CS512 - Assignment 4

Mohammadreza Asherloo
Department of Mechanical, Materials and Aerospace Engineering
Illinois Institute of Technology

December 1, 2020

Topic:

Camera calibration

Problem statement:

Calibrating the camera using non-planar method

Approach:

We used non-planar calibration method to find the camera intrinsic and extrinsic parameters.

Details:

First we need to figure out whether the user wants to do a manual point picking or using the test point file provided. For this purpose we will check the argument passed to the program by command line:

```
if len(sys.argv) == 2:
    filename = sys.argv[1]
    if filename == 'test_points.txt':
        pass
    else:
```

If the file name is not "test_points.txt", the program will open the test image and lets the user to pick the points on image manually. For this purpose, I set 12 world points as follows by real world measurement of the box:

```
objpoints = [[0,0,0],[5,0,0],[10,0,0],[10,0,3],
             [10,5,3],[10,10,3],[5,10,3],[0,10,3],
             [0,5,3],[0,0,3],[0,5,0],[0,10,0]]
```

After user chooses the points based on the real world points, key q on keyboard must be pressed to save the points and start the calibration:

```
imgpoints = []
cv2.namedWindow('img')
mouse_point = []
img = cv2.imread('test.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
cv2.imshow('img',img)
cv2.setMouseCallback("img", mouse_click)
h = cv2.waitKey(0)
if h == ord('q'):
    with open("points.txt", 'w') as file:
        for i in range(len(mouse_point)):
            s = " ".join(map(str, objpoints[i])) + " " +
                " ".join(map(str, mouse_point[i]))[1:-1]
            file.write(s+'\n')
            cv2.destroyAllWindows()
```

After pressing the q key, points will be saved in a file named "points.txt" in the root folder by the 3D-2D format. Then this file will be opened for the following calibration process. Calibration process starts with reading the corresponding points from either "test_points.txt" file or "points.txt" file based on user preference:

```
points = pd.read_csv("test_points.txt", delimiter='\s+',
                     header = None, dtype = float)
```

After reading the file, world points and image points should be divided:

```
objpoints_file = points[[0, 1, 2]]
imgpoints_file = points[[3, 4]]
```

Then the matrix A should be created to be solved to find the camera parameters:

```
A = np.zeros((objpoints_file.shape[0]*2, 12))
j = 0
for i in range(0, objpoints_file.shape[0]*2, 2):
    A[i][0] = objpoints_file[0][j]
    A[i][1] = objpoints_file[1][j]
    A[i][2] = objpoints_file[2][j]
    A[i][3] = 1
    A[i][8] = objpoints_file[0][j] * -1 * imgpoints_file[3][j]
    A[i][9] = objpoints_file[1][j] * -1 * imgpoints_file[3][j]
    A[i][10] = objpoints_file[2][j] * -1 * imgpoints_file[3][j]
    A[i][11] = 1 * -1 * imgpoints_file[3][j]
    A[i+1][4] = objpoints_file[0][j]
    A[i+1][5] = objpoints_file[1][j]
    A[i+1][6] = objpoints_file[2][j]
    A[i+1][7] = 1
```

```
        A[i+1][8] = objpoints_file[0][j] * -1 * imgpoints_file[4][j]
        A[i+1][9] = objpoints_file[1][j] * -1 * imgpoints_file[4][j]
        A[i+1][10] = objpoints_file[2][j] * -1 * imgpoints_file[4][j]
        A[i+1][11] = 1 * -1 * imgpoints_file[4][j]
        j += 1
```

After filling the matrix A, we will solve the $Ax = 0$ equation to find the x using Singular Value Deposition technique and choose the column of V which is corresponds to zero singular value in D. Then we will arange the x vector to find the projection matrix M:

```
u, d, v = np.linalg.svd(A)
x = np.transpose(v)[:,-1]
M = x.reshape(3,4)
```

After that we will use the predefined equations to extract the camera parameters from the projection matrix:

```
a1 = M[0][0:3].reshape(3,1)
a2 = M[1][0:3].reshape(3,1)
a3 = M[2][0:3].reshape(3,1)
b = M[:, -1]

rho_abs = 1/(np.sqrt(a3[0]**2+a3[1]**2+a3[2]**2))

U0 = rho_abs**2 *
(np.dot(np.squeeze(np.asarray(a1)),np.squeeze(np.asarray(a3))))

V0 = rho_abs**2 *
(np.dot(np.squeeze(np.asarray(a2)),np.squeeze(np.asarray(a3))))

alpha_v = np.sqrt((rho_abs**2 *
(np.dot(np.squeeze(np.asarray(a2)),np.squeeze(np.asarray(a2))))) - V0**2)

s = (1/alpha_v) * rho_abs**4 *
(np.dot(np.squeeze(np.asarray(np.cross(a1,a3,axis=0))),
        np.squeeze(np.asarray(np.cross(a2,a3,axis=0)))))

rho_sign = np.sign(b[2])

alpha_u = np.sqrt((rho_abs**2 *
(np.dot(np.squeeze(np.asarray(a1)),
        np.squeeze(np.asarray(a1))))) - s**2 - U0**2)

kstar = np.zeros((3,3))
kstar[0][0] = alpha_u
kstar[1][1] = alpha_v
kstar[0][1] = s
```

```
kstar[0][2] = U0
kstar[1][2] = V0
kstar[2][2] = 1

Tstar = rho_sign * rho_abs * (np.linalg.inv(kstar) @ b)
r3 = rho_sign * rho_abs * a3
r1 = rho_abs**2 / alpha_v * np.cross(a2,a3,axis=0)
r2 = np.cross(r3, r1, axis = 0)
Rstar = np.concatenate([r1,r2,r3]).T.reshape(3,3)

M_computed = kstar @ np.concatenate([Rstar, Tstar.reshape(3,1)], axis = 1)
```

