QUESTION 1:

Q1.) $V = \begin{pmatrix} 2 \\ 5 \\ 1 \end{pmatrix}$

$\theta_1 = \frac{\pi}{2}$ about Y axis

$\theta_2 = -\frac{\pi}{2}$ about X axis

$(-1, 3, 2)^T$ - translation

$$R_Y(\theta_1) = \begin{bmatrix} \cos \pi/2 & 0 & \sin \pi/2 \\ 0 & 1 & 0 \\ -\sin \pi/2 & 0 & \cos \frac{\pi}{2} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

$$R_x(\theta_2) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos -\frac{\pi}{2} & -\sin -\frac{\pi}{2} \\ 0 & \sin -\frac{\pi}{2} & \cos -\frac{\pi}{2} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

1. Coordinate transformation matrix =

$T \cdot R_x(\theta_2)\, R_y(\theta_1)$

$$\begin{bmatrix} I & \begin{bmatrix} -1 \\ 3 \\ 2 \end{bmatrix} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_x(\theta_2) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_y(\theta_1) & 0 \\ 0 & 1 \end{bmatrix}$$

In part 1, finding out homogeneous transformation matrix

$$= \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 3 \\ 0 & -1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. $P' = HP$ $\qquad P = \begin{bmatrix} 2 \\ 5 \\ 1 \end{bmatrix}$

$$P' = \begin{bmatrix} 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 3 \\ 0 & -1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 5 \\ 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 - 1 \\ -2 + 3 \\ -5 + 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ -3 \\ 1 \end{bmatrix}$$

new coordinator of vector $(2,5,1)^T$ is $(0,1,-3)^T$

Origin mapped to

$$P' = \begin{bmatrix} 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 3 \\ 0 & -1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} -1 \\ 3 \\ 2 \\ 1 \end{bmatrix}$$

Origin mapped to $(-1, 3, 2)^T$

3. dir^n of axis & angle of rotation

$$\theta = \cos^{-1}\left(\frac{\text{trace}(R) - 1}{2}\right)$$

$$= \cos^{-1}\left(\frac{0-1}{2}\right)$$

$$= \cos^{-1}\left(\frac{-1}{2}\right)$$

$$= 120^\circ$$

$$n = \frac{1}{2\sin(120^\circ)} \begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix}$$

$$= \frac{1}{2\sin(120)} \begin{bmatrix} -1 - 0 \\ 1 - 0 \\ -1 - 0 \end{bmatrix}$$

Page No.
Date / /

$$n = \frac{1}{2\sin(120)}\begin{bmatrix}-1\\1\\-1\end{bmatrix}$$

$$= \frac{2}{2\cdot\sqrt{3}}\begin{bmatrix}-1\\1\\-1\end{bmatrix}$$

$$= \left(\frac{-1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{-1}{\sqrt{3}}\right)$$

$$\vec{n} = \frac{-1}{\sqrt{3}}\hat{i} + \frac{1}{\sqrt{3}}\hat{j} - \frac{1}{\sqrt{3}}\hat{k}$$

4. Rodrigue's formula

$$R = I + (\sin\theta)N + (1-\cos\theta)N^2$$

for R_y

$\theta = +\pi/2$ $\quad \hat{n} = (0,1,0)$

$$N = \begin{bmatrix}0 & 0 & 1\\0 & 0 & 0\\-1 & 0 & 0\end{bmatrix}$$

$$R = \begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix} + \sin\frac{\pi}{2}\begin{bmatrix}0 & 0 & 1\\0 & 0 & 0\\-1 & 0 & 0\end{bmatrix} +$$

$$\left(1-\cos\left(\frac{\pi}{2}\right)\right)\begin{bmatrix}0 & 0 & 1\\0 & 0 & 0\\-1 & 0 & 0\end{bmatrix}\begin{bmatrix}0 & 0 & 1\\0 & 0 & 0\\-1 & 0 & 0\end{bmatrix}$$

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} +$$

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

$$R_y = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

for R_x $\quad \theta = \frac{-\pi}{2} \quad \hat{n} = (1,0,0)$

$$N = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \sin\left(-\frac{\pi}{2}\right) \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} +$$

$$\left(1 - \cos\left(-\frac{\pi}{2}\right)\right) \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

Combined $R = R_x R_y$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

QUESTION 2:

Q2.) Rotation R, angle θ, axis $\hat{u}$ (unit)

Tp: $Rx = \cos\theta\, x + \sin\theta\,(u \times x) + (1-\cos\theta)(u^T x)\, u$

We know that given vector x,

$Rx = x'$ where x' point (vector) mapped after rotating vector x.

vector x can be defined as

$$x = x_{\parallel} + x_{\perp} = x_{\text{parallel}} + x_{\text{perpendicular}}$$

$x = \hat{n}\,(\hat{n} \cdot x) - \hat{n} \times (\hat{n} \times x)$ — Assuming x & u to be column vectors, $u^T \rightarrow$ row vector

A.T.Q $\quad x = u\,(u^T \cdot x) - u \times (u \times x)$

The rotation only affects the perpendicular component of x i.e. $x_\perp$

$\cancel{x_\perp' = u\,(u^T \cdot x)}$

$x_\perp' = \cos\theta\, x_\perp + \sin\theta\,(u \times x)$

$x' = u\,(u^T \cdot x) + \cos\theta\, x_\perp + \sin\theta\,(u \times x)$

$x_\perp = x - x_\parallel$

$x' = u\,(u^T \cdot x) + \cos\theta\,(x - u\cdot(u^T \cdot x)) + \sin\theta\,(u \times x)$

Q3

$$x' = (\cos\theta)\, x + \sin\theta\,(u \times x) + (1-\cos\theta)(u^T \cdot x)\, u$$

Thus $Rx = \cos\theta\, x + \sin\theta\,(u \times x) + (1-\cos\theta)(u^T \cdot x)\, u$

Also,
Given Rodrigues formula

$$x' = x + \sin\theta\,(\hat{n} \times x) + (1-\cos\theta)\, \hat{n} \times (\hat{n} \times x)$$

$$\Rightarrow x + \sin\theta\,(u \times x) + (1-\cos\theta)\left[u\,(u^T \cdot x) - x\,(u^T \cdot u)\right]$$

$$\Rightarrow x + \sin\theta\,(u \times x) + (1-\cos\theta)(u^T \cdot x)\, u - x(1-\cos\theta)$$

$$\Rightarrow x\cos\theta + \sin\theta\,(u \times x) + (1-\cos\theta)(u^T \cdot x)\, u$$

[assuming u & x to be column vectors
u^T gives row vector $\Rightarrow$ $u^T \cdot x$ is defined

$\Rightarrow Rx$

QUESTION 3:

Q3.) Given:

$x = K [R \mid t] X$ — 3D point

image point (homogenous) — intrinsic parameter matrix — extrinsic parameters

Cameras: C_1 K_1 x_1 — known-frame of reference world coord.

C_2 K_2 x_2 — pure 3D R applied to first camera's orientation

Tp: $x_1 = Hx_2$ where $H \rightarrow$ invertible 3×3 matrix

Find: H wrt K_1, K_2 & R

For Camera C_1, uses world coordinate frame

$\Rightarrow x_1 = K_1 [I \ \ 0] X$

For C_2,

$x_2 = K_2 [R \ \ 0] X$

Let $K_2 [R \ \ 0] = M$

$$\underbrace{\begin{bmatrix} \alpha_x & s & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{bmatrix}}_{K_1} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{[I \quad 0]}$$

$$K_1[I \; 0]X = \begin{bmatrix} \alpha_x & s & p_x & 0 \\ 0 & \alpha_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x_1\alpha_x + sx_2 + p_x x_3 + 0 \\ 0 + x_2\alpha_y + x_3 p_y + 0 \\ 0 + 0 + x_3 + 0 \end{bmatrix}$$

$$= \begin{bmatrix} a \\ b \\ x_3 \end{bmatrix}$$

$$K_1 [I \quad 0] \; X = [K_1 \quad 0] \; X_{4\times1} =$$

$$x_1 = K_{1_{3\times3}} \; X_{3\times1}$$

x_1 ↓ homogenous

$X_{3\times1}$ ↳ non homogenous (eucledian)

Similarly,

$$x_2 = K_{2(3\times3)} \; R_{(3\times3)} \; X_{3\times1}$$

x_2 ↓ homogenous coord.

$X_{3\times1}$ ↳ eucledian

Date

$$M = K_2 R$$

$$x_2 = M_{3\times3} \, \cancel{} \, X_{3\times1}$$

homogenous

$$M^{-1} x_2 = M^{-1} M X$$
$$= I X$$
$$= X$$

$$x_1 = K_1 X$$
$$= K_1 (M^{-1} x_2)$$

$$(AB)^{-1} = B^{-1} A^{-1}$$

$$(K_2 R)^{-1} = R^{-1} K_2^{-1}$$

$$x_1 = K_1 R^{-1} K_2^{-1} x_2$$

$R^{-1} = R^T$ [prop of rot matrix]

$$x_1 = K_1 R^T K_2^{-1} x_2$$

$$\therefore H = K_1 R^T K_2^{-1}$$

QUESTION 4

PART 1. The estimated intrinsic camera parameters with RMS Reprojection error estimate

```
 Camera matrix:
[[1.86349876e+03 0.00000000e+00 4.68234969e+02]
 [0.00000000e+00 1.86021775e+03 5.87379215e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]

Focal length (fx, in pixels): 1863.4987551658044
Focal length (fy, in pixels): 1860.2177517430914
Principal point (cx, cy, in pixels): (468.23496946072356, 587.3792145211783)
Skew parameter: 0.0
RMS Reprojection Error:
2.179318198285322
```

PART 2. The extrinsic camera parameters, i.e., rotation matrix and translation vector for each of the selected images.

```
 Rotation Vectors:
(array([[-0.5260311 ],
       [-0.00924322],
       [ 0.00830635]]), array([[-0.61301299],
       [ 0.07890199],
       [ 1.48797854]]), array([[-0.38879902],
       [ 0.01023163],
       [ 0.00642834]]), array([[-0.46901342],
       [-0.21214107],
       [ 0.10595139]]), array([[-0.4648468 ],
       [ 0.26374784],
       [-0.04209609]]), array([[-0.01734941],
       [-0.28329269],
       [-0.0273258 ]]), array([[0.02179978],
       [0.3748336 ],
       [1.52234853]]), array([[ 0.06048106],
       [-0.39835481],
       [-1.52944463]]), array([[-0.37915219],
       [ 0.32415953],
       [ 1.53684259]]), array([[0.00668641],
       [0.07812816],
       [0.00812555]]), array([[-0.29344987],
       [-0.01607604],
       [-0.00308403]]), array([[-0.14774696],
       [ 0.20528685],
       [ 1.53533215]]), array([[-0.13149708],
       [-0.02285825],
       [ 1.58117888]]), array([[-0.37776663],
       [-0.27555296],
```

```
[-1.37812335]]), array([[-0.31245573],
[-0.00595588],
[ 1.55373632]]), array([[-0.23397063],
[ 0.37117452],
[ 1.57675599]]), array([[-0.17266649],
[ 0.44273083],
[-0.08909975]]), array([[0.21372661],
[0.26446275],
[0.01699479]]), array([[-0.46971353],
[ 0.24886848],
[ 0.08464455]]), array([[ 0.04739668],
[-0.20803184],
[ 1.57374346]]), array([[-0.46463714],
[ 0.33669408],
[ 1.48940724]]), array([[-0.25718821],
[ 0.0704691 ],
[ 0.01047128]]), array([[0.09352868],
[0.1906243 ],
[1.54555185]]), array([[-0.16167531],
[ 0.05359493],
[ 0.01080996]]), array([[ 0.13382143],
[-0.15662397],
[-1.56024799]]))
```

```
 Translation Vectors:
(array([[-1.01331941],
       [-2.54736807],
       [13.98889437]]), array([[ 1.24581506],
       [-2.45392342],
       [14.3285548 ]]), array([[-1.28807971],
       [-2.06239279],
       [13.38327538]]), array([[-1.98525251],
       [-1.83079939],
       [16.59401811]]), array([[-2.4295278 ],
       [-1.00803914],
       [17.39736086]]), array([[-0.53855978],
       [-2.16037396],
       [13.03086116]]), array([[ 1.19816893],
       [-2.24816415],
       [13.07612697]]), array([[-1.10091648],
       [ 2.47614267],
       [12.02371653]]), array([[ 1.60739916],
       [-2.80836982],
       [13.86323484]]), array([[-2.53238239],
       [-0.7200573 ],
       [15.78821609]]), array([[-2.20316558],
       [ 0.29229304],
       [17.14515434]]), array([[ 1.53536904],
       [-2.11472051],
       [14.09788633]]), array([[ 1.48335986],
       [-1.71011291],
       [14.45999136]]), array([[-1.46903453],
       [ 2.43318706],
```

