Assignment 3

Task 1

In [1]:

```
import numpy as np
import cv2
import os

def callibrate_camera(imgCategory, cameraName):
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
    objp = np.zeros((7*10,3), np.float32)
    objp[:,:2] = np.mgrid[0:10,0:7].T.reshape(-1,2)

    objpoints = [] # 3d point in real world space
    imgpoints = [] # 2d points in image plane.

    imagesFolder = os.path.join(os.getcwd(),'Downloads',str(imgCategory), str(cameraNam
    images = [ os.path.join(imagesFolder, f) for f in os.listdir('Downloads/'+ str(imgC

    for fname in images:
        img = cv2.imread(fname)
        gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
        ret, corners = cv2.findChessboardCorners(gray, (10,7),None)
        if ret == True:
            objpoints.append(objp)

            corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)
            imgpoints.append(corners2)

            img = cv2.drawChessboardCorners(img, (10,7), corners2,ret)
            cv2.imshow('img',img)
            cv2.waitKey(50)

    ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape
    print('Intrinsic  parameters from '+ str(cameraName) + '= \n',mtx)
    print('Distortion parameters from '+ str(cameraName) + '= \n',dist,'\n')

    np.save('Parameters/'+str(imgCategory)+'/'+ str(cameraName) +'/'+'intrinsic_paramet
    np.save('Parameters/'+str(imgCategory)+'/'+ str(cameraName) +'/'+'distortion_parame

    cv2.destroyAllWindows()

##Practice Images
# callibrate_camera('Practice_Images', 'L')
# callibrate_camera('Practice_Images', 'R')

##Test Images
#Left Camera Callibration
callibrate_camera('Test_Images','Left_Cali')
#Right Camera Callibration
callibrate_camera('Test_Images','Right_Cali')
```

```
Intrinsic  parameters from Left_Cali=
 [[1.75217083e+03 0.00000000e+00 3.37377746e+02]
 [0.00000000e+00 1.75518793e+03 2.25045638e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
Distortion parameters from Left_Cali=
 [[-4.40812031e-01  3.52324408e-01  5.87232218e-03  3.25912689e-04
   5.83560509e+00]]
```

```
Intrinsic  parameters from Right_Cali=
 [[1.74935190e+03 0.00000000e+00 3.07413499e+02]
 [0.00000000e+00 1.75443478e+03 2.08938564e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
Distortion parameters from Right_Cali=
 [[-5.28875909e-01  3.68491062e-01  4.57006164e-03  5.58489376e-03
   2.69162283e+01]]
```

Task 2

In [1]:

```
import numpy as np
import cv2
import os

def find_corners(imageSet, CameraL, CameraR, chessBoardSize):
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
    objp = np.zeros((7*10,3), np.float32)
    objp[:,:2] = chessBoardSize * np.mgrid[0:10,0:7].T.reshape(-1,2)

    objpoints = [] # 3d point in real world space
    L_imgpoints = [] # 2d points in image plane.
    R_imgpoints = [] # 2d points in image plane.

    head_list = [str(CameraL), str(CameraR)]
    for head in head_list:
        imagesFolder = os.path.join(os.getcwd(),'Downloads', str(imageSet), str(head))
        images = [ os.path.join(imagesFolder, f) for f in os.listdir('Downloads/'+str(i
        for pic in images:
            img = cv2.imread(pic)
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            ret, corners = cv2.findChessboardCorners(gray, (10,7),None)
            if ret == True:
                corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)
                if head == str(CameraL):
                    objpoints.append(objp)
                    L_imgpoints.append(corners2)
                elif head == str(CameraR):
                    R_imgpoints.append(corners2)

                img = cv2.drawChessboardCorners(img, (10,7), corners2,ret)
                cv2.imshow('image',img)
                cv2.waitKey(50)
    return objpoints, L_imgpoints, R_imgpoints, gray.shape[::-1]

def  calculate_extrinsic_parameters(imageSet, CameraL, CameraR, chessBoardSize, paramL,
    objpoints, L_imgpoints, R_imgpoints, shape = find_corners(imageSet, CameraL, Camera
    L_intrinsic = np.load('Parameters/'+ str(imageSet) + '/' + str(paramL)+'/intrinsic_
    L_distortion = np.load('Parameters/'+ str(imageSet) + '/' + str(paramL)+'/distortio
    R_intrinsic = np.load('Parameters/'+ str(imageSet) + '/' + str(paramR)+'/intrinsic_
    R_distortion = np.load('Parameters/'+ str(imageSet) + '/' + str(paramR)+'/distortio

    termination_criteria_extrinsics = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_IT
    # termination_criteria_extrinsics = (cv2.TERM_CRITERIA_COUNT + cv2.TERM_CRITERIA_EP

    rms_stereo, stereo_camera_matrix_l, stereo_dist_coeffs_l, stereo_camera_matrix_r, s
        cv2.stereoCalibrate(objpoints, L_imgpoints, R_imgpoints, L_intrinsic, L_distort

    print('R = \n', R, '\n')
    print('T = \n', T, '\n')
    print('E = \n', E, '\n')
    print('F = \n', F, '\n')

    if CameraL == "Stereo_Left" and CameraR == "Stereo_Right":
        np.save('rotation_matrix.npy', R)
        np.save('translation_vector.npy', T)
        np.save('essential_matrix.npy', E)
        np.save('fundamental_matrix.npy', F)
    cv2.destroyAllWindows()


calculate_extrinsic_parameters("Test_Images", "Stereo_Left", "Stereo_Right", 3.88, "Lef
# calculate_extrinsic_parameters("Practice_Images", "SL", "SR", 2.,"L", "R")
```

