $\mathbf{F}$ again and computing the *left* and *right*[3] eigenvector of the matrix $\mathbf{F}$,

$$\mathbf{e}_1 = \begin{bmatrix} V_{13} & V_{23} & V_{33} \end{bmatrix} / V_{33} \ , \ \mathbf{e}_1 = \begin{bmatrix} U_{13} & U_{23} & U_{33} \end{bmatrix} / U_{33}, \tag{7}$$

which are then used to obtain both epipoles in homogeneous coordinates.

A simple eight-point algorithm for estimation of a fundamental matrix and the epipoles can be summarized as:

- Construct the matrix $\mathbf{A}$ as in equation (5)

[2]This is a solution in column form as seen in equation (4) that has to be reshaped to the form in equation (3).

[3]Attention: the terms left and right eigenvector are mathematical terms and do not hold any relation to the left and right camera in our system.

- Decompose the matrix using SVD $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$, and transform the last eigenvector $\mathbf{v}_9 = \mathbf{V}(:, 9)$ in a 3×3 fundamental matrix $\mathbf{F}_t$
- Decompose $\mathbf{F}_t = \mathbf{U}\mathbf{D}\mathbf{V}^T$ and set the lowest eigenvalue to 0, reconstruct $\mathbf{F} = \mathbf{U}\mathbf{D}\mathbf{V}^T$.
- Decompose $\mathbf{F}_t = \mathbf{U}\mathbf{D}\mathbf{V}^T$ again, compute both epipoles as `e1 = V(:, 3) / V(3, 3)` and `e2 = U(:,3)/U(3,3)`.

Once we have the fundamental matrix, we can take any point $\mathbf{x}_1$ in the first image plane and determine an epipolar line for that point in the second image plane $\mathbf{l}_2 = \mathbf{F}\mathbf{x}_1$. Likewise, we can take point $\mathbf{x}_2$ in the second image plane and find an epipolar line in the first image plane $\mathbf{l}_1 = \mathbf{F}^T\mathbf{x}_2$. Recall that a line in a projective geometry is defined as a single 3D vector $\mathbf{l}_1 = [a, b, c]$. A line can be written using a classical Euclidean formula as

$$ax + by + c = 0, \tag{8}$$

and used to insert the coordinates $\mathbf{x}_1 = [X, Y, W]$. This gives us

$$aX + bY + cW = 0 \tag{9}$$
$$\mathbf{l}_1^T\mathbf{x}_1 = \mathbf{x}_1^T\mathbf{l}_1 = 0. \tag{10}$$

The interpretation of the parameters is as follows: $-a/b$ is the angle of the line, $-c/a$ and $-c/b$ are the intersections with axes x and y.

(a) Solve the following task analytically. We are given a system of two cameras and a fundamental matrix that connects the left camera to the right one

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & 0.002 \\ 0 & 0 & -0.012 \\ -0.001 & 0.011 & -0.085 \end{bmatrix}. \tag{11}$$

Question: Compute the Euclidean equation of the epipolar line in the right camera that corresponds to the point at row = 120 and column = 300 in the left camera. Take into account that the point has to be first written in homogeneous coordinates, i.e. $\mathbf{x} = [\text{column}, \text{row}, 1]^T$. Also compute the epipolar line for another point at row = 170 and col = 300 in the left camera.

Question: Compute the intersection of these two lines. What is the name of the point of intersection?

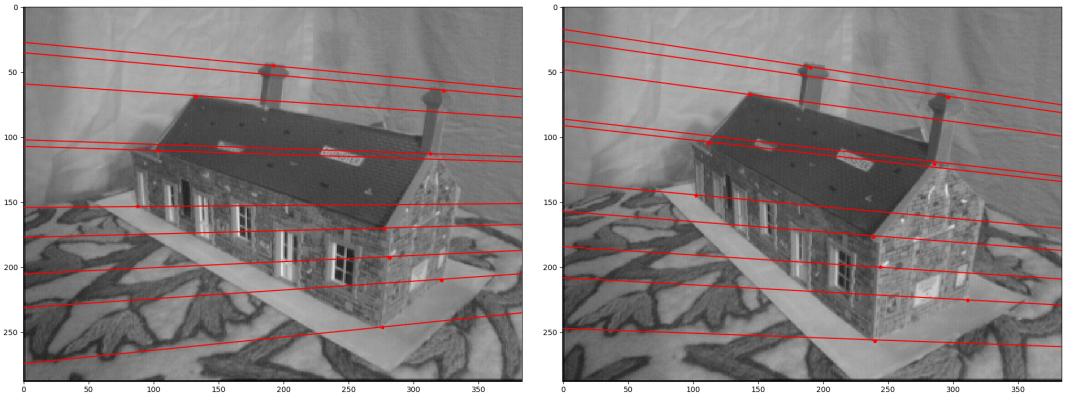
Estimating a fundamental matrix

(a) Implement a function `fundamental_matrix` that is given a set of (at least) eight pairs of points from two images and computes the fundamental matrix using the eight-point algorithm.

