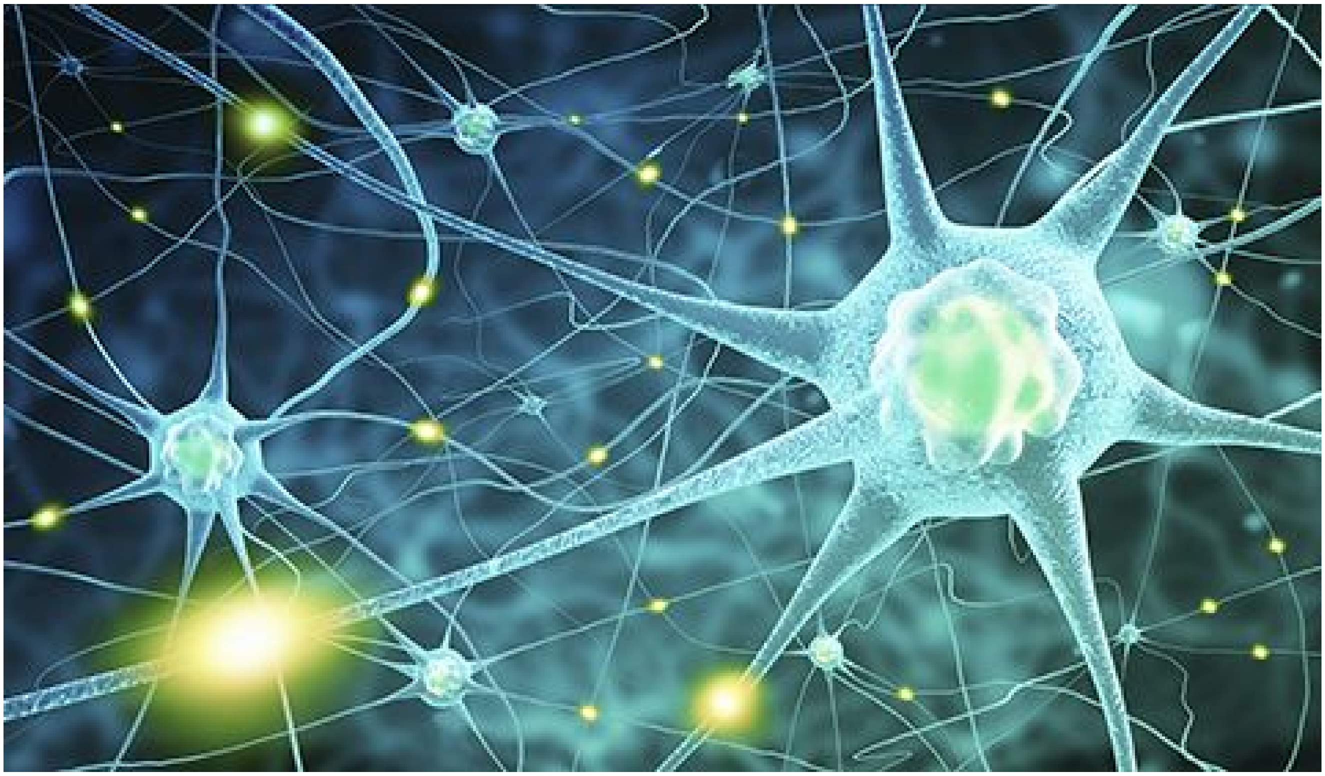
Assignment 4

Computer Vision - CS512



Akshay R

A20442409

Problem Statement:

1. WRITE A PROGRAM TO EXTRACT FEATURE POINTS FROM THE CALIBRATION TARGET AND SHOW THEM ON THE IMAGE USING OPENCV FUNCTIONS.

2. WRITE A PROGRAM THAT DOES THE CALIBRATION PROCESS FROM 3 TEXT FILES WITH 2D POINTS AND ITS CORRESPONDENT 3D POINTS. THE PROGRAM SHOULD DISPLAY THE INTRINSIC AND EXTRINSIC PARAMETERS AS WELL AS THE MENA SQUARE ERROR BETWEEN THE KNOWN AND THE COMPUTED POSITION OF THE IMAGE POINTS.

3. IMPLEMENT THE RANSAC ALGORITHM FOR ROBUST ESTIMATION

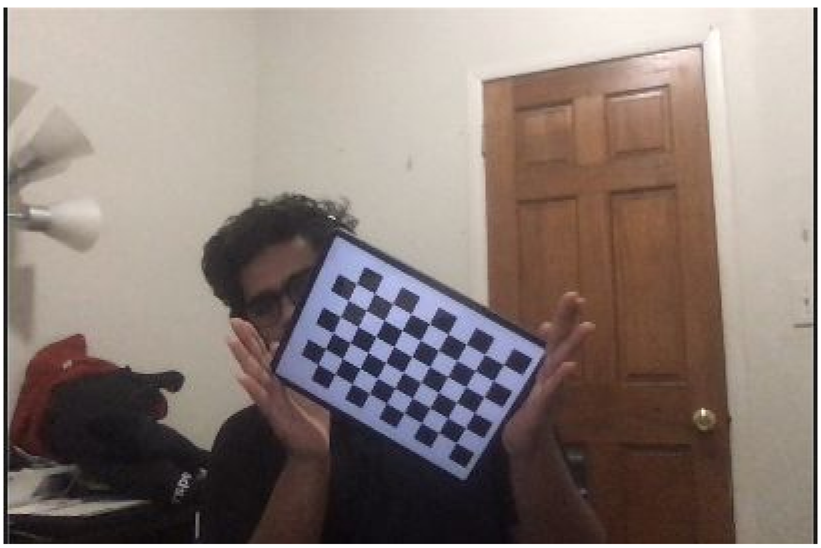
Proposed Solution:

I have chosen to perform planar camera calibration. First I plan to use three images that are taken from the camera of my laptop and to find the 3D and 2D points from the images using findchessboardcorners function. I then plan to store it in a file to perform camera calibration for the camera of my laptop.

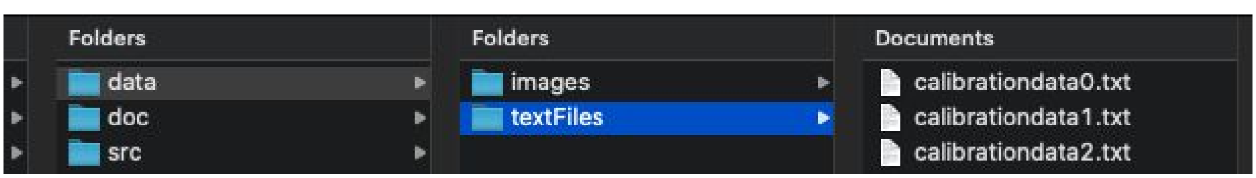
The algorithm to perform calibration is tested on the data provided by the professor. Noise images are then used to run a RANSAC algorithm.

Implementation and methodology:

I use the following three images to obtain my 3D and 2D points:



I then process these images by using openCV functions like findchessboardcorners to obtain the 3d and 2d points corresponding to these three images and store them in the textFiles folder present inside the data folder.



The ipynb notebook A4Q1.ipynb performs these tasks.

Once we have the calibration I have moved on to calculate the camera parameters:

First I calculate the A matrix comprising the world and image points:

$$\overset{A}{\underbrace{\begin{bmatrix} P_1^{*T} & 0 & -x_1 P_1^{*T} \\ 0 & P_1^{*T} & -y_1 P_1^{*T} \\ \vdots & \vdots & \vdots \\ P_m^{*T} & 0 & -x_m P_m^{*T} \\ 0 & P_m^{*T} & -y_m P_m^{*T} \end{bmatrix}}_{2m \times 9}} \overset{X}{\underbrace{\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}}_{9 \times 1}} \overset{=}{=} \overset{0}{\underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{2m \times 1}}$$

SVD is performed on A to obtain the h matrix.

This process is performed on all three data files to obtain the H matrix:

$$H = [h_1, h_2, h_3] = \begin{bmatrix} h_{11} & h_{21} & h_{31} \\ h_{12} & h_{22} & h_{32} \\ h_{13} & h_{23} & h_{33} \end{bmatrix}$$

I have the calculated the V matrix using the formula:

$$V_{ij} = [h_{i1}h_{j1},\ h_{i1}h_{j2} + h_{i2}h_{j1},\ h_{i2}h_{j2},\ h_{i3}h_{j1} + h_{i1}h_{j3},\ h_{i3}h_{j2} + h_{i2}h_{j3},\ h_{i3}h_{j3}]^T$$

all equations needed to solve for S

One image corresponds to two rows in the V matrix. We then perform SVD on the obtained V matrix to calculate the S matrix. The obtained values of the s matrix are then used to calculate the intrinsic and extrinsic parameters of the camera.

$$\begin{bmatrix} V_{12}^T \\ V_{11}^T - V_{22}^T \\ \vdots \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} \begin{bmatrix} s_{11} \\ s_{12} \\ s_{22} \\ s_{13} \\ s_{23} \\ s_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \end{bmatrix}$$

$$K^* = \begin{bmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$|\alpha| = 1 / |(K^*)^{-1} h_1|$$

$$\text{sign}(\alpha) = \text{sign}(((K^*)^{-1} h_3)_z)$$

$$\alpha = |\alpha| \, \text{sign}(\alpha)$$

$$r_1 = \alpha (K^*)^{-1} h_1$$

$$r_2 = \alpha (K^*)^{-1} h_2$$

$$r_3 = r_1 \times r_2$$

$$T^* = \alpha (K^*)^{-1} h_3$$

$$c_1 = (s_{12} s_{13} - s_{11} s_{23})$$

$$c_2 = (s_{11} s_{12} - s_{12}^2)$$

$$v_0 = c_1 / c_2$$

$$\lambda = s_{33} - (s_{13}^2 + v_0 c_1) / s_{11}$$

$$\alpha_u = \sqrt{\lambda / s_{11}}$$

$$\alpha_v = \sqrt{\lambda s_{11} / c_2}$$

$$s = - s_{12} \alpha_u^2 \alpha_v / \lambda$$

$$u_0 = s v_0 / \alpha_u - s_{13} \alpha_u^2 / \lambda$$

Once all the intrinsic and extrinsic parameters are found we then validate our algorithm using the data provided by the professor.

RANSAC is performed using the two noisy image datasets given by the professor. The parameters of RANSAC are estimated using the following formula:

Estimating RANSAC parameters:

probability that all k experiments failed:

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

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

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

large $p \rightarrow$ large k

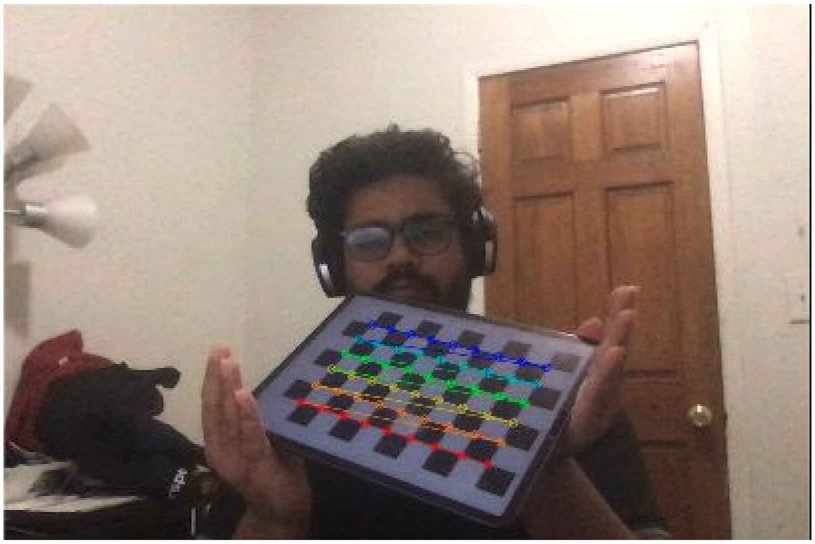
small $w \rightarrow$ large k

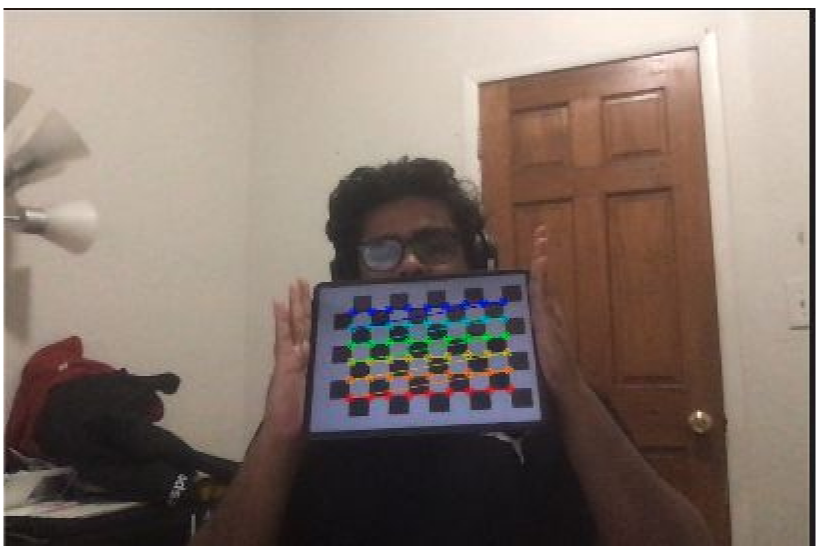
$$w = \frac{\#\ \text{inliers}}{\#\ \text{points}}$$

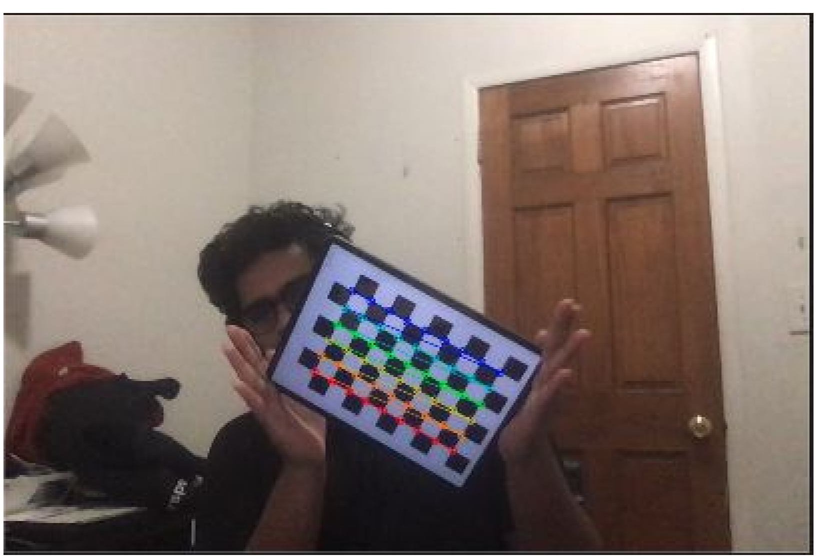
update w, k every iteration but set upperbound for k.

Results and discussion:

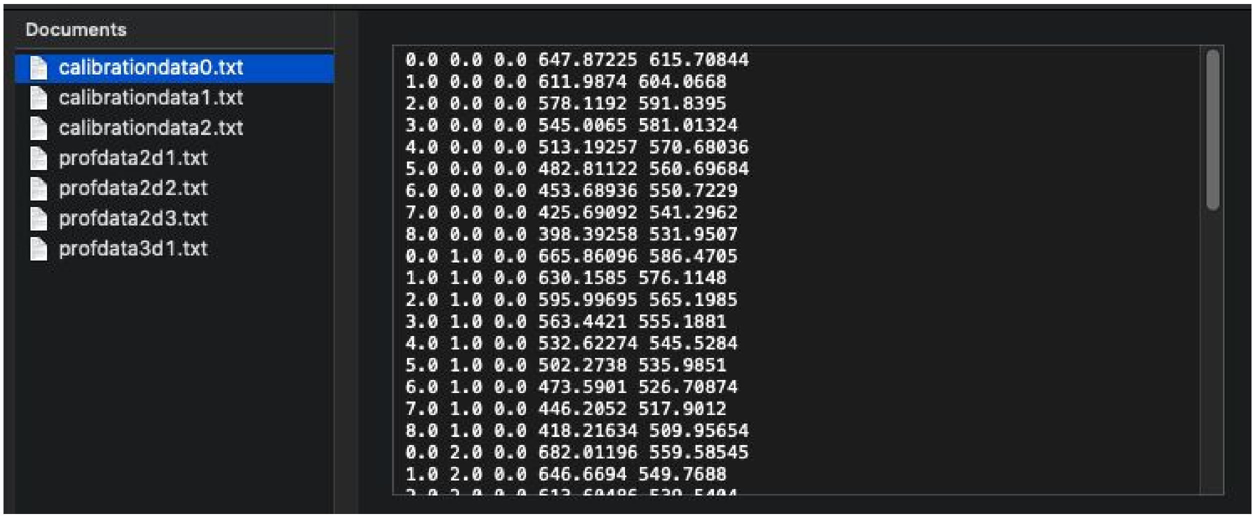
The output of the findchessboardcorners are as follows:







The output points are of the following format:



The first three values are the 3D world points (z=0) and the next two are the corresponding 2D image points.

The results of the algorithm for the calibrated image is shown below:

```
(u0,v0)            = 544.9279257549949 , 381.0022749464423
(alphaU, alphaV) = 952.3820448477614 , 943.4187001914473
 s                 = 1.8173327102012815
================================================================================
Image1:

Rstar =   -0.8425829027520126 0.5090208855434368 0.17539622362187368
          -0.17527707708332707 -0.5672258657589488 0.8046436664342123
          0.5092465004682093 0.6473268802684418 0.5671545094796189

Tstar =   2.515442007330661
          5.797327548552454
          23.325295309423293
================================================================================
Image2:

Rstar =   -0.9920542601009686 -0.005711627764796298 0.12559657175983852
          0.07205115954902717 -0.8512981142531746 0.519832884341464
          0.10313571361638058 0.524590214545251 0.8449454502641489

Tstar =   4.845560157904968
          4.625042565883247
          34.71938177855337
================================================================================
Image3:

Rstar =   -0.8714819519277838 0.42296064316347437 0.24752225329404032
          -0.48433454577015583 -0.8223568923031072 -0.2987449839612769
          0.07706656368203293 -0.38020421027615886 0.9215236406708125

Tstar =   2.066967086663892
          7.555582207231187
          35.47562726078178
================================================================================
```

The algorithm is validated for professors data and the results are as shown below:

```
(u0,v0)          = 319.9991020450845 , 239.9982648275653
(alphaU, alphaV) = 1304.3493974889295 , 1304.348884301188
 s               = -0.0007131915461616466
==================================================================================
Image1:

Rstar =   -0.4472136265295159 0.8944271632495162 -1.1855703657692729e-07
          0.6666660603069079 0.333333034536825 -0.6666674725845276
          -0.5962854486579267 -0.29814290608529403 -0.7453553083723404

Tstar =   -89.44224549785918
          -199.99892465937654
          849.7068890093235
==================================================================================
Image2:

Rstar =   -0.6000001109855112 0.7999999119224105 -8.072175664008974e-08
          0.4997554536435224 0.3748165734326175 -0.7808693418858565
          -0.6246954084759478 -0.46852169687940654 -0.6246943045563418

Tstar =   -39.99952075385564
          -174.91355252579936
          858.9565971305186
==================================================================================
Image3:

Rstar =   -0.24253555506183952 0.970142518768906 1.1657523960040272e-07
          0.7995555021188924 0.1998888740611215 -0.5663527460041881
          -0.5494429029137161 -0.1373605789191681 -0.8241629477422598

Tstar =   -145.5208899644412
          -199.88790412361178
          865.3725699173315
==================================================================================
```

These results are close to the known parameters for these images.

The results for the mean squared errors are as follows:

For my images:

```
mse1 0.16725956897977698
mse2 0.04516003716317788
mse3 0.025901091309016685
```

Professors data:

```
mse1 1.6037478658105047e-09
mse2 1.5877812740401116e-09
mse3 1.3648170932051296e-09
```

My image seems to contain some noise and hence giving a larger mean squared error.

The results of the ransac algorithm are as shown below:

```
(u0,v0)          = 521.6938283951562 , -1281.6884681111294
(alphaU, alphaV) = 1119.1360284228801 , 206.50920680737661
 s               = 38.15967709849881
================================================================================
Image1:

Rstar =  -0.5866541291456843 1.551102515145404 -0.014964406445051412
         0.1635468268613931 0.06420368239382201 -1.466603071720781
         -0.7931515417496763 -0.40286722393922114 -0.29134324987145055

Tstar =  -605.6843566423203
         6997.861019044271
         1195.2137495802076
================================================================================
Image2:

Rstar =  -0.32278821034625343 0.6329942497562023 -1.839516342028169e-07
         -0.8689131998401571 -0.6516855855751578 -0.32835840575492364
         -0.3752300926165202 -0.2814226538687257 0.7603734829123368

Tstar =  -227.9908877166536
         3138.173728981786
         515.9416239046441
================================================================================
Image3:

Rstar =  -0.1872916339040474 0.9862573343056242 5.444747668836847e-07
         0.8612124452452169 0.21530403358076577 -0.48810712491891406
         -0.47247747885407887 -0.11811924346915637 -0.8897017347538495

Tstar =  -429.90996465781916
         4397.695474903355
         744.1520273276224
================================================================================
```

For Noise1 data

```
(u0,v0)          = 319.99820985741405 , 239.9408216130002
(alphaU, alphaV) = 1304.3785332790249 , 1304.3523334655688
 s               = -0.0012611474717532099
================================================================================
Image1:

Rstar =  -0.44721370629107027 0.8944270699632682 -4.501489263009084e-07
         0.6666526857667029 0.33332645308569875 -0.6666843563972729
         -0.5963003416613525 -0.2981509446408396 -0.7453403669027673

Tstar =  -89.44175386584081
         -199.96539293988798
         849.7264608457813
================================================================================
Image2:

Rstar =  -0.6000003182217494 0.7999997353647477 -8.0725146733851e-08
         0.49973777490242793 0.3748033143702773 -0.7808867580854609
         -0.6247093519944854 -0.46853215452144625 -0.6246721955664429

Tstar =  -39.9990059870284
         -174.87916515353422
         858.9757694776698
================================================================================
Image3:

Rstar =  -0.24253559617968035 0.9701425134221344 1.1658014051341148e-07
         0.7995470537825251 0.19988676198346633 -0.5663654022794266
         -0.5494551786756139 -0.1373636478563625 -0.8241542433419272

Tstar =  -145.52038267290638
         -199.85372992041852
         865.3919042422718
================================================================================
```

For Noise2 data

The RANSAC algorithm doesn't seem to handling the noise 1 data very well and outputs a mean squared error as shown below:

```
mse1 2.997734874218969
mse2 3.1755674523544406e-09
```

Mse1 corresponds to Noise1 data and mse2 corresponds to Noise2 data. RANSAC seems to handle Noise2 data well. This can be directly correlated with the number of inliers found for both images by the RANSAC algorithm. The number of inliers found by the RANSAC algorithm for Noise1 data are 68 whereas the number of inliers found for Noise2 are 111. This means that out of the 121 3D-2D points provided by the professor for both Noise1 and Noise 2 images, Noise1 had 53 points that were outliers, hence, RANSAC did not handle this image efficiently.

References:

- https://www.kite.com/python/answers/how-to-load-a-text-file-to-a-numpy-array-of-strings-in-python#:~:text='d'%5D%5D-,Use%20numpy.,the%20strings%20contained%20in%20fname%20.
- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html
- I have used this github link to reference my RANSAC algorithm:

  https://github.com/chen910/cs512/blob/master/AS4/src/RANSAC.py