Ganeev Singh
2021389

Coding Questions

Q4:

Intrinsic Matrix:

```
[[887.63361768   0.         583.72520646]
 [  0.         792.17411885 390.42828058]
 [  0.           0.           1.        ]]
```

Fx: focal length in x axis = 887.633 mm
Fy: focal length in y axis = 792.174 mm
S: skew-parameter= 0
Ux: principal point x: 583.725
Uy: principal point y: 390.428

Distortion Matrix:

```
array([[ 1.56485640e-01, -1.37284861e+00,  1.48242323e-03,
        -4.71475474e-03,  3.24383382e+00]])
```

Radial distortion coefficients:
K1 = 0.156
K2 = -1.37
K3 = 0.0148

Tangential distortion coefficients:
P1 = -0.0047
P2 = 3.24

Extrinsic Parameters for different Images is reported in the colab file
Extrinsic matrix includes [R|t] for each image
It is a 3*4 matrix

Example:

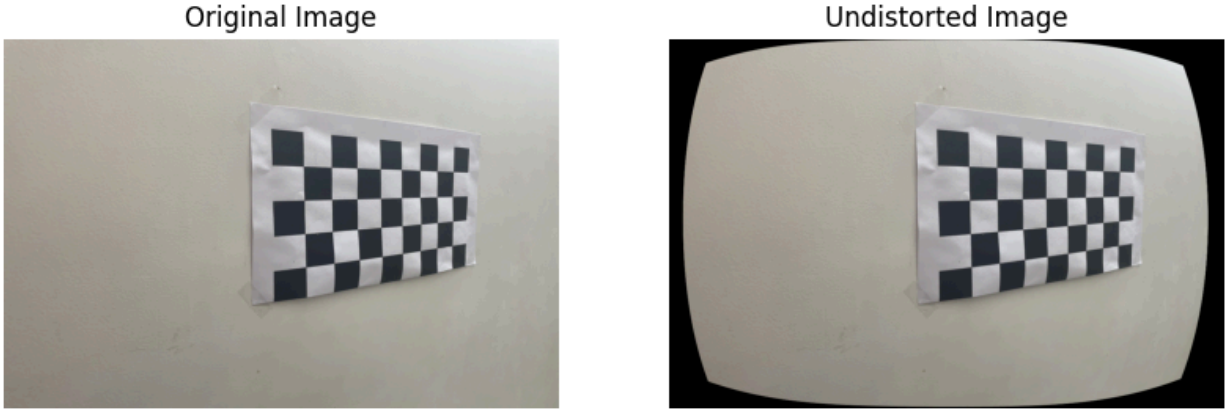
Rotation matrix (R) for image 1:
[[0.00695805 -0.99989627 0.01261055]
 [0.99810015 0.00617241 -0.06130247]
 [0.06121827 0.01301314 0.99803957]]

Translation vector (T) for image 1:
[[3.3067295]
 [-0.05750608]
 [12.23051961]]

Extrinsic parameters for image 1:
[[6.95804610e-03 -9.99896274e-01 1.26105478e-02 3.30672950e+00]
 [9.98100150e-01 6.17240935e-03 -6.13024708e-02 -5.75060774e-02]
 [6.12182747e-02 1.30131350e-02 9.98039569e-01 1.22305196e+01]]

Distortion

Can find the undistorted images in the colab file
One example:



Following the correction of distortion, the image exhibits a pronounced bulge, necessitating cropping to emphasize the checkerboard. This area remains undistorted and exhibits straight lines, unlike the original images where perspective projection and extreme angles caused distortion. The plane of the board appears distinctly rectangular, with lines that are more parallel than in the original images.

Reprojection ERROR:

The re-projection error is computed by first projecting the 3D object points onto the image plane using the estimated camera parameters, which include the camera matrix, distortion coefficients, rotation vector, and translation vector. This process replicates the imaging procedure, generating the expected image coordinates of the object points based on the estimated camera parameters.

