Computer Vision

LAB 3

Submitted By:

Gaurav Agarwal	BT17CSE001
Abhibha Gupta	BT17CSE020
Thogaru Himabindu	BT17CSE056

Submitted to:

Dr. Mayur Parate

Task:

CALIBRATE STEREO CAMERA(BINOCULAR VISION) AND FIND STEREO PARAMETERS

भारतीय सूचना प्रौद्योगिकी संस्थान, नागपूर

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, NAGPUR

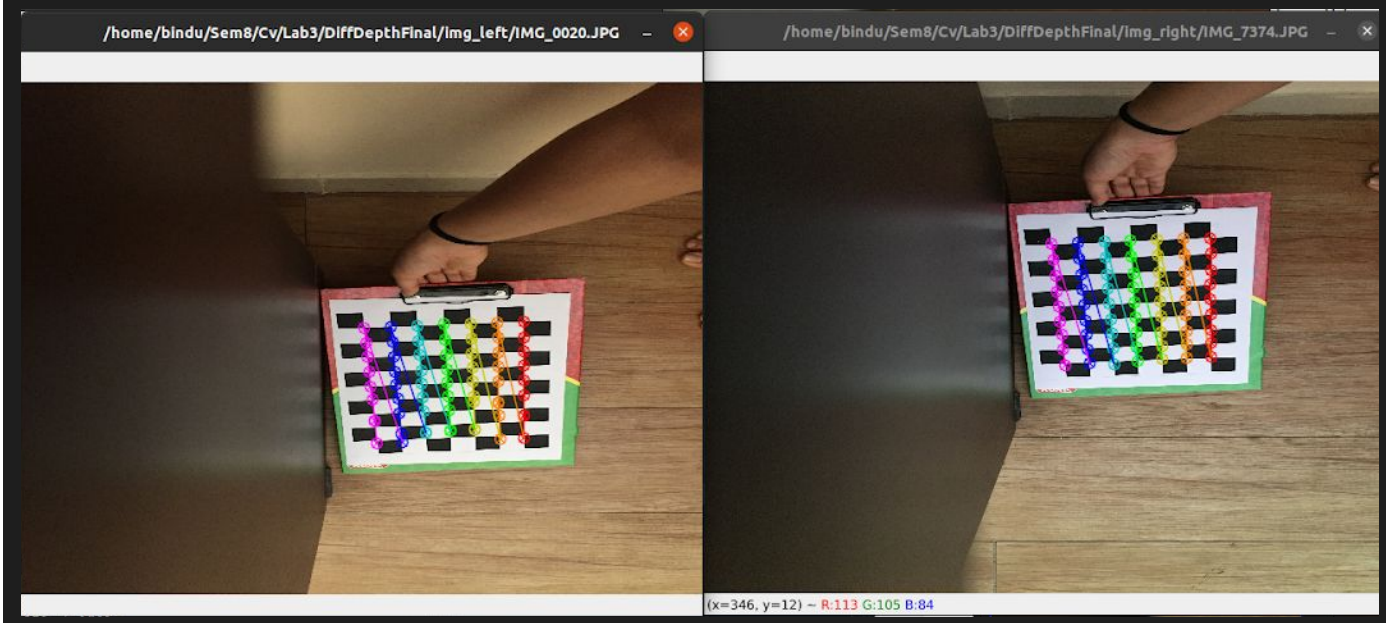
(An Institution of National Importance by Act of Parliament)

BSNL RTTC, Near TV Tower, Besides Balaji Temple, Seminary Hills, Nagpur-440006

Stereo Calibration and finding parameters code:

```
import numpy as np
import cv2
import glob

criteria = (cv2.TERM_CRITERIA_EPS +cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)

objp = np.zeros((9*7, 3), np.float32)
objp[:, :2] = np.mgrid[0:9, 0:7].T.reshape(-1, 2)

objpoints = []
L_imgp = []
R_imgp = []

# images_right = glob.glob('/home/bindu/Sem8/Cv/Lab3/Images/Right/*.jpg')
# images_left = glob.glob('/home/bindu/Sem8/Cv/Lab3/Images/Left/*.jpg')
images_left = glob.glob('/home/bindu/Sem8/Cv/Lab3/DiffDepthFinal/img_left/*.JPG')
images_right = glob.glob('/home/bindu/Sem8/Cv/Lab3/DiffDepthFinal/img_right/*.JPG')
images_left.sort()
images_right.sort()
print(len(images_right))
print(len(images_left))
for i, fname in enumerate(images_right):
    L_img = cv2.imread(images_left[i])
    L_img=cv2.resize(L_img,(640,480))
    R_img = cv2.imread(images_right[i])
    R_img=cv2.resize(R_img,(640,480))

    gray_l = cv2.cvtColor(L_img, cv2.COLOR_BGR2GRAY)
    gray_r = cv2.cvtColor(R_img, cv2.COLOR_BGR2GRAY)
    ret_l, corners_l = cv2.findChessboardCorners(gray_l, (9, 7), None)
    ret_r, corners_r = cv2.findChessboardCorners(gray_r, (9, 7), None)
    objpoints.append(objp)
    if ret_l and ret_r :
        rt = cv2.cornerSubPix(gray_l, corners_l, (11, 11),(-1, -1), criteria)
        L_imgp.append(corners_l)
        ret_l = cv2.drawChessboardCorners(L_img, (9, 7),corners_l, ret_l)
        cv2.imwrite("Leftt%d.JPG"%i,L_img)
        cv2.imshow(images_left[i], L_img)
```

```
        rt = cv2.cornerSubPix(gray_r, corners_r, (11, 11),(-1, -1), criteria)
        R_imgp.append(corners_r)
        ret_r = cv2.drawChessboardCorners(R_img, (9, 7),corners_r, ret_r)
        cv2.imwrite("Rightt%d.JPG"%i,R_img)
        cv2.imshow(images_right[i], R_img)
        cv2.waitKey(0)
```



