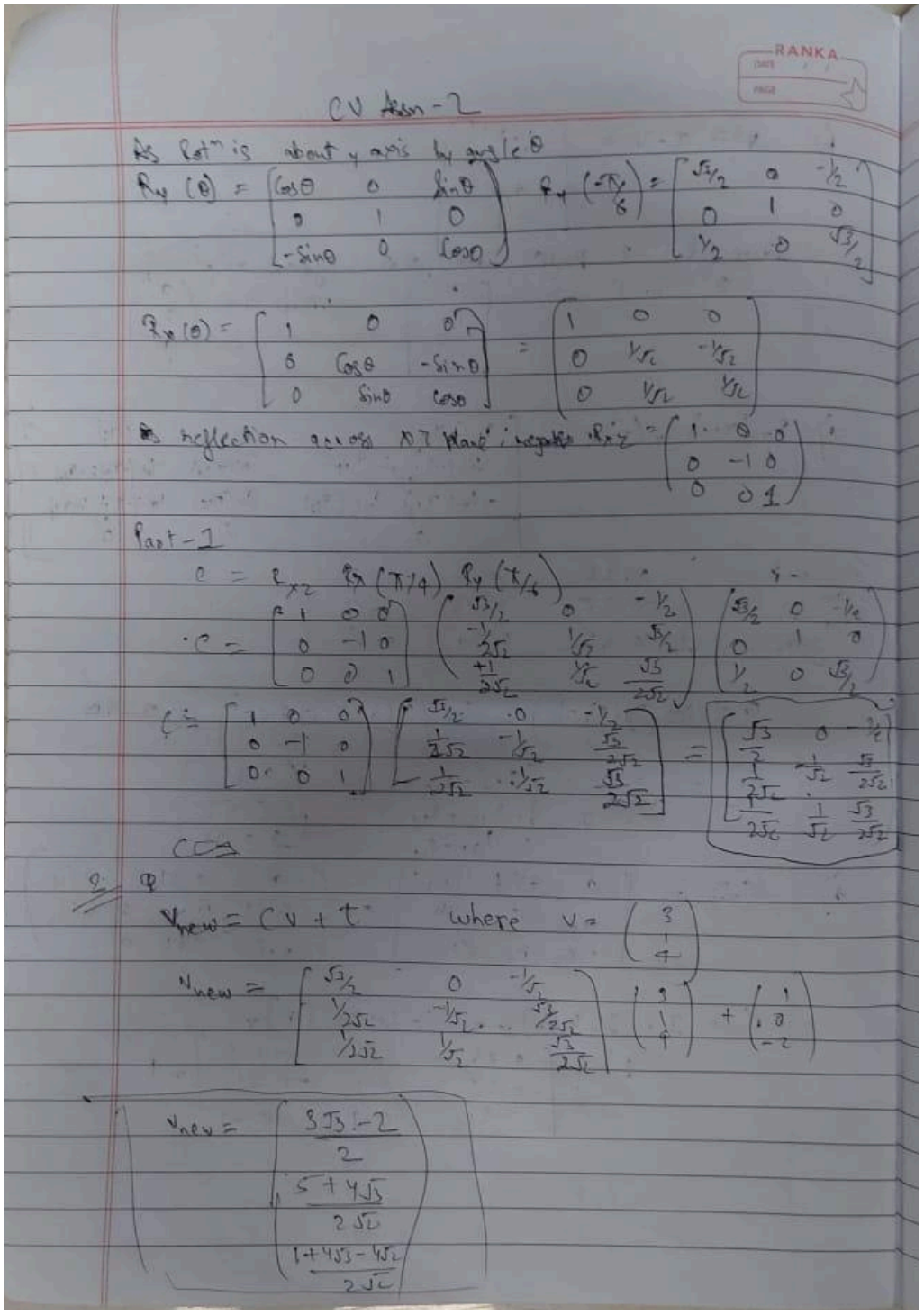
REPORT

-Krishna Shukla(2022254)

Theoretical Questions(Questions 1 and 2)



$u = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \rightarrow u_{new} = \underbrace{Cu}_{0} + t = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix}$ → origin no

3

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

$\theta = \arccos\left(\frac{tr(\text{Combined Rotn}) - 1}{2}\right)$

$\theta = \arccos\left(\frac{\sqrt{6} + 2 + \sqrt{3} - 2\sqrt{2}}{4\sqrt{2}}\right) \approx 55^\circ$

10

$u = \frac{1}{2\sin\theta}\begin{bmatrix} C_{R32} - C_{R23} \\ C_{R13} - C_{R31} \\ C_{R21} - C_{R12} \end{bmatrix} \Rightarrow u = \begin{pmatrix} \left(\frac{1}{\sqrt{2}} + \frac{\sqrt{3}}{2\sqrt{2}}\right)\left(2\sin(55^\circ)\right) \\ \left(\frac{-1}{\sqrt{2}} - \frac{1}{2\sqrt{2}}\right)\left(2\sin(55^\circ)\right) \\ \left(\frac{-1}{\sqrt{2}} - 0\right)\left(2\sin(55^\circ)\right) \end{pmatrix}$

$u = \begin{bmatrix} 0.81 \\ -0.53 \\ -0.21 \end{bmatrix}$ → Axis of Rotn

11

$R = I + \sin\theta\,[u]_x + (1 - \cos\theta)\,[u_x]^2$

θ = Angle of Rotn $(u)_x$ → Skew symm matrix → $\begin{pmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{pmatrix}$

$R = I + 0.805\begin{bmatrix} 0 & 0.21 & -0.53 \\ -0.21 & 0 & -0.81 \\ 0.53 & 0.81 & 0 \end{bmatrix} + 0.407\begin{pmatrix} -0.32 & -0.43 & -0.17 \\ -0.43 & -0.071 & 0.11 \\ -0.17 & 0.11 & -0.95 \end{pmatrix}$

$= \begin{pmatrix} 0.86 & 0 & -0.50 \\ -0.35 & 0.70 & -0.61 \\ -0.38 & 0.70 & 0.61 \end{pmatrix} \approx \begin{bmatrix} \sqrt{3}/2 & 0 & -1/2 \\ -1/2\sqrt{2} & 1/\sqrt{2} & -\sqrt{3}/2\sqrt{2} \\ 1/2\sqrt{2} & 1/\sqrt{2} & \sqrt{3}/2\sqrt{2} \end{bmatrix}$

Hence proved

Hence Fatrigues and Rotation give the same answers.

Q.2 [Theory]

$x = K[R|t]X$, ($t = 0$ as pure Rotation R)

Camera 1 $\rightarrow x_1 = K_1[I|0]X$

Cam 2 $\rightarrow x_2 = K_2[R|0]X$

$x_1 = K_1 X \Rightarrow X = K_1^{-1} x_1$

$x_2 = K_2 R X$

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

$x_2 = (K_2 R K_1^{-1}) x_1 \rightsquigarrow x_2 = H' x_1$

$x_1 = (H')^{-1} x_2$

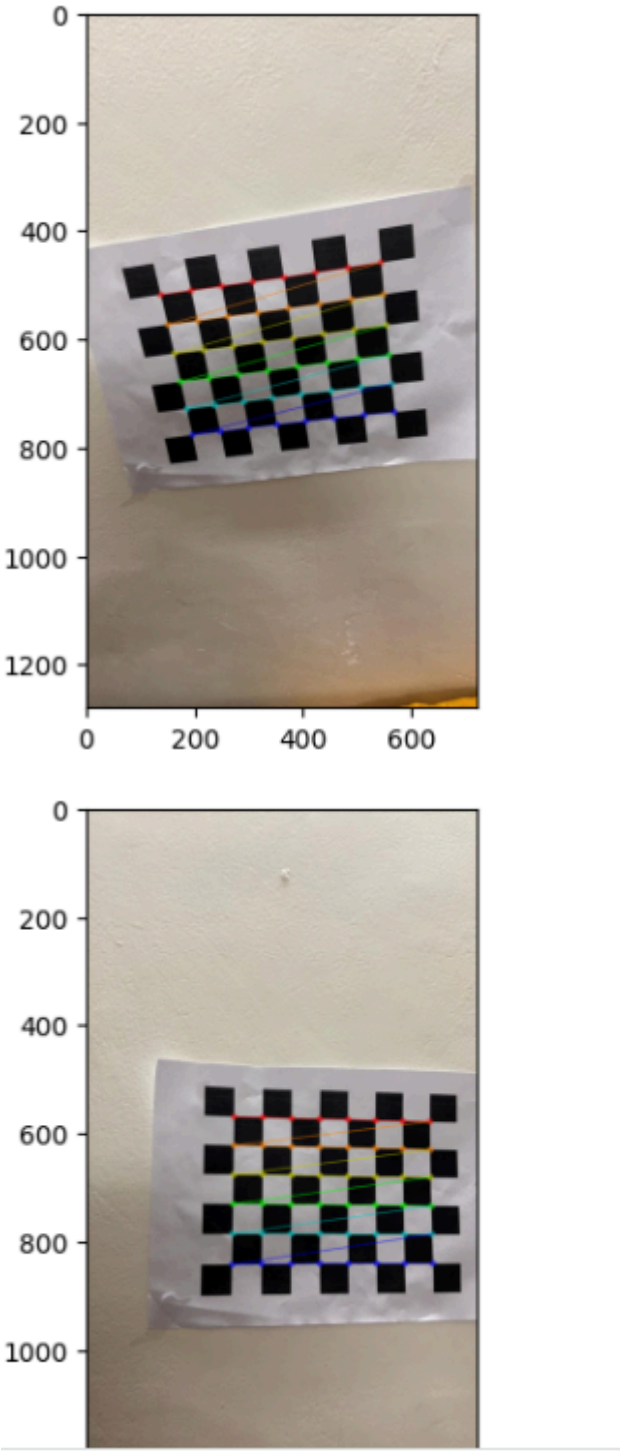
$= (K_1 R^{-1} K_2^{-1}) x_2$

As K_1 & K_2 are camera intrinsic parameters, they are invertible & R is a pure rotation matrix (orthonormal $R^T = R^{-1}$), thus identity

Coding Question(Questions 3,4)

Camera Calibration:

The image visualization for all image 2 examples are shown below



1)

```
Camera matrix :

[[956.63719897   0.         369.05485204]
 [  0.         957.55141318 651.41419982]
 [  0.           0.           1.        ]]
fx =  956.637198973105
fy =  957.5514131759558
cx =  369.0548520367341
cy =  651.414199816039
s =  0.0
```

2)

```
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/1.jpeg:
Rotation Matrix:
[[ 0.91736941 -0.19600125 -0.34643452]
 [-0.01803989  0.84899005 -0.5281008 ]
 [ 0.39762788  0.49071316  0.77530166]]
Translation Vector:
[[-3.93829473]
 [-3.61716296]
 [14.65911195]]

[[ 0.99699847  0.01387206 -0.0761683 ]
 [-0.02511535  0.9885627  -0.14870443]
 [ 0.07323431  0.15017108  0.9859439 ]]
```

```
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/2.jpeg:
Rotation Matrix:
[[ 0.91736941 -0.19600125 -0.34643452]
 [-0.01803989  0.84899005 -0.5281008 ]
 [ 0.39762788  0.49071316  0.77530166]]
Translation Vector:
[[-1.49771589]
 [-0.657919  ]
 [13.91817088]]
```

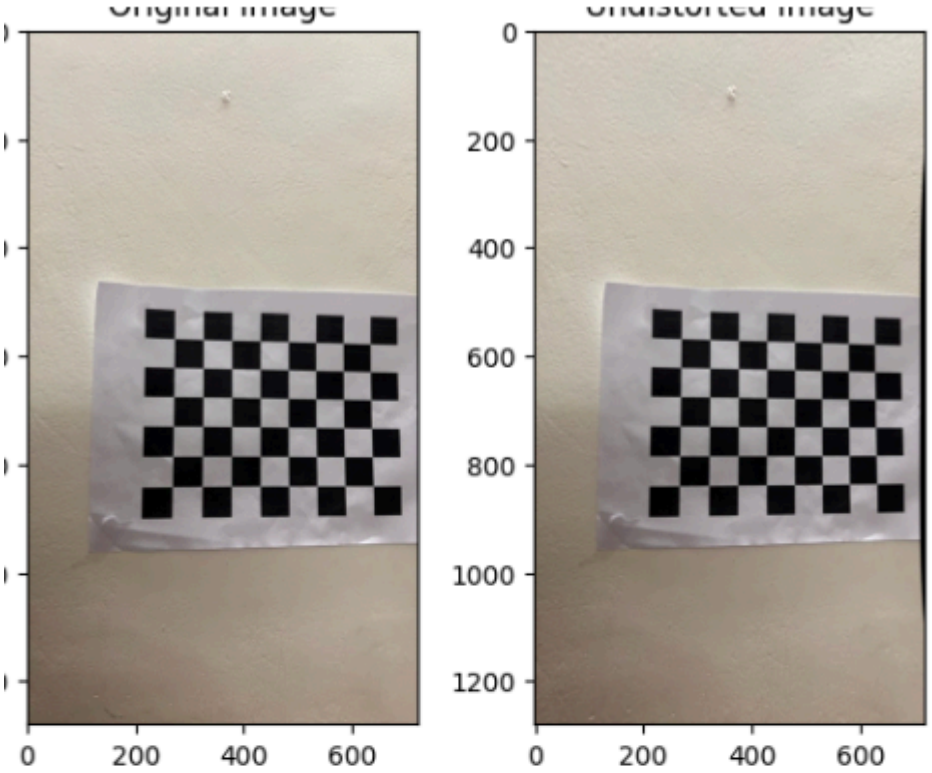
In notebook printed for all images

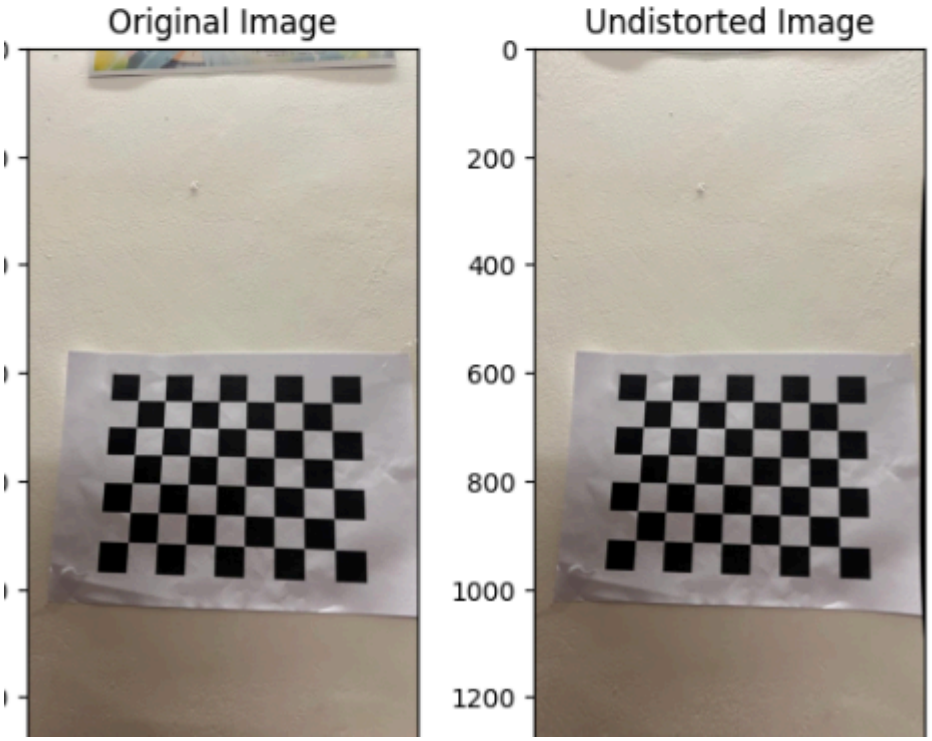
3)

