**EE6222 Assignment 2 Report**

**LI RIXUAN G1801134G**

**1. Hand Pick Points:**

**1.1** Take two pictures at the same focal length and set two picture’s size scale down at the same proportion for better programming speed. Two pictures are shown below left and right respectively:

**1.2** Put each picture into the same getting point coordinate python program and click 8 corresponding points, record their coordinates. Make sure the picking point’s order are paired.

 Get 8 points shown below left and right respectively:

Write down their coordinates as PL&PR:

PL=(u,v)=[1122,524];[764,548];[1314,97];[1206,482];

[777,482];[813,530];[1123,848];[404,340]

PR=(u’,v’) [1295,724];[810,523];[1302,29];[1203,442];

[734,432];[800;485]; [1203,442];[391,213]

Core codes are as flows:

frame1 = cv2.imread('./LEFT2.jpg')

def callback(event,x,y,flags,param):

global display

if event == cv2.EVENT\_LBUTTONDOWN:

display = frame1.copy()

print(x,y)

cv2.putText(display,'(%d,%d)'%(x,y),(x+5,y),cv2.FONT\_HERSHEY\_COMPLEX\_SMALL,1,(0,0,255),2)

cv2.circle(display, tuple([x,y]), 5, (0,255,255), 2)

cv2.imshow('display',display)

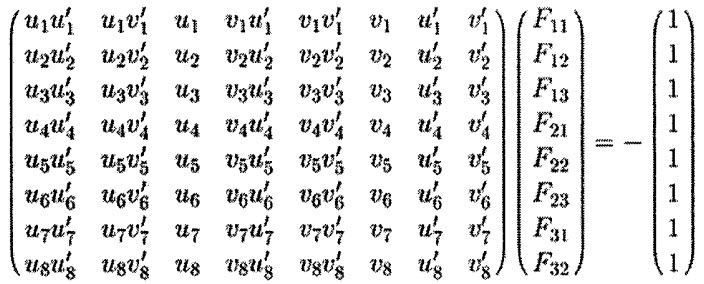
cv2.namedWindow('display')

cv2.imshow('display',frame1)

cv2.setMouseCallback('display',callback)

cv2.waitKey(0)

**2. Calculate F Matrix and Epipole:**

2.1 Use the eight-point algorithm to calculate Fundamental Matrix, the equation is as follow:

Here is the F result:

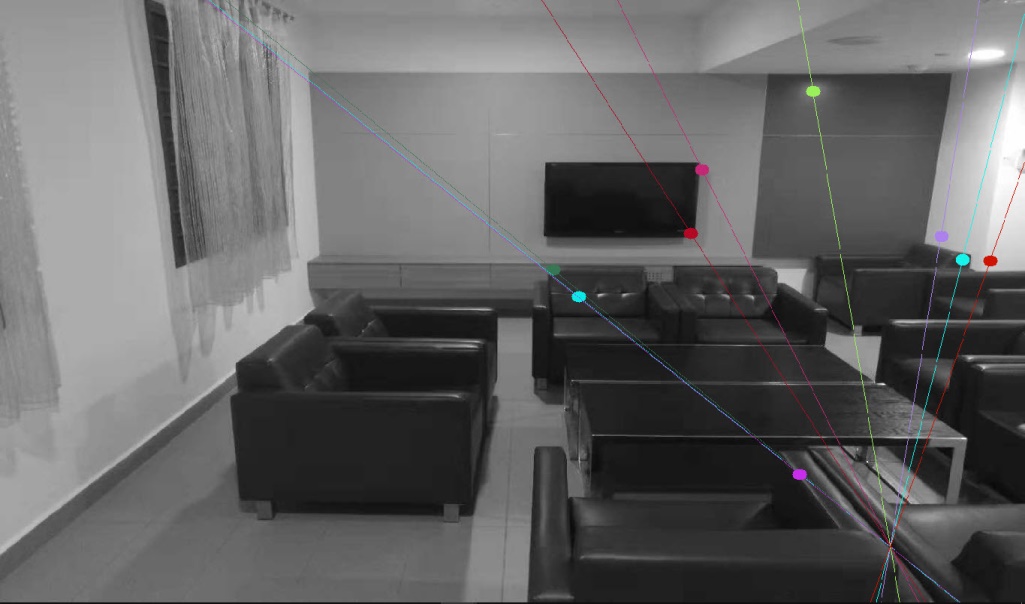
[[ 4.19791486e-07, -2.98208562e-06, 2.39008664e-03]

