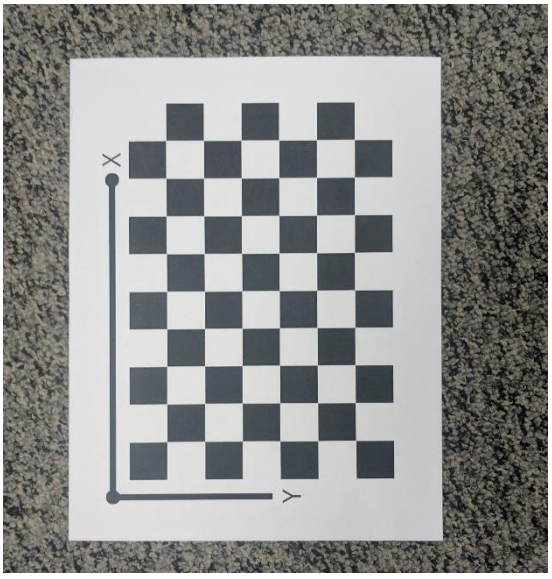
Report

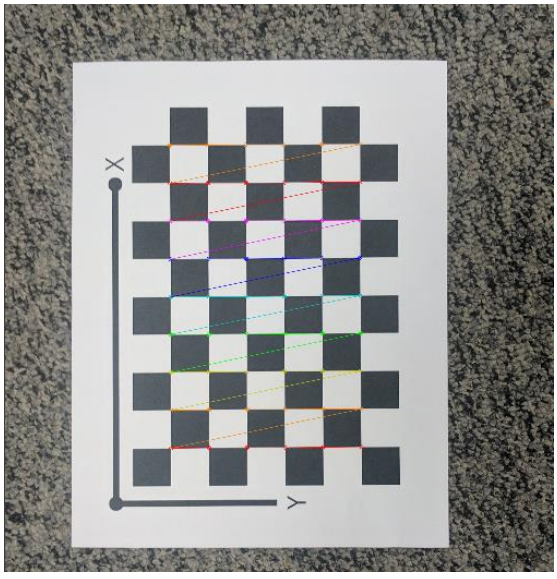
Omkar. A. Chittar

Q 2. In this problem, you will perform camera calibration using the concepts you have learned in class. Assuming a pinhole camera model and ignoring radial distortion, we will be relying on a calibration target (checkerboard in our case) to estimate the camera parameters. The calibration target used can be found here.
This was printed on an A4 paper, and the size of each square is 21.5 mm. Note that the Y axis has an odd number of squares and X axis has an even number of squares. It is a general practice to neglect the outer squares (extreme squares on each side and in both directions).
Thirteen images taken from a Google Pixel XL phone with focus locked can be downloaded from here which you will use to calibrate.

Pipeline:

1. Create an empty list **`reproj_error_list`** to store reprojection errors for each image.
2. Create an empty list **`image_pts`** to store 2D points of corners in each image.
3. Create an empty list **`world_pts`** to store 3D points of corners in each image.
4. Create a NumPy array **`world_coords`** of shape (1, 6x9, 3) initialized with zeros.
    a. Reshape the first two columns to a **(6,9)** grid.
    b. multiply by **21.5** to define the world coordinates for 3D points.
5. Iterate through each image in the folder 'Calibration_Imgs/'.
6. Read the image and convert it to grayscale.
7. Use the OpenCV function **`cv2.findChessboardCorners()`** to detect the corners in the chessboard pattern. If successful, append the image points and world points to **`image_pts`** and **`world_pts`** respectively, and draw the corners on the image using **`cv2.drawChessboardCorners()`.**
8. Resize the image to a third of its size, display it using **`cv2.imshow()`**.





9. Use the OpenCV function **`cv2.calibrateCamera()`** to estimate the camera matrix, distortion coefficients, rotation and translation vectors.
10. Store the results in variables `flag`, `K`, `dist`, `R`, and `T`.
11. Print the intrinsic matrix `K`.

```
Intrinsic matrix:
 [[2.04039471e+03 0.00000000e+00 7.64587598e+02]
 [0.00000000e+00 2.03217467e+03 1.35929491e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
```

12. Iterate through each image in the folder 'Calibration_Imgs/' and do the following:
    a. Use **`cv2.projectPoints()`** to project the world points onto the image plane using the estimated camera matrix, distortion coefficients, rotation and translation vectors.
    b. Calculate the reprojection error as the **L2 norm** between the projected image points and the actual image points, divided by the number of corners (54).
    c. Append the reprojection error to **`reproj_error_list`** and print the image name and its corresponding reprojection error.
    d. Add the reprojection error to **`error_tot`.**
13. Calculate and print the mean reprojection error.

Results:

```
Intrinsic matrix:
 [[2.04039471e+03 0.00000000e+00 7.64587598e+02]
 [0.00000000e+00 2.03217467e+03 1.35929491e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
```

```
Images and their reprojection errors:
IMG_20170209_042606.jpg :  0.07811860604254106
IMG_20170209_042608.jpg :  0.09300490101596924
IMG_20170209_042608.jpg :  0.09300490101596924
IMG_20170209_042610.jpg :  0.11927474143340673
IMG_20170209_042612.jpg :  0.1434668335059415
IMG_20170209_042614.jpg :  0.06785202612852255
IMG_20170209_042616.jpg :  0.07945822777751176
IMG_20170209_042619.jpg :  0.11452972761669802
IMG_20170209_042621.jpg :  0.06683274545547946
IMG_20170209_042624.jpg :  0.07528112226845364
IMG_20170209_042627.jpg :  0.08243326538824611
IMG_20170209_042629.jpg :  0.11240184156012407
IMG_20170209_042630.jpg :  0.12624306572562138
IMG_20170209_042634.jpg :  0.1230613007443954

Mean error =  0.09861218497407008
```

Ways to improve accuracy of the K-matrix:

- **Increasing the number of calibration images:** By capturing a larger number of images from different viewpoints, it becomes possible to obtain a more accurate estimate of the intrinsic parameters.
- **Increasing the diversity of calibration images:** Capturing images with different lighting conditions, different object sizes, and different object distances can make the calibration process more robust and accurate.
- **Using a higher resolution camera:** Higher resolution cameras capture more detail, which can help to improve the accuracy of the corner detection process and reduce the impact of noise.
- **Increasing the size of the calibration target:** A larger target provides more data points, which can help to improve the accuracy of the corner detection process and reduce the impact of noise.
- **Using a more advanced calibration technique:** Using advanced techniques, such as the bundle adjustment method to reduce the reprojection error can help to improve the accuracy of the K-matrix.