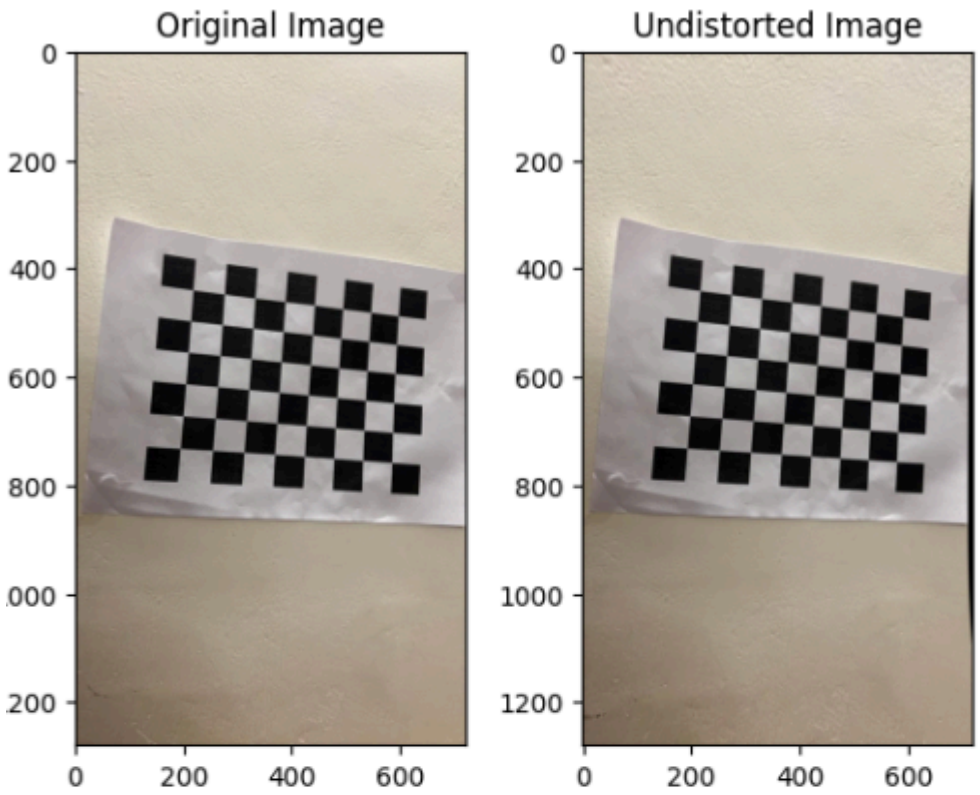
```
Distortion coefficients: [[ 0.19763718 -0.70686979  0.00336936  0.00698242  0.04879612]]
```

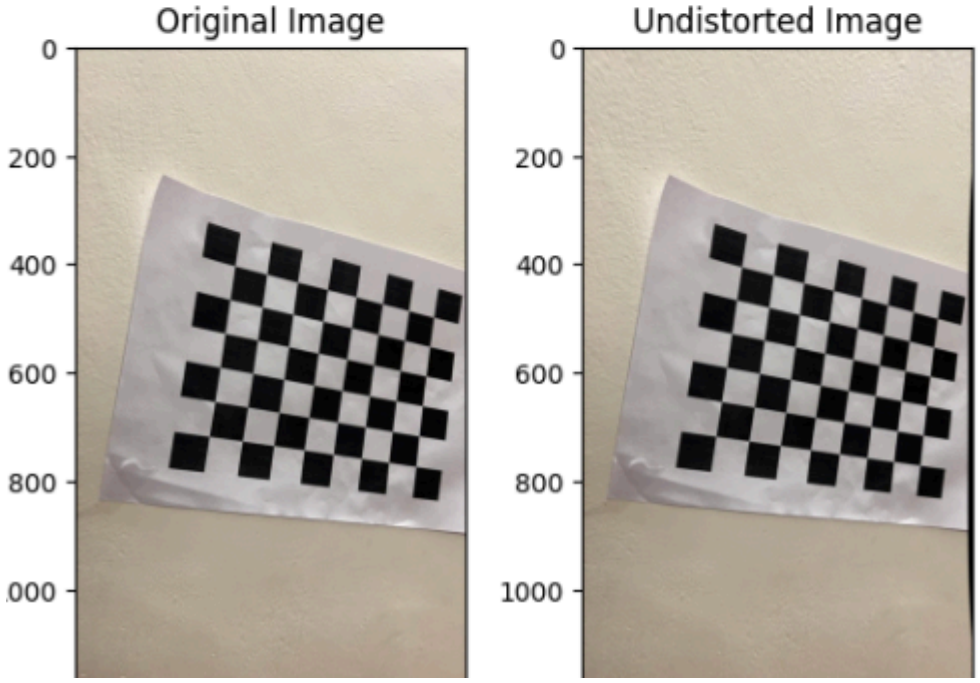
Where 0th value, 1st value, 4th value are k1,k2,k3 that is radial distortion coefficient whereas 2nd and 3rd value are p1 and p2 which is tangential distortion coefficient

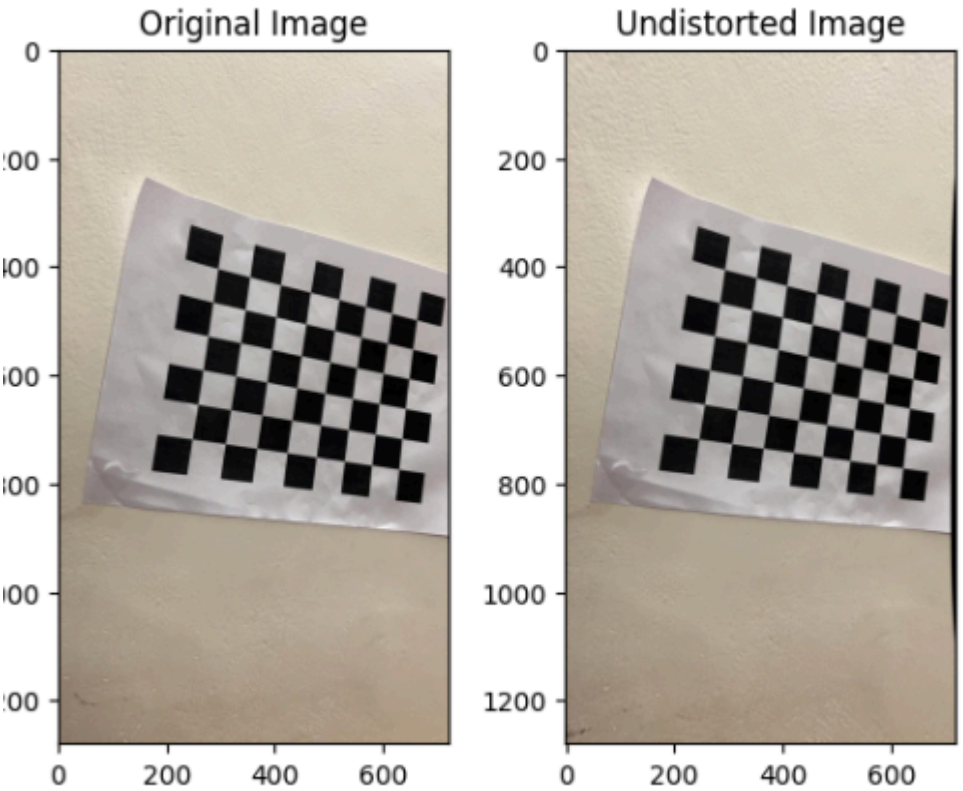
k1, k2, k3 = 0.19763718, -0.70686979, 0.04879612 [Radial]
p1, p2 = 0.00336936, 0.00698242 [Tangential]


0
200
400
600
0
200
400
600
800
1000
1200


Original Image
Undistorted Image
0
200
400
600
800
1000
1200


Original Image
Undistorted Image
0
200
400
600
800
000
200
1000
1200

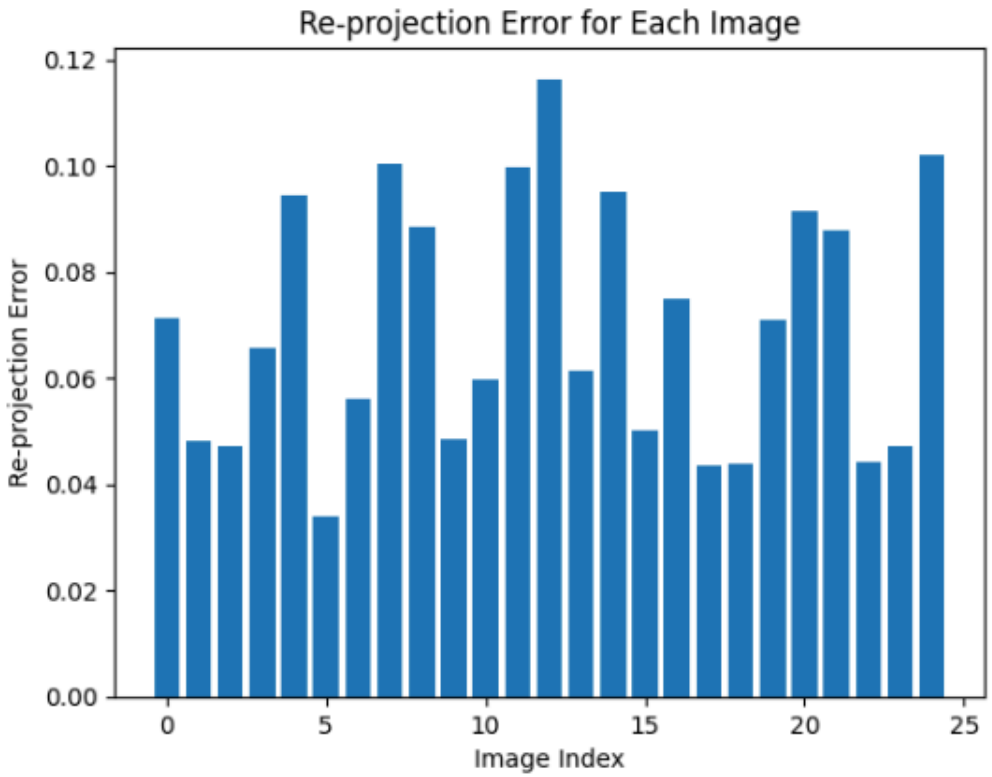

Original Image
Undistorted Image
0
200
400
600
800
000
1000



Comment-

After applying the radial distortion coefficients to undistort the images, it is observed that the previously curved straight lines, especially those located at the corners, have been corrected to appear straight. The distortion, which caused the lines to bow outward or inward, has been effectively compensated, resulting in a more geometrically accurate representation of the scene. This correction is particularly noticeable in the corners, where the distortion was most pronounced. The undistorted images now exhibit improved clarity and accuracy, highlighting the effectiveness of the radial distortion coefficients in restoring the true geometry of the captured scenes.

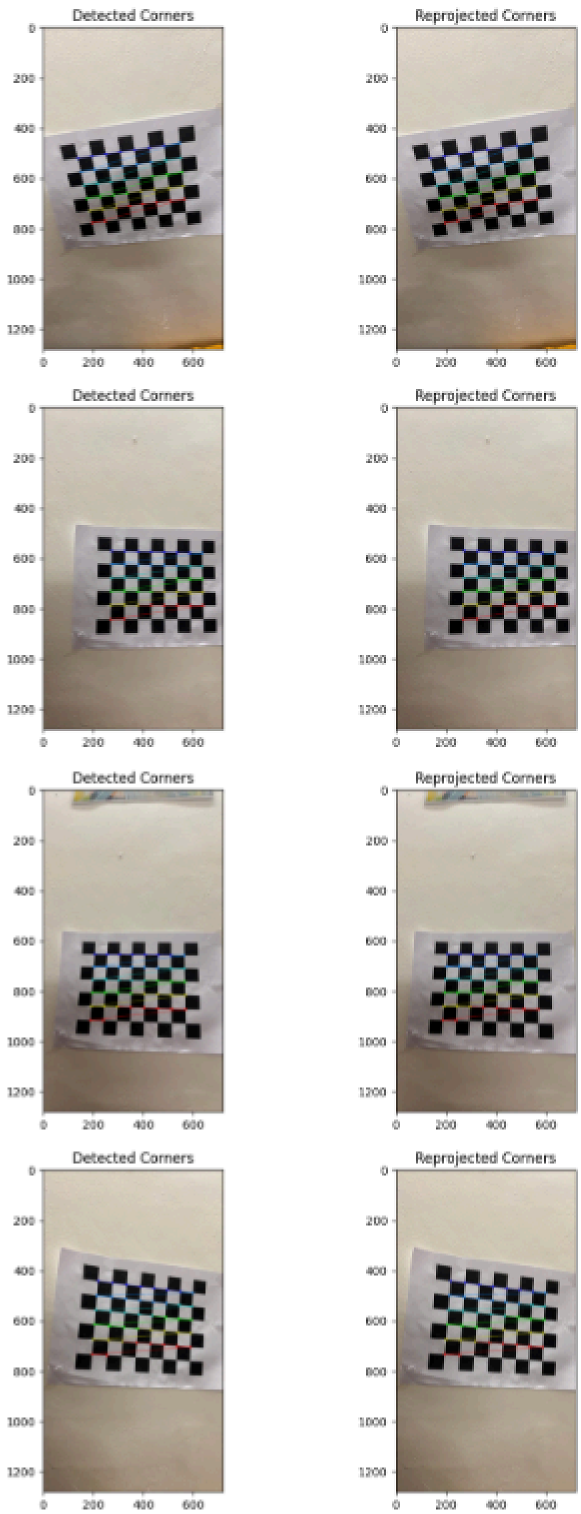
4)



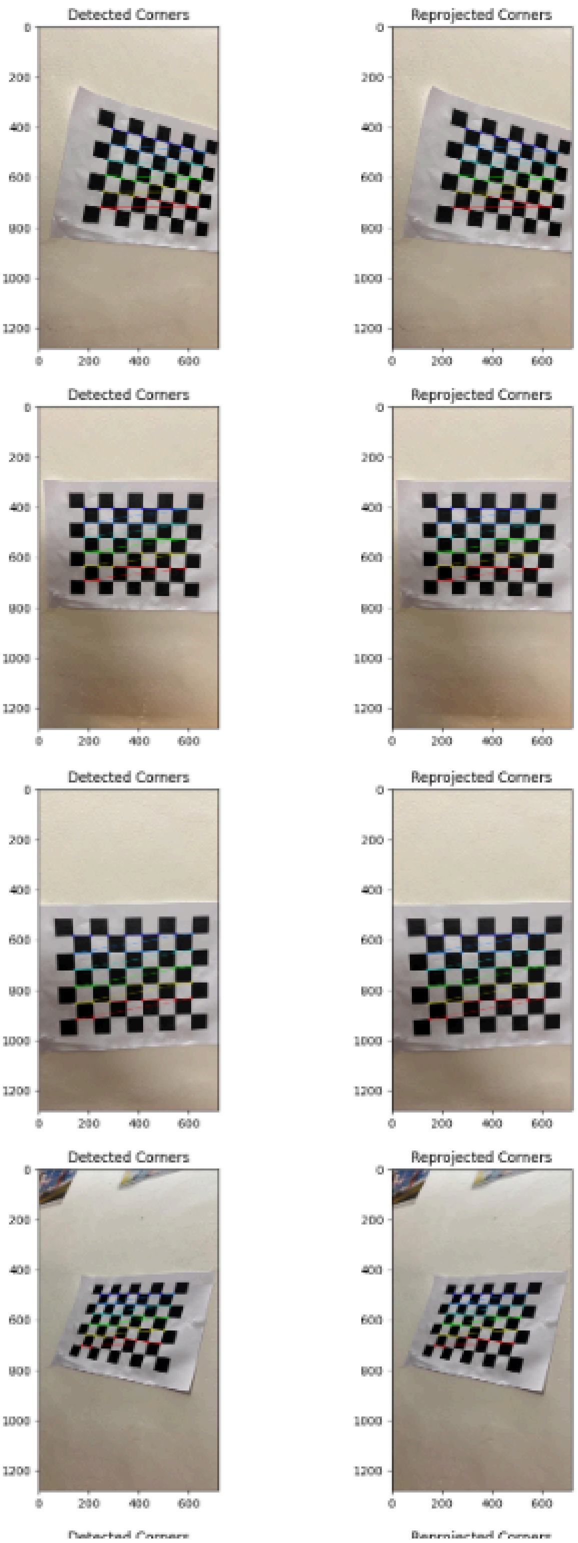

```
Mean re-projection error: 0.06971439945721061
Standard deviation of re-projection error: 0.023213790898823205
```

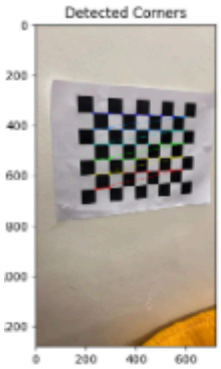
The reprojection error is computed by measuring the difference between the observed image points (i.e., the actual 2D points detected in the image) and the projected points obtained by projecting the corresponding 3D points back onto the image plane using the estimated camera calibration parameters. This involves first transforming the 3D world coordinates into the camera coordinate system using the extrinsic parameters (rotation and translation vectors), and then projecting these points onto the image plane using the intrinsic parameters (focal length, principal point, and distortion coefficients). The reprojection error is then calculated as the Euclidean distance between the observed 2D points and the projected 2D points for each image point. This error is typically averaged over all points and images to obtain the mean reprojection error, which provides an indication of the accuracy of the camera calibration process. A lower reprojection error signifies a more accurate calibration.

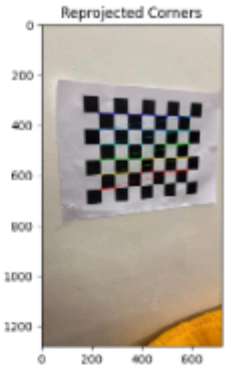
5)

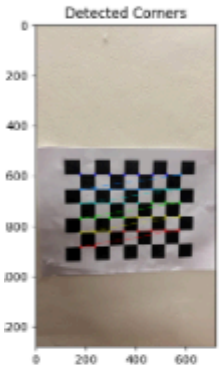


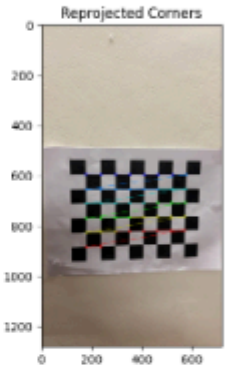
Detected Corners
Reprojected Corners
Detected Corners
Reprojected Corners
Detected Corners
Reprojected Corners
Detected Corners
Reprojected Corners
0
200
400
600
800
1000
1200

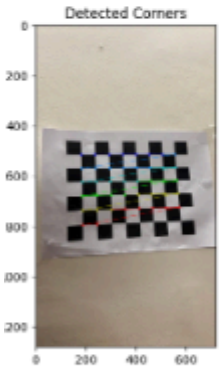

Detected Corners
Reprojected Corners
Detected Corners
Reprojected Corners
Detected Corners
Reprojected Corners
Detected Corners
Reprojected Corners

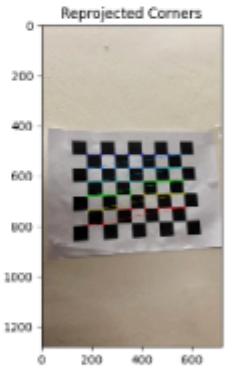

Detected Corners

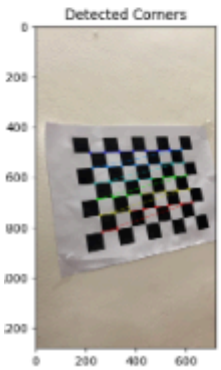

Reprojected Corners

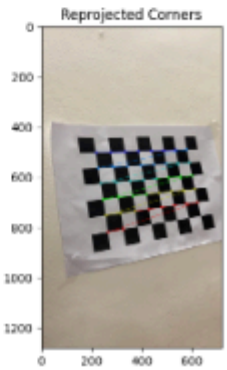

Detected Corners

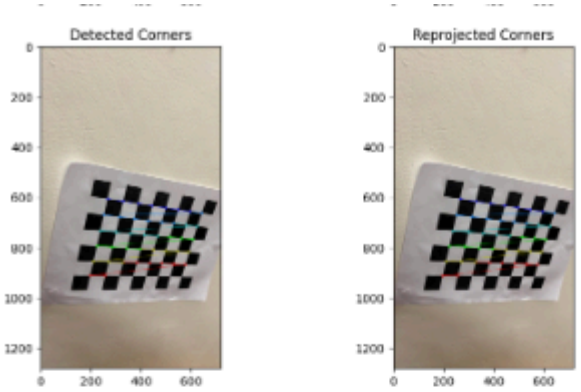

Reprojected Corners

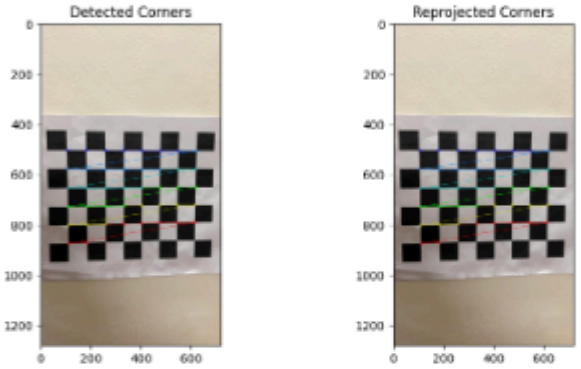

Detected Corners

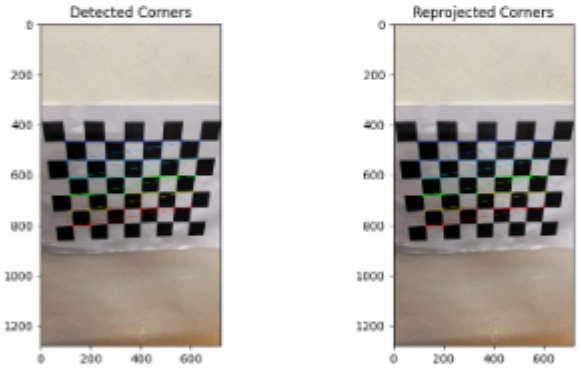

Reprojected Corners

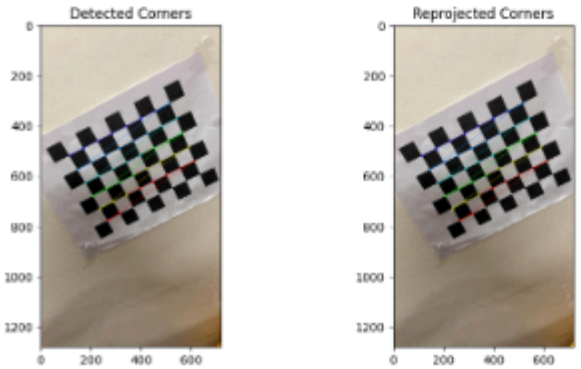

Detected Corners

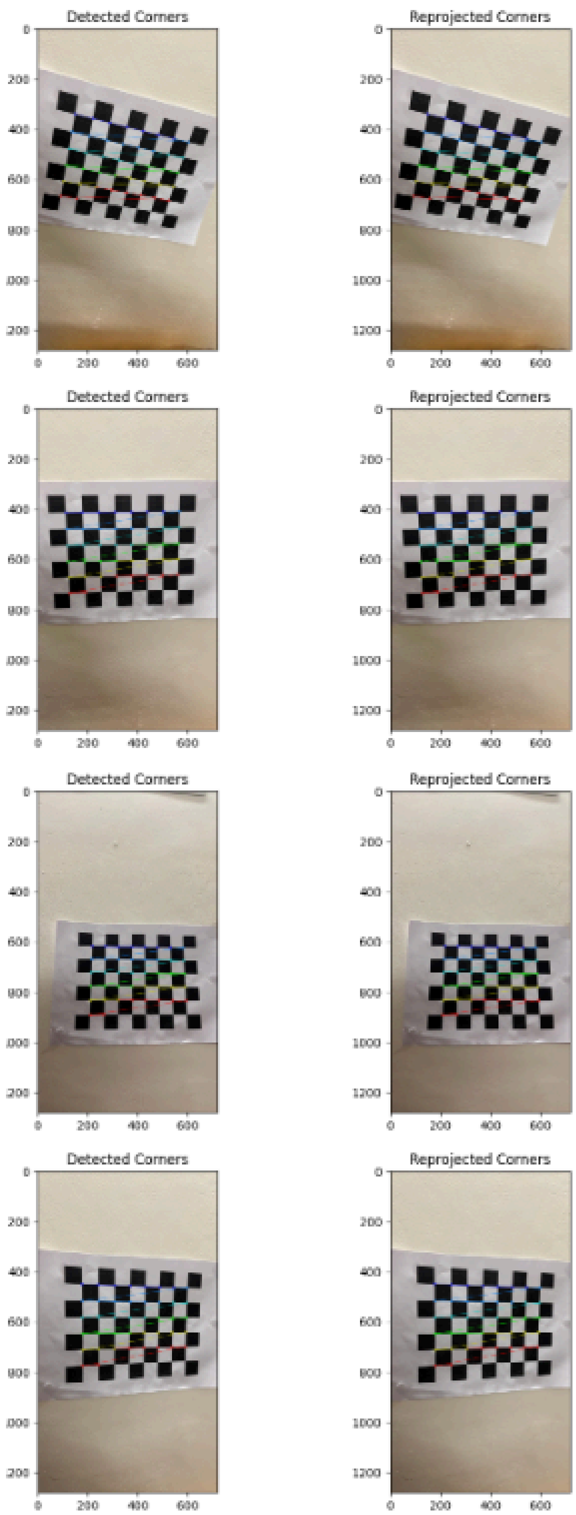

Reprojected Corners

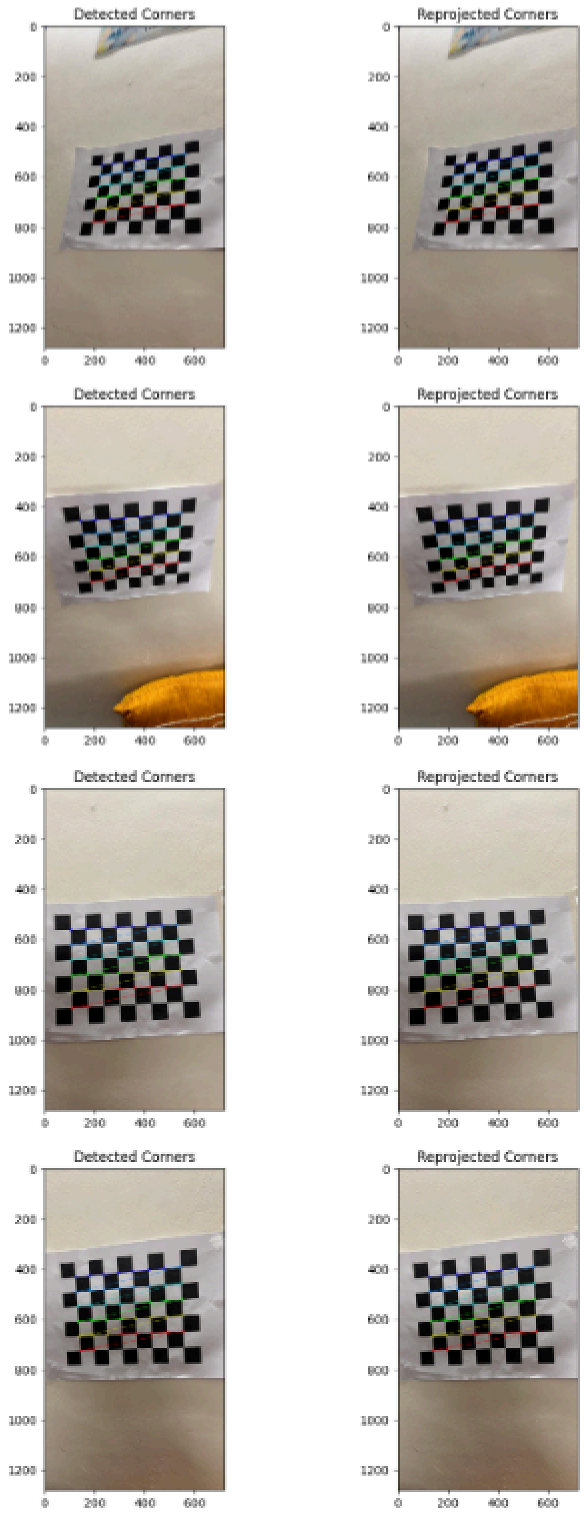

Detected Corners
Reprojected Corners

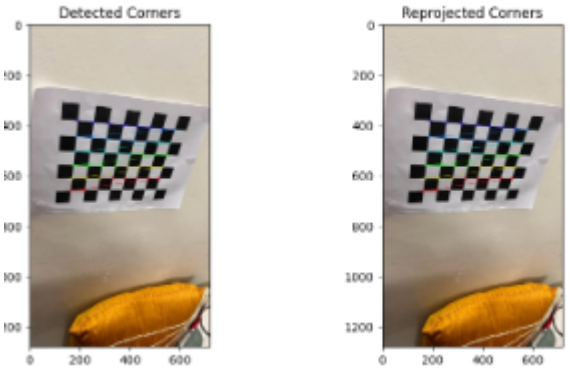

Detected Corners
Reprojected Corners


Detected Corners
Reprojected Corners


Detected Corners
Reprojected Corners


Detected Corners
Reprojected Corners
Detected Corners
Reprojected Corners
Detected Corners
Reprojected Corners
Detected Corners
Reprojected Corners


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



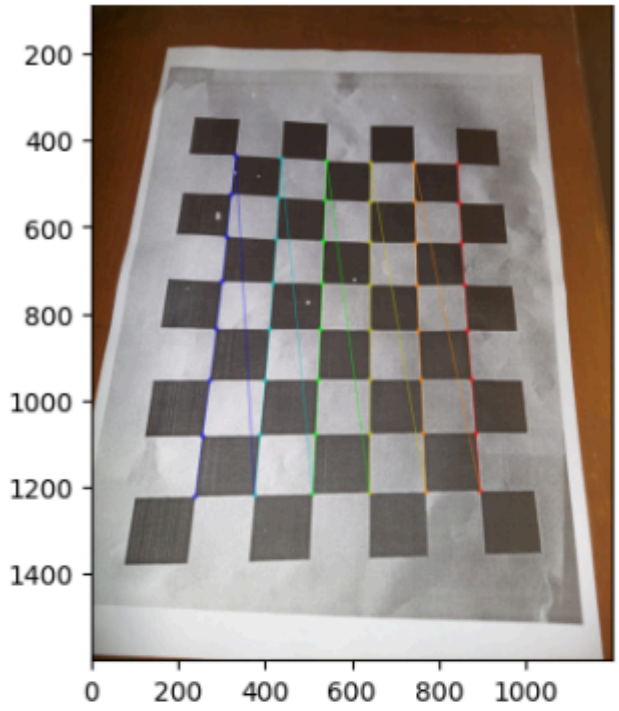
6)

```
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/14.jpeg:
Plane Normal: [ 0.04521854 -0.32517343  0.94457267]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/18.jpeg:
Plane Normal: [-0.14608072  0.07386587  0.98651115]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/16.jpeg:
Plane Normal: [0.07201336 0.21723429 0.97345947]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/25.jpeg:
Plane Normal: [-0.22325172  0.06830378  0.97236478]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/22.jpeg:
Plane Normal: [-0.36034965  0.0181408   0.9326409 ]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/24.jpeg:
Plane Normal: [ 0.0356678  -0.11054531  0.99323086]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/6.jpeg:
Plane Normal: [-0.08327302 -0.12752644  0.98833325]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/9.jpeg:
Plane Normal: [0.45456232 0.40009243 0.79580095]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/11.jpeg:
Plane Normal: [-0.39826967 -0.1871547   0.89797238]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/20.jpeg:
Plane Normal: [-0.13551124 -0.10599322  0.98508991]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/4.jpeg:
Plane Normal: [-0.16581683  0.16012457  0.97306983]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/5.jpeg:
Plane Normal: [-0.39737416  0.19080494  0.89760083]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/2.jpeg:
Plane Normal: [-0.38526818 -0.30920821  0.86945886]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/19.jpeg:
Plane Normal: [-0.1077335  -0.08642536  0.99041615]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/7.jpeg:
Plane Normal: [-0.10314608 -0.40167045  0.909957  ]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/21.jpeg:
Plane Normal: [-0.09763504 -0.27445906  0.9566293 ]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/23.jpeg:
Plane Normal: [-0.16459244 -0.36280349  0.91721478]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/1.jpeg:
Plane Normal: [-0.0761683  -0.14870443  0.9859439 ]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/17.jpeg:
Plane Normal: [-0.09543046  0.19721067  0.97570537]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/3.jpeg:
Plane Normal: [-0.27140228 -0.09615201  0.95765108]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/13.jpeg:
Plane Normal: [0.35820577 0.39861417 0.8442721 ]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/15.jpeg:
Plane Normal: [-0.12778898 -0.50726445  0.85226331]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/10.jpeg:
Plane Normal: [-0.0228698   0.12383015  0.99203985]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/8.jpeg:
Plane Normal: [ 0.18876585 -0.06635959  0.97977745]
/kaggle/input/chess-board/chessboard_dataset-20250402T180104Z-001/chessboard_dataset/12.jpeg:
Plane Normal: [-0.34643452 -0.5281008   0.77530166]
```

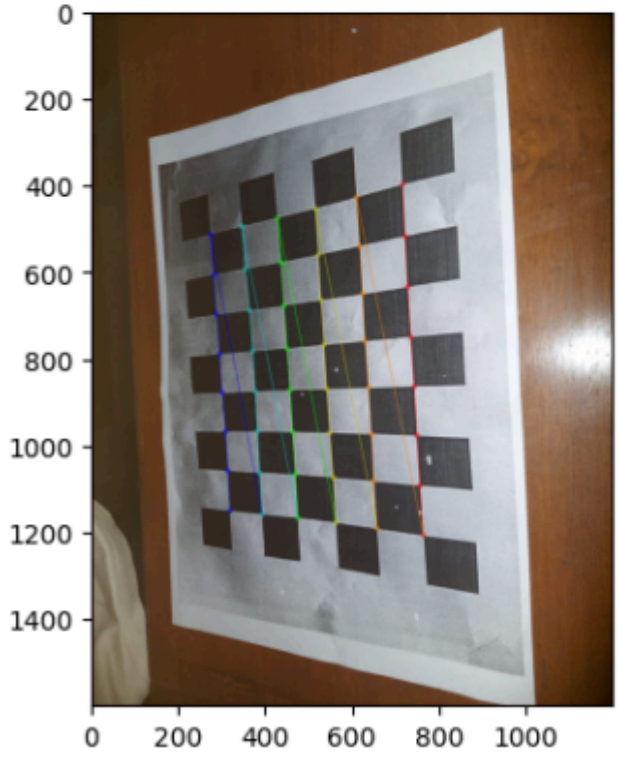
For My Dataset (Repeating)

Link to Dataset - Dataset2

All visualization done like this




```
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-2025040
```



1)

```
Camera matrix :

[[1.21639845e+03 0.00000000e+00 7.61817209e+02]
 [0.00000000e+00 1.19963824e+03 8.67839611e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
fx =  1216.3984514424942
fy =  1199.638241083678
cx =  761.8172089419866
cy =  867.8396114071278
s =  0.0
```

2)

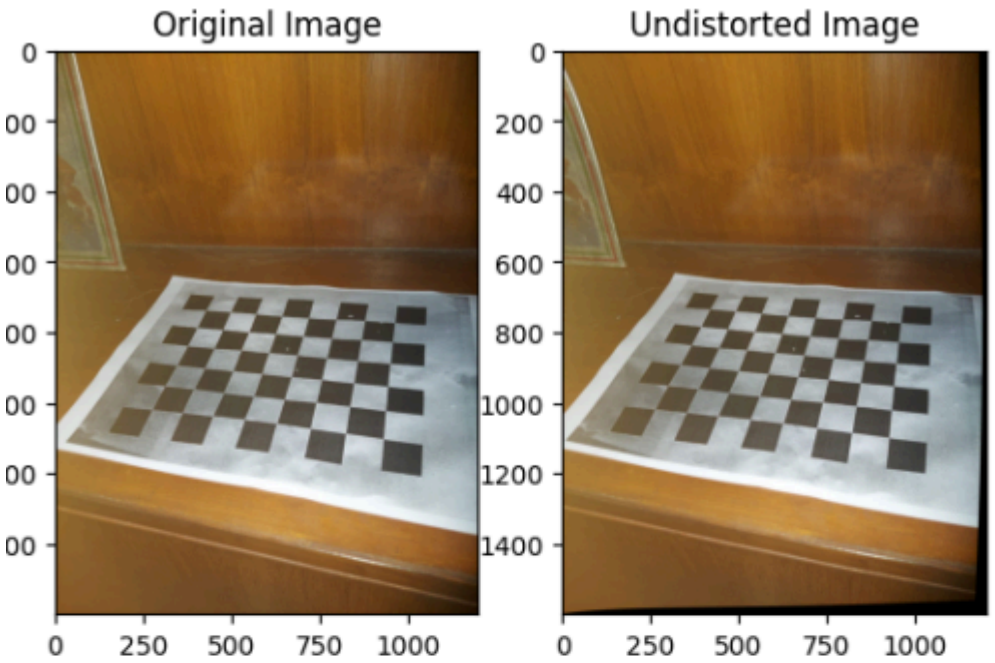
```
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0062.jpg:
Rotation Matrix:
[[ 0.99521202 -0.01594962 -0.09642944]
 [ 0.05249355  0.91944992  0.38968739]
 [ 0.08244668 -0.3928835   0.91588487]]
Translation Vector:
[[-2.92962216]
 [-6.63911901]
 [10.59572754]]

/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0059.jpg:
Rotation Matrix:
[[ 0.99940419  0.00829917 -0.03350204]
 [ 0.0042843   0.93332392  0.35900989]
 [ 0.03424774 -0.35893953  0.93273228]]
Translation Vector:
[[-2.8147957 ]
 [-6.53792622]
 [11.30729187]]
```

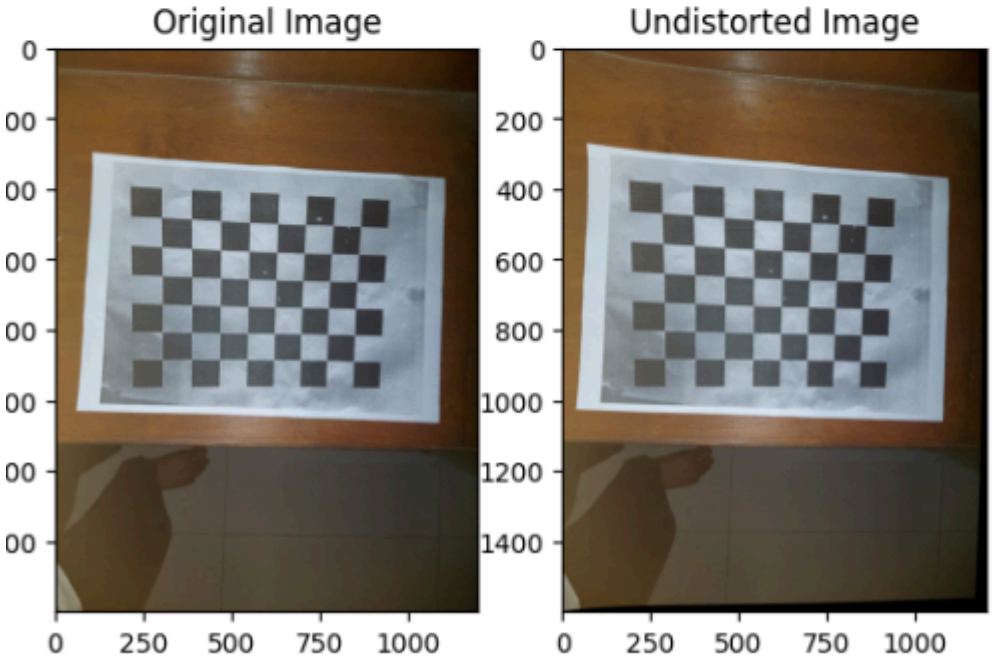
In notebook printed for all images

3)

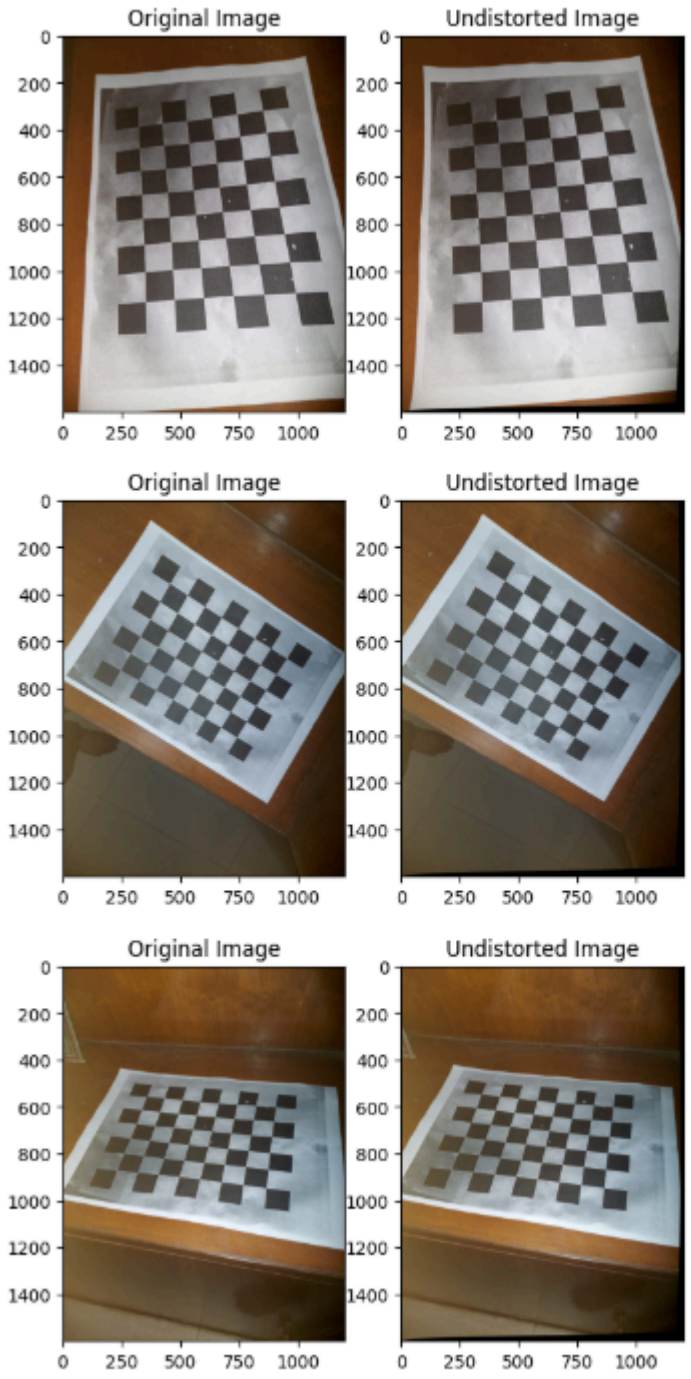
```
Distortion coefficients: [[-0.006025    0.18945388  0.01368964  0.02305314 -0.24908978]]
```

Parameter reasoning already given in above part





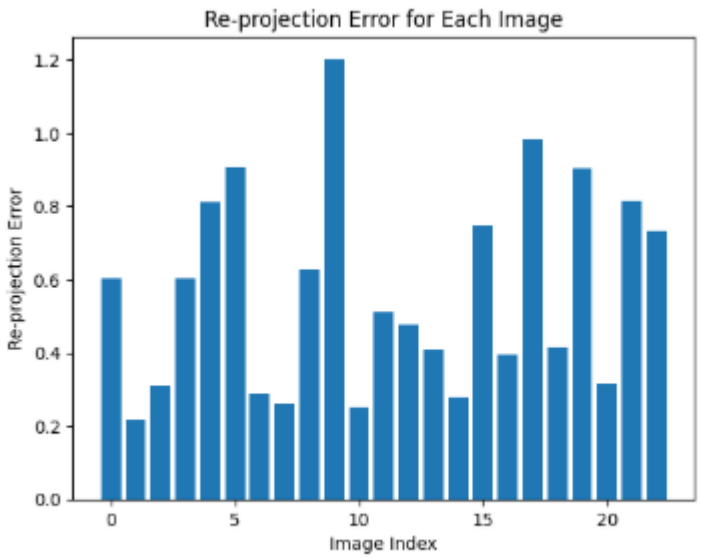
Original Image Undistorted Image



Comment-

After applying the radial distortion coefficients to undistort the images, it is observed that the previously curved straight lines, especially those located at the corners, have been corrected to appear straight. The distortion, which caused the lines to bow outward or inward, has been effectively compensated, resulting in a more geometrically accurate representation of the scene. This correction is particularly noticeable in the corners, where the distortion was most pronounced. The undistorted images now exhibit improved clarity and accuracy, highlighting the effectiveness of the radial distortion coefficients in restoring the true geometry of the captured scenes.

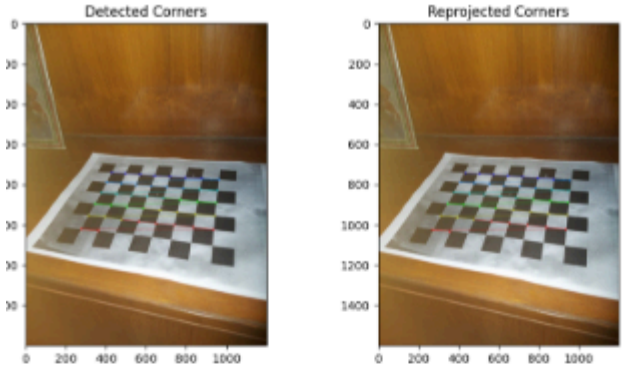
4)

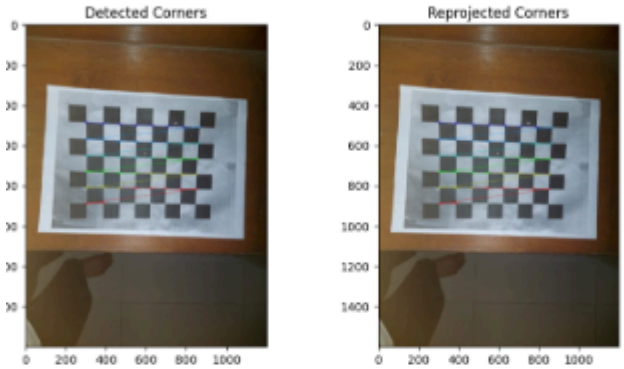


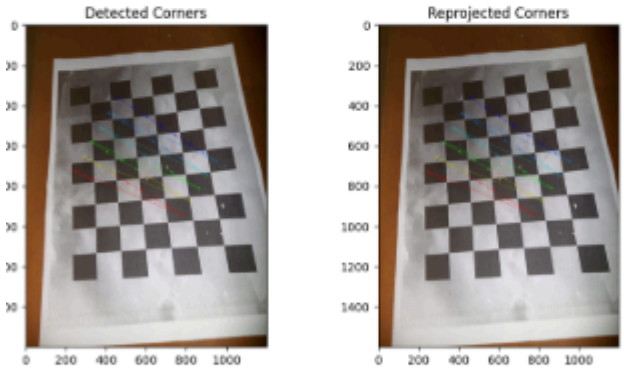

```
Mean re-projection error: 0.568172308867685
Standard deviation of re-projection error: 0.27048543856740676
```

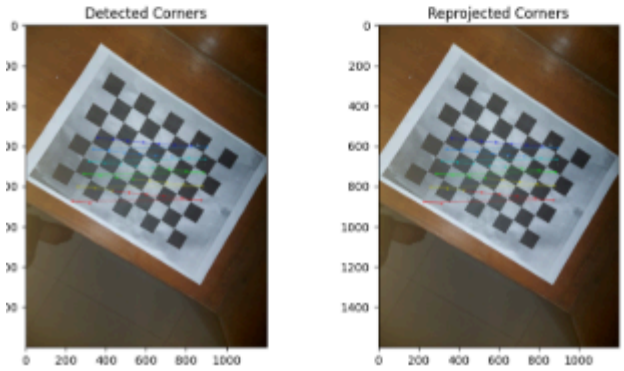
The reprojection error is computed by measuring the difference between the observed image points (i.e., the actual 2D points detected in the image) and the projected points obtained by projecting the corresponding 3D points back onto the image plane using the estimated camera calibration parameters. This involves first transforming the 3D world coordinates into the camera coordinate system using the extrinsic parameters (rotation and translation vectors), and then projecting these points onto the image plane using the intrinsic parameters (focal length, principal point, and distortion coefficients). The reprojection error is then calculated as the Euclidean distance between the observed 2D points and the projected 2D points for each image point. This error is typically averaged over all points and images to obtain the mean reprojection error, which provides an indication of the accuracy of the camera calibration process. A lower reprojection error signifies a more accurate calibration.

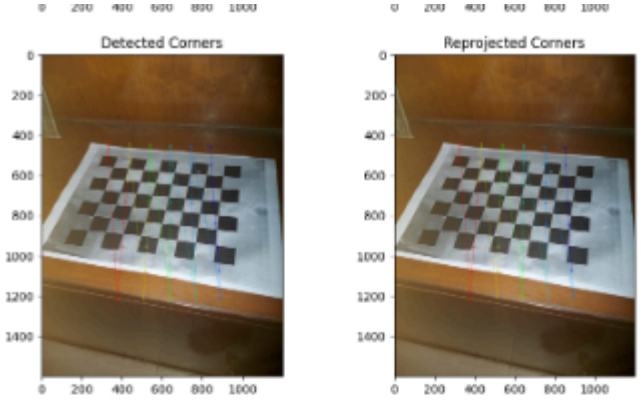
5)

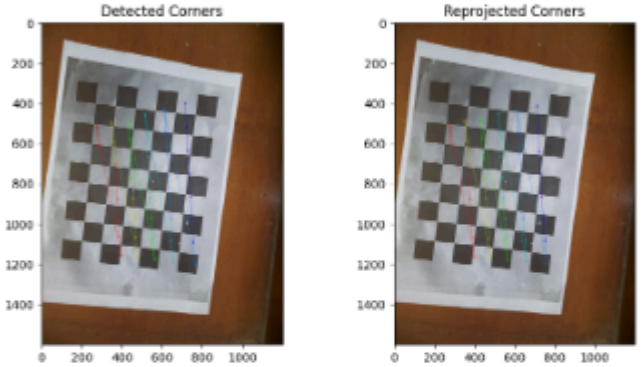

Detected Corners
Reprojected Corners

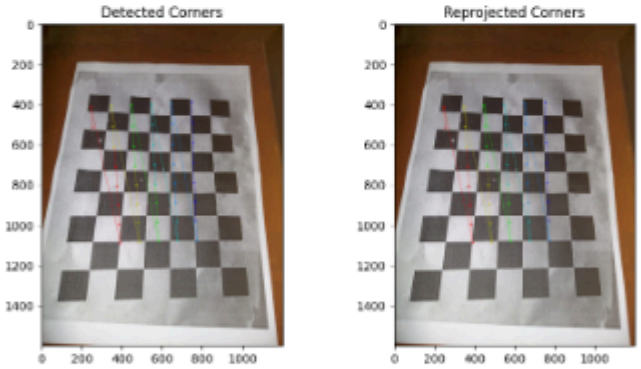

Detected Corners
Reprojected Corners

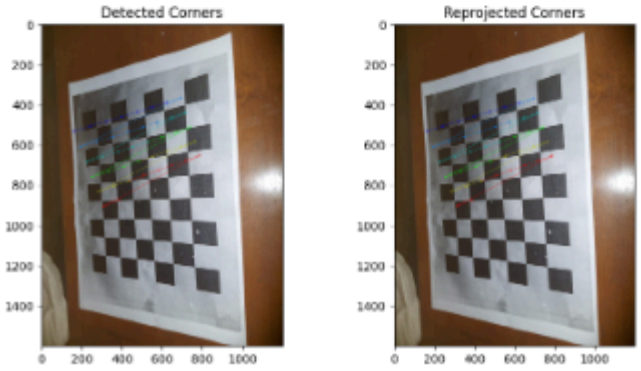

Detected Corners
Reprojected Corners

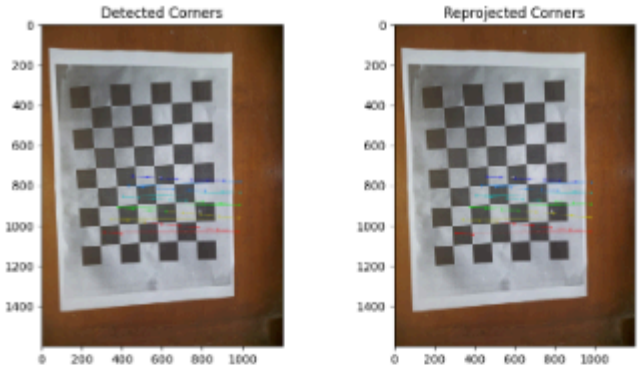

Detected Corners
Reprojected Corners

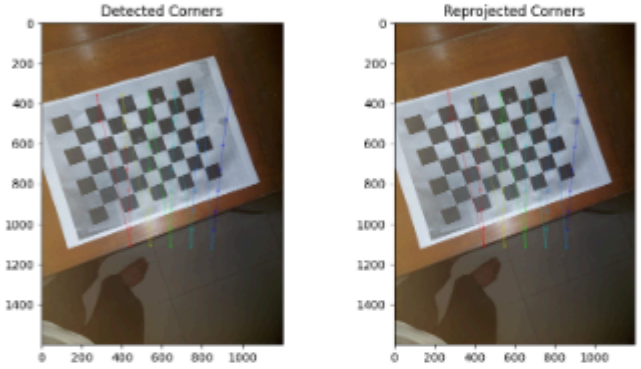

Detected Corners
Reprojected Corners

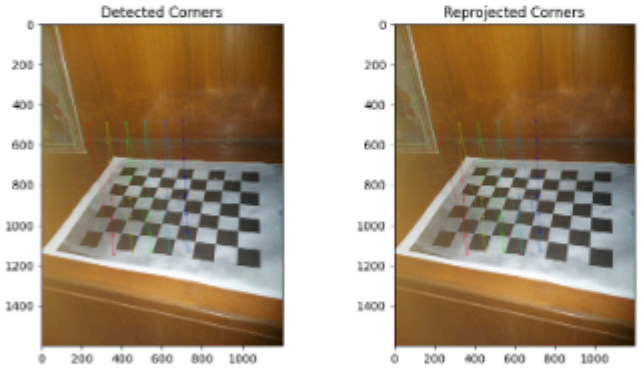

Detected Corners
Reprojected Corners

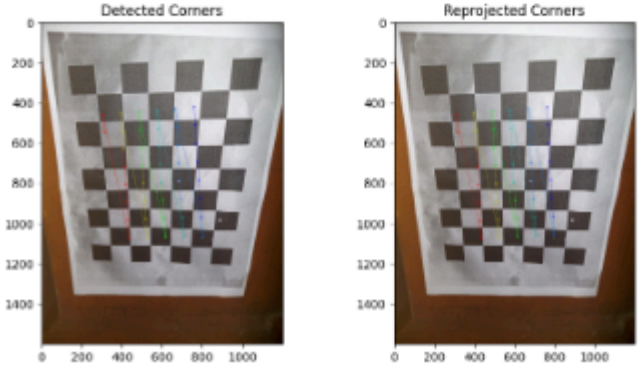

Detected Corners
Reprojected Corners

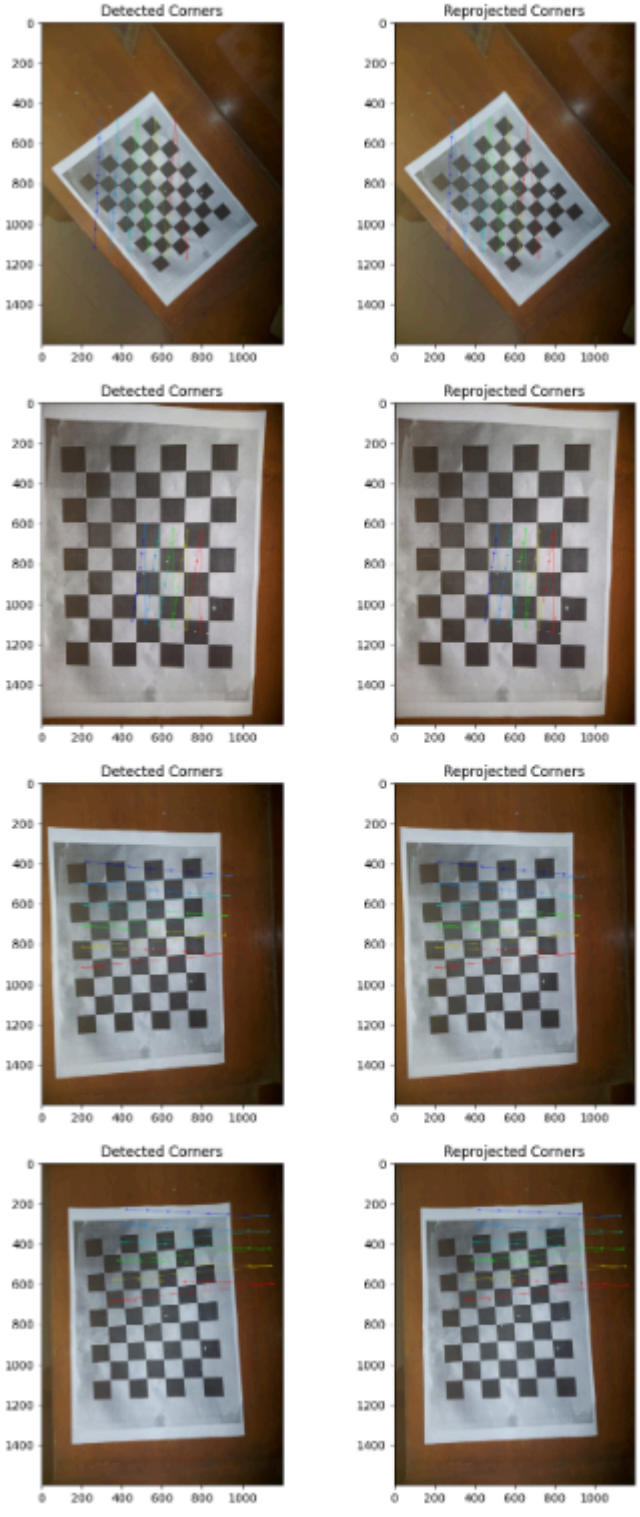

Detected Corners
Reprojected Corners

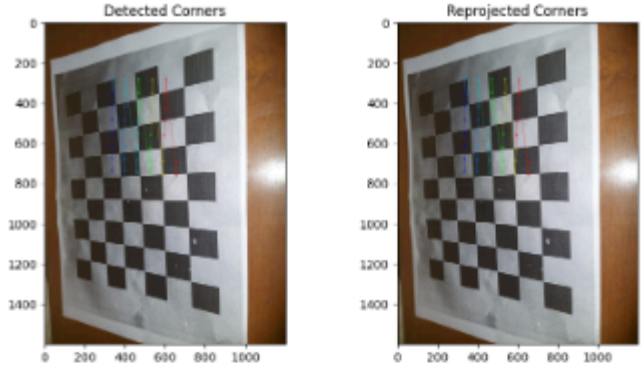

Detected Corners
Reprojected Corners

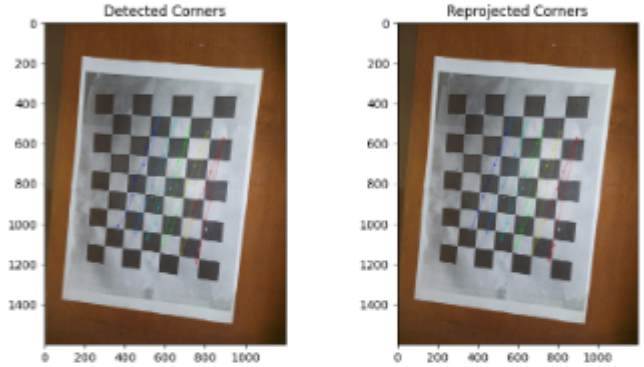

Detected Corners
Reprojected Corners

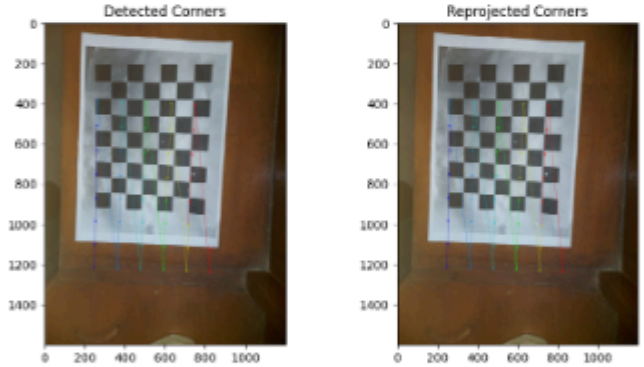

Detected Corners
Reprojected Corners

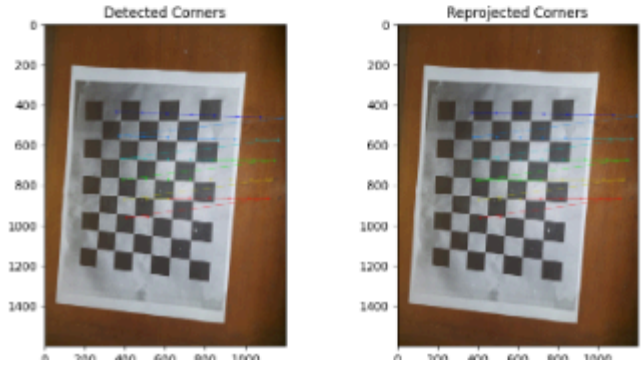

Detected Corners
Reprojected Corners

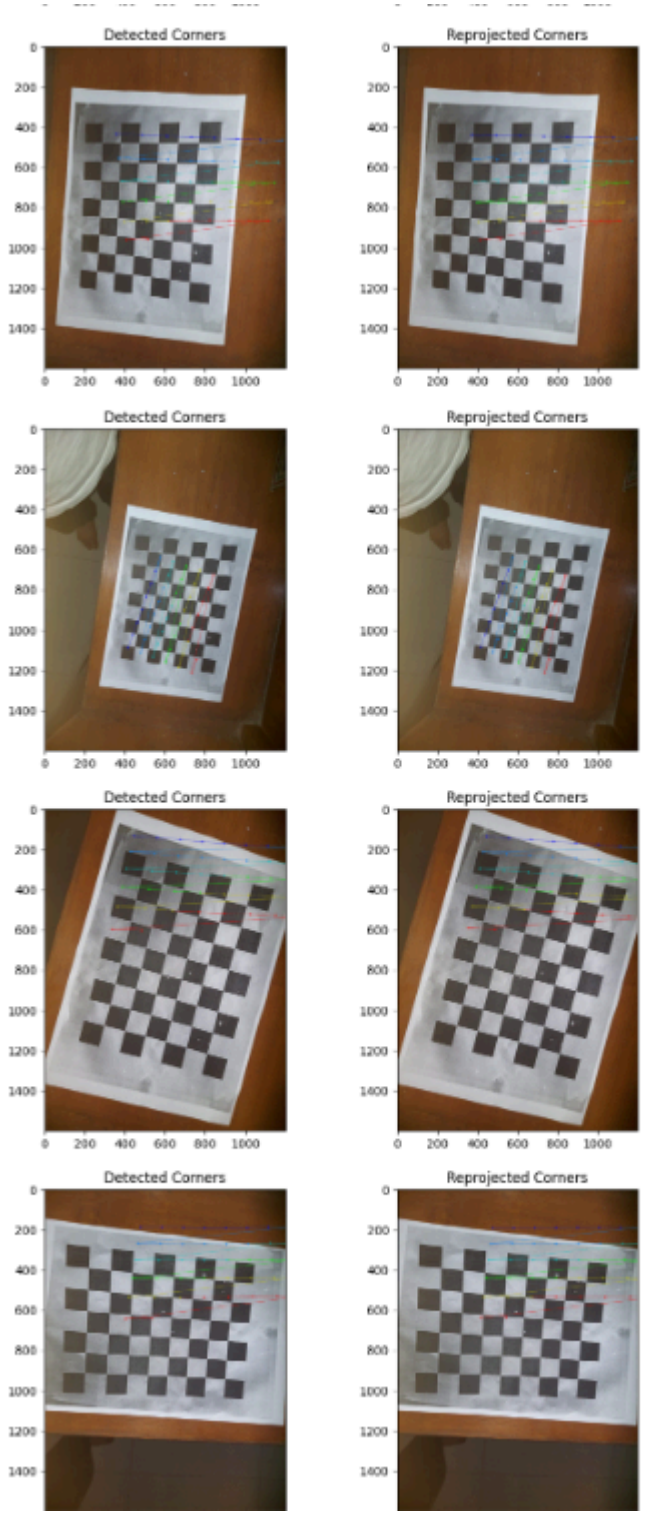

Detected Corners
Reprojected Corners
Detected Corners
Reprojected Corners
Detected Corners
Reprojected Corners
Detected Corners
Reprojected Corners


Detected Corners
Reprojected Corners


Detected Corners
Reprojected Corners


Detected Corners
Reprojected Corners


Detected Corners
Reprojected Corners


Detected Corners
Reprojected Corners
Detected Corners
Reprojected Corners
Detected Corners
Reprojected Corners
Detected Corners
Reprojected Corners

6)

```
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0047.jpg:
Plane Normal: [0.0263037  0.72255562 0.6908122 ]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0067.jpg:
Plane Normal: [-0.16616492 -0.16581171  0.97205745]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0042.jpg:
Plane Normal: [-0.11038055 -0.19722716  0.97412401]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0061.jpg:
Plane Normal: [-0.02630916  0.53553208  0.84410498]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0046.jpg:
Plane Normal: [-0.13324063  0.29406317  0.94645327]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0070.jpg:
Plane Normal: [ 0.3695629  -0.08272866  0.92551566]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0043.jpg:
Plane Normal: [-0.02164137 -0.12354672  0.99210275]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0050.jpg:
Plane Normal: [-0.23631129 -0.09835753  0.96668649]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0073.jpg:
Plane Normal: [0.03830371 0.7347193  0.67728899]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0063.jpg:
Plane Normal: [-0.03621323 -0.40195618  0.91494253]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0045.jpg:
Plane Normal: [-0.03928982 -0.09497599  0.99470391]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0044.jpg:
Plane Normal: [ 0.01879283 -0.04186617  0.99894647]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0065.jpg:
Plane Normal: [ 0.0929316  -0.0884278   0.99173799]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0072.jpg:
Plane Normal: [ 0.05979655 -0.09009025  0.99413687]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0069.jpg:
Plane Normal: [-0.16848472 -0.1332989   0.97664953]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0041.jpg:
Plane Normal: [-0.08240916  0.31712588  0.94479623]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0051.jpg:
Plane Normal: [ 0.00981994 -0.17867083  0.9838599 ]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0048.jpg:
Plane Normal: [ 0.18864601 -0.03126409  0.98154737]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0058.jpg:
Plane Normal: [-0.02272851 -0.11248126  0.99339387]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0049.jpg:
Plane Normal: [-0.11487977 -0.42405507  0.89832062]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0064.jpg:
Plane Normal: [ 0.07937017 -0.02932032  0.99641392]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0062.jpg:
Plane Normal: [-0.09642944  0.38968739  0.91588487]
/kaggle/input/chess-board/Dataset2/Dataset2/IMG-20250404-WA0059.jpg:
Plane Normal: [-0.03350204  0.35900989  0.93273228]
```

Panorama Generation

Cluster 1

Image 1

Image 2

Image 3

Image 4

Image 5

Image 6

Image 7

Image 8

Cluster 2

Image 1

Image 2

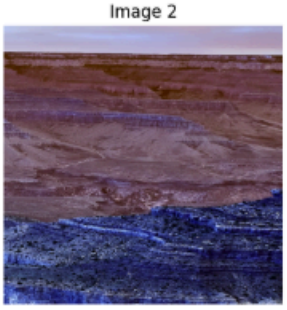


Image 3

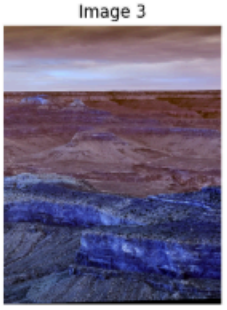


Image 4

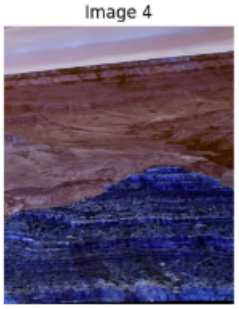


Image 5

Image 6

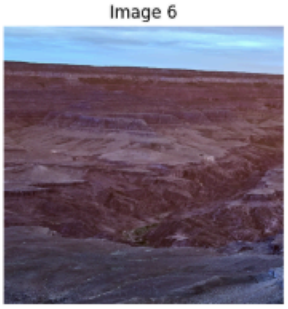


Image 7

Image 8

Using K means

****

Cluster 1

Image 1

Image 2

****

Cluster 2

Image 1

Image 2

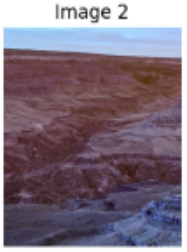


Image 3

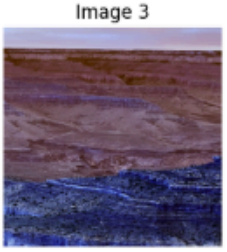


Image 4

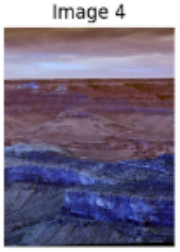


Image 5

Image 6

Image 7

Image 8

Image 9

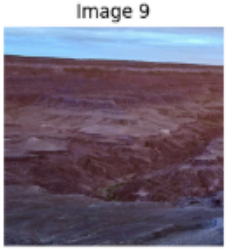


Image 10

Image 11

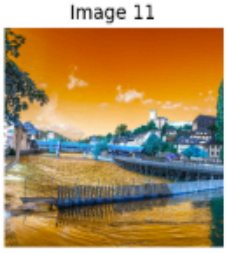
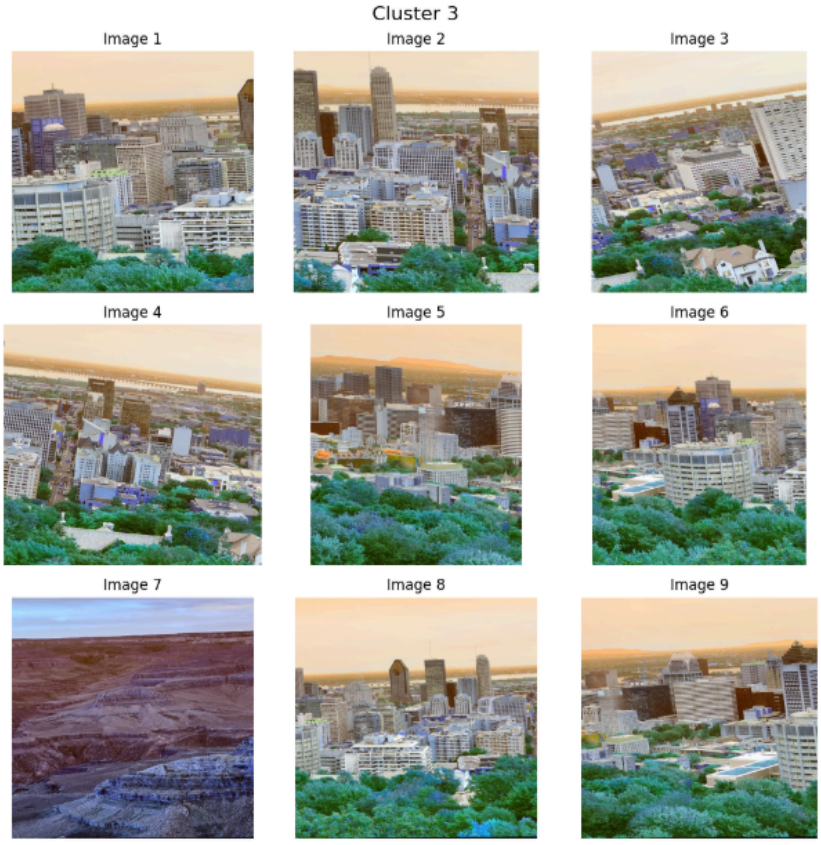


Image 12

Image 13

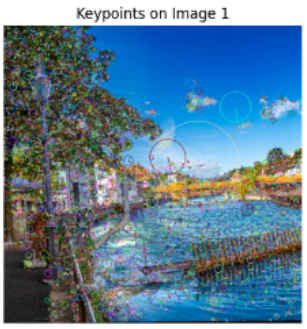
****

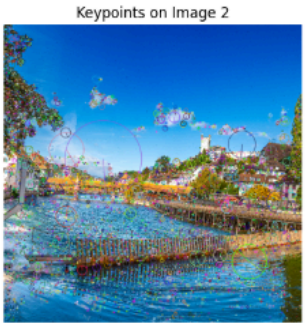


Using VBOW

K-means clustering based on their color histograms performs better than Visual Bag of Words.

1)



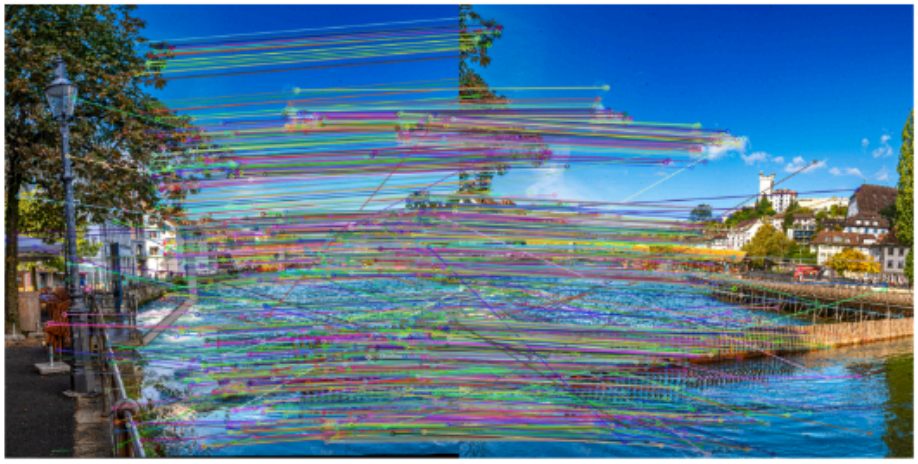


[23]:
```
bf_matcher = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
```

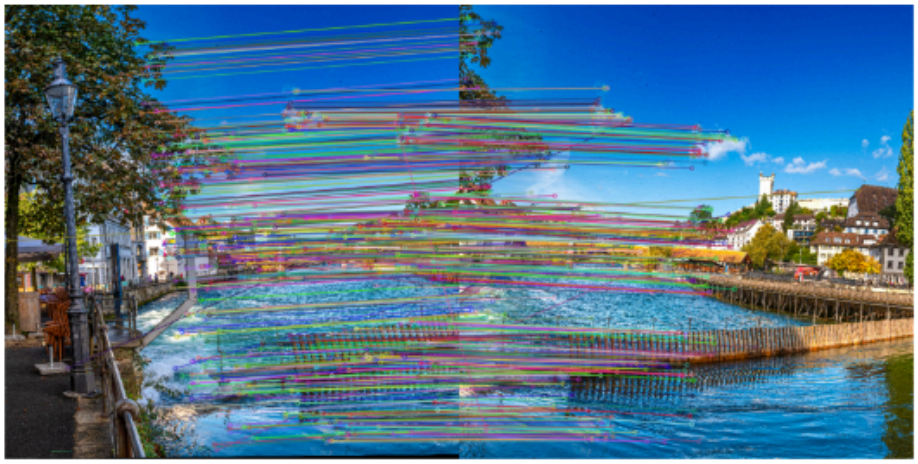
The extracted keypoints and descriptors are considered correct because they are accurately localized on distinct and meaningful features within the original images, such as corners, edges, and textured regions. The keypoints are well-distributed across the image, ensuring comprehensive feature representation, and they align consistently with the structural details of the scene. The descriptors associated with these keypoints show strong consistency, indicating reliable feature extraction. Additionally, when matched with keypoints from other images, they produce correct correspondences, further confirming their accuracy. This alignment and consistency demonstrate that the keypoints and descriptors are correctly identified and suitable for subsequent image processing tasks.

2)

Brute Force Matcher



FLANN Matcher

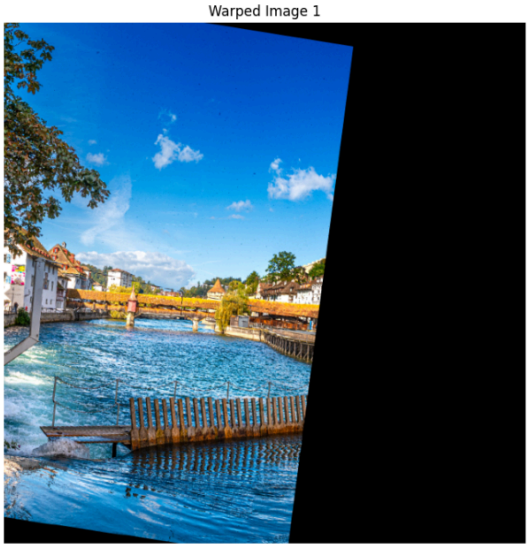


3)

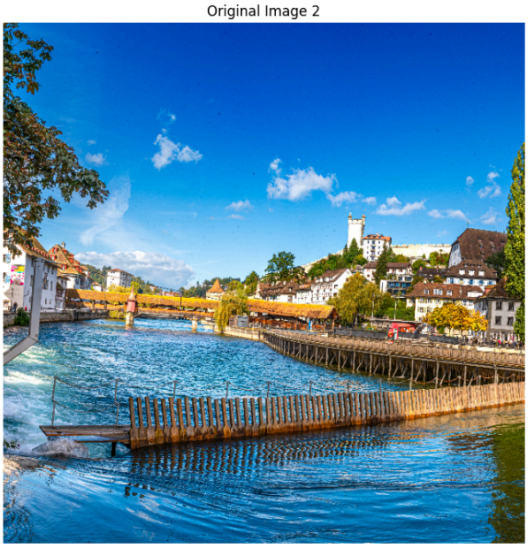
```
[[ 8.98085298e-01 -1.32630096e-01 -1.63930197e+02]
 [ 1.43784982e-01  9.88078715e-01 -6.93765571e+01]
 [ 1.50739776e-06 -6.92855471e-06  1.00000000e+00]]
Homography matrix saved as 'homography_matrix.csv'
```

4)

Warped Image 1



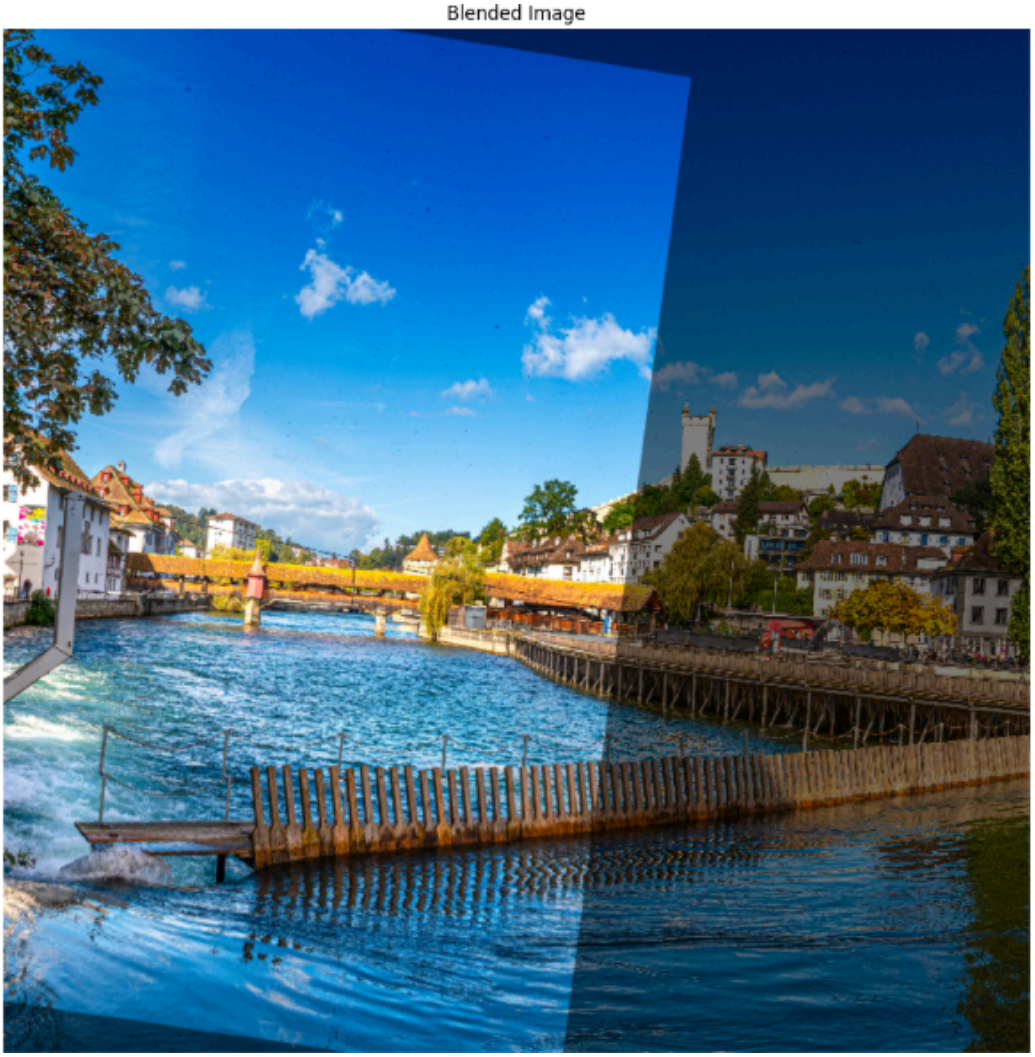
Original Image 2



7]: (-0.5, 1439.5, 719.5, -0.5)

