Q1. $v = \begin{bmatrix} 2 \\ 5 \\ 1 \end{bmatrix}$; $R_y(\alpha) = \begin{bmatrix} c\alpha & 0 & s\alpha \\ 0 & 1 & 0 \\ -s\alpha & 0 & c\alpha \end{bmatrix}$; $R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\alpha & -s\alpha \\ 0 & s\alpha & c\alpha \end{bmatrix}$

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

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

i) $T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 3 \\ 0 & -1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

ii) $T \cdot v = \begin{bmatrix} 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 3 \\ 0 & -1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 5 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ -3 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \\ -3 \end{bmatrix}$ ↳ v_2

Origin will simply get translated at the end → $\begin{bmatrix} -1 \\ 3 \\ 0 \end{bmatrix}$

iii) Finding eigenvector for R

$Rv = \lambda v$ $\quad \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} v = \lambda v$ $\quad \therefore \det \begin{bmatrix} -\lambda & 0 & 1 \\ -1 & -\lambda & 0 \\ 0 & -1 & -\lambda \end{bmatrix} = 0$

$\therefore \quad -\lambda^3 + 1 = 0 \quad \therefore \quad \lambda = 1$ (real eigenvalue)

$\therefore \quad v = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \rightarrow n = \frac{1}{2\sin\theta} \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} = \begin{bmatrix} -1/\sqrt{3} \\ 1/\sqrt{3} \\ 1/\sqrt{3} \end{bmatrix}$

VISION

$$\theta = \cos^{-1}\left(\frac{tr(R)-1}{2}\right) = \cos^{-1}\left(\frac{-1}{2}\right) = \frac{2\pi}{3}$$

iv) $R = I + \sin\theta N + (1-\cos\theta)N^2$ ... $N = \frac{1}{\sqrt{3}}\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \frac{\sqrt{3}}{2}\times\frac{1}{\sqrt{3}}\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} + \frac{3}{2}\times\frac{1}{3}\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1/2 & 1/2 \\ -1/2 & 0 & 1/2 \\ -1/2 & -1/2 & 0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} -2 & -1 & 1 \\ -1 & -2 & -1 \\ 1 & -1 & -2 \end{bmatrix}$$

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

2) We know

$$Rx = u(u\cdot x) + \sin\theta(u\times x) - \cos\theta(u\times(u\times x)) \quad \cdots (i)$$

$\rightarrow$ components $\parallel$ & $\perp$ to u

$$x = u(u\cdot x) - u\times(u\times x) \quad \cdots (ii)$$

$\hookrightarrow$ substitute in (i)

$$Rx = u(u^Tx) + \sin\theta(u\times x) + \cos\theta(x - u(u^Tx))$$

$\hookrightarrow$ ~~open~~ expanding ;

$$Rx = \cos\theta x + \sin\theta(u\times x) + u(u^Tx)(1-\cos\theta)$$

3. $x_1 = K_1 [I | 0] x$

$x_2 = K_2 [R | 0] x$

$K_2^{-1} x_2 = K_2^{-1} K_2 R R^{-1} [R | 0] x$

$= [R \; R [I | 0] x$

$R^{-1} K_2^{-1} x_2 = [I | 0] x$

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

$H = K_1 R^{-1} K_2^{-1}$

Question 4.

Parameters :

Focal length x :
1362.7128611324936
Focal length y :
1363.4158320466925

Skew :
0.0

Principal point x :
621.1520932746042
Principal point y :
670.0923559740186

Distortion :

[[0.07365253 -1.34949601 -0.00698695 -0.01223193 2.61600918]]

Rotation Vectors :

(array([[-0.03487554],
[0.12390295],
[0.03323081]]), array([[-0.18768038],
[-0.23459738],
[0.04034896]]), array([[-0.68214222],
[-0.12580861],
[-0.94504924]]), array([[-0.63654735],
[0.14825845],
[0.11920153]]), array([[0.03495266],
[0.32078146],
[-1.2597104]]), array([[-0.64715624],
[-0.23425763],
[-0.19147764]]), array([[0.50513419],
[-0.01887497],
[1.09944658]]), array([[-0.84951272],
[0.21107611],
[-0.04916291]]), array([[0.14490243],
[-0.3319971],
[-0.68242002]]), array([[-0.74398993],

[-0.05806428],
[-0.44838545]]), array([[-0.1556952],
[0.07353481],
[0.07950969]]), array([[-0.41193988],
[0.14688177],
[0.1599685]]), array([[-0.13701005],
[-0.28348396],
[-0.60073537]]), array([[-0.66507102],
[-0.2192288],
[-1.20756454]]), array([[-0.39531024],
[0.41960668],
[-1.15668881]]), array([[-0.46413571],
[0.53412347],
[1.37856192]]), array([[-0.32918565],
[-0.01435928],
[-0.26881737]]), array([[-0.32925787],
[0.08029327],
[-1.38409418]]), array([[-0.17237612],
[0.26411906],
[0.28822611]]), array([[-0.20911129],
[-0.16524074],
[-0.5650971]]), array([[-0.38603232],
[0.16367506],
[0.54724746]]), array([[-0.46132447],
[0.29244395],
[1.15639584]]))