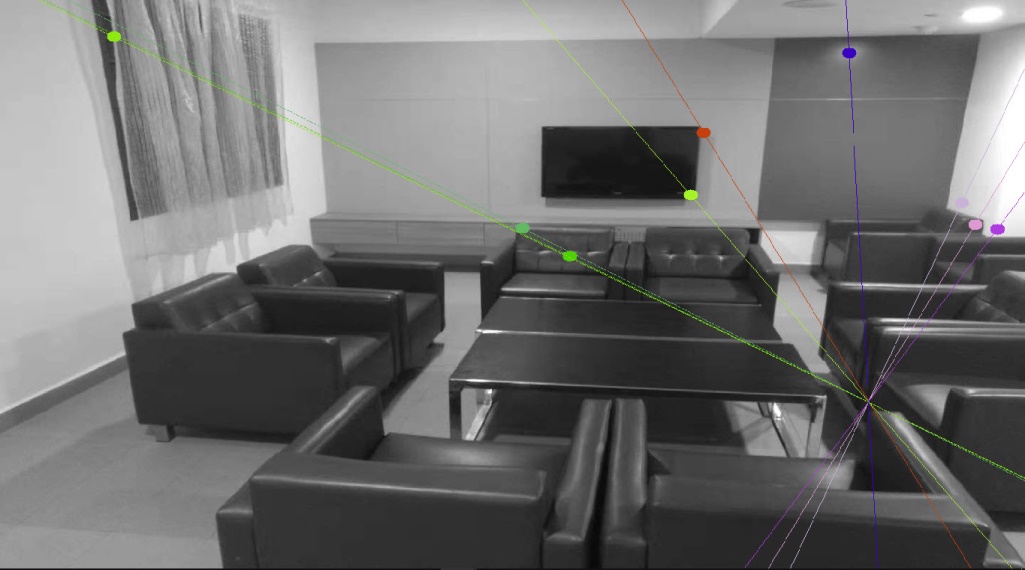
[ 4.48565311e-06, -4.76079415e-07, -5.14304123e-03]

[-3.92803358e-03, 4.00129186e-03, 1.00000000e+00]]

2.2 For every , we can find corresponding and draw each pair of them as a line. That is our left and right eight epipolar line.

Result epipolar line poltted on the page follows:

Left epipolar line

Right epipolar line

This part’s codes is as follows:

import numpy as np

import cv2 as cv

img1 = cv.imread('./LEFT2.jpg',0)

img2 = cv.imread('./RIGHT2.jpg',0)

sift = cv.xfeatures2d.SIFT\_create()

kp1, des1 = sift.detectAndCompute(img1,None)

kp2, des2 = sift.detectAndCompute(img2,None)

FLANN\_INDEX\_KDTREE = 1

index\_params = dict(algorithm = FLANN\_INDEX\_KDTREE, trees = 5)

search\_params = dict(checks=50)

flann = cv.FlannBasedMatcher(index\_params,search\_params)

matches = flann.knnMatch(des1,des2,k=2)

good = []; pts1 = []; pts2 = []

for i,(m,n) in enumerate(matches):

if m.distance < 0.62\*n.distance:

good.append(m)

pts2.append(kp2[m.trainIdx].pt)

pts1.append(kp1[m.queryIdx].pt)

pts1 = np.int32(pts1)

pts2 = np.int32(pts2)

F, mask = cv.findFundamentalMat(pts1,pts2,cv.FM\_LMEDS)

print(F)

pts1 = pts1[mask.ravel()==1]

pts2 = pts2[mask.ravel()==1]

print(pts1)

print(pts2)

def drawlines(img1,img2,lines,pts1,pts2):

r,c = img1.shape

img1 = cv.cvtColor(img1,cv.COLOR\_GRAY2BGR)

img2 = cv.cvtColor(img2,cv.COLOR\_GRAY2BGR)

for r,pt1,pt2 in zip(lines,pts1,pts2):

color = tuple(np.random.randint(0,255,3).tolist())

x0,y0 = map(int, [0, -r[2]/r[1] ])

x1,y1 = map(int, [c, -(r[2]+r[0]\*c)/r[1] ])

img1 = cv.line(img1, (x0,y0), (x1,y1), color,1)

img1 = cv.circle(img1,tuple(pt1),10,color,-1)

img2 = cv.circle(img2,tuple(pt2),10,color,-1)

return img1,img2

#left

lines1 = cv.computeCorrespondEpilines(pts2.reshape(-1,1,2), 2,F)

lines1 = lines1.reshape(-1,3)

img5,img6 = drawlines(img1,img2,lines1,pts1,pts2)

#right

lines2 = cv.computeCorrespondEpilines(pts1.reshape(-1,1,2), 1,F)

lines2 = lines2.reshape(-1,3)

img3,img4 = drawlines(img2,img1,lines2,pts2,pts1)

cv.namedWindow('rightline',cv.WINDOW\_NORMAL); cv.imshow('rightline', img3)

cv.namedWindow('left',cv.WINDOW\_NORMAL); cv.imshow('left', img4)

cv.namedWindow('leftline',cv.WINDOW\_NORMAL); cv.imshow('leftline', img5)

cv.namedWindow('right',cv.WINDOW\_NORMAL); cv.imshow('right', img6)

cv.waitKey(0)