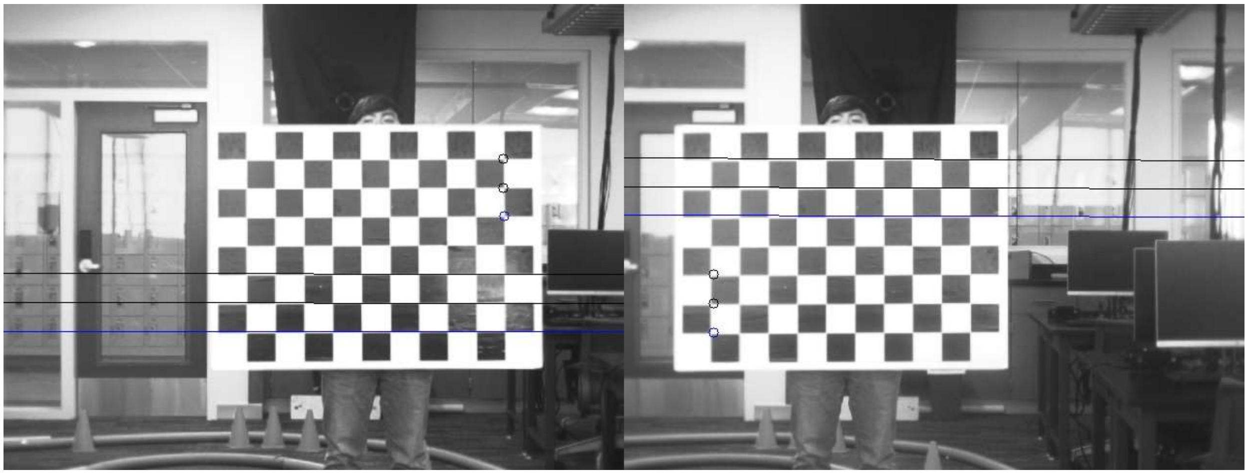
```
R =
 [[ 9.99899140e-01  6.26226123e-04  1.41886351e-02]
 [-7.64301854e-04  9.99952389e-01  9.72809983e-03]
 [-1.41818676e-02 -9.73796306e-03  9.99852012e-01]]

T =
 [[-20.34983933]
 [ -0.06394045]
 [ -0.6384341 ]]

E =
 [[ 4.18838679e-04  6.39026355e-01 -5.77202404e-02]
 [-9.26968437e-01 -1.98565788e-01  2.03377693e+01]
 [ 7.94874244e-02 -2.03488304e+01 -1.97058041e-01]]

F =
 [[-5.44824899e-09 -8.29815912e-06  3.18487500e-03]
 [ 1.20230614e-05  2.57103130e-06 -4.66834849e-01]
 [-4.31918643e-03  4.64266655e-01  1.00000000e+00]]
```

