计算摄像学作业报告（相机标定）

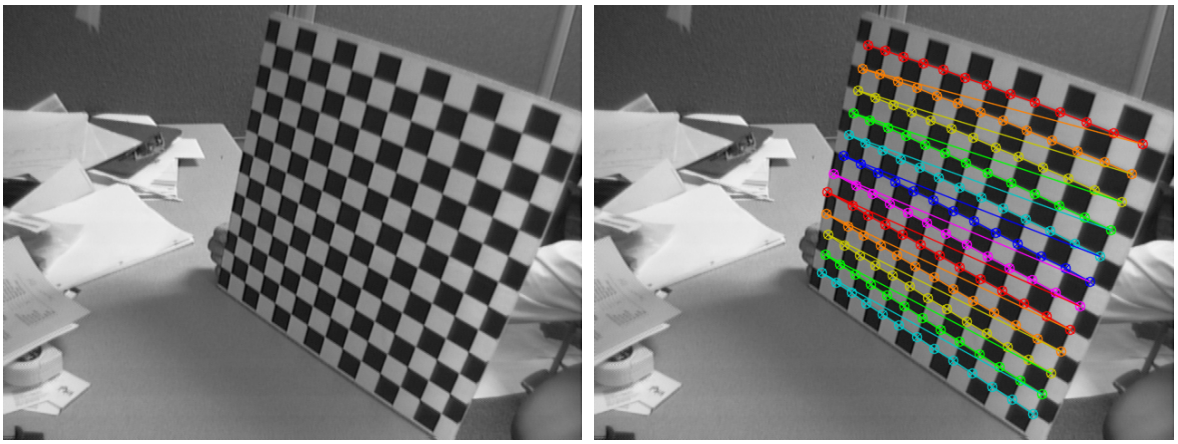
信息科学技术学院
陈思硕
1700012765
chensishuo@pku.edu.cn

2021 年 4 月 14 日

1 任务一: 基于 opencv 的经典相机标定

1.1 实现经典相机标定

利用 opencv-python 库 [2]，使用附件 chess-boardexample.zip 中的 20 组黑白棋盘格数据进行测试。根据对图像的观察，棋盘网格大小参数设为 13*12，参考 opencv 官方文档提供的相机标定样例[1]，修改代码进行试验。20 张样例图片中，有 5 张图像检测到棋盘网格角点。一个示例如下：



(a) 原图　　(b) 标出棋盘角点

图 1: 棋盘图像角点的检测

基于检测到的棋盘网格角点，调用 opencv 的 cv2.calibr-ateCamera() 函数进行标定，得到拍照所用相机的内参与畸变系数如下。

内参矩阵：

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 656.86 & 0 & 301.62 \\ 0 & 661.74 & 230.66 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

畸变系数：

$$\begin{pmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{pmatrix} = \begin{pmatrix} -3.37e^{-1} & 7.67e^{-1} & 1.48e^{-3} & 2.13e^{-4} & -1.38e^{0} \end{pmatrix} \tag{2}$$

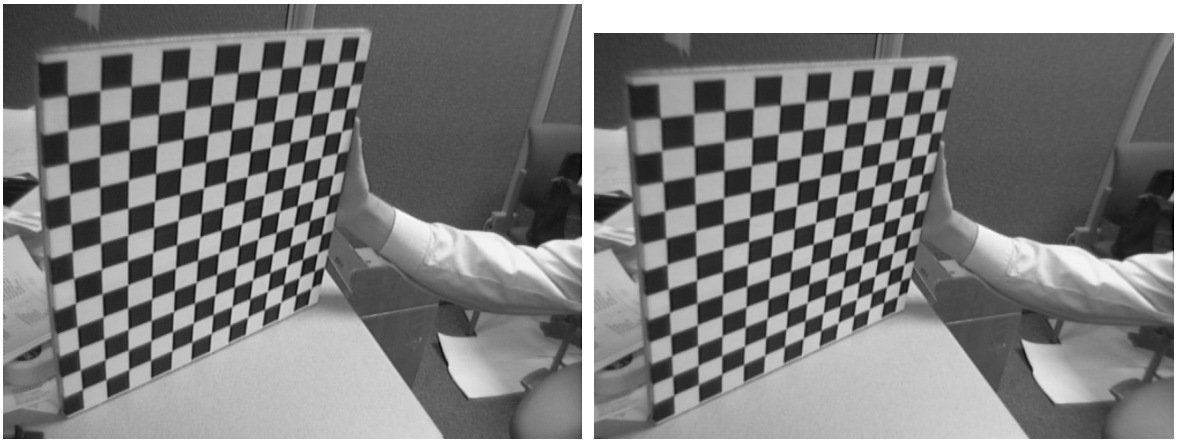
[1] https://docs.opencv.org/4.5.1/dc/dbb/tutorial_py_calibration.html

五张照片对应的旋转向量、平移向量、旋转矩阵请在日志文件**part1.log**中查看，以下为截图：

```
rvecs:
[array([[0.59390995],
       [0.22657808],
       [0.18081543]]), array([[0.28112282],
       [0.4834412 ],
       [0.30315035]]), array([[0.3162188 ],
       [0.68118286],
       [0.34240767]]), array([[ 0.54931466],
       [-0.9375824 ],
       [-0.14925678]]), array([[0.4984649 ],
       [0.01754041],
       [0.13679762]])]
tvecs:
[array([[-4.47522915],
       [-4.31306134],
       [16.31782323]]), array([[-0.93976627],
       [-9.4864958 ],
       [28.73859908]]), array([[-2.26788594e-02],
       [-7.07046413e+00],
       [ 2.43719851e+01]]), array([[-4.81591272],
       [-2.67854675],
       [13.27637618]]), array([[-4.98211728],
       [-7.36926442],
       [23.43606699]])]
rotation_matrixs:
[(array([[ 0.95949123, -0.10306775,  0.26220912],
       [ 0.23280768,  0.81420113, -0.53186193],
       [-0.15867315,  0.57136115,  0.8052138 ]]), array([[ 0.00403926,  0.13701366,  0.03907591,  0.06850008, -0.55407924,
        -0.81822853,  0.12492961,  0.81428994, -0.55318309],
       [-0.21690952,  0.29690426,  0.91043376,  0.27076616,  0.00706794,
         0.12934027, -0.91437235,  0.04348656, -0.21104068],
       [-0.17309966, -0.9203196 ,  0.27166231,  0.91638101, -0.16868901,
```

图 2: 外参结果截图

利用标定得到的相机参数可进行图像去畸变，选取测试数据中的 Image8.tif 进行测试，效果如下图所示。对比矫正前后的图像可以看出，棋盘网格线弯曲的畸变现象得到消除，去畸变的图像更符合真实观感。

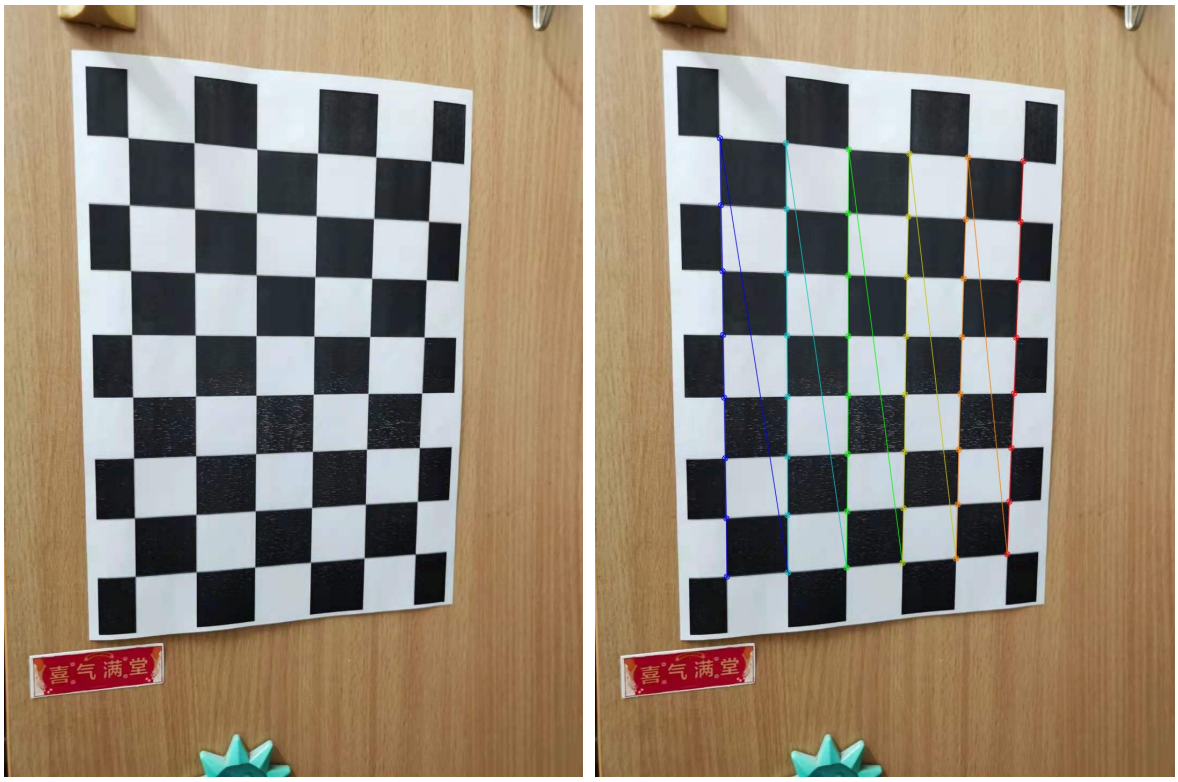


(a) 原图　　(b) 去畸变后

图 3: 图像去畸变的示例

1.2 自拍棋盘格图像进行标定

打印棋盘图像，使用华为 Mate 30 拍摄 20 张图像，按1.1中相同的步骤进行标定。棋盘格大小参数设为 8*6，20 张图像均成功检测到角点，一个例子如下：



(a) 原图 (b) 标出棋盘角点

图 4: 手机拍摄棋盘图像角点的检测

基于检测到的棋盘网格角点，调用 opencv 的 cv2.calibrateCamera() 函数进行标定，得到拍照所用相机的内参与畸变系数如下。

内参矩阵：

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 858.79 & 0 & 418.95 \\ 0 & 847.56 & 809.56 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

畸变系数：

$$\begin{pmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{pmatrix} = \begin{pmatrix} -0.00863484 & -0.00262332 & 0.0230366 & 0.01958648 & 0.00028345 \end{pmatrix} \tag{4}$$

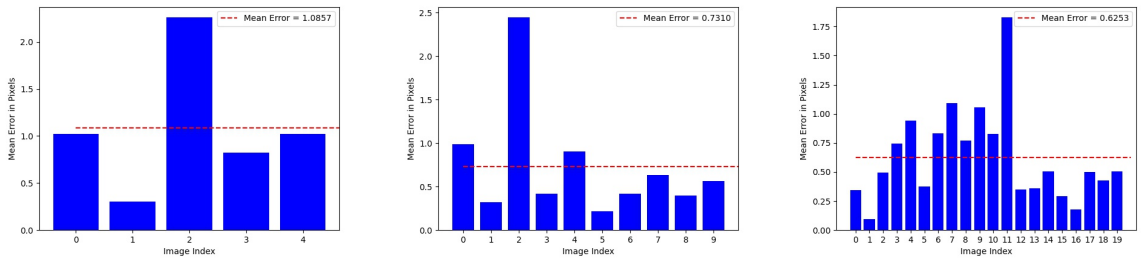
20 张照片对应的旋转向量、平移向量、旋转矩阵请在日志文件**part2.log**中查看，以下为截图：

```
 [ 0.          0.          1.        ]]
distCoeffs:
[[-0.00863484 -0.00262332  0.0230366   0.01958648  0.00028345]]
rvecs:
> [array([[-0.04634063], ···
tvecs:
> [array([[ 3.3394935 ], ···
rotation_matrixs:
[(array([[ 0.00654497, -0.97831095, -0.20703827],
        [ 0.98879668,  0.03720747, -0.14455699],
        [ 0.14912506, -0.20377264,  0.96759415]]), array([[-0.00800561, -0.13153653,  0.62129205, -0.0946541 ,  0.02986194,
         -0.63976548,  0.62796996,  0.63695852,  0.03735932],
        [ 0.17812391, -0.12947418,  0.61743089,  0.09236195, -0.04666505,
          0.61976181, -0.62023786,  0.6130839 ,  0.22470447],
        [-0.98378716, -0.02771467,  0.09985934,  0.0249077 , -0.99199531,
         -0.08495598, -0.1219768 , -0.04807355,  0.00867484]])), (array([[ 0.01120447, -0.96497448, -0.26210438],
        [ 0.9887851 ,  0.04973076, -0.14082215],
        [ 0.14892443, -0.25758706,  0.9547096 ]]), array([[-0.01568865, -0.16632066,  0.61166246, -0.09423109,  0.05904739,
         -0.64079288,  0.62682855,  0.63447155,  0.07340617],
        [ 0.2057033 , -0.16328012,  0.60993236,  0.09048522, -0.05307447,
          0.61660045, -0.6162537 ,  0.60143436,  0.25839998],
        [-0.97777767, -0.03983064,  0.104844  ,  0.03350931, -0.98800578,
         -0.11362398, -0.14892133, -0.04153441,  0.01202386]])), (array([[ 0.03705702, -0.92978629, -0.36622976],
        [ 0.99553985,  0.00253107,  0.09430796],
        [-0.08675929, -0.36809109,  0.9257331 ]]), array([[-6.15734870e-02, -2.31448826e-01,  5.81373334e-01,
          5.66012876e-02,  2.32470103e-01, -6.03737345e-01,
          6.23185211e-01,  5.86230823e-01,  2.91502433e-01],
        [ 1.41837473e-01, -2.33902711e-01,  6.08185443e-01,
         -5.98074154e-02, -3.85488598e-02,  6.32377552e-01,
         -6.25691964e-01,  5.90565675e-01,  1.76181851e-01],
        [-9.77145817e-01, -3.86392544e-02, -7.75085244e-04,
          2.11327328e-02, -9.67953476e-01, -1.97104448e-01,
```


图 5: 外参结果截图

1.3 误差分析

分别使用 5 张、10 张、20 张棋盘格图像进行标定，计算重投影误差。各组实验中重投影误差的分布如图 6,7,8 所示：



(a) n=5 (b) n=10 (c) n=20

图 6: 重投影误差的分布

平均误差随样本数量的变化如下表所示：

n	Mean Error
5	1.0957
10	0.7310
20	0.6253

可以明显看出，用于标定的图像越多，平均的重投影误差越小，标定的效果越好。

(a) 没有插值操作时，有原棋盘图像的残影　　(b) 插值操作后，实现正常的投影效果

图 7: 插值操作前后对比

1.4 增强现实

1.4.1 思路简述

将需要投射的图像 resize 为和 chessboard.jpg 同样大小，将上面的点与 chessboard.jpg 的像素点一一对应，利用之前标定得到的相机参数，使用 cv2.projectPoints 得到在对应目标图像上相应的像素点位置，将对应位置各通道的像素值改为需要投射的图像所对应位置的像素值。一个需要注意的细节是，投射后得到的点位置为浮点值，若用 round 取整之后修改对应的像素点位置，目标图像上会有相当部分的点没有被覆盖到 (目标图像的尺寸比需要投射的图像素材大)，留下棋盘的残影，需要通过类似插值的方式填补。我实现的简单操作是每次投射点时，将横纵坐标差不小于 2 的周围像素点都赋为素材图像对应点的像素值，可成功消除原来棋盘的残影，实现增强现实效果，如图7所示。

1.4.2 实现效果

实现了三组图像的叠加，如下图所示:

(a) 原图 (b) 阿那亚风景照 (c) 增强现实

图 8: 增强现实示例：阿那亚风景照

(a) 原图 (b) 动漫素材 (c) 增强现实

图 9: 增强现实示例：动漫素材

(a) 原图 (b) 博雅塔 (c) 增强现实

图 10: 增强现实示例：博雅塔

2 任务二：基于深度学习的相机标定

利用 **DeepCalib** [1] 提供的模型和代码[2]，以1中用手机拍摄的 20 张图像为样本预测相机焦距。模型对 20 张图像给出的焦距预测并不相同，从第 1 到第 20 张图片预测值如下：（单位为像素 px）

$$380, 230, 480, 480, 400, 360, 360, 380, 500, 490, 430, 480, 360, 230, 500, 480, 490, 480, 500, 360$$

均值为 418.5px，需要以此计算 FOV 与经典标定方法的效果相比较，并做出评价。
图片宽为 1080px, 可得估计的 FOV 值为：

$$FOV_{deep} = 2\arctan\frac{w/2}{f} = 104.45^\circ \tag{5}$$

经典标定方法对相机焦距的估计约为 850mm, 代入上式得：

$$FOV_{conventional} = 2\arctan\frac{w/2}{f} = 64.86^\circ \tag{6}$$

手机拍摄时未开启广角模式，使用的是"1x"，用物理距离测算视角应该在 60 度到 80 度，并非广角镜头拍摄，因此，此处经典标定方法对 FOV 的估计更合理。深度学习模型效果较差的原因可能有：

- 训练时使用的高分辨率全景图生成的模拟图像，与测试所用手机拍摄图像分布差异较大，泛化性能不好
- 输入需要 resize 为 299x299，信息损失大
- 没有使用标定物（如棋盘格），缺乏真实世界的坐标信息输入，不能显式地优化重投影误差，对焦距的估计误差较大

[2] https://github.com/alexvbogdan/DeepCalib/blob/master/prediction/Classification/Single_net/predict_classifier_dist_focal.py

参考文献

[1] Oleksandr Bogdan, Viktor Eckstein, Francois Rameau, and Jean-Charles Bazin. Deepcalib: a deep learning approach for automatic intrinsic calibration of wide field-of-view cameras. In *Proceedings of the 15th ACM SIGGRAPH European Conference on Visual Media Production*, 2018.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.