Perception for Autonomous Robots

ENPM673

Homework 1

Author: Cheng Liu (117694802)

Date: 02/14/2022

Contents

Problem 1

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.

Answer:

1. The formula of Field of View is

$$FOV = 2 \times arctan\left(\frac{sensor\ size}{2f}\right)$$

The width and height of the sensor are both 14mm, and the focal length is 25mm. Therefore, both the horizontal and vertical direction of Field of View are

$$FOV = 2 \times arctan\left(\frac{14}{50}\right)$$

$$FOV = 31.28^{\circ}$$

2. The image size can be determined by

$$Image\ size = focal\ length \times object\ size \div object\ distance$$

Plugging the known values, we get

$$Image\ size = 25 \times 50 \div 20000 = 0.625mm$$

Implementing the area of sensor and the area of image, the minimum number of pixels can be calculated by

$$Area\ of\ sensor = 14 \times 14 = 196mm^2$$

$$Area\ of\ image = 0.625 \times 0.625 = 0.390625mm^2$$

$$\frac{Area\ of\ image}{Area\ of\ sensor} \times resolution\ of\ camera$$

$$= \frac{0.390625}{196} \times 5 \times 10^6$$

$$= 9964.92 \cong 9964\ pixels$$

Problem 2

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.)

Answer:

1. Based on the videos, the equation of a parabola can be assumed as

$$y = ax^2 + bx + c$$

To find the topmost and bottom most of the ball pixels, the contours of the ball need to be detected first. Converting the color to gray and using Canny function, the area of the ball could be found. Then utilizing findContours function determines the edge of the ball. Once we found the edge, the topmost and bottom most pixels would be available.
After storing those data in the numpy array, Least Squares method can be implemented. In the ls function, A matrix represents the equation of the parabola. Then estimating the state with the Least Squares

$$A = (X^T X)^{-1} X^T B$$

Then visualizing the result of the curves of the video1 and video2, we get

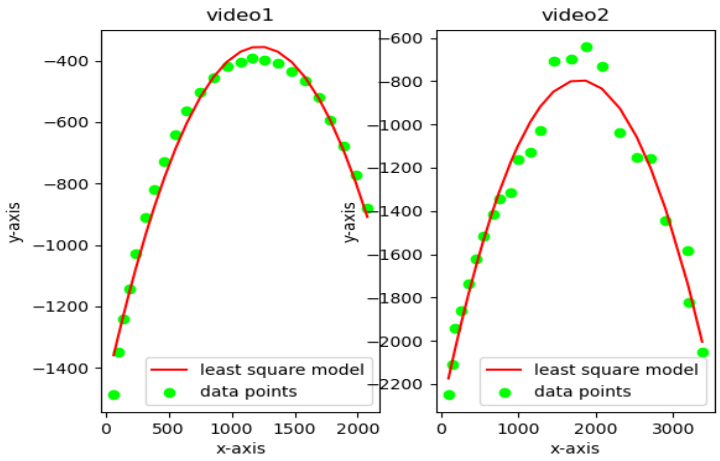

Fig 1. least square method applying for video 1 and video 2

Interesting Problem:

1. Originally, I tried to use inRange function to find the red pixels under rgb image. However, when I ran the code, the data showed nothing. Therefore, I changed the image into gray color and utilized Canny function to find the ball.
2. For Least Squares, OpenCV has an inbuilt function called solve function. When I implemented the function, the error said it couldn't find the DECOMP_NORMAL. I tried to search what is going on, but it seems like the version issue. Therefore, I wrote the function to achieve Least Squares.

Problem 3

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.
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.
3. Briefly explain all the steps of your solution and discuss which would be a better choice of outlier rejection technique for each case.

Answer:

1. Covariance matrix can be obtained by

$$cov[X, Y] = E[(X - E[X])(Y - E[Y])^T]$$

To get the eigenvalues, we can first assume that covariance matrix is A. Then eigenvalues are obtained by solving the equation given by

$$|A - \lambda \mathrm{I}| = 0$$

Assuming we have the difference between the matrix A minus the j^{th} eigenvalue times, the corresponding eigenvectors e are obtained by solving the expression

$$\left(A - \lambda_j I\right) e_j = 0$$

In the code, following the above equation, the covariance matrix, eigenvalues, and eigenvectors could be observed. Then the quiver function can plot the arrows of eigenvectors.

2. Least squares method is similar with the Problem 2. The parabola would be linear in this case. Therefore, the equation can be assumed as

$$Ax = B$$

Then estimating the state with the Least Squares

$$A = (X^T X)^{-1} X^T B$$

Total least squares method can be achieved by the U matrix

$$U = [(X_i - \bar{x})(Y_i - \bar{y})]$$

Then utilizing smallest eigenvalue and eigenvector to solve the linear equation

$$y = \frac{(A\bar{x} + B) - (Ax)}{B}$$

Ransac method requires the desired probability p provides a useful result during

executing the code and the probability of choosing an inlier w which is

$$w = \frac{number\ of\ inliers\ in\ data}{number\ of\ points\ in\ data}$$

That probability of k is the probability that the algorithm never selects a set of n points which are inliers, and this must be the same as $1 - p$

$$1 - p = (1 - w^n)^k$$

Then taking the logarithm of both sides can obtain the number of iterations

$$k = \frac{log(1-p)}{log(1-w^n)}$$

The result of above methods would be

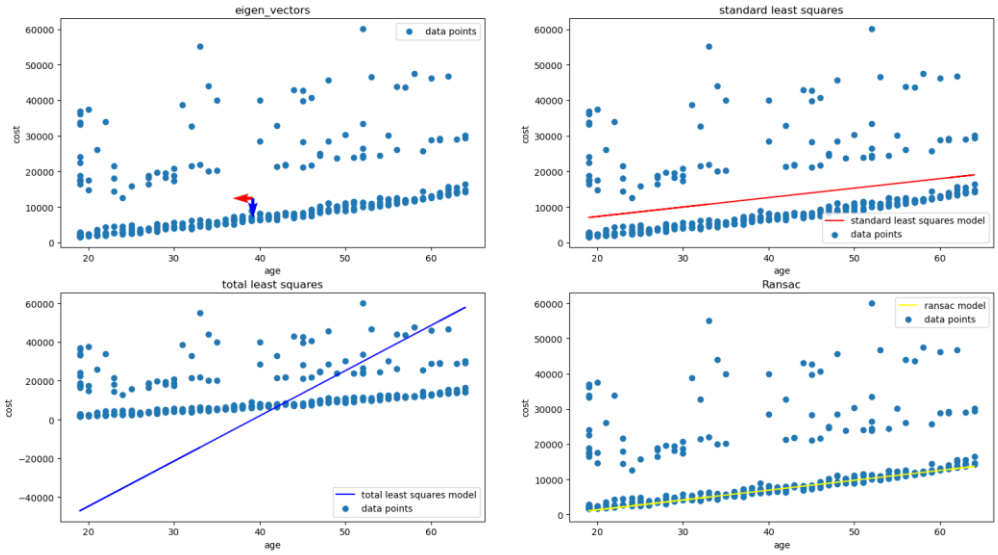

Fig 2. Eigenvectors, standard least squares, total least squares, and Ransac method on the dataset of health insurance

Based on the results of fig 2, we can observe the properties of least square and Ransac method.

Table 1. Least Square method properties

	Advantages	Disadvantages
1	Simplicity	Sensitivity of outliers
2	Large Applicability	Unreliable when the data is not normally distributed.
3	Theoretical Underpinning	Tendency to overfit data

Table 2. Ransac method properties

	Advantages	Disadvantages
1	Excluding outliers	Lots of parameters to tune
2	Often works well in practice	Doesn't work well for low inlier ratios

3. To complete the Problem 3, there are the following steps
    1. Input the data through pandas
    2. Generate covariance matrix by above equation
    3. Generate eigenvalues and eigenvectors by numpy.linalg.eig function
    4. Define the least square method and plug in the data
    5. Define the Ransac method and plug in the data
    6. Set up the parameters in the Ransac function (desired probability=0.95, sample number=3). If threshold is larger than the inlier count, the point is counted as an inlier.
    7. Plot all the result with matplotlib

    The result points out that Ransac method provides the best performance in this data. It ignores the outliers and follows the main trend of data points. Therefore, Ransac method is the adequate way to filter the outliers in this case.

Problem 4

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

Answer:

Singular Value Decomposition (SVD) can be achieved by

$$M = U\Sigma V^*$$

M is an $m \times n$ complex matrix, U is an $m \times m$ complex unitary matrix, Σ is an $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal, and V is an $n \times n$ complex unitary matrix.

U can be solved by

$$(AA^T - \lambda I) = 0$$

Similarly, V can be solved by

$$(A^T A - \lambda I) = 0$$

Σ can be computed by creating a diagonal matrix. Utilizing the np.diag function and np.sqrt function, Σ can be obtained by plugging λ into the functions. Computing SVD within U, V, and Σ, the homography matrix can be observed as follows

$$H = \begin{bmatrix} 0.0531 & -0.00492 & 0.615 \\ 0.0177 & -0.00393 & 0.787 \\ 0.000236 & -0.0000492 & 0.00762 \end{bmatrix}$$

Reference

- https://en.wikipedia.org/wiki/Singular_value_decomposition
- https://en.wikipedia.org/wiki/Least_squares
- https://en.wikipedia.org/wiki/Random_sample_consensus
- https://opencv.org/
- https://en.wikipedia.org/wiki/Total_least_squares