CV Assignment 2

Vikranth Udandarao
2022570

Theory

CV Theory

NS = N20225 70

1- $V = \begin{bmatrix} 3 \\ -1 \\ 4 \end{bmatrix}$

Step 1 - Rotation of $-\frac{\pi}{6}$ about y axis -

A rotation about Y-axis by an angle θ has the matrix -

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$\theta = \frac{-\pi}{6}$

$\rightarrow \cos\frac{-\pi}{6} = \frac{\sqrt{3}}{2}, \quad \sin\frac{-\pi}{6} = \frac{-1}{2}$

$$R_y\left(\frac{-\pi}{6}\right) = \begin{bmatrix} \frac{\sqrt{3}}{2} & 0 & \frac{-1}{2} \\ 0 & 1 & 0 \\ \frac{1}{2} & 0 & \frac{\sqrt{3}}{2} \end{bmatrix}$$

Step 2

Rotation $\frac{\pi}{4}$ about x axis -

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_x\left(\frac{\pi}{4}\right) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

Step 3 - Reflection about XZ plane -

$$x \to x$$
$$y \to -y$$
$$z \to z$$

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

$$\Rightarrow V' = \underbrace{S_{xz} \cdot R_x\left(\frac{\pi}{4}\right) \cdot R_y\left(-\frac{\pi}{6}\right)}_{M} \cdot V$$

$$R_x\left(\frac{\pi}{4}\right) \cdot R_y\left(-\frac{\pi}{6}\right) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{\sqrt{3}}{2} & 0 & \frac{1}{2} \\ 0 & 1 & 0 \\ \frac{1}{2} & 0 & \frac{\sqrt{3}}{2} \end{bmatrix}$$

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

$$M = S_{y2} \cdot \left(R_y\left(\frac{\pi}{4}\right) \cdot R_y\left(-\frac{\pi}{6}\right) \right)$$

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

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

1-2- $V = M \cdot V + t$ $t = \begin{bmatrix} 1 \\ 0 \\ -2 \end{bmatrix}$

$$M - V = \begin{bmatrix} \frac{\sqrt{3}}{2} & 0 & -\frac{1}{2} & 1 \\ \frac{1}{2\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2\sqrt{2}} & 0 \\ \frac{1}{2\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2\sqrt{2}} & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ -1 \\ 4 \\ 1 \end{bmatrix}$$

$$M \cdot V = \begin{bmatrix} \frac{3\sqrt{3} - 4}{2} \\ \frac{5\sqrt{3} + 4\sqrt{6}}{4} \\ \frac{\sqrt{2} + 4\sqrt{6}}{4} \\ 1 \end{bmatrix}$$

$$\rightarrow V' = \quad + \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$$

$$\rightarrow V' = \begin{bmatrix} \frac{3\sqrt{3} - 2}{2} \\ \frac{5\sqrt{2} + 4\sqrt{6}}{4} \\ \frac{\sqrt{2} + 4\sqrt{6} - 8}{4} \\ 1 \end{bmatrix}$$

$$o = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$v = m \cdot o + t$$

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

1-3. $R = R_y\left(\frac{\pi}{4}\right) \cdot R_y\left(-\frac{\pi}{6}\right)$, $tr(R) = 1 + 2\cos\theta$

$$= \frac{\sqrt{3}}{2} + \frac{\sqrt{2}}{2} + \frac{\sqrt{6}}{4}$$

$$\rightarrow 1 + 2\cos\theta = \frac{\sqrt{3}}{2} + \frac{\sqrt{2}}{2} + \frac{\sqrt{6}}{4}$$

$$\rightarrow \cos\theta \approx 0.5925$$

$$\rightarrow \theta = \cos^{-1}(0.5925)$$

$$= 53.7^\circ$$

Ans Axis of combined rotation in original frame of reference.

Rotation Matrix R,

$$R u = u \;\rightarrow [R - I]\, u = 0$$

$$\rightarrow u = \left(\frac{1}{\sqrt{2}} + \frac{\sqrt{3}}{2\sqrt{2}}, -\frac{1}{2} - \frac{1}{2\sqrt{2}}, -\frac{1}{2\sqrt{2}} \right)$$

1.4- Rodrigues formula

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

K is skew symmetric,

$$K = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}$$

$$\rightarrow u' = (1.32, -0.85, -0.35)$$

Normalize:

$$\|u'\| = \sqrt{(1.32)^2 + (-0.85)^2 + (-0.35)^2}$$

$$\therefore = \sqrt{2.6} = 1.6$$

$$\rightarrow u = (0.82, -0.53, -0.22)$$ a unit vector

using

$\theta = 53.7^\circ = 0.93$ rad,

$\sin\theta = \sin(0.93) \approx 0.8$

$\cos\theta = 0.6$

$1 - \cos\theta = 0.4$

$$K = \begin{bmatrix} 0 & 0.22 & -0.53 \\ -0.22 & 0 & -0.82 \\ 0.53 & 0.82 & 0 \end{bmatrix}$$

$$K^2 = \begin{bmatrix} -0.33 & -0.18 & -0.43 \\ -0.18 & -0.48 & 0.28 \\ -0.93 & 0.28 & -0.19 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.86 & 0 & -0.5 \\ -0.25 & 0.7 & -0.61 \\ 0.25 & 0.7 & 0.61 \end{bmatrix}$$

This matches our previous R

2- Image formation equation

$$= u = K [R|t] X$$

K → intrinsic parameters

[R|t] → extrinsic parameters

X → 3D point

u → image plot in homogenous system

→ The intrinsic matrix [R|t] transforms a world point into camera's coordinate.

$$X_{cam} = [R|t] X = R X_{3D} + t$$

Step1 - Defining camera coordinate frame (C)

→ The world is aligned with C's frame.

→ $u_1 = K_1 [I|0] = K_1 \begin{bmatrix} I & 0 \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$

$$= K_1 \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

→ So, $u_1 = K_1 X_{3D}$,

$$X_{3D} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}^T$$

Step 2 - Camera 2

C_2's orientation is obtained by a pure 3D rotation R applied to C_1's orientation and then no translation between the cameras.

for the same world point X -

$$u_2 = K_2 [R | 0] X$$

$$= K_2 \begin{bmatrix} R & 0 \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\rightarrow u_2 = K_2 R X_{3D}$$

Now, we need $u_1 = H u_2$.

$$\rightarrow u_1 = K_1 X_{3D}$$

$$u_2 = K_2 R X_{3D}$$

$$X_{3p} = R^{-1} K_2^{-1} u_2$$

$$\therefore u_1 = K_1 X_{3p} = K_1 (R^{-1} K_2^{-1} u_2)$$

$$= K_1 R^T \cdot R_2^{-1} u_2$$

$$u_1 = H u_2$$

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

H is a 3x3 matrix expressed in terms of K_1 (intrinsic matrix of C_1) & K_2 (intrinsic matrix of C_2) and R (Rotation from C_1 to C_2).

Coding

3. Given Images:

1. **Estimated Intrinsic Camera Parameters:**

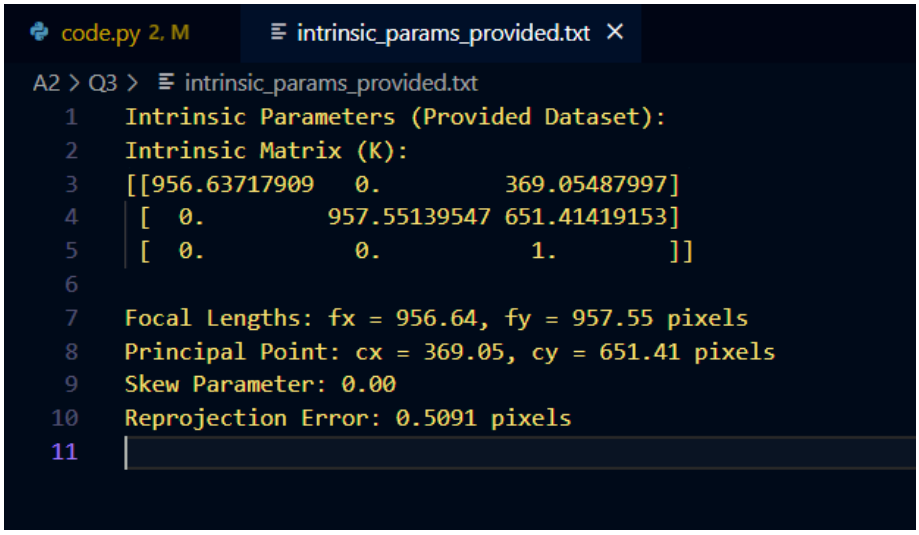
- **Focal Lengths**:
    - fx = 956.64 pixels
    - fy = 957.55 pixels
- **Principal Point**:
    - cx = 369.05 pixels
    - cy = 651.41 pixels
- **Skew Parameter**:
    - 0.00 (indicating no skew between image axes)
- **Intrinsic Matrix (K)**:

    [[956.63717909 0. 369.05487997]

    [0. 957.55139547 651.41419153]

    [0. 0. 1.]]

- **Reprojection Error**:
    - 0.5091 pixels (indicating the average discrepancy between observed and projected points)



2. Estimated Extrinsic Camera Parameters

Image 1

- **Rotation Matrix (R):**

[[0.99699847 0.01387206 -0.07616832]

[-0.02511535 0.9885627 -0.14870444]

[0.07323433 0.1501711 0.9859439]]

- **Translation Vector (t):**

[[-118.14885488]

[-108.51488489]

[439.77334073]]

Image 2

- **Rotation Matrix (R):**

[[0.99828333 0.05391999 -0.02286982]

[-0.05069018 0.99100787 0.12383014]

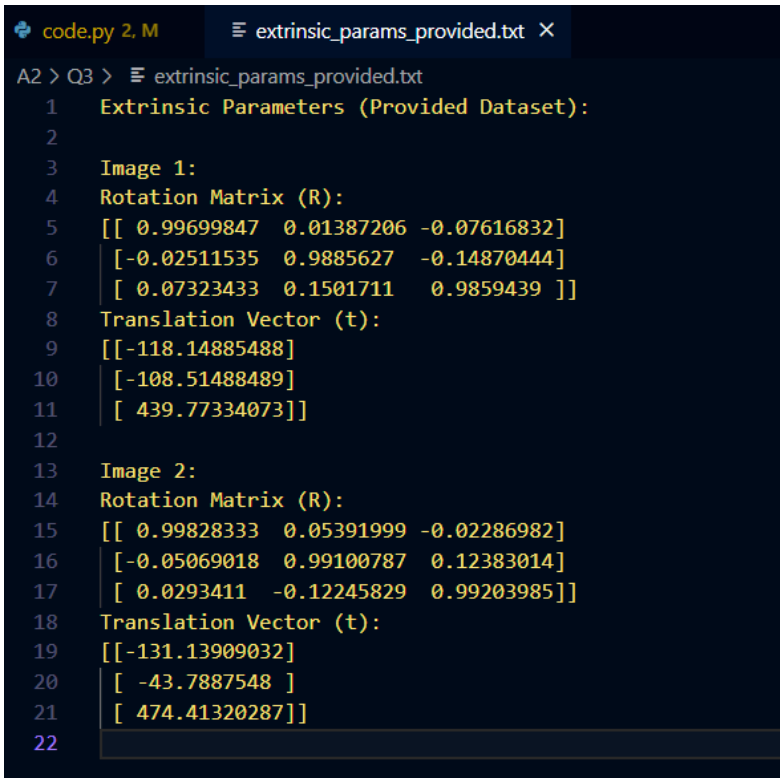
[0.0293411 -0.12245829 0.99203985]]

- **Translation Vector (t):**

[[-131.13909032]

[-43.7887548]

[474.41320287]]



3. **Radial Distortion Coefficients and Undistortion Results**

The estimated **radial distortion coefficients** obtained from camera calibration are:

- **k_1 = 0.1976**
- **k_2 = -0.7069**
- **k_3 = 0.0488**

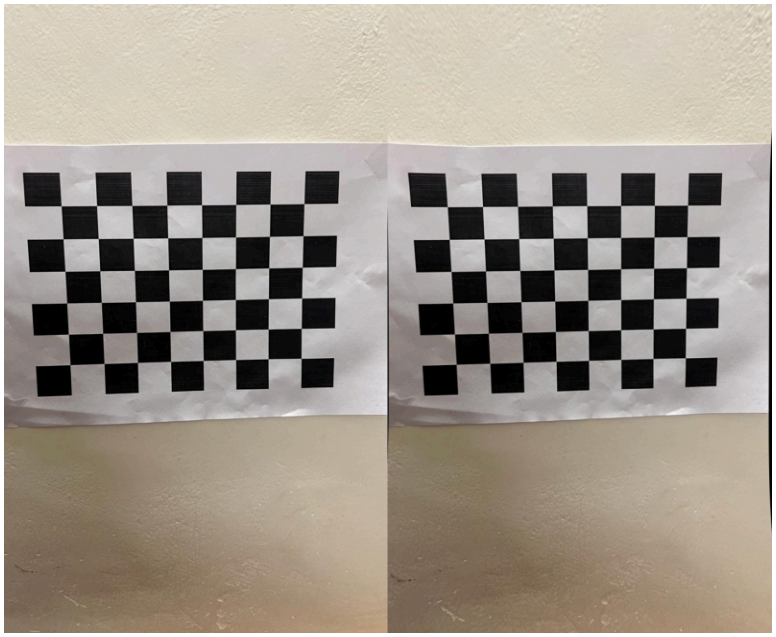
In addition to radial distortion, the tangential distortion coefficients are:

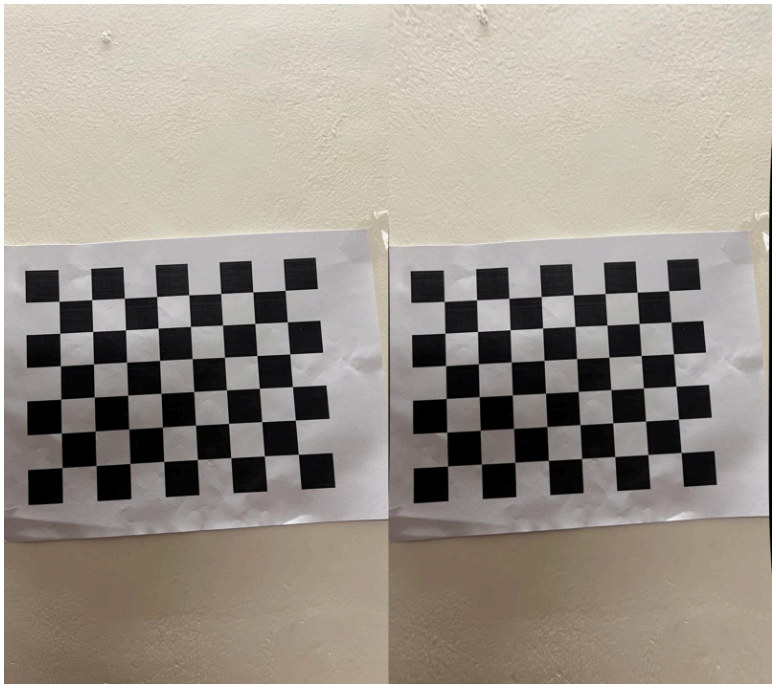
- **p_1 = 0.0034**
- **p_2 = 0.0070**

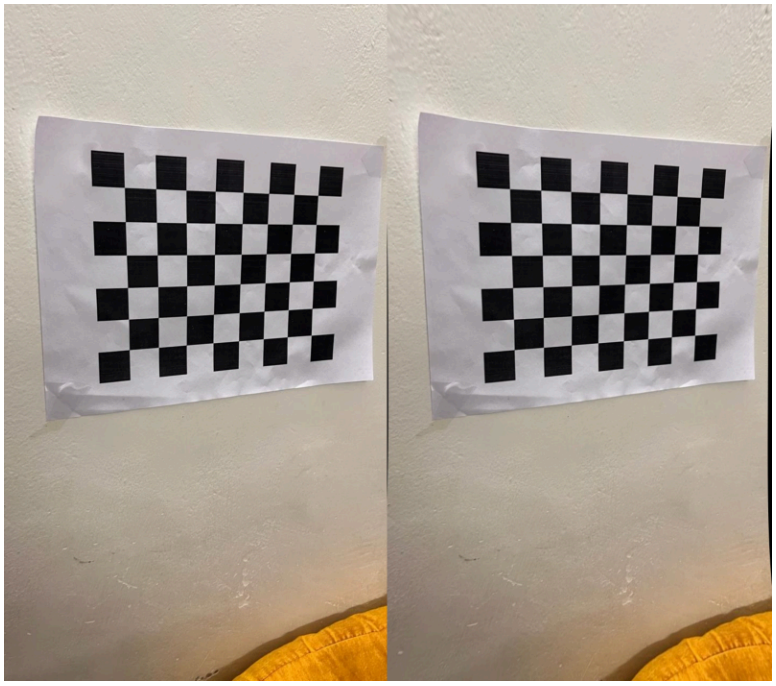
These coefficients were used to **undistort five raw images** of the calibration checkerboard. The undistortion was performed using OpenCV's `cv2.undistort()` function, applying the estimated intrinsic matrix and distortion coefficients.

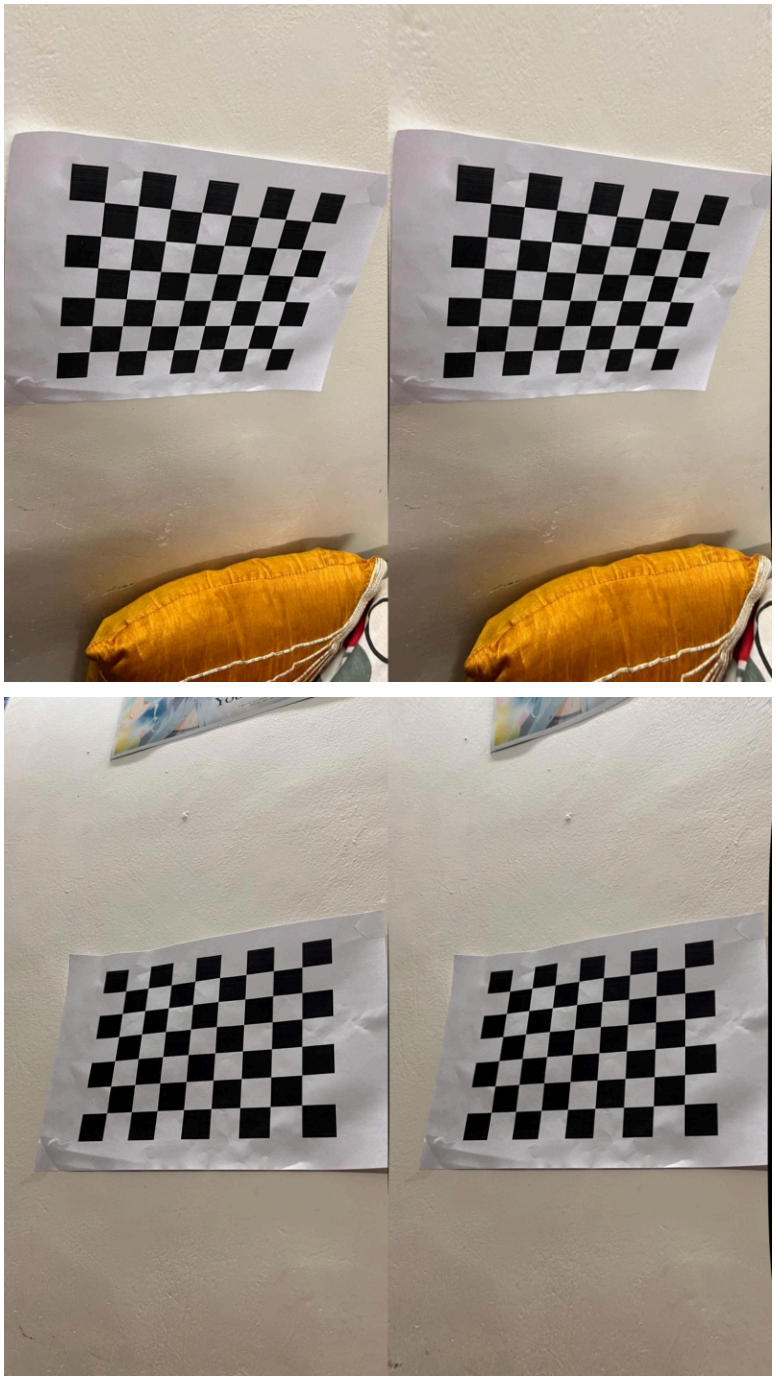
Visual Results

The raw and undistorted images are shown side-by-side below:









Observation

Upon examining the undistorted images:

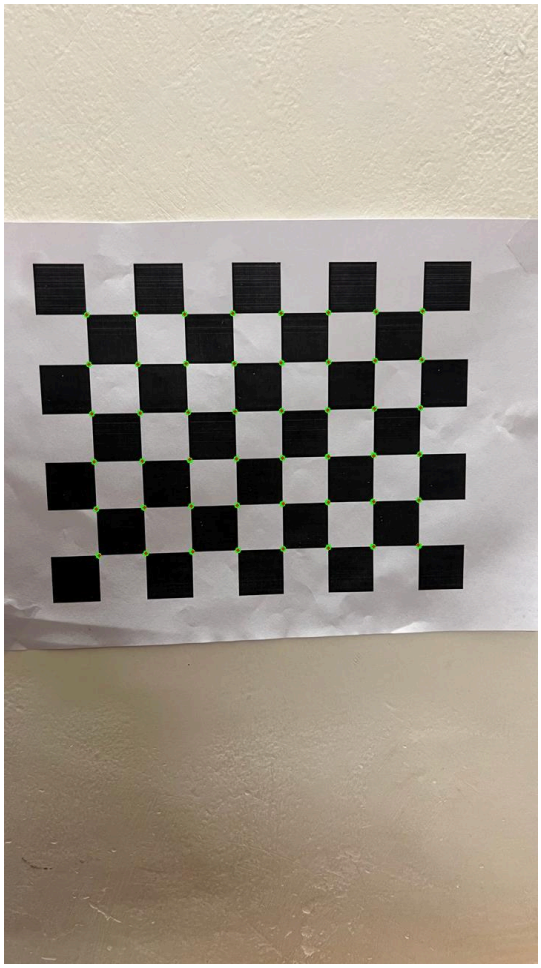
- The **curved lines near the image corners** in the raw images, caused by **barrel distortion**, were successfully straightened.