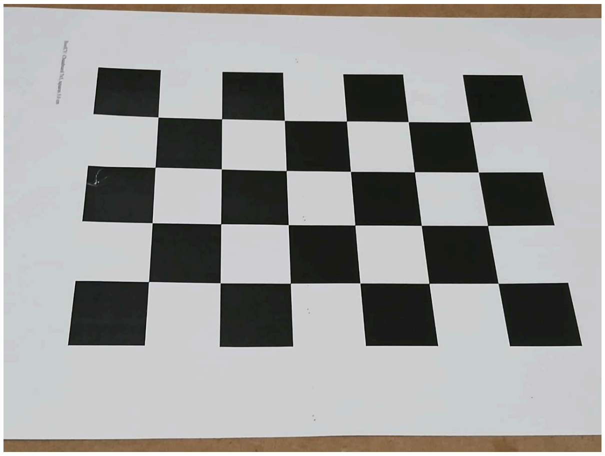
```
[13.59710134]]), array([[ 1.68091052],
[-2.20203859],
[14.14023691]]), array([[ 1.70931424],
[-2.36441514],
[14.19129731]]), array([[-1.76682446],
[-2.05844499],
[15.04505241]]), array([[-0.95756912],
[-2.86566391],
[21.89673467]]), array([[-1.15727165],
[-2.43111803],
[13.45196861]]), array([[ 1.76459876],
[-2.46196905],
[16.31712927]]), array([[ 1.55197216],
[-2.68115945],
[14.42616575]]), array([[-2.27866368],
[-0.50238119],
[15.87689701]]), array([[ 1.43030316],
[-2.42570157],
[14.87136706]]), array([[-2.45191605],
[-0.92158818],
[16.15330692]]), array([[-1.52483893],
[ 3.1932485 ],
[16.40700943]]))
```

PART 3:
estimated radial distortion coefficients:

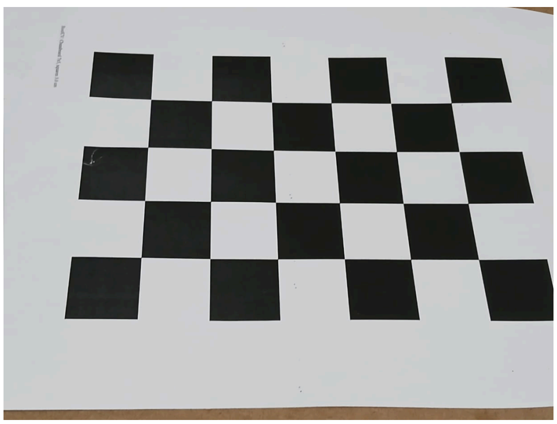
```
 Distortion coefficient:
[[ 3.98439392e-01 -5.41592723e+00 -5.30244941e-03 -5.95353510e-03
   1.91257133e+01]]
```

The undistorted 5 of the raw images are present in the folder undistorted images.

ORIGINAL:



AFTER APPLYING DISTORTION COEFFICIENTS:



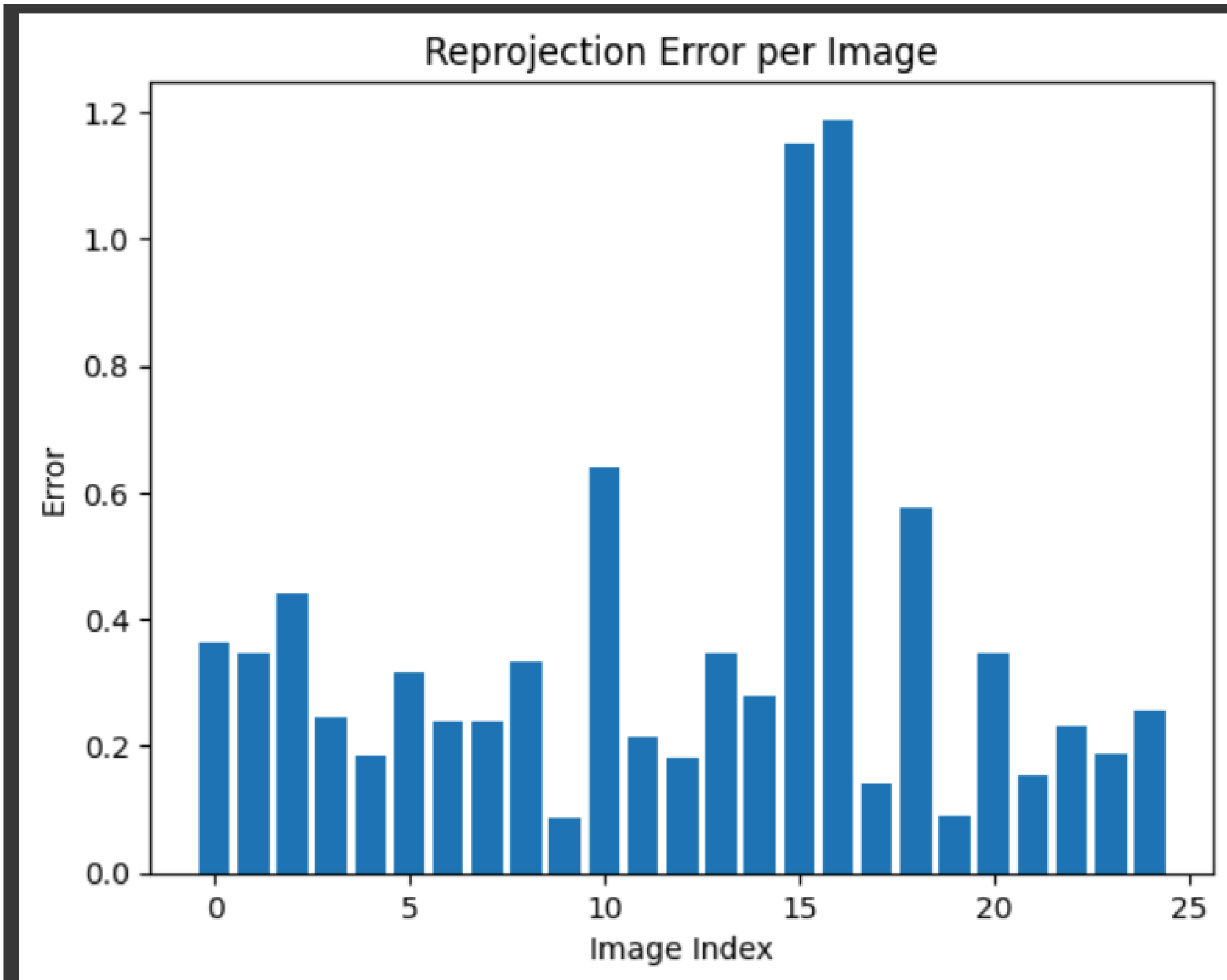
The corner points are wider in the undistorted image. The straight lines bow out towards the edges due to radial distortion. This results in magnification at the corners
This happens due to the spherical aberration of the lenses.

Radial distortion can be represented as follows:

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

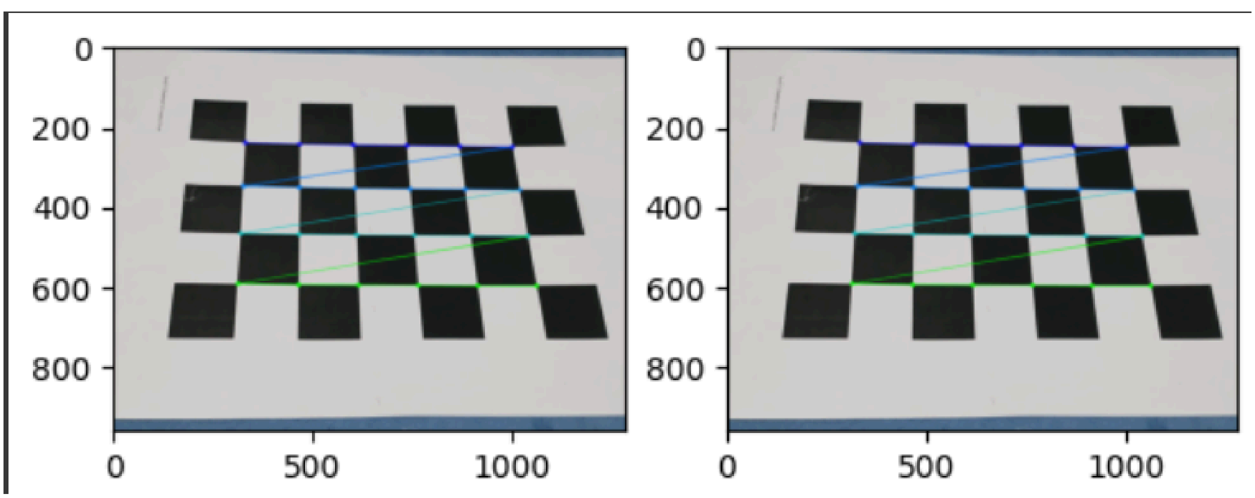
Source: opencv documentation

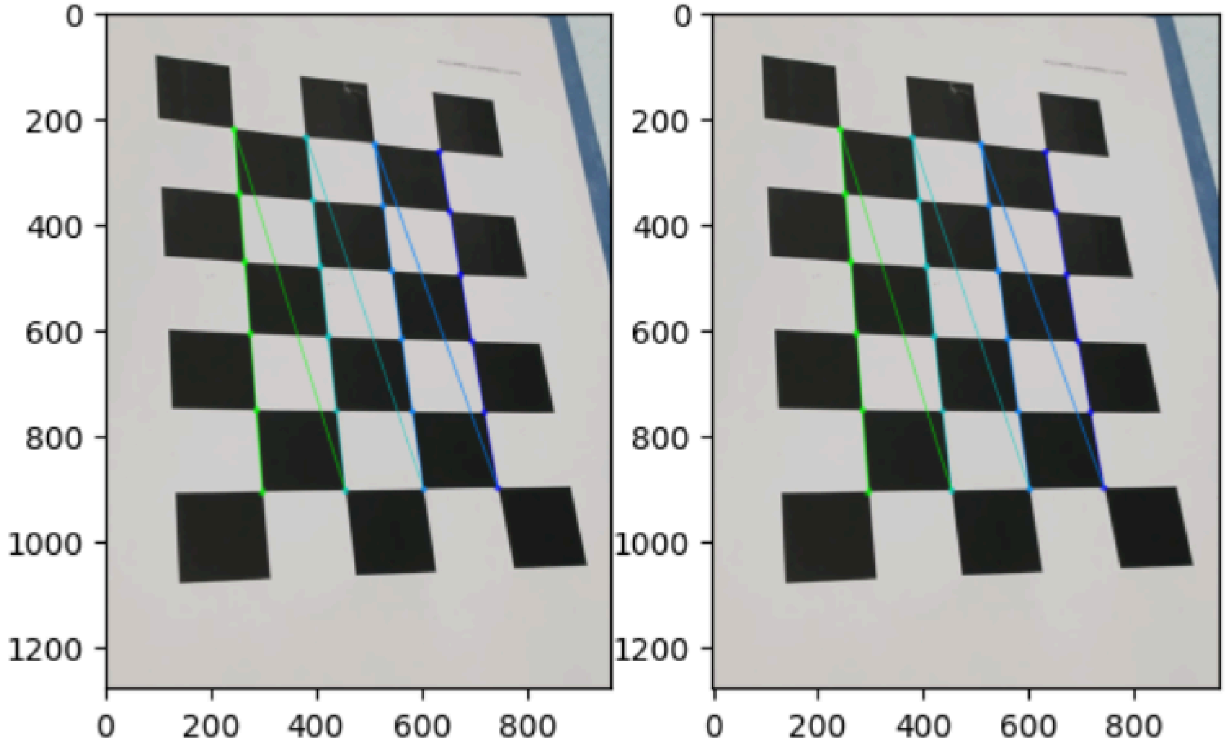
PART 4: REPROJECTION ERROR



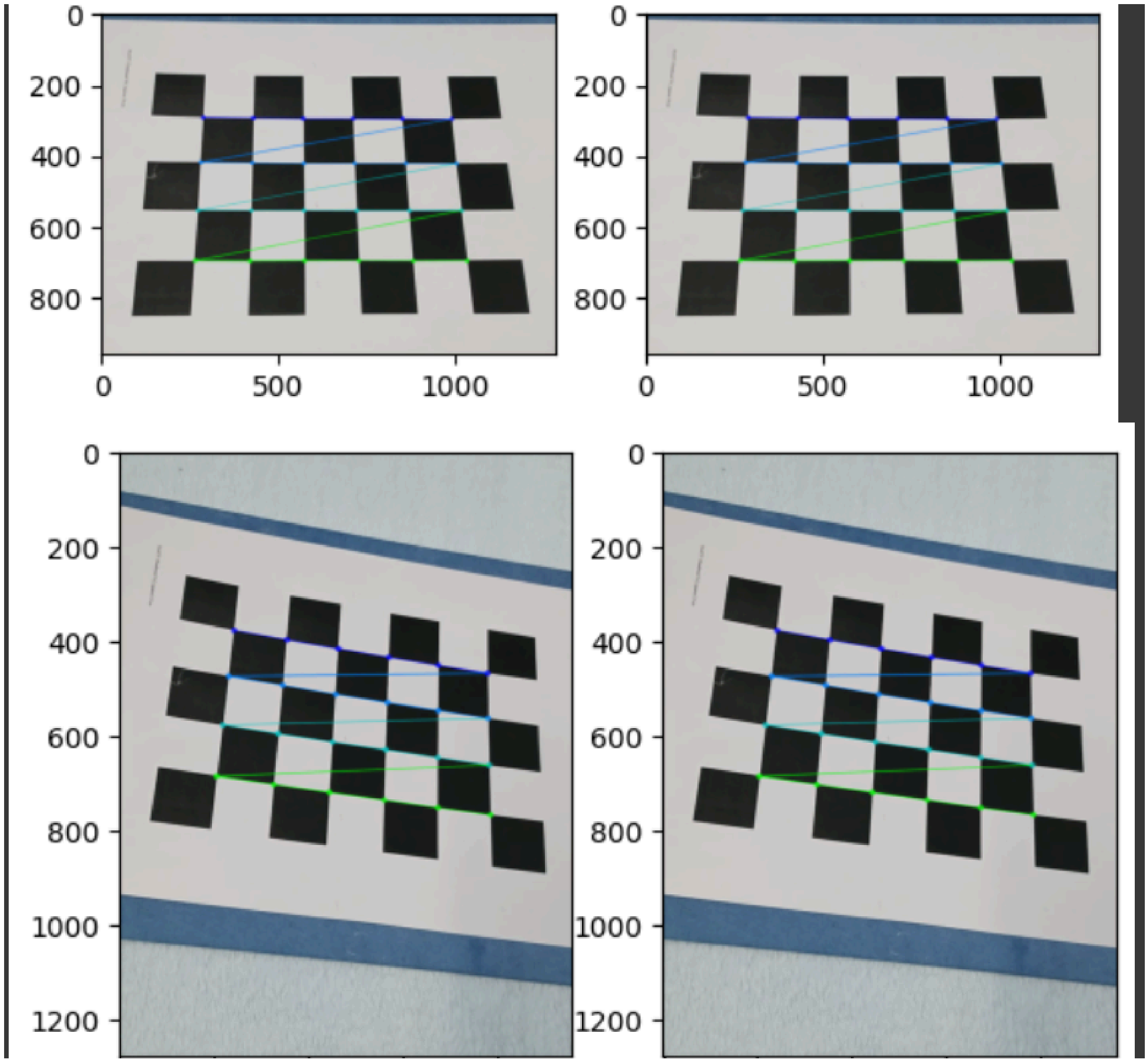

```
Mean reprojection error: 0.35083158802585424
Standard deviation of reprojection error: 0.27351369750979293
```

PART 5:

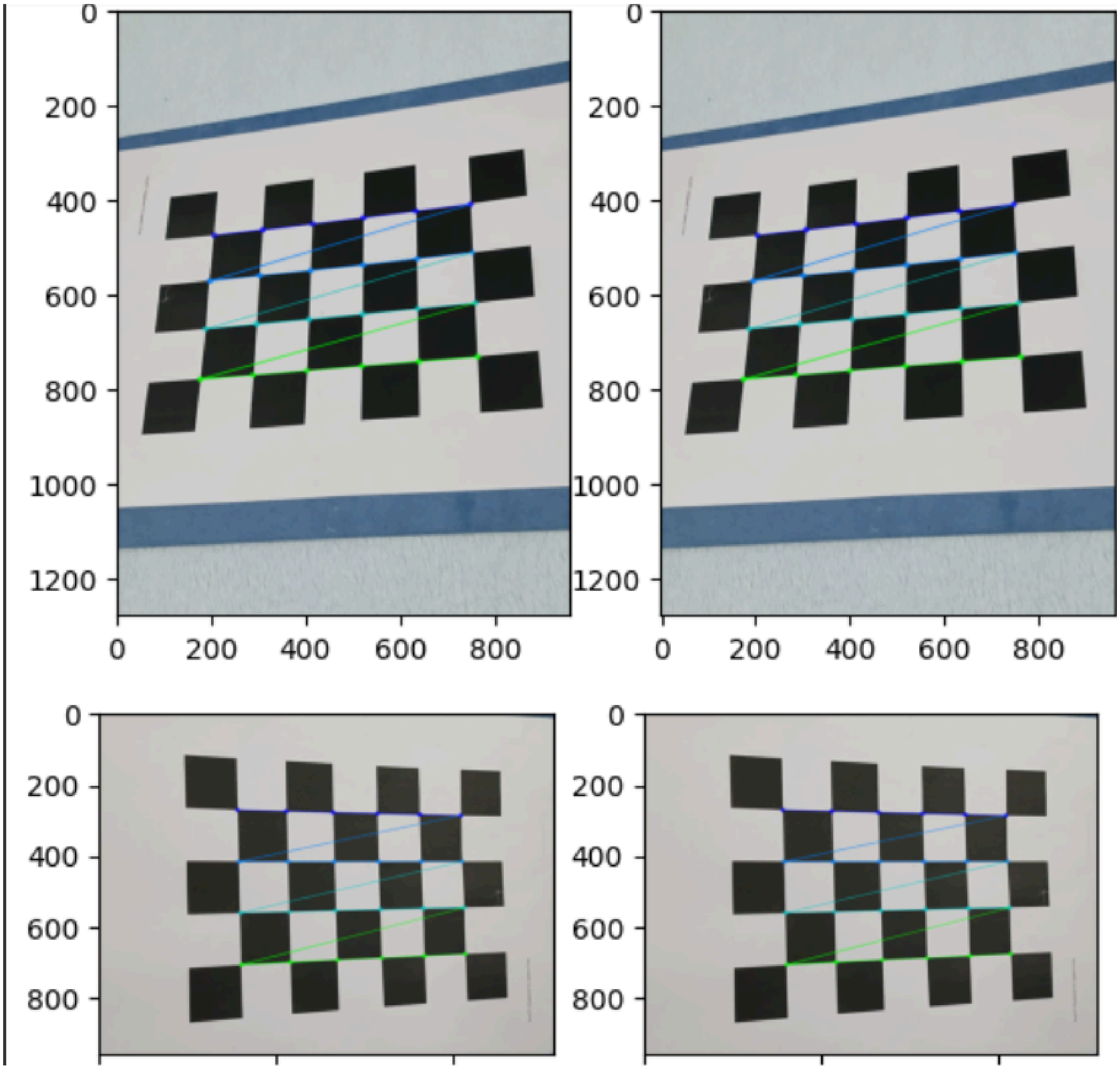


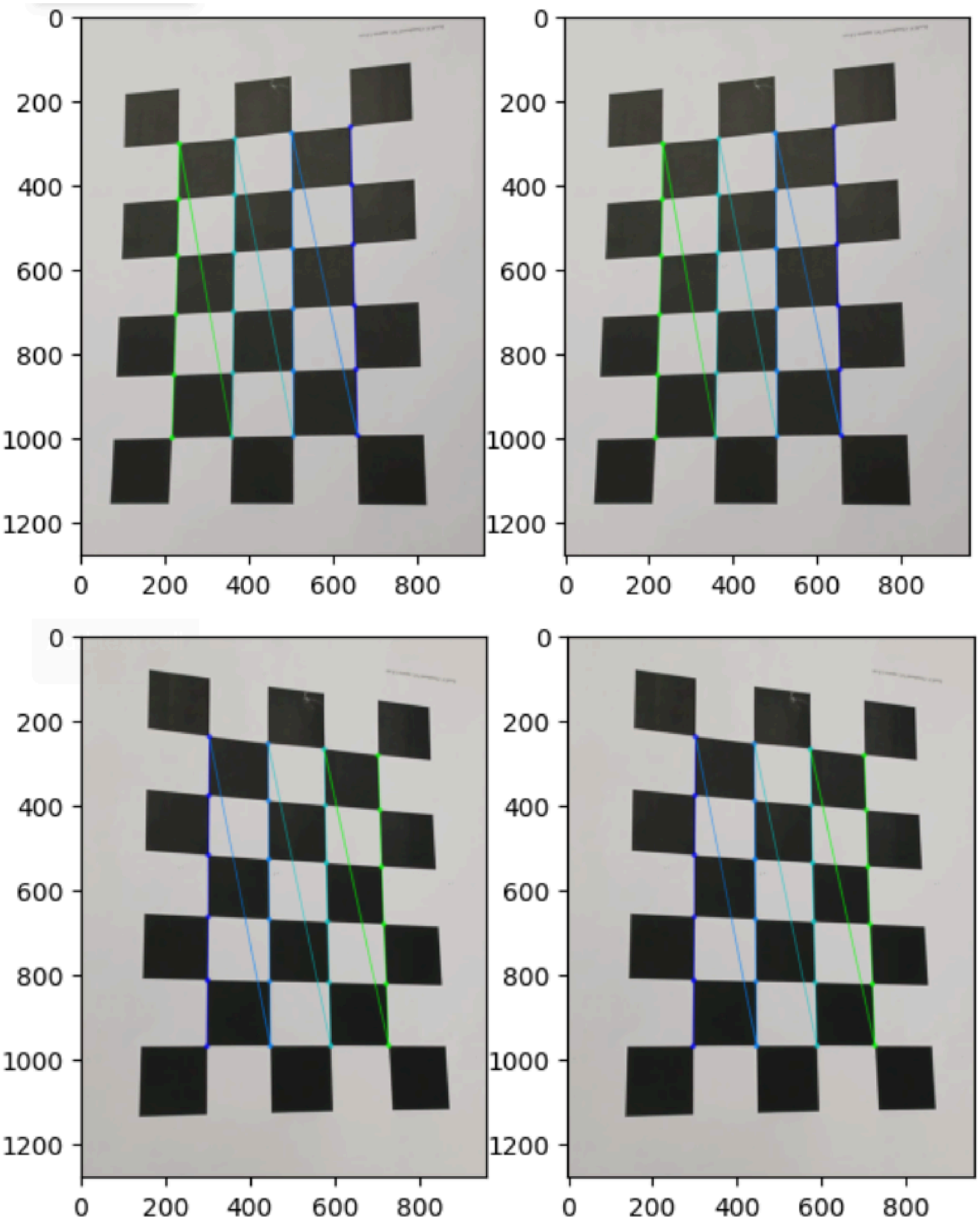


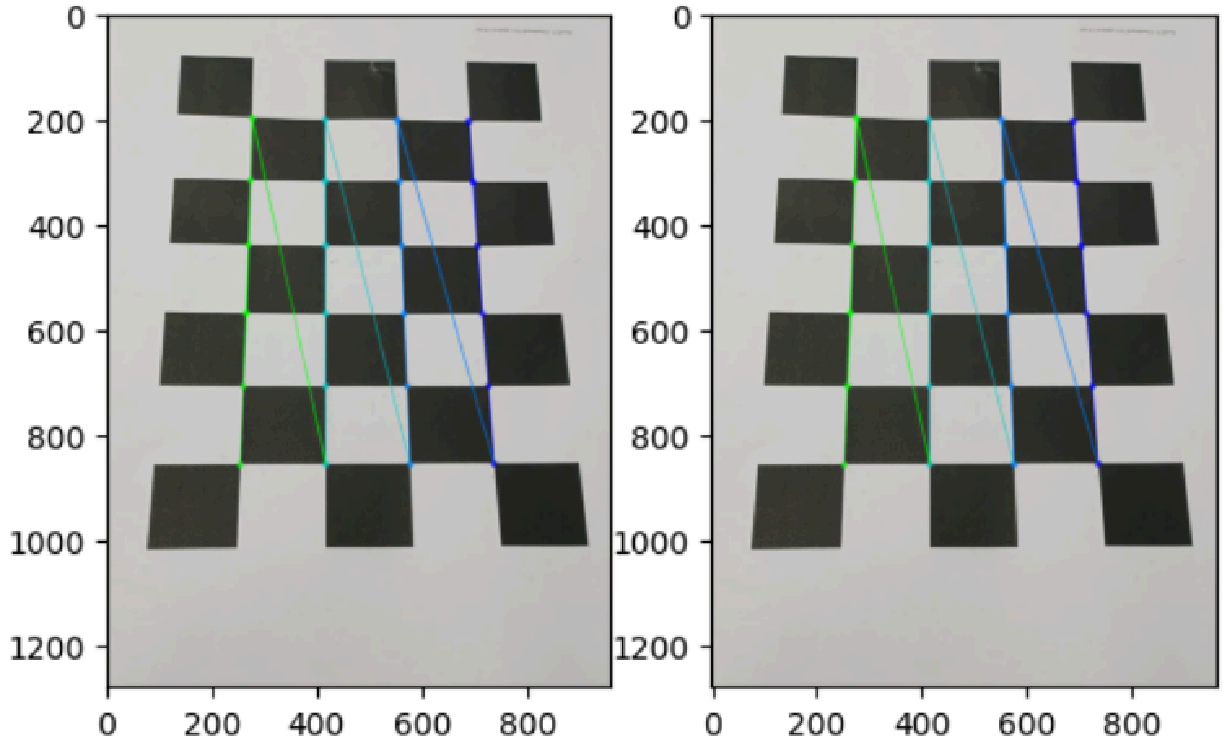
0
200
400
600
800
1000
1200
0
200
400
600
800