Finding STEREO PARAMETERS:

```
rt, M1, d1, r1, t1 = cv2.calibrateCamera(objpoints, L_imgp, gray_l.shape[::-1], None, None)
rt, M2, d2, r2, t2 = cv2.calibrateCamera(objpoints, R_imgp, gray_l.shape[::-1], None, None)

flags = 0
flags |= cv2.CALIB_FIX_INTRINSIC
flags |= cv2.CALIB_USE_INTRINSIC_GUESS
flags |= cv2.CALIB_FIX_FOCAL_LENGTH
flags |= cv2.CALIB_ZERO_TANGENT_DIST
flags |= cv2.CALIB_SAME_FOCAL_LENGTH

stereocalib_criteria = (cv2.TERM_CRITERIA_MAX_ITER +cv2.TERM_CRITERIA_EPS, 100, 1e-5)
```

StereoCalibrate:

```
ret, M1, d1, M2, d2, R, T, E, F = cv2.stereoCalibrate(
                            objpoints, L_imgp,R_imgp, M1,1, M2,d2, gray_l.shape[::-1],
```

```
                            criteria=stereocalib_criteria, flags=flags)
```

StereoRectify:

```
R1, R2, P1, P2, Q, roi_left, roi_right=cv2.stereoRectify(M1, d1,M2, d2,
                                    gray_l.shape[::-1],R, T, flags=flags, alpha=-1 )


print('Intrinsic_mtx_1', M1)
print('dist_1', d1)
print('Intrinsic_mtx_2', M2)
print('dist_2', d2)
print('R', R)
print('T', T)
print('E', E)
print('F', F)
print('R1',R1)
print('R2',R2)
print('P1',P1)
print('P2',P2)
print('Q',Q)
```

OUTPUT:

Stereo Parameters:

```
Intrinsic_mtx_1 : First camera intrinsic matrix
 [[756.78804989   0.         307.10702366]
 [  0.         450.15053796 239.88300551]
 [  0.           0.           1.        ]]


dist_1 : First camera distortion parameters
[[ 0.08020368 -0.05909418  0.01615246 -0.00879274 -0.02287194]]



Intrinsic_mtx_2 : Second camera intrinsic matrix
[[749.66170942   0.         318.19368287]
 [  0.         449.82019968 239.18192862]
 [  0.           0.           1.        ]]
```

```
dist_2 : Second camera distortion parameters
[[-1.09046851e-01  1.78447549e+00  2.04526298e-03  5.09967045e-03
  -6.88785699e+00]]


R : Output Rotation matrix from the coordinate system of the first camera to the
second cam
[[ 0.99967074  0.02549011  0.00294468]
 [-0.02462468  0.98527574 -0.16919024]
 [-0.007214    0.16906202  0.98557901]]


T :  Output translation vector
 [[-0.07186457]
 [-0.21078138]
 [-0.22105945]]


E   :  Output essential matrix
[[-0.00392294  0.18216939 -0.2451428 ]
 [-0.22150509  0.00651474  0.07017727]
 [ 0.21248162 -0.06543358  0.01277947]]


F  :  Output fundamental matrix
[[-4.24460060e-07  3.29933674e-05 -2.77490466e-02]
 [-4.01176015e-05  1.97503436e-06  2.16432326e-02]
 [ 2.70591094e-02 -1.97148778e-02  1.00000000e+00]]


R1   :  Output 3x3 rectification transform (rotation matrix) for the first camera
[[ 0.97533891 -0.21222811 -0.06060727]
 [ 0.20732638  0.78679007  0.58135803]
 [-0.07569532 -0.57958659  0.81138745]]


R2   :  Output 3x3 rectification transform (rotation matrix) for the second camera
 [[ 0.96942958 -0.22286646 -0.10264905]
 [ 0.22902539  0.67173972  0.70449494]
 [-0.08805484 -0.70646747  0.70224643]]
```

```
P1  :    Output 3x4 projection matrix in the new (rectified) coordinate systems for
the first camera
 [[ 753.22487966    0.          444.63934898    0.        ]
 [   0.          753.22487966 -539.62276173    0.        ]
 [   0.            0.            1.            0.        ]]


P2  :  Output 3x4 projection matrix in the new (rectified) coordinate systems for
the second camera
 [[ 7.53224880e+02  0.00000000e+00  4.44639349e+02  0.00000000e+00]
[ 0.00000000e+00  7.53224880e+02 -1.14327922e+03 -2.36350141e+02]
[ 0.00000000e+00  0.00000000e+00  1.00000000e+00  0.00000000e+00]]

Q   :  Output 4×4 disparity-to-depth mapping matrix

 [[ 1.00000000e+00  0.00000000e+00  0.00000000e+00 -4.44639349e+02]
 [ 0.00000000e+00  1.00000000e+00  0.00000000e+00  5.39622762e+02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  7.53224880e+02]
 [ 0.00000000e+00  0.00000000e+00  3.18690260e+00 -1.92379433e+03]]



                                    THE END
```