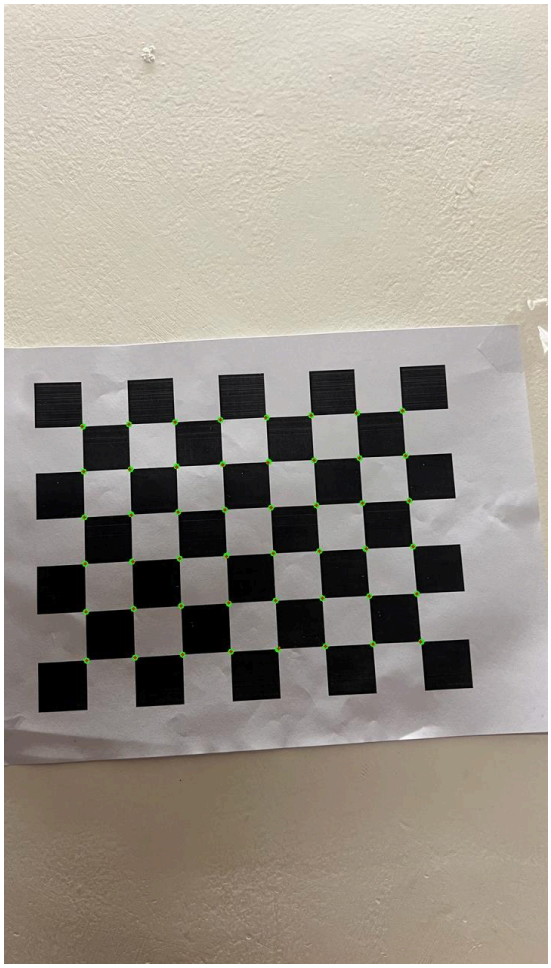
- The checkerboard squares near the edges, which appeared **warped outward**, now appear **rectangular and properly aligned**.

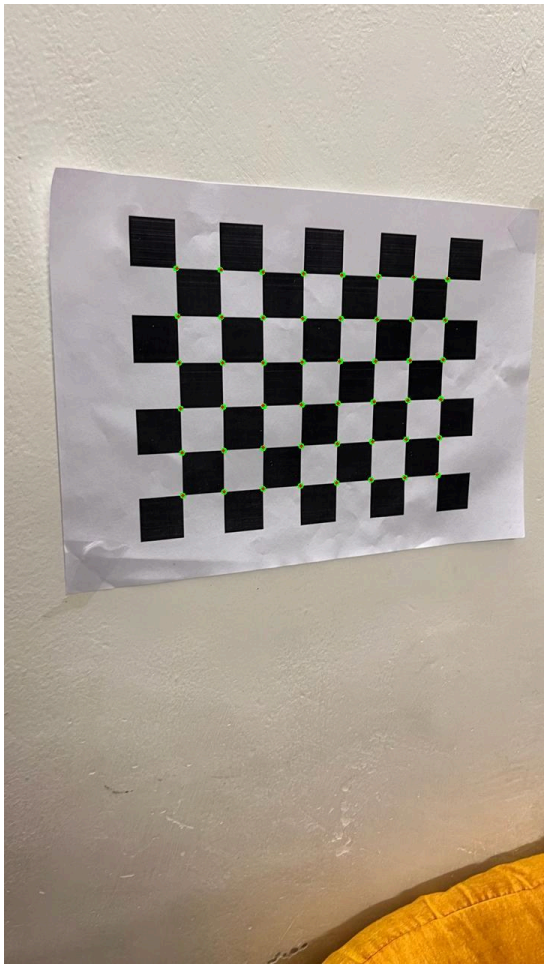
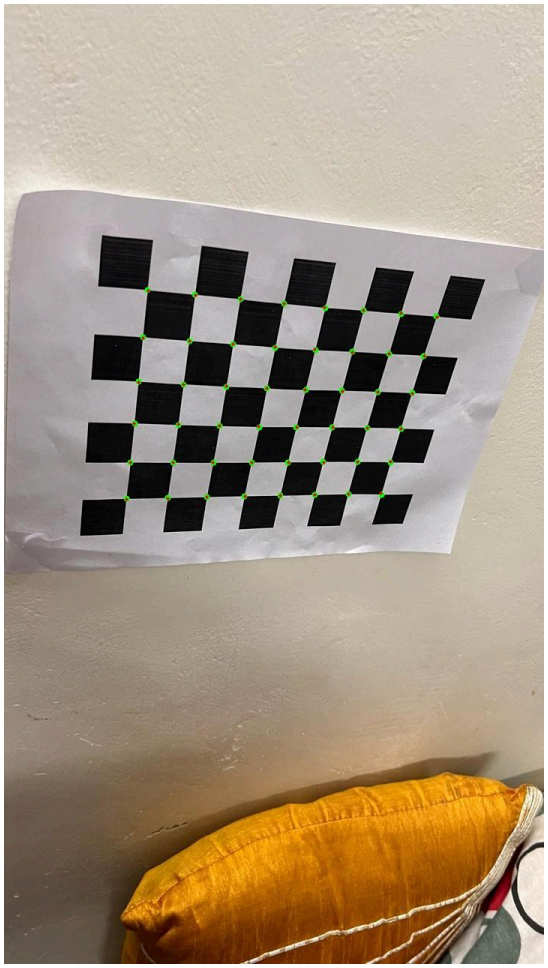
- This confirms that the distortion correction removed the radial distortion effectively, resulting in a more geometrically accurate image projection.

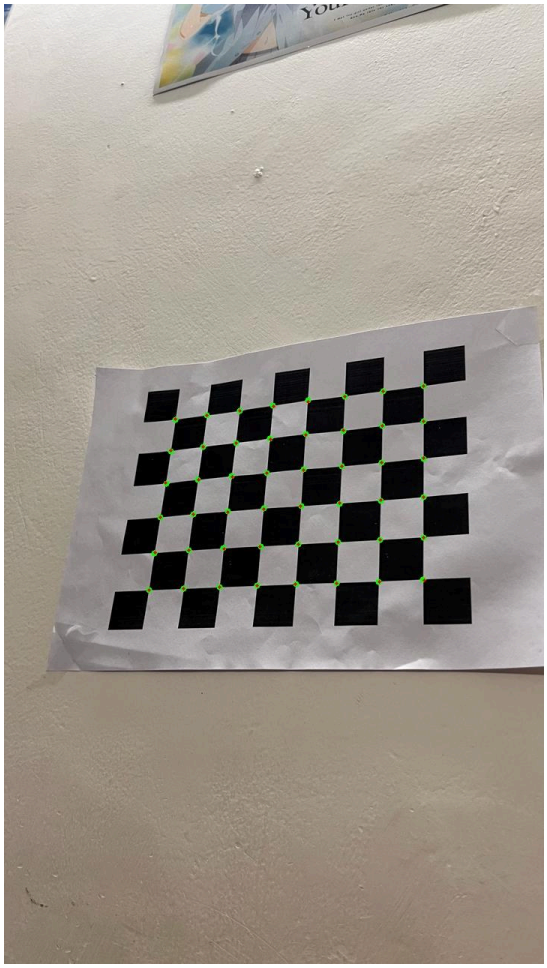
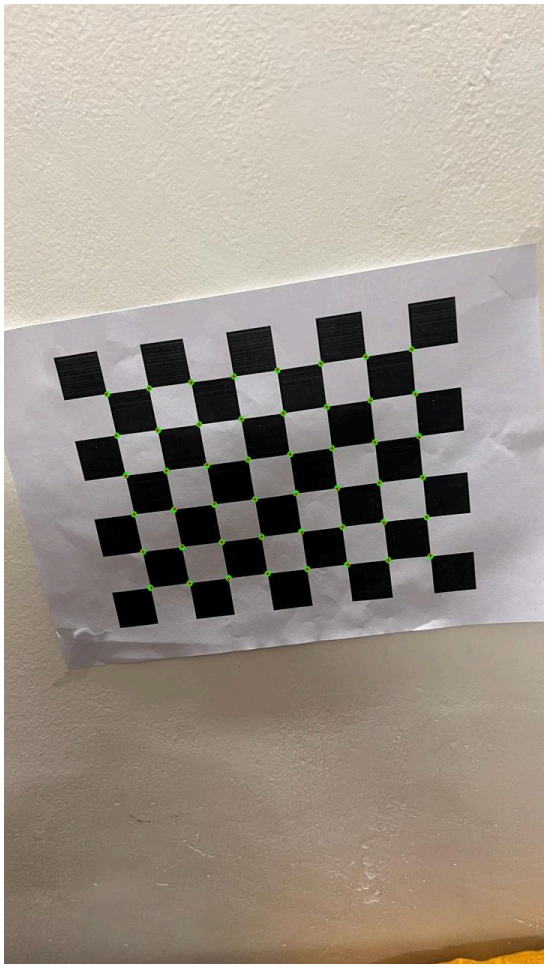
4. **Reprojection Error Analysis**

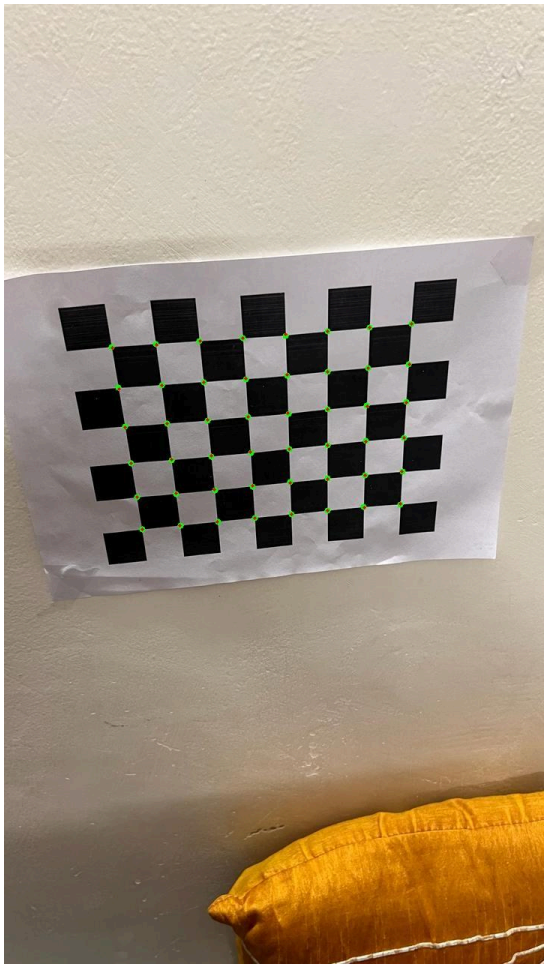
The **reprojection error** quantifies the difference between the actual detected image points and the projected image points (obtained by projecting known 3D world points using the estimated intrinsic and extrinsic parameters). It is a key metric used to evaluate the accuracy of camera calibration.

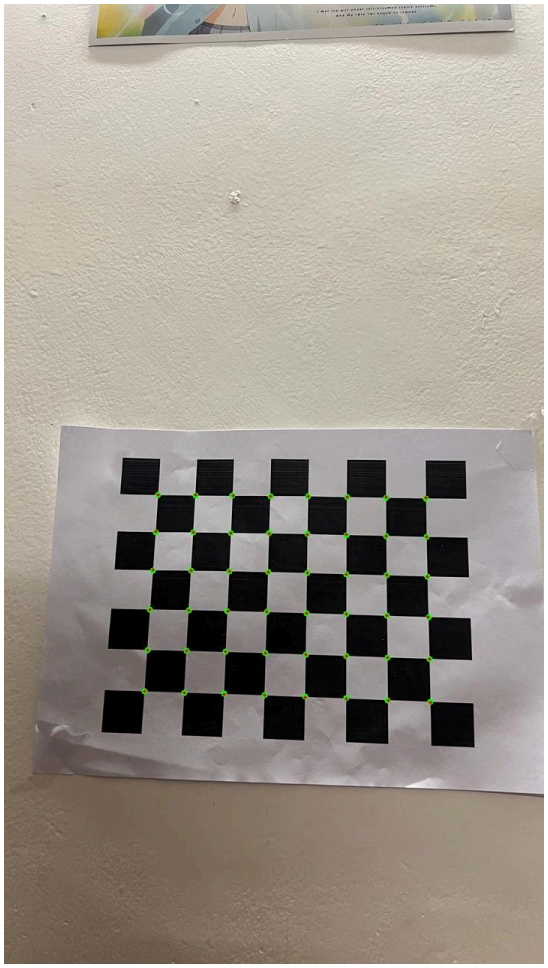


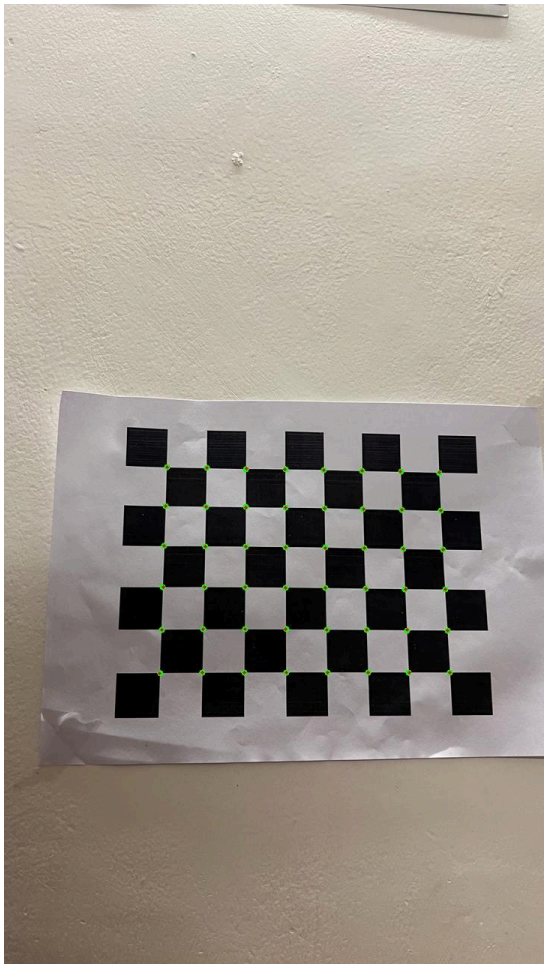
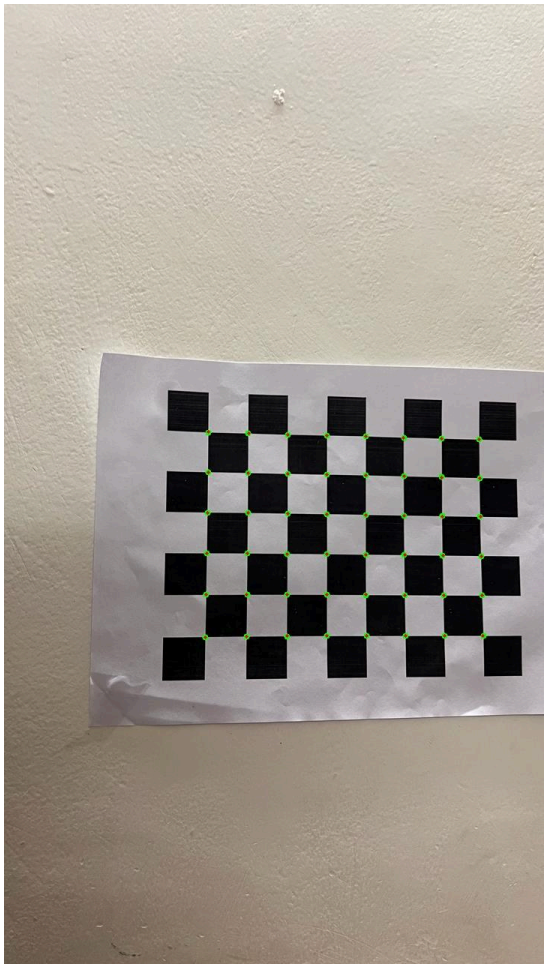


