Homework 1

Gokul Hari
Directory ID: hgokul@umd.edu

I. Problem 1

Question

Assume that you have a camera with a resolution of 9MP where the camera sensor is square shaped with a width of 14mm. It is also given that the focal length of the camera is 15mm.
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. To compute the field of view of the camera, we consider the equation 1

$$\phi = \arctan\left(\frac{h}{2f}\right) \tag{1}$$

where ϕ is half field of view, h is width of the sensor and f is the focal length. Substituting these values we obtain,

$$\phi = \arctan\left(\frac{14}{30}\right) = 25.017^o \tag{2}$$

$$Angular\ Field\ of\ view\ (AFOV) = 2\phi = 50.034^o \tag{3}$$

The question asks for field of view in horizontal and vertical directions. Field of view can be calculated from angular field of view using the equation 4, referred from [1]

$$Field\ of\ View = 2 \times D \times \tan\left(\frac{AFOV}{2}\right) \tag{4}$$

where D is the distance betweem camera and image plane. Since D was not given, we assume D as 20 meters from the camera, for which the **Field of view is 18.6667 meters**

2. To compute the object's image area occupied in the 9 MP sensor, in terms of pixels, we first find the sensor area occupied in terms of mm. Figure 1 shows the image formation of a distant object y. The image formed in the sensor is given as $y^{'}$. The distance between the lens and the object is denoted as D and the distance between the sensor and object is given as $D^{'}$. The focal length is given as f.

In this figure 1, we notice the similarity of green triangles and arrive the relationship.

$$\frac{y^{'}}{y} = \frac{D^{'}}{D} \tag{5}$$

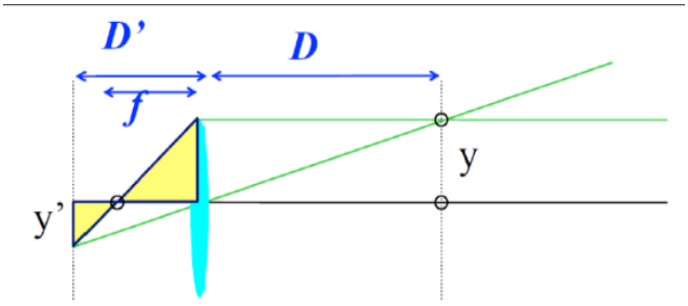

Fig. 1. to compute occupied object area

We need to calculate $y^{'}$ to obtain the height of the image formed in the sensor (in mm). To calculate that, we need to find the $D^{'}$ of the image. We also notice the similarity of green triangles and arrive the relationship.

$$\frac{1}{D} + \frac{1}{D^{'}} = \frac{1}{f} \tag{6}$$

Substituting the known values f = 15 mm, D = 20,000 mm, we get $D^{'}$ **= 15.01125 mm**. With $D^{'}$ known, we calculate the height of image formed in sensor, $y^{'}$ as 0.0375 mm. Since the object is square in shape, the image formed in the sensor is a square spanning an area of $(0.0375)^2$ = 0.00140625 mm^2. Since we need to find the image area in terms of pixels, we compute it using the equation 7

$$\frac{A_{sensor}}{A_{image}} = \frac{Total\ megapixels}{no\ of\ pixels\ occupied\ by\ image} \tag{7}$$

where A_{sensor} = Area of sensor in mm^2, A_{image} = Area of image formed in mm^2.

Substituting the known values A_{sensor} = $14^2 mm^2$, A_{image} = 0.00140625 mm^2, The total number of pixels = 9×10^6, The **number of pixels occupied by the object in the image sensor is 64 pixels**

II. Problem 2

Question

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 video 2. Assuming that the trajectory of the ball follows the equation of a parabola:

- Use Standard Least Squares, TLS and RANSAC methods 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.)
- Briefly explain all the steps of your solution and discuss which would be a better choice of outlier rejection technique for each case.

Answer

Data Processing: The video data was downsized to width = 800 and height = 640 and collected as a set of *n* images. Each image has a white background with a cluster of red pixels forming a circular ball. To track the trajectory of the ball over time t, the location of the ball in every instant of timeframe needs to be found. Since there are no other color pixels in the background, a binary mask of the red ball is found with inverse binary thresholding. The white regions of this mask belong to the red ball and black regions belong to the background. The coordinates of the white (HIGH) pixel cluster is estimated, and the centroid of these pixels is recorded. This process is repeated for *n* frames. Thus, we obtain the (x,y) centre coordinate position of the red ball in the white space for all n images, forming a dataset of shape (n×2). The trajectory of this curve can be estimated three curve fitting algorithms explained further in this section

Least Squares: In the method, with the given data set of x,y coordinates, of shape (n×2), we perform least squares curve fit using the following parabola/polynomial equation:

$$y = ax^2 + bx + c \tag{8}$$

To find the a,b,c coefficients of our polynomial, we need to minimize the equation,

$$E = \sum_{i=1}^{n} (y_i - ax_i^2 - bx_i - c) \tag{9}$$

This equation can be written in matrix form as,

$$E = \|Y - XB\|^2 \tag{10}$$

where,

$$X = \begin{bmatrix} x_1^2 & x_1 & 1 \\ . & . & . \\ . & . & . \\ x_n^2 & x_n & 1 \end{bmatrix} Y = \begin{bmatrix} y_1 \\ . \\ . \\ y_n \end{bmatrix} B = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{11}$$

Equation 10 can be minimized by equating the first derivative of it, with respect to B, to zero. This is given as,

$$\frac{dE}{dB} = 2X^T XB - 2X^T Y = 0 \tag{12}$$

Thus the coefficients to be estimated can be given as,

$$B = (X^T X)^{-1}(X^T Y) \tag{13}$$

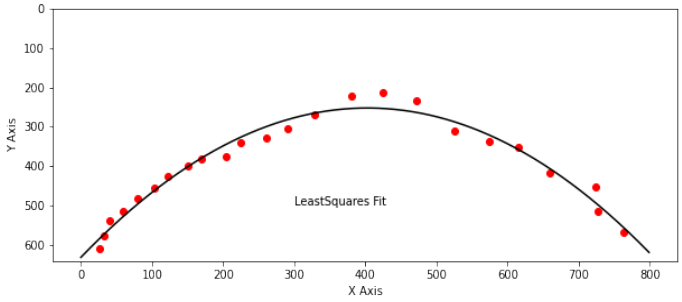

Fig. 2. Least Squares Plot

The resultant curve fit with the estimated coefficients using Least Squares is shown in figure 2

Total Least Squares: In total least squares we consider the presence of erroneous values/outliers in both the dependant and independant variables, in order to obtain a better fit. We perform Total least squares curve fit using the following parabola/polynomial equation:

$$ax^2 + bx + cy - d = 0 \tag{14}$$

To find the a,b,c coefficients of our polynomial, we need to minimize the equation,

$$E = \sum_{i=1}^{n} (ax_i^2 + bx_i + cy_i - d) \tag{15}$$

In order to estimate d, we find $\frac{\partial E}{\partial d}$ with respect to d and equate to 0, which results in,

$$d = \frac{a}{n}\sum_{i=1}^{n} x_i^2 + \frac{b}{n}\sum_{i=1}^{n} x_i + \frac{c}{n}\sum_{i=1}^{n} y_i \tag{16}$$

This equation can be given as, $d = a\bar{x^2} + b\bar{x} + c\bar{y}$ where $\bar{x}$ is the mean of x. Now equation 15 can be written as

$$E = \sum_{i=1}^{n} (a(x_i^2 - \bar{x_i^2}) + b(x_i - \bar{x_i}) + c(y_i - \bar{y_i}))^2 \tag{17}$$

With *n* data points given, the total least squares equation can be written in matrix form as,

$$E = \|UB\|^2 \tag{18}$$

where,

$$U = \begin{bmatrix} x_1^2 - \bar{x_1^2} & x_1 - \bar{x_1} & y_1 - \bar{y_1} \\ . & . & . \\ . & . & . \\ x_n^2 - \bar{x_n^2} & x_n - \bar{x_n} & y_n - \bar{y_n} \end{bmatrix} B = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{19}$$

Equation 18 can be minimized by equating the first derivative of it, with respect to B, to zero. That is,

$$\frac{\partial E}{\partial B} = 2U^T UB = 0 \tag{20}$$

Thus the homogenous set of equations $U^T UB = 0$ can be solved by performing singular value decomposition on the

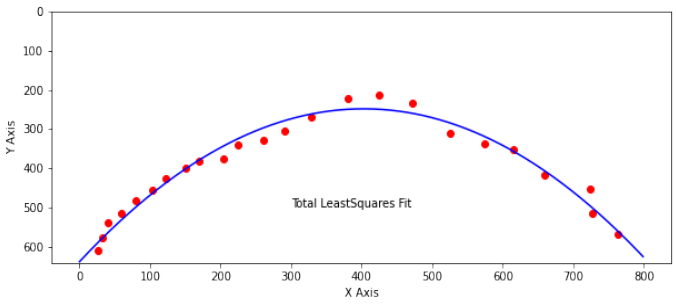

Fig. 3. Total Least Squares Plot

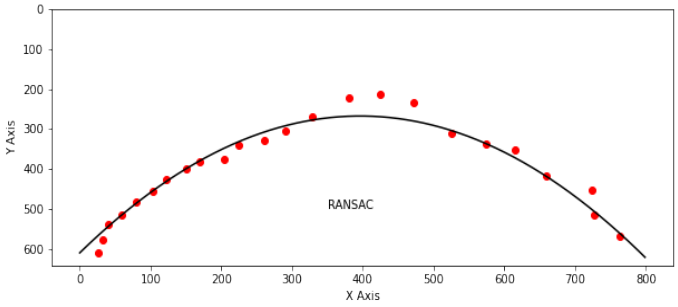

Fig. 4. Ransac with Least Squares Plot

matrix U^TU and the coefficient matrix B corresponds to the right singular vector that corresponds to the smallest eigen value.

The resultant curve fit with the estimated coefficients using Total Least Squares is shown in figure 3

RANSAC: In RANSAC, we pick three random points (x_r, y_r) and perform a Least Squares Fit to obtain a curve y_{pred} of polynomial coefficients B. We compute an error vector such that $Error = |y - y_{pred})|$ where y is the y-axis coordinates in the original distribution of data points (x, Y). The number of inlier data points present with the fit y_{pred} is estimated using the error vector by comparing the error values with a predetermined threshold. This procedure is repeated for n_{iter} times and the best set of coefficients that produce a curve with maximum number of inliers is obtained.

RANSAC Algorithm is given in 1

Algorithm 1 RANSAC Algorithm

procedure RANSAC(x, Y)
  $n\_iter \leftarrow total\ iteration\ for\ RANSAC$
  $s \leftarrow 3\ (min\ no.\ of\ data\ points\ to\ fit\ the\ curve)$
  $thresh \leftarrow threshold\ value$
  $B\_best \leftarrow final\ coefficients\ (initially\ a, b, c = 0, 0, 0)$
  $max\_inliers, inlierCount \leftarrow 0, 0$
  for $iter$ **in range** n_iter **do**
    $x_r, y_r \leftarrow pick\ random\ s\ points\ from\ x, Y$
    $B_{a,b,c} \leftarrow findLeastSquares(x_r, y_r)$
    $y\_pred \leftarrow B_a x^2 + B_b x + B_c$
    $Error \leftarrow |y - y_{pred}|$
    for e **in** $Error$ **do**
      if $e < thresh$ **do**
        inlierCount += 1
    end for
    if $inlierCount > max\_inliers$:
    $B\_best \leftarrow B_{a,b,c}$
    $max\_inliers \leftarrow inlierCount$
  end for
  return B_best
end procedure

The resultant curve fit with the estimated coefficients using Total Least Squares is shown in figure 4

Inference: Due to the representation of the location of the red ball with centroids, there has not been a notable compromise in fitting the curve since the outliers in the data are small. Thus the resultant curves of all three curve fitting procedures were found to be almost the same.

TABLE I
DATA 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

III. Problem 3

Question

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 a system of equations forming Ax = 0, which is given in equation (figure) 5 The given dataset of point correspondences to be fit in equation(figure) 5 is shown in table I

Answer

To obtain the homography matrix, we first substitute the data from table I in the equation (figure) 5 to obtain the A matrix as shown in figure 6

$$\begin{bmatrix} -x1 & -y1 & -1 & 0 & 0 & 0 & x1*xp1 & y1*xp1 & xp1 \\ 0 & 0 & 0 & -x1 & -y1 & -1 & x1*yp1 & y1*yp1 & yp1 \\ -x2 & -y2 & -1 & 0 & 0 & 0 & x2*xp2 & y2*xp2 & xp2 \\ 0 & 0 & 0 & -x2 & -y2 & -1 & x2*yp2 & y2*yp2 & yp2 \\ -x3 & -y3 & -1 & 0 & 0 & 0 & x3*xp3 & y3*xp3 & xp3 \\ 0 & 0 & 0 & -x3 & -y3 & -1 & x3*yp3 & y3*yp3 & yp3 \\ -x4 & -y4 & -1 & 0 & 0 & 0 & x4*xp4 & y4*xp4 & xp4 \\ 0 & 0 & 0 & -x4 & -y4 & -1 & x4*yp4 & y4*yp4 & yp4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0$$

Fig. 5. Homography equation

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

Fig. 6. A matrix

$$\begin{bmatrix} 6.96774e+00 & -6.45161e-01 & 8.06451e+01 \\ 2.32258e+00 & -5.16129e-01 & 1.03225e+02 \\ 3.09677e-02 & -6.45161e-03 & 1.00000e+00 \end{bmatrix}$$

Fig. 7. H matrix

To solve this homography equation, we can perform singular value decomposition of A matrix and the right singular column vector corresponding to the smallest eigen value is the resultant x. This vector is of shape (9×1). The algorithm to perform singular value decomposition for any given A matrix of shape (m×n) where m!=n is explained as follows, referred from [2] and it's mathematical derivation is shown in [3].

- Compute eigen values and eigen vectors of AA^T
- Sort the eigen vectors in descending order of the corresponding eigen values.
- The sorted eigen vector of AA^T is U matrix (left singular vector)
- Compute eigen values and eigen vectors of A^TA
- Sort the eigen vectors in descending order of the corresponding eigen values.
- The sorted eigen vector of A^TA is V matrix (right singular vector)
- The sorted common set of eigen values of A^TA or AA^T is reshaped in a diagonal matrix S of shape m×n. Note: if there are n eigen values and m rows in the diagonal matrix S, then all rows below the nth row is zero

The steps involved in solving SVD to find the U, sigma and V matrices is shown in Figure 8

The column vector x is divided by last element h_{33} and is reshaped to (3×3), which is our final homography matrix as shown in figure 7. This homography matrix was also cross verified with the opencv homography estimation function.

```
Printing all steps involved in SVD:

####### U Matrix #######

U = Eigen Vectors of AAT (sorted in descending order) :
 [[ 0.012  0.    -0.052 -0.466 -0.261 -0.066 -0.761 -0.359]
 [ 0.012  0.    -0.087 -0.459 -0.25  -0.087 -0.017  0.843]
 [ 0.359  0.655  0.013 -0.465  0.17   0.293  0.287 -0.171]
 [ 0.143  0.262 -0.445  0.136 -0.501 -0.588  0.256 -0.178]
 [ 0.775  0.023  0.408  0.285  0.032 -0.234 -0.259  0.167]
 [ 0.282  0.008 -0.692  0.316  0.012  0.503 -0.259  0.147]
 [ 0.185 -0.317  0.249 -0.034 -0.698  0.467  0.275 -0.148]
 [ 0.369 -0.634 -0.289 -0.393  0.319 -0.176  0.251 -0.168]]


####### V Matrix #######

V = Eigen Vectors of ATA (sorted in descending order) :
 [[ 0.003  0.003 -0.246 -0.159 -0.175  0.177  0.914 -0.12   0.053]
 [ 0.002 -0.001 -0.377  0.177  0.69   0.59  -0.053 -0.002 -0.005]
 [ 0.     0.    -0.002 -0.004  0.005  0.008  0.066  0.786  0.615]
 [ 0.001  0.001  0.661  0.341  0.502 -0.232  0.372 -0.043  0.018]
 [ 0.002 -0.003  0.574 -0.071 -0.315  0.75  -0.062  0.005 -0.004]
 [ 0.    -0.     0.006 -0.002  0.003 -0.006 -0.122 -0.605  0.787]
 [-0.696 -0.718 -0.    -0.004  0.003 -0.     0.004 -0.001  0.   ]
 [-0.718  0.696  0.002 -0.004  0.002  0.004 -0.001  0.    -0.   ]
 [-0.006  0.    -0.173  0.907 -0.378  0.062  0.025 -0.002  0.008]]


####### Sigma Matrix #######

Eigen Values of AAT :
[ 3.6258335e+09  1.0128001e+09  6.8063438e+04  3.4670289e+04
 2.1201779e+04  3.6888501e+03  9.1199999e+00 -6.4200001e+00]


Eigen Values of ATA :
[ 3.62583347e+09  1.01280006e+09  6.80652031e+04  3.46776211e+04
 2.12012402e+04  3.70648999e+03  1.51999998e+01  6.60000026e-01
-0.00000000e+00]


S = sigma matrix diagonal values :
 [6.021490e+04 3.182452e+04 2.608900e+02 1.862000e+02 1.456100e+02
6.074000e+01 3.020000e+00 2.530000e+00]
```

Fig. 8. Steps to solve SVD

REFERENCES

[1] https://www.edmundoptics.com/knowledge-center/application-notes/imaging/understanding-focal-length-and-field-of-view/

[2] https://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm

[3] https://math.mit.edu/classes/18.095/2016IAP/lec2/SVD_Notes.pdf