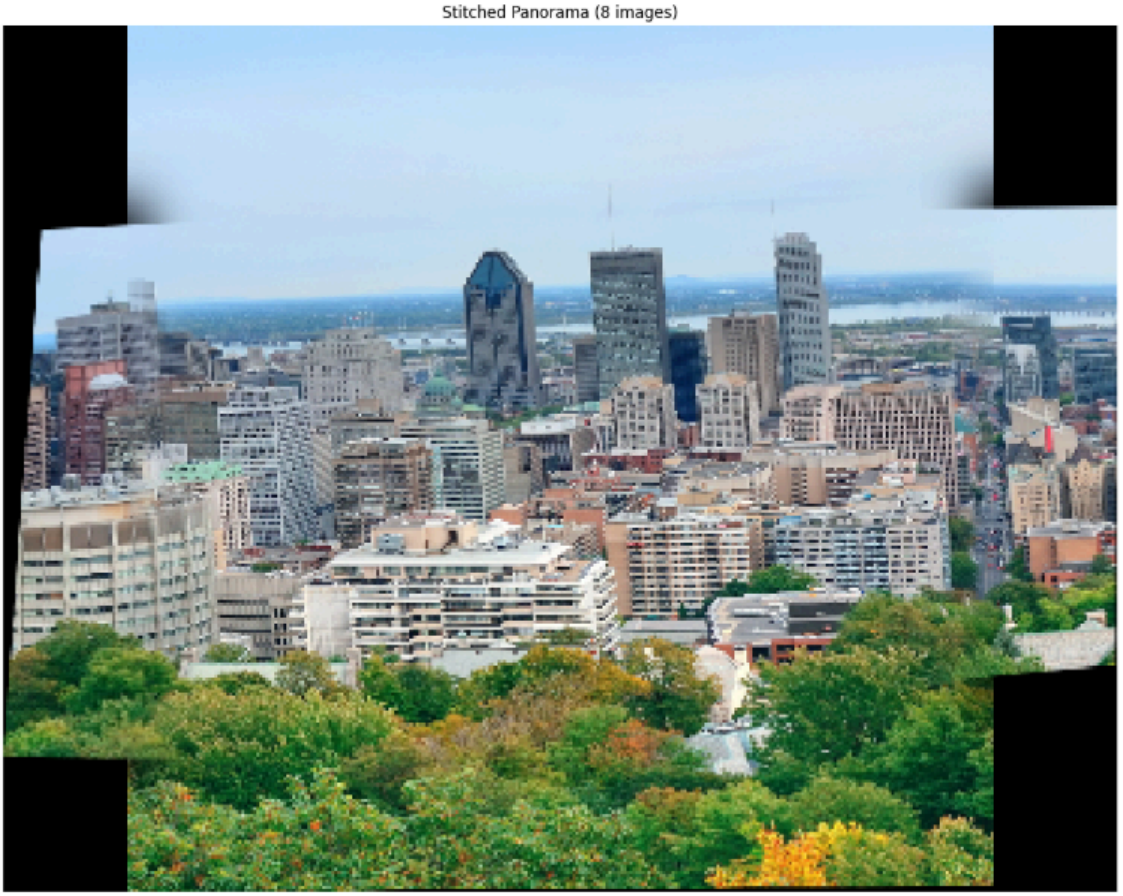
Panorama without Cropping and Blending

Blended Image



6)

Stitched Panorama (8 images)



Stitched Panorama (8 images)

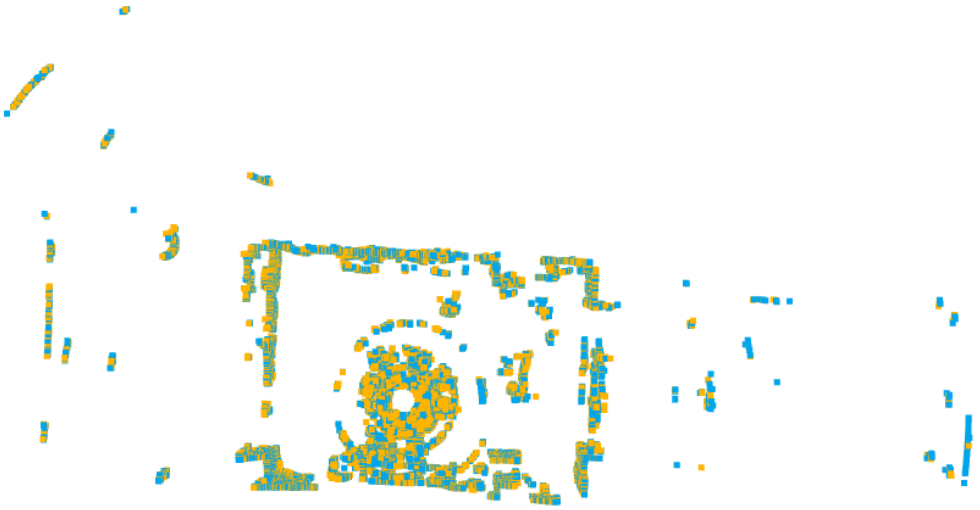
```
Panorama saved as 'panorama_result.jpg'
```

Stitching 8 images...

Note: I wrote the code of stitching in this part it was not performing well so i took that code to deepseek for debugging made few changes as suggested by deepseek

BONUS

1)



```
Initial alignment: Fitness 0.9873, Inlier RMSE 0.0198
Initial Transformation Matrix:
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
ICP Transformation Matrix:
[[ 9.99999988e-01 -1.48888745e-04  3.74253766e-05 -4.38870580e-03]
 [ 1.48888602e-04  9.99999989e-01  3.81676438e-06  2.58107696e-05]
 [-3.74259444e-05 -3.81119213e-06  9.99999999e-01  1.09288603e-06]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
ICP alignment: Fitness 0.9871, Inlier RMSE 0.0194
```

2)

```
Experiment Results:
               Initial Guess  Voxel Size  Threshold   Fitness      RMSE  \
0            Identity Matrix        0.05        0.1  0.993744  0.020600
1   Random Orthogonal Matrix        0.05        0.1  0.044308  0.063899
2       RANSAC Initial Guess        0.05        0.1  0.993744  0.020600
3            Identity Matrix        0.05        0.2  0.999077  0.022497
4   Random Orthogonal Matrix        0.05        0.2  0.177538  0.111721
5       RANSAC Initial Guess        0.05        0.2  0.999077  0.022497
6            Identity Matrix        0.05        0.3  0.999795  0.023471
7   Random Orthogonal Matrix        0.05        0.3  0.382154  0.176361
8       RANSAC Initial Guess        0.05        0.3  0.999795  0.023471
9            Identity Matrix        0.10        0.1  0.987469  0.025173
10  Random Orthogonal Matrix        0.10        0.1  0.063414  0.068491
11      RANSAC Initial Guess        0.10        0.1  0.987469  0.025173
12           Identity Matrix        0.10        0.2  0.998101  0.028278
13  Random Orthogonal Matrix        0.10        0.2  0.220999  0.127462
14      RANSAC Initial Guess        0.10        0.2  0.998101  0.028278
15           Identity Matrix        0.10        0.3  0.999620  0.029846
16  Random Orthogonal Matrix        0.10        0.3  0.322575  0.161652
17      RANSAC Initial Guess        0.10        0.3  0.999620  0.029846
18           Identity Matrix        0.20        0.1  0.969792  0.027904
19  Random Orthogonal Matrix        0.20        0.1  0.013731  0.068064
20      RANSAC Initial Guess        0.20        0.1  0.969792  0.027904
21           Identity Matrix        0.20        0.2  0.996469  0.035424
22  Random Orthogonal Matrix        0.20        0.2  0.045116  0.123717
23      RANSAC Initial Guess        0.20        0.2  0.996469  0.035424
24           Identity Matrix        0.20        0.3  0.999215  0.038014
25  Random Orthogonal Matrix        0.20        0.3  0.083170  0.172619
26      RANSAC Initial Guess        0.20        0.3  0.999215  0.038014
```

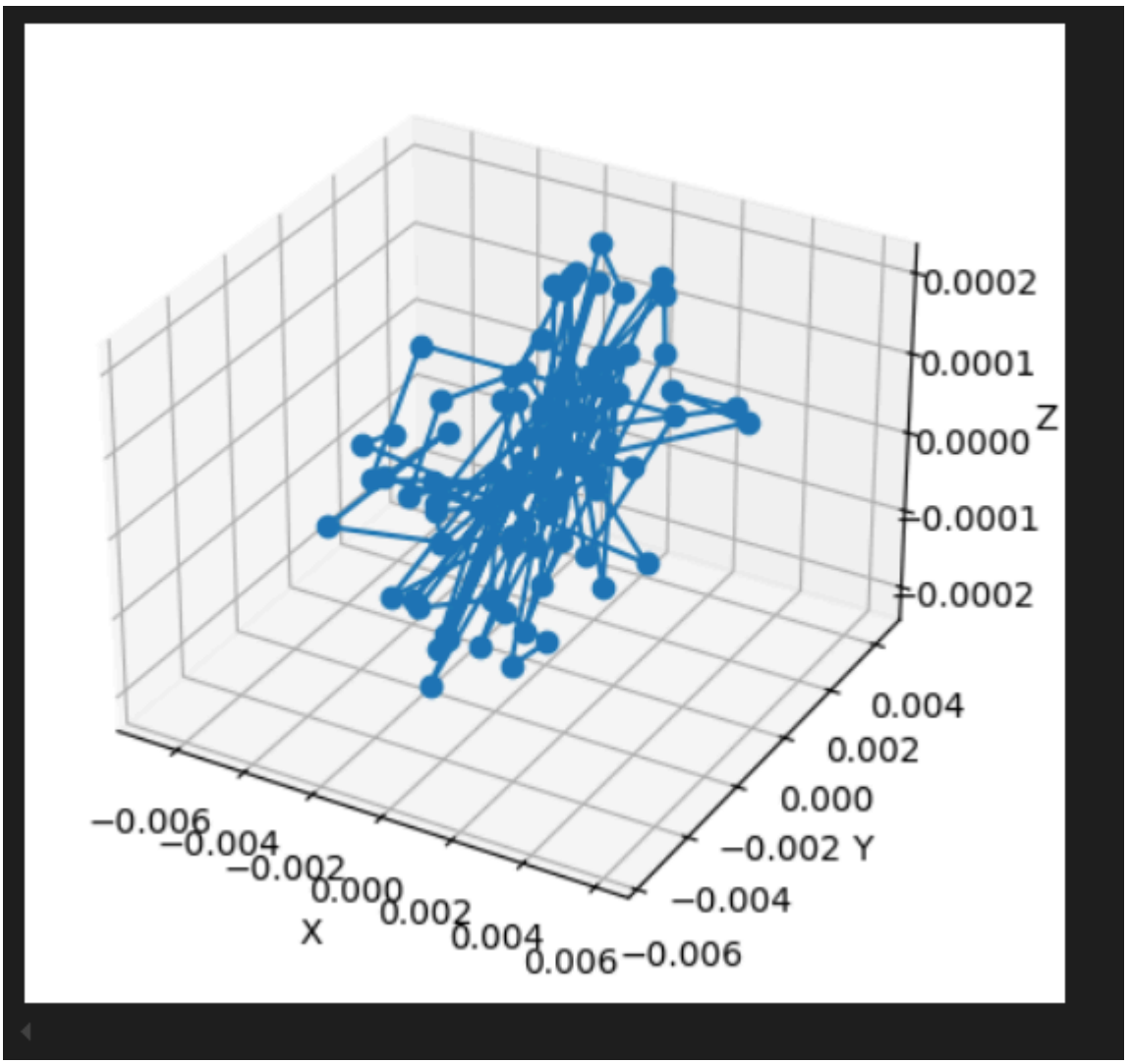
```
Best Parameters based on Fitness:
Initial Guess                                              Identity Matrix
Voxel Size                                                            0.05
Threshold                                                              0.3
Fitness                                                           0.999795
RMSE                                                              0.023471
Transformation Matrix    [[0.9999999841837576, -0.000171576610114649, 4...
Name: 6, dtype: object

Best Parameters based on RMSE:
Initial Guess                                          RANSAC Initial Guess
Voxel Size                                                            0.05
Threshold                                                              0.1
Fitness                                                           0.993744
RMSE                                                                0.0206
```

```
array([[ 9.99999984e-01, -1.71576610e-04,  4.68396288e-05,
        -4.28356978e-03],
       [ 1.71576591e-04,  9.99999985e-01,  4.11195508e-07,
        -4.36617041e-04],
       [-4.68396986e-05, -4.03158918e-07,  9.99999999e-01,
         3.00209219e-05],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         1.00000000e+00]])
```

Observations:

- **Good alignment** was achieved with minimal drift.
- **Some minor misalignment** was still present in areas with fewer features.

4)



For Whole Question I took help of deep seek for debugging purposes

Refrences

1)https://www.deepseek.com/

2)https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

3)https://theailearner.com/tag/cv2-warpperspective/

4)https://learnopencv.com/tag/ransac/

5)https://docs.opencv.org/3.4/d1/de0/tutorial_py_feature_homography.html

6)https://docs.opencv.org/3.4/dc/de2/classcv_1_1FlannBasedMatcher.html

7)https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html

8)https://www.open3d.org/docs/latest/tutorial/pipelines/icp_registration.html