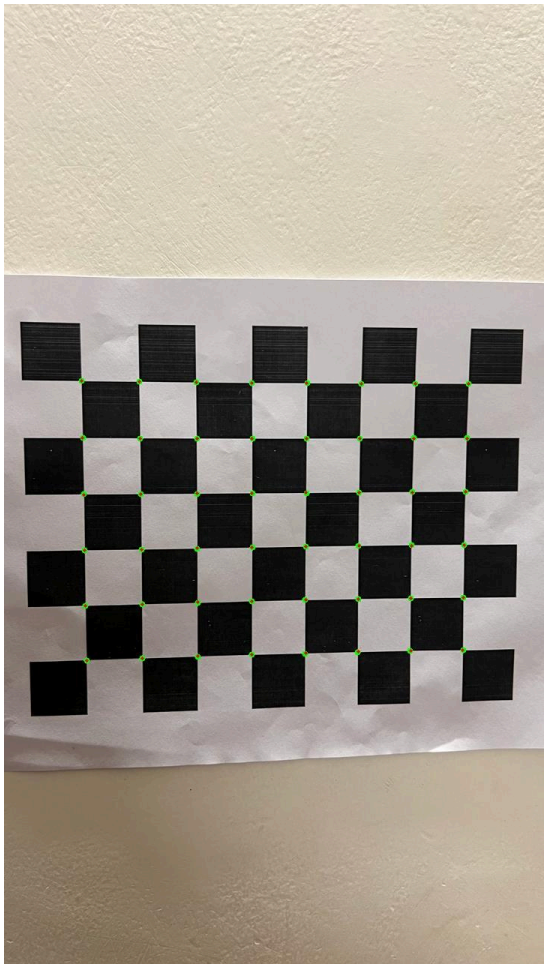



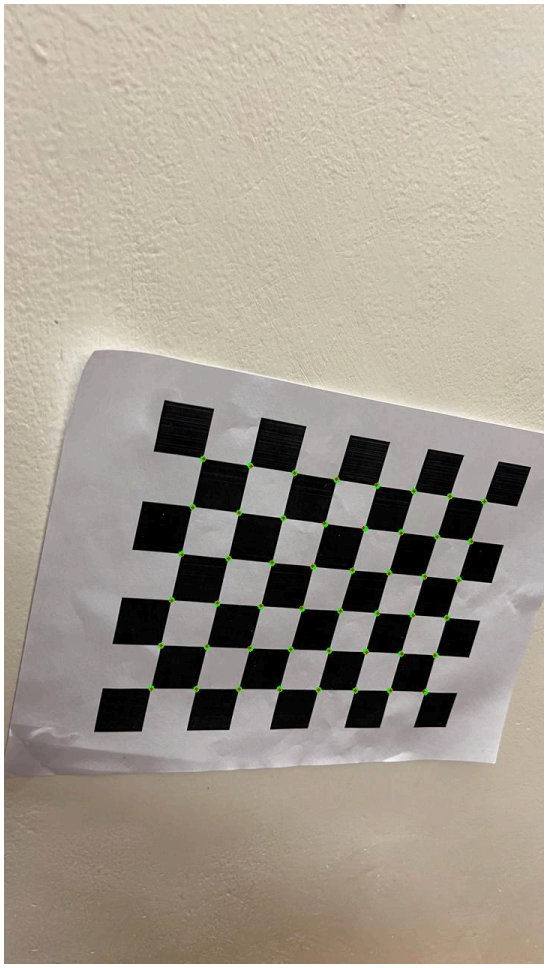



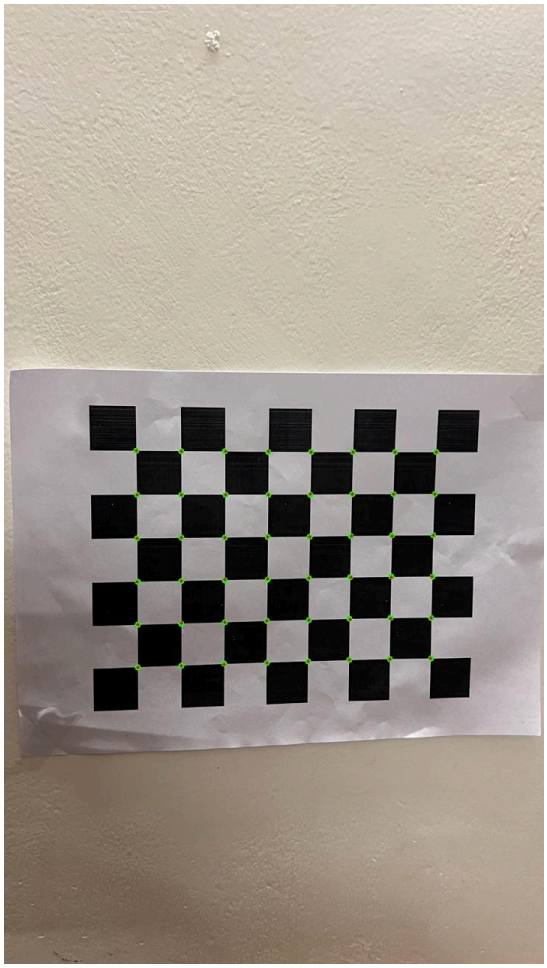


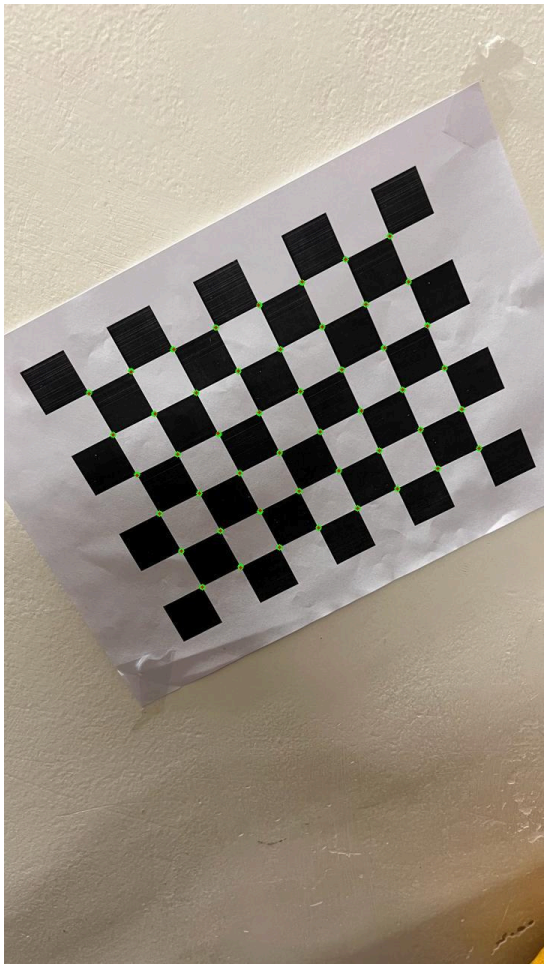


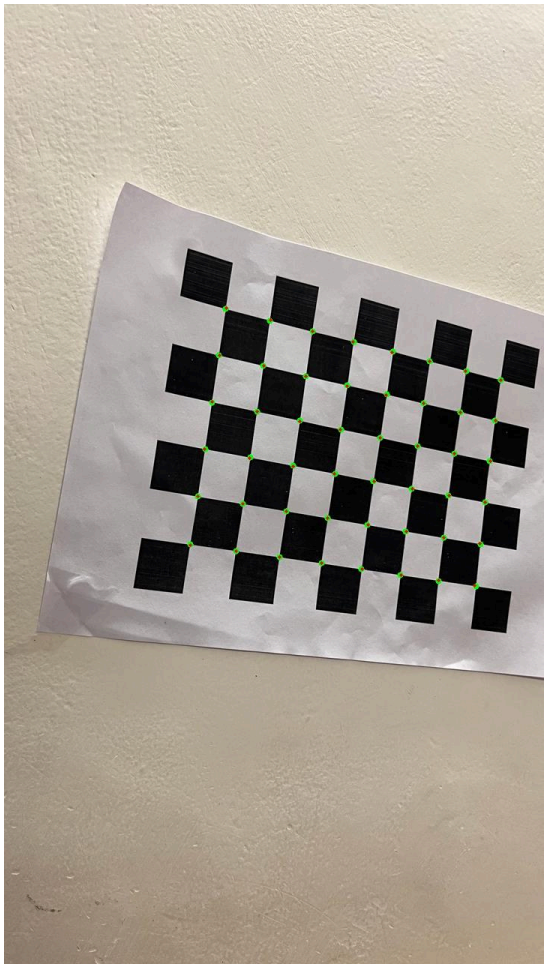



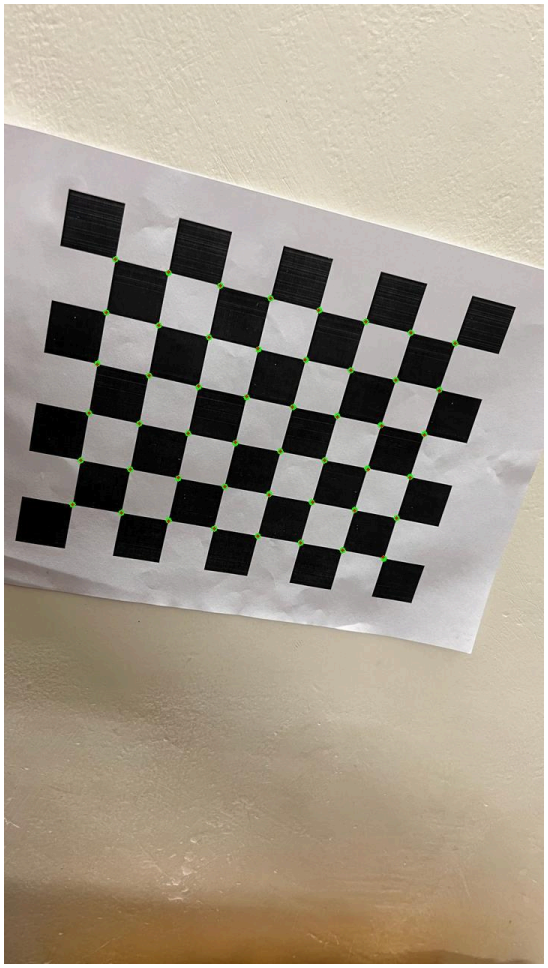


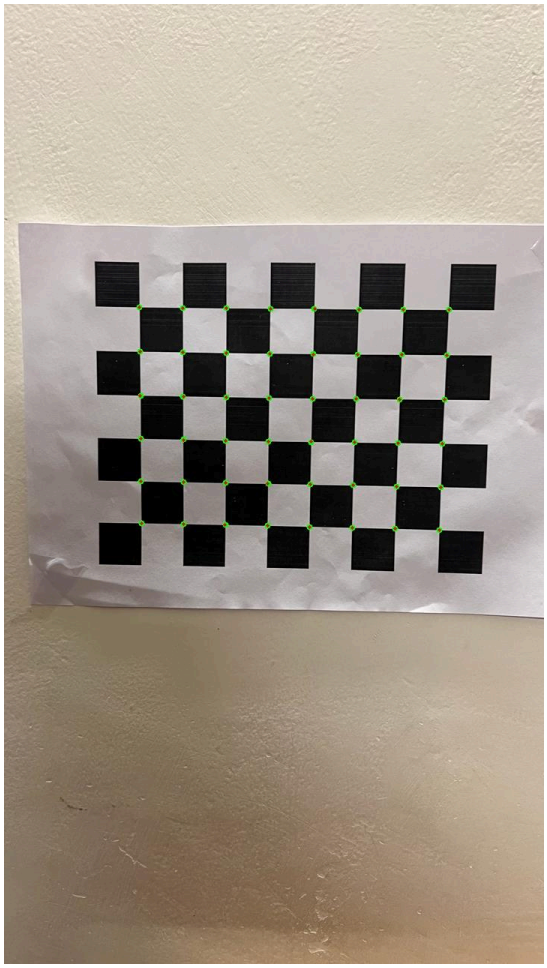


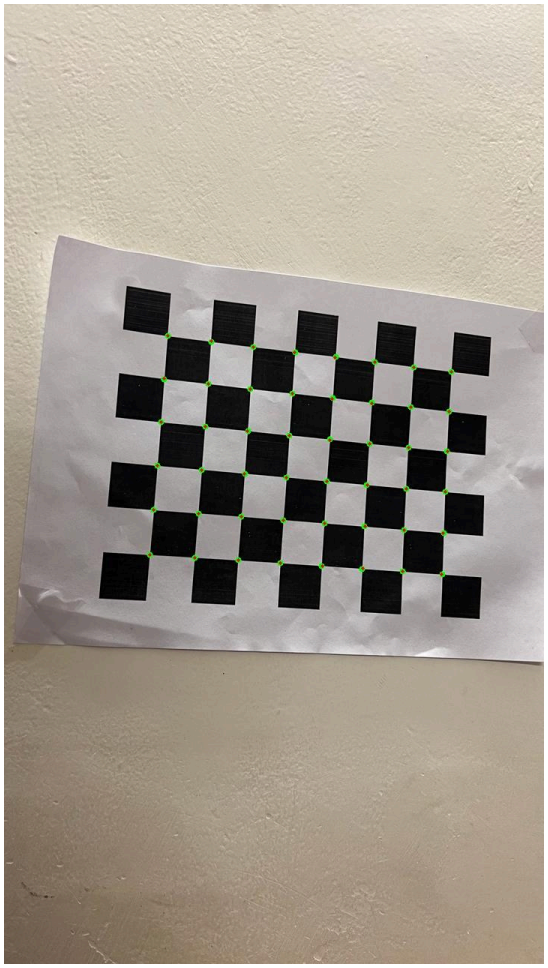


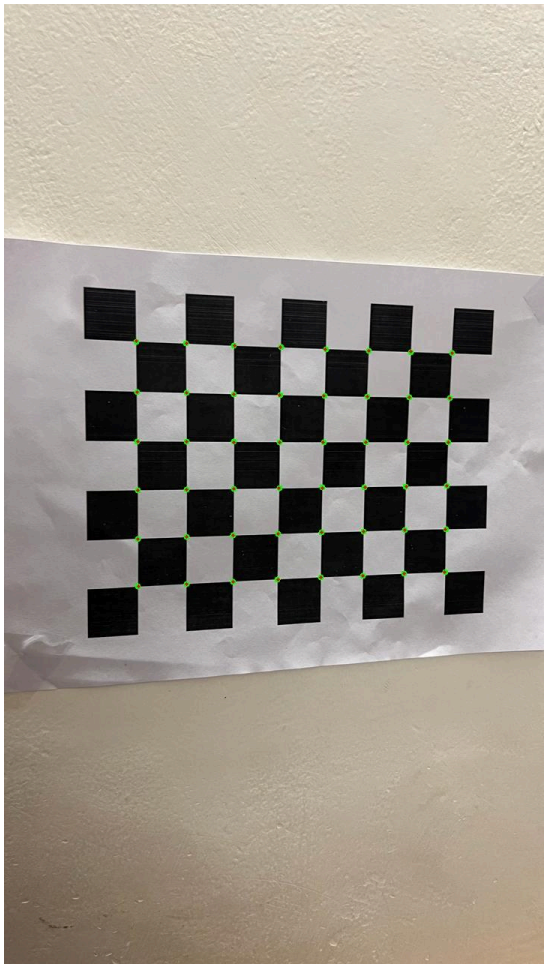


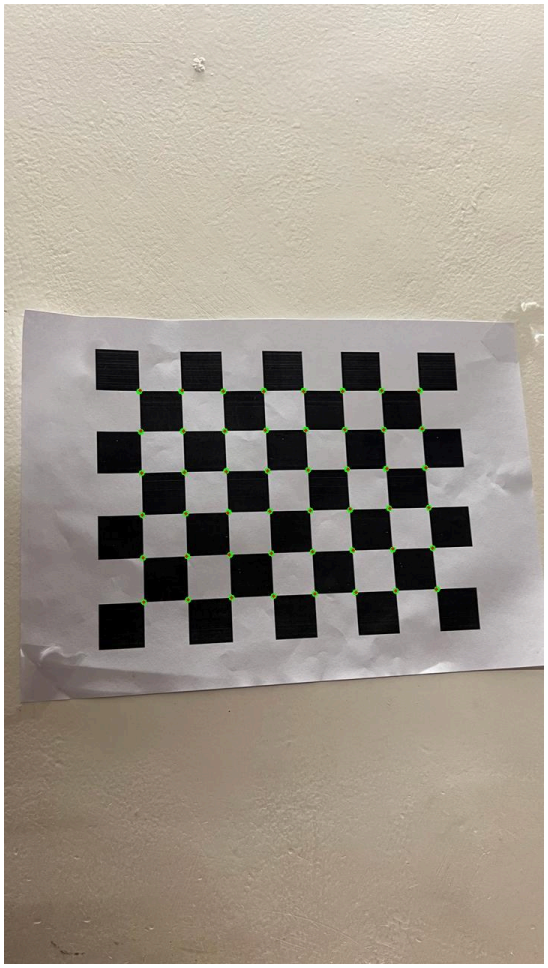


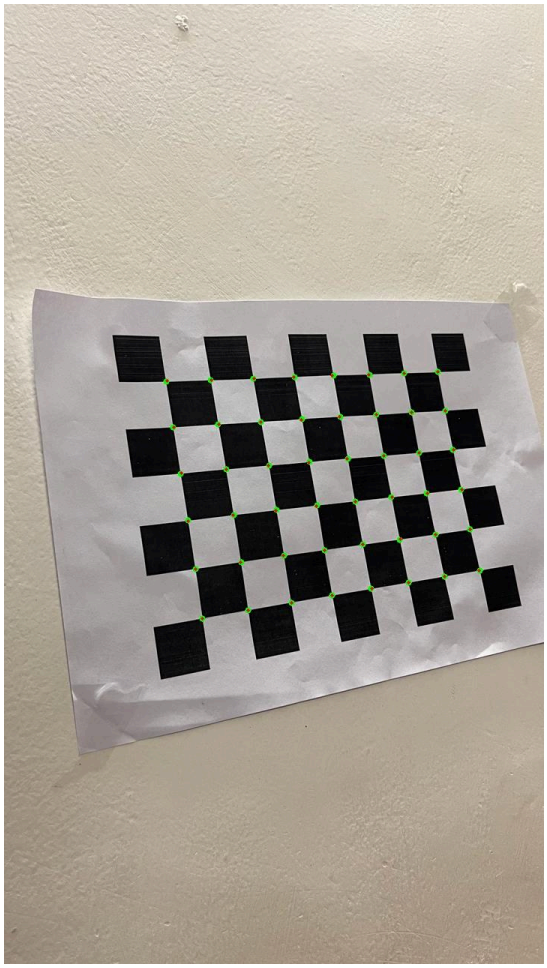
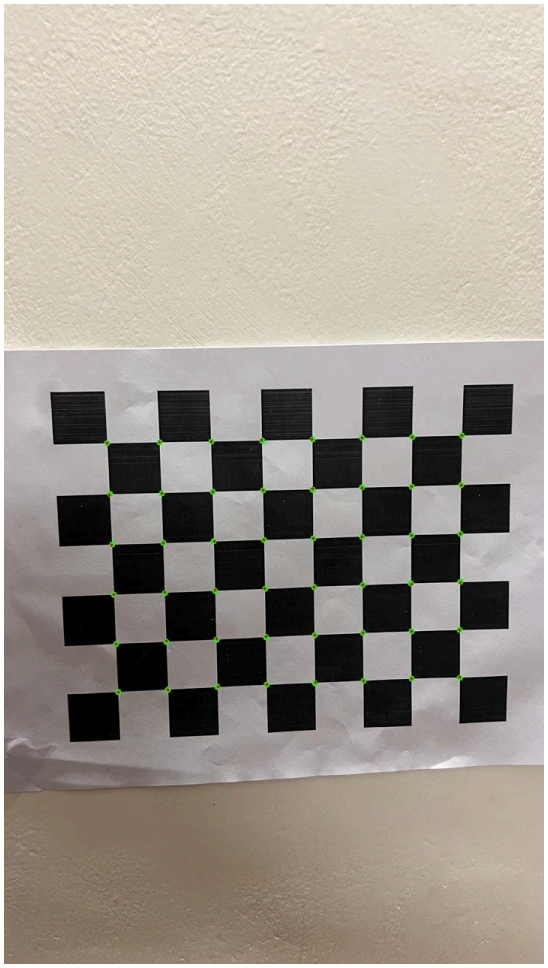


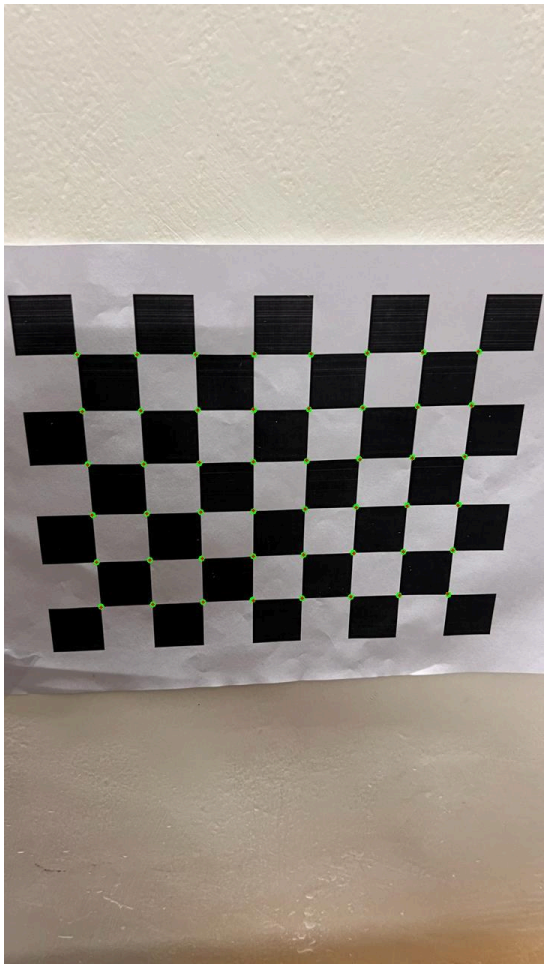
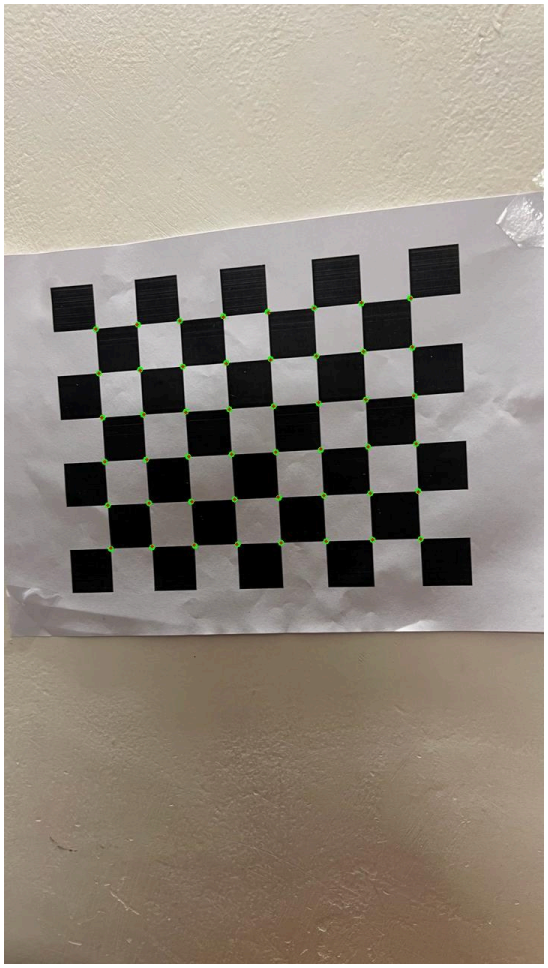


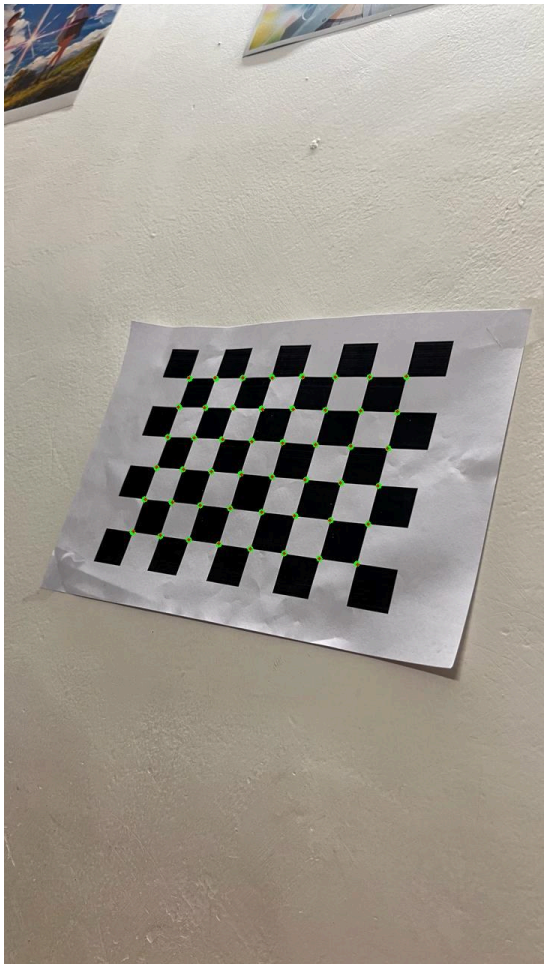






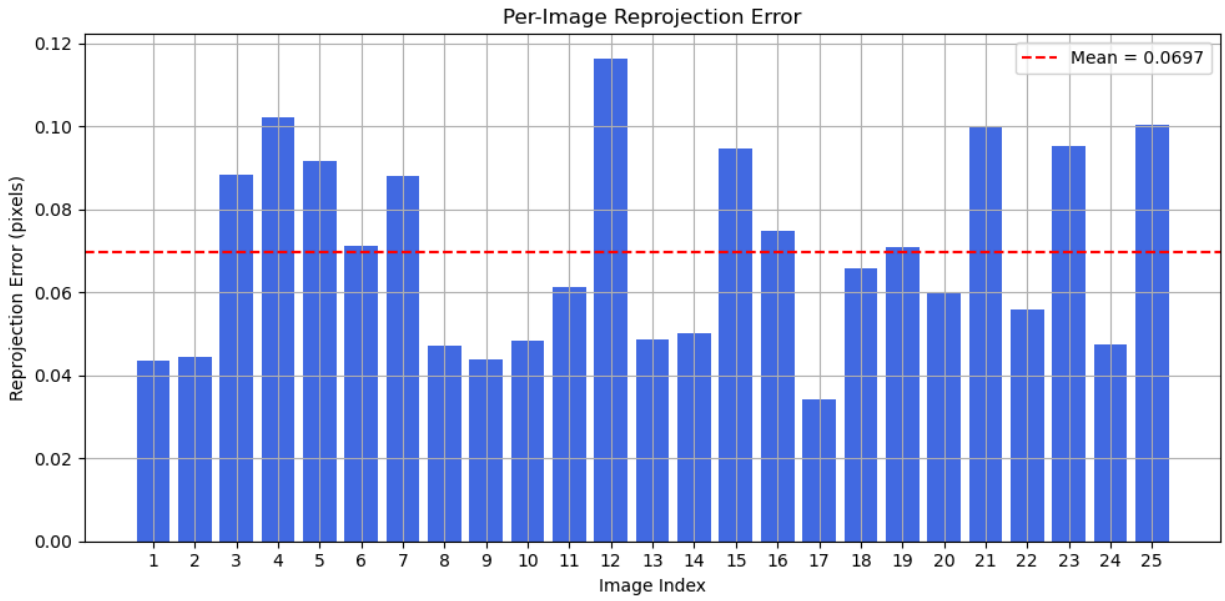





Results

- **Mean Reprojection Error**: 0.0697 pixels
- **Standard Deviation**: 0.0232 pixels



5. The **reprojection error** quantifies how accurately the estimated camera parameters (intrinsic, extrinsic, and distortion coefficients) model the actual projection of 3D points to the 2D image plane.

Computation:

For each image:

1. We **project the known 3D object points** (i.e., the 3D checkerboard corners) into the image using the camera matrix, distortion coefficients, and the pose (rotation and translation vectors).

2. We compare these **projected 2D points** to the **actual detected corner locations** in the image.

3. The reprojection error is the **average Euclidean distance** between the detected and projected points:
 Reprojection Error=1N∑i=1N // pi−p^i // 2\text{Reprojection Error} = \frac{1}{N} \sum_{i=1}^{N} \| p_i - \hat{p}_i \|_2Reprojection Error=N1i=1∑N // pi−p^i // 2
 where pip_ipi is the detected corner, and p^i\hat{p}_ip^i is the reprojected point.

