ENPM673 Homework 1 – Report

By Divyam Garg UID : 117032475

Ans1) Part A:

Given :

f = 15mm

h (image plane width) = 14mm

X (object width) = 5cm

Z (distance) = 20m,

We Know that

$$FOV = 2 * \tan^{-1} \frac{h/2}{f}$$

Hence,

$$FOV = 2 * 24.985^{\circ}$$

$$FOV = 49.97^{\circ} \sim 50^{\circ}$$

Part B:

Using given information from part A, we must calculate minimum number of pixels covered by the image formed on image plane. The minimum area will be covered when the image converges exactly from focal point. Camera Resolution given : 9MP

Since the image plane is square , $Height = Width = 3 * 10^3 pixels$

Now since we concluded that the image must be properly focused, to calculate width and height of the image we can use the formula,

$$h = x = X * \frac{f}{Z}$$

Plugging in the values we get, $x = \ 0.0375mm$

Now,

For 14 mm there are $3 * 10^3 pixels$,

For 0.0375 mm total number of pixels used would be

$$\frac{3 * 10^3}{14} * 0.0375 = 8.035$$

Thus, the minimum area nor the minimum number of pixels required are:

$$8.035 * 8.035 = 64.56 \sim 65\, pixels$$

Ans 2)

The steps used fit the parabolic curve using three different method namely standard least squares, total least squares and Ransac were as follows.

Step 1) Pre-Processing the video:

- Video was opened using open-cv library and was converted into gray scale to remove excessive information (i.e. [RGB] values which would increase the computation)
- The video was run frame by frame and all the cells which did not have a value of 255 (white) were stored in a list.
- The center point of ball was extracted by taking average of topmost and bottommost cell and was stored in a different list for curve fitting and plotting.

Step 2) Least square:

- For least square curve fitting the standard formula of $AX = B$ is used where A, X and B are:

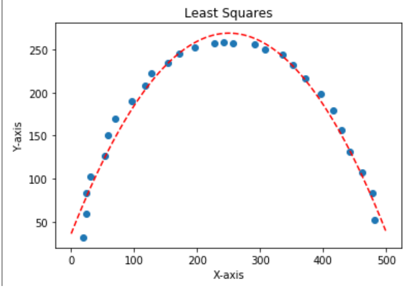
$$A = \begin{pmatrix} n & \sum x & \sum x^2 \\ \sum x & \sum x^2 & \sum x^3 \\ \sum x^2 & \sum x^3 & \sum x^4 \end{pmatrix} \qquad B = \begin{pmatrix} \sum y \\ \sum x * y \\ \sum x^2 * y \end{pmatrix} \qquad X = (A)^{-1} * B$$

- All the center points stored in a list were used for the equation of parabola which is:

$$y = a + b * x + c * x^2$$

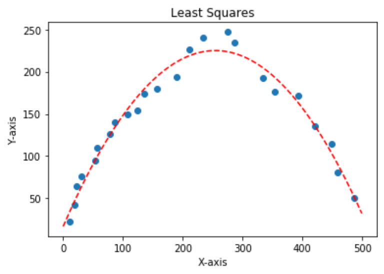
- Here, a,b and c were calculated using above method and a model was estimated.
- Note: Using inverse of A.T*A might result in loss of some information, hence this method was used instead of directly forming equation using three variables.
- For both the videos this method was implemented, and the following results were as follows.

```
[ 3.62345882e+01  1.85890214e+00 -3.71134430e-03]
```



Video 1 – smooth

```
[ 1.67950603e+01  1.63750413e+00 -3.21989392e-03]
```



Video 2 – noise

- As you can see, it gives a pretty accurate curve fit for smooth trajectory, but it can do much better in video2. Although it is still a good fit, the noise is affecting the trajectory of curve.
- Since conventionally Least squares are regressed with the influence of y-coordinates of all the points it takes noise into consideration.
- To fix this problem the concept of ransac (Random Sampling Consensus) is introduced which does not take outliers into account after a certain threshold.

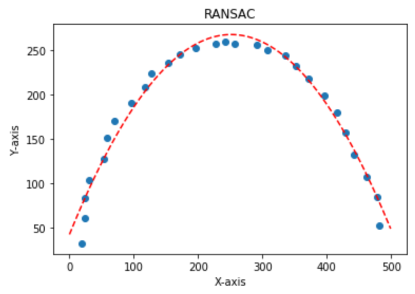
Step 3) RANSAC

- As discussed above RANSAC (random sample consensus) is an iterative method where it finds a model with most inliers given a threshold.
- This allows all the outliers or noise to be completely ignored from the original trajectory making it a very robust curve fitting method.
- In this implementation a certain threshold and randomly 3 points were taken as an input to model the curve using least square method.
- This sampling was done randomly for many steps until the termination and best model was preserved.
- Parameters set were desired probability = 0.95, outlier ratio = 0.6, threshold = std_deviation/2 (of y values) , no. of samples (s) = 3.
- The number of iterations (N) hence for the best model was calculated using this formula and was updated and incremented by one after each step in the loop:

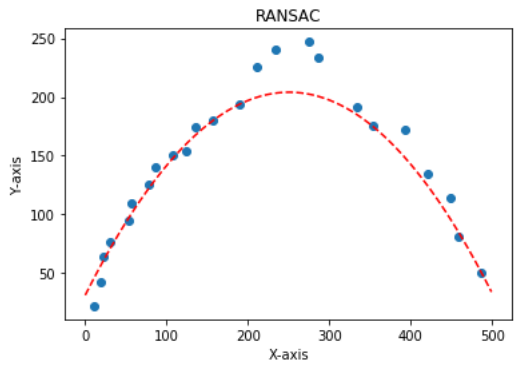
$$N = \frac{\log\,(1-P)}{log(1-(1-w)^S)}$$

- Here P is desired probability, w is outlier ratio, S is no. of samples
- The following results are as follows:

```
[ 4.20517851e+01  1.79173075e+00 -3.56479703e-03]
```



```
[ 3.12176309e+01  1.38016529e+00 -2.75482094e-03]
```

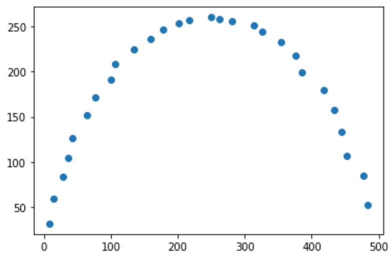


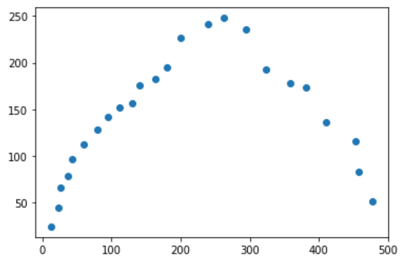
- Here an important thing to observe was with the increase in sample no. and outlier ratio the iterations were increasing exponentially thus, it is important to find the correct balance.
- It proved to be a very robust curve fitting solution, ignoring the noise creating and estimating a correct model.

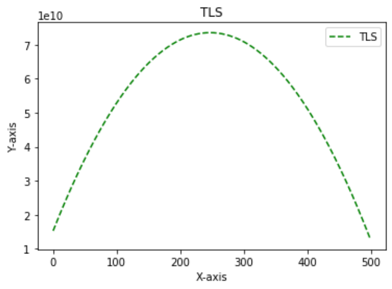
Step 4) Total Least Square

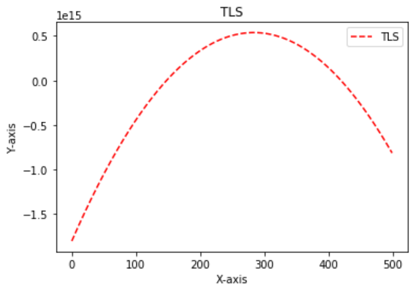
- This method is very similar to standard least square. The only difference is, during regression the perpendicular distance to curve is taken instead of horizontal, estimating both x and y.

$$\text{least squares: } \widehat{x}_{\text{ls}} = (A^\top A)^{-1} A^\top b,$$
$$\text{total least squares: } \widehat{x}_{\text{tls}} = (A^\top A - \sigma_{n+1}^2 I)^{-1} A^\top b, \ (*)$$
$$\text{where } \sigma_{n+1} \text{ is the smallest singular value of } [A \ \ b].$$ [1]



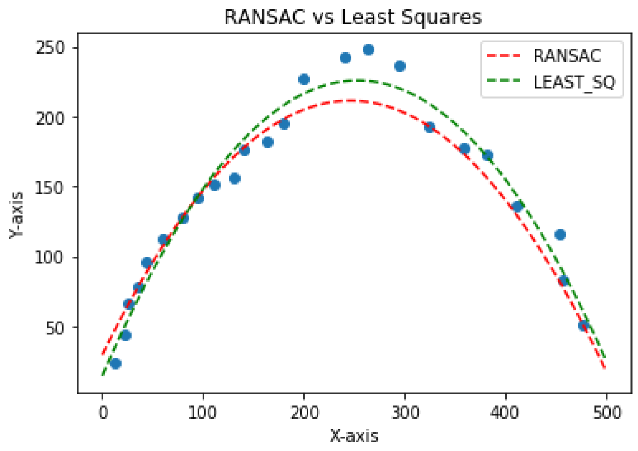


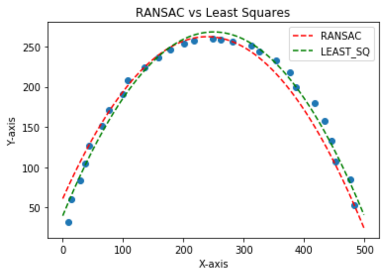




Step 5) Comparison between Least square methods and RANSAC.

- Although for smoother trajectory both were equally accurate, RANSAC performed much better in estimating the trajectory in video 2 with a lot of noise. Here are the results





Video2 Video 1

ANS 3) Part A

To Mathematically compute SVD with the given matrix A, the equation is as follows:

$$A_{mxn} = U_{mxm} * \Sigma_{mxn} * V^T_{nxn}$$

- Here, U is calculated by stacking eigen vectors column wise of the matrix $A * A^T$
- V is calculated by stacking eigen vectors column wise of the matrix $A^T * A$.
- Sigma (middle term) is calculated by taking absolute square root of eigen values diagonally of U if (n >m) , V if (m>n) and either if (m=n).
- For the given A matrix, (working provided in the code)

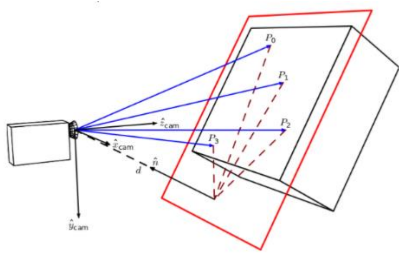
$$A = \begin{bmatrix} -5 & -5 & -1 & 0 & 0 & 0 & 500 & 500 & 100 \\ 0 & 0 & 0 & -5 & -5 & -1 & 500 & 500 & 100 \\ -150 & -5 & -1 & 0 & 0 & 0 & 30000 & 1000 & 200 \\ 0 & 0 & 0 & -150 & -5 & -1 & 12000 & 400 & 80 \\ -150 & -150 & -1 & 0 & 0 & 0 & 33000 & 33000 & 220 \\ 0 & 0 & 0 & -150 & -150 & -1 & 12000 & 12000 & 80 \\ -5 & -150 & -1 & 0 & 0 & 0 & 500 & 15000 & 100 \\ 0 & 0 & 0 & -5 & -150 & -1 & 1000 & 30000 & 200 \end{bmatrix}$$

```
U Matrix is:
[[-7.10167309e-03  1.71750997e-02  1.05764676e-03  1.72784335e-01
  -7.13736127e-01 -2.62681002e-01  6.25366036e-01 -1.71985444e-02]
 [-7.10110327e-03  1.71751083e-02  1.05711843e-03  1.34097547e-01
   6.99599433e-01 -2.66202925e-01  6.49121541e-01  3.03010527e-03]
 [-2.22216603e-01  5.14699028e-01  6.77034899e-01  4.24266390e-01
   1.11716256e-02 -2.10473815e-02 -1.26289423e-01  1.75526783e-01]
 [-8.88821425e-02  2.05882679e-01  2.70813694e-01 -4.21534765e-01
  -1.00282530e-02  6.36357313e-01  3.53913035e-01 -4.10335278e-01]
 [ 9.24254201e-01  3.81546169e-01  1.32926105e-02  7.66954821e-06
   4.88219852e-09  1.10120090e-06 -7.16972204e-06 -2.91262160e-06]
 [-1.70358487e-01  4.11805107e-01  2.53480565e-02 -6.73258907e-01
  -1.79077311e-02 -5.81158591e-01 -9.26135933e-02 -2.88303063e-02]
 [-1.08957001e-01  2.74572788e-01 -3.05764692e-01  3.68951100e-01
   2.39220812e-02 -1.48883411e-01 -1.67304850e-01 -7.95230625e-01]
 [-2.17897722e-01  5.49141260e-01 -6.11531020e-01  7.11858329e-02
  -4.80103685e-03  3.07943979e-01  9.89287677e-02  4.09007828e-01]]
```

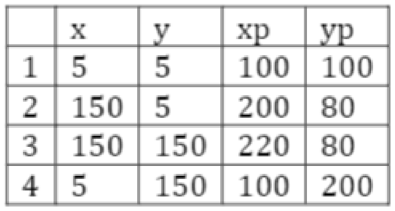
```
V Matrix is:
[[-9.70227994e-04 -1.53441206e-02  3.73766415e-01 -3.85409780e-01
  -4.36589321e-01 -3.14265549e-01  6.49705752e-01  4.32191809e-03
   3.50321685e-03]
 [ 3.95966842e-04 -1.30796275e-02  3.16983519e-01 -3.34815091e-01
   1.71719762e-01  8.57651398e-01  1.48942835e-01  3.09429542e-04
   3.37762918e-03]
 [-3.50206984e-06 -1.19317104e-04  3.00084725e-03 -3.19220317e-03
  -7.62669349e-04  1.30812365e-03 -2.98253565e-03  5.81339961e-01
  -8.13642090e-01]
 [-3.62331930e-04 -5.89391637e-03  1.45983490e-01  5.79617172e-01
  -7.27332046e-01  3.29924173e-01 -6.94210032e-02 -3.19796822e-03
  -2.55301166e-03]
 [ 8.96787175e-04 -8.83095282e-03  2.18049971e-01  6.34256635e-01
   4.83187230e-01 -2.74548199e-02  5.62012749e-01  1.83737760e-03
  -2.92733046e-03]
 [ 3.51990316e-06 -7.23291376e-05  1.85801270e-03  5.53984498e-03
  -1.93190889e-04  4.85359572e-05 -6.43750176e-03  8.13619439e-01
   5.81332762e-01]
 [-7.17971430e-01 -6.95502852e-01 -2.81345451e-02  5.58207376e-04
   9.10606967e-04  2.05567575e-04 -2.70005112e-04  5.49381015e-08
  -3.66202493e-07]
 [ 6.96071112e-01 -7.17390591e-01 -2.88924545e-02 -2.86788998e-04
  -7.69440510e-04 -5.04365526e-04 -2.12885032e-04  7.43558375e-07
  -2.00680613e-06]
 [-7.08343819e-06  3.32755263e-02 -8.30255601e-01 -3.28532854e-02
  -1.31978195e-01  2.36781558e-01  4.84737462e-01  5.90597173e-03
   9.16829102e-06]]

Sigma Matrix is:
[[5.72904485e+04 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 2.58926320e+04 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.14870541e+03 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 2.65159651e+02
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  1.63859703e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 1.59933295e+02 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 9.18281454e+01 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 3.18717390e-01
  0.00000000e+00]]
```

Part B

- Homography is the matrix which relates two separate image plane.

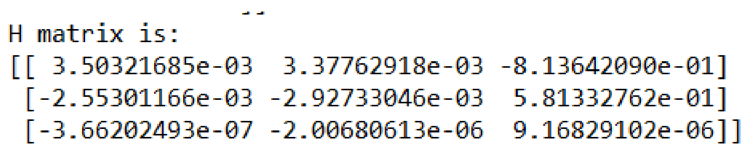


- To calculate homography of the given A matrix with the given data points as following



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

- SVD is extracted of matrix A by the method discussed above. A is 8*9 matrix, hence rank is one less than 9. Thus, last row of column V (9*9 because it's AT*A) is take as the elements of homography and is reshaped to 3*3. (Working shown in the code)



```
H matrix is:
[[ 3.50321685e-03  3.37762918e-03 -8.13642090e-01]
 [-2.55301166e-03 -2.92733046e-03  5.81332762e-01]
 [-3.66202493e-07 -2.00680613e-06  9.16829102e-06]]
```

REFERENCES:

1. http://people.duke.edu/~hpgavin/SystemID/References/Markovsky+VanHuffel-SP-2007.pdf
2. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3516195/
3. https://www.cse.iitd.ac.in/~suban/vision/geometry/node24.html
4. Lecture slides
5. https://www.youtube.com/watch?v=Cu1f6vpEilg