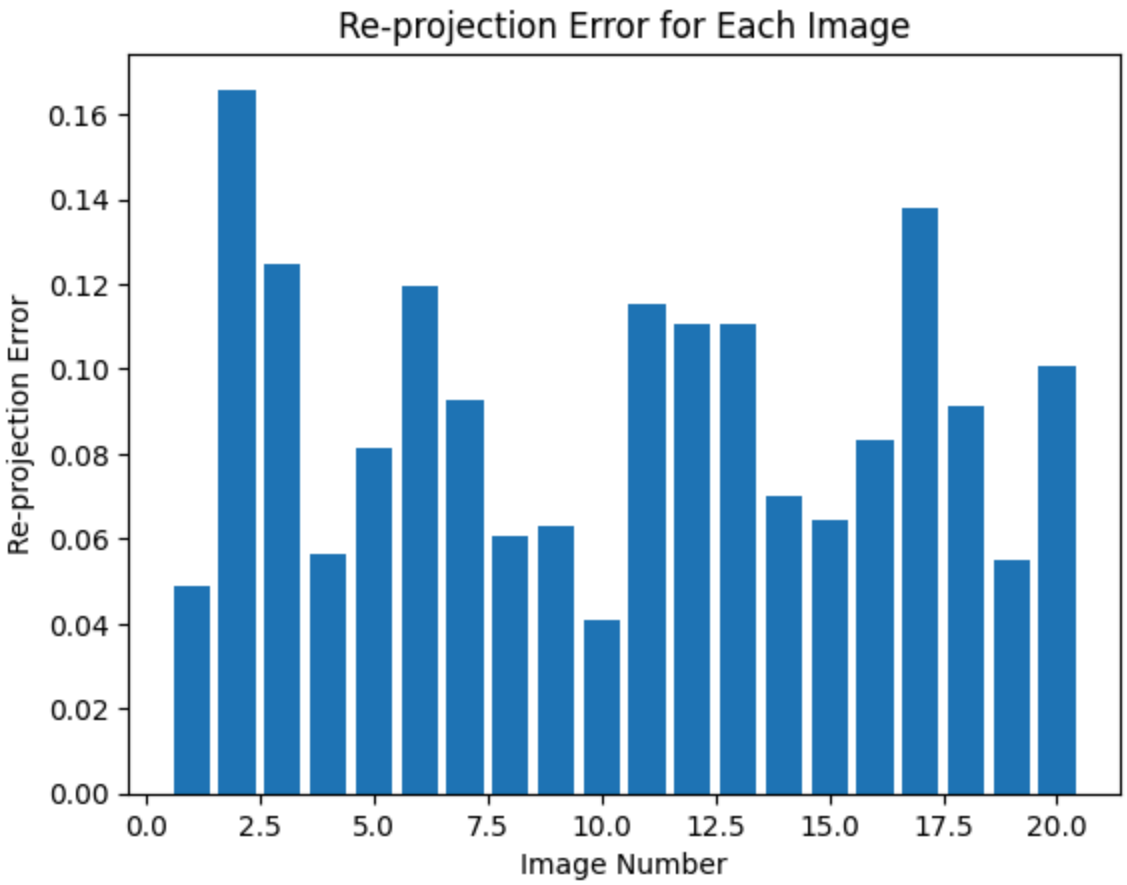
The re-projected image coordinates are then compared with the actual detected image coordinates of the object points. The re-projection error is calculated as the L2 norm (Euclidean distance) between the re-projected image coordinates and the actual detected image coordinates, normalized by the number of object points.

A lower re-projection error indicates that the estimated camera parameters effectively model the imaging process, accurately projecting the 3D object points onto the image plane with minimal deviation from the observed image coordinates. On the other hand, a higher re-projection error suggests that the estimated camera parameters may not accurately represent the true camera characteristics or that other factors, such as noise or inaccuracies in the detected image coordinates, contribute to the error.

Results for re-projection error:

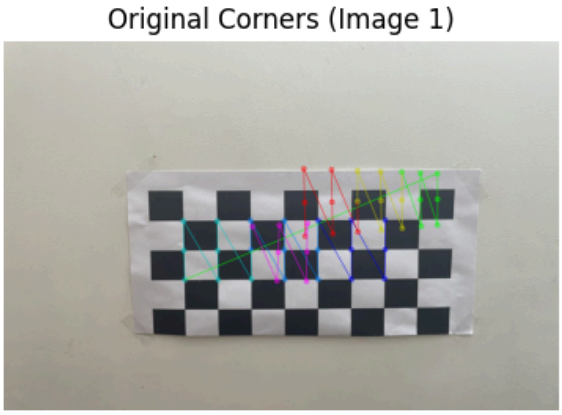
Mean re-projection error: 0.089
Standard deviation of reprojection error: 0.032

