In [1]:

```
# Import required modules
import cv2
import numpy as np
import os
import glob



# stop the iteration when specified
# accuracy, epsilon (desired Accuracy), is reached or
# specified number of iterations are completed.
criteria = (cv2.TERM_CRITERIA_EPS +
            cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)


# Vector for 3D points
threedpoints = []

# Vector for 2D points
twodpoints = []


#  3D points real world coordinates
objectp3d = np.zeros((6*9,3), np.float32)
objectp3d[:,:2] = np.mgrid[0:6,0:9].T.reshape(-1, 2)
prev_img_shape = None


# Extracting path of individual image stored
# in a given directory. Since no path is
# specified, it will take current directory
# jpg files alone
images = glob.glob('*.jpg')

for filename in images:
    image = cv2.imread(filename)
    grayColor = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Find the chess board corners
    # If desired number of corners are
    # found in the image then ret = true
    #retval, corners=cv.findChessboardCorners(image, patternSize[, corners[, flags]]
```

```
    ret, corners = cv2.findChessboardCorners(
                    grayColor, (6,9),
                    cv2.CALIB_CB_ADAPTIVE_THRESH
                    + cv2.CALIB_CB_FAST_CHECK +
                    cv2.CALIB_CB_NORMALIZE_IMAGE)

    # If desired number of corners can be detected then,
    # refine the pixel coordinates and display
    # them on the images of checker board
    if ret == True:
        threedpoints.append(objectp3d)

        # Refining pixel coordinates
        # for given 2d points. corners=cv.cornerSubPix(image, corners, winSize, zeroZone, criteria
        corners2 = cv2.cornerSubPix(
            grayColor, corners, (11, 11), (-1, -1), criteria)

        twodpoints.append(corners2)

        # Draw and display the corners
        image = cv2.drawChessboardCorners(image,
                                          (6,9),
                                          corners2, ret)

    cv2.imshow('img', image)
    cv2.waitKey(0)

cv2.destroyAllWindows()


# Perform camera calibration by
# passing the value of above found out 3D points (threedpoints)
# and its corresponding pixel coordinates of the
# detected corners (twodpoints)
ret, matrix, distortion, r_vecs, t_vecs = cv2.calibrateCamera(
    threedpoints, twodpoints, grayColor.shape[::-1], None, None)


# Displayig required output
print(" Camera matrix:")
print(matrix)

print("\n Distortion coefficient:")
```

```
print(distortion)

print("\n Rotation Vectors:")
print(r_vecs)

print("\n Translation Vectors:")
print(t_vecs)
```

```
 Camera matrix:
[[20.10654304  0.         84.16362263]
 [ 0.         20.34239482 95.42267081]
 [ 0.          0.          1.        ]]

 Distortion coefficient:
[[-9.40501496e-04  3.73198946e-05 -2.32754445e-03  3.95213785e-04
  -6.01340412e-07]]

 Rotation Vectors:
[array([[-0.04742568],
       [ 0.02932197],
       [ 1.50950267]]), array([[-0.07882398],
       [-0.00961833],
       [ 3.07805624]]), array([[-0.01784273],
       [ 0.04617962],
       [-0.07272072]])]

 Translation Vectors:
[array([[ 4.63365547],
       [-3.7646618 ],
       [ 1.35562517]]), array([[2.32806935],
       [3.99318851],
       [1.36446905]]), array([[-3.16534453],
       [-3.45998477],
       [ 1.38547247]])]
```