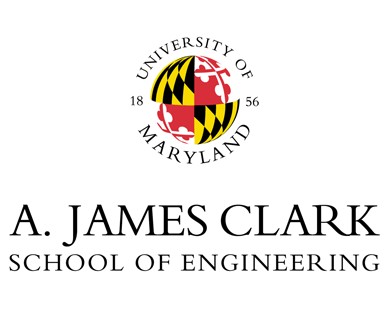
**Homework 1**

**Nisargkumar S Upadhyay**

**118221625**



**ENPM 673**

**--------------------------**

**PERCEPTION FOR**

**AUTONOMOUS ROBOTS**

**--------------------------**

**Dr. Mohammed Samer Charifa**

Contents

1. **Problem 1**
   1. FOV
   2. Pixels used by an image
2. **Problem 2**
   1. Standard Least Square Method
3. **Problem 3**
   1. Covariance Matrix and Eigenvalues, Eigenvectors
   2. LS, TLS, RANSAC
   3. Pros and Cons of LS, TLS, RANSAC
4. **Problem 1**
   1. Homography matrix
   2. SVD
   3. How to compute SVD

**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.

**Question 1**

Compute the Field of View of the camera in the horizontal and vertical direction.

**Solution 1**

The Field of View (FOV) represents how wide a camara can capture. Higher FOV will allow camera to capture larger object. FOV is the angular size of the view cone, its value is in degrees of angle, and it depends upon two factors.

The formula to find FOV is,

𝐹𝑂𝑉 = 2 × tan -1(d/2f)

Here d = sensor width and f = focal length of the camera

In our case

d = 14

f = 25

Thus,

𝐹𝑂𝑉 = 2 × tan -1(14/2\*25)

𝐹𝑂𝑉 = 2 × tan -1(0.28)

𝐹𝑂𝑉 = 31.28 degrees

Here the FOV in the horizontal and vertical direction will be same as the shape of the sensor is square

**Question 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.

**Solution**

The size of an image formed on the image sensor can be found using the formula that consist of following parameters. For this example, we will assume that the camera has thin lens.

h/w = 5 cm (Square object)

d = 2,000 cm

f = 25 mm

size of the image on the sensor = f \* (h/d)

size of the image on the sensor = 25\*(5/2000)

size of the image on the sensor = 0.0625 mm

Area of the image on the sensor

Area of the image = image height \* image width

Area of the image = 0.0625\*0.0625

Area of the image = 0.00390625 mm2

Area of sensor

Area of the sensor = sensor height \* sensor width

Area of the sensor = 14\*14

Area of the sensor = 196 mm2

Max numbers of pixels occupied is = resolution\*(Area of the image/ Area of the sensor)

= (5\*106  ) \* ( 0.00390625/196)

= 99.89 pixels

So, the numbers of pixels used by the object on the sensor is equivalent to 100

**Problem 2**

**Question 1**

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:

**Solution**

The goal here is to extract the images from the videos. Using those images, we have to find the x and y coordinates of the ball in each frame of the video and to plot the curve using least square method that fits the given data.

There are 2 videos in the problem, one is with the noise and the other has some noise present in it.

To solve this problem, we have to use some **libraries** in our python code

Cv2, glob, NumPy & matplotlib.pyplot

1. Cv2 - OpenCV is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. This library is written in the C++ language. We will use this library to perform various tasks on videos and images.
2. Glob - The glob module is a useful part of the Python standard library. glob is used to return all file paths that match a specific pattern. Glob will be used to call any object such as images from the computer into our code.
3. NumPy -It is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. As image is the set of arrays consisting of the intensity as there elements, NumPy will be used to perform mathematical and logical operations on it.
4. matplotlib.pyplot – Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. This library will be used to plot the graphs to visualize our data.

**STEP 1** – The first step will be to extract the frames from the videos. To do that,

We will call our video by VideoCapture method and store it in a variable called video. After that we will use read method to read the video. As video is nothing but series of frames put together this method will read every frame. We can store this frame in another variable and by using write method from open cv2 we can store this image at any place we want.

After that we can use glob library to call this image back into our code. In order to call the images in order we will use sort function.

**STEP 2** – The next step after we call the images back in our code, we have to iterate each image using the for loop. In the for loop, we will first separate out the red channel and convert the image in a binary value so it will be easy to operate and it will only contain 2 values.

**Step 3** – After we are left with images which only contains 2 values, we will use arg functions from the NumPy library to figure out the location of the desired pixels in an image. In our case we will take the topmost pixel using argmax function and will obtain the x and y coordinates. As we will go through the for loop we will record all the values of x and y coordinates and by using append function from the NumPy library we will from an array or list of those 2 coordinates. After that we will plot those points on the graph using matplotlib.pyplot and we will get the result as displayed below.

Chart

Description automatically generated Chart, scatter chart

Description automatically generated

Points from video 2

Points from video 1

As we can see the points that are plotted from the coordinates, we obtained from the second video are not in perfect curve shape as the points from video 1 due to the noise present in the video 2.

**Step 4 -** We got the points and we have plotted it on the graph. Now we have to use Standard least square method to draw the line which fits the given data. As the data we obtained is in curved shape so we will use a second-degree polynomial regression to get the curve line that fits our data.

Polynomial Regression = As we know that the linear regression line formal is

Y = M\*X + C. Here to predict the values of y based on the given data we have to find 2 unknows that are M and C. This linear regression is only useful when our data is in straight line that means the ratio between x and y is almost same.

In our case we will required the higher degree polynomial to compute the trajectory

Polynomial Regression is a form of Linear regression known as a **special case of Multiple linear regression** which estimates the relationship as an nth degree polynomial. Polynomial Regression is sensitive to outliers so the presence of one or two outliers can also badly affect the performance.

Text

Description automatically generated with low confidence

Will be the general formula which we will use to reach our goal as we can see above that it is a second order polynomial and contains 3 unknowns.

When we put all the coefficients on one side and represents them in a form of a matrix we end up getting

Text

Description automatically generated

We form a function to return all the 3 unknows by using the for loop going on each value of the X coordinate.

As we get the 3 values from the function we put those values in the formula

Text

Description automatically generated with low confidence

And we can get all the y values that are based on the x values that we obtained from the data. For our data our unknows where

B0 = 484.3896230068974 B1 = -1.6475953196285325 B2 = 0.002332437671316665

**Step 5 –** Our final step will be to take the max and min values of the x coordinates and the new predicted y coordinates and to plot that data on the graph on top of out data set of x and y coordinates.

Chart

Description automatically generatedChart

Description automatically generated

As we can in both the images the green line is the line formed by the polynomial regression of the second degree. In the image 1 we get a almost perfect curve passing through all the points of our data set.

In the Image 2 we get some of the outliers, this is due to the noise that was present in our video, so the movement of the ball was not in a perfect curve shape and so was our data. Because of which our regression model missed few point as it was predicting the values of the y coordinates.

**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.

**Question 1**

Compute the covariance matrix (from scratch) and find its eigenvalues and

eigenvectors. Plot the eigenvectors on the same graph as the data.

**Solution**

**Step 1** - Covariance matrix a covariance matrix is a square matrix giving the covariance between each pair of elements of a given random vector

It is matrix which contains all the covariance between 2 or more elements.

As per our data set we have to find the covariance between age and insurance cost.

As there are 2 elements the covariance matrix will be 2X2 square matrix

Suppose we take our age variable as X

And our cost of insurance as Y

Text

Description automatically generated with medium confidence

This will be our covariance matrix which gives us some kind of relation between 2 elements of a given matrix.

[[196.84122130177502, 52371.89683313604],

[52371.89683313604, 122217890.0842225]]

We have to compute eigenvalues and eigenvectors based on this covariance matrix.

Let us say we have a vector vv in RnRn. A linear transformation of vv is given by a matrix AA multiplied by vv. One could have a special vector vv such that the function AvAv returns a scaled version of vv, i.e., the direction of the vv is maintained upon a linear transformation by AA. This can mathematically be written as:

Av=λv

In this case the lambda is the eigenvalue, and the vector v is the eigenvector correspond to the eigenvalue.

To compute the eigenvalues and eigenvectors we will use the linalg class from the NumPy and from that we will us the object eig

When we pass our matrix through it our values will be stored in eigenvalues and eigenvectors

In our case

**import numpy as np**

**from numpy.linalg import eig**

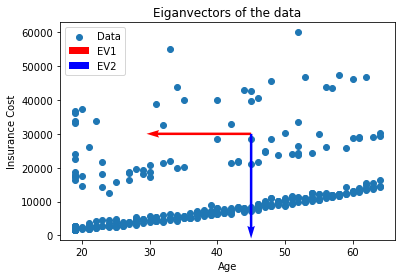
**w,v= eig(covariance\_matrix)**

Here w are the eigenvalues and v are the eigenvectors

**Step 2** – Once we get out eigenvectors, we can plot them on the graph using a quiver plot. Quiver plots is used to plot vectors on the graphs.

Eigenvectors shows us the direction of the spread of our data points we will use quiver plot to plot the graph.

Here EV1 is the larger eigenvector and the EV2 is the small eigenvector of the computed covariance matrix



**Question 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.

**Solution**

**Least Squares (LS) error:**

The Least Squares (LS) error correction method will minimize the error in vertical axis only. The error for each point is squared hence the name Least Squared error. This method is also known as Ordinary Least Squares (OLS) method.

Least Squares method takes into account only the error in vertical axis. Thus, the error in X-axis is not taken into consideration. One of the major effect of this inconsideration will be visible when the curve is a vertical one or has high slope. The major error in this case will be in the X-axis, which are not accounted for in this method.

Chart, scatter chart

Description automatically generated

In the least square method there are mainly 2 unknowns the slope and the intercept

Text, letter

Description automatically generated

We can obtain both of these variable by using for loop and iterating over each element of x and y coordinate.

Then we have to plug in the values in the linear equation to obtain the new y values which will be the new values of the cost of insurance with respect to age.

Chart, scatter chart

Description automatically generated

When we obtain the new y value we can plot this graph. The reason the line is not passing through the points because we have lots of outliers in our data due to which the line is shifted towards the outliers and away from majority of inliers.

**Total Least Square Method:**

The Total Least Squares (TLS) method will overcome the constrains of the LS method. TLS will consider the error in both the axis. The LS method was implementing Ax=B format where matrix A will have all the X parameters, matrix B will have all the Y parameters, while x will be a column matrix of parameters.

In the Total least square Matrix the error is calculated in the diagonal direction and not in any vertical or horizontal direction

A picture containing line

Description automatically generated

Here also are the 2 unknowns

To find the slope we will use

A picture containing text

Description automatically generated

From this equation we can get a slope of the line and to find the y intercept we will use



When we plug in this value in the equation of a line we get a new values of y for each values of x and then we plot the graph by min and max coordinates of a x and new\_y cooridinates.

Chart, scatter chart

Description automatically generated

Here the red line shows the Total least square method in compare to the least square method

**RANSAC:**

The Random Sample Consensus (RANSAC) method is used when there is a lot of noise. To understand the RANSAC, we need to investigate some drawbacks of TLS method. The TLS method will consider the normal error from all the points from the data set. So, if there is a data point that has ridiculously high error and can be considered completely false data will have an impact on the final output.

In ransac we have to choose 2 random points and draw a line on the graph,

From that line we have to calculate the distance of the points that are in the date.

Then we have to set the tolerance in the form of the distance, if the distance of the point is in the tolerance limit then it is an inlier or an outlier.

Timeline

Description automatically generated with medium confidence

We have to repeat this algorithm for n times and after that we have to check which line has max inliers, that line will be our RANSAC line.

Chart, scatter chart

Description automatically generated

When we run this algorithm on our dataset we get a black line passing through most of the data set when we iterate it for 700 times and with a tolerance of 7.

**Question 3**

Briefly explain all the steps of your solution and discuss which would be a better

choice of outlier rejection technique for each case

**Solution**

All three methods have their advantages and disadvantages. Least Squares error method is a faster method compared to Total Least Squares and less complex to implement for nonlinear systems, but LS does not consider the horizontal errors while TLS does. We can use the LS method where there are very few numbers of outliers and mostly the ration between the x and y coordinates remain same.

The major drawback of both cannot differentiate between noise and real data. If there are to many outliers present in the dataset then the line can be shifted towards the outliers and away from the inliers.

While RANSAC can differentiate between noise and real data.

Ransac is mostly used when there are outliers present in the dataset. As its algorithm consists of the parameter of the tolerance, it can help to find the line which consist most of the inliers.

But primary drawback of RANSAC is that it will take much larger time period to compute the plot, as it has to usually compute much larger number of iterations for some type of data set.

To conclude, I would recommend understanding the data set to which you want to implement the curve fitting method to. If the data is reliable and you want fast results with a compromise of the accuracy Least Square Method is the best option.

If you have reliable data set and want bit of a accuracy and the time to compute is not a tight constrain TLS is best option. However, TLS will also perform poorly if the dataset has to much of a noise

But, if you have data set that is full of noise and errors, I recommend using RANSAC as it will generate more reliable results compared to LS and TLS.

**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 homograph between them is computed using the following system of equations Ax = 0, where A is given by

Calendar

Description automatically generated

**Homography Matrix:**

Homography matrix is used to project an image on a plane that is not aligning with the current frame of reference either parallelly or perpendicularly. To put it into simpler words, let’s say we have an image in rectangular shape, and we want it to place on a plain that extends to horizon. The shape of image will now become trapezoidal.

Homography matrix is a mapping between two planes. We have considered it here as a mapping from the image plane to a physical plane, but it could map between two image planes. The inverse of a Homography will also provide the reverse mapping between the two planes.

The solution for homography matrix will be the last column of the VT matrix. After running the python code for the homography matrix. This will be compute as we compute the SVD of the matrix

**What is Singular Value Decomposition (SVD):**

The Singular Value Decomposition (SVD) is used for several image processing algorithms including but not limited to image compression, homography, image denoising and finding pseudoinverse for applications such as Least Squares (LS) and Total Least Squares (TLS) method.

SVD converts the matrix to a multiplication of three different matrices with first and last matrices are the singular orthogonal matrices, and the middle matrix is the diagonal matrix with diagonal values are singular values of that matrix, in this case matrix A

The diagonal sigma contains the values of the square of the eigenvalues of the matrix

𝑆𝑉𝐷(Z) = 𝑈∑𝑉𝑇

U = Orthogonal vectors of ZZT

V = Orthogonal vectors of ZTZ

∑ = Diagonal matrix with values σ1, σ2, … σr

SVD is majorly used to convert any given matrix to three distinct matrices which are U, ∑, and VT that corresponds to final rotation, scaling along with axis, and first rotation, respectively.

The SVD is used to find the homography matrix

The homography matrix will need to have the VT. The last column of that matrix will give us the values of the elements of the homography matrix.

The homography matrix is in a (3x3) format

So basically the homography matrix is the transpose of the orthogonal vectors of ZTZ.

## **How to find SVD?**

Suppose we have a matrix A, of which we want to find the SVD. The final three matrices after implementing SVD will be U, ∑, and VT.

Diagram

Description automatically generated

UTU = I and VTV = I as both are orthogonal matrices.

**V Transpose** - We need to find the ATA and AAT to calculate the VT and U. First we have to calculate the eigen values and eigen vectors of the matrix ATA. Then we have to sort the eigen values of that matrix in descending order. Then assemble the eigenvector matrix corresponding to those eigenvalues. The transpose of this matrix will be out VT matrix

**Sigma** - Now, we have to create a diagonal matrix with values of sigma (σ) with relation between eigenvalues (λ) being defined by



The components in the sigma matrix are all present in the diagonal of the matrix. In the case of the diagonal matrix, we need to adjust the shape of the matrix by adding rows and columns as we have to match to our input matrix

**U** - The U matrix is simply AAT. All the multiplication, transposes, and matrix formation is supported by different NumPy library.

Diagram

Description automatically generated with medium confidence

A screenshot of a game

Description automatically generated with low confidence

**References**

Singular Value Decomposition (SVD)

From <https://en.wikipedia.org/wiki/Singular_value_decomposition>

From <https://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm>

<https://jonathan-hui.medium.com/machine-learning-singular-value-decomposition-svd-principal-component-analysis-pca-1d45e885e491>

From <https://www.youtube.com/watch?v=cOUTpqlX-Xs&t=2s>

Finding Homography Matrix using Singular-value Decomposition and RANSAC in OpenCV and Matlab - Robotics with ROS

From <https://ros-developer.com/2017/12/26/finding-homography-matrix-using-singular-%20value-decomposition-and-ransac-in-opencv-and-matlab/>

From <http://faculty.salina.k-state.edu/tim/mVision/ImageFormation/homography.html#:~:text=Note%20that%20the%20homography%20matrix,mapping%20between%20the%20two%20planes>.

Least squares

From <https://ros-developer.com/2017/12/26/finding-homography-matrix-using-singular-%20value-decomposition-and-ransac-in-opencv-and-matlab/>

Total least squares

from <https://www.real-statistics.com/regression/total-least-squares/>

Covariance Matrix

From <https://www.youtube.com/watch?v=152tSYtiQbw&t=237s>

Polynomial Regression

From <https://www.youtube.com/watch?v=T-hTR77lel4&list=LL&index=2&t=799s>

OpenCV: Introduction to OpenCV-Python Tutorials

from <https://docs.opencv.org/master/d0/de3/tutorial_py_intro.html>