DATA.ML.300 Computer Vision
Exercise Round 6

For these exercises you will need Python. Return all your answers and output figures in a pdf-file. For pen & paper task(s), the submitted pdf-file should include a screenshot of hand written task, or text converted from Word or Latex format. In addition, submit runnable .py files, where you have filled in your codes to the template files. Do not include all the code from the .py files in the pdf. Exercise points will be granted after a teaching assistant has checked your answers. Returns done before the deadline will result in maximum of 4 points, whereas returns after the deadline will result in maximum of 1 point.

Task 1. Fundamental Matrix and Essential Matrix (Pen & paper) (1 point)
a) For what purpose are these matrices used and what are their differences?
b) How can you derive essential matrix from fundamental matrix and what additional information you need in order to do that?
c) How many degrees of freedom does fundamental matrix have and why?
d) How many degrees of freedom does essential matrix have and why?

Task 2. Camera calibration. (Programming exercise) (1.5 points)
In this exercise you will need to implement the direct linear transform (DLT) method for camera calibration. The algorithm is described in the lecture slides. It is also presented in the book by Hartley & Zisserman (Section 7.1 in the second edition).

The calibration object is a bookshelf whose dimensions are known. That is, width of a shelf is 758 mm, depth is 295 mm, and height between shelves is 360 mm.
Proceed as follows:

Run the script `cam_calibration.py`. The corners of the bookshelf are already manually localized from the given two images and visualized by the script. See the comments in the source code. Implement the missing function `camcalibDLT.py`, which should use the DLT algorithm described below.

$$\begin{pmatrix} 0^\top & \mathbf{X_1}^\top & -y_1\mathbf{X_1}^\top \\ \mathbf{X_1}^\top & 0^\top & -x_1\mathbf{X_1}^\top \\ \dots & \dots & \dots \\ 0^\top & \mathbf{X_n}^\top & -y_n\mathbf{X_n}^\top \\ \mathbf{X_n}^\top & 0^\top & -x_n\mathbf{X_n}^\top \end{pmatrix} \begin{pmatrix} \mathbf{P_1} \\ \mathbf{P_2} \\ \mathbf{P_3} \end{pmatrix} = 0$$

Here $\mathbf{X_n}$ is the known 3D coordinate, and x_n and y_n are the known image projection coordinates. Let's denote the above equation as $\mathbf{Ap} = 0$. The task is to find $\mathbf{p}$ minimizing $||\mathbf{Ap}||^2$ (homogeneous least squares).
The solution is given by the eigenvector of $\mathbf{A}^\top\mathbf{A}$ with the smallest eigenvalue.

After implementing the algorithm, both cameras should be calibrated correctly. Check the results visually.
Include both output images in your pdf, and return also your runnable version of `camcalibDLT.py`.

Task 3. Triangulation. (Programming exercise) (1.5 points)
In this exercise you will need to implement the linear triangulation method also described in the lecture slides. (The method is also presented in the book by Hartley & Zisserman.) You must again find a least-squares solution to a system of linear equations. In a similar manner as in exercise problem 1, it can be computed by solving the eigenvectors and eigenvalues of a real symmetric matrix.

As illustrated by the script, the points that will be triangulated are the corner points of the picture on the cover of the course book. As a result of the triangulation we will get the coordinates of these corner points in the world coordinate frame. By computing the distances between the points, we can measure the width and height of the picture in millimeters.

Proceed as follows:

Run the script `triangulation_task.py`. The corners of the picture on the book cover are already manually localized from the given two images and visualized by the script. See the comments in the source code. Your task is to implement the missing function `trianglin.py`.

Cross-product as a matrix multiplication can be written as

$$\mathbf{a} \times \mathbf{b} = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = [\mathbf{a}_\times]\mathbf{b}$$

The goal is to find the least squares solution to the two independent equations $[\mathbf{x}_{1\times}]\mathbf{P_1}\mathbf{X} = 0$ and $[\mathbf{x}_{2\times}]\mathbf{P_2}\mathbf{X} = 0$ in terms of $\mathbf{X}$. Start by calculating $[\mathbf{x}_{1\times}]\mathbf{P_1}$ and $[\mathbf{x}_{2\times}]\mathbf{P_2}$ and stack these vertically to acquire $\mathbf{A}$. This can then be used to calculate the least squares solution for the corresponding world coordinate $\mathbf{X}$ similarly as in task 1.

Triangulate the three given point correspondences and calculate the estimated width and height of the picture by computing the distances between triangulated world points.
Write the estimated width and height (printed by the program to the console) in your pdf, and return also your runnable version of `trianglin.py` and `triangulation_task.py`.