Калибровка камеры с помощью Python - OpenCV

Предварительные требования: OpenCV

Камера является неотъемлемой частью нескольких областей, таких как робототехника, исследование космоса и т. Д. Камера играет важную роль. Это помогает запечатлеть каждый момент и полезно для многих анализов. Чтобы использовать камеру в качестве визуального датчика, мы должны знать параметры камеры. **Калибровка камеры** - это не что иное, как оценка параметров камеры, параметры камеры необходимы для определения точной взаимосвязи между трехмерной точкой в реальном мире и ее соответствующей двухмерной проекцией (пикселем) в изображении, захваченном этой откалиброванной камерой.

Нам необходимо учитывать как внутренние параметры, такие как фокусное расстояние, оптический центр, коэффициенты радиального искажения объектива и т. Д., Так и внешние параметры, такие как вращение и перемещение камеры относительно некоторой системы координат реального мира.

Необходимые библиотеки:

- OpenCV Библиотека в Python - это библиотека компьютерного зрения, которая в основном используется для обработки изображений, обработки и анализа видео, распознавания и обнаружения лиц и т. д.
- Numpy - это универсальный пакет для обработки массивов. Он предоставляет высокопроизводительный объект многомерного массива и инструменты для работы с этими массивами.

Калибровку камеры можно выполнить поэтапно:

- **Шаг 1:** Сначала определите реальные координаты трехмерных точек, используя известный размер шахматной доски.
- **Шаг 2:** Захвачены различные точки зрения изображения шахматной доски.
- **Шаг 3:** *findChessboardCorners ()* - это метод в *OpenCV, который* используется для нахождения пиксельных координат *(u, v)* для каждой трехмерной точки на разных изображениях.
- **Шаг 4:** Затем *метод calibrateCamera ()* используется для поиска параметров камеры.

Он принимает наши вычисленные *(три точки, две точки, grayColor.shape [:: - 1], None, None) в* качестве параметров и возвращает список, содержащий такие элементы, как *матрица камеры, коэффициент искажения, векторы вращения* и *векторы перемещения.*

Камера Матрица помогает трансформировать объекты 3D очки для 2D точек изображения и *искажение Коэффициент* возвращает положение камеры в мире, со значениями векторов *вращения* и *перевода*

Ниже представлена полная программа описанного выше подхода:

Python3

```
# Import required modules
import cv2
import numpy as np
import os
import glob


# Define the dimensions of checkerboard
CHECKERBOARD = (6, 9)


# stop the iteration when specified
# accuracy, epsilon, is reached or
# specified number of iterations are completed.
criteria = (cv2.TERM_CRITERIA_EPS +
            cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)


# Vector for 3D points
threedpoints = []

# Vector for 2D points
twodpoints = []


#  3D points real world coordinates
objectp3d = np.zeros((1, CHECKERBOARD[0]
                      * CHECKERBOARD[1],
                      3), np.float32)
objectp3d[0, :, :2] = np.mgrid[0:CHECKERBOARD[0],
                               0:CHECKERBOARD[1]].T.reshape(-1,  2)
prev_img_shape = None


# Extracting path of individual image stored
# in a given directory. Since no path is
# specified, it will take current directory
# jpg files alone
images = glob.glob("*.jpg")

for filename in images:
    image = cv2.imread(filename)
    grayColor = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Find the chess board corners
    # If desired number of corners are
    # found in the image then ret = true
```

```
    ret, corners = cv2.findChessboardCorners(
                    grayColor, CHECKERBOARD,
                    cv2.CALIB_CB_ADAPTIVE_THRESH
                    + cv2.CALIB_CB_FAST_CHECK +
                    cv2.CALIB_CB_NORMALIZE_IMAGE)

    # If desired number of corners can be detected then,
    # refine the pixel coordinates and display
    # them on the images of checker board
    if ret == True:
        threedpoints.append(objectp3d)

        # Refining pixel coordinates
        # for given 2d points.
        corners2 = cv2.cornerSubPix(
            grayColor, corners, (11, 11), (-1, -1), criteria)

        twodpoints.append(corners2)

        # Draw and display the corners
        image = cv2.drawChessboardCorners(image,
                                          CHECKERBOARD,
                                          corners2, ret)

    cv2.imshow("img", image)
    cv2.waitKey(0)

cv2.destroyAllWindows()

h, w = image.shape[:2]


# Perform camera calibration by
# passing the value of above found out 3D points (threedpoints)
# and its corresponding pixel coordinates of the
# detected corners (twodpoints)
ret, matrix, distortion, r_vecs, t_vecs = cv2.calibrateCamera(
    threedpoints, twodpoints, grayColor.shape[::-1], None, None)


# Displayig required output
print(" Camera matrix:")
print(matrix)

print(" Distortion coefficient:")
print(distortion)

print(" Rotation Vectors:")
print(r_vecs)

print(" Translation Vectors:")
```

print(t_vecs)

Вход:

Выход:

Матрица камеры:

[[36.26378216 0. 125.68539168]
[0. 36.76607372 142.49821147]
[0. 0. 1.]]

Коэффициент искажения:

[[-1.25491812e-03 9.89269357e-05 -2.89077718e-03 4.52760939e-04
-3.29964245e-06]]

Векторы вращения:

[массив ([[- 0,05767492],
[0,03549497],
[1.50906953]]), массив ([[- 0.09301982],
[-0.01034321],
[3.07733805]]), массив ([[- 0.02175332],
[0,05611105],
[-0.07308161]])]

Векторы перевода:

[массив ([[4.63047351],
[-3,74281386],
[1.64238108]]), массив ([[2.31648737],
[3.98801521],
[1.64584622]]), массив ([[- 3.17548808],

```
[-3,46022466],
[1.68200157]])]
```

Задание:

1. Выполнить калибровку камеры.
2. Используя найденные параметры, следует исправить исходного изображения.
3. Получить исправленное изображение путем репроектирования.