Computer Vision

LAB 2

Submitted By:

Gaurav Agarwal	**BT17CSE001**
Abhibha Gupta	**BT17CSE020**
Thogaru Himabindu	**BT17CSE056**

Task:

CALIBRATE A CAMERA AND FIND CAMERA PARAMETERS AND MATRICES

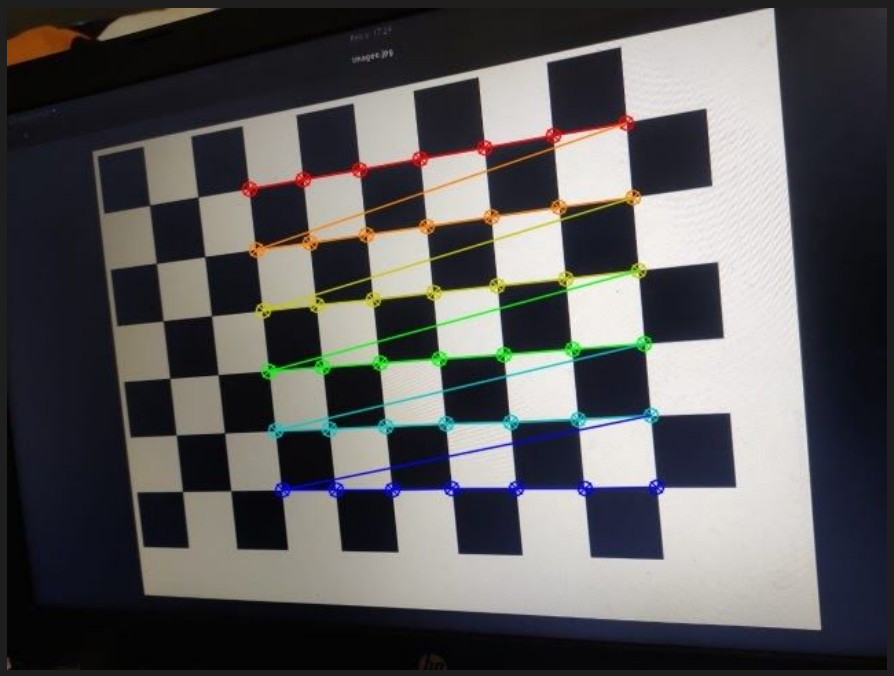
CODE

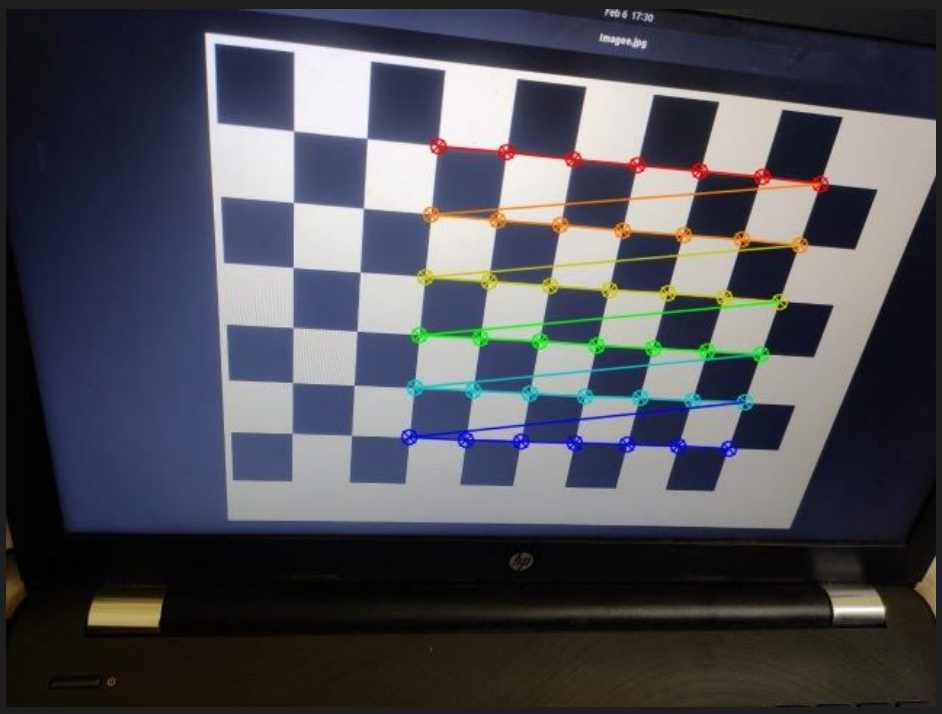
#python code

```
import numpy as np
import cv2
import glob

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

# prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
objp = np.zeros((7*6,3), np.float32)
objp[:,:2] = np.mgrid[0:7,0:6].T.reshape(-1,2)
```

```
# Arrays to store object points and image points from all the images.
objpoints = [] # 3d point in real world space
imgpoints = [] # 2d points in the image plane.


images = glob.glob('/home/bindu/Sem8/Cv/Lab2/images/*.jpeg')
x=0
for fname in images:
    x=x+1
    img = cv2.imread(fname)
    img=cv2.resize(img,(640,480))
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    # Find the chess board corners
    ret, corners = cv2.findChessboardCorners(gray, (7,6),None)

    # If found, add object points, image points (after refining them)
    if ret == True:
        objpoints.append(objp)

        corners2 = v2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)
        imgpoints.append(corners2)

        # Draw and display the corners
        img = cv2.drawChessboardCorners(img, (7,6), corners2,ret)
        cv2.imshow('img',img)
        filename = "outputfile_%d.jpeg"%x
        cv2.imwrite(filename,img)
        cv2.waitKey(0)

cv2.destroyAllWindows()
```