6. Image Normal Vector

1 [-0.0762, -0.1487, 0.9859]

2 [-0.0229, 0.1238, 0.9920]

3 [-0.3983, -0.1872, 0.8980]

4	[-0.3464, -0.5281, 0.7753]
5	[0.3582, 0.3986, 0.8443]
6	[0.0452, -0.3252, 0.9446]
7	[-0.1278, -0.5073, 0.8523]
8	[0.0720, 0.2172, 0.9735]
9	[-0.0954, 0.1972, 0.9757]
10	[-0.1461, 0.0739, 0.9865]
11	[-0.1077, -0.0864, 0.9904]
12	[-0.3853, -0.3092, 0.8695]
13	[-0.1355, -0.1060, 0.9851]
14	[-0.0976, -0.2745, 0.9566]
15	[-0.3603, 0.0181, 0.9326]
16	[-0.1646, -0.3628, 0.9172]
17	[0.0357, -0.1105, 0.9932]
18	[-0.2233, 0.0683, 0.9724]
19	[-0.2714, -0.0962, 0.9577]
20	[-0.1658, 0.1601, 0.9731]
21	[-0.3974, 0.1908, 0.8976]
22	[-0.0833, -0.1275, 0.9883]
23	[-0.1031, -0.4017, 0.9100]
24	[0.1888, -0.0664, 0.9798]
25	[0.4546, 0.4001, 0.7958]

Custom Images:

1. **Estimated Intrinsic Camera Parameters:**
   Focal Lengths:

- fx = 1306.50 pixels
- fy = 1319.42 pixels

Principal Point:

- cx = 631.95 pixels
- cy = 531.41 pixels

Skew Parameter:

- 0.00 (indicating no skew between image axes)

Intrinsic Matrix (K):

[[1.30650443e+03 0.00000000e+00 6.31947651e+02]

[0.00000000e+00 1.31941647e+03 5.31411939e+02]

[0.00000000e+00 0.00000000e+00 1.00000000e+00]]

Reprojection Error:

- 1.8691 pixels (indicating the average discrepancy between observed and projected points)

2. **Estimated Extrinsic Camera Parameters:**

Image 1:
Rotation Matrix (R):

[[0.07173739 0.99658906 0.04079217]

[-0.84302223 0.038724   0.53648296]

[0.53307341 -0.0728746 0.84292469]]

Translation Vector (t):

[[-48.40408507]

[120.3335496]

[455.45193068]]

Image 2:
Rotation Matrix (R):

[[0.74750942 0.66145545 -0.06087978]

[-0.50668867 0.6270634 0.59165706]

[0.42953027 -0.41142214 0.80388779]]

Translation Vector (t):

[[-137.16624448]

[158.24317515]

[608.50635196]]

3. **Estimated Radial Distortion Coefficients:**

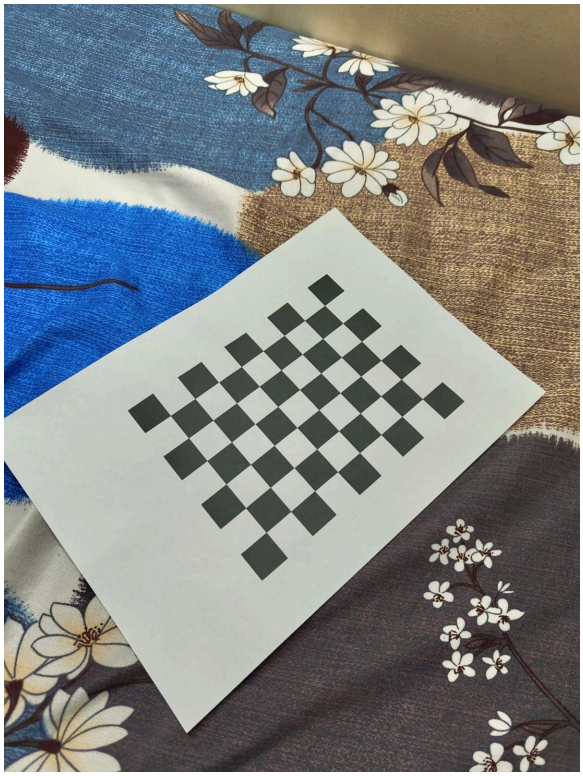
- Distortion Coefficients:
  [[-0.23592874 0.67182843 -0.02704358 0.00346771 -0.70792782]]

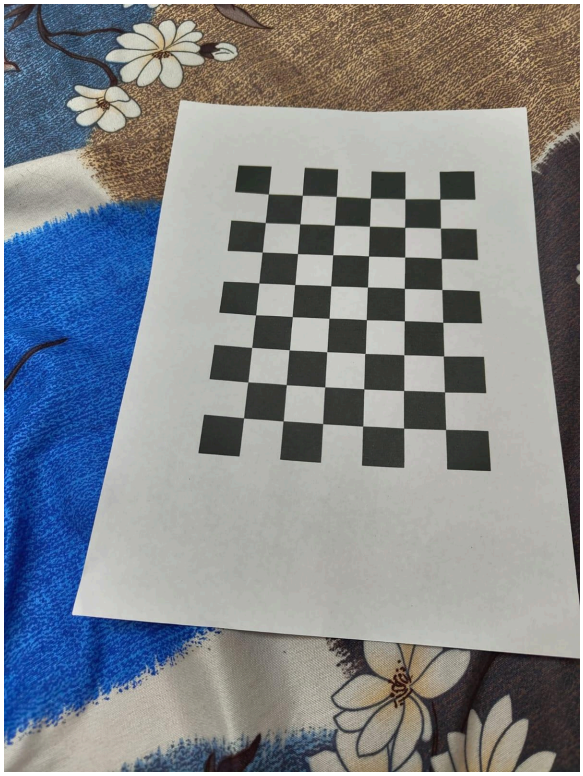
  where:

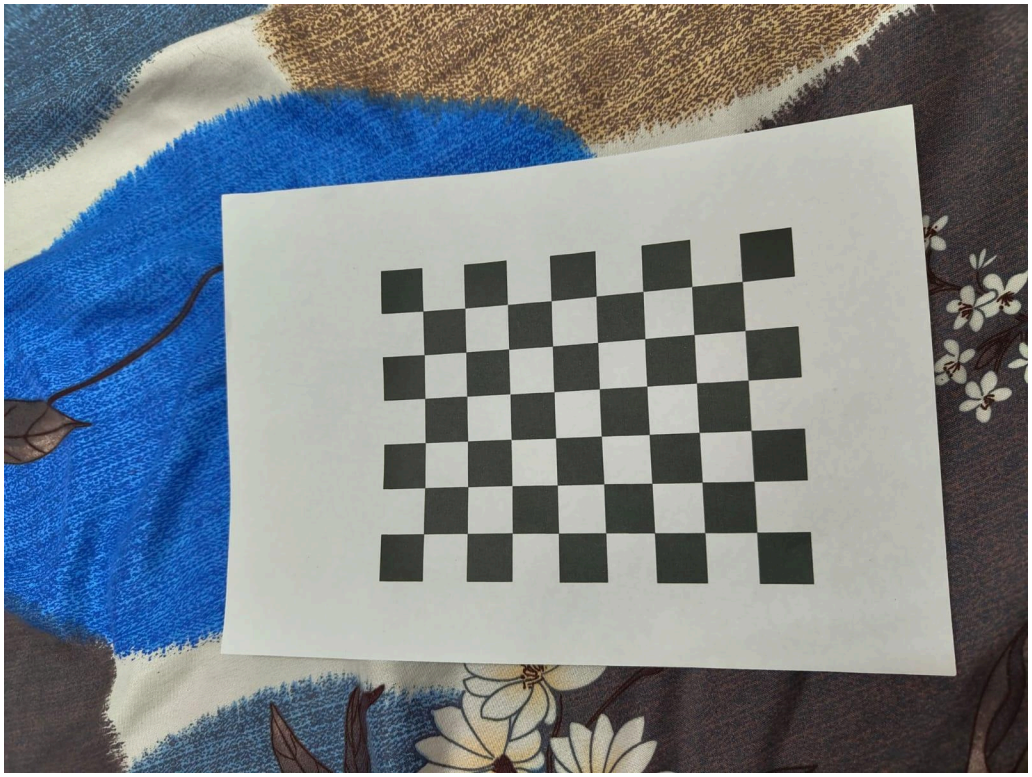
  - k1=−0.2359k_1 = -0.2359k1=−0.2359
  - k2=0.6718k_2 = 0.6718k2=0.6718
  - p1=−0.0270p_1 = -0.0270p1=−0.0270

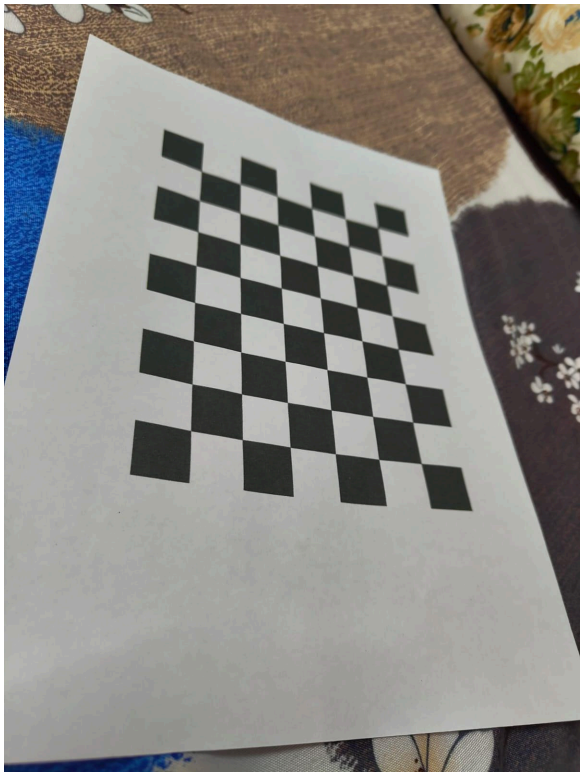
- p2=0.0035p_2 = 0.0035p2=0.0035
- k3=−0.7079k_3 = -0.7079k3=−0.7079

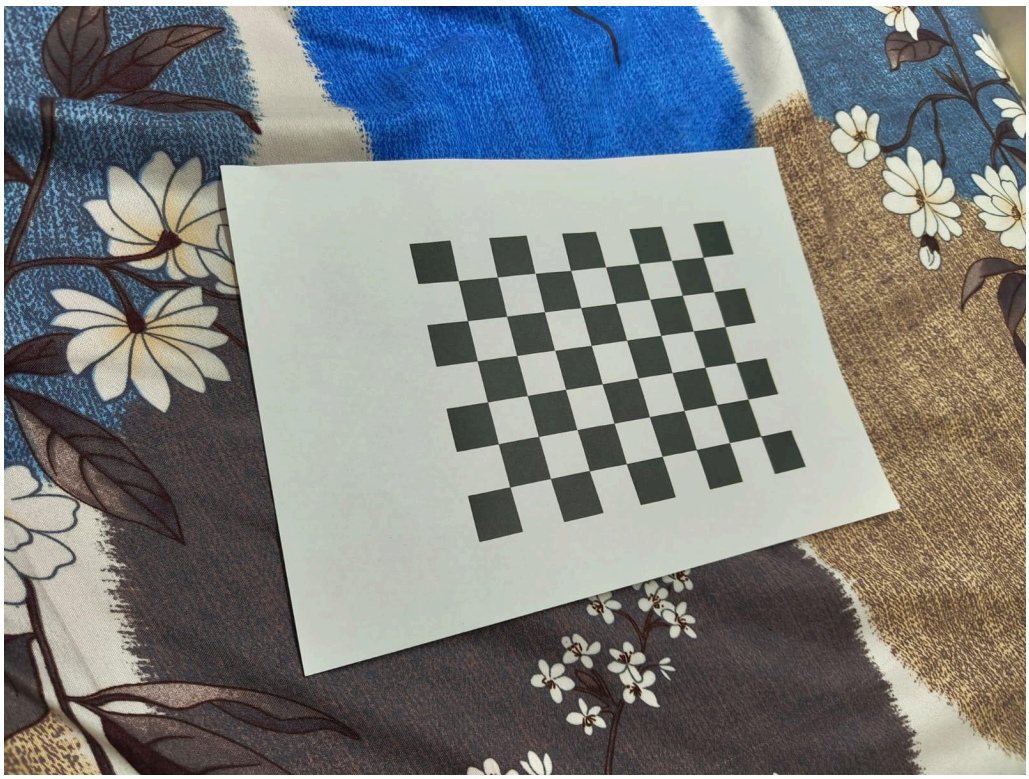
Raw Images:



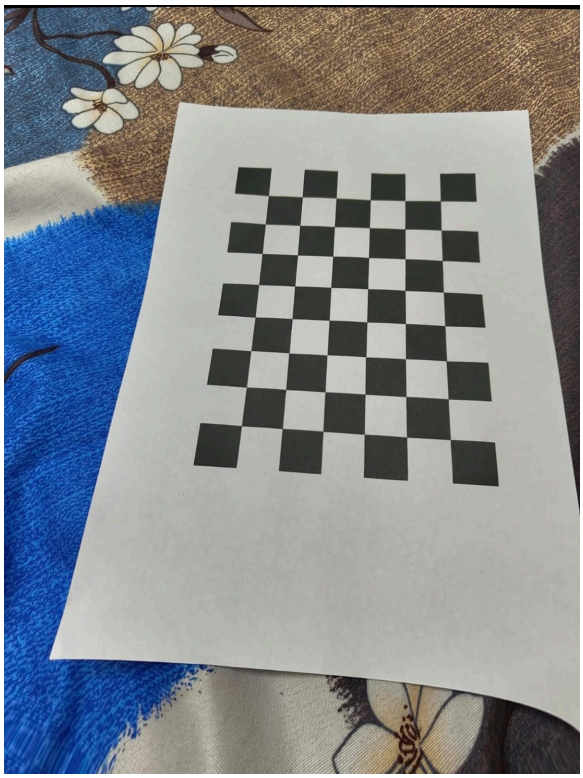


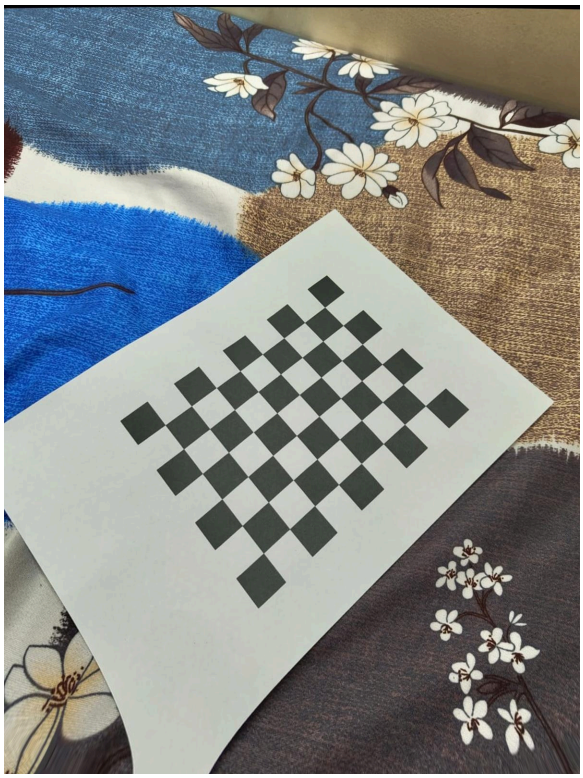


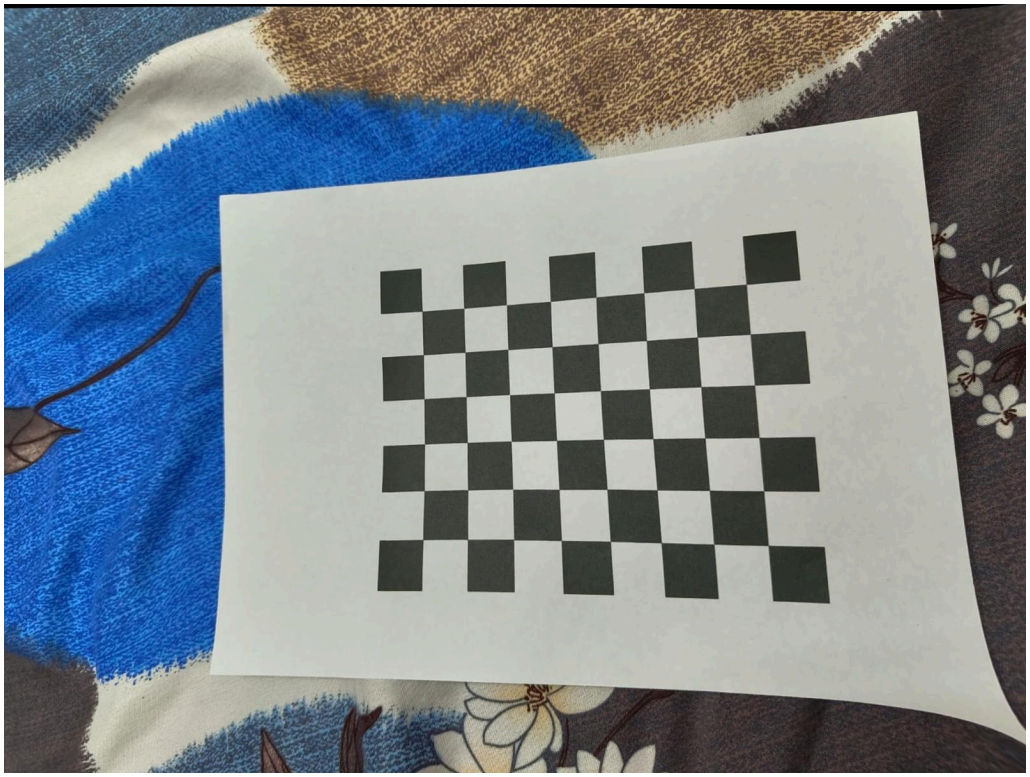


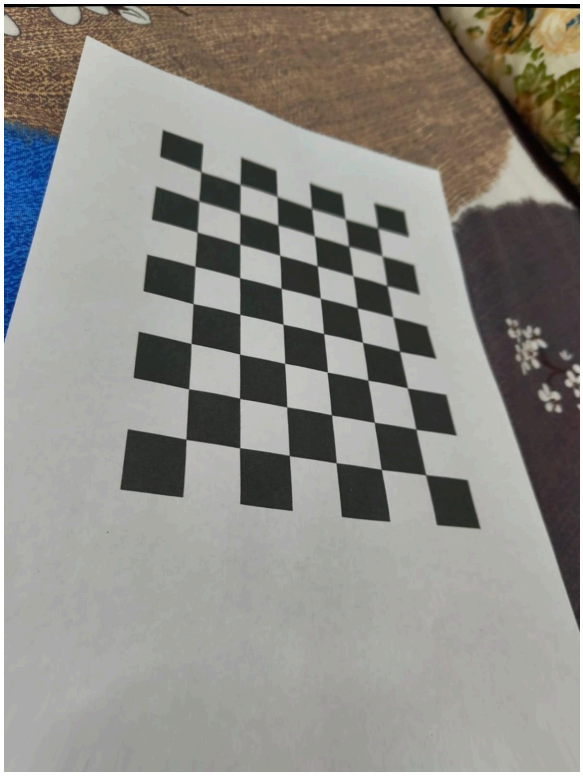


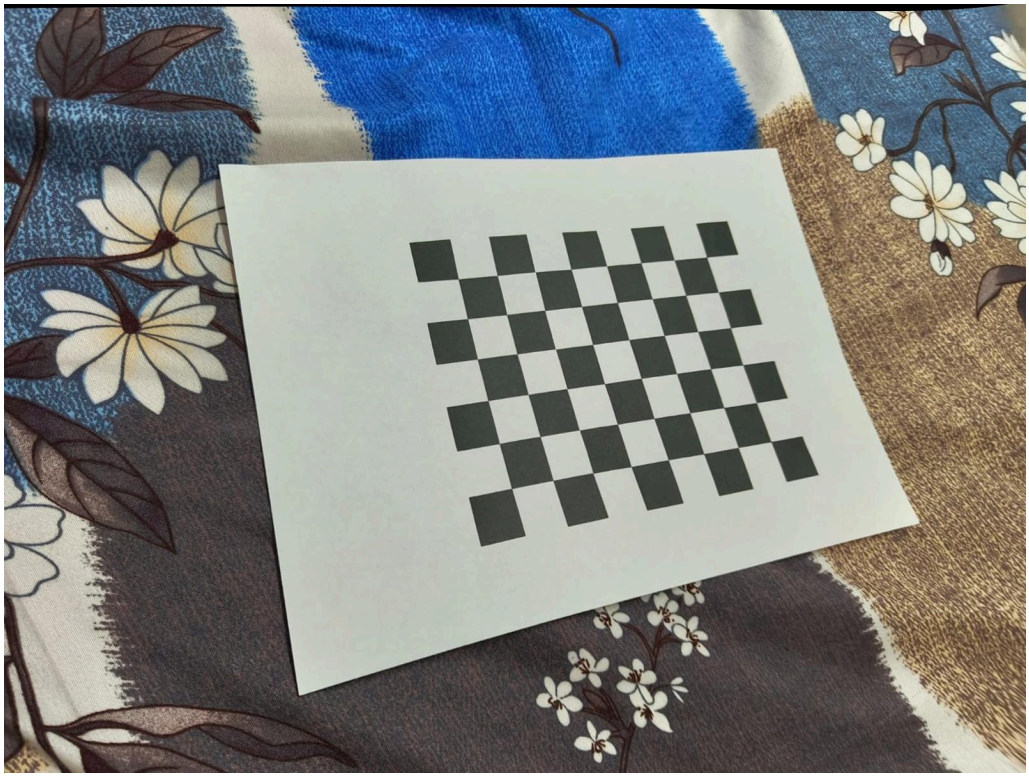
Undistorted Images:











Observation:

After applying the distortion correction:

4. The **checkerboard corners** near the image edges appear **straighter** and more **uniformly spaced**.

5. In the **raw images**, slight curvature or warping was visible along the edges — especially near the corners — due to radial distortion.

6. In the **undistorted images**, this effect is **corrected**, and the edges appear more linear, indicating that the **camera calibration and distortion correction were effective**.