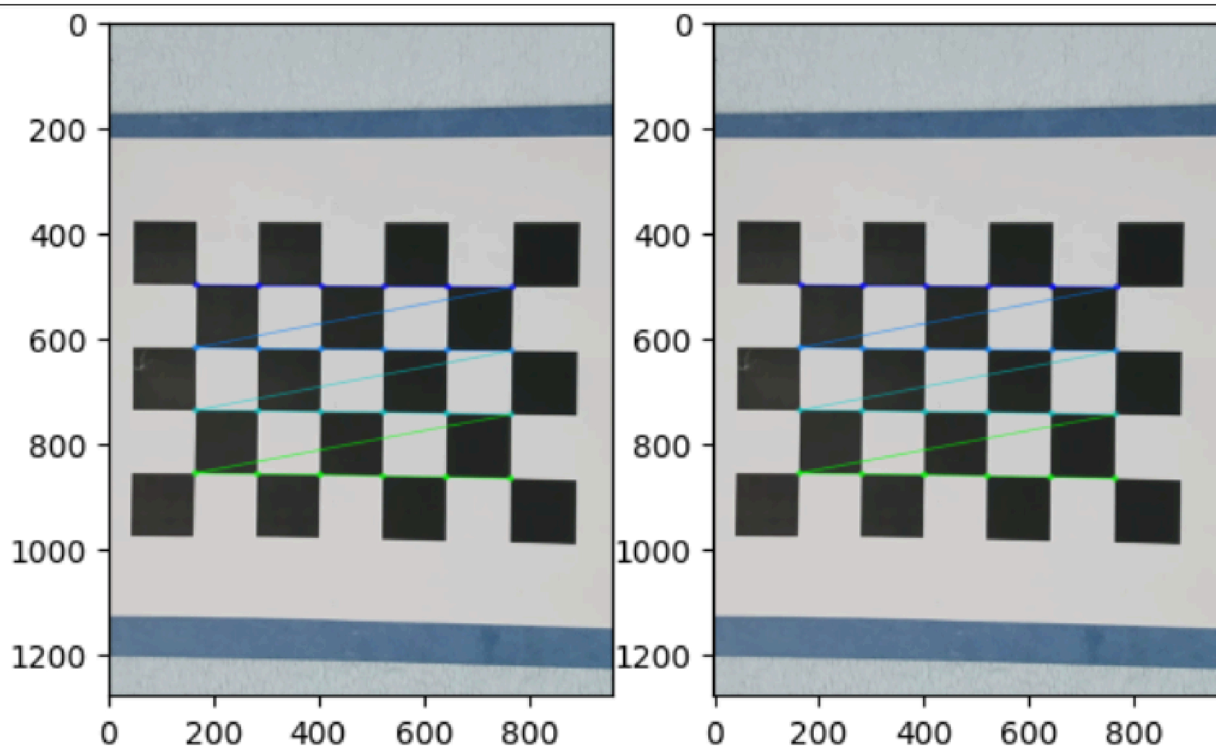


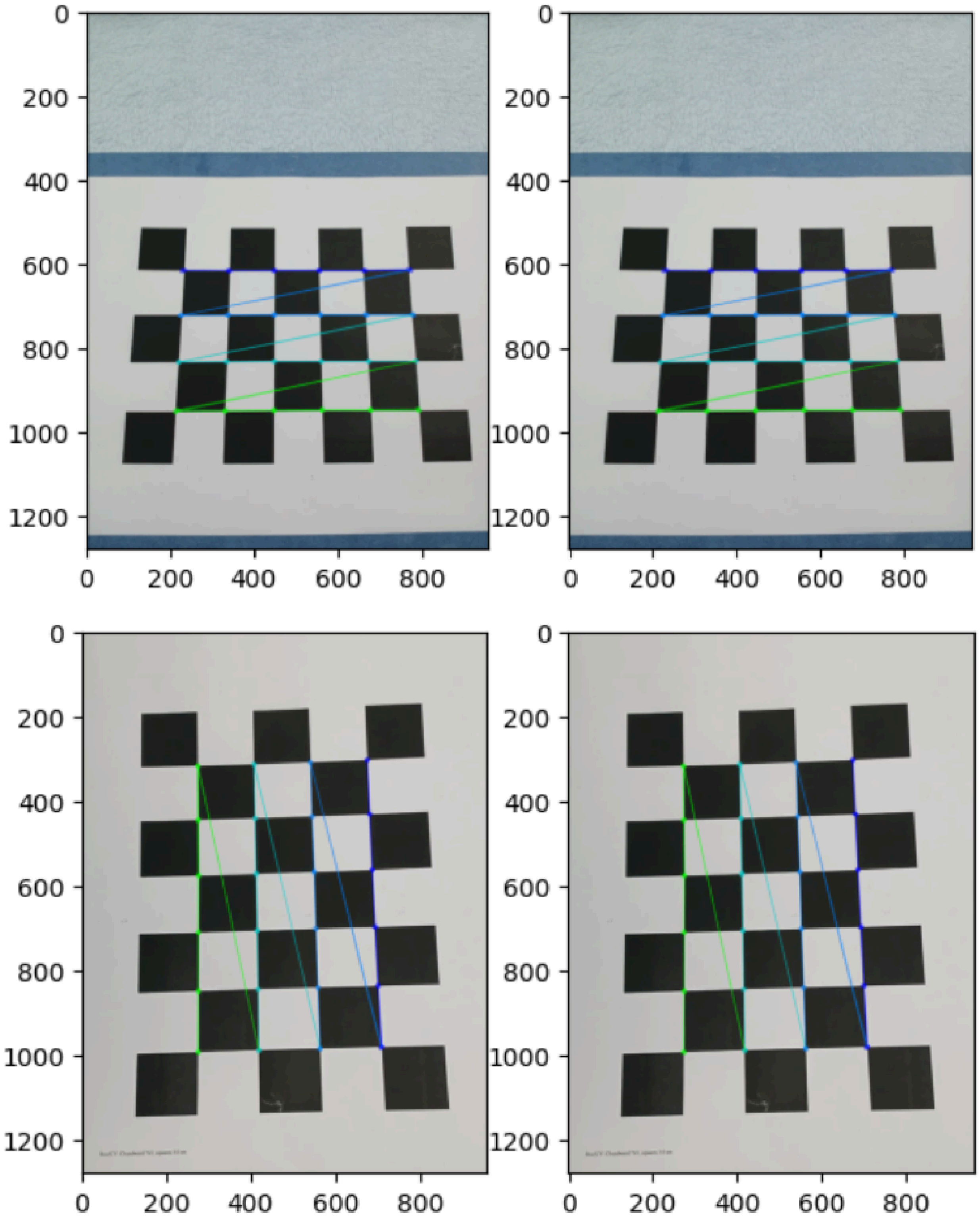
0
200
400
600
800
0
500
1000
0
200
400
600
800
0
500
1000
0
200
400
600
800
1000
1200
0
200
400
600
800
1000
1200


0
200
400
600
800
1000
1200
0
200
400
600
800
0
200
400
600
800

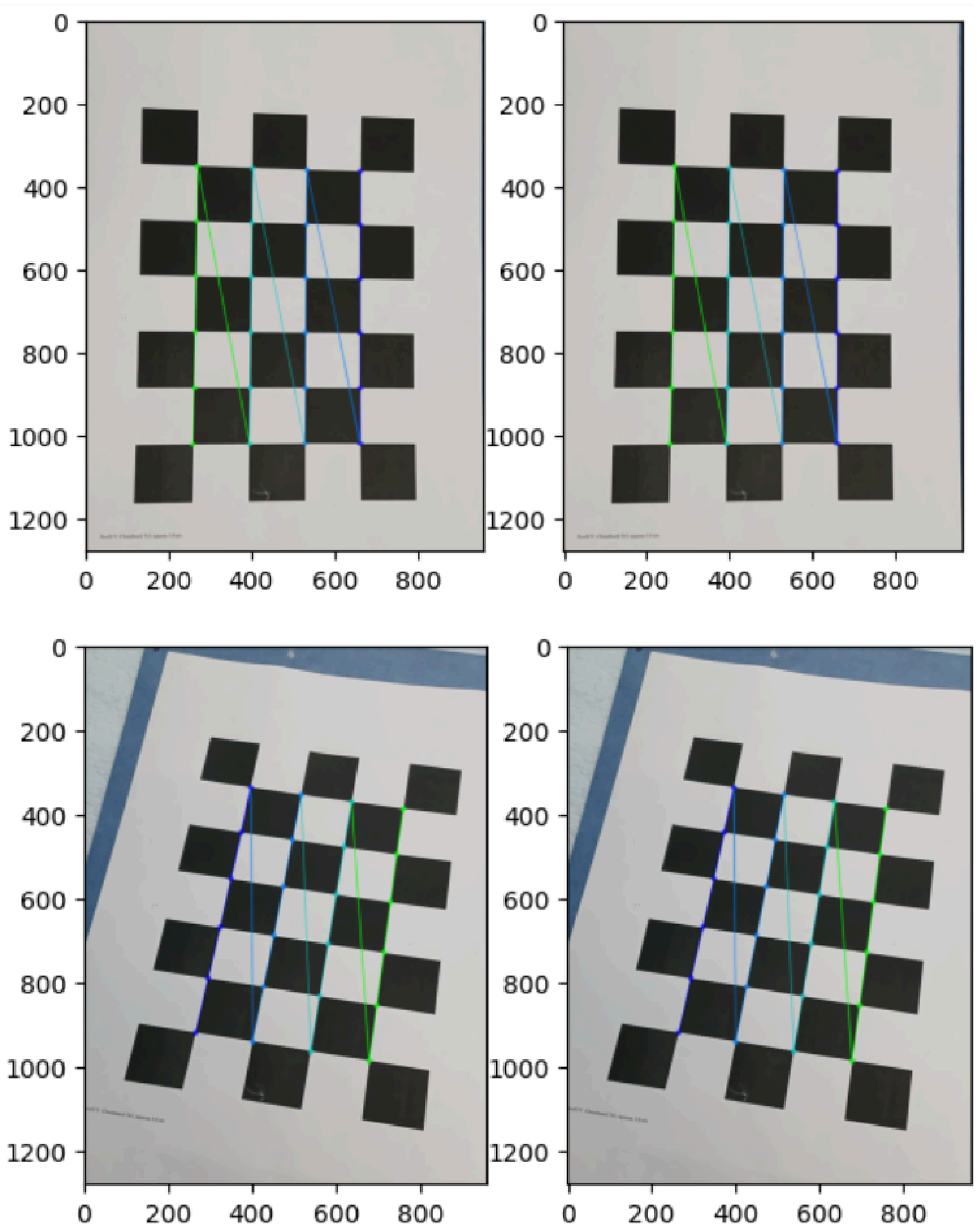

0
200
400
600
800
1000
1200
0
200
400
600
800

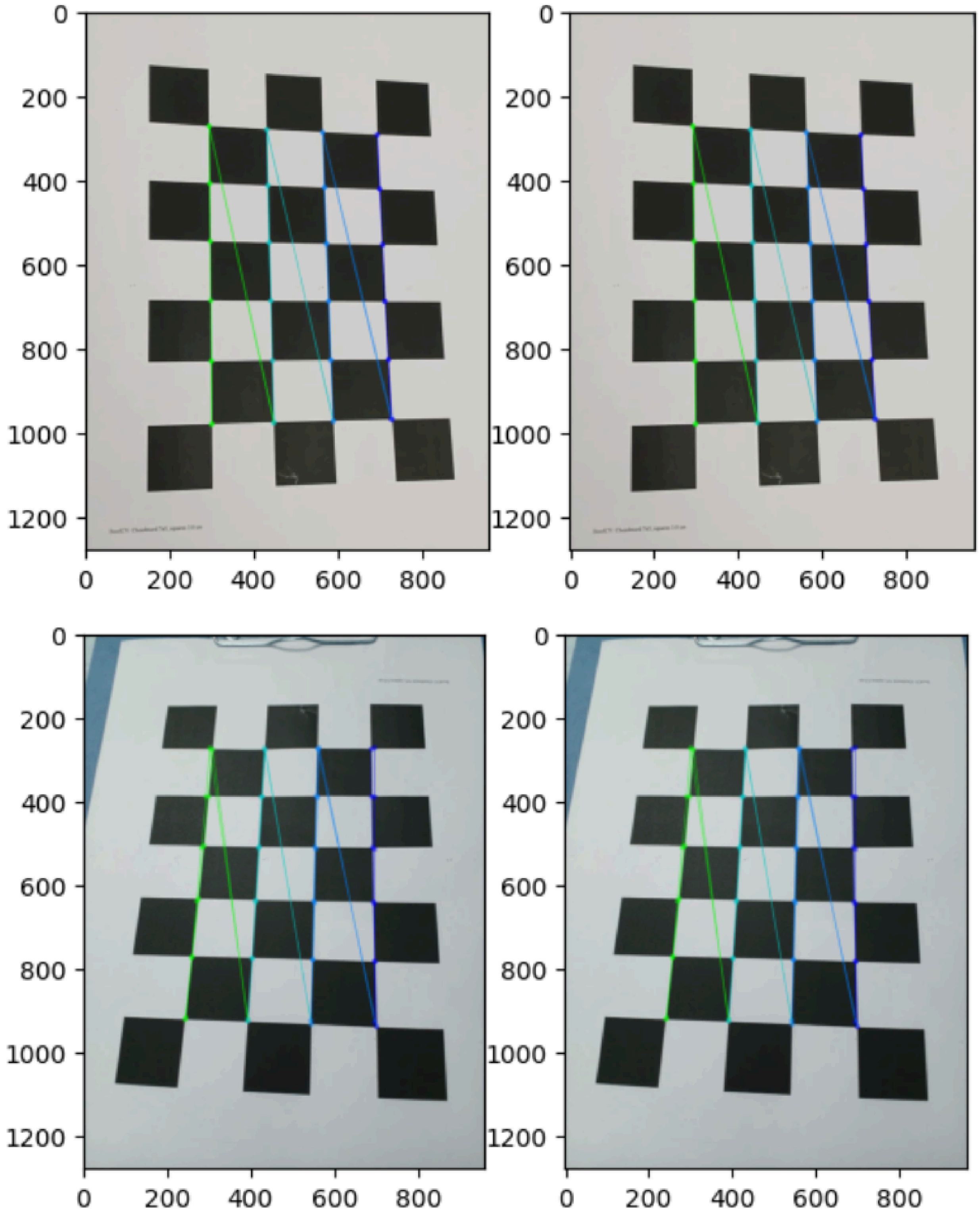

0
200
400
600
800
1000
1200
0 200 400 600 800
0
200
400
600
800
1000
1200
0 200 400 600 800

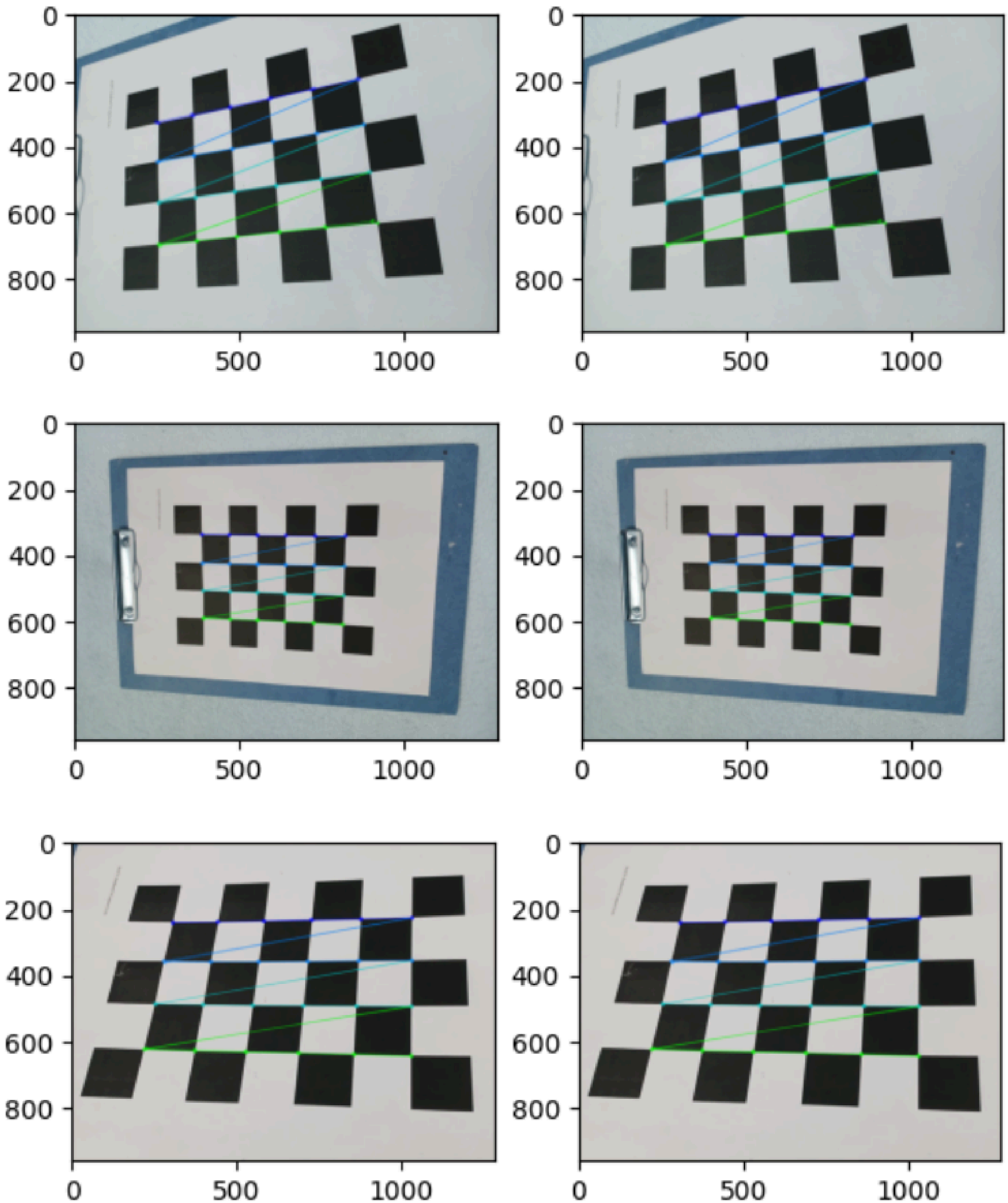

0
200
400
600
800
1000
1200
0 200 400 600 800
0
200
400
600
800
1000
1200
0 200 400 600 800

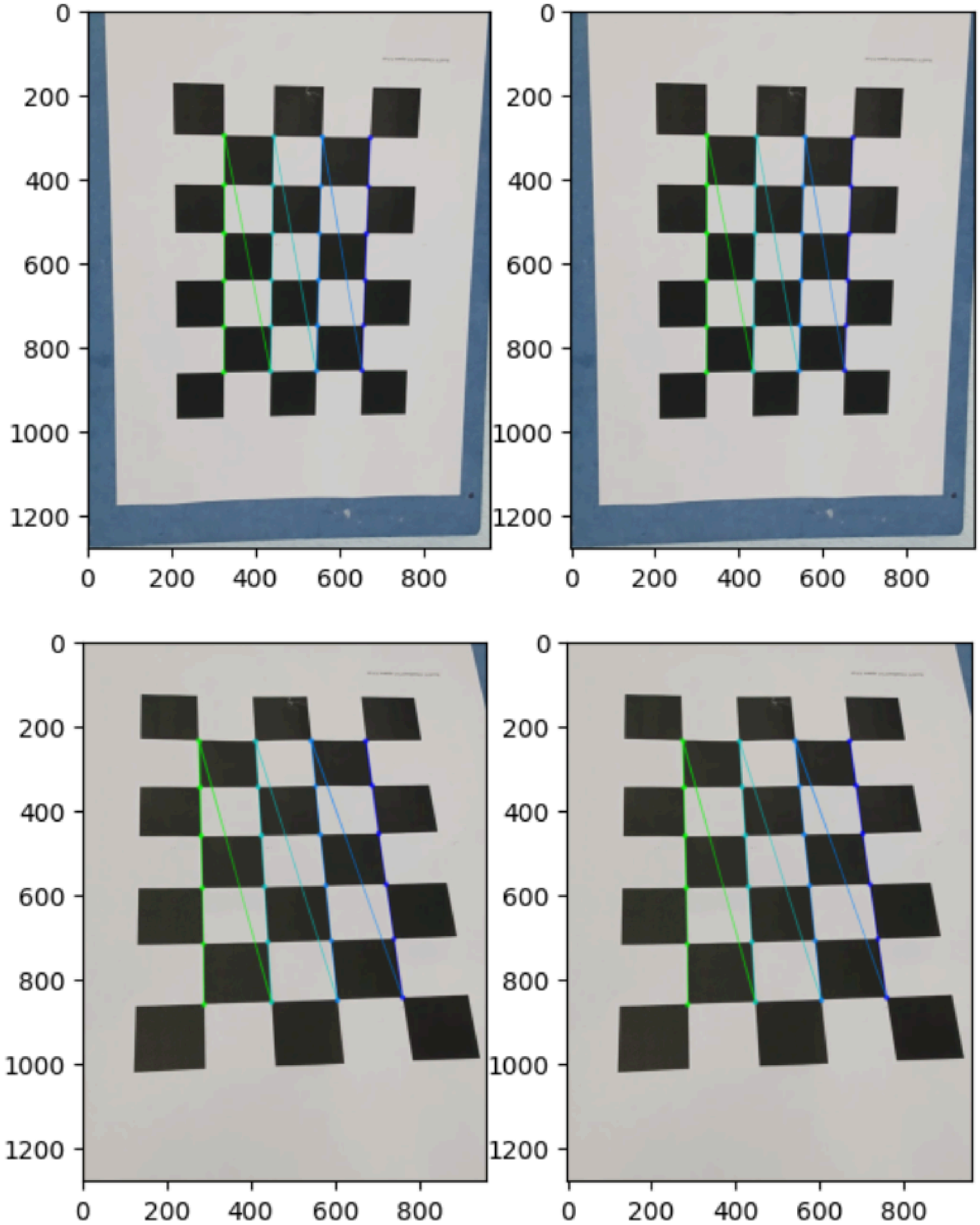


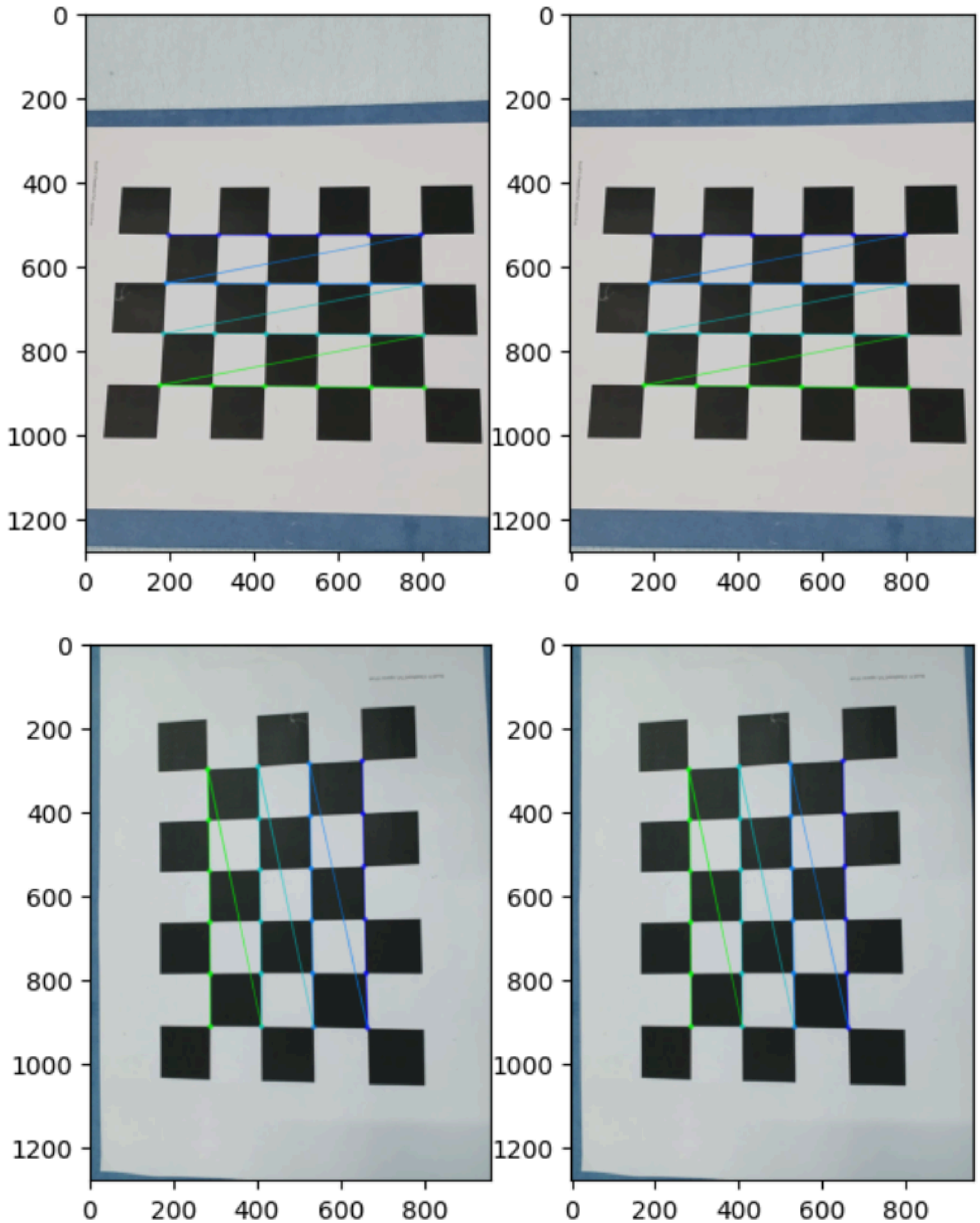
0
200
400
600
800
1000
1200
0 200 400 600 800

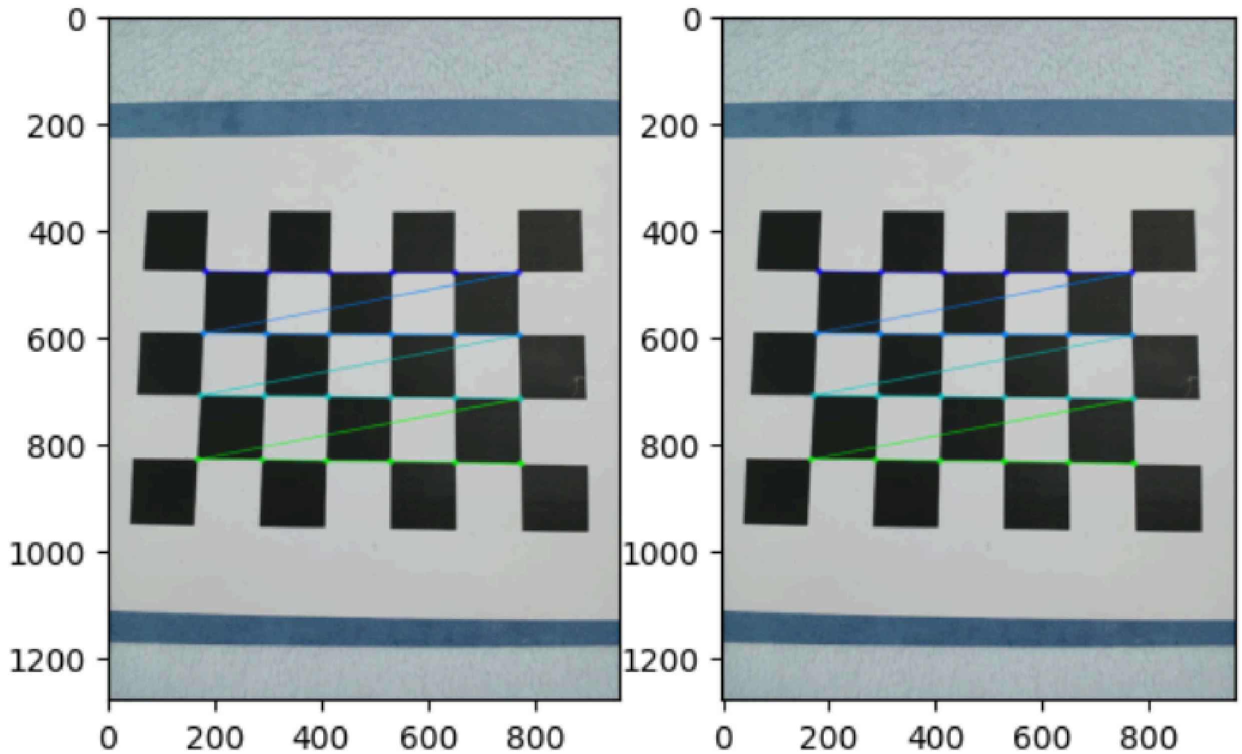

0
200
400
600
800
1000
1200
0
200
400
600
800
0
200
400
600
800
1000
1200
0
200
400
600
800
0
200
400
600
800
1000
1200
0
200
400
600
800
0
200
400
600
800
1000
1200
0
200
400
600
800

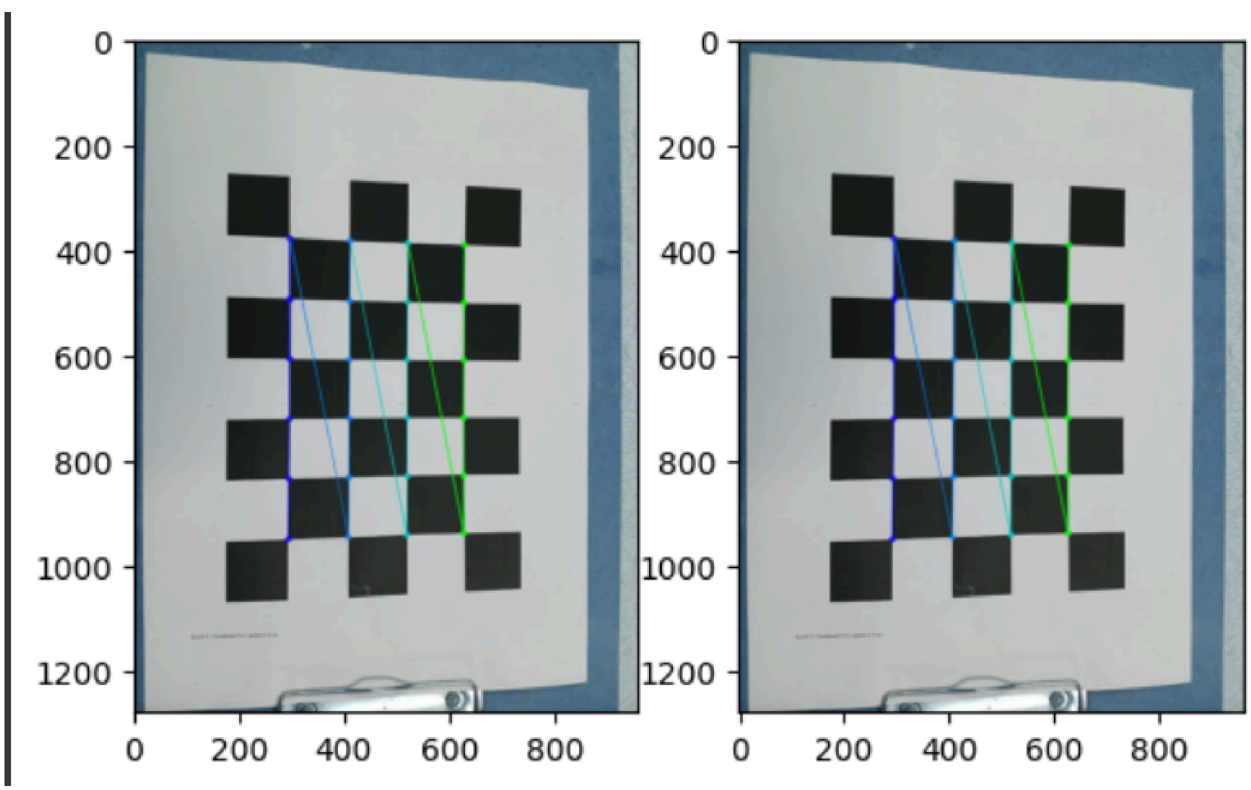

0
200
400
600
800
1000
1200
0
200
400
600
800


0
200
400
600
800
0
500
1000


0 200 400 600 800 1000 1200
0 200 400 600 800
0 200 400 600 800 1000 1200
0 200 400 600 800
0 200 400 600 800 1000 1200
0 200 400 600 800
0 200 400 600 800 1000 1200
0 200 400 600 800


0
200
400
600
800
1000
1200
0
200
400
600
800
0
200
400
600
800
1000
1200
0
200
400
600
800
0
200
400
600
800
1000
1200
0
200
400
600
800
0
200
400
600
800
1000
1200
0
200
400
600
800


0
200
400
600
800
1000
1200
0
200
400
600
800


0
200
400
600
800
1000
1200
0
200
400
600
800

Reprojected error:
Reprojection error is calculated as the euclidean distance between the 2 points. The cv.projectPoints projects the 3D object points into the image plane. The output are the projected points which act as our first point.
The observed image points are detected in the calibration images using features. We find the reprojection error between the projected points and the actual image points.

PART 6: Normals

```
[-0.01095732  0.50205429  0.86476669]
[-0.31663138  0.42720158  0.84690222]
[0.00874156 0.3791005  0.92531421]
[-0.22678905  0.43673712  0.87053284]
[0.26084588 0.43747742 0.86056547]
[-0.27923409  0.02095972  0.95999428]
[0.25251664 0.21754209 0.94282065]
[-0.28922772  0.20795449  0.93439941]
[-0.03280991  0.43382084  0.90040157]
[ 0.07807442 -0.00636224  0.99692723]
[-0.01539634  0.28926795  0.95712434]
[0.03961089 0.22319412 0.97396887]
[-0.09830058  0.06827471  0.99281195]
[0.02940283 0.41649326 0.90866321]
[-0.1997548   0.19339569  0.96057073]
[0.07950731 0.3755124  0.92340079]
[0.43322588 0.14667514 0.88927034]
[ 0.26118201 -0.20739606  0.94274643]
[0.2176199  0.45763891 0.86209524]
[-0.10088459 -0.16189419  0.9816377 ]
[-0.06764186  0.4928382   0.8674878 ]
[0.06829701 0.25451295 0.96465469]
[0.18093924 0.05980591 0.9816742 ]
[0.05246339 0.16118055 0.98552951]
[-0.18407186  0.01392832  0.9828141 ]
```

QUESTION 5:
PART 1:

```
Normal Vector: [ 0.95207118 -0.1539658   0.26430097]
Normal Offset: 5.743694294895788
Normal Vector: [ 0.60407281 -0.77412903  0.18926246]
Normal Offset: 5.062352211986537
Normal Vector: [ 0.75932657 -0.56254883  0.32705042]
Normal Offset: 4.954015451590526
Normal Vector: [ 0.47346333 -0.7599202  -0.44536924]
Normal Offset: 5.254984226428699
Normal Vector: [ 0.52528321 -0.84450165 -0.1043768 ]
Normal Offset: 6.1036455719220495
Normal Vector: [ 0.40409023 -0.88184713  0.24301591]
Normal Offset: 5.436720512190625
Normal Vector: [ 0.96334688 -0.25648046 -0.07861659]
Normal Offset: 8.213132600834637
Normal Vector: [ 0.87718352 -0.41582556  0.24007951]
Normal Offset: 7.435546784352129
Normal Vector: [ 0.9065703  -0.30866042 -0.28785247]
Normal Offset: 8.020993504372285
Normal Vector: [0.94902101 0.13112718 0.28664401]
Normal Offset: 9.499025451119042
Normal Vector: [ 0.98745566  0.06712949 -0.14291586]
Normal Offset: 10.309979614425062
Normal Vector: [ 0.90149712  0.39908052 -0.16744454]
Normal Offset: 8.140582366167767
Normal Vector: [ 0.95936444 -0.00899235 -0.2820266 ]
Normal Offset: 8.724272669222737
Normal Vector: [ 0.9493562   0.22833398 -0.21583881]
Normal Offset: 7.271752317637934
Normal Vector: [ 0.43460444 -0.89601842  0.09093933]
Normal Offset: 6.595058223525193
```

```
Normal Vector: [ 0.97607976 -0.21597961  0.02492227]
Normal Offset: 4.772067004354305
Normal Vector: [ 0.60371533 -0.79239849  0.08736381]
Normal Offset: 3.7038756210944084
Normal Vector: [ 0.73737765 -0.5739658  -0.35614247]
Normal Offset: 3.9715421503819686
Normal Vector: [ 0.89093965  0.24536379 -0.38212975]
Normal Offset: 7.618700179547204
Normal Vector: [ 0.99121488 -0.12226682 -0.05043704]
Normal Offset: 6.09367867924695
Normal Vector: [ 0.825289   -0.10756967 -0.55437067]
Normal Offset: 5.978131037863134
Normal Vector: [ 0.93223066 -0.19842704 -0.30260984]
Normal Offset: 6.265358115913809
Normal Vector: [ 0.98936009 -0.13850217 -0.04453952]
Normal Offset: 7.325197139472717
Normal Vector: [ 0.96691202 -0.15794554 -0.2003356 ]
Normal Offset: 8.493221054907726
Normal Vector: [ 0.99427448 -0.1042191   0.02359328]
Normal Offset: 8.846168921169085
Normal Vector: [ 0.662242   -0.74563698  0.07389876]
Normal Offset: 7.749464153135399
Normal Vector: [ 0.73370584 -0.67940958  0.00885226]
Normal Offset: 6.969188884408078
```

PART 2:

Q5.) Part 2

Given

$\theta\_l = [\theta_{l,1}, \theta_{l,2}, \ldots \theta_{l,n}]^T$

lidar normals

$\theta\_c = [\theta_{c,1}, \theta_{c,2} \ldots \theta_{c,n}]^T$

camera normals

$\alpha_c = [\alpha_{c_1}, \alpha_{c_2} \ldots \alpha_{c_n}]$

camera translation normals

$\alpha_l = [\alpha_{l_1}, \ldots \alpha_{l_n}]$ - lidar offsets

~~AT~~ According to the paper,

$$U, S, Vt = SVD(\underbrace{\theta_l \theta_c^T}_{\text{known}})$$

$$R\_1 = V U^T$$

$$t\_1 = (\theta_c^T \theta_c)^{-1} \theta_c^T (\alpha_c - \alpha_l)$$

Using R_1 & t_1 we can estimate transformation matrix

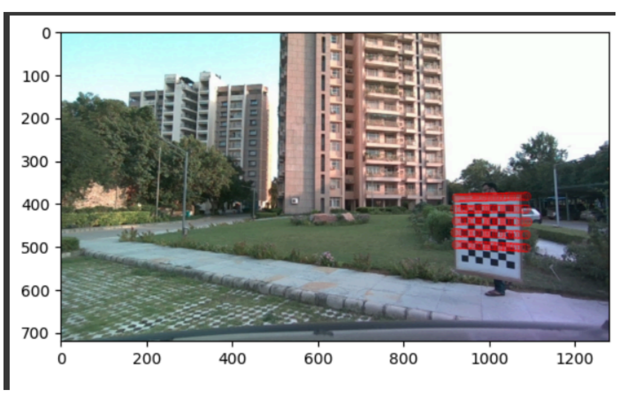
$$ {}^{c}\tilde{T}_{l} = \begin{bmatrix} R\_l & T\_l \\ 0 & 1 \end{bmatrix} $$

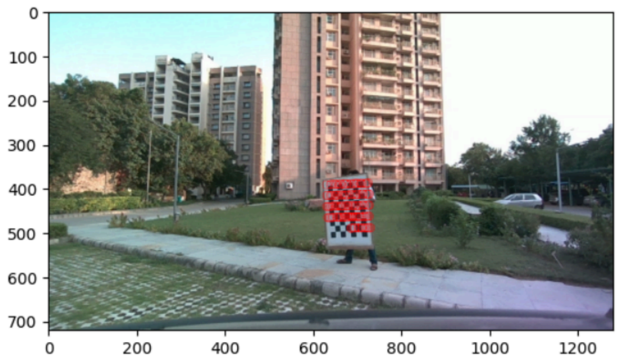
This maps lidar plane to camera plane i.e converts lidar coordinate system to camera coordinate system

PART 3:

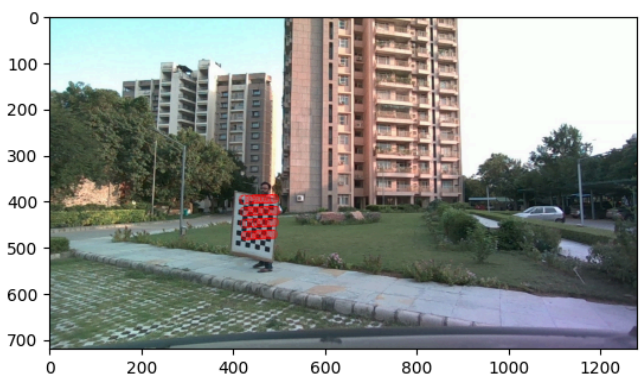
```
estimated transformation matrix:
[[-1.75158876e-01 -9.84540172e-01  1.36164316e-04  8.85565624e-02]
 [ 1.53559935e-02 -2.87025999e-03 -9.99877970e-01 -3.62081439e-01]
 [ 9.84420419e-01 -1.75135410e-01  1.56213440e-02 -5.99256711e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
```

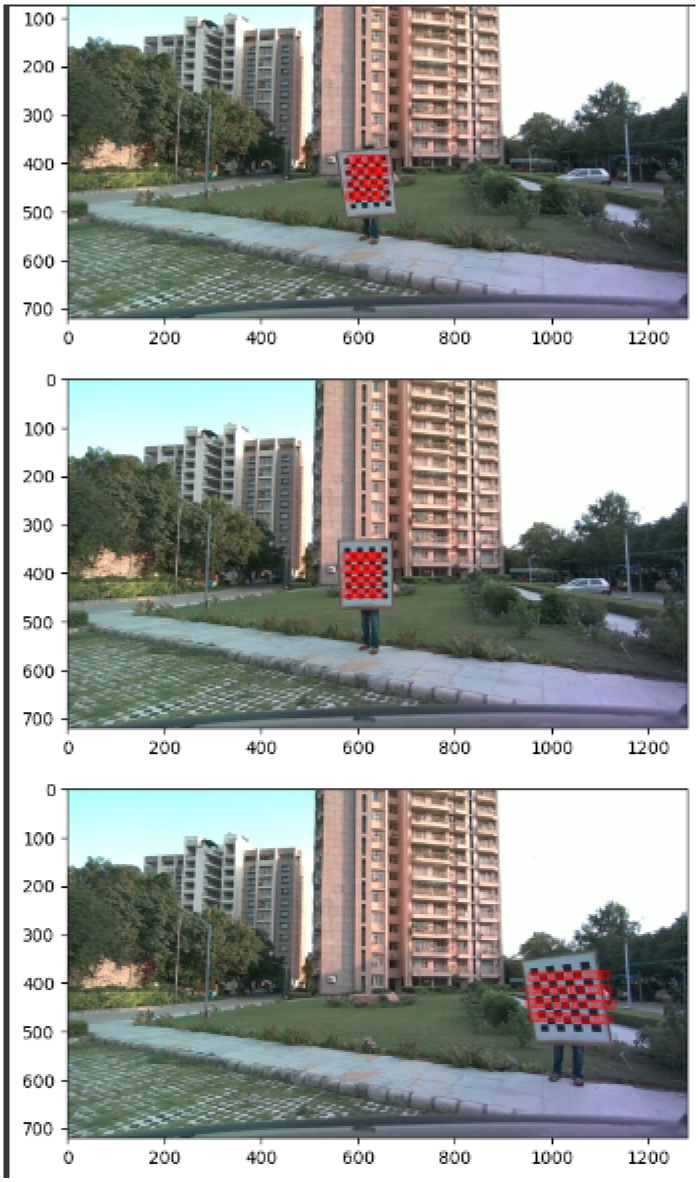
PART 4:




0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

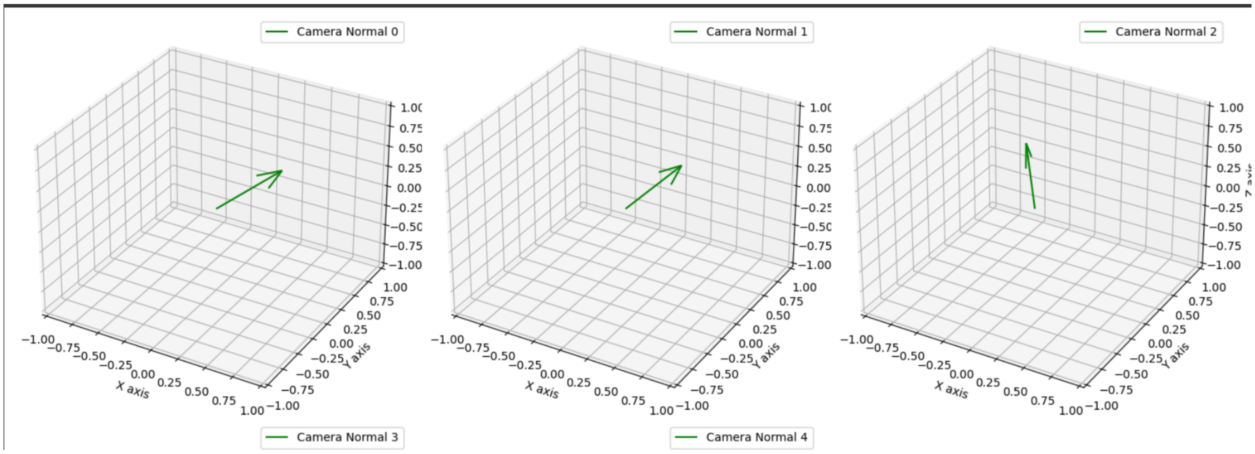

0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

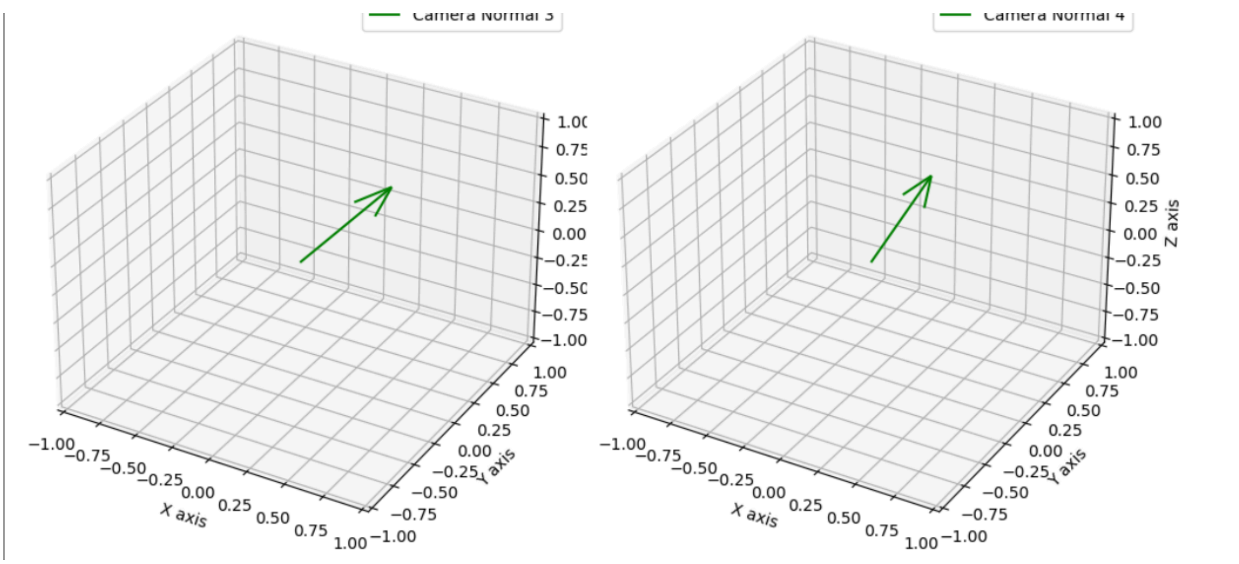

0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

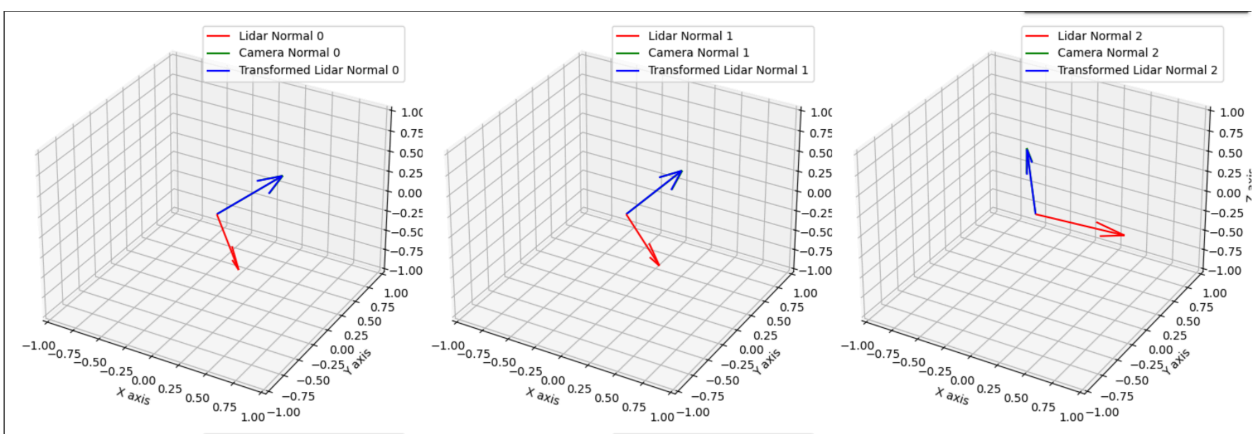


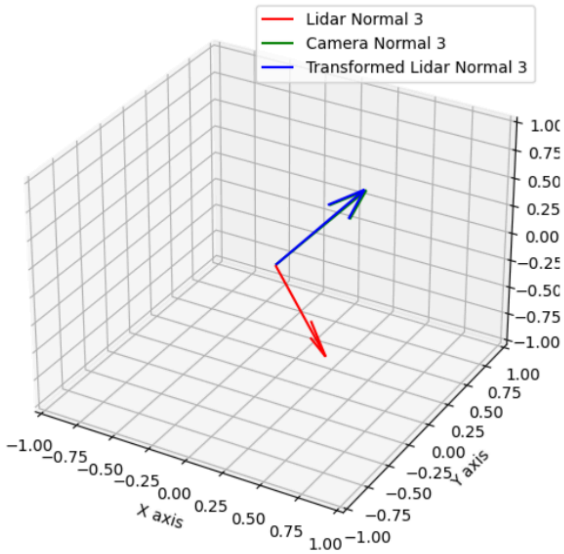
The rest of the points can be found in the jupyter notebook. No since we are using lidar scans some of the points are outside the checkerboard pattern’s boundary.