Task 3

In [4]:

```
import numpy as np
import cv2
from IPython.display import Image

L_intrinsic = np.load('Parameters/Test_images/Left_Cali/intrinsic_parameters.npy')
R_intrinsic = np.load('Parameters/Test_images/Left_Cali/intrinsic_parameters.npy')
L_distortion = np.load('Parameters/Test_images/Right_Cali/distortion_parameters.npy')
R_distortion = np.load('Parameters/Test_images/Right_Cali/distortion_parameters.npy')
fundamental_matrix = np.load('fundamental_matrix.npy')

def undistortion(name, camera_matrix, dist_coeffs):
    img = cv2.imread(name)
    h, w = img.shape[:2]
    h += 1
    w += 1
    newcameramtx, roi = cv2.getOptimalNewCameraMatrix(camera_matrix, dist_coeffs, (w, h
    dst = cv2.undistort(img, camera_matrix, dist_coeffs, None, camera_matrix)
    x, y, w, h = roi
    dst = dst[y:y + h, x:x + w]
    # cv2.imwrite('Undistortion' + name, dst)
    return dst

def find_corners_of_an_image(image_name, chessboard_size):
    image = cv2.imread(image_name)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    ret, corners = cv2.findChessboardCorners(gray, chessboard_size, None)
    return corners.reshape(-1,2)

def drawPoints(img, pts, colors):
    for pt, color in zip(pts, colors):
        cv2.circle(img, tuple(pt), 5, color, -1)

def drawLines(img, lines, colors):
    _, c, _ = img.shape
    for r, color in zip(lines, colors):
        x0, y0 = map(int, [0, -r[2]/r[1]])
        x1, y1 = map(int, [c, -(r[2]+r[0]*c)/r[1]])
        cv2.line(img, (x0, y0), (x1, y1), color, 1)


undstL = undistortion('Downloads/Test_Images/Stereo_Left/L1.png', L_intrinsic, L_distor
undstR = undistortion('Downloads/Test_Images/Stereo_Right/R1.png', R_intrinsic, R_disto

imgptsL = find_corners_of_an_image('Downloads/Test_Images/Stereo_Left/L1.png', (7,10))
imgptsR = find_corners_of_an_image('Downloads/Test_Images/Stereo_Right/R1.png', (7,10))

ptsL = np.array([imgptsL[0], imgptsL[1], imgptsL[2]])
ptsR = np.array([imgptsR[-1], imgptsR[-2], imgptsR[-3]])
drawPoints(undstL, ptsL, (0, 0, 255))
drawPoints(undstR, ptsR, (255, 0, 0))

epilinesR = cv2.computeCorrespondEpilines(ptsR.reshape(-1, 1, 2), 2, fundamental_matrix
epilinesR = epilinesR.reshape(-1, 3)
drawLines(undstL, epilinesR, (255, 0, 0))

epilinesL = cv2.computeCorrespondEpilines(ptsL.reshape(-1, 1, 2), 1, fundamental_matrix
epilinesL = epilinesL.reshape(-1, 3)
drawLines(undstR, epilinesL, (0, 0, 255))

img = cv2.hconcat([undstL, undstR])
cv2.imshow('Epipolar_lines',img)
cv2.imwrite('Epipolar_lines.jpg',img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

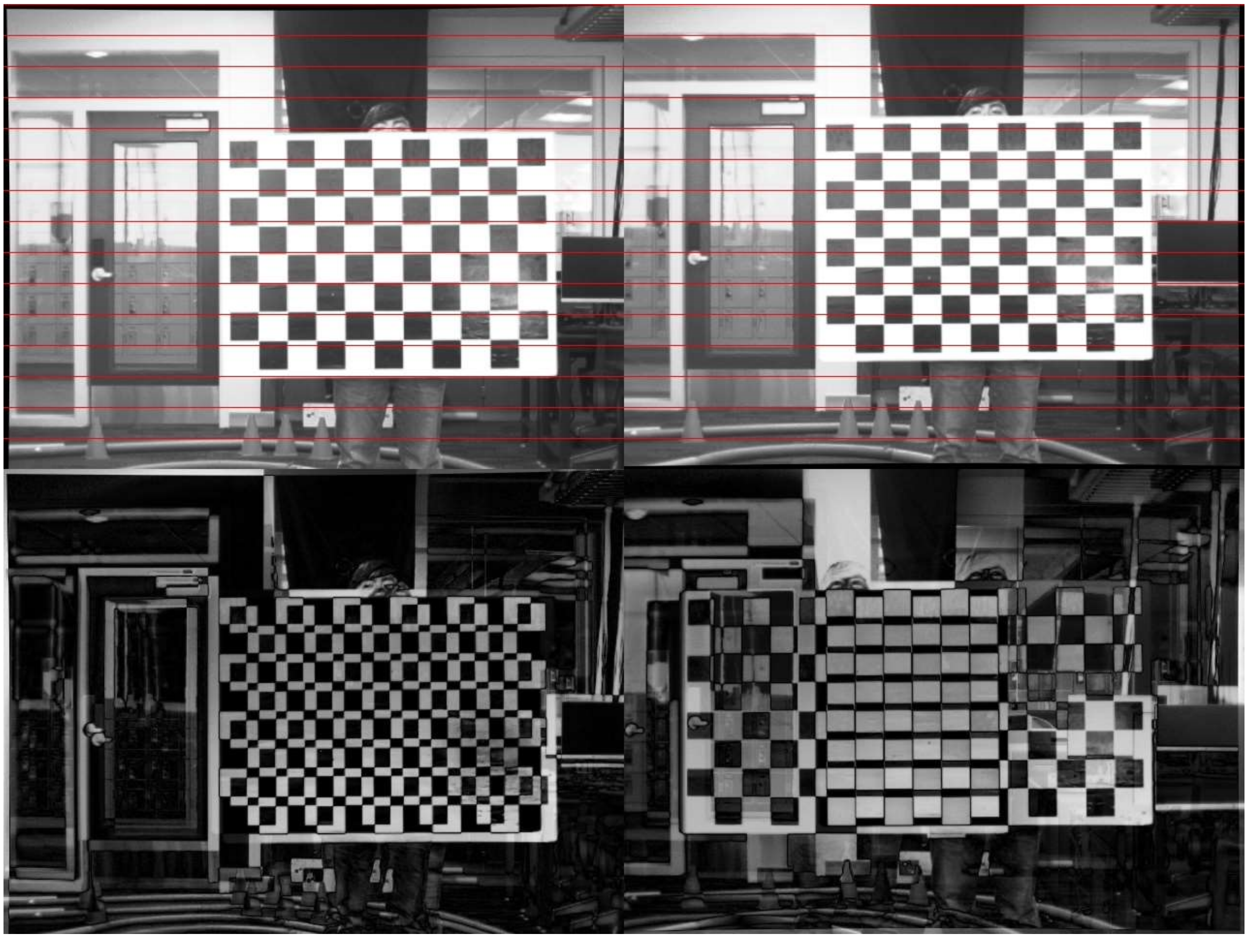
```
<ipython-input-4-3cb7598ece3b>:30: DeprecationWarning: an integer is required (got type
numpy.float32).  Implicit conversion to integers using __int__ is deprecated, and may be
removed in a future version of Python.
  cv2.circle(img, tuple(pt), 5, color)
```



Task 4

In [6]:

```
import cv2
import numpy as np

L_intrinsic = np.load('Parameters/Test_images/Left_Cali/intrinsic_parameters.npy')
R_intrinsic = np.load('Parameters/Test_images/Left_Cali/intrinsic_parameters.npy')
L_distortion = np.load('Parameters/Test_images/Right_Cali/distortion_parameters.npy')
R_distortion = np.load('Parameters/Test_images/Right_Cali/distortion_parameters.npy')
cameraParameters = [ L_intrinsic, L_distortion, R_intrinsic, R_distortion]

rotation_matrix = np.load('rotation_matrix.npy')
translation_vector = np. load('translation_vector.npy')
pose = [ rotation_matrix, translation_vector]

def compute_stereo_rectification_maps(imgLName, imgRName, camParams, PoseParams):
    imgL = cv2.imread(imgLName)
    imgR = cv2.imread(imgRName)
    h, w = imgL.shape[:2]
    imgSize = (w, h)

    R1, R2, P1, P2, Q, roi1, roi2 = cv2.stereoRectify(camParams[0], camParams[1], camPa
                                                       imgSize, PoseParams[0], PoseParam
    map1x, map1y = cv2.initUndistortRectifyMap(camParams[0], camParams[1], R1, P1, imgS
    map2x, map2y = cv2.initUndistortRectifyMap(camParams[2], camParams[3], R2, P2, imgS

    remapL = cv2.remap(imgL, map1x, map1y, cv2.INTER_LINEAR)
    remapY = cv2.remap(imgL, map2x, map2y, cv2.INTER_LINEAR)
    return remapL, remapY

def Difference(imagename_ori, remap):
    img_ori = cv2.imread(imagename_ori)
    gray1 = cv2.cvtColor(img_ori, cv2.COLOR_BGR2GRAY)
    gray2 = cv2.cvtColor(remap, cv2.COLOR_BGR2GRAY)
    diff = cv2.absdiff(gray1, gray2)
    return diff

imgL = 'Downloads/Test_Images/Stereo_Left/L1.png'
imgR = 'Downloads/Test_Images/Stereo_Right/R1.png'

remapL, remapR = compute_stereo_rectification_maps(imgL, imgR, cameraParameters, pose)
diffL = Difference(imgL, remapL)
diffR = Difference(imgR, remapR)

diffLC = cv2.cvtColor(diffL, cv2.COLOR_GRAY2BGR)
diffRC = cv2.cvtColor(diffR, cv2.COLOR_GRAY2BGR)

for y in range(20):
    cv2.line(remapL, (0, y*32), (640, y*32), (0, 0, 255), 1)
    cv2.line(remapR, (0, y*32), (640, y*32), (0, 0, 255), 1)
hor1 = cv2.hconcat([remapL, remapR])
hor2 = cv2.hconcat([diffLC, diffRC])
finalImg = cv2.vconcat([hor1, hor2])
cv2.imshow('Task_4', finalImg)
cv2.imwrite('Task_4.jpg', finalImg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



```
In [ ]:
```