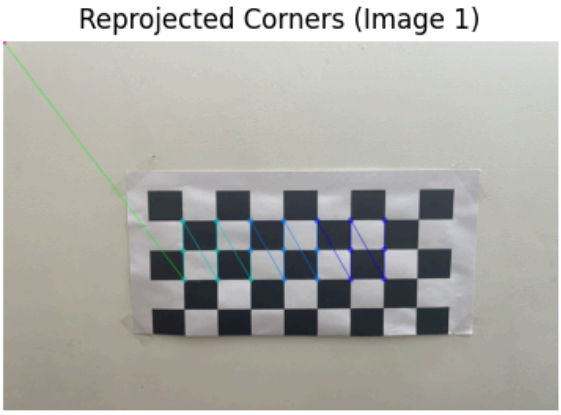
Bar plot:



Corner Detection:

These images provide a visual representation of the re-projection process, allowing for a comparison between the actual detected corners and their re-projected counterparts based on the estimated camera parameters.
Reprojection corner detection can be found in Colab file
One example:





Checkerboard Plane Normals

The normals of the checkerboard planes, denoted as $\mathbf{n}iC$, are calculated for each of the 25 chosen images within the camera coordinate frame of reference, represented by Oc.
Can be found in the colab file
One example:

Normal for image 1: [0.01261055 -0.06130247 0.99803957]

Q5

Computed Plane Normal and Offset
The plane normals and offsets are determined through the singular value decomposition (SVD) of planar LIDAR points. For each image, the normals (denoted as nL) and offsets (denoted as dL) are calculated. Similarly, the normals (denoted as nC) and offsets (denoted as dC) for the camera are computed.

I am using a dictionary called dict
Stored the values under normal_l and offset_l keys

As per 2 in theory
I have theta_c, theta_l, alpha_c and alpha_l of shapes (3,38), (3,38), (38,1) and (38,1) respectively (as per given thesis)
Used to calculate t1 and R1 by formulas derived in theory

Transformation Matrix [R | t
0 ..1]

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

Det(R1) is 0.9999999999999992 (approx =1) which satisfies condition for Rotation Matrix

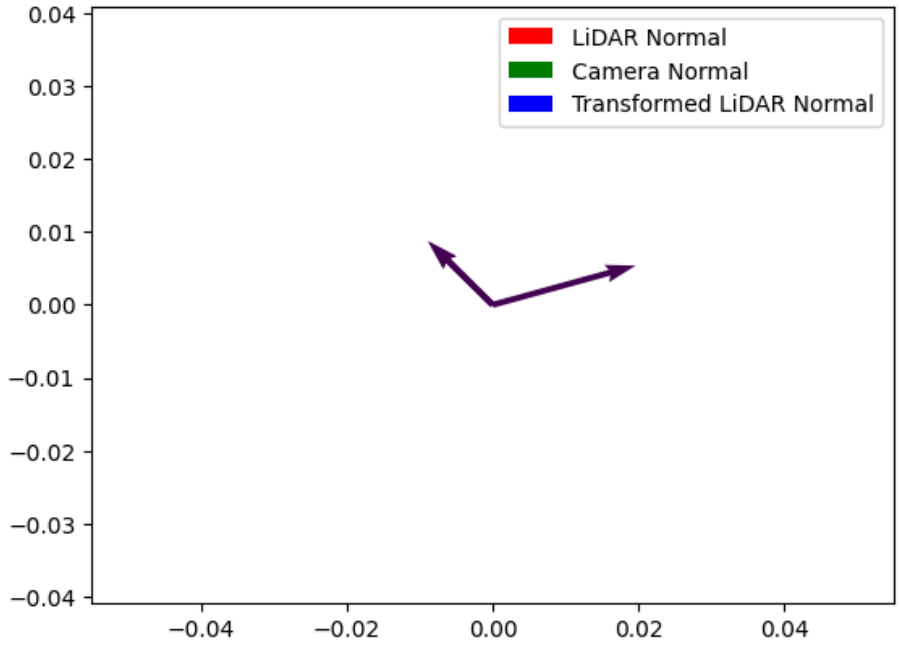
MAPPING lidar points on camera image
Mapped Lidar points on camera image using lidar points and transforming them based on found transformation matrix
Displaying lidar points on given image
Results for all images are present in colab file
Example:

Frame_457

Yes, for most images the found lines are on the checkerboard present in image
There is a slight error in a few frames

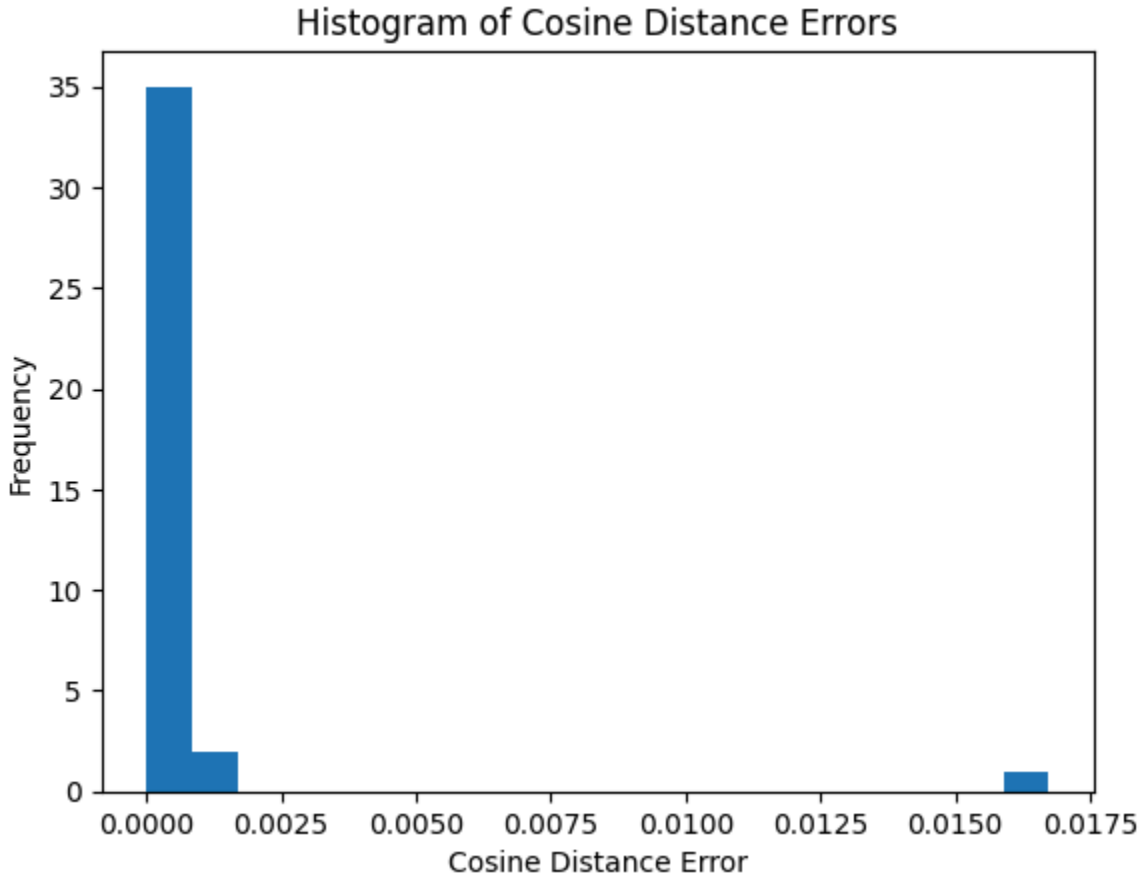
NORMAL VECTORS

Have plotted normal vectors for 5 images, Lidar normals, camera normals and transformed lidar normals. All results in google colab. Example:



It is noticed that the transformed Lidar normal and camera nomal are almost same
And point in the same direction

PLOTS and Histograms:
Calculated the cosine distance between the camera normal and transformed lidar normal and plotted them on a histogram



Average error: 0.00064
Standard Deviation: 0.00266

The error and standard deviation indicate that the camera normal and transformed LIDAR normal are very very close to each other and indicate almost the same normal.