Intelligent Acquisition of Visual Information
Lab Assignment 2

郑欣