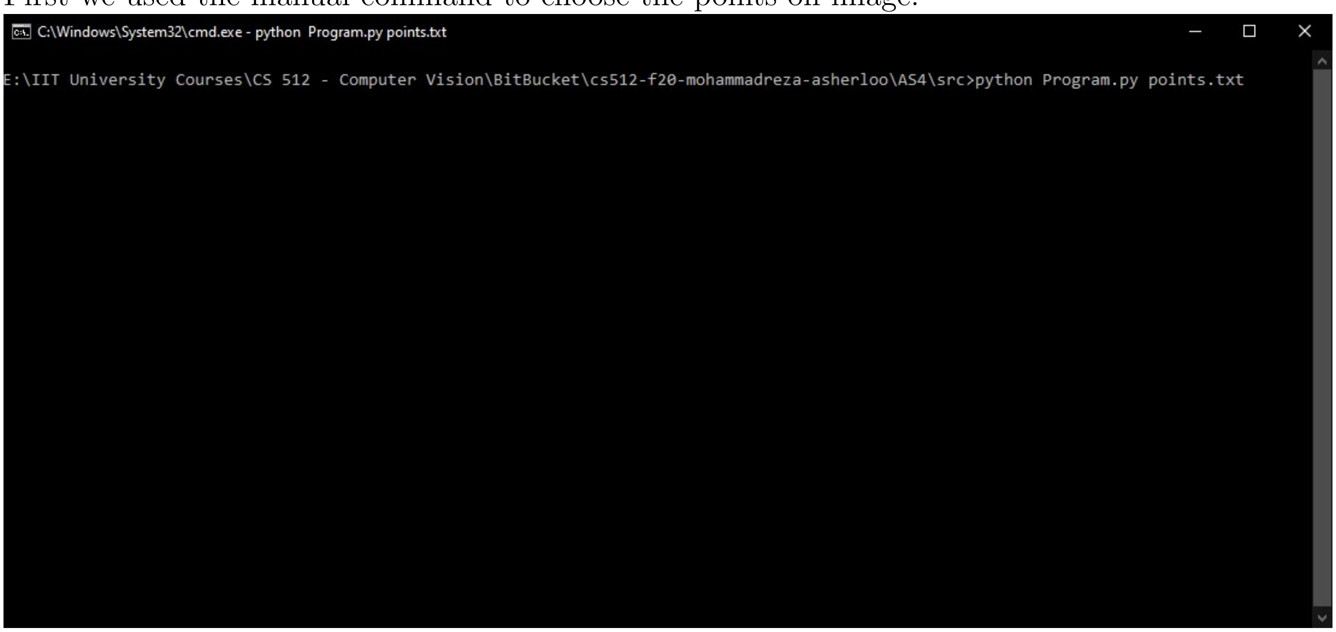
Then we will predict the image points using the computed projection matrix and will find the error by using the true data:

```
predicted_points = imgpoints_file.copy()
predicted_points[2] = 1
objpoints_file[3] = 1
for i in range(predicted_points.shape[0]):
    predict = M_computed @ np.array(objpoints_file.iloc[[i]]).reshape(4,1)
    predicted_points.iloc[[i]] = predict.reshape(1,3) / predict[2]

predicted_points = predicted_points.drop([2], axis = 1)
rms = mean_squared_error(imgpoints_file, predicted_points)
```

Results:

First we used the manual command to choose the points on image:



Then we chose the points on the image as follows:

 $\longrightarrow$ 

The results of calibration are as follows:

```
C:\Windows\System32\cmd.exe

E:\IIT University Courses\CS 512 - Computer Vision\BitBucket\cs512-f20-mohammadreza-asherloo\AS4\src>python Program.py points.txt
T* = [-6.21044444  2.12360951  8.7676611 ]

R* = [[ 0.62077776  0.33604142 -0.7083157 ]
 [ 0.32846566 -0.93184118 -0.15421521]
 [-0.71186043 -0.13692401 -0.68884435]]

K* = [[29.23516389 -2.81458189 67.47710116]
 [ 0.          17.4296934  76.10224457]
 [ 0.           0.           1.        ]]

(u0, v0) = (67.48, 76.10)

(alpha_u, alpha_v) = (29.24, 17.43)

MSE = 2660.949423665177619114

E:\IIT University Courses\CS 512 - Computer Vision\BitBucket\cs512-f20-mohammadreza-asherloo\AS4\src>
```

And for testing the calibration algorithm, the test command was passed to cmd:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.329]
(c) 2020 Microsoft Corporation. All rights reserved.

E:\IIT University Courses\CS 512 - Computer Vision\BitBucket\cs512-f20-mohammadreza-asherloo\AS4\src>python Program.py test_points.txt
T* = [-2.57709099e-04  3.26857535e-05  1.04880905e+03]

R* = [[-7.68221190e-01  6.40184508e-01  1.46341836e-07]
 [ 4.27274298e-01  5.12729182e-01 -7.44678091e-01]
 [-4.76731452e-01 -5.72077427e-01 -6.67423808e-01]]

K* = [[ 6.52174069e+02 -3.39862304e-05  3.20000170e+02]
 [ 0.00000000e+00  6.52174075e+02  2.39999971e+02]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e+00]]

(u0, v0) = (320.00, 240.00)

(alpha_u, alpha_v) = (652.17, 652.17)

MSE = 0.000000000830270531

E:\IIT University Courses\CS 512 - Computer Vision\BitBucket\cs512-f20-mohammadreza-asherloo\AS4\src>
```

It can be seen from the results that the algorithm is doing a good job of estimating the projection matrix and camera parameters. the MSE is so small that can be ignore