4. Image 01 - Reprojection Error: 0.3172 pixels

Image 02 - Reprojection Error: 0.1248 pixels

Image 03 - Reprojection Error: 0.3472 pixels

Image 04 - Reprojection Error: 0.3203 pixels

Image 05 - Reprojection Error: 0.1732 pixels

Image 06 - Reprojection Error: 0.2241 pixels

Image 07 - Reprojection Error: 0.2066 pixels

Image 08 - Reprojection Error: 0.2184 pixels

Image 09 - Reprojection Error: 0.2650 pixels

Image 10 - Reprojection Error: 0.2567 pixels

Image 11 - Reprojection Error: 0.2168 pixels

Image 12 - Reprojection Error: 0.1763 pixels

Image 13 - Reprojection Error: 0.2654 pixels

Image 14 - Reprojection Error: 0.2183 pixels

Image 15 - Reprojection Error: 0.1676 pixels

Image 16 - Reprojection Error: 0.2241 pixels

Image 17 - Reprojection Error: 0.2406 pixels

Image 18 - Reprojection Error: 0.1930 pixels

Image 19 - Reprojection Error: 0.6814 pixels

Image 20 - Reprojection Error: 0.1401 pixels

Image 21 - Reprojection Error: 0.1874 pixels

Image 22 - Reprojection Error: 0.2546 pixels

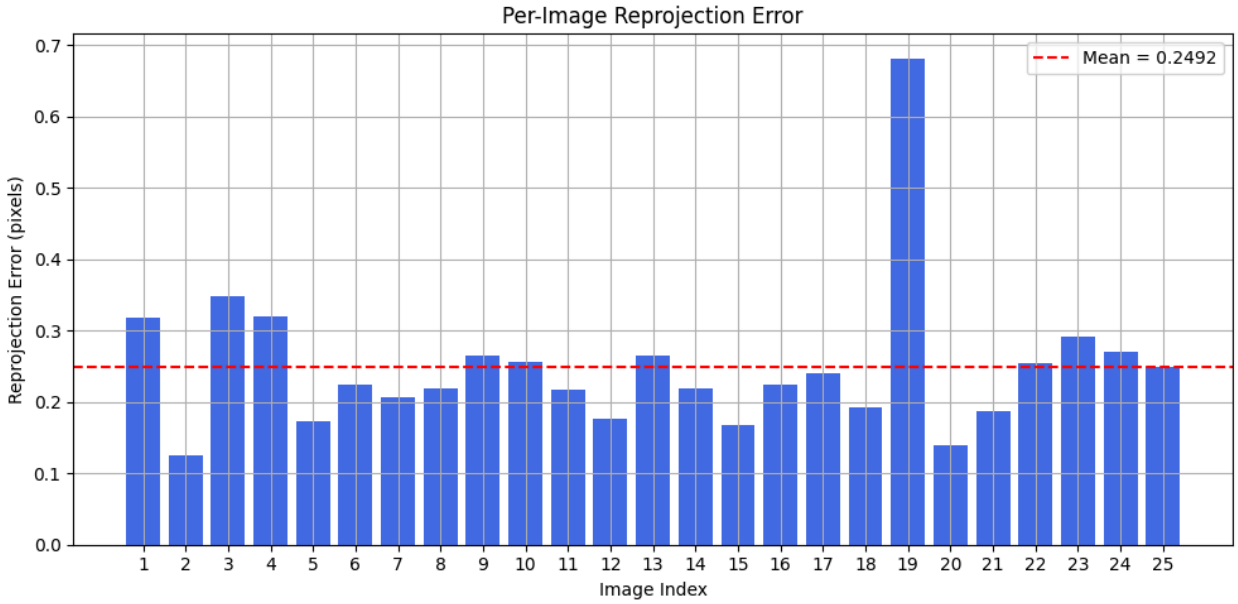
Image 23 - Reprojection Error: 0.2913 pixels

Image 24 - Reprojection Error: 0.2705 pixels

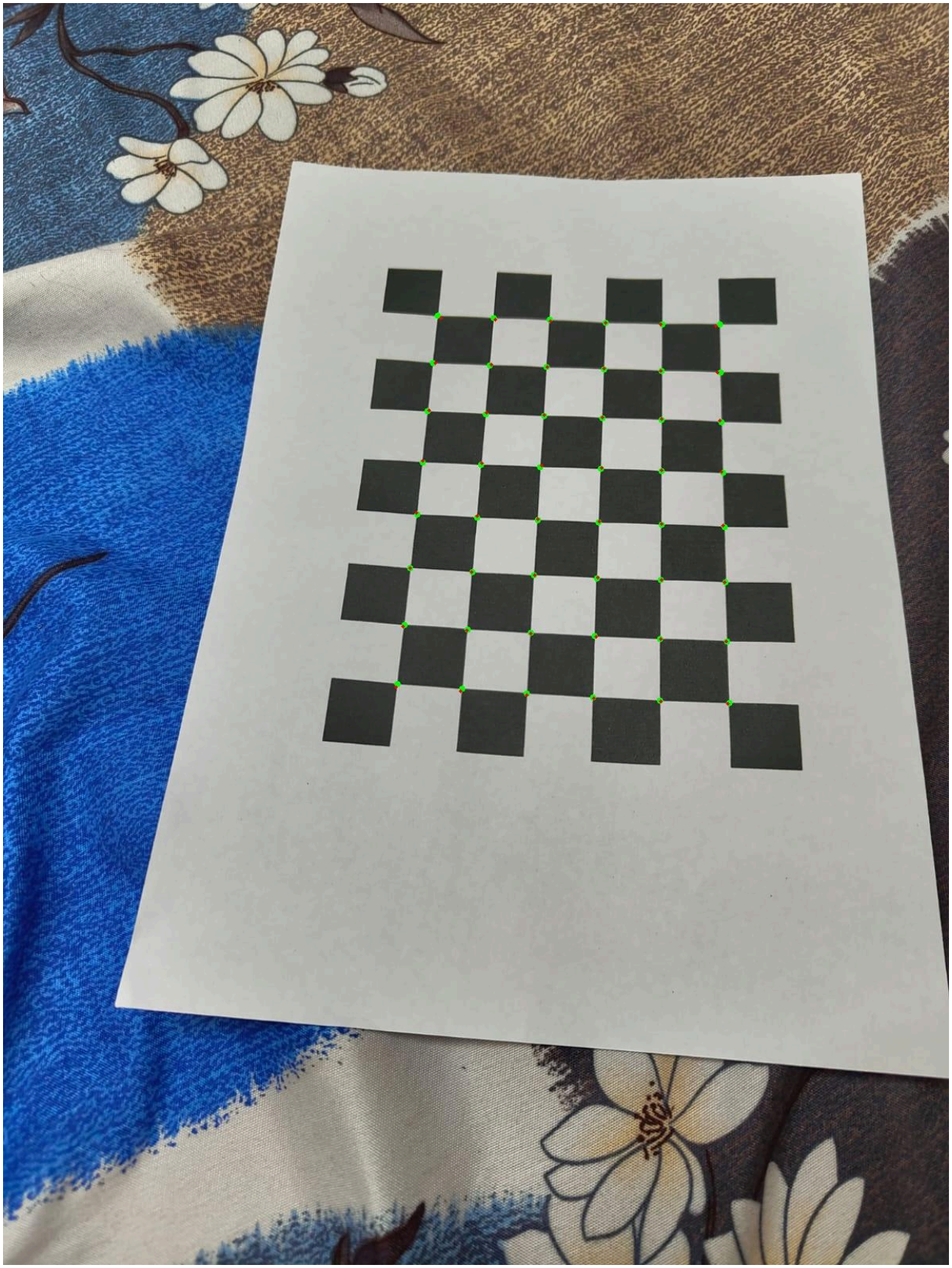
Image 25 - Reprojection Error: 0.2485 pixels

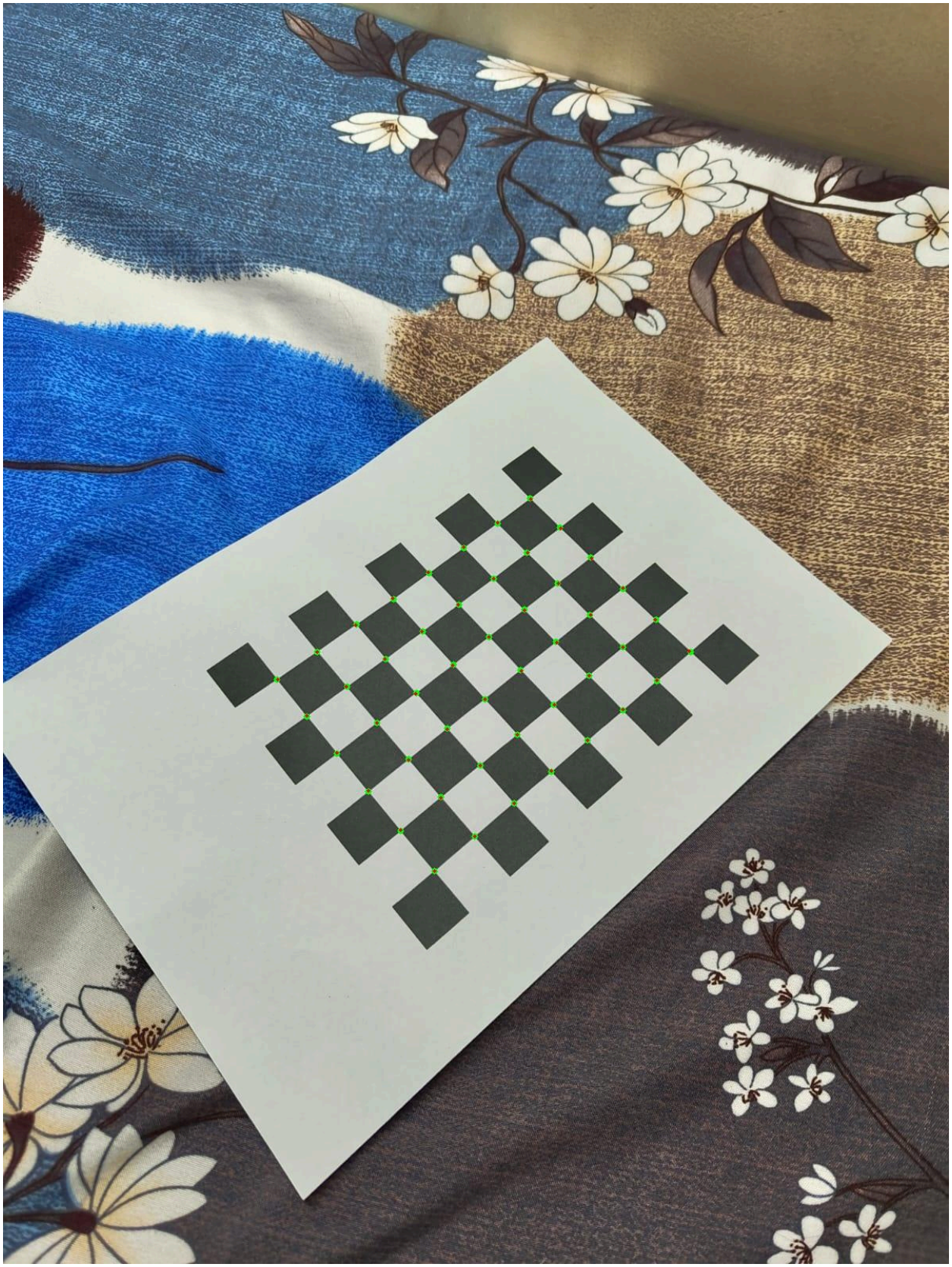
Mean Reprojection Error: 0.2492 pixels

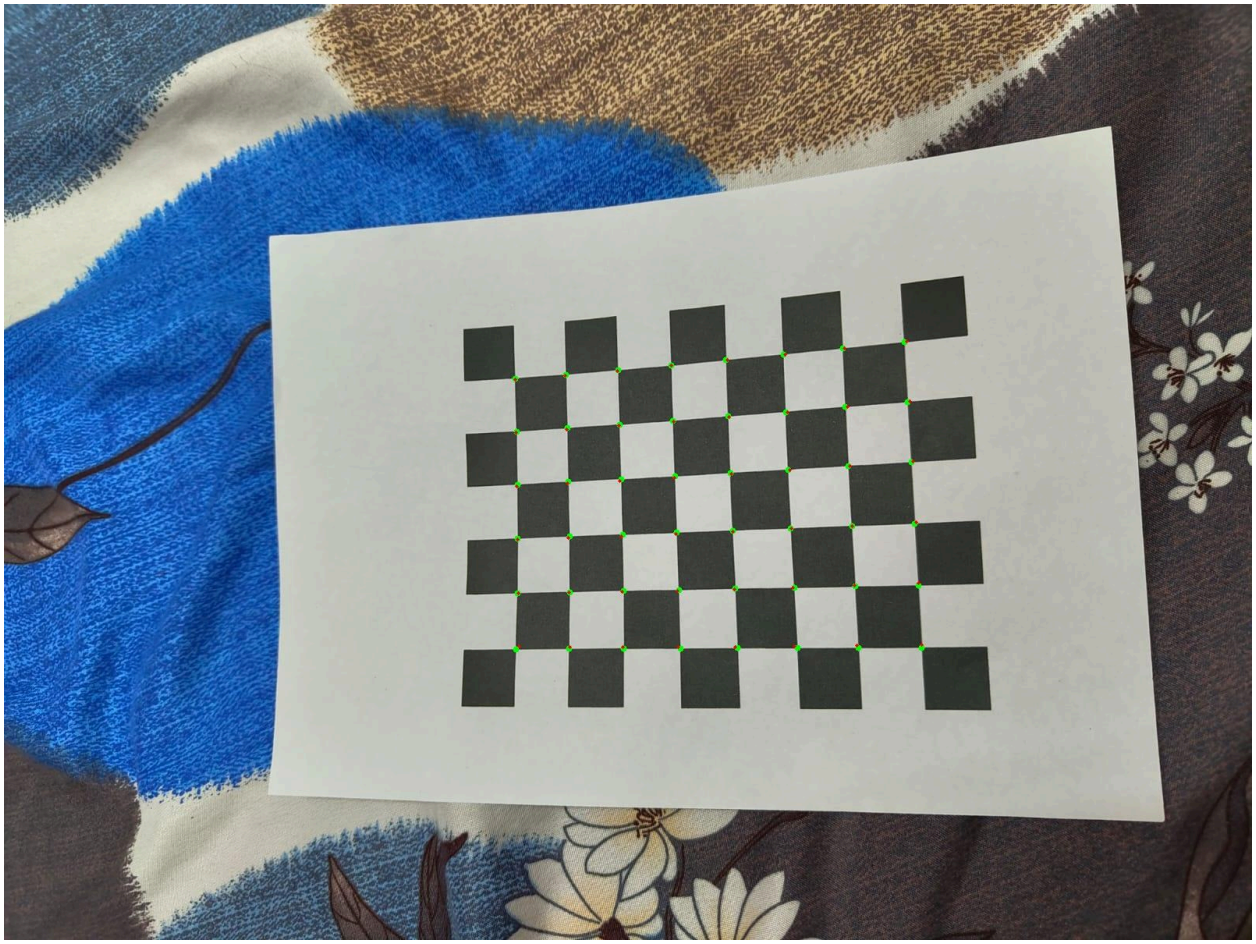
Standard Deviation: 0.1034 pixels

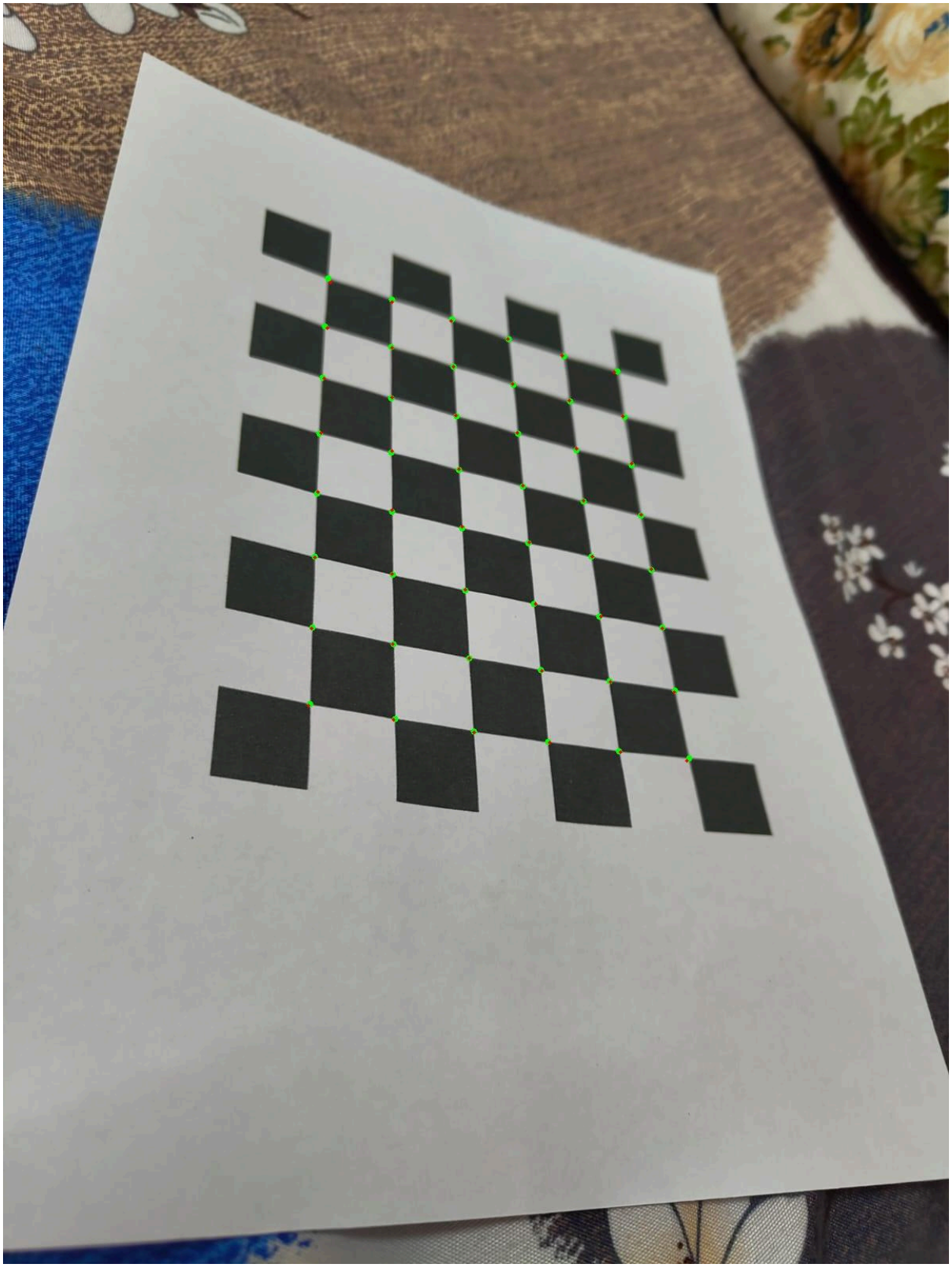


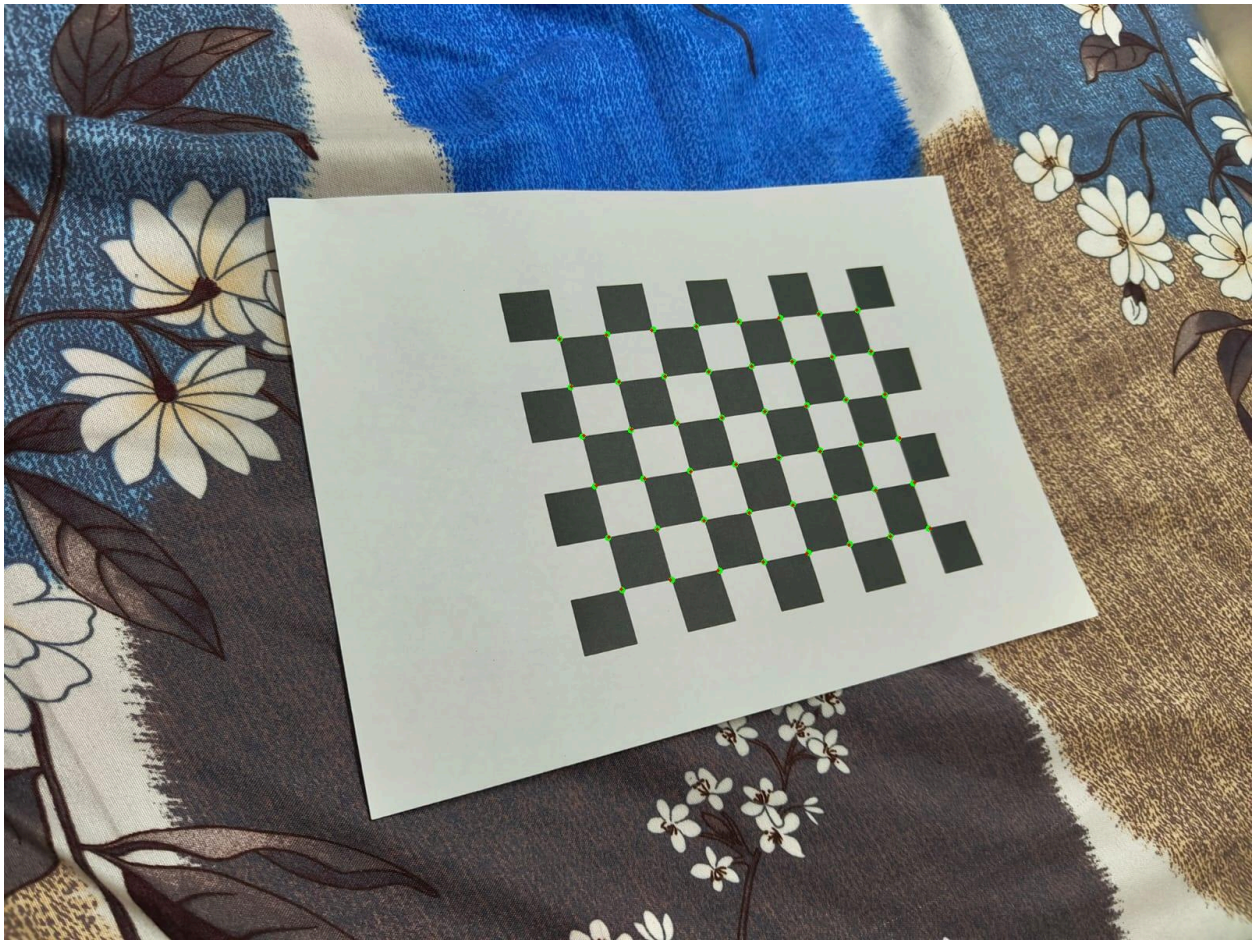
5.