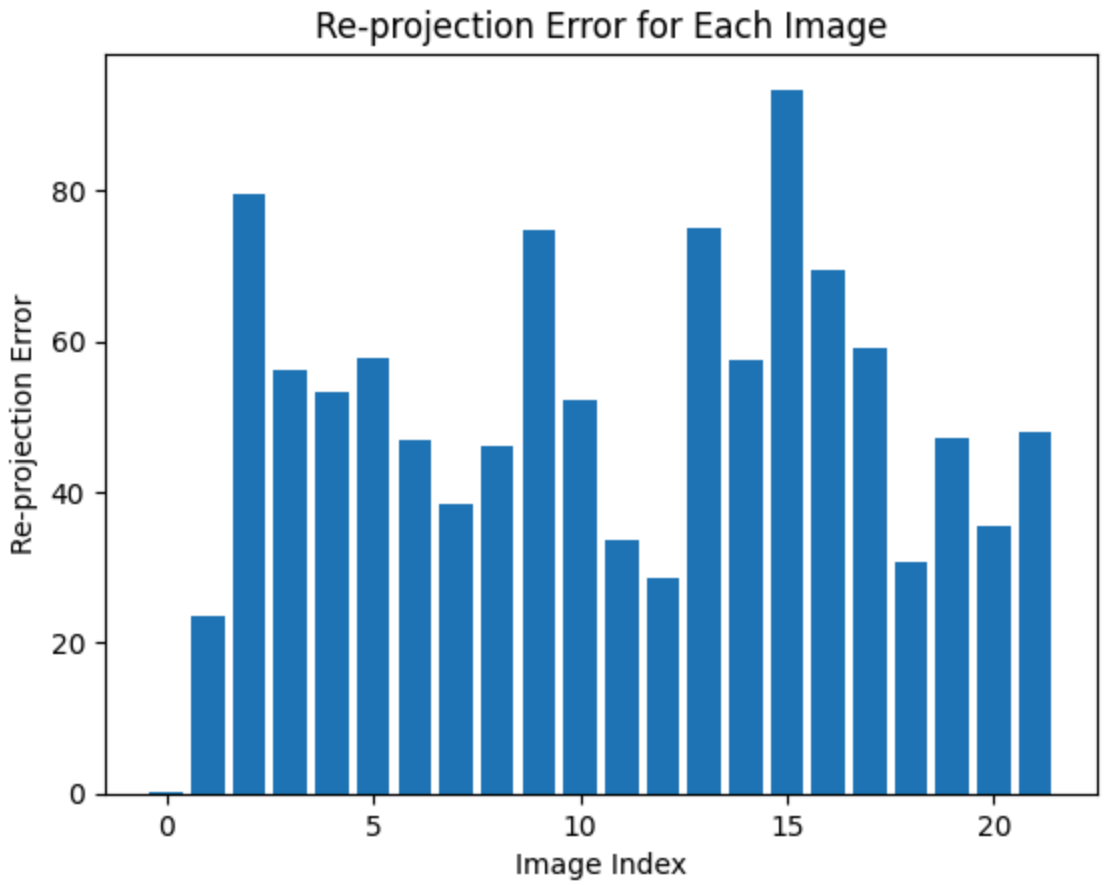
Translation Vectors :

(array([[-2.80471796],
[-2.77589874],
[13.05882777]]), array([[-1.56176054],
[-1.44817514],
[16.28567549]]), array([[-2.53403095],
[-3.21078469],
[15.27668239]]), array([[0.09666664],
[-5.46258072],
[17.60133654]]), array([[-2.88620586],
[1.59159396],
[16.8870042]]), array([[-3.50639338],
[-5.70016001],
[15.63521344]]), array([[-1.32878124],
[-0.95441963],
[18.41094912]]), array([[-2.55428721],

[-3.68320915],
[17.18085681]]), array([[-1.20084813],
[0.44646397],
[16.36972745]]), array([[-1.49006833],
[-4.12995077],
[14.11556635]]), array([[-1.88007881],
[-7.40508044],
[21.28970616]]), array([[-2.7534497],
[-5.06732916],
[18.02437724]]), array([[-3.75583676],
[-0.34824172],
[14.9660771]]), array([[-3.60322078],
[-1.54284327],
[13.60403869]]), array([[-2.26775704e+00],
[-4.76700475e-03],
[1.85364552e+01]]), array([[5.63411074],
[-5.81582925],
[14.01846857]]), array([[-0.26191195],
[-5.83671841],
[17.2260248]]), array([[-3.73931529],
[0.95793254],
[15.98905205]]), array([[-1.71489348],
[-5.15296572],
[15.53565153]]), array([[-2.23445619],
[-2.91813106],
[16.20418927]]), array([[-0.08914462],
[-5.11303884],
[18.07762771]]), array([[2.51785031],
[-3.68951151],
[17.79691819]]))

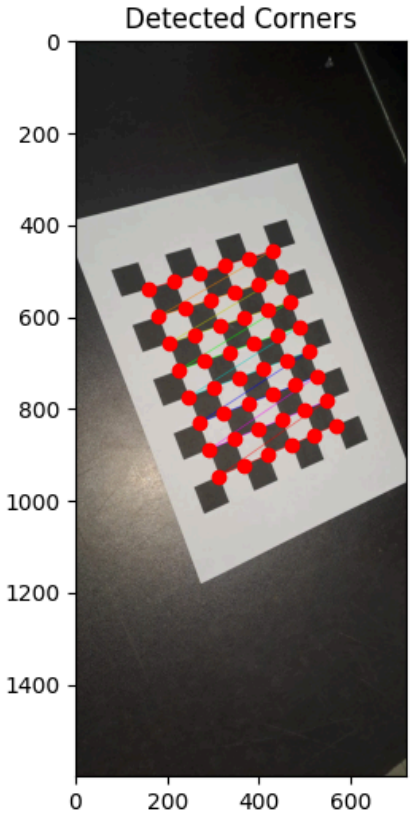
Reprojection Error:

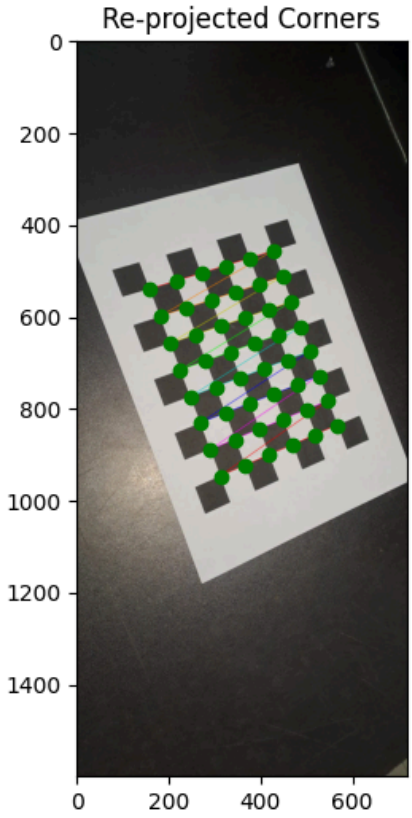
The reprojection error is computed by first detecting corner points in the original image. Then, using the intrinsic and extrinsic camera parameters (such as camera matrix, distortion coefficients, rotation, and translation), these corner points are projected onto the image plane to obtain their corresponding points in the image. The distance between the original corner points and their re-projected counterparts is computed. This distance represents the reprojection error.

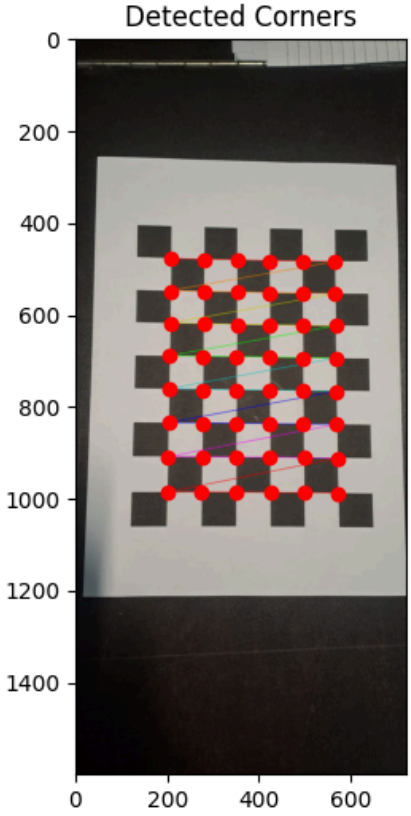


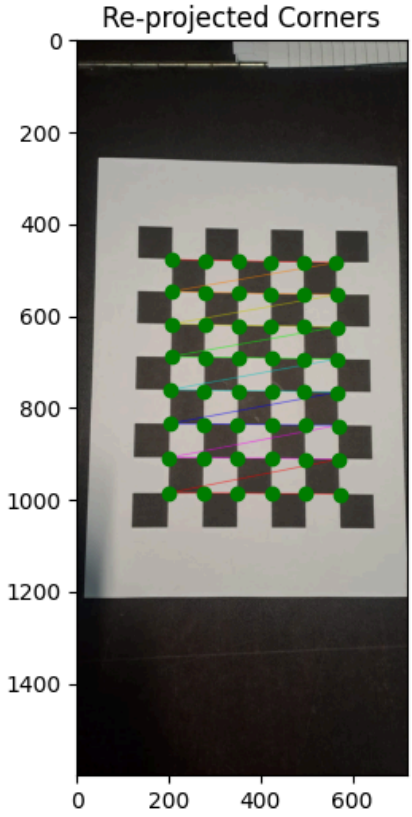
Mean Re-projection Error: 50.327528049537904
Standard Deviation of Re-projection Error: 20.566748545317726

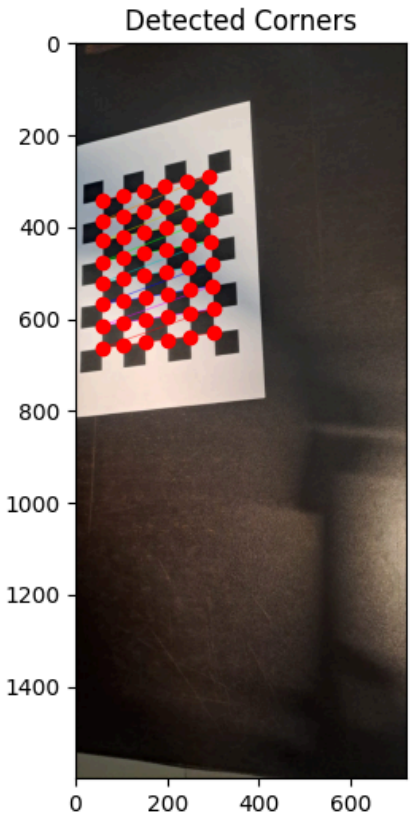
Reprojection Errors:
[0.09193955793220104, 72.19513300785889, 40.217962867933714,
57.8112256042881, 29.160971477238473, 66.20442842909578,
98.33158324732567, 63.63597910030933, 45.09508153927215, 46.7024693847049,
63.96402759405757, 73.11633552706421, 55.17347350812835,
37.66782079199279, 22.821434045899178, 90.76362113166029,
51.20689161761862, 32.012552856018544, 75.50521094149302,
38.45602469589908, 73.50850157981795, 81.92015064367003]