As the eight-point algorithm can be numerically unstable, it is usually not executed directly on given pairs of points. Instead, the input is first normalized by centering them to their centroid and scaling their positions so that the average distance to the centroid is $\sqrt{2}$. For a set of points $\mathbf{x}_1$ we achieve this by multiplying them with a transformation matrix $\mathbf{T}_1$. To achieve this, you can use the function `normalize_points` from the supplementary material.

Extend the function `fundamental_matrix` so that it first *normalizes* the input point-set of the left camera (we get transformed points and the transformation matrix $\mathbf{T}_1$) and then transform the input point set of the right camera (we get the transformed points and the transformation matrix $\mathbf{T}_2$). Using the transformed points the algorithm computes fundamental matrix $\hat{\mathbf{F}}$, then transforms it into the original space using both transformation matrices $\mathbf{F} = \mathbf{T}_2^T \hat{\mathbf{F}} \mathbf{T}_1$.

(b) Test your function for fundamental matrix estimation using the ten correspondence pairs that you load from the file `house_points.txt`. Compute the fundamental matrix $\mathbf{F}$ and for each point in each image calculate the corresponding epipolar line in the other image. You can draw the epipolar lines using `draw_epiline` from the supplementary material. According to epipolar geometry the corresponding epipolar line should pass through the point. As a testing reference the correct fundamental matrix is included in the supplementary material in file `house_fundamental.txt`.



(c) We use the *reprojection error* as a quantitative measure of the quality of the estimated fundamental matrix.

Write a function `reprojection_error` that calculates the reprojection error of a fundamental matrix F given two matching points. For each point, the function should calculate the corresponding epipolar line from the point's match in the other image, then calculate the perpendicular distance between the point and the line using the equation:

$$distance(ax + by + c = 0, (x_0, y_0)) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}, \tag{12}$$

where a, b and c are the parameters of the epipolar line. Finally, the function should return the average of the two distances.

Write a script that performs two tests: (1) compute the reprojection error for points $p_1 = [85, 233]^T$ in the left image-plane and $p_2 = [67, 219]^T$ in right image-plane using the fundamental matrix (the error should be approximately 0.15 pixels). (2) Load the points from the file `house_points.txt` and compute the average of symmetric reprojection errors for all pairs of points. If your calculation is correct, the average error should be approximately 0.33 pixels.

RANSAC algorithm

In practice automatic correspondence detection algorithms very rarely find perfect matches. In most cases the locations of these points contain some noise, or in some cases several correspondences might also be completely wrong. These correspondences are called *outliers*[4]. A good meta-algorithm for such cases is RANdom SAmple Consensus (RANSAC). The algorithm can be applied to a wide variety of problems. A version that can be used for robust estimation of fundamental matrix is structured as follows:

- Randomly select a minimal set of point matches (correspondences) that are required to estimate a model (in case of fundamental matrix this number is 8).
- Estimate a fundamental matrix using the selected sub-set.
- Determine the *inliers* for the estimated fundamental matrix (i.e. matched pairs that have a reprojection error lower than a given threshold).
- If the percentage of inliers is big enough ($\geq m$) use the entire inlier subset to estimate a new fundamental matrix (using mean squares method and SVD).
- Otherwise repeat the entire procedure for k iterations.

The value of parameter k is defined by the properties of our data that are set by knowing the estimation problem that we are trying to solve. E.g., suppose that we constantly encounter at least w percent of inliers (correctly matched point pairs). The number of pairs required to estimate a fundamental matrix $\mathbf{F}$ is at least $n = 8$. We can now compute the probability of successful estimation of $\mathbf{F}$, i.e. the probability that all n selected points will be inliers as w^n. The probability that this will not be true is $1 - w^n$. The probability that we do not manage to select a clean set of inliers in k repetitions is $p_{\text{fail}} = (1 - w^n)^k$. In practice we therefore select k high enough to reduce the probability of failure p_{fail} to an acceptable level.

You will now implement all the required steps of the RANSAC algorithm.