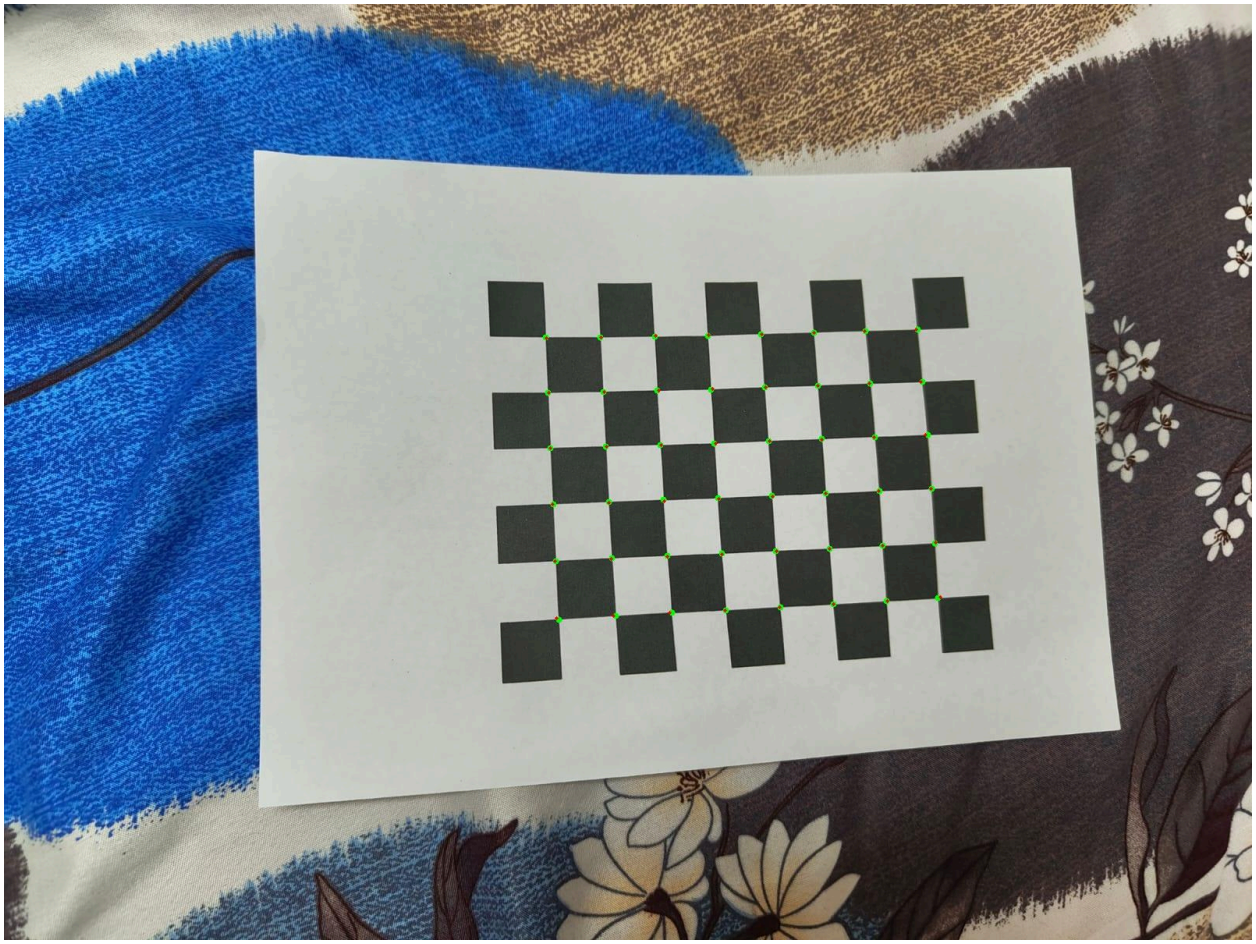


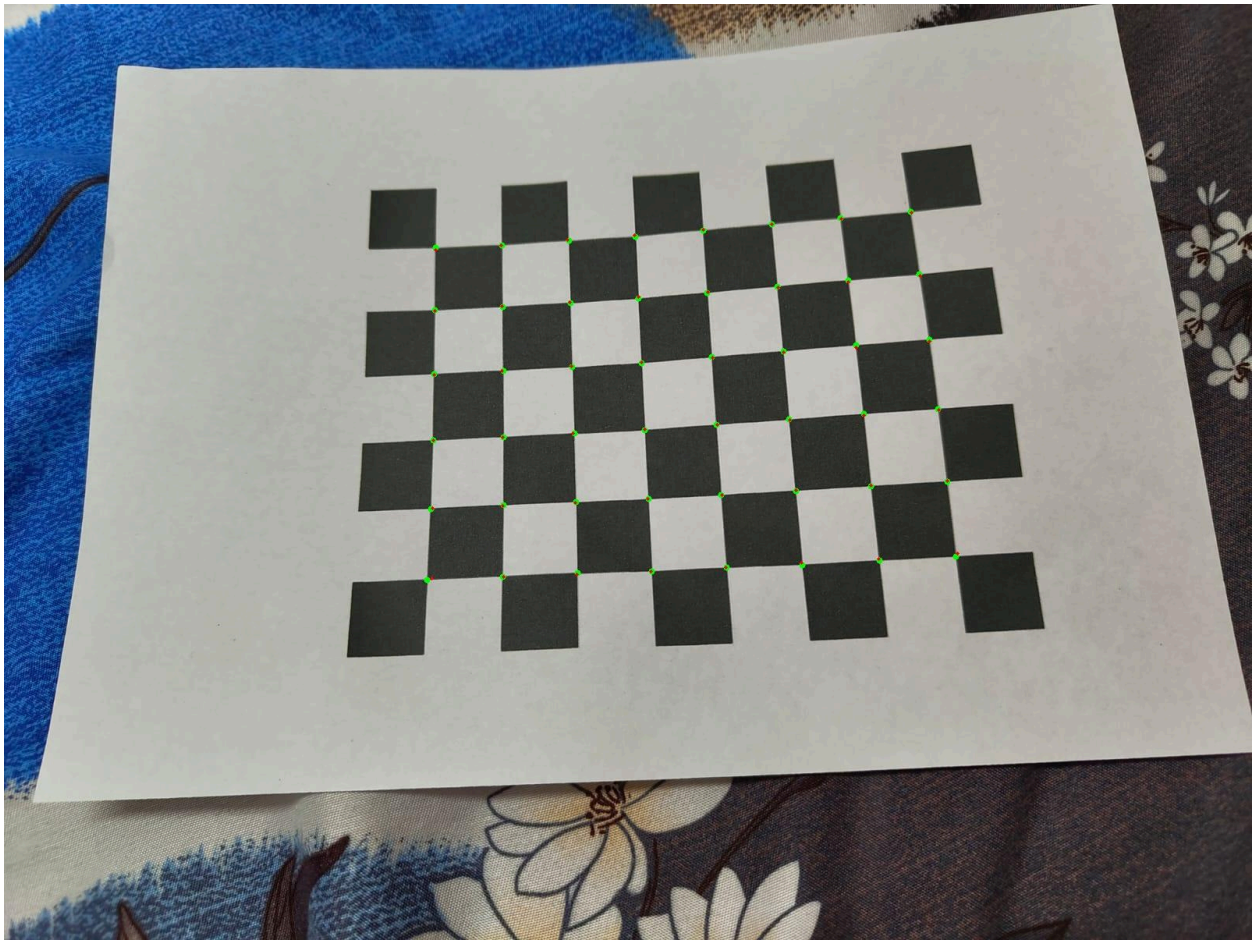


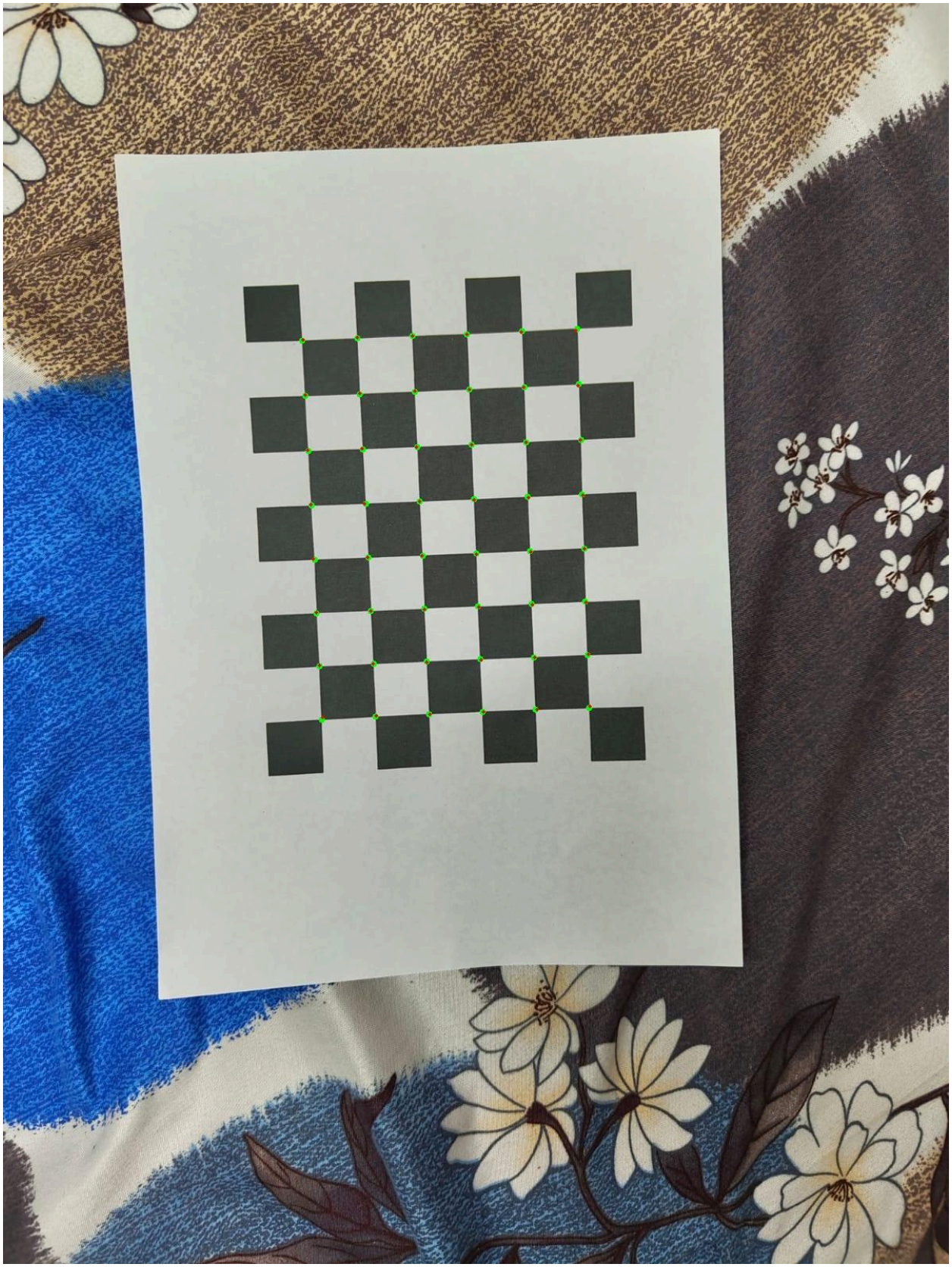


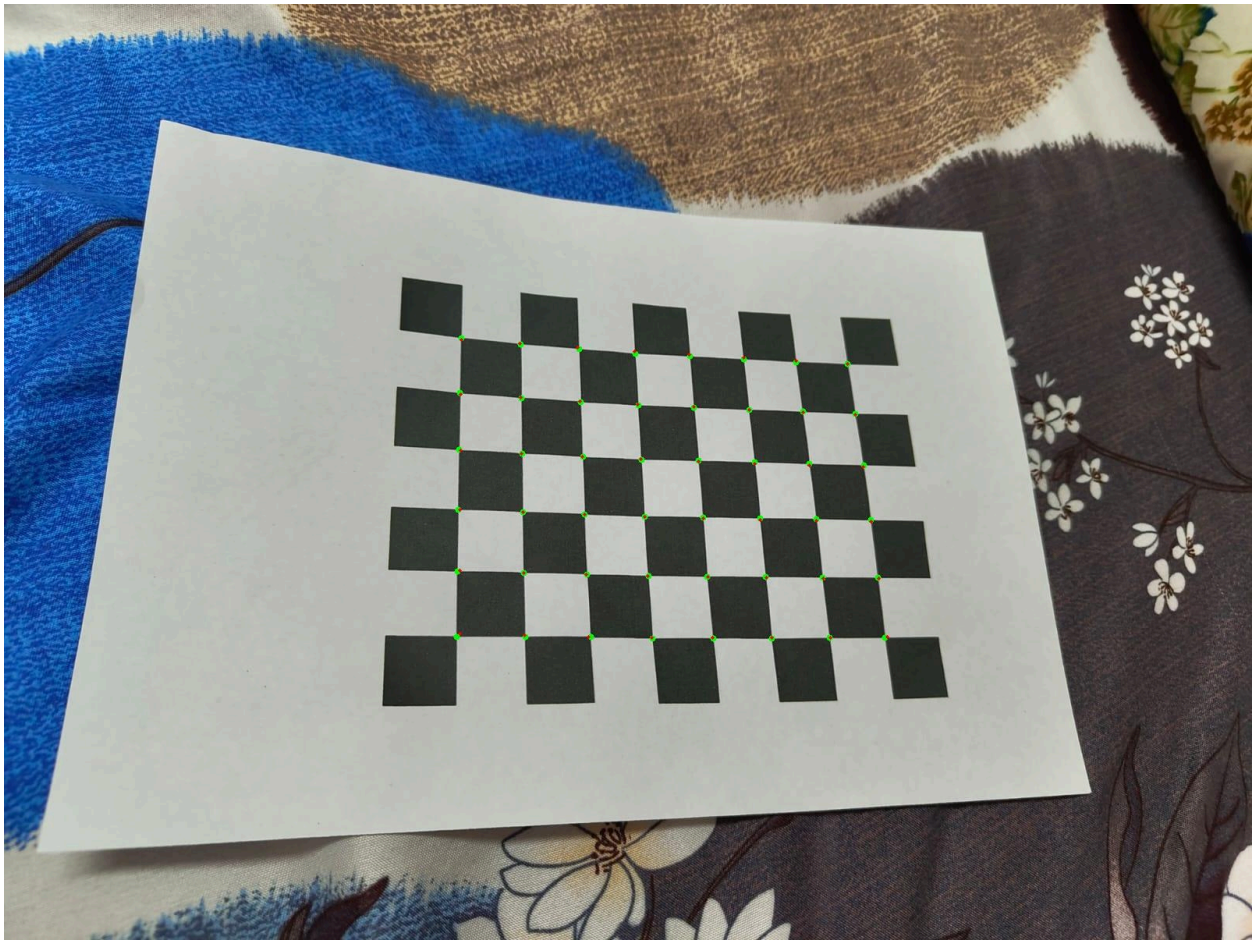


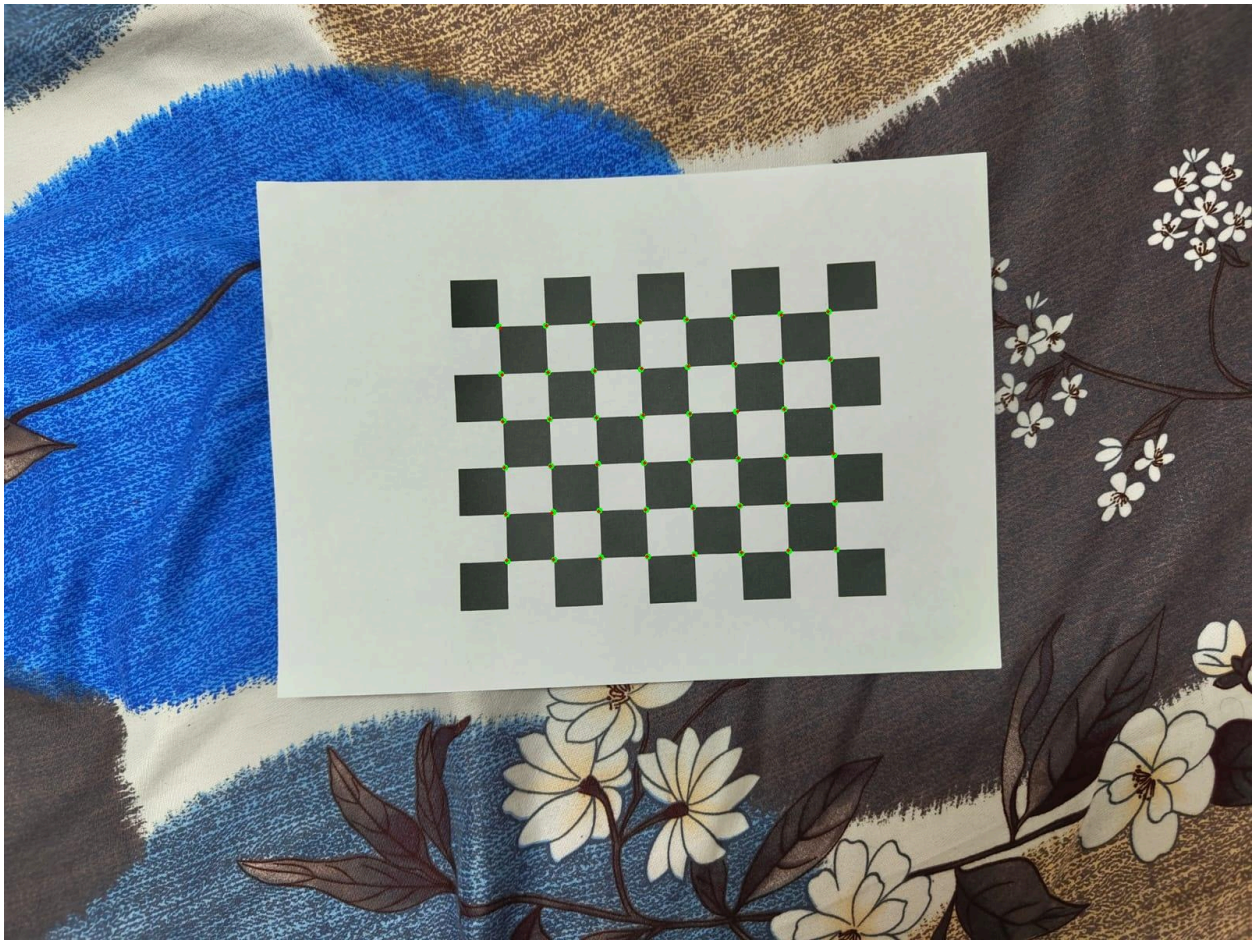


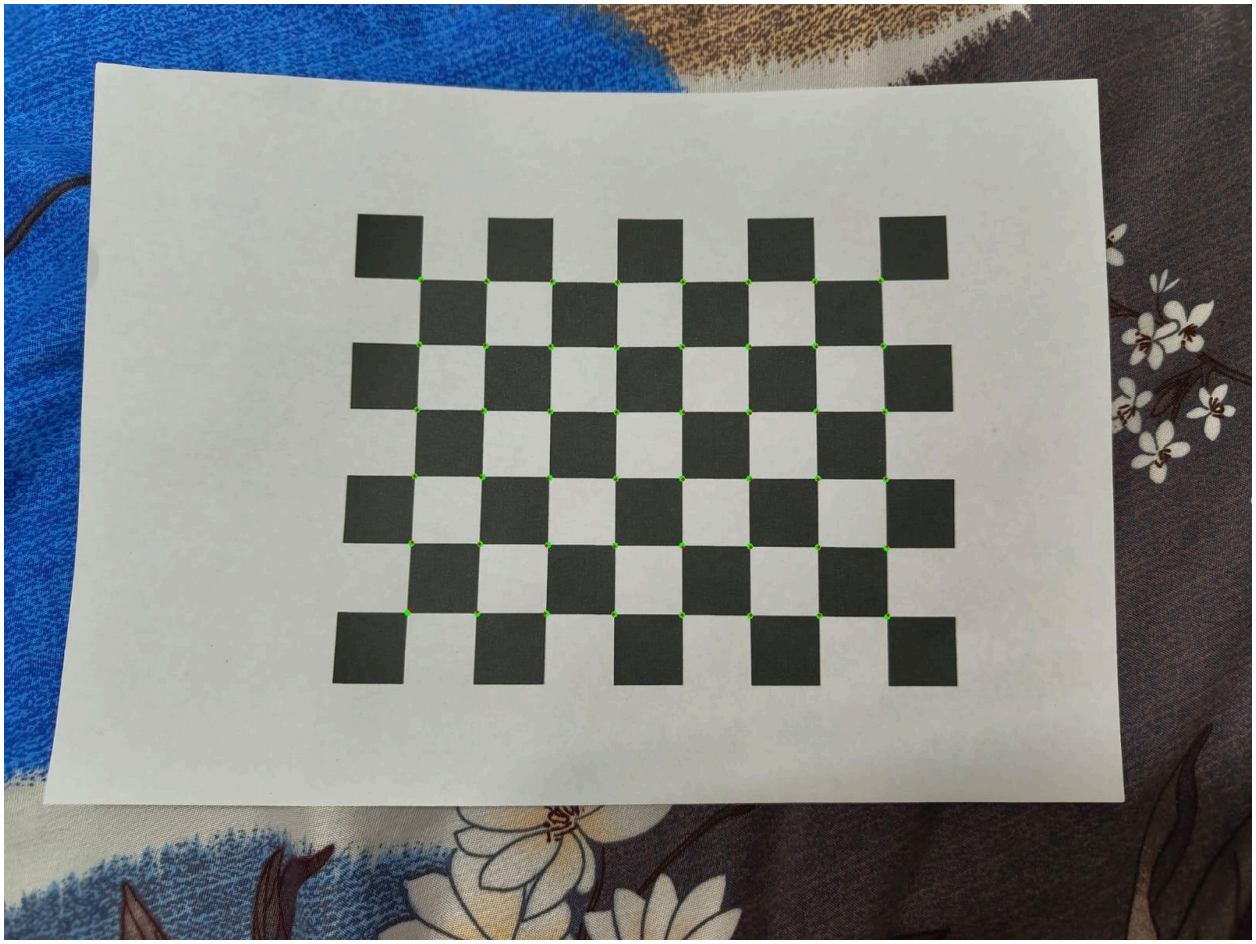


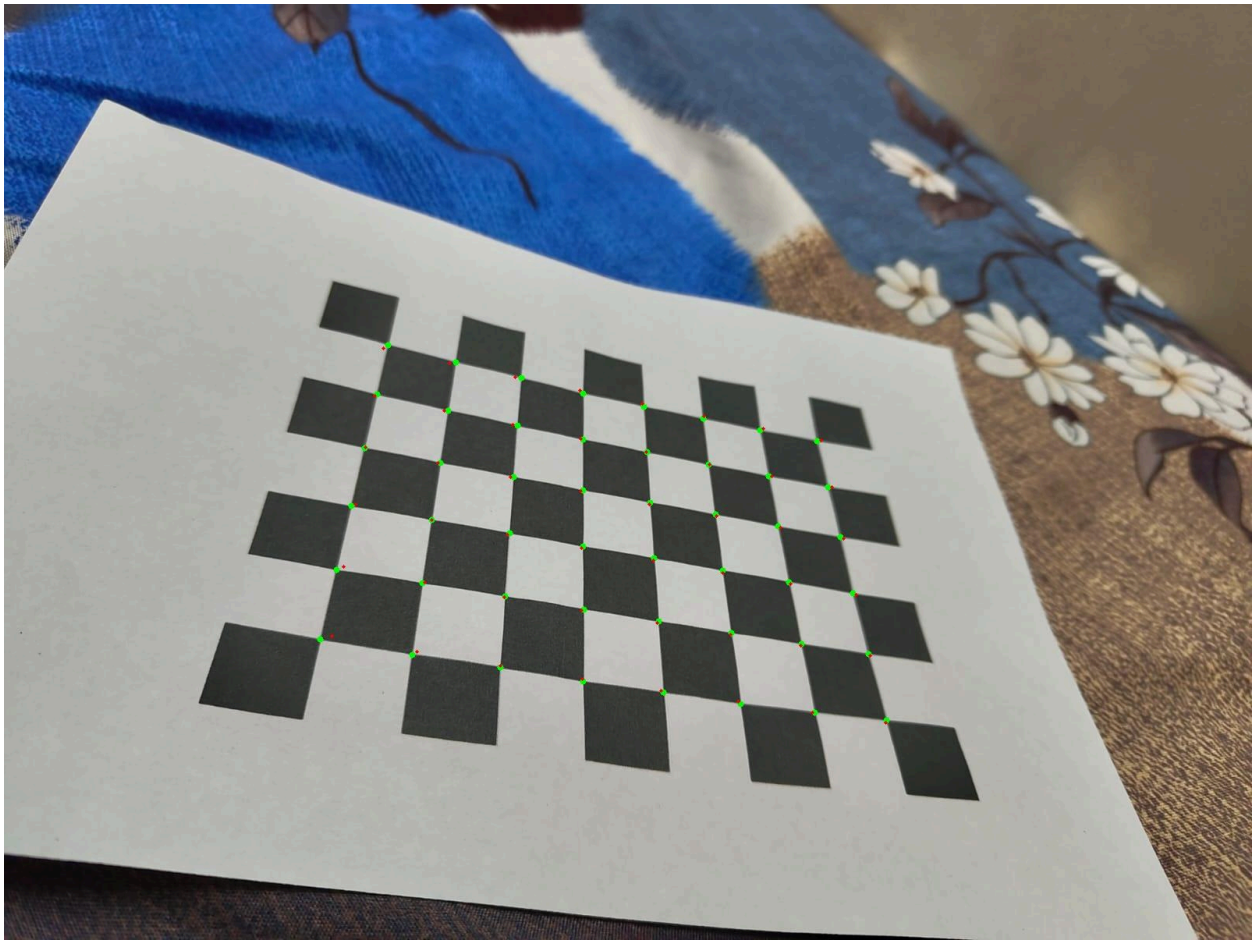


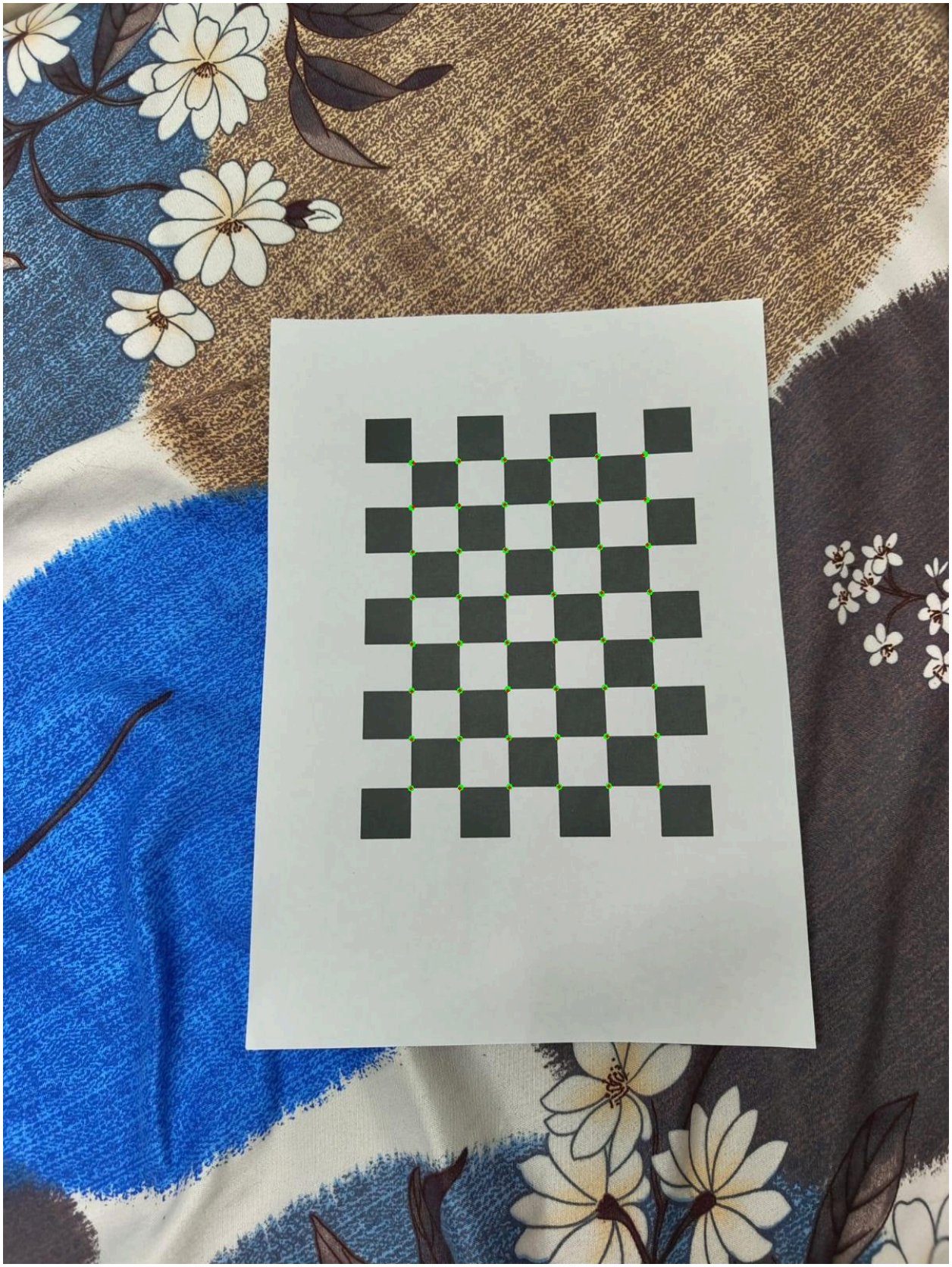


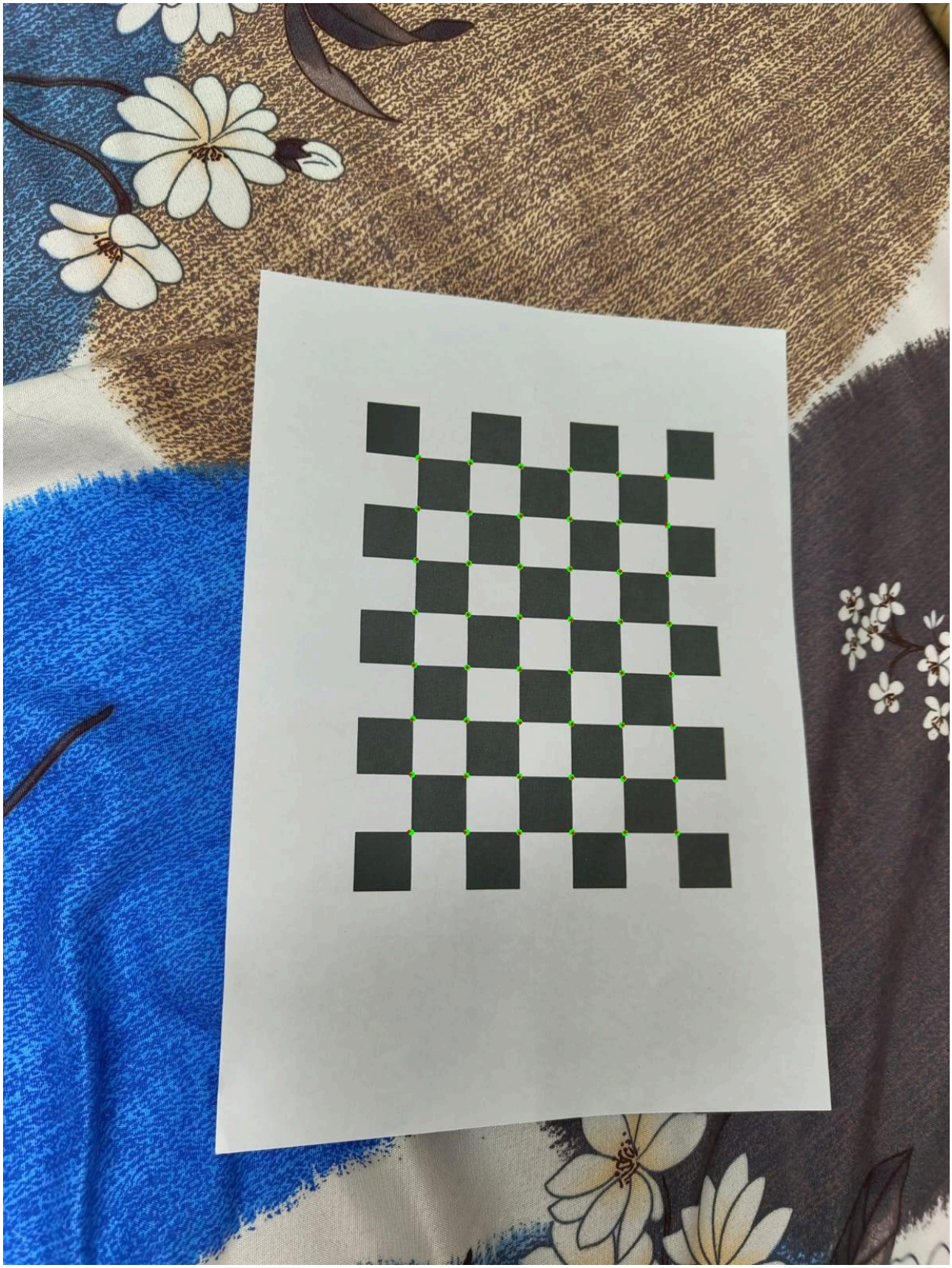












For each calibration image, the **3D world coordinates** of the checkerboard corners are known (in real-world scale).

These 3D points are **projected back onto the 2D image plane** using the estimated **intrinsic** (K) and **extrinsic** (R, t) parameters.

The **Euclidean distance** (L2 norm) is calculated between the projected 2D points and the **actual detected corner points** in the image.

The average of these distances across all corners in an image gives the **per-image reprojection error**.

This value quantifies the calibration accuracy — the lower the error, the better the model fits the observed data.

6. **Checkerboard Plane Normals in Camera Frame**

- The plane normal nCi∈R3\mathbf{n}_{C_i} \in \mathbb{R}^3nCi∈R3 for each checkerboard image was computed using the third column of the **rotation matrix** RiR_iRi obtained during camera calibration.
- This vector represents the **z-axis of the checkerboard plane** in the **camera's coordinate frame** and indicates its **orientation** relative to the camera.

Image 01 - Plane Normal (Camera Frame): [0.04079217 0.53648296 0.84292469]

Image 02 - Plane Normal (Camera Frame): [-0.06087978 0.59165706 0.80388779]

Image 03 - Plane Normal (Camera Frame): [0.37785269 0.13268537 0.91630887]

Image 04 - Plane Normal (Camera Frame): [-0.43568572 0.62526014 0.64747797]

Image 05 - Plane Normal (Camera Frame): [-0.24641055 0.37073319 0.89545449]

Image 06 - Plane Normal (Camera Frame): [0.38033675 0.64320056 0.66455774]

Image 07 - Plane Normal (Camera Frame): [-0.05397362 0.20087913 0.97812802]

Image 08 - Plane Normal (Camera Frame): [0.27481746 0.71103838 0.64722468]

Image 09 - Plane Normal (Camera Frame): [-0.09064862 0.72586098 0.68184211]

Image 10 - Plane Normal (Camera Frame): [0.14050772 0.32569077 0.9349776]

Image 11 - Plane Normal (Camera Frame): [0.11688001 0.56613103 0.81598696]

Image 12 - Plane Normal (Camera Frame): [-0.0470609 0.03257829 0.99836062]

Image 13 - Plane Normal (Camera Frame): [-0.14823305 0.41211664 0.89899212]

Image 14 - Plane Normal (Camera Frame): [0.36275149 0.56828491 0.73855508]

Image 15 - Plane Normal (Camera Frame): [0.10157754 0.07916255 0.99167298]

Image 16 - Plane Normal (Camera Frame): [0.23857529 0.59602273 0.76670642]

Image 17 - Plane Normal (Camera Frame): [0.06265055 0.18668666 0.98041981]

Image 18 - Plane Normal (Camera Frame): [-0.32193488 0.51615939 0.79368597]

Image 19 - Plane Normal (Camera Frame): [-0.23444451 0.68422356 0.69056057]

Image 20 - Plane Normal (Camera Frame): [0.07009761 0.34120311 0.93737226]

Image 21 - Plane Normal (Camera Frame): [-0.21349236 0.34599446 0.91362402]

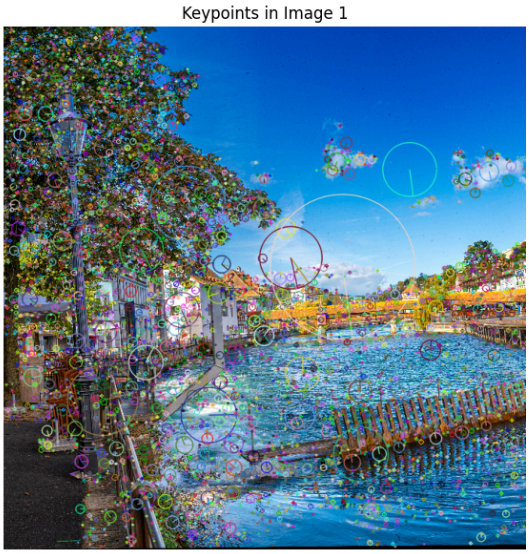
Image 22 - Plane Normal (Camera Frame): [0.16121956 0.05441965 0.98541705]

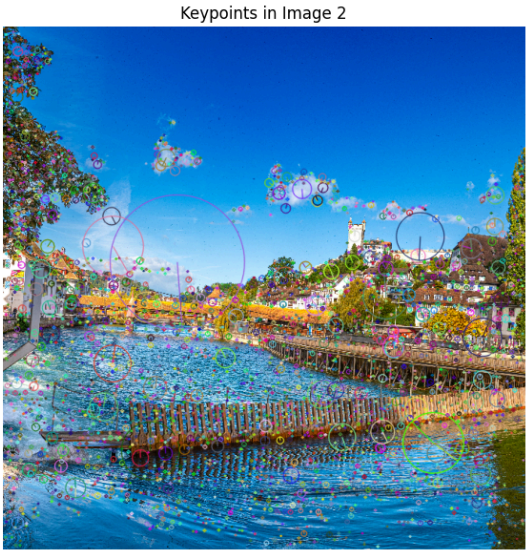
Image 23 - Plane Normal (Camera Frame): [0.15584567 0.8196933 0.55119418]

Image 24 - Plane Normal (Camera Frame): [-0.38253519 0.76517249 0.51785895]

Image 25 - Plane Normal (Camera Frame): [-0.21382321 0.87811181 0.42801785]

4. 1.





The keypoints are spread across **foreground and background**, including:

- Trees
- Water ripples
- Railings
- Buildings
- Sky texture (clouds)

The keypoints are displayed with **varying circle sizes** (scale)

Their **directionality** is captured (shown as orientation vectors)

This confirms that SIFT has correctly encoded **scale- and rotation-invariant** descriptors

2.

After extracting SIFT keypoints and descriptors from the first two images, we matched these features using two different approaches:

1. BruteForce Matcher

- Compares **each descriptor** in Image 1 with **all descriptors** in Image 2
- Retains matches based on **Lowe's Ratio Test** (0.75 threshold)

2. FLANN-Based Matcher

- Uses **approximate nearest neighbor search** for faster matching
- A hierarchical KD-Tree structure was used with:
    - `trees = 5`
    - `checks = 50`
- Lowe's Ratio Test was applied with a stricter threshold of **0.7**

Left: BruteForce Matcher — **667 matches**

Right: FLANN-Based Matcher — **608 matches**

Both methods result in **dense and geometrically consistent matches** across the overlapping region.

Matched keypoints align well along edges, water ripples, and buildings — confirming **correctness and reliability** of detected features.

FLANN is slightly faster and still produces **high-quality matches**, making it ideal for large-scale datasets or multi-image stitching.

3. To geometrically align the two images (`image1` and `image2`), we compute the **homography matrix** — a 3×3 transformation matrix that maps points from one image plane to another.

Methodology:

- Used **BruteForce-matched keypoints** (filtered with Lowe's ratio test) for establishing correspondences.
- **RANSAC (Random Sample Consensus)** was applied via `cv2.findHomography()` to robustly estimate the transformation:
    - RANSAC mitigates the impact of outliers