Some output images with pattern drawn on it is shown below:





Calibration

```
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints,
gray.shape[::-1],None,None)
```

Parameters and matrices

```
matrix=[
         4.930116392799483833e+02   0         3.173879449057979514e+02
         0        4.941837326867249089e+02    2.405454965968862666e+02
         0                   0                          1
]
Distortion coefficients=
k1=  -5.969114411690056021e-02
k2=  5.346075355665105278e-01
p1=  5.671754647779813502e-04
p2=  -1.613125666727051365e-03
k3=  -1.071608653774593467e+00
rvecs=[array([[ 0.4882305 ],
       [-0.14180271],
       [ 0.08822365]]), array([[ 0.04209217],
       [ 0.41899054],
       [-0.01366359]]), array([[ 0.0598714 ],
       [ 0.32231956],
       [-0.08858659]])]
tvecs=[array([[-0.42338894],
       [-3.03703289],
       [10.20377826]]), array([[-3.14168404],
       [-2.50748866],
       [10.19925203]]), array([[-3.18628047],
       [-2.47243231],
       [11.10133887]])]
```

Undistortion

```
img = cv2.imread('/home/bindu/Sem8/Cv/Lab2/images/img26.jpeg')
img=cv2.resize(img,(640,480))
h,  w = img.shape[:2]
newcameramtx, roi=cv2.getOptimalNewCameraMatrix(mtx,dist,(w,h),1,(w,h))
```

Parameters

```
newcameramtx=[
         3.289212951660156250e+02 0    2.495577079381382646e+02
         0    3.893707275390625000e+02 2.571721438580425456e+02
                                  0    0    1
]

roi=(38, 68, 427, 377)
```

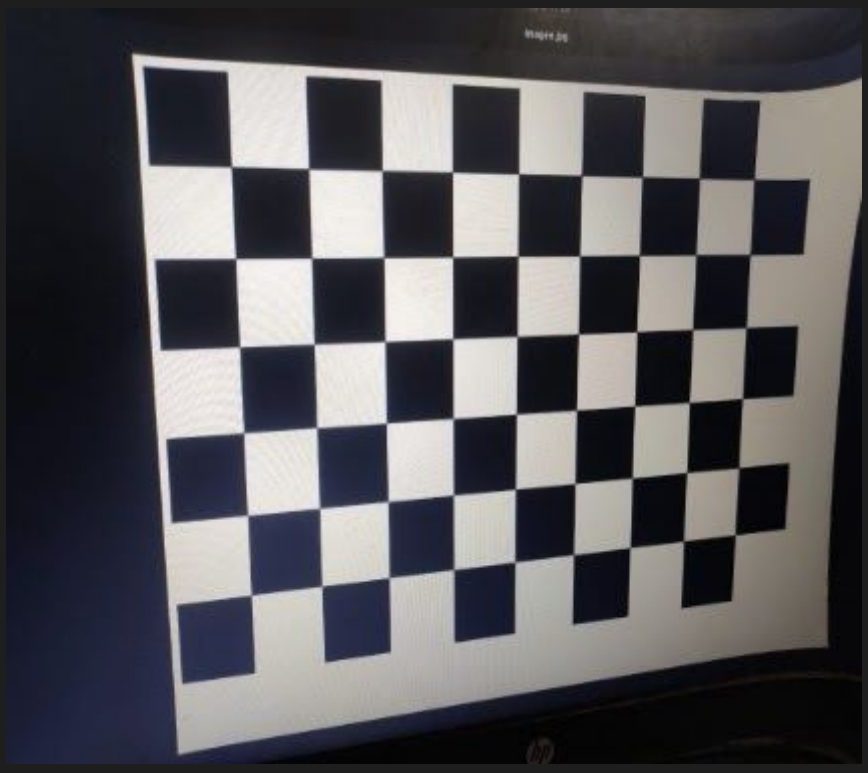
METHODS OF DISTORTION:

OpenCv comes with 2 methods of distortion:

1. cv2.undsitort()
2. remapping

Method 1: cv2.undistort()

```
# undistort
dst = cv2.undistort(img, mtx, dist, None, newcameramtx)
# crop the image
x,y,w,h = roi
dst = dst[y:y+h, x:x+w]
cv2.imwrite('calibresult.jpeg',dst)
```



Method 2: Remapping

```
# undistort
mapx,mapy =
cv2.initUndistortRectifyMap(mtx,dist,None,newcameramtx,(w,h),5)
dst = cv2.remap(img,mapx,mapy,cv2.INTER_LINEAR)

# crop the image
x,y,w,h = roi
dst = dst[y:y+h, x:x+w]
cv2.imwrite('calibresult1.jpeg',dst)
```

Reprojection Error

```
mean_error = 0
for i in range(len(objpoints)):
    imgpoints2, _ = cv2.projectPoints(objpoints[i], rvecs[i], tvecs[i],
mtx, dist)
    error = cv2.norm(imgpoints[i],imgpoints2,
cv2.NORM_L2)/len(imgpoints2)
    mean_error += error

total_error=mean_error/len(objpoints)
print("total error: ",total_error )
```

```
----------------------------------------
total error:  0.019291838747248054
----------------------------------------
```

Difficulties:

- One problem when calibrating a camera is that the residual error cannot be trusted as a way to verify correctness.
- Extreme angle calibration images must be avoided.

Conclusion :
It is essential to know the parameters of a camera to use it effectively as a visual sensor and camera calibration determines the geometric parameters of the image formation process.