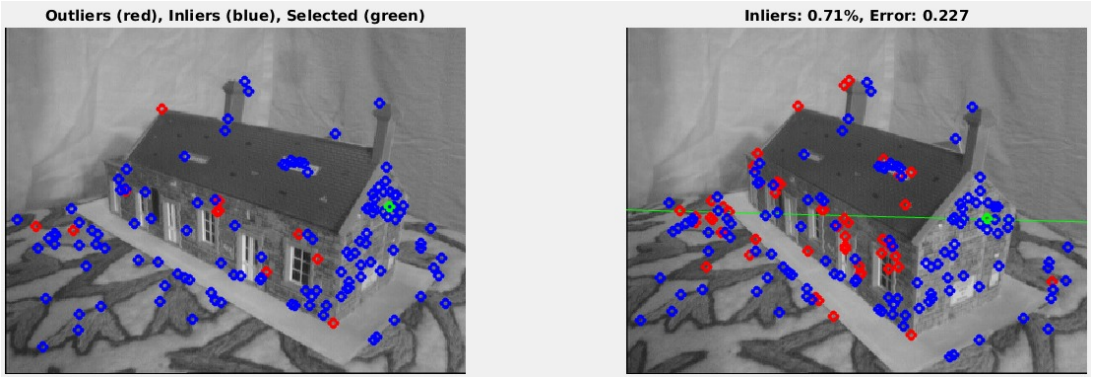
(d) Implement the function `get_inliers`, that accepts an estimation of a fundamental matrix, a complete set of correspondences and a threshold ε as an input and returns a sub-set of correspondences that are considered inliers for the given fundamental matrix. A point pair, $\mathbf{x}$ and $\mathbf{x}'$, are considered inliers if the distance between $\mathbf{x}$ and the epipolar line $\mathbf{F}^T\mathbf{x}'$ (as well as the other way around) is lower than ε.

(e) Implement the function `ransac_fundamental` that robustly estimates a fundamental matrix using RANSAC and a normalized eight-point algorithm. Implement a simple version of RANSAC algorithm that repeats k iterations and then returns the solution that is supported by the largest fraction of inliers.

Apply your version of RANSAC algorithm with parameters $\varepsilon = 2$ and $k = 100$ to a set of correspondences that you read from the file `house_matches.txt`. In order to visually inspect the success of the estimation of $\mathbf{F}$ choose a point in the left image-plane and display its epipolar line in the right image-plane. Make sure that the line passes the correspondence point in the right image-plane. You can also do

[4]You have probably noticed some outliers in the previous exercise when computing a homography.

that for multiple points. Set the parameters ε and k for optimal result. In a single figure plot all potential correspondence pairs from `house_matches.txt` as well as correspondence pairs that were chosen by the RANSAC algorithm as inliers (using a different color). What is the difference between the sets? Note that the examples below may be a bit different to your own results as the RANSAC algorithm is stochastic and therefore produces different result each time.



(f) Perform fully automatic fundamental matrix estimation. Detect the correspondence points using the function `find_matches` that you have implemented for the previous assignment. Using these matches, run the RANSAC-based fundamental matrix estimation. Display both images with both the matched point as well as the points retained by the RANSAC algorithm. Also display the percentage of inliers and the mean reprojection error. Set the parameters correctly so that any randomly chosen inlier point will lie on the corresponding epipolar line.

Note: The points lying on their corresponding epipolar lines is a necessary condition for successful completion of this task (i.e. the maximum reprojection error should be less than 1px).

(g) ★ (15 points) Implement robust homography estimation using RANSAC. The reprojection error for determining inliers can be calculated using Euclidean distance between the mapped points. Test your implementation on images from the previous assignment. Then, use the robust homography estimation to reconstruct the panorama in the folder `panorama`. You should detect and match feature points in both images, then calculate the homography matrix and use it to merge both images. Repeat the procedure until you get the entire panorama image. You can use the function `warpTwoImages` from the supplementary material to merge images.

Hint: You can reduce the area in which you search for feature points when that makes sense.

Note: Even the robust homography estimation can produce some small distortion. This error can accumulate when merging multiple images sequentially. Think how you can minimize the effect of the errors.

Exercise 3: Triangulation

If we know the intrinsic parameters of the cameras as well as the fundamental matrix, we can calculate the 3D position of the points observed in both cameras. You can

find the calibration matrices for both cameras stored in files `house1_camera.txt` and `house2_camera.txt`. The calibration matrices have the size 3×4.

We will use an algebraic approach to perform the triangulation. Assuming we have a 2D correspondence between $\mathbf{x}_1$ in the first image plane and $\mathbf{x}_2$ in the second image plane (in homogeneous coordinates), a location of the common point $\mathbf{X}$ in 3D space is given by relations

$$\lambda_1 \mathbf{x}_1 = \mathbf{P}_1 \mathbf{X} \tag{13}$$
$$\lambda_2 \mathbf{x}_2 = \mathbf{P}_2 \mathbf{X}. \tag{14}$$

We know that a vector product between parallel vectors is 0 so we use $\mathbf{x}_1 \times \lambda_1 \mathbf{x}_1 = 0$ and get:

$$\mathbf{x}_1 \times \mathbf{P}_1 \mathbf{X} = [\mathbf{x}_{1\times}] \mathbf{P}_1 \mathbf{X} = 0 \tag{15}$$
$$\mathbf{x}_2 \times \mathbf{P}_2 \mathbf{X} = [\mathbf{x}_{2\times}] \mathbf{P}_2 \mathbf{X} = 0, \tag{16}$$

where we have used the following form (shear-symmetric form) to get rid of a vector product:

$$\mathbf{a} \times \mathbf{b} = [\mathbf{x}_{1\times}]\mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \mathbf{b}. \tag{17}$$

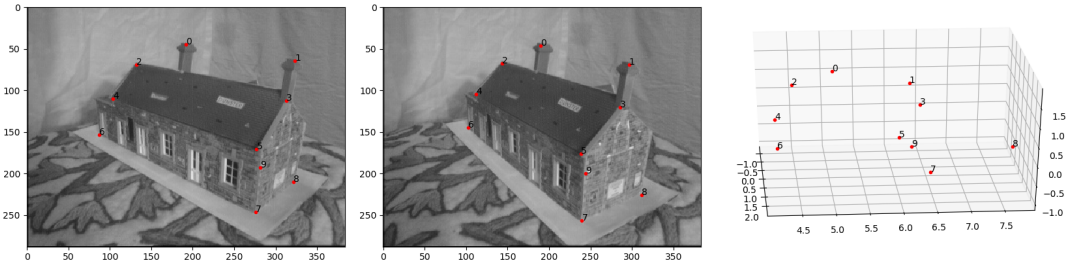
For each pair of 2D points we get two independent linear equations for three unknown variables. If we combine in a matrix $\mathbf{A}$ the first two lines of the product $[\mathbf{x}_{1\times}]\mathbf{P}_1$ and first two lines of the product $[\mathbf{x}_{2\times}]\mathbf{P}_2$, we can compute the mean quadratic estimate of $\mathbf{X}$ by solving a linear equation system $\mathbf{AX} = 0$. As you already know by now, such a solution can be obtained by computing the eigenvector of matrix $\mathbf{A}$ that has the lowest eigenvalue. Note that the solution of the system is a point $\mathbf{X}$ in homogeneous coordinates (4D space), therefore you have to first normalize the values so the last coordinate becomes 1.

(a) Implement the function `triangulate` that accepts a set of correspondence points and a pair of calibration matrices as an input and returns the triangulated 3D points. Test the triangulation on the ten points from the file `house_points.txt`. Visualize the result using `plt.plot` or `plt.scatter`. Also plot the index of the point in 3D space (use `plt.text`) so the results will be easier to interpret.

Note: The coordinate system used for plotting in 3D space is usually not the same as the camera coordinate system. In order to make the results easier to interpret, the ordering of the axes can be modified by using a transformation matrix. One such matrix is shown in the code snippet a.

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
from mpl_toolkits import mplot3d

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d') # define 3D subplot

res = triangulate(...) # calculate 3D coordinates
```

```
T = np.array([[-1,0,0],[0,0,1],[0,-1,0]]) # transformation matrix
res = np.dot(res,T)

for i, pt in enumerate(res):
    plt.plot([pt[0]],[pt[1]],[pt[2]],'r.') # plot points
    ax.text(pt[0],pt[1],pt[2], str(i)) # plot indices

plt.show()
```



(b) ★ (25 points) Perform a 3D reconstruction of an object you own. For that you will need to calibrate a camera with which you will take images of the object. You can use a webcam or a cell phone. Print out a calibration pattern[5] and take multiple images with the pattern visible on a planar surface (if you scale it correctly you can also take pictures of your screen). Take care that you change the orientation and distance between the pattern and the camera. Detect the circle centers using `cv2.findCirclesGrid` and check the correctness with `cv2.drawChessboardCorners`. Finally, use `cv2.calibrateCamera` to obtain the camera intrinsic parameters from the detected patterns (you can use the function `get_grid` from the supplementary material to get the pattern points' coordinates).

When you have the intrinsic parameters of your camera, you will need to take two images of your object from different viewpoints. First, undistort your images using `cv2.undistort`, then detect and match feature points. Calculate the fundamental matrix and, using the calibration matrix, also calculate the essential matrix. Then, you can use `cv2.recoverPose` to obtain the rotation and translation parameters (extrinsics) for both camera viewpoints. Use the extrinsic parameters to calculate the projection matrices for both cameras and triangulate the matched points. Display the final 3d points as well as the matches in both images.

Note: `cv2.recoverPose` will only return one rotation matrix and one translation vector. This is because the first camera lies in the origin of the coordinate system and its rotation matrix equals to the 3×3 identity matrix, while its translation vector only contains zeroes.

References

[1] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[5] You can use this one: https://nerian.com/nerian-content/downloads/calibration-patterns/pattern-a4.pdf