    - A reprojection threshold of `5.0` pixels was used

Homography matrix saved to homography_matrix.csv

[[8.99023648e-01 -1.31282351e-01 -1.64442160e+02]

[1.43181983e-01 9.90713800e-01 -6.94375875e+01]

[-1.27006065e-07 3.60666010e-07 1.00000000e+00]]

4.

Before stitching the images together, I first warp them into a **common perspective space** using the estimated **homography matrix**. This transformation adjusts the viewpoint of the images, allowing us to align and overlap them correctly for panorama creation.

- We used SIFT features and **BruteForce matching with Lowe's ratio test** to identify good feature correspondences.
- Using these matches, a **homography matrix** was computed via **RANSAC**.
- Both images were then warped into a **shared coordinate frame**:
    - `image1` was warped using the homography matrix
    - `image2` was placed in the same canvas without transformation
- A translation matrix was added to avoid negative pixel coordinates in the combined space.

Left: Warped `Image 1` using homography
Right: Unwarped `Image 2`, but shifted to align with `Image 1`

The overlapping content (bridge, tree, and water) appears **aligned**, verifying the success of the homography.

5.

Homography Matrix from earlier steps was reused to align the images.

`image1` was warped using this transformation, and `image2` was placed directly on the shared canvas.

To ensure a **clean and polished output**, we removed empty (black) borders using **binary thresholding + contour detection** for cropping.

Left: Direct stitching without cropping. Notice the visible black borders due to warping into a larger canvas.

Right: Final panorama after cropping out unnecessary black regions. The final result looks visually cohesive and clean.

6.

Panorama - Cluster 0

Panorama - Cluster 1

Panorama - Cluster 2

5. 1. I implemented the **Iterative Closest Point (ICP)** algorithm using the `open3d` library to align two consecutive point clouds, `pointcloud_0000.pcd` and `pointcloud_0001.pcd`, recorded using a 3D LiDAR mounted on a TurtleBot.

- We selected **Point-to-Point ICP** as our registration strategy.
- The point clouds were **voxel downsampled** for faster processing.
- An **initial transformation matrix** was randomly generated using a valid orthonormal rotation matrix and small translation offsets.
- The ICP algorithm was then executed to refine this transformation and align the source point cloud with the target.

Results

Metric	Initial Transformation	After ICP
Fitness	0.0311	**0.0360**
Inlier RMSE	0.0690	**0.0651**

- The fitness score slightly improved after ICP, indicating a better overlap between the two point clouds.
- The RMSE reduced, reflecting improved point correspondence accuracy.

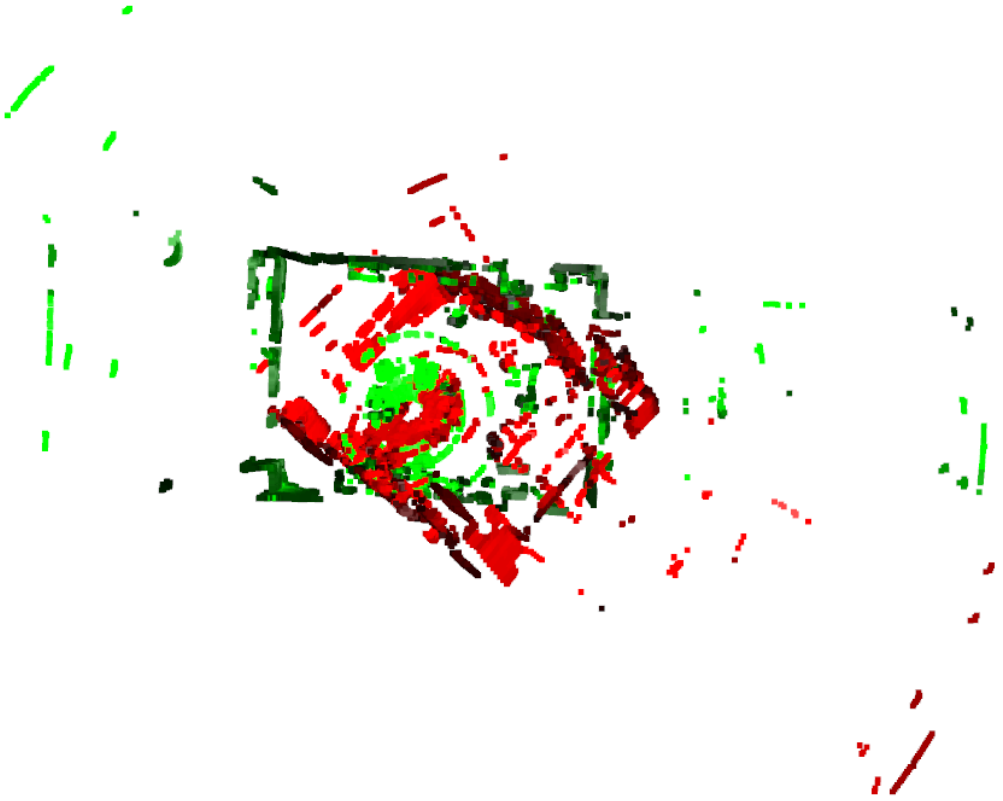
Estimated Transformation Matrix (After ICP):

[[0.66587624 0.60855351 0.43159176 0.17543967]

 [-0.28463572 0.74195671 -0.6070278 -0.00261867]

[0.68963131 -0.28135896 -0.66726741 -0.00356774]

[0. 0. 0. 1.]]



This figure shows the source point cloud (in **red**) and the target point cloud (in **green**) after registration.

2. To improve registration accuracy and robustness, I conducted a series of experiments by varying:

- The **initial transformation matrix** (identity, random orthonormal, and RANSAC-based),
- The **ICP method** (point-to-point vs point-to-plane),
- The **maximum correspondence threshold**.

Results

Init Guess	Method	Threshold	Fitness	Inlier RMSE	‖T_est - T_init‖
Identity	Point-to-Point	0.100	0.9937	0.0206	0.0044
Random	Point-to-Point	0.100	0.0474	0.0653	0.1564
Random	Point-to-Plane	0.100	0.0462	0.0651	0.2719
Identity	Point-to-Plane	0.050	0.9674	**0.0179**	**0.0005**
RANSAC	Point-to-Point	0.100	0.9937	0.0206	0.0118

Best configuration: *Identity matrix + Point-to-Plane ICP + 0.05 threshold*
This configuration yielded the **lowest inlier RMSE (0.0179)** and **minimum transformation correction (‖T_est - T_init‖ = 0.0005)**, indicating a highly accurate alignment with minimal deviation from the initial guess.

Best Estimated Transformation Matrix

The learned transformation matrix from the best configuration (Identity Init + Point-to-Plane ICP, threshold = 0.05) is:

[[9.99999995e-01 -5.12637423e-05 -8.79082824e-05 -4.56397355e-04]

 [5.12554576e-05 9.99999994e-01 -9.42423543e-05 2.32611106e-05]

 [8.79131131e-05 9.42378481e-05 9.99999992e-01 -6.31491481e-05]

 [0.00000000e+00 0.00000000e+00 0.00000000e+00 1.00000000e+00]]

3. Based on the hyperparameter tuning in Part 2, the best result was achieved using the following configuration:

- **Initialization**: Identity matrix
- **ICP Type**: Point-to-Plane
- **Correspondence Threshold**: 0.05
- **Fitness**: 0.9674
- **Inlier RMSE**: 0.0179
- ‖**T_est - T_init**‖: 0.0005

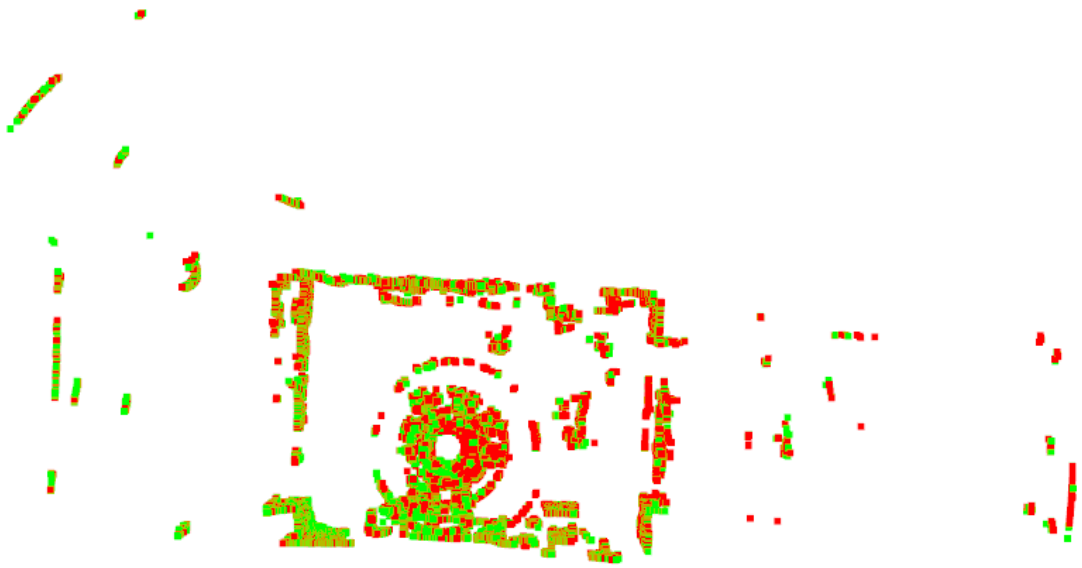
Using the above configuration, we applied the estimated transformation matrix to the original source point cloud (`pointcloud_0000.pcd`). The matrix used is:

[[9.99999995e-01 -5.12637423e-05 -8.79082824e-05 -4.56397355e-04]

 [5.12554576e-05 9.99999994e-01 -9.42423543e-05 2.32611106e-05]

 [8.79131131e-05 9.42378481e-05 9.99999992e-01 -6.31491481e-05]

 [0.00000000e+00 0.00000000e+00 0.00000000e+00 1.00000000e+00]]



This figure shows the transformed source point cloud (**red**) overlaid on the target point cloud (**green**).

