相机标定章节报告

相机标定基础知识

相机标定（Camera Calibration）是指确定从三维世界到图像平面的投影变换过程中的各种参数，包括：

1. 内参（Intrinsic Parameters）

描述相机内部成像过程的参数：

- 焦距 f_x, f_y（单位通常是像素）
- 主点坐标 c_x, c_y（图像中心）
- 畸变参数（径向、切向畸变）

内参矩阵通常表示为：

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

2. 外参（Extrinsic Parameters）

描述世界坐标系到相机坐标系的变换：

- 旋转矩阵 R
- 平移向量 t

组成一个外参矩阵：

$$[R \mid t]$$

3. 投影关系

将三维点 $\mathbf{X}_w$ 投影到图像点 $\mathbf{x}$ 的数学表达为：

$$\mathbf{x} \sim K[R \mid t]\mathbf{X}_w$$

张正友标定法的数学原理

张正友标定法是一种**基于平面模板**的标定方法，其主要优点是无需已知三维物体坐标，只需要已知模板平面几何关系即可。

基本思想：

通过拍摄多个视角下的棋盘格（或其他已知结构的平面图案）图像，利用图像中角点与实际模板点的对应关系，恢复相机的内外参数和畸变参数。

数学原理步骤：

1：求解单应矩阵（Homography）

由于所有点在一个平面（通常 $Z = 0$），世界坐标与图像坐标之间存在单应变换：

$$s \cdot \mathbf{x} = H \cdot \mathbf{X}$$

其中：

- $\mathbf{x} = [u, v, 1]^T$：图像坐标
- $\mathbf{X} = [X, Y, 1]^T$：世界坐标（模板上的平面坐标）
- $H \in \mathbb{R}^{3\times3}$：单应矩阵
- s：尺度因子

2：利用单应矩阵求内参矩阵 KK

从多个视角下拍摄图像，估计多个单应矩阵 HiH_i，根据下列关系构造方程：

$$H_i = K[r_1^{(i)}\ r_2^{(i)}\ t^{(i)}]$$

利用 H_i 的列向量与 K 的约束，构造线性方程组，解出 K。

3：求解外参 R, t

对每幅图像，用已知 K 和 H_i 反解出：

$$r_1 = K^{-1}h_1, \quad r_2 = K^{-1}h_2, \quad r_3 = r_1 \times r_2, \quad t = K^{-1}h_3$$

并对 r_1, r_2, r_3 做正交归一化处理构成旋转矩阵 RR。

张正友法实践结果

非常幸运地，Python所调用的OpenCV2库之相机标定法本身就是张正友标定法。

[318] Qi Zhang, Li Xu, and Jiaya Jia. 100+ times faster weighted median filter (wmf). In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2830–2837. IEEE, 2014.

[319] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.

[320] Wenyi Zhao, Rama Chellappa, P Jonathon Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. *Acm Computing Surveys (CSUR)*, 35(4):399–458, 2003.

因此，在实践中，只需要创建好棋盘点的矩阵：

```
chessboard_size = (8, 6)
square_size = 30 # mm

objp = np.zeros((chessboard_size[0]*chessboard_size[1], 3), np.float32)

objp[:, :2] = np.mgrid[0:chessboard_size[0],
0:chessboard_size[1]].T.reshape(-1, 2)

objp *= square_size
objpoints = []
imgpoints = []
```

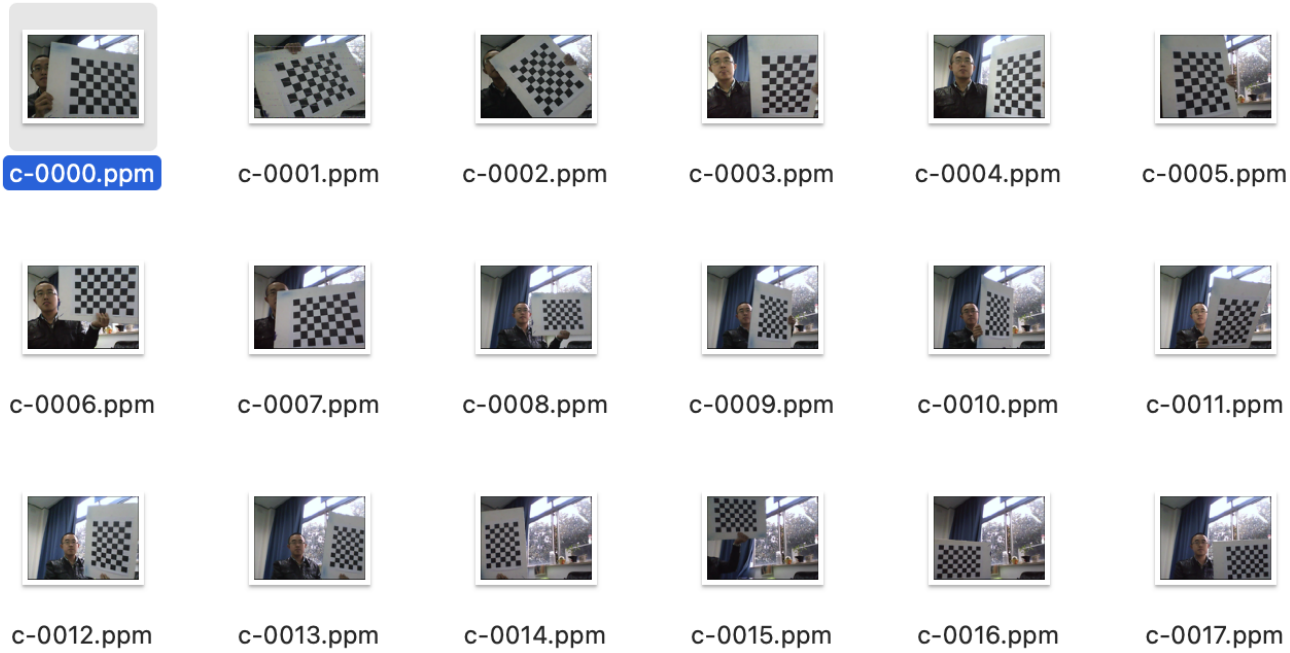
判断是否可以找到角点：

```
img = cv2.imread(fname)

if img is None:
    print(f"无法读取图像: {fname}")
    continue

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, corners = cv2.findChessboardCorners(gray, chessboard_size, None)

print(f"{os.path.basename(fname)}: 找到角点?  {ret}")

if ret:
    objpoints.append(objp)
    corners2 = cv2.cornerSubPix(gray, corners, (11, 11), (-1, -1),
    (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001))
    imgpoints.append(corners2)
```

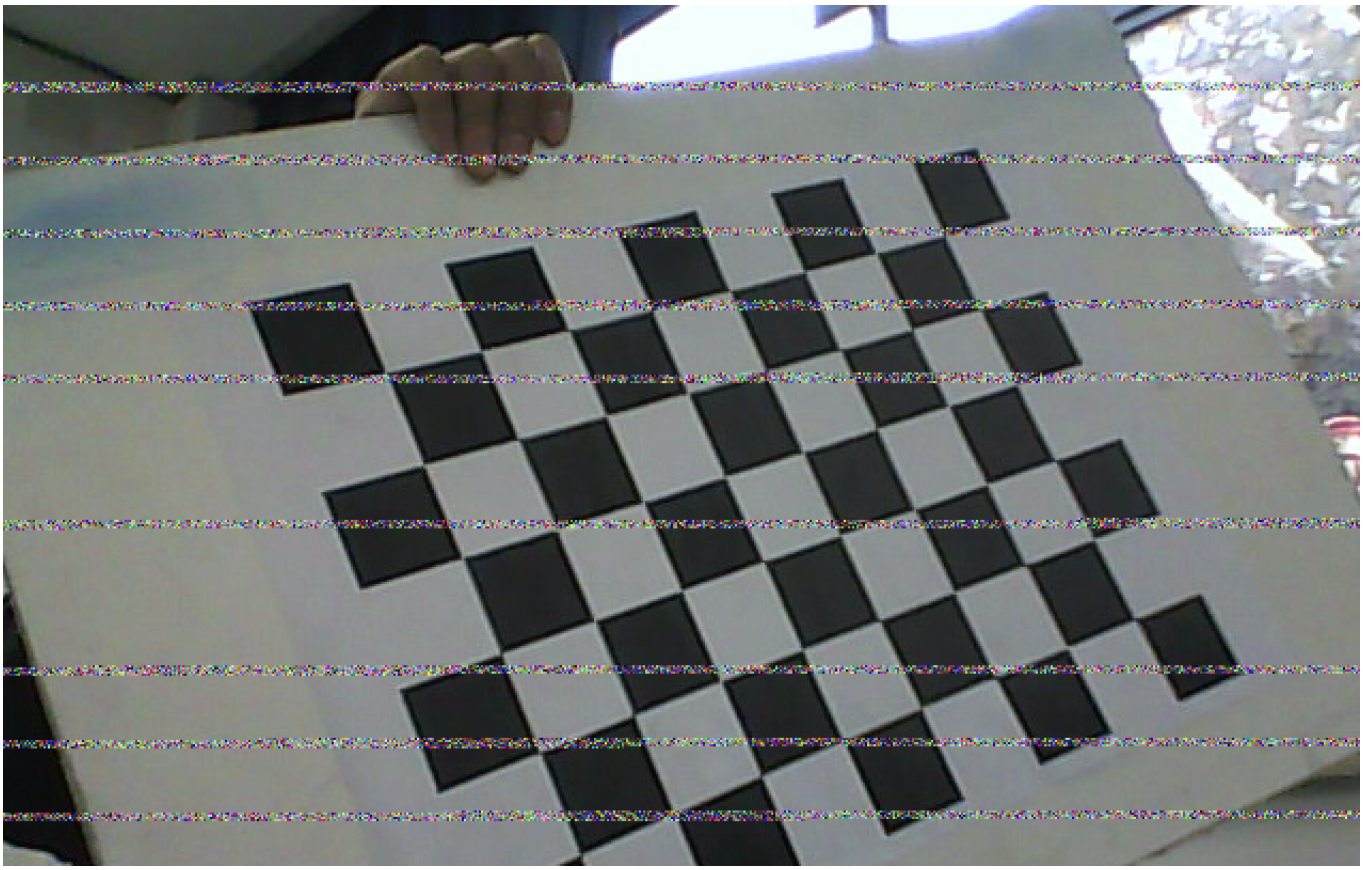
最后利用OpenCV2库的魔法即可：

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

张正友贴心地给我们准备了27个棋盘素材以使用，来检测该标定算法的效果。



然而在实践中，我们发现，c-0001因具有严重的彩色条纹行噪声干扰，识别失败。这显然是相机的cmos阵列受到了强烈干扰导致的。不过，因为26张照片已经具有非常好的统计意义，因此直接剔除它，不作处理。



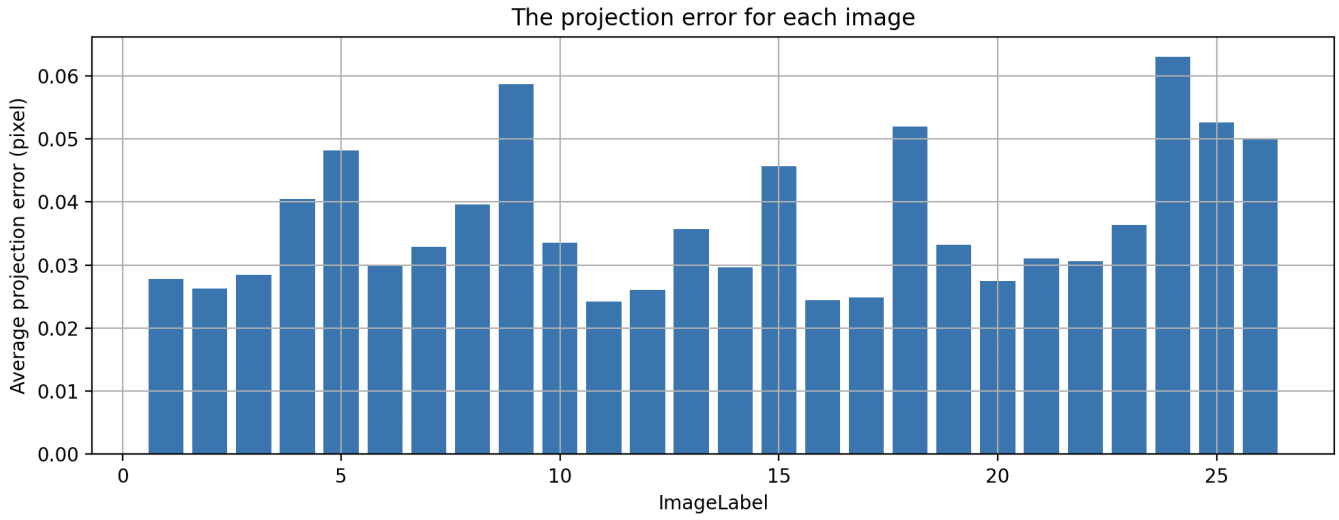
得到相机内参矩阵及其畸变系数。

```
相机内参矩阵：
 [[687.05251338   0.         338.79697095]
 [  0.         687.58890578 276.34857814]
 [  0.           0.           1.        ]]
畸变系数：
 [ 0.3772124  -1.34301311  0.00238067  0.00213362  1.35514045]
```

同时，因为格子的大小已经给定，也可以照此得到各个图片的外参。这里只展示其中的一张：

```
--- 图像 2 的外参 ---
旋转向量 rvec:
 [-0.20092419 -0.15860762 -0.03919194]
旋转矩阵 R:
 [[ 0.98672823  0.05460056 -0.15292543]
 [-0.02291015  0.97916357  0.20177669]
 [ 0.16075613 -0.19559521  0.96741924]]
平移向量 t:
 [ 51.52705149 -67.71146826 832.80060888]
```

26张图像的重投影误差（以范数计，当然也可以画成一个三维图像，但是在这份报告里显得很丑）如下：



误差非常非常小（甚至可以怀疑是否为固有的截断误差引入）。因此，该算法成功完成了相机标定的任务。