Question 1:

Part 1:

I have used
findChessboardCorners() function to find the chess board corners,
cornerSubPix() for refining the pixel coordinates and calibratCamera() to get intrinsic camera matrix, lens distortion, rvec and tvec.

The intrinsic matrix is given below.

intrinsic matrix:

```
[[715.29687544   0.         477.48751653]
 [  0.         537.4991797  268.14595421]
 [  0.           0.           1.        ]]
```

Part 2:

calibrateCamera(): to find the rotation and translation vectors.
Rotation matrices for the images

cv2.Rodrigues(): to find rotation matrices

```
For image :  1
[[-0.12704071 -0.97100582 -0.2025052 ]
 [ 0.9737439  -0.08320793 -0.21189446]
 [ 0.18890071 -0.22410742  0.95608179]]
For image :  2
[[ 0.0904718  -0.90845547  0.40807293]
 [ 0.97539591 -0.00188763 -0.22045239]
 [ 0.20104147  0.4179774   0.88593297]]
For image :  3
[[ 0.04950886 -0.96611099 -0.2533346 ]
 [ 0.98008528 -0.00184385  0.1985685 ]
 [-0.19230632 -0.25812041  0.94677987]]
For image :  4
[[ 0.00853827 -0.99975635 -0.02035519]
 [ 0.99996106  0.00849109  0.00240336]
 [-0.00222993 -0.02037492  0.99978992]]
For image :  5
[[ 0.12423905 -0.94575249  0.30019475]
 [ 0.97648946  0.1702476   0.1322274 ]
 [-0.17616183  0.2767092   0.94466874]]
For image :  6
[[-0.00387368 -0.99997727  0.00551841]
 [ 0.99998859 -0.00385819  0.00281566]
 [-0.00279431  0.00552925  0.99998081]]
For image :  7
[[-0.01315933 -0.99986847  0.00948067]
 [ 0.97602399 -0.01078419  0.21739565]
```

```
 [-0.21726481  0.01211415  0.97603752]]
For image :  8
[[ 0.01129213 -0.95794927  0.28671535]
 [ 0.99925788  0.02137067  0.03204665]
 [-0.03682636  0.28614069  0.95747967]]
For image :  9
[[-0.01870594 -0.97728212 -0.21111549]
 [ 0.99967796 -0.02190305  0.01281544]
 [-0.01714837 -0.21080777  0.97737711]]
For image :  10
[[-0.0567126  -0.9026198  -0.42668627]
 [ 0.98706309  0.01350467 -0.15976258]
 [ 0.14996713 -0.43022682  0.8901768 ]]
For image :  11
[[-0.0012174  -0.91961321 -0.3928232 ]
 [ 0.97291841 -0.09188871  0.21209959]
 [-0.23114561 -0.38192671  0.89482048]]
For image :  12
[[ 0.11498665 -0.88865535  0.44392537]
 [ 0.97911409  0.17682078  0.10034947]
 [-0.16767133  0.42311473  0.890427  ]]
For image :  13
[[-0.01117077 -0.99993072  0.00370938]
 [ 0.97788416 -0.01169912 -0.20881979]
 [ 0.20884872  0.00129466  0.9779471 ]]
For image :  14
[[ 0.01640851 -0.98136832  0.191434  ]
 [ 0.9868831  -0.01485682 -0.16075141]
 [ 0.16060044  0.19156067  0.96825204]]
For image :  15
[[ 0.03425522 -0.94533988  0.32428244]
 [ 0.94916208  0.13236582  0.28560575]
 [-0.31291841  0.2980131   0.90181498]]
For image :  16
[[-0.08706249 -0.95143738 -0.29527452]
 [ 0.96959592 -0.01288426 -0.24437215]
 [ 0.2287004  -0.30757262  0.92363153]]
For image :  17
[[-0.00584655 -0.9920168  -0.12597017]
 [ 0.9573742  -0.04193319  0.28579058]
 [-0.28879139 -0.1189297   0.94997645]]
For image :  18
[[ 0.04851548 -0.94320558  0.32864795]
 [ 0.96715823 -0.03783011 -0.25134407]
 [ 0.24950192  0.33004865  0.9103937 ]]
For image :  19
[[ 0.02959403 -0.94627072  0.3220185 ]
 [ 0.95694361  0.11988972  0.26435845]
 [-0.28876137  0.30033011  0.90907574]]
For image :  20
[[-0.01154688 -0.95630866 -0.29213081]
 [ 0.95849036 -0.09382198  0.26924649]
```

[-0.28489104 -0.27689561 0.91769598]]
For image : 21
[[-0.01029687 -0.99994369 -0.00256839]
 [0.97138441 -0.0093932 -0.23732697]
 [0.23728948 -0.00493862 0.97142643]]
For image : 22
[[-0.02410187 -0.99938587 -0.02543592]
 [0.99959432 -0.02447753 0.0145621]
 [-0.01517576 -0.02507462 0.99957039]]
For image : 23
[[-0.02791459 -0.99959592 0.00536428]
 [0.94200003 -0.02450994 0.3347166]
 [-0.33444987 0.01439663 0.94230357]]
For image : 24
[[-0.01399606 -0.94773247 0.31875895]
 [0.99956689 -0.00500801 0.02899919]
 [-0.02588713 0.31902677 0.94739209]]
For image : 25
[[-0.02576652 -0.94696431 -0.32030403]
 [0.99931588 -0.03290292 0.01688707]
 [-0.02653039 -0.31964978 0.94716427]]

Translation vectors for the images

For image : 1
[[4.4804786]
 [-1.23215097]
 [10.74360978]]
For image : 2
[[3.07101543]
 [-1.43776466]
 [8.56259329]]
For image : 3
[[4.13074271]
 [-1.97798392]
 [10.73332697]]
For image : 4
[[3.46407814]
 [-1.9364719]
 [9.43126268]]
For image : 5
[[2.42557049]
 [-2.10121951]
 [9.38774274]]
For image : 6
[[3.24117788]
 [-1.7649118]
 [8.96358383]]
For image : 7
[[3.62224279]
 [-2.02204913]
 [9.92684873]]