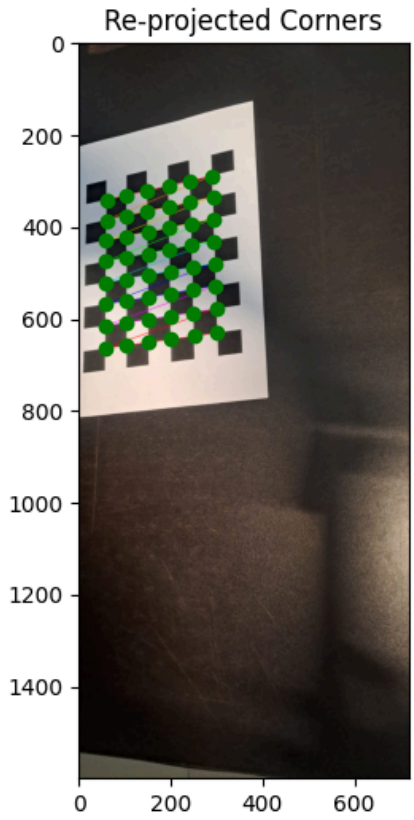

Detected Corners
0
200
400
600
800
1000
1200
1400
0
200
400
600

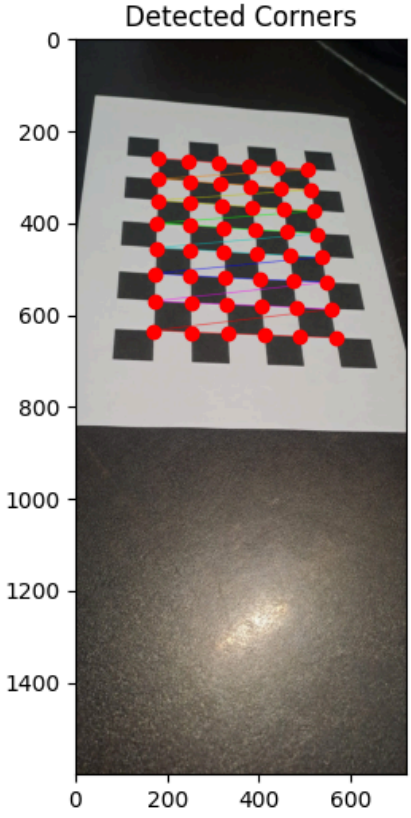

Re-projected Corners
0
200
400
600
800
1000
1200
1400
0
200
400
600

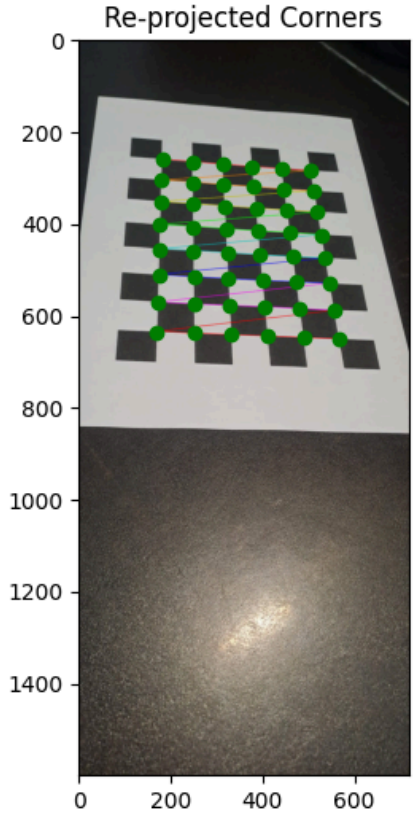

Detected Corners
0
200
400
600
800
1000
1200
1400
0
200
400
600

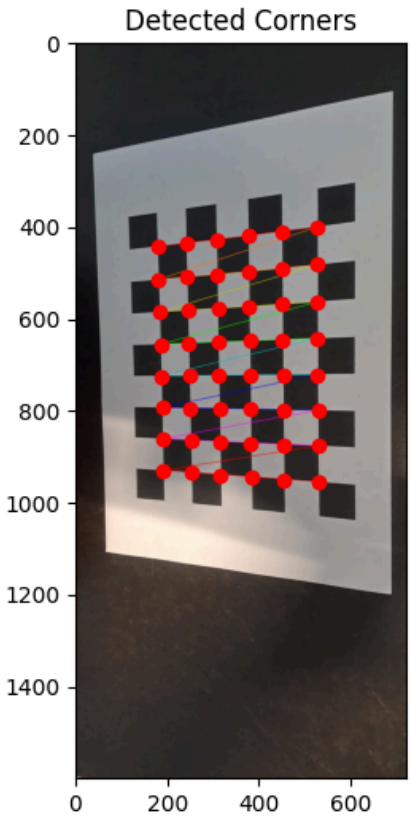

Re-projected Corners
0
200
400
600
800
1000
1200
1400
0
200
400
600

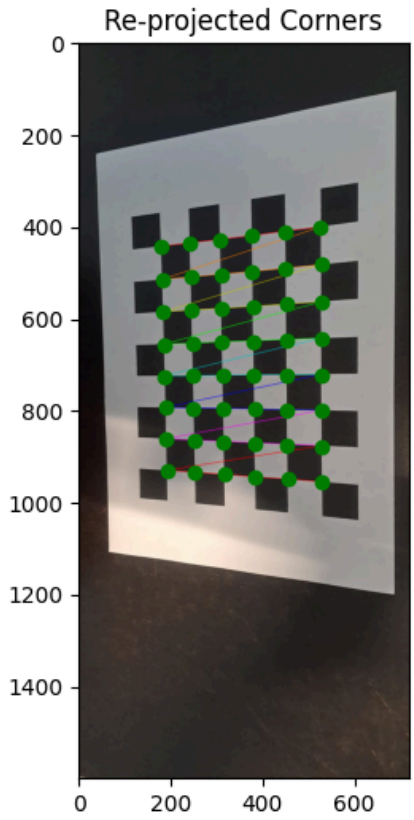

Detected Corners
0
200
400
600
800
1000
1200
1400
0
200
400
600


Re-projected Corners
0
200
400
600
800
1000
1200
1400
0
200
400
600


Detected Corners
0
200
400
600
800
1000
1200
1400
0
200
400
600

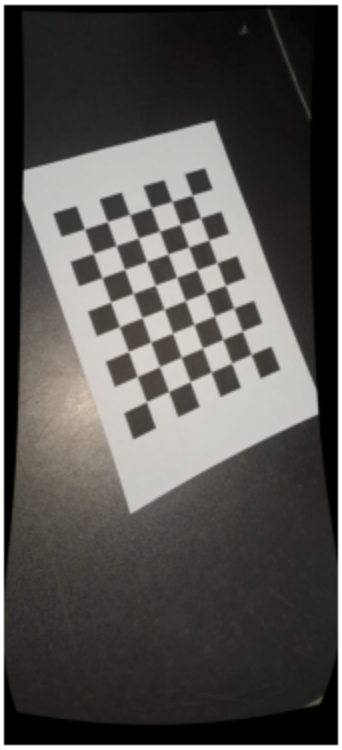

Re-projected Corners
0
200
400
600
800
1000
1200
1400
0
200
400
600





Distorted Images:

Undistorted Img 1



Undistorted Img 2

Undistorted Img 3

Undistorted Img 4

Undistorted Img 5

Checkerboard Plane Normals:

```
Image 1 - Checkerboard Plane Normal: [0.12295971 0.03682857 0.99172807]
Image 2 - Checkerboard Plane Normal: [-0.23477852  0.18012255  0.95521459]
Image 3 - Checkerboard Plane Normal: [0.18815598 0.58923671 0.78574641]
Image 4 - Checkerboard Plane Normal: [0.10102657 0.59925926 0.79415488]
Image 5 - Checkerboard Plane Normal: [ 0.21863563 -0.20103364  0.95487378]
Image 6 - Checkerboard Plane Normal: [-0.15546696  0.61498755  0.77305908]
Image 7 - Checkerboard Plane Normal: [ 0.23081671 -0.39974375  0.88708995]
Image 8 - Checkerboard Plane Normal: [0.20463467 0.73992636 0.64080702]
Image 9 - Checkerboard Plane Normal: [-0.3469711  -0.02315412  0.93758996]
Image 10 - Checkerboard Plane Normal: [0.10552212 0.66572036 0.73870257]
Image 11 - Checkerboard Plane Normal: [0.06692365 0.1576781  0.9852202 ]
Image 12 - Checkerboard Plane Normal: [0.10927272 0.40874915 0.90608146]
Image 13 - Checkerboard Plane Normal: [-0.22264404  0.2086794
0.95230381]
Image 14 - Checkerboard Plane Normal: [0.18578872 0.58139356 0.7921263 ]
Image 15 - Checkerboard Plane Normal: [0.51069655 0.08337296 0.85570905]
Image 16 - Checkerboard Plane Normal: [0.08380033 0.59960333 0.79589783]
Image 17 - Checkerboard Plane Normal: [0.02965226 0.32125484 0.94652843]
Image 18 - Checkerboard Plane Normal: [0.24757056 0.18183111 0.95165449]
Image 19 - Checkerboard Plane Normal: [0.23169036 0.20466682 0.95101581]
Image 20 - Checkerboard Plane Normal: [-0.097511    0.24095872
0.96562441]
Image 21 - Checkerboard Plane Normal: [0.04950451 0.3992107  0.91552177]
Image 22 - Checkerboard Plane Normal: [-0.01331076  0.49272461
0.87008349]
```

Q5

a. We have R_c & T_c present for each image
We also have n^c_i (normal) for each camera image

Using R_c & T_c we can find the corresponding world point 'Cp' say using the transformation matrix on an arbitrary point say the origin

Now this Cp should lie on the camera plane, thus satisfy the normal eqn

$$n^c_i{}^T C_{pi} + \alpha^c_i = 0$$

↳ offset

∴ We calculate the offsets for each image aswell

⇒ We obtained n^c_i & α^c_i using SVD on the p'cd pts

→ Now to obtain t^a or translation vector b/w camera & lidar

$$t^a = \underset{t}{\operatorname{argmin}} \sum_i^{3n} \left(\alpha^a_i - (\alpha_{ci} - n^{c^T}_i t)\right)^2$$

or $$t^a = \underset{t}{\operatorname{argmin}} \| n^{c^T}_i t - (\alpha^a - \alpha_c) \|^2_F$$

closed form: $t_1 = (n^c n^{c^T})\, n^c (\alpha^c - \alpha^a)$

↳ maximise sum of cos of angles i.e. min angular diff from origin b/w LIDAR & img

→ $$R^a = \max_R \operatorname{tr}(O^{c^T} R^c n^c) = \max_R \operatorname{tr}(R n^a n^{c^T})$$

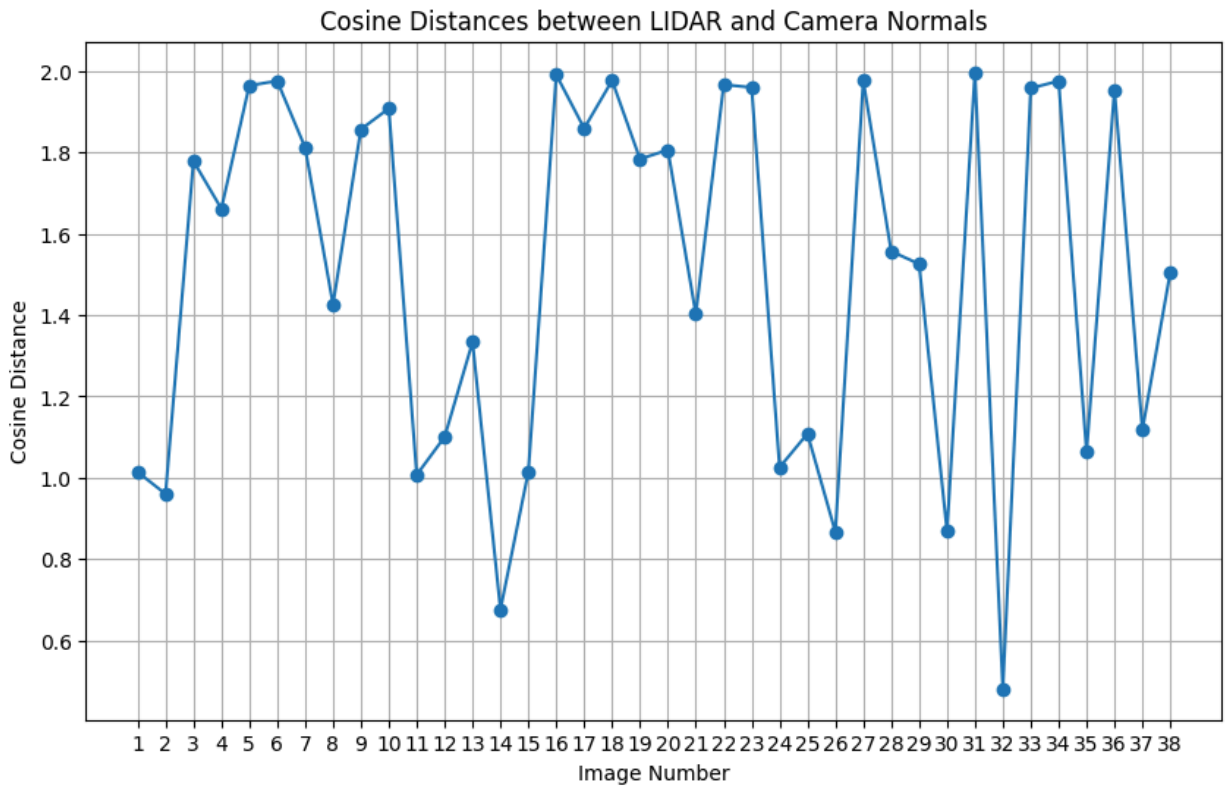
closed form $R = V U^T$

where $n^a n^{c^T} = U S V^T$

↳ SVD of mul of normals

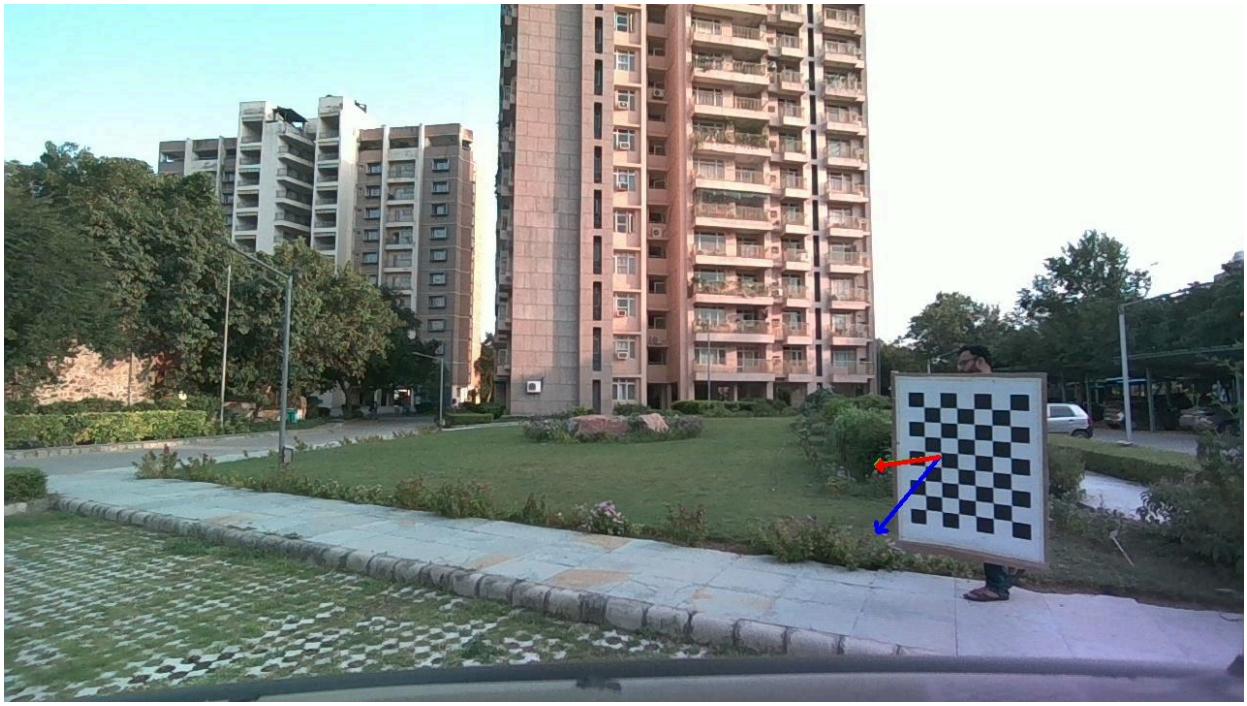
VISION

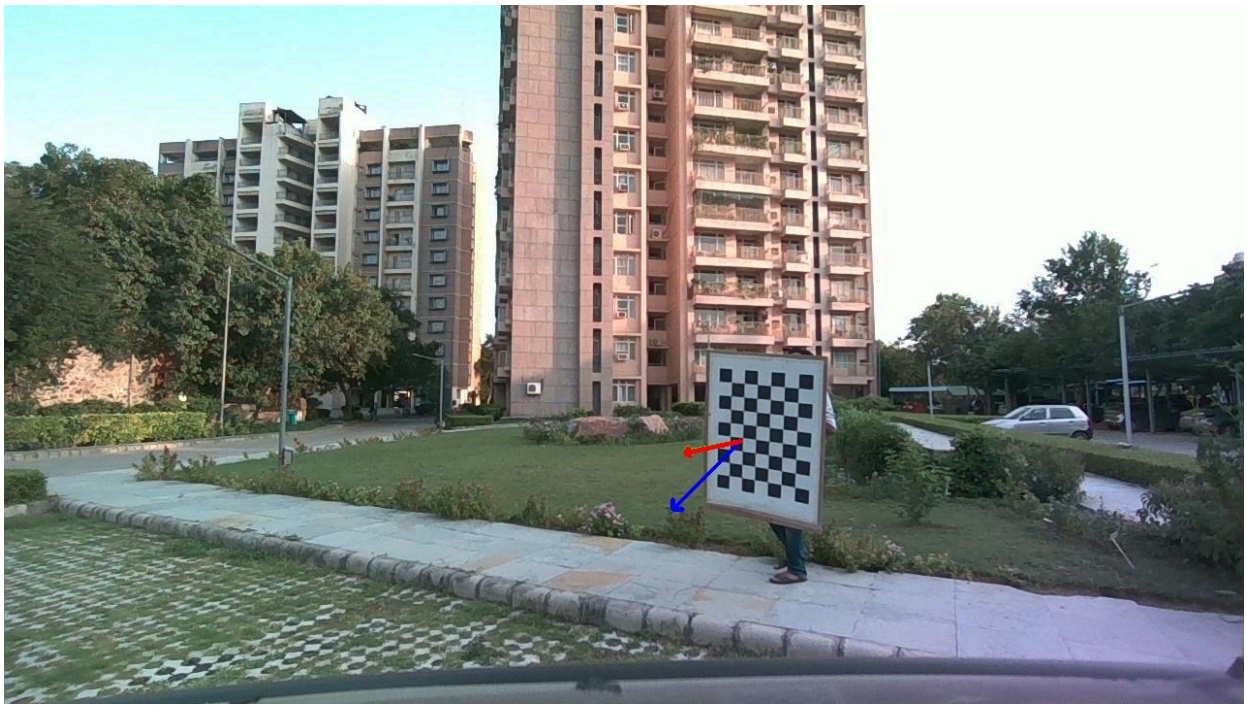
Cosine Distances:

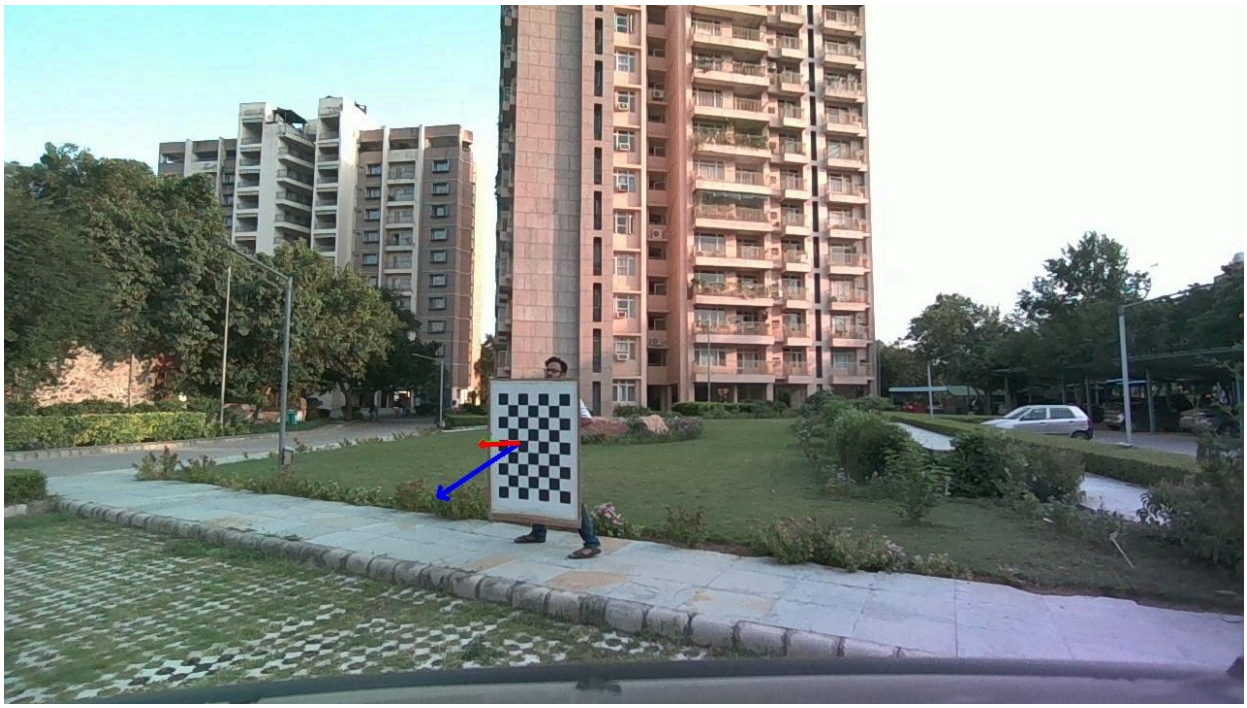


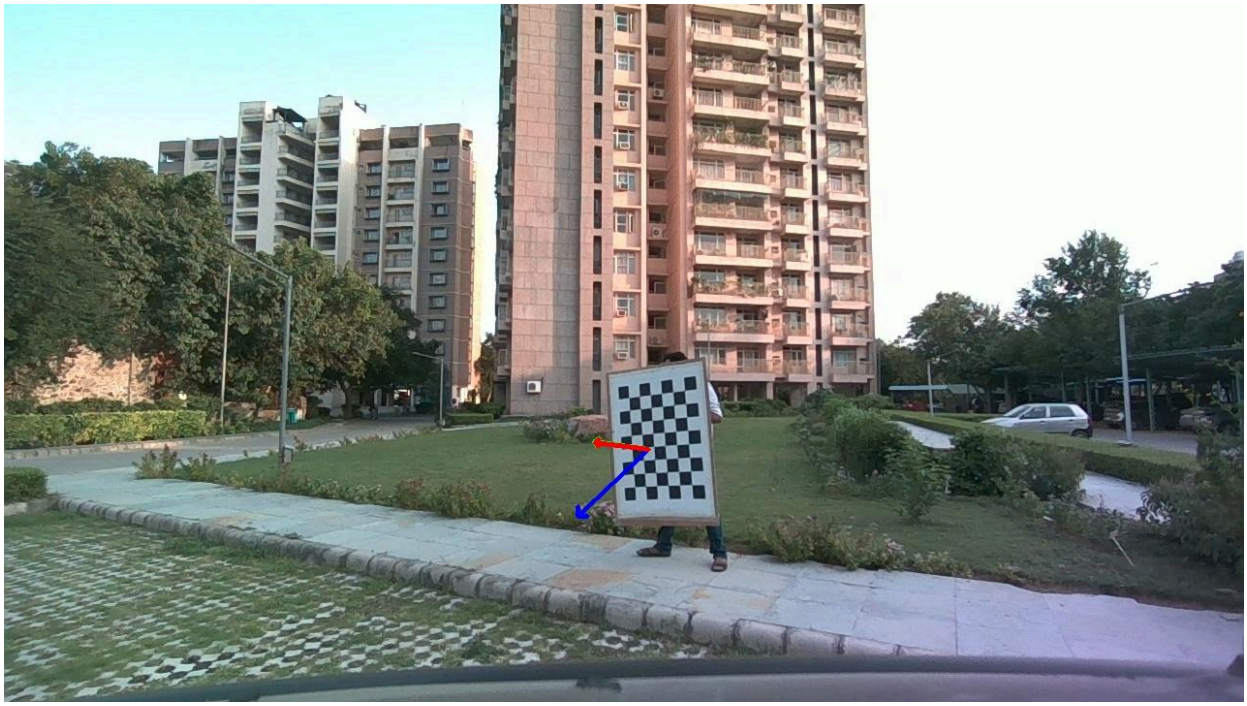
Average Error: 1.505648917105319
Standard Deviation: 0.45022460321767743

Plot Lidar, Camera and Estimated Normals :









Projection Points: