

ENPM673 – Perception for Autonomous Robots

Homework 1

Gokul Hari, Sakshi Kakde, Jayesh Jayashankar, Samer Charifa

Due date: 14th February 2022, 11:59PM

Submission guidelines:

- This homework is to be done and submitted individually.
- Your submission on ELMS/Canvas must be a zip file, following the naming convention
- YourDirectoryID_hw1.zip. If your email ID is abc@umd.edu or abc@terpmail.umd.edu, then your Directory ID is abc. Remember, this is your directory ID and NOT your UID.
- Please provide detailed instructions on how to run your code in the README.md file.
- For each section of the homework, explain briefly what you did, and describe any interesting problems you encountered and/or solutions you implemented.
- For Problem 4, you are not allowed to use built in functions for computing Singular Value Decomposition(SVD) and pseudoinverse directly. However, use of low-level inbuilt functions like functions for computing eigenvectors and transpose are allowed.

Problem 1: [20 Points]

Assume that you have a camera with a resolution of 5MP where the camera sensor is square shaped with a width of 14mm. It is also given that the focal length of the camera is 25mm.

1. Compute the Field of View of the camera in the horizontal and vertical direction.
2. Assuming you are detecting a square shaped object with width 5cm, placed at a distance of 20 meters from the camera, compute the minimum number of pixels that the object will occupy in the image.

Problem 2: [30 Points]

A ball is thrown against a white background and a camera sensor is used to track its trajectory. We have a near perfect sensor tracking the ball in [video1](#) and the second sensor is faulty and tracks the ball as shown in [video2](#). Clearly, there is no noise added to the first video whereas there is significant noise in the second video. Assuming that the trajectory of the ball follows the equation of a parabola:

1. Use Standard Least Squares to fit curves to the given videos in each case. You have to plot the data and your best fit curve for each case. Submit your code along with the instructions to run it.

(Hint: Read the video frame by frame using OpenCV's inbuilt function. For each frame, filter the red channel for the ball and detect the topmost and bottom most colored pixel and store it as X and Y coordinates. Use this information to plot curves.)

Problem 3: [30 Points]

In the above problem, we used the least squares method to fit a curve. However, if the data is scattered, this might not be the best choice for curve fitting. In this problem, you are given data for health insurance costs based on the person's age. There are other fields as well, but you have to fit a line only for age and insurance cost data. The data is given in .csv file format and can be downloaded from [here](#).

1. Compute the covariance matrix (from scratch) and find its eigenvalues and eigenvectors. Plot the eigenvectors on the same graph as the data. Refer to [this article](#) for better understanding. **[10]**
2. Fit a line to the data using linear least square method, total least square method and RANSAC. Plot the result for each method and explain drawbacks/advantages for each. **[15]**
3. Briefly explain all the steps of your solution and discuss which would be a better choice of outlier rejection technique for each case. **[5]**

The original dataset is provided by [kaggle](#).

Problem 4: [20 Points]

The concept of homography in Computer Vision is used to understand, explain and study visual perspective, and specifically, the difference in appearance of two plane objects viewed from different points of view. This concept will be taught in more detail in the coming lectures. For now, you just need to know that given 4 corresponding points on the two different planes, the homography between them is computed using the following system of equations $Ax = 0$, where A is given by

$$A = \begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1 * xp_1 & y_1 * xp_1 & xp_1 \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1 * yp_1 & y_1 * yp_1 & yp_1 \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2 * xp_2 & y_2 * xp_2 & xp_2 \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2 * yp_2 & y_2 * yp_2 & yp_2 \\ -x_3 & -y_3 & -1 & 0 & 0 & 0 & x_3 * xp_3 & y_3 * xp_3 & xp_3 \\ 0 & 0 & 0 & -x_3 & -y_3 & -1 & x_3 * yp_3 & y_3 * yp_3 & yp_3 \\ -x_4 & -y_4 & -1 & 0 & 0 & 0 & x_4 * xp_4 & y_4 * xp_4 & xp_4 \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & x_4 * yp_4 & y_4 * yp_4 & yp_4 \end{bmatrix}, x = \begin{bmatrix} H_{11} \\ H_{12} \\ H_{13} \\ H_{21} \\ H_{22} \\ H_{23} \\ H_{31} \\ H_{32} \\ H_{33} \end{bmatrix}$$

For the given point correspondences,

	x	y	xp	yp
1	5	5	100	100
2	150	5	200	80
3	150	150	220	80
4	5	150	100	200

Find the homography matrix:

$$H = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix}$$

1. Show mathematically how you will compute the Singular Value Decomposition (SVD) for the matrix A . **[10]**
2. Write python code to compute the SVD. **[10]**