- The near-perfect alignment seen in the visualization validates the numerical results.
- The **high fitness score (0.967)** indicates that a large proportion of the source points found close correspondences in the target cloud.
- The **low RMSE (0.0179)** implies minimal alignment error between corresponding points.
- The success of the identity initialization suggests that the point clouds were already closely aligned, and the point-to-plane ICP only needed to make fine adjustments.

4. To evaluate the global performance of point cloud registration, I applied **point-to-point ICP** iteratively across a sequence of **100 point cloud frames** (`pointcloud_0000.pcd` to `pointcloud_0396.pcd`).

- **Pairwise ICP** was applied between every two consecutive frames (4-frame skips).
- The **estimated transformation** from each ICP step was accumulated to compute a **global transformation** for each frame.
- Each point cloud was transformed using its global pose and merged to create a **single registered map**.

- The **translation component** of each global transformation matrix was recorded to estimate the **3D trajectory** of the TurtleBot.

Registered pointcloud_0000.pcd -> pointcloud_0004.pcd | Fitness: 0.9937

Registered pointcloud_0004.pcd -> pointcloud_0008.pcd | Fitness: 0.9915

Registered pointcloud_0008.pcd -> pointcloud_0012.pcd | Fitness: 0.9912

Registered pointcloud_0012.pcd -> pointcloud_0016.pcd | Fitness: 0.9909

Registered pointcloud_0016.pcd -> pointcloud_0020.pcd | Fitness: 0.9917

Registered pointcloud_0020.pcd -> pointcloud_0024.pcd | Fitness: 0.9887

Registered pointcloud_0024.pcd -> pointcloud_0028.pcd | Fitness: 0.9881

Registered pointcloud_0028.pcd -> pointcloud_0032.pcd | Fitness: 0.9918

Registered pointcloud_0032.pcd -> pointcloud_0036.pcd | Fitness: 0.9844

Registered pointcloud_0036.pcd -> pointcloud_0040.pcd | Fitness: 0.9600

Registered pointcloud_0040.pcd -> pointcloud_0044.pcd | Fitness: 0.9683

Registered pointcloud_0044.pcd -> pointcloud_0048.pcd | Fitness: 0.9671

Registered pointcloud_0048.pcd -> pointcloud_0052.pcd | Fitness: 0.9781

Registered pointcloud_0052.pcd -> pointcloud_0056.pcd | Fitness: 0.9785

Registered pointcloud_0056.pcd -> pointcloud_0060.pcd | Fitness: 0.9677

Registered pointcloud_0060.pcd -> pointcloud_0064.pcd | Fitness: 0.9749

Registered pointcloud_0064.pcd -> pointcloud_0068.pcd | Fitness: 0.9692

Registered pointcloud_0068.pcd -> pointcloud_0072.pcd | Fitness: 0.9831

Registered pointcloud_0072.pcd -> pointcloud_0076.pcd | Fitness: 0.9689

Registered pointcloud_0076.pcd -> pointcloud_0080.pcd | Fitness: 0.9807

Registered pointcloud_0080.pcd -> pointcloud_0084.pcd | Fitness: 0.9754

Registered pointcloud_0084.pcd -> pointcloud_0088.pcd | Fitness: 0.9842

Registered pointcloud_0088.pcd -> pointcloud_0092.pcd | Fitness: 0.9758

Registered pointcloud_0092.pcd -> pointcloud_0096.pcd | Fitness: 0.9881

Registered pointcloud_0096.pcd -> pointcloud_0100.pcd | Fitness: 0.9781

Registered pointcloud_0100.pcd -> pointcloud_0104.pcd | Fitness: 0.9863

Registered pointcloud_0104.pcd -> pointcloud_0108.pcd | Fitness: 0.9804

Registered pointcloud_0108.pcd -> pointcloud_0112.pcd | Fitness: 0.9875

Registered pointcloud_0112.pcd -> pointcloud_0116.pcd | Fitness: 0.9696

Registered pointcloud_0116.pcd -> pointcloud_0120.pcd | Fitness: 0.9849

Registered pointcloud_0120.pcd -> pointcloud_0124.pcd | Fitness: 0.9773

Registered pointcloud_0124.pcd -> pointcloud_0128.pcd | Fitness: 0.9878

Registered pointcloud_0128.pcd -> pointcloud_0132.pcd | Fitness: 0.9782

Registered pointcloud_0132.pcd -> pointcloud_0136.pcd | Fitness: 0.9874

Registered pointcloud_0136.pcd -> pointcloud_0140.pcd | Fitness: 0.9803

Registered pointcloud_0140.pcd -> pointcloud_0144.pcd | Fitness: 0.9873

Registered pointcloud_0144.pcd -> pointcloud_0148.pcd | Fitness: 0.9834

Registered pointcloud_0148.pcd -> pointcloud_0152.pcd | Fitness: 0.9847

Registered pointcloud_0152.pcd -> pointcloud_0156.pcd | Fitness: 0.9835

Registered pointcloud_0156.pcd -> pointcloud_0160.pcd | Fitness: 0.9888

Registered pointcloud_0160.pcd -> pointcloud_0164.pcd | Fitness: 0.9864

Registered pointcloud_0164.pcd -> pointcloud_0168.pcd | Fitness: 0.9909

Registered pointcloud_0168.pcd -> pointcloud_0172.pcd | Fitness: 0.9870

Registered pointcloud_0172.pcd -> pointcloud_0176.pcd | Fitness: 0.9846

Registered pointcloud_0176.pcd -> pointcloud_0180.pcd | Fitness: 0.9826

Registered pointcloud_0180.pcd -> pointcloud_0184.pcd | Fitness: 0.9885

Registered pointcloud_0184.pcd -> pointcloud_0188.pcd | Fitness: 0.9891

Registered pointcloud_0188.pcd -> pointcloud_0192.pcd | Fitness: 0.9819

Registered pointcloud_0192.pcd -> pointcloud_0196.pcd | Fitness: 0.9887

Registered pointcloud_0196.pcd -> pointcloud_0200.pcd | Fitness: 0.9844

Registered pointcloud_0200.pcd -> pointcloud_0204.pcd | Fitness: 0.9878

Registered pointcloud_0204.pcd -> pointcloud_0208.pcd | Fitness: 0.9838

Registered pointcloud_0208.pcd -> pointcloud_0212.pcd | Fitness: 0.9891

Registered pointcloud_0212.pcd -> pointcloud_0216.pcd | Fitness: 0.9860

Registered pointcloud_0216.pcd -> pointcloud_0220.pcd | Fitness: 0.9864

Registered pointcloud_0220.pcd -> pointcloud_0224.pcd | Fitness: 0.9805

Registered pointcloud_0224.pcd -> pointcloud_0228.pcd | Fitness: 0.9917

Registered pointcloud_0228.pcd -> pointcloud_0232.pcd | Fitness: 0.9853

Registered pointcloud_0232.pcd -> pointcloud_0236.pcd | Fitness: 0.9860

Registered pointcloud_0236.pcd -> pointcloud_0240.pcd | Fitness: 0.9901

Registered pointcloud_0240.pcd -> pointcloud_0244.pcd | Fitness: 0.9886

Registered pointcloud_0244.pcd -> pointcloud_0248.pcd | Fitness: 0.9892

Registered pointcloud_0248.pcd -> pointcloud_0252.pcd | Fitness: 0.9898

Registered pointcloud_0252.pcd -> pointcloud_0256.pcd | Fitness: 0.9887

Registered pointcloud_0256.pcd -> pointcloud_0260.pcd | Fitness: 0.9888

Registered pointcloud_0260.pcd -> pointcloud_0264.pcd | Fitness: 0.9879

Registered pointcloud_0264.pcd -> pointcloud_0268.pcd | Fitness: 0.9842

Registered pointcloud_0268.pcd -> pointcloud_0272.pcd | Fitness: 0.9869

Registered pointcloud_0272.pcd -> pointcloud_0276.pcd | Fitness: 0.9900

Registered pointcloud_0276.pcd -> pointcloud_0280.pcd | Fitness: 0.9886

Registered pointcloud_0280.pcd -> pointcloud_0284.pcd | Fitness: 0.9866

Registered pointcloud_0284.pcd -> pointcloud_0288.pcd | Fitness: 0.9881

Registered pointcloud_0288.pcd -> pointcloud_0292.pcd | Fitness: 0.9906

Registered pointcloud_0292.pcd -> pointcloud_0296.pcd | Fitness: 0.9888

Registered pointcloud_0296.pcd -> pointcloud_0300.pcd | Fitness: 0.9880

Registered pointcloud_0300.pcd -> pointcloud_0304.pcd | Fitness: 0.9897

Registered pointcloud_0304.pcd -> pointcloud_0308.pcd | Fitness: 0.9894

Registered pointcloud_0308.pcd -> pointcloud_0312.pcd | Fitness: 0.9889

Registered pointcloud_0312.pcd -> pointcloud_0316.pcd | Fitness: 0.9881

Registered pointcloud_0316.pcd -> pointcloud_0320.pcd | Fitness: 0.9864

Registered pointcloud_0320.pcd -> pointcloud_0324.pcd | Fitness: 0.9925

Registered pointcloud_0324.pcd -> pointcloud_0328.pcd | Fitness: 0.9911

Registered pointcloud_0328.pcd -> pointcloud_0332.pcd | Fitness: 0.9853

Registered pointcloud_0332.pcd -> pointcloud_0336.pcd | Fitness: 0.9861

Registered pointcloud_0336.pcd -> pointcloud_0340.pcd | Fitness: 0.9896

Registered pointcloud_0340.pcd -> pointcloud_0344.pcd | Fitness: 0.9883

Registered pointcloud_0344.pcd -> pointcloud_0348.pcd | Fitness: 0.9846

Registered pointcloud_0348.pcd -> pointcloud_0352.pcd | Fitness: 0.9849

Registered pointcloud_0352.pcd -> pointcloud_0356.pcd | Fitness: 0.9896

Registered pointcloud_0356.pcd -> pointcloud_0360.pcd | Fitness: 0.9900

Registered pointcloud_0360.pcd -> pointcloud_0364.pcd | Fitness: 0.9872

Registered pointcloud_0364.pcd -> pointcloud_0368.pcd | Fitness: 0.9854

Registered pointcloud_0368.pcd -> pointcloud_0372.pcd | Fitness: 0.9884

Registered pointcloud_0372.pcd -> pointcloud_0376.pcd | Fitness: 0.9901

Registered pointcloud_0376.pcd -> pointcloud_0380.pcd | Fitness: 0.9832

Registered pointcloud_0380.pcd -> pointcloud_0384.pcd | Fitness: 0.9874

Registered pointcloud_0384.pcd -> pointcloud_0388.pcd | Fitness: 0.9847

Registered pointcloud_0388.pcd -> pointcloud_0392.pcd | Fitness: 0.9891

Registered pointcloud_0392.pcd -> pointcloud_0396.pcd | Fitness: 0.9828

Trajectory:

```
x,y,z
0,0,0
-0.0043660826205065215,-0.00016952890859987547,5.251785200055065e-06
-0.009152816027674277,-0.0008648978095404534,2.3424807349124813e-06
-0.005039400761475491,-0.0025294558970605863,-1.0675960874680467e-06
-0.007498011966125214,-0.00019508470876557823,3.9504206313727394e-05
-0.008053494479506379,-0.0008949925877909074,0.00010525243392392223
-0.012002413938078065,-0.0009755353565196759,3.5120991286818895e-05
-0.007637615242239352,0.0009430250748799069,0.00010406167463968856
-0.005914193149483954,0.003458054639122488,0.00015679659398876096
0.0001193008608031329,0.0025409629149067622,0.0003412504290226131
0.0017961997647422054,-0.0004255855732969373,0.0004896642586124315
0.0011862999635071985,-0.00400555599639389,0.0005362236780599271
0.0009182165493810945,-0.006257040866450254,0.0005657850699405314
-7.819803541445427e-05,-0.009776177441207267,0.0006465700461654895
-0.0012029680180015879,-0.012911044760771047,0.000644442987387952
```

```
-0.003814010823541326,-0.01845770554664803,0.0007154245187361631
-0.0053948029655769185,-0.020933339298173988,0.0008319856612639093
-0.008429349711718298,-0.024100352609585336,0.0008963012248390961
-0.009881345603323407,-0.02546059022776425,0.0008969908603519333
-0.011931587796477836,-0.028809432126608615,0.0008413385981335652
-0.01297193999198645,-0.03028661631792569,0.0008831662085904139
-0.014285458328856993,-0.03347535245294288,0.0007951007141514124
-0.014964432013264932,-0.034778988227149574,0.0008746869643309652
-0.017139159216700044,-0.03733791757199622,0.0007637296792556474
-0.018479164691301207,-0.03809211690955326,0.0009055274167620322
-0.020619564877452402,-0.04104560483633177,0.0009708387446446787
-0.020862456670043397,-0.04177630896398181,0.001074530713557033
-0.022677266444912657,-0.043569881186280225,0.0009450717128086256
-0.0226095373514193,-0.04264206334845639,0.001074312143693509
-0.02591414593817742,-0.04376471985898808,0.0010288377930601715
-0.026693539494835425,-0.04551443431193998,0.001051514953616136
-0.030278427730626136,-0.04838863465009295,0.0009904268774796247
-0.030900176578996224,-0.04914247916154738,0.001152692271255882
-0.03342658435500966,-0.05107629526109681,0.0010360670924483195
-0.034510712411705044,-0.05232560211698997,0.0011089092575289527
-0.03767125056458774,-0.05434942985540462,0.0009279049042787266
-0.03795281294785317,-0.056852350474993985,0.0009946156597578884
-0.039024333095868245,-0.058957130446384697,0.0009439124005942654
```

```
-0.03954430003461621,-0.05992882985743454,0.0008400956628468162
-0.04124420622244345,-0.061210178728521705,0.0006765547937718071
-0.04165641105288191,-0.06288563643830392,0.0007248120504238393
-0.04452443563681413,-0.06337791419236145,0.0007620452715278209
-0.046967470290636165,-0.06384190829774845,0.0006820034020514056
-0.050019417844588056,-0.06599290130609108,0.0006340286590258012
-0.05318663513579704,-0.06456526956177627,0.0006512446818787676
-0.05939918557488429,-0.064133610330228,0.0006528998307570948
-0.06553674456478521,-0.060669743876200666,0.0006277966102540579
-0.06338445487919692,-0.057969044741430324,0.0006042898570725258
-0.06181776118509736,-0.058209210778614844,0.0004696149207912085
-0.060835757682202556,-0.05847748285339164,0.0005077147656849711
-0.06015556303577791,-0.058797501056072184,0.0005545460558230201
-0.05839431337424539,-0.060593372558217826,0.0006456724315926454
-0.05570353228171102,-0.06026325837979646,0.0006297171195764498
-0.055069266398081684,-0.060242461855761216,0.0006200309490380298
-0.053686879149599145,-0.06033326548058963,0.000624301988052966
-0.052399688257393266,-0.058590830496124444,0.0007150116368076724
-0.051560710794059196,-0.05712261157151264,0.0006194085393818942
-0.05086431817928784,-0.05740996523672961,0.0006429244625099639
-0.0489824006869683,-0.0591169234107433,0.0007169500141541872
-0.04839633542660667,-0.05828604846144124,0.0008915729563104534
-0.04814678916047527,-0.05664913707051733,0.0008922285665467027
```

-0.04693779272207393,-0.0573461857921602,0.0008612826737263824

-0.04569123378176049,-0.05800376887148931,0.0010884857904380986

-0.045411313967878085,-0.05626426486731831,0.0012680856570418749

-0.045470154780877,-0.0564631286086613,0.0012196960916872265

-0.04599077655213694,-0.05415003687163548,0.0010823543508698102

-0.045515362153490786,-0.05271633254432825,0.001164170657658072

-0.044726870522402914,-0.05122716572761566,0.0013433296048011082

-0.045660097292985376,-0.04964334673918672,0.0013286992093026829

-0.04517739660426816,-0.04987988090122164,0.0011554471973217284

-0.04271842759961138,-0.047810295486488914,0.0012410197669138938

-0.04128268284228807,-0.04441446522501568,0.0013564339516343488

-0.04105565474829679,-0.041934581448628716,0.0014143482294145544

-0.03960274330418402,-0.041113631798256174,0.0014151434416066764

-0.037586218613205505,-0.04025006968785214,0.0015286735154784261

-0.03644056591799288,-0.039427838088668364,0.0016370535458498899

-0.03555141193948189,-0.03876623766430633,0.0016065438141000545

-0.03364907403464296,-0.03866222207247682,0.0014512251425556073

-0.03181398867119866,-0.03730645920622639,0.0015215885455604561

-0.030574223145178395,-0.036690489584590744,0.0016368476533528615

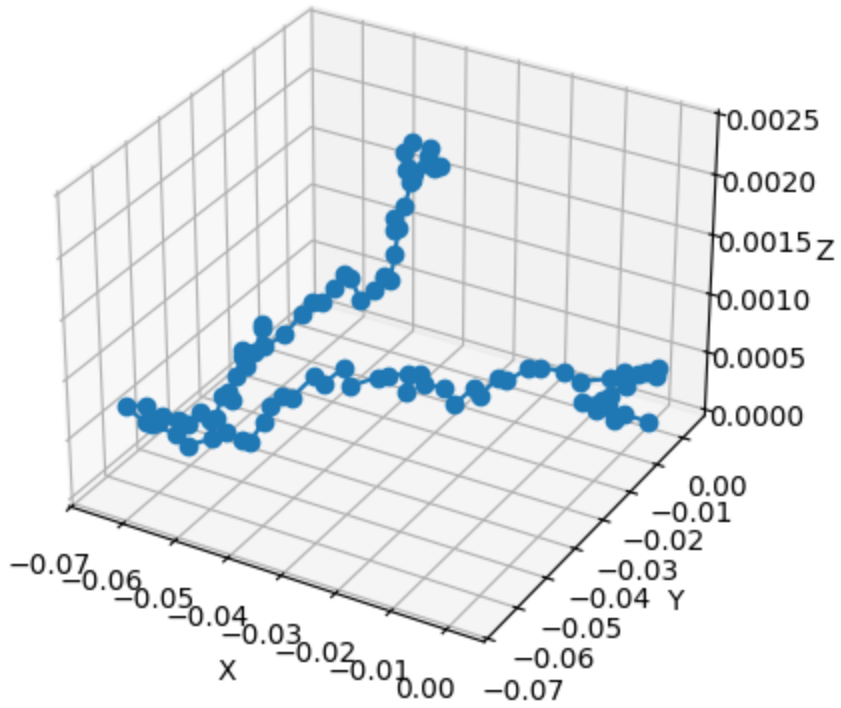
-0.03076867353927092,-0.036004757375339896,0.0016030138605045683

-0.02972575774249185,-0.03590400780719109,0.001596111463276597

-0.029553123135695428,-0.03472800323482742,0.001794358484670498

-0.028979058569657195,-0.03436201374374597,0.0020098729591888583

```
-0.029178589977891054,-0.03541378340081337,0.0021085584574104858
-0.029405448804045935,-0.0354363741865469,0.0020098594542242733
-0.028673013791999408,-0.03298495055897715,0.002160695135997903
-0.028584518991962283,-0.030552914101489585,0.002355254942130736
-0.029171752535482917,-0.029916266306177407,0.002292465711150108
-0.03118693825808143,-0.028230428495087747,0.002320921576030658
-0.0322177599952533,-0.026828669479891556,0.0024253007259555817
-0.031826794200093614,-0.025273429055791604,0.002476474543949314
-0.03257740783446562,-0.025061496641423174,0.002255885074259409
-0.032425159225295826,-0.024611252263183468,0.0021312529441158898
-0.03284706788482216,-0.022141120045027496,0.0021616229090021728
-0.03213055245804159,-0.0184433630878969,0.0023028524099868823
-0.033077923746051084,-0.017239388791452478,0.002194656710041377
-0.03344698504363322,-0.01693033319258584,0.0021121490097124
-0.032889276567064724,-0.015697550767446788,0.0020626350888426235
-0.0324399616956926,-0.014267522963597292,0.0020549767501461765
```


Estimated 3D Trajectory of TurtleBot
0.0025
0.0020
0.0015
Z
0.0010
0.0005
0.0000
0.00
−0.01
−0.02
−0.03
−0.04
Y
−0.05
−0.06
−0.07
−0.07
−0.06
−0.05
−0.04
−0.03
−0.02
−0.01
0.00
X

References:

3.

1. **OpenCV Camera Calibration Tutorial**
   OpenCV: Camera Calibration

2. **OpenCV `cv2.findChessboardCorners` Documentation**
   Camera Calibration and 3D Reconstruction

3. **OpenCV `cv2.cornerSubPix` Documentation**
   Detecting corners location in subpixels

4. **OpenCV `cv2.calibrateCamera` Documentation**
   Camera Calibration and 3D Reconstruction

5. **LearnOpenCV: Camera Calibration Using OpenCV**
   https://learnopencv.com/camera-calibration-using-opencv/

6. **BoofCV Tutorial on Camera Calibration**
   Tutorial Camera Calibration - BoofCV

7. **OpenCV API: `cv2.projectPoints` for Reprojection**
   Camera Calibration and 3D Reconstruction

8. **OpenCvSharp `Cv2.CalibrateCamera` Documentation**
   Cv2.Undistort Method

4.

1. SIFT Keypoint Detection and Description

   OpenCV: Introduction to SIFT (Scale-Invariant Feature Transform)

2. BruteForce and FLANN Feature Matching

   OpenCV: Feature Matching

cv::FlannBasedMatcher Class Reference

3. Homography Estimation with RANSAC

   Feature Matching + Homography to find Objects

   RANSAC | LearnOpenCV

4. Perspective Warping and Stitching

   cv2.warpPerspective() | TheAILearner

5. Image Clustering using KMeans (for multi-stitching)

   2.3. Clustering — scikit-learn 1.6.1 documentation

   5.

      1. **Open3D ICP Registration Tutorial**

         https://www.open3d.org/docs/release/tutorial/pipelines/icp_registration.html

      2. **Scipy `ortho_group` Documentation**

         https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ortho_group.html

      3. **Open3D Normal Estimation (v0.7.0)**

         https://www.open3d.org/docs/0.7.0/python_api/open3d.geometry.estimate_normals.html