For image : 8
[[3.17921194]
 [-1.373576]
 [9.33357675]]
For image : 9
[[4.43136085]
 [-1.90016664]
 [10.38085173]]
For image : 10
[[3.8410576]
 [-1.65886947]
 [11.19311647]]
For image : 11
[[4.11934104]
 [-1.41918053]
 [11.71069848]]
For image : 12
[[2.12132297]
 [-1.93943594]
 [9.00536141]]
For image : 13
[[3.86321452]
 [-1.77393219]
 [9.48058565]]
For image : 14
[[3.16754169]
 [-1.46140951]
 [9.08492318]]
For image : 15
[[2.36595624]
 [-2.04139749]
 [9.89139603]]
For image : 16
[[3.03860594]
 [-2.00492354]
 [11.60803583]]
For image : 17
[[3.55744361]
 [-1.67078102]
 [10.61415989]]
For image : 18
[[3.28198041]
 [-1.58761645]
 [9.66581742]]
For image : 19
[[2.5592349]
 [-2.15559828]
 [9.73608884]]
For image : 20
[[3.67635176]
 [-1.77322857]
 [11.63690553]]

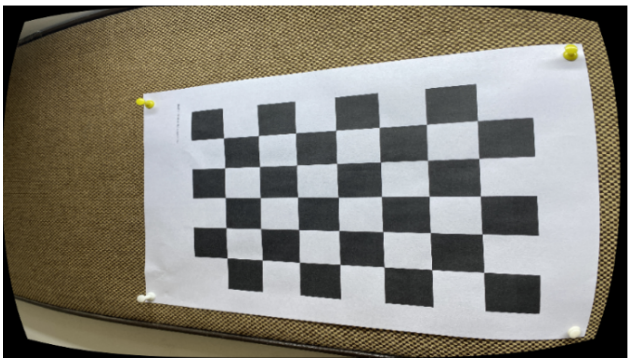
For image : 21
[[3.71402306]
 [-1.51771843]
 [9.48156698]]
For image : 22
[[3.6723898]
 [-1.6182402]
 [10.21373416]]
For image : 23
[[3.62038613]
 [-1.3746773]
 [10.39916766]]
For image : 24
[[2.84408625]
 [-1.61397959]
 [9.34796952]]
For image : 25
[[3.55435529]
 [-1.39135741]
 [11.95467446]]

Part 3:

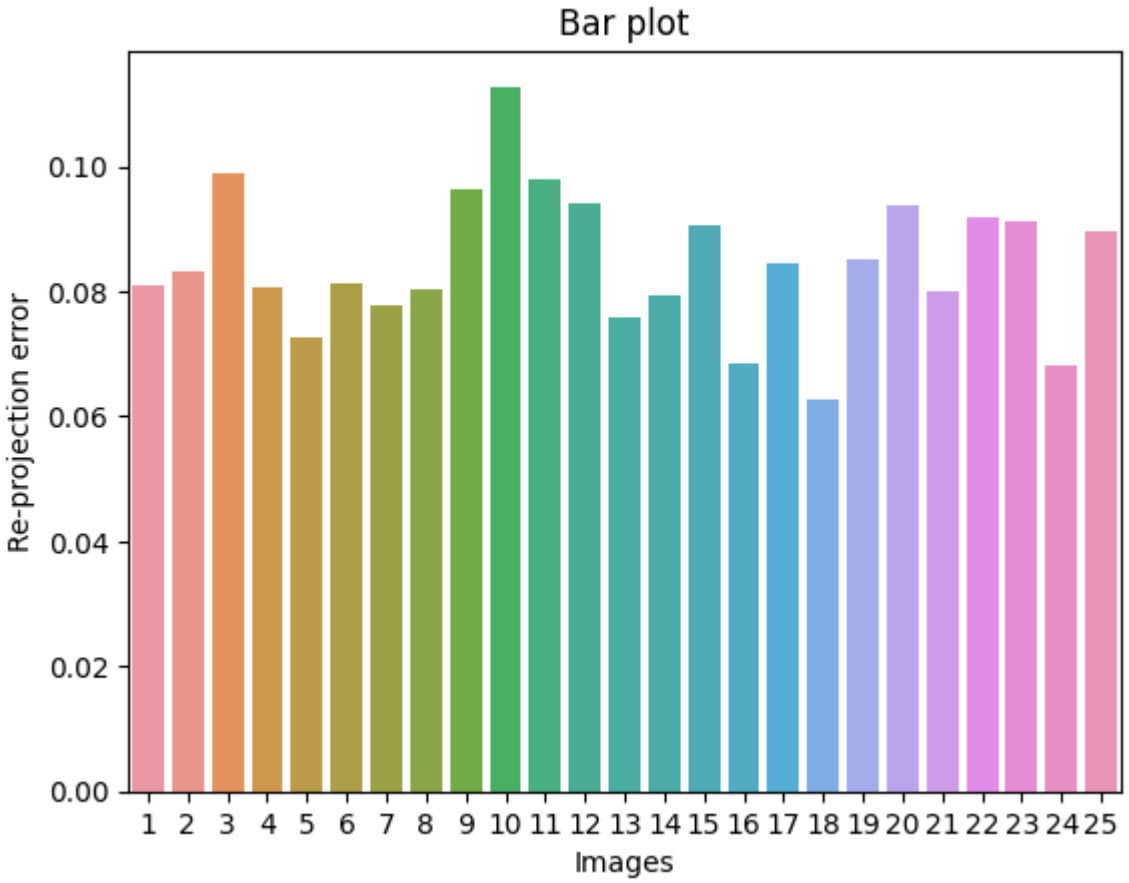
calibrateCamera(): to obtain distortion coefficients

\Distortion matrix:
Below are the radial distortion coefficients:
[[0.02265107 -0.25819242 0.   0.00181977 0.91198486]]

getOptimalNewCameraMatrix(): to refine the matrix
undistort(): to transform an image for radial and tangential distortion

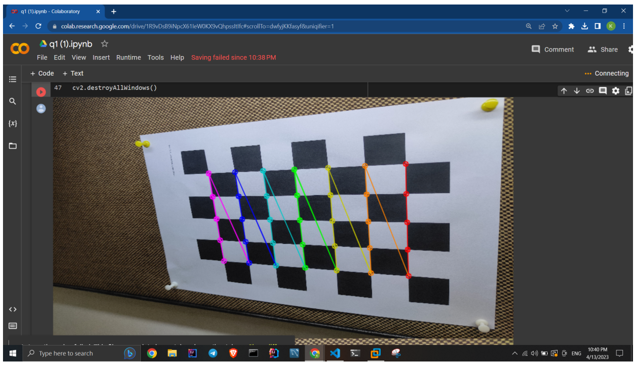


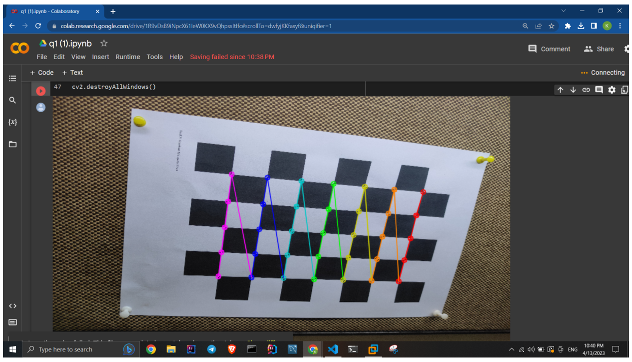
Part 4:
Re-projection error mean : 0.0847430777268711
Re-projection errors:
[0.08101602485305368, 0.08311379060603129, 0.098877481179955, 0.08070031408167916, 0.07279260964644814, 0.08123344260353452, 0.07769547646490807, 0.08020875154210626, 0.09638897534067106, 0.1127998474683328, 0.09808770914341654, 0.09420301667911245, 0.0759098654384246, 0.07931788085903224, 0.09067455255111442, 0.06837205697469244, 0.08444394978156988, 0.0627843894628323, 0.08519940871983742, 0.0939107142213041, 0.0799462779364633, 0.09180949107758692, 0.09125707682071639, 0.06824730533439337, 0.08958653438456166]
Re-projection error standard deviation : 0.011020949764219962

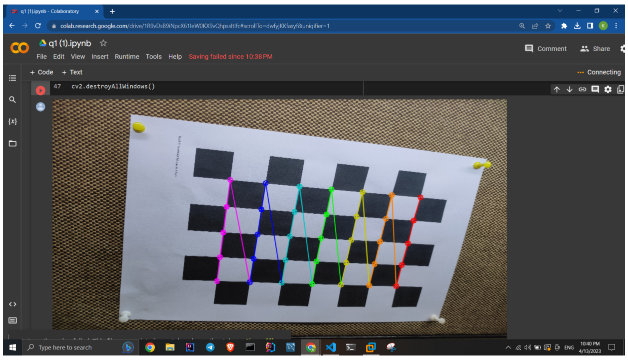


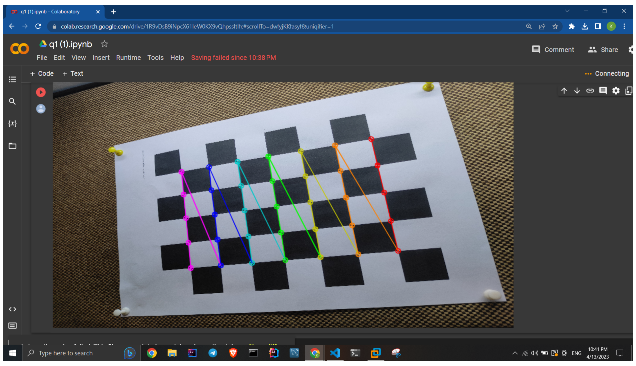
Part 5:

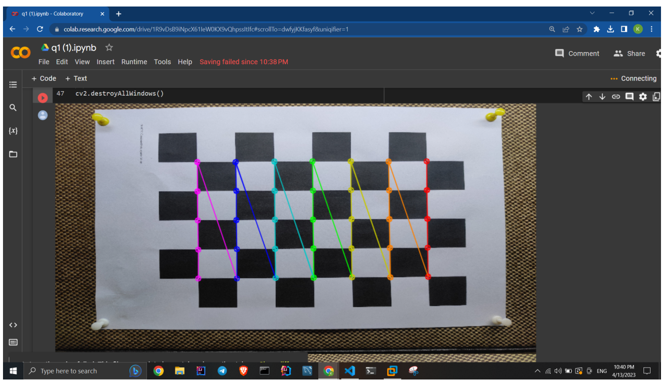
projectPoints(): to compute re-projection error. Where the points are converted to image points.
Parameters: object points, rotation vector, translation vector, camera intrinsic matrix and distortion coefficient matrix.
Output: array of image points and jacobian matrix.
To caluclate the norm between transformation i have used this output images and original images.
Here this reprojection error tells us how the accuracy of the camera calibration process.
Below are the output images:

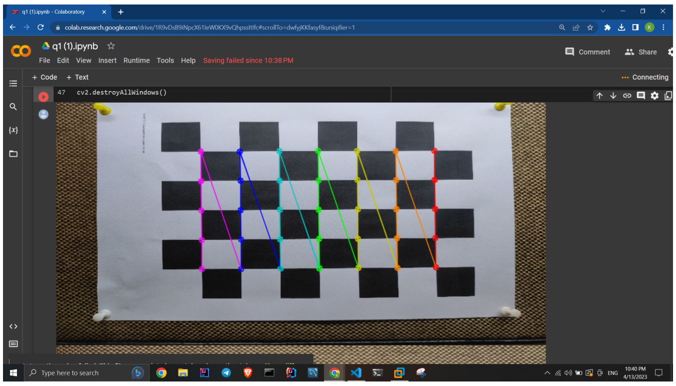

q1 (1).ipynb
File Edit View Insert Runtime Tools Help Saving failed since 10:38 PM
Comment Share
+ Code + Text
Connecting
cv2.destroyAllWindows()

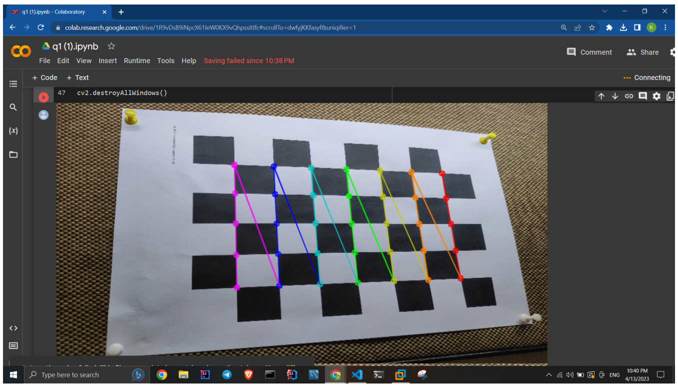

q1 (1).ipynb
File Edit View Insert Runtime Tools Help Saving failed since 10:38 PM
Comment Share
+ Code + Text
Connecting
cv2.destroyAllWindows()

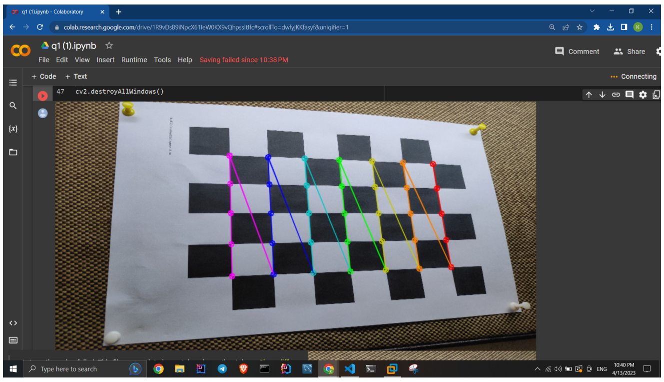

q1 (1).ipynb
File Edit View Insert Runtime Tools Help Saving failed since 10:38 PM
Comment Share
+ Code + Text
Connecting
cv2.destroyAllWindows()

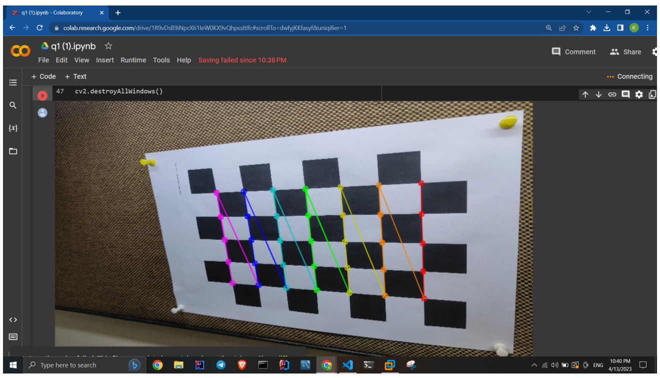

q1 (1).ipynb
File Edit View Insert Runtime Tools Help Saving failed since 10:38 PM
Comment Share
+ Code + Text
Connecting


q1 (1).ipynb
File Edit View Insert Runtime Tools Help Saving failed since 10:38 PM
Comment Share
+ Code + Text
Connecting
cv2.destroyAllWindows()


q1 (1).ipynb
File Edit View Insert Runtime Tools Help Saving failed since 10:38 PM
Comment Share
+ Code + Text
Connecting
47 cv2.destroyAllWindows()
Type here to search


q1 (1).ipynb
File Edit View Insert Runtime Tools Help Saving failed since 10:38 PM
Comment Share
+ Code + Text
Connecting
47 cv2.destroyAllWindows()
Type here to search


q1 (1).ipynb
File Edit View Insert Runtime Tools Help Saving failed since 10:38 PM
Comment Share
+ Code + Text
Connecting
47 cv2.destroyAllWindows()
Type here to search


q1 (1).ipynb
File Edit View Insert Runtime Tools Help Saving failed since 10:38 PM
Comment Share
+ Code + Text
Connecting
47 cv2.destroyAllWindows()
Type here to search

Part 6:

We need to convert the rotation vector for each image with it's corresponding rotation matrix. Then the last column of that matrix will be checkerboard plane normal for that image.

Output:

```
[[-0.2025052 ]
 [-0.21189446]
 [ 0.95608179]]
[[ 0.40807293]
 [-0.22045239]
 [ 0.88593297]]
[[-0.2533346 ]
 [ 0.1985685 ]
 [ 0.94677987]]
[[-0.02035519]
 [ 0.00240336]
 [ 0.99978992]]
[[0.30019475]
 [0.1322274 ]
 [0.94466874]]
[[0.00551841]
 [0.00281566]
 [0.99998081]]
[[0.00948067]
 [0.21739565]
 [0.97603752]]
[[0.28671535]
 [0.03204665]
 [0.95747967]]
[[-0.21111549]
 [ 0.01281544]
 [ 0.97737711]]
[[-0.42668627]
 [-0.15976258]
 [ 0.8901768 ]]
[[-0.3928232 ]
 [ 0.21209959]
 [ 0.89482048]]
[[0.44392537]
 [0.10034947]
 [0.890427  ]]
[[ 0.00370938]
 [-0.20881979]
 [ 0.9779471 ]]
[[ 0.191434  ]
 [-0.16075141]
 [ 0.96825204]]
[[0.32428244]
 [0.28560575]
 [0.90181498]]
```

```
[[-0.29527452]
 [-0.24437215]
 [ 0.92363153]]
[[-0.12597017]
 [ 0.28579058]
 [ 0.94997645]]
[[ 0.32864795]
 [-0.25134407]
 [ 0.9103937 ]]
[[0.3220185 ]
 [0.26435845]
 [0.90907574]]
[[-0.29213081]
 [ 0.26924649]
 [ 0.91769598]]
[[-0.00256839]
 [-0.23732697]
 [ 0.97142643]]
[[-0.02543592]
 [ 0.0145621 ]
 [ 0.99957039]]
[[0.00536428]
 [0.3347166 ]
 [0.94230357]]
[[0.31875895]
 [0.02899919]
 [0.94739209]]
[[-0.32030403]
 [ 0.01688707]
 [ 0.94716427]]
```

---

vecs:

```
(array([[-0.01045495],
       [-0.33506421],
       [ 1.66480887]]), array([[0.50552998],
       [0.16393442],
       [1.4916963 ]]), array([[-0.35931826],
       [-0.04801644],
       [ 1.53124772]]), array([[-0.01779487],
       [-0.01415983],
       [ 1.56222108]]), array([[0.10557413],
       [0.34807797],
       [1.40459921]]), array([[0.00213652],
       [0.00654495],
       [1.57465682]]), array([[-0.16373348],
       [ 0.18085325],
       [ 1.57598101]]), array([[0.20019362],
       [0.25490949],
       [1.54202896]]), array([[-0.17925849],
```

[-0.15548588],
[1.58474998]]), array([[-0.22343463],
[-0.47638215],
[1.5610958]]), array([[-0.49849834],
[-0.13567751],
[1.58818533]]), array([[0.23977149],
[0.45433468],
[1.3875033]]), array([[0.16742567],
[-0.16346135],
[1.57598389]]), array([[0.27939684],
[0.02445218],
[1.560898]]), array([[0.00953798],
[0.48983923],
[1.45637185]]), array([[-0.05263187],
[-0.43635408],
[1.59979167]]), array([[-0.32815887],
[0.13202014],
[1.5806224]]), array([[0.46846363],
[0.06377279],
[1.53929685]]), array([[0.02773727],
[0.47096429],
[1.46754344]]), array([[-0.45661553],
[-0.00605299],
[1.6009148]]), array([[0.18537562],
[-0.19133403],
[1.57252359]]), array([[-0.03162579],
[-0.00818649],
[1.59496877]]), array([[-0.26079729],
[0.27666901],
[1.58080358]]), array([[0.23313009],
[0.27703356],
[1.56527901]]), array([[-0.27412807],
[-0.23929504],
[1.58535394]]))

vecs:

(array([[-0.01045495],
[-0.33506421],
[1.66480887]]), array([[0.50552998],
[0.16393442],
[1.4916963]]), array([[-0.35931826],
[-0.04801644],
[1.53124772]]), array([[-0.01779487],
[-0.01415983],
[1.56222108]]), array([[0.10557413],
[0.34807797],
[1.40459921]]), array([[0.00213652],
[0.00654495],
[1.57465682]]), array([[-0.16373348],
[0.18085325],
[1.57598101]]), array([[0.20019362],
[0.25490949],

[1.54202896]]), array([[-0.17925849],
[-0.15548588],
[1.58474998]]), array([[-0.22343463],
[-0.47638215],
[1.5610958]]), array([[-0.49849834],
[-0.13567751],
[1.58818533]]), array([[0.23977149],
[0.45433468],
[1.3875033]]), array([[0.16742567],
[-0.16346135],
[1.57598389]]), array([[0.27939684],
[0.02445218],
[1.560898]]), array([[0.00953798],
[0.48983923],
[1.45637185]]), array([[-0.05263187],
[-0.43635408],
[1.59979167]]), array([[-0.32815887],
[0.13202014],
[1.5806224]]), array([[0.46846363],
[0.06377279],
[1.53929685]]), array([[0.02773727],
[0.47096429],
[1.46754344]]), array([[-0.45661553],
[-0.00605299],
[1.6009148]]), array([[0.18537562],
[-0.19133403],
[1.57252359]]), array([[-0.03162579],
[-0.00818649],
[1.59496877]]), array([[-0.26079729],
[0.27666901],
[1.58080358]]), array([[0.23313009],
[0.27703356],
[1.56527901]]), array([[-0.27412807],
[-0.23929504],
[1.58535394]]))
\Error estimate:

0.45913760323847386

Question 2:

(i)
Camera offset: negation of the dot product of camera normal and translation vector.
Camera normal and offset are provided in data.
To find lidar normals and lidar offset, i first found the centroid and then taking diff of points from centroid. Then took the svd decomposition to get lidar normals.
Lidar offset: negation of the dot product of lidar normal and translation vector.

Lidar normal [-0.73737765 0.5739658   0.35614247]

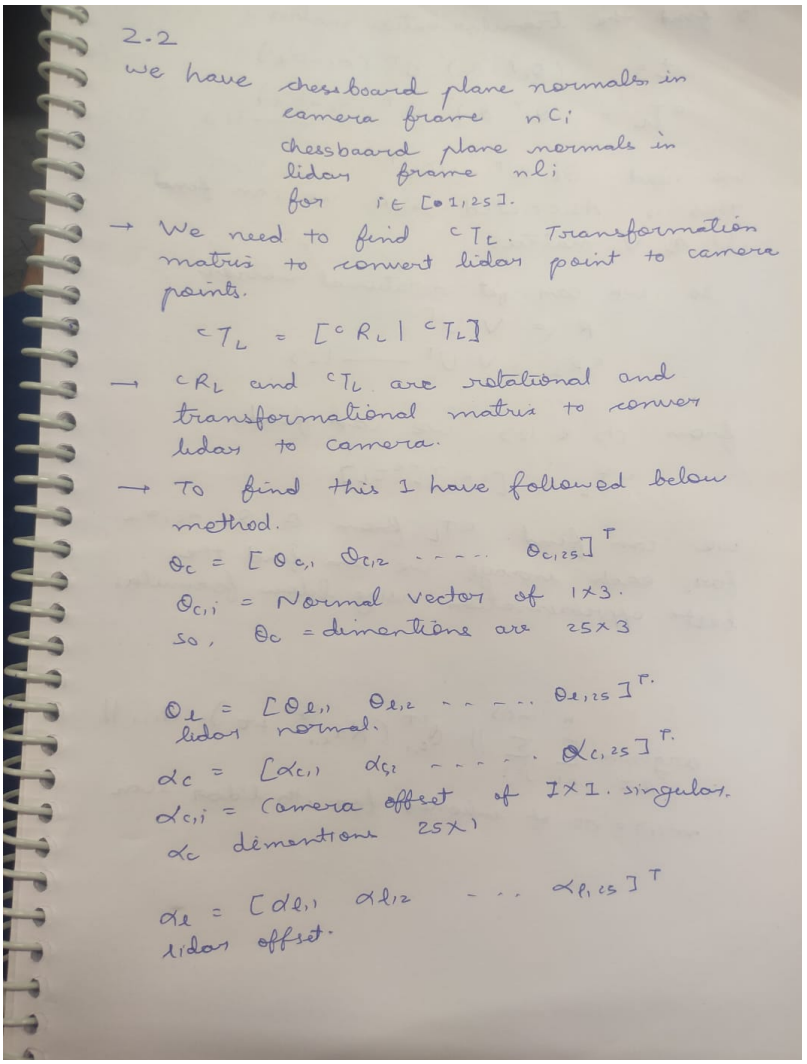
Lidar offset -0.6173963843995338
Lidar normal [-0.43460444 0.89601842 -0.09093933]
Lidar offset 1.6581593500032712
Lidar normal [-0.76477664 0.64422625 0.00944619]
Lidar offset 2.440154424620167
Lidar normal [0.94902101 0.13112718 0.28664401]
Lidar offset -0.5166637933458826
Lidar normal [0.98745566 0.06712949 -0.14291586]
Lidar offset 4.485033369802012
Lidar normal [-0.61872738 0.78293781 -0.06469016]
Lidar offset 2.59786331743304
Lidar normal [-0.97607976 0.21597961 -0.02492227]
Lidar offset 0.12024157412940559
Lidar normal [-0.9493562 -0.22833398 0.21583881]
Lidar offset -0.2863429655511951
Lidar normal [0.95936444 -0.00899235 -0.2820266]
Lidar offset 1.1827325017995012
Lidar normal [0.60371533 -0.79239849 0.08736381]
Lidar offset -0.27591558972753
Lidar normal [-0.95207118 0.1539658 -0.26430097]
Lidar offset 1.503227211795524
Lidar normal [0.75932657 -0.56254883 0.32705042]
Lidar offset -3.0119529682117365
Lidar normal [-0.52528321 0.84450165 0.1043768]
Lidar offset 0.3968984253614216
Lidar normal [0.93288206 -0.2440387   0.26490788]
Lidar offset -1.446075973376584
Lidar normal [0.60407281 -0.77412903 0.18926246]
Lidar offset -2.507059299159562
Lidar normal [-0.9065703   0.30866042 0.28785247]
Lidar offset -1.72362590053854
Lidar normal [-0.40409023 0.88184713 -0.24301591]
Lidar offset 2.5254512198308587
Lidar normal [0.87718352 -0.41582556 0.24007951]
Lidar offset -2.2410852053273342
Lidar normal [-0.47346333 0.7599202   0.44536924]
Lidar offset -1.4118421075881364
Lidar normal [-0.96334688 0.25648046 0.07861659]
Lidar offset -0.6663197888621081
Lidar normal [0.91935323 -0.3607026   0.15710911]
Lidar offset 1.3571817737126528
Lidar normal [0.93809241 -0.22958973 0.25936688]
Lidar offset -1.987745182534 5347
Lidar normal [0.72352642 -0.68812782 -0.05467752]
Lidar offset -0.004849213148022 92
Lidar normal [0.94151039 0.18949708 0.27865578]
Lidar offset 1.2738537349261618
Lidar normal [0.81878735 0.50763718 -0.26812639]
Lidar offset 3.3563638050211697

Camera normals

```
[[-0.43432021 -0.35965743 -0.82584047]
 [-0.81629617  0.09162133 -0.57032105]
 [-0.45613391 -0.00554963 -0.88989385]
 [ 0.29001348  0.28627248 -0.91320329]
 [ 0.27176261 -0.16526395 -0.94806799]
 [-0.65671387  0.05477644 -0.75214788]
 [-0.03246712 -0.00217473 -0.99947044]
 [ 0.39126525 -0.23402562 -0.89002445]
 [ 0.16303408 -0.29304993 -0.94209428]
 [-0.70422432  0.07305109 -0.70620935]
 [ 0.00739864  0.25390736 -0.96720024]
 [-0.42402941  0.31924201 -0.84751613]
 [-0.73875133 -0.10276552 -0.66609738]
 [-0.06913268  0.23786099 -0.96883581]
 [-0.65477401  0.164822   -0.73763454]
 [-0.13917323 -0.2935912  -0.94574575]
 [-0.795691    0.24197359 -0.55526986]
 [-0.24936746  0.22474881 -0.94196807]
 [-0.68682933 -0.43663579 -0.58104618]
 [-0.06925356 -0.07564818 -0.99472675]
 [-0.18954575  0.13129915 -0.97305341]
 [-0.05275962  0.22970552 -0.9718291 ]
 [-0.55917687 -0.07439273 -0.82570391]
 [ 0.34835151  0.25517693 -0.90196228]
 [ 0.65789388 -0.28542777 -0.69692656]]

Camera offset :
[[3.47754136]
 [6.25263754]
 [6.54620464]
 [8.94983673]
 [9.58596338]
 [6.27168249]
 [4.24353332]
 [6.62148444]
 [8.00874141]
 [3.26695946]
 [5.2813932 ]
 [4.65765205]
 [5.77919912]
 [5.25699533]
 [4.82438366]
 [7.4045124 ]
 [5.29235244]
 [7.00994496]
 [4.82780894]
 [7.6113801 ]
 [5.20549855]
 [4.78574898]
 [4.34327318]
 [5.80327345]
 [5.77199038]]
```

(ii)



2.2

we have chessboard plane normals in camera frame nC_i

chessboard plane normals in lidar frame nl_i

for $i \in [0 1, 25]$.

→ We need to find ${}^{c}T_{L}$. Transformation matrix to convert lidar point to camera points.

$${}^{c}T_{L} = [{}^{c}R_{L} \mid {}^{c}T_{L}]$$

→ ${}^{c}R_{L}$ and ${}^{c}T_{L}$ are rotational and transformational matrix to convert lidar to camera.

→ To find this I have followed below method.

$\theta_c = [\theta_{c,1}\ \theta_{c,2} \ldots\ \theta_{c,25}]^T$

$\theta_{c,i}$ = Normal vector of 1×3.

So, θ_c = dimentions are 25×3

$\theta_l = [\theta_{l,1}\ \theta_{l,2} \ldots\ \theta_{l,25}]^T$.

lidar normal.

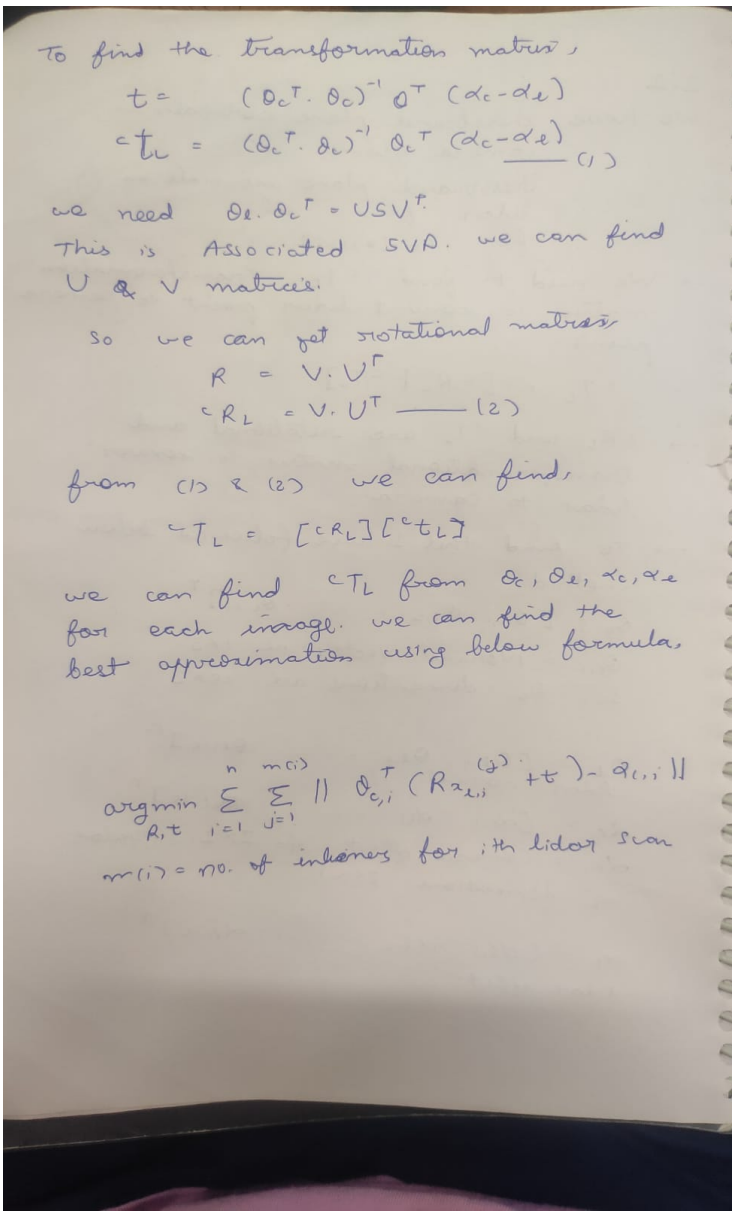
$\alpha_c = [\alpha_{c,1}\ \alpha_{c,2} \ldots\ \alpha_{c,25}]^T$.

$\alpha_{c,i}$ = Camera offset of 1×1. singular.

α_c dimentions 25×1

$\alpha_l = [\alpha_{l,1}\ \alpha_{l,2} \ldots\ \alpha_{l,25}]^T$

lidar offset.



(iii)

I used the above calculation to calculate the transformation matrix.

---

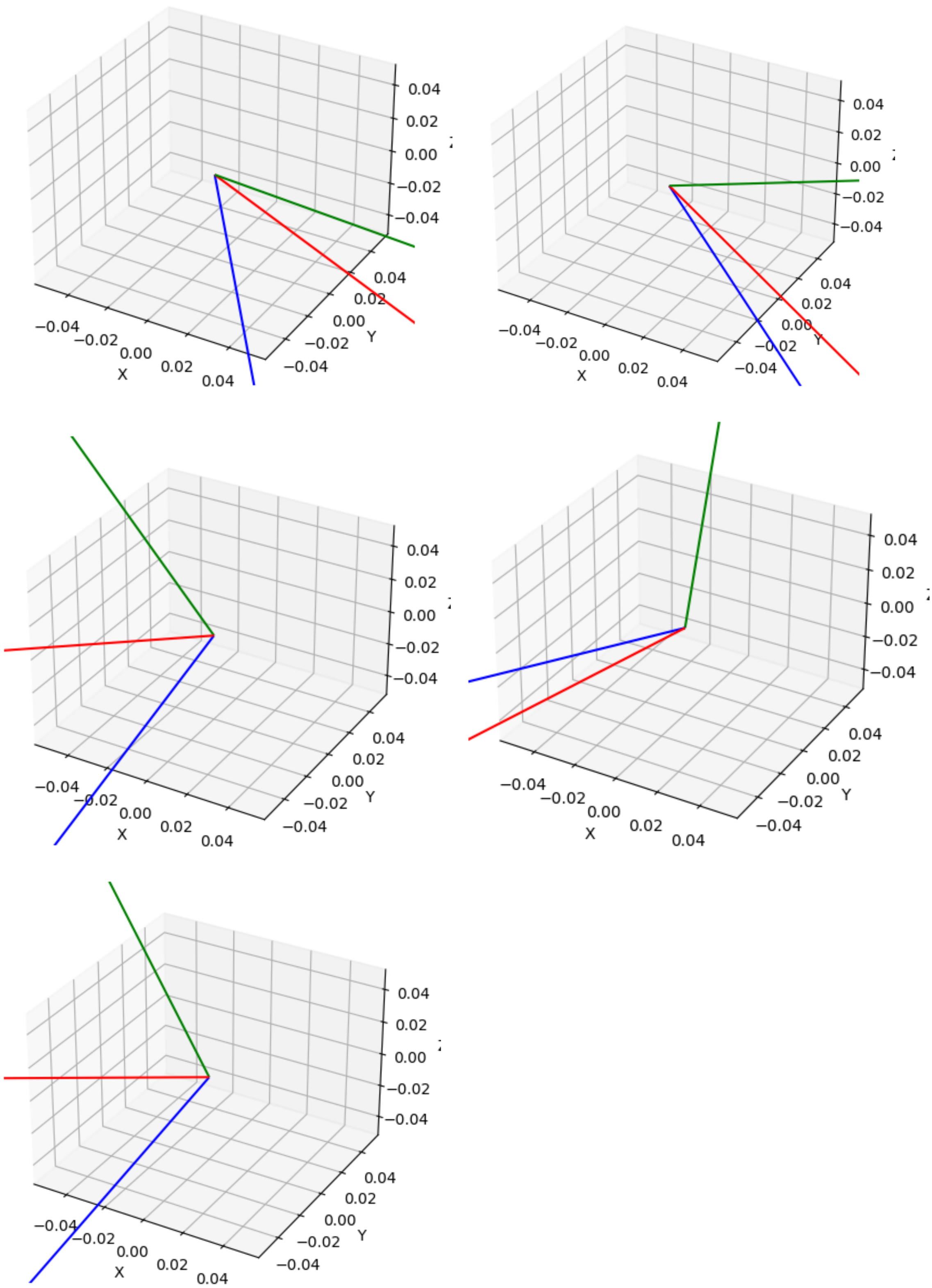
cRL [[0.6984812 -0.22088819 -0.68068526]
[0.30438785 -0.76913844 0.56193781]
[-0.64766662 -0.59969532 -0.46999306]]
ctl [[-0.50528257]
[-0.03824241]
[-6.57232298]]
cTL [[0.6984812 -0.22088819 -0.68068526 -0.50528257]
[0.30438785 -0.76913844 0.56193781 -0.03824241]
[-0.64766662 -0.59969532 -0.46999306 -6.57232298]]

(iv):

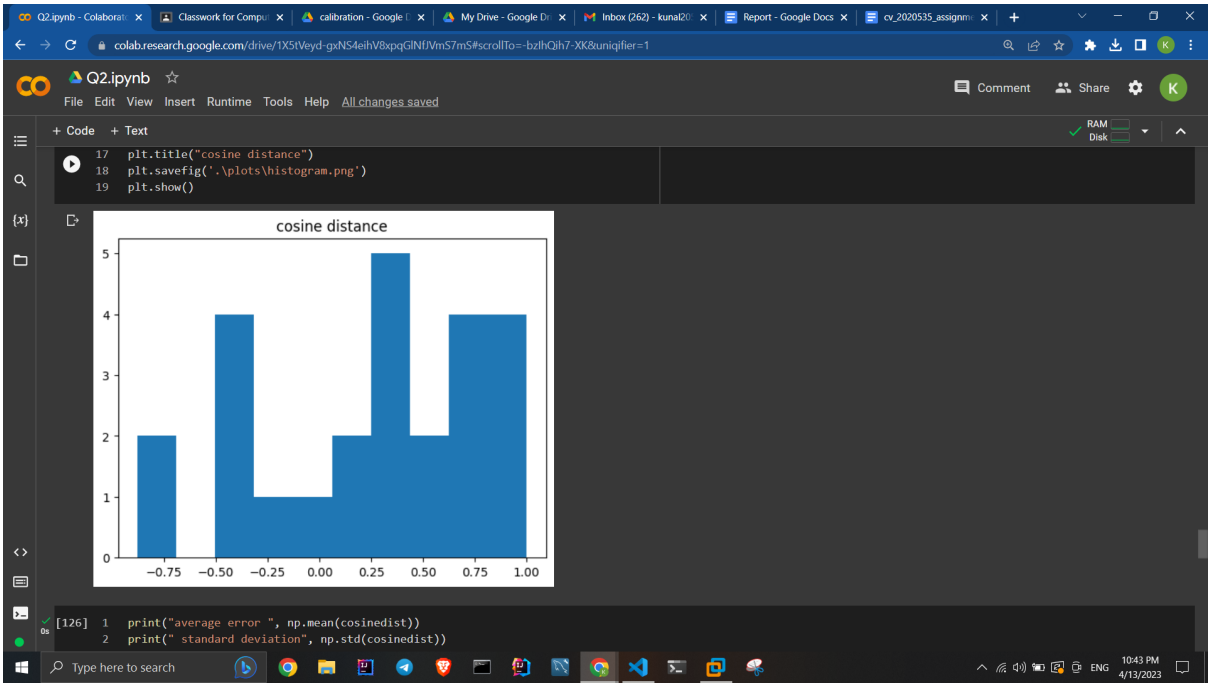
(iv):

Just like the question 1, i have used projectPoints() function to get the image points from lidar points. Then i calculated the output images and jacovian matrix from intrinsic matrix and distortion matrix.

The projected points are inside for some of the image but not all. Some are on boundary and some are even outside.



average error 0.2696878579001874
standard deviation 0.5497982427658992


Q2.ipynb - Colaborat
Classwork for Compu
calibration - Google
My Drive - Google Dr
Inbox (262) - kunal20
Report - Google Docs
cv_2020535_assignm
colab.research.google.com/drive/1X5tVeyd-gxNS4eihV8xpqGlNfJVmS7mS#scrollTo=-bzIhQih7-XK&uniqifier=1
Q2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
17 plt.title("cosine distance")
18 plt.savefig('.\plots\histogram.png')
19 plt.show()
cosine distance
5
4
3
2
1
0
−0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
[126] 1 print("average error ", np.mean(cosinedist))
2 print(" standard deviation", np.std(cosinedist))
Type here to search
ENG
10:43 PM
4/13/2023