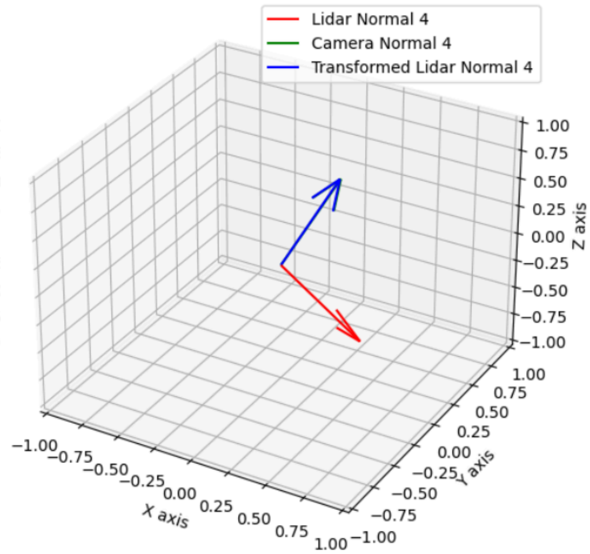
PART 5:


Camera Normal 0
Camera Normal 1
Camera Normal 2
Camera Normal 3
Camera Normal 4
X axis
Y axis

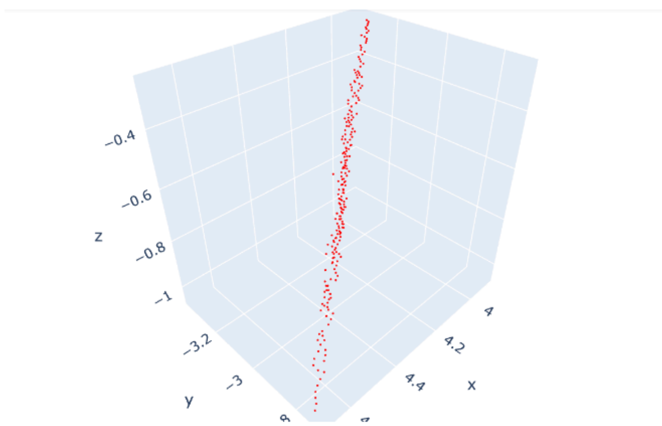

Camera Normal 3
Camera Normal 4
X axis
Y axis
Z axis


Lidar Normal 0
Camera Normal 0
Transformed Lidar Normal 0
Lidar Normal 1
Camera Normal 1
Transformed Lidar Normal 1
Lidar Normal 2
Camera Normal 2
Transformed Lidar Normal 2
X axis
Y axis

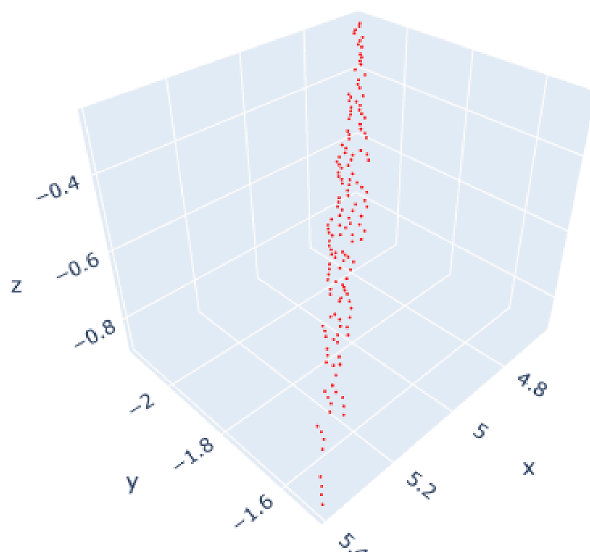




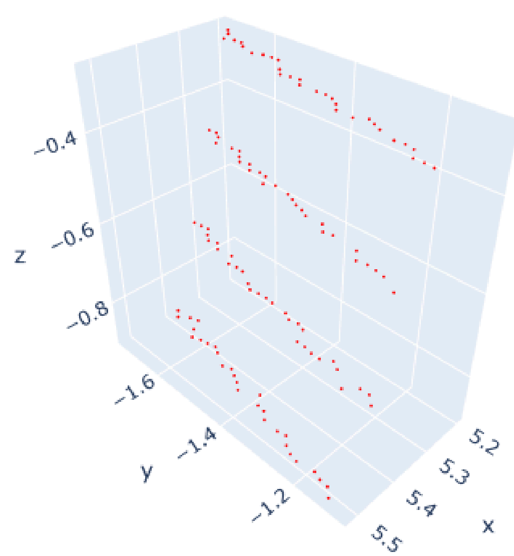
We see camera normal and transformed camera normal coinciding meaning that the rotation matrix applied to lidar normals are correct



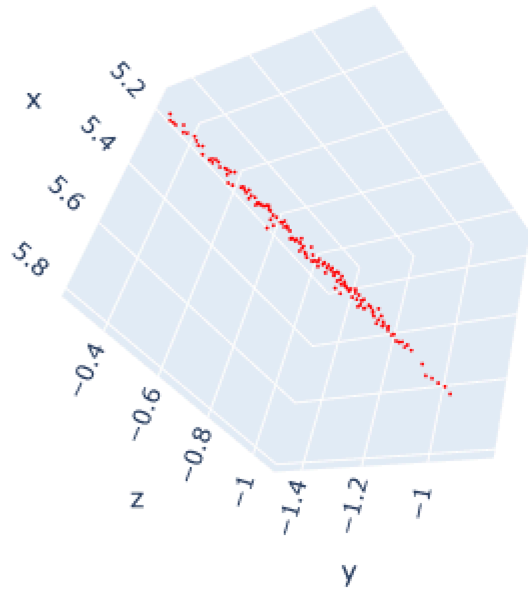
−0.4
−0.6
z
−0.8
−1
−3.2
−3
y
4
4.2
4.4
x

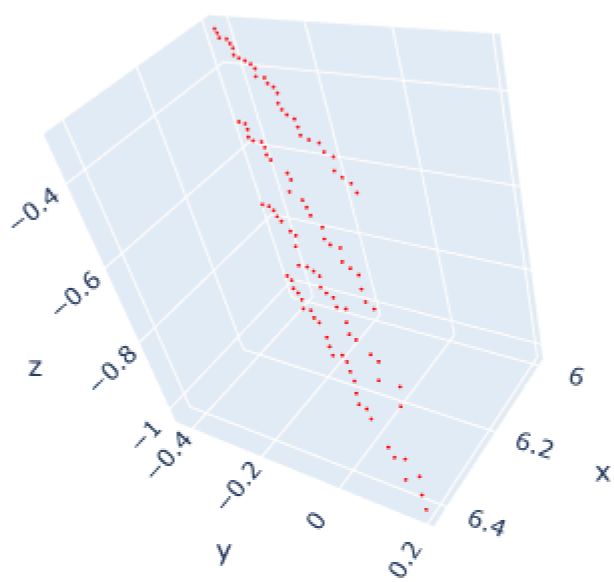


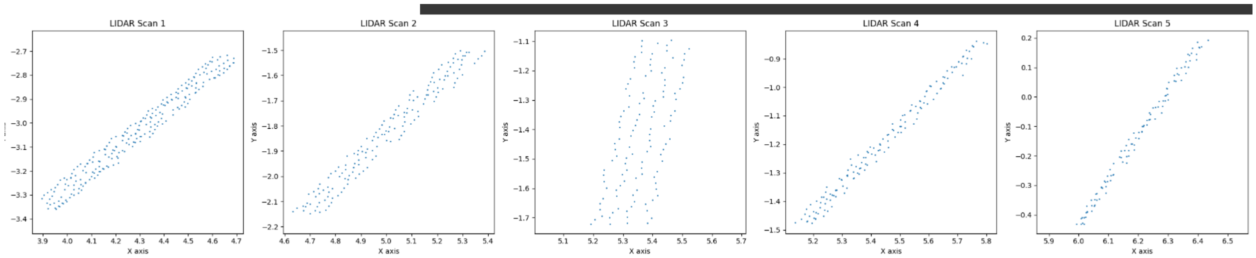
−0.4
−0.6
z
−0.8
−2
−1.8
y
−1.6
4.8
5
x
5.2

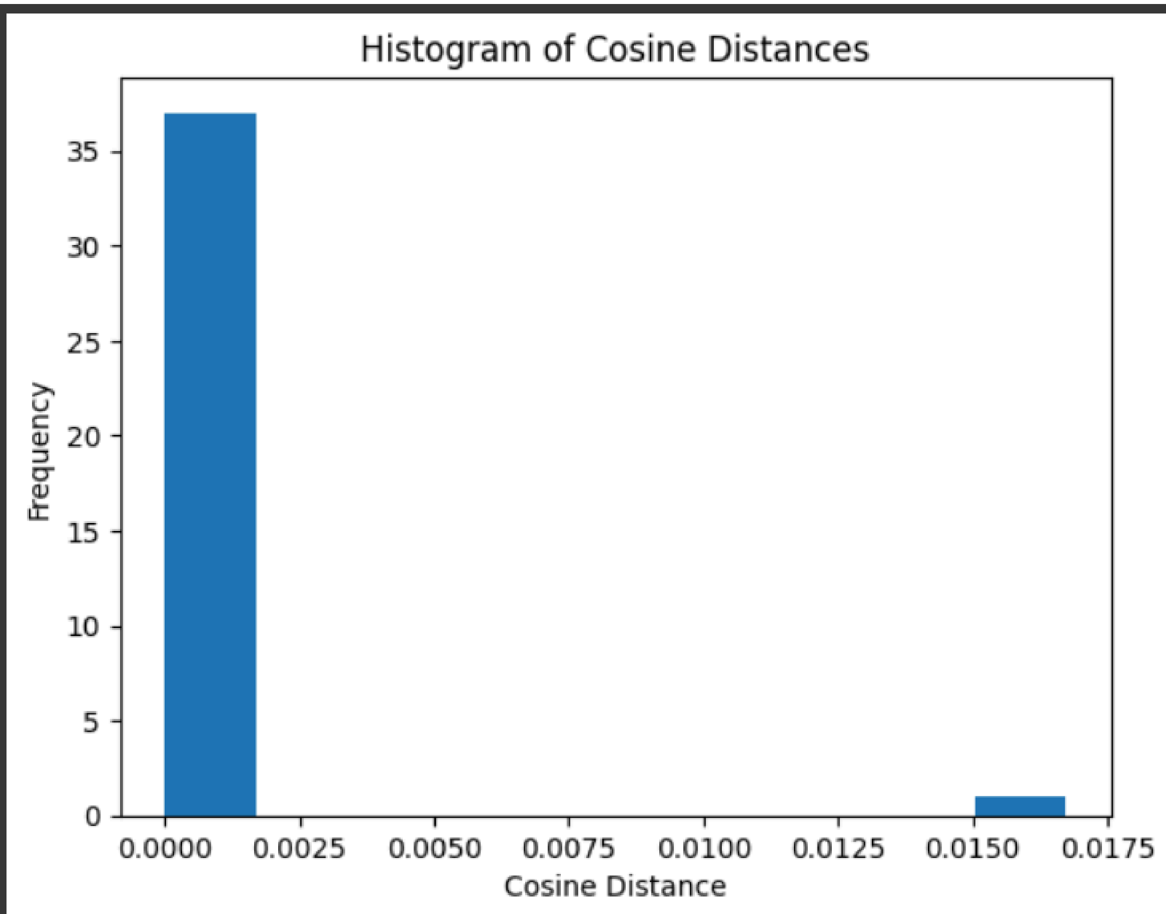


−0.4
−0.6
z
−0.8
−1.6
−1.4
y
−1.2
5.2
5.3
5.4
x
5.5


x
5.2
5.4
5.6
5.8
−0.4
−0.6
−0.8
−1
z
−1.4
−1.2
−1
y


−0.4
−0.6
−0.8
−1
z
−0.4
−0.2
0
0.2
y
6
6.2
6.4
x






```
Average Error: 0.0006469021634402106
Standard Deviation: 0.0026610000720887644
```

REFERENCE:

Used opencv documentation https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html