ENPM673 HW1

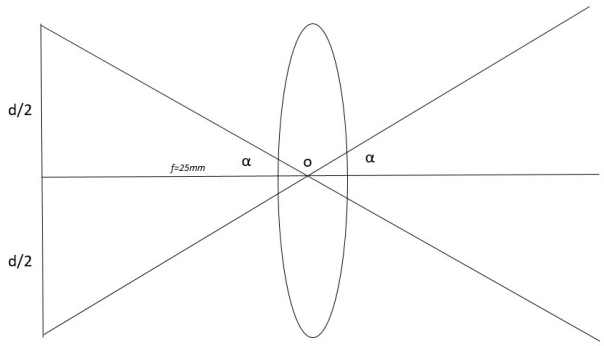
Sahruday Patti

February 2022

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.**

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

   The field of view depends on the Focal length and height of the camera sensor for Vertical FOV and width for Horizontal FOV.



$$\tan(\alpha) = Oppositeside/adjacentside$$

$$\alpha = \tan^{-1}(\frac{d/2}{f})$$

$$\alpha = \tan^{-1}(\frac{14/2}{25})$$

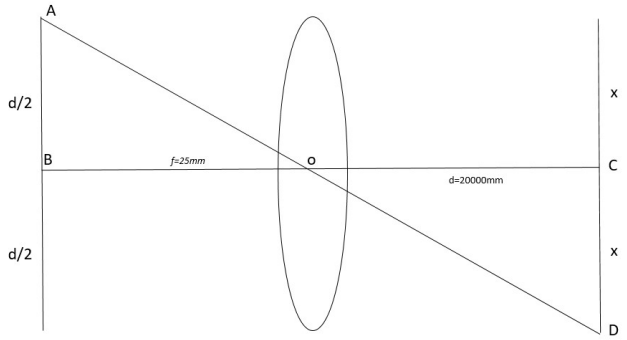
$$\alpha = 15.64224646$$

Total FOV is 2 times the angle.

FOV in Vertical direction = 31.2844

As the camera sensor is a square, the width is same as height. Therefore, FOV in horizontal direction is same as the Vertical FOV.

1. **b)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.**



Area of Field of View at 20m Distance

Here, i assumed that image plane is on the focal length so that the image is not inverted on the camera sensor and i am calculating the total FOV area which is a square area, so finding one side of that square which is 2x.

ΔABO and Δ OCD are similar triangles. Therefore,

$$\frac{d/2}{x} = \frac{25}{20000}$$
$$x = 5600$$

Area of FOV = $(2x)^2$

Therefore, FOV area = 125,440,000 mm^2

Resolution = Total No of Pixels = 5000000

Number of pixels per mm of FOV = $\frac{FOV area}{resolution}$

Area occupied per pixel in the FOV = 25.088 mm^2/pixel

Total area of a object = $2500mm^2$

Minimum no of pixels occupied by object = 2500 x (1/25.088)
= 99.64 pixels = 100pixels

Therefore an object at 20m distance with the given camera parameters placed in the FOV will occupy 100 pixels.

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:**

   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.

   The video is read using opencv frame by frame. each frame is first converted to gray scale and the thresholded. white is white and any value less than 255 is thresholded to black(0). Then, we can do it in two ways, first is to iterate through the loop and find the max and min coordinates of where the pixel value changes. I have used moments function is open cv which gives the center coordinate directly in the frame as there is only one object in the frame it gives me the center coordinate of the ball. The equation of parabola is $ax^2 + bx + c = y$

   Now, LLS (linear least square) output say h(x) = $\Sigma_{i=0}^{d} w_i x_i$ which can be written in matrix form as h(x)= XW

   For a set of data points, The matrices look like this where Y is the true value:

$$X = \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ . & . & . \\ . & . & . \\ x_n^2 & x_n & 1 \end{bmatrix} \quad W = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_n \end{bmatrix}$$

   The error measure for optimization is taken as squared error and the error is(for each point we caluclate the error and take the mean of errors for total error):

$$E = \frac{1}{N}\Sigma_{n=1}^{N}(h(x_n) - y_n)^2$$

In the matrix form it can be written as:

$$E = \frac{1}{N}\|XW - Y\|^2$$

Point to note is, Here we are learning, so the error is wrt W and X,Y are constants.
Now, in order to find the minima, we derivate and equate it to zero. here, we are taking the derivate on a bunch of points so it is called the gradient.

$$\nabla E = \frac{2}{N}X^T(XW - Y) = 0$$

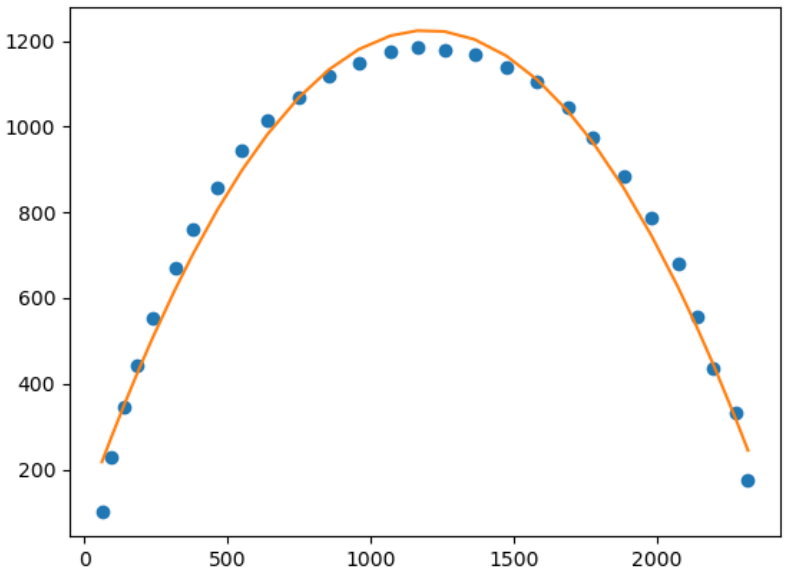
$$X^T XW = X^T Y$$

$$W = (X^T X)^{-1} X^T Y$$

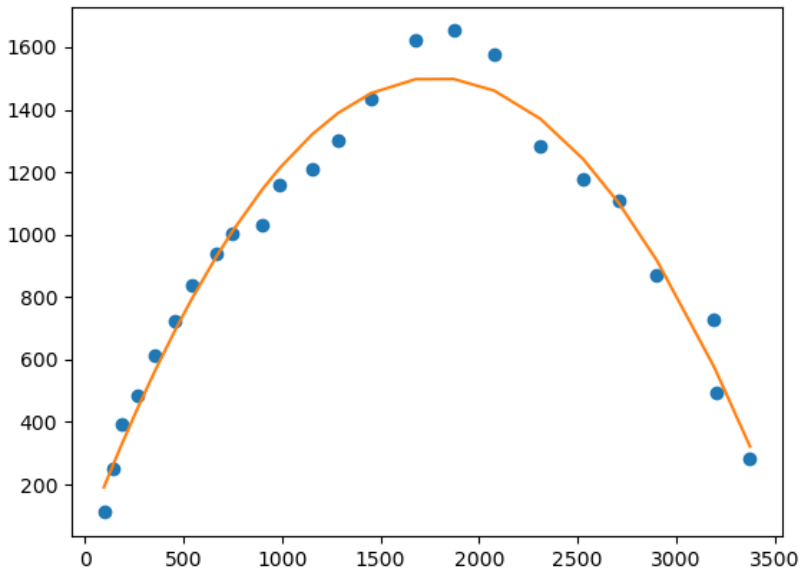
As you can see from above, the error measure is the difference between the Predicted value and True Value which is called the Residual Error.

In Total Least square, we minimize the shortest distance from point to the line i.e the perpendicular distance.

Below are the figures plotted for the ball Trajectory.



In Video1, as mentioned in question there is no noise from the sensor data so we are able to fit a perfect cure. we miss some points because, the ball doesn't follow a perfect parabolic path. It is assumed that it is a parabola so we get a small error, but if we can use a non linear transform we can fit the curve perfectly.



In Video2, There is Noise in the sensor data, but LLS does a decent job, it fits the curve in such a way that the error overall is minimized.

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.**

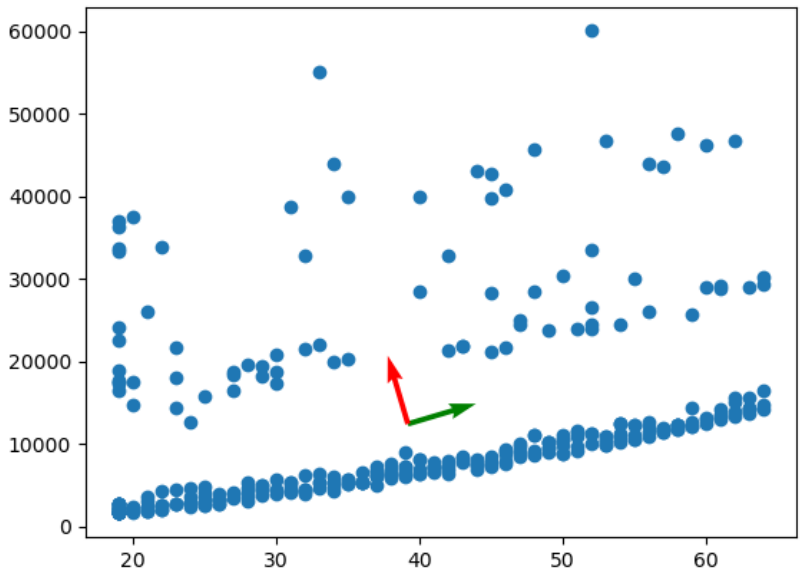
   a)Compute the covariance matrix (from scratch) and find its eigenvalues and eigenvectors. Plot the eigenvectors on the same graph as the data.

   Co variance plot: This is similar to doing Principle Component Analysis. For doing PCA we normalize the data, then compute the eigen vectors and then rescale it back. We normalize data because while calculating covariance we use Expected value of x squared and expected value of x

square which on a higher level can be called mean. I have used true mean to calculate the variance so i need to make it a normal distribution so as to calculate the true mean.

Note the directions of the vectors, these are two lines that produce same error in the data. one direction of eigen vector has maximum variance and minimum error line and other direction has minimum variance and maximum error line. In a data set, while fitting the model we can reduce the bias and variance by a process called validation.

Variance is always present and depends on the data set given. In the TLS, we can observe the same two lines, when we plot the second eigen vector corresponding to highest eigen value of UU^T.

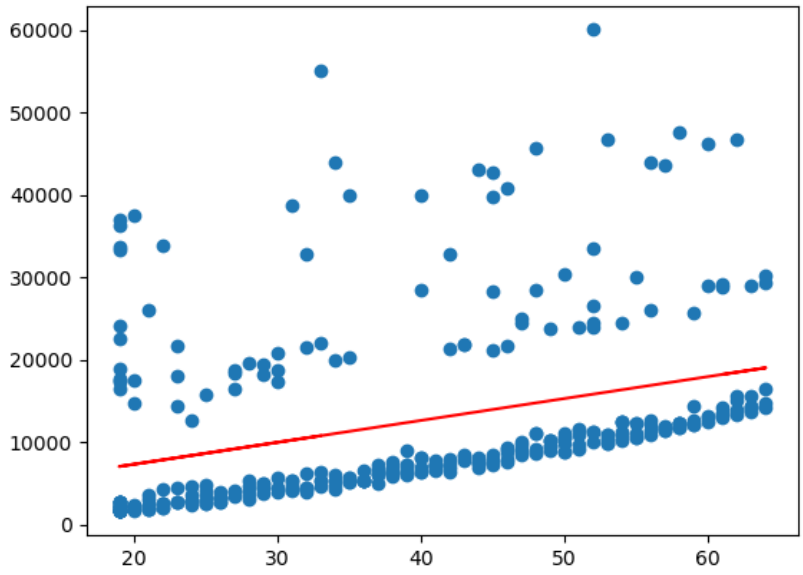


(a) **Fit a line to the data using linear least square method, total least square method and RANSAC. Plot the result for each method and explain drawbacks/advantages for each.**

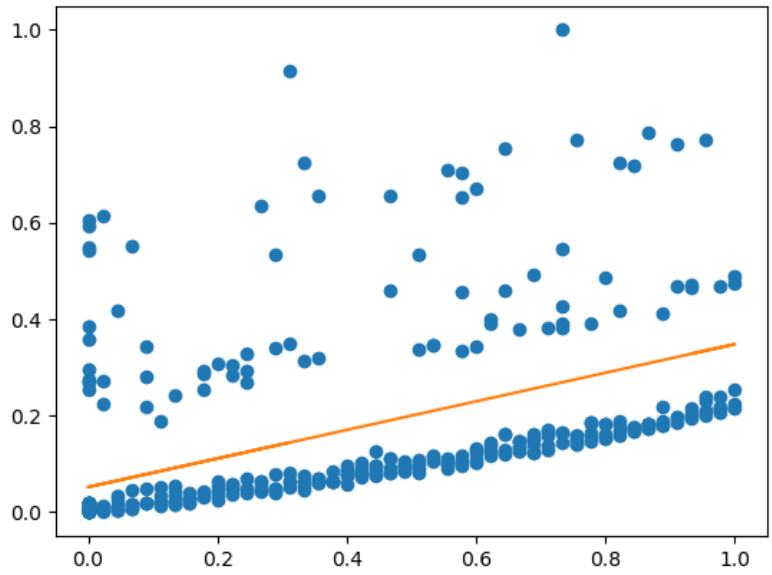
The LLS plot:

As Solved in the previous Problem, the LLS minimizes the mean squared error of the data points with respect to the Residual error i.e the difference between the predicted value and true value. so, it is susceptible to outliers but not very much. The advantage is that it is a simple one step learning process and LLS combined with non

linear transforms and regularization can produce incredible results for as simple a process as this.



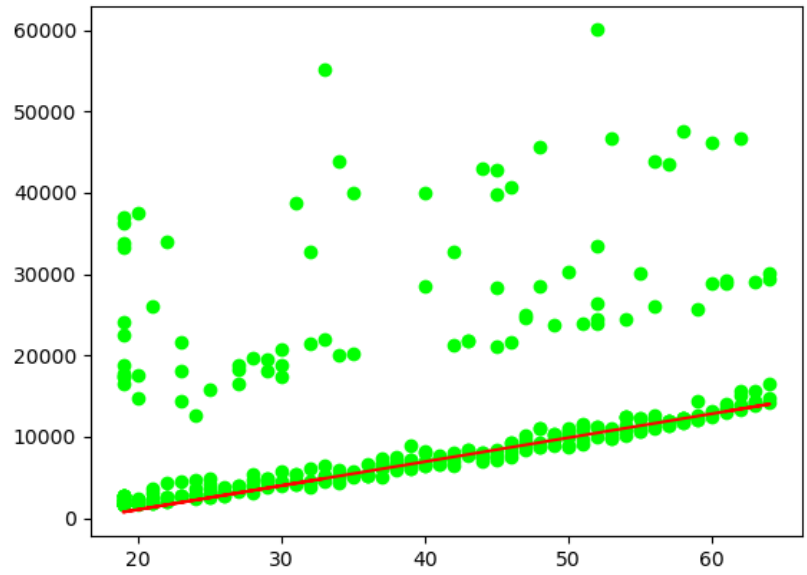
The Total Least square Plot:



The Total Least square follows the same optimization procedure as LLS but the error function is between the predicted value and the shortest distance to the True Value i.e the perpendicular distance. As this measure includes shortest distance it fits a line which minimizes the Overall Perpendicular distance to points so it also takes the outliers into account. In other words,total least squares weights the residuals by multiplying them with the inverse of the corresponding error covariance matrix in order to derive a consistent estimate.
One of the advantages is that similar to LLS,solutions are direct solutions and thus no starting values are needed.
It is similar if not exactly a Perceptron model.If we are trying to fit a line to a data which has vertical offsets or for classification then TLS would be a good model.

Point to note is LLS is a scale invariant model, because we use absolute difference between magnitude as an error measure and optimize it, but in TLS we use a squared perpendicular distance error measure which is also called lasso regression which is not invariant to scale, so inorder to get proper slope and accuracy normalizing the data is important.

The Ransac Plot:



The advantage of Ransac is if there are more inliers than outliers in the data, it gives a model which fits the inliers and rejects the outliers. It is because, the process is such that we take in voting, i.e we see how many points lie within the threshold of that line and we take the best line which has most no of points within the threshold. The disadvantage is that, there is no upper bound on the time limit it takes to converge we place a probabilistic measure on the convergence and how good a model it gives is dependent on the threshold which is user defined, if the threshold is too high then bad hypothesis are also ranked equally with a good model and if the threshold is too low, good hypothesis are also qualified as bad. so it is sensitive to user defined threshold. Also, it gives us one of the many good models which may not be the optimal solution.
The process of Ransac used is explained in the next section.

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

The procedure for LLS has been explained in the previous question. we find the model parameters and we fit a line, in this case it is a straight line so the line is of the form y = mx + c so we only need 2 model parameters. so weight matrix has only two elements which are needed to fit a line. After calculating Pseudo inverse, found the model parameters and used them to plot a line which separates the

data.

For TLS, the line equation is of the form $ax + by = c$, so similar to LLS we minimize the equation with a contrain $a^2 + b^2 = 1$. The error function becomes $E = \Sigma_{i=1}^{N}(ax_i + by_i - d)^2$. To minimize the equation we take the gradient and equate it to zero. After the mathematical solving similar to LLS, we get that the solution to above Optimization problem as $(U^T U)N = 0$ subject to $\|N\|^2 = 1$. where U and N are: $U = \begin{bmatrix} x_1 - xmean & y_1 - ymean \\ . & . \\ . & . \\ x_n - xmean & y_n - ymean \end{bmatrix}$ $N = \begin{bmatrix} a \\ b \end{bmatrix}$
The solution to the above is given the eigenvector of $U^T U$ associated with the smallest eigenvalue. This is an approximate solution because minimum norm solution is not always unique. This is a least square problem and why this is the solution is explained in the next question. so in the code, calculated the $U^T U$ which is called the second moment matrix and took the eigenvalues and eigenvectors of second moment matrix and sorted them from big to small and took the eigen vector corresponding to the smallest eigen value as my model and plotted the line with the data. As, we can observe LLS and TLS are similar models based on same optimization procedure and similar error funtions except that the error measure is Residual in LLS and perpendicular in TLS. Differences between the least-squares and total least-squares solution increases when the ratio between the second smallest singular value of $[UN]$ and the smallest singular value of U is growing. This can be seen by plotting the line of total least square with its highest in this case the second eigen vector corresponding to highest singular value.

RANSAC: Random Sample Consensus(RANSAC) is a curve fitting algorithm that is generally used when the data contains outliers. RANSAC estimates a model that fits the data by neglecting the effect of outliers in the estimation. Implementation of RANSAC iteratively follows the following steps:
1. Randomly select two points from the data.
2. Generate a model (line in this case) that fits the selected points. (I have used my implementation of linear least squares for computing the model.)
3. For each remaining point, calculate the error. If error is greater than a specified threshold (specified by the user) mark it as an outlier, otherwise add the point to the set of inliers.
4. If the number of inliers > the number of inliers of previous best model update the model as the best model.
5. The number of iterations are calculated dynamically, i chose to use

that with 95 percent probability we get a good sample to calculate the number of iterations. The number of iterations to be done is calculated by the formula $N = \frac{\log(1-p)}{\log(1-(p)^s)}$. where p is the desired probability i.e 95 percent, s is the total number of samples.
6. The threshold can be played around with, and i took the threshold as the standard deviation of values divided by 2.
7. The iterations process ends when the total iterations done are greater than the iterations to be done given by the formula above.

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

$$A = \begin{bmatrix} -x1 & -y1 & -1 & 0 & 0 & 0 & x1*xp1 & y1*xp1 & xp1 \\ 0 & 0 & 0 & -x1 & -y1 & -1 & x1*yp1 & y1*yp1 & yp1 \\ -x2 & -y2 & -1 & 0 & 0 & 0 & x2*xp2 & y2*xp2 & xp2 \\ 0 & 0 & 0 & -x2 & -y2 & -1 & x2*yp2 & y2*yp2 & yp2 \\ -x3 & -y3 & -1 & 0 & 0 & 0 & x3*xp3 & y3*xp3 & xp3 \\ 0 & 0 & 0 & -x1 & -y1 & -1 & x3*yp3 & y3*yp3 & yp3 \\ -x4 & -y4 & -1 & 0 & 0 & 0 & x4*xp4 & y4*xp4 & xp4 \\ 0 & 0 & 0 & -x4 & -y4 & -1 & x4*yp4 & y4*yp4 & yp4 \end{bmatrix} \tag{1}$$

$$X = \begin{bmatrix} H_{11} \\ H_{12} \\ H_{13} \\ H_{21} \\ H_{22} \\ H_{23} \\ H_{31} \\ H_{32} \\ H_{33} \end{bmatrix} \tag{2}$$

for the given point correspondences,

4.

	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:

$$H = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \tag{3}$$

Ans: The Homogeneous linear equation system given by the expression Ax = 0, we have 9 unknowns in x and A is a matrix of M x N (8x9) coefficients, which means that we have more unknowns than the number of independent equations. So, the system is under constrained, so we cannot have an exact solution except for the trivial solution. In order to find the Non trivial solution, which is approximate we constrain the vector x magnitude to $\|x\|^2 = 1$. This becomes a quadratic optimization problem.But for a simple quadratic problem in matrices, we can solve for Ax= 0 subject to constraint $\|x\|^2 = 1$ by Singular Value Decomposition of matrix A.

This is a constrained optimization problem $min_{\|x\|^2=1}\|Ax\|$. This solution is approximate because the minimum-norm solution for homogeneous systems is not always unique.

So, we are minimizing the $E = \|Ax\|^2$ subject to the constraint $\|x\|^2 = 1$. Here E is the error measure, which is the Total least square i.e we are minimizing the squared error of shortest distance between the point and the vector x which is essentially a TLS.

so to find the solution we should minimize $\frac{\|Ax\|^2}{\|x\|^2}$.

SVD factors the matrix into a two orthonormal matrices U and V and a diagonal matrix D such that $A = U\Sigma V^T$.

Now, lets say the solution to the above optimization problem is $\hat{x}$.

$$\hat{x} = argmin_x \|U\Sigma V^T x\|^2 \tag{4}$$

Constrain is carried throughout the development of the solution,so not mentioning it throughout.
For an orthonormal vector set, the magnitude of vectors is 1 and are orthogonal to each other. So the orthonormal basis set just rotates the data but doesn't have any effect on the magnitude. we have constrained the above optimization problem with a magnitude i.e $\|x\|^2 = 1$. Therefore, we can write the above equation as,

$$\hat{x} = argmin_x \|\Sigma V^T x\|^2 \tag{5}$$

Now, say $y = V^T x$, therefore the equation becomes,

$$\hat{y} = argmin_y \|\Sigma y\|^2 \tag{6}$$

Now, if we have sorted diagonal entries in descending order, minimum deviation from the ideal line would give us the best-fit solution. The above equation is minimum when all the other diagonal entries are zero and we multiply with the least singular value. i.e $y = [0, 0, ..., 1]^T$

we know x = Vy (since for an orthogonal matrix, its transpose is equal to inverse). Now if we multiply y with V we get the last column of V as our Solution.

Therefore, the last column of V is the solution for x

For the given equation, U is a M x M (8 x 8) matrix with columns as orthogonal eigen vectors of AA^T.

$A = U\Sigma V^T$
$AA^T = U\Sigma V^T V\Sigma^T U^T$
$AA^T = U\Sigma\Sigma^T U^T$
This is similar to eigen vector decomposition. Therefore the columns of U are orthogonal eigen vectors of of AA^T and Σ^2 are eigen values of $A^T A$

.

$$U = \begin{bmatrix} -7.101e{-}03 & 1.717e{-}02 & 1.0576e{-}03 & 1.727e{-}01 & -7.137e{-}01 & -2.626e{-}01 & 6.253e{-}01 & -1.719e{-}02 \\ -7.10e{-}03 & 1.717e{-}02 & 1.057e{-}03 & 1.340e{-}01 & 6.995e{-}01 & -2.662e{-}01 & 6.491e{-}01 & 3.030e{-}03 \\ -2.2e{-}01 & 5.146e{-}01 & 6.770e{-}01 & 4.242e{-}01 & 1.117e{-}02 & -2.104e{-}02 & -1.262e{-}01 & 1.755e{-}01 \\ -8.88e{-}02 & 2.058e{-}01 & 2.708e{-}01 & -4.215e{-}01 & -1.002e{-}02 & 6.363e{-}01 & 3.539e{-}01 & -4.103e{-}01 \\ 9.242e{-}01 & 3.815e{-}01 & 1.329e{-}02 & 7.669e{-}06 & 4.882e{-}09 & 1.101e{-}06 & -7.169e{-}06 & -2.912e{-}06 \\ -1.703e{-}01 & 4.118e{-}01 & 2.534e{-}02 & -6.732e{-}01 & -1.790e{-}02 & -5.811e{-}01 & -9.261e{-}02 & -2.883e{-}02 \\ -1.089e{-}01 & 2.745e{-}01 & -3.057e{-}01 & 3.689e{-}01 & 2.392e{-}02 & -1.488e{-}01 & -1.673e{-}01 & -7.952e{-}01 \\ -2.178e{-}01 & 5.491e{-}01 & -6.115e{-}01 & 7.118e{-}02 & -4.801e{-}03 & 3.079e{-}01 & 9.892e{-}02 & 4.090e{-}01 \end{bmatrix}$$

V is a N x N (9 x 9) matrix with columns as orthogonal eigen vectors of $A^T A$. this can simillarly be shown like above.

$$.V = \begin{bmatrix} -0.0009 & -0.0153 & 0.3737 & -0.3854 & -0.4365 & -0.3142 & 0.6497 & 0.0043 & 0.0035 \\ 0.0003 & -0.013 & 0.3169 & -0.3348 & 0.1717 & 0.8576 & 0.1489 & 0.0003 & 0.0033 \\ -0.0000 & -0.0001 & 0.0030 & -0.0031 & -0.0007 & 0.0013 & -0.0029 & 0.5813 & -0.8136 \\ -0.0003 & -0.0058 & 0.1459 & 0.5796 & -0.7273 & 0.3299 & -0.0694 & -0.0031 & -0.0025 \\ 0.0008 & -0.0088 & 0.2180 & 0.6342 & 0.4831 & -0.0274 & 0.5620 & 0.0018 & -0.002 \\ 0.0000 & 0.0000 & 0.0001 & 0.0055 & -0.0001 & 0.0000 & -0.0064 & 0.8136 & 0.5813 \\ -0.7179 & -0.6955 & -0.0281 & 0.0005 & 0.0009 & 0.0002 & -0.0002 & 0.0000 & -0.0000 \\ 0.6960 & -0.7173 & -0.0288 & -0.0002 & -0.0007 & -0.0005 & -0.0002 & 0.0000 & -0.0000 \\ -0.0000 & 0.0332 & -0.8302 & -0.0328 & -0.1319 & 0.2367 & 0.4847 & 0.0059 & 0.0000 \end{bmatrix}$$

Thus, U and V are aptly called Left eigen vectors and right eigen vectors of the matrix A.

Σ is a M x N (8 x 9) diagonal matrix, with Singular values as the diagonal entries of the matrix.

Eigen values $(\lambda_1, ..., \lambda_n)$ of AA^T , A^TA are equal. Here, we get 8 eigen values. Then the singular values of A are given by: $\sigma_i = \lambda_i^{1/2}$

We only have 8 eigen values, so in order to obtain the 8x9 matrix we append the last column of Σ with One. This is intutive because as the U and V have the dimensions we expected this means that the singular value of zero exists. if otherwise, we should have padded zeros to U or V depending on $M < NorM > N$.

$$\Sigma = \begin{bmatrix} 53387.12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 46716.00 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 31804.12 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 221.61 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 147.08 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 136.31 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 47.22 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.130 & \end{bmatrix}$$

This gives us the smallest singular value as zero. So the solution lies in the Null Space of A.The solution to x is given by the last column of V (as shown above that $y = [0, 0, ...1]^T$).

Therefore, x is equal to the last column of the matrix V as x = Vy and we multiply the last column with 1 and others with zero.

Therefore,

$$H = \begin{bmatrix} 0.0035 & 0.0033 & -0.8136 \\ -0.0025 & -0.002 & 0.5813 \\ -0.0000003 & -0.000002 & 0.000009 \end{bmatrix}$$

Normalizing the above matrix, with normalizer being the last element H_{33}. This is because, the system is under determined and we bound the values to a scale with last element being 1. The normalized H matrix is:

$$H = \begin{bmatrix} 3.82101200e{+}02 & 3.68403e{+}02 & -8.874e{+}04 \\ -2.784e{+}02 & -3.192e{+}02 & 6.3406e{+}04 \\ 3.9942e{+}02 & -2.1888e{+}02 & 1.00 \end{bmatrix}$$

Note: If i use the Ubuntu numbers here, the H match looks like this without normalization:

$$H = \begin{bmatrix} 0.0531 & -0.0049 & -0.6146 \\ 0.0177 & -0.0039 & 0.0786 \\ 0.00023 & -0.00004 & 0.0076 \end{bmatrix}$$

I have used windows all through out the assignment because i got some problems playing videos using opencv on Ubuntu and not able to sort it. Hence, i am using all the values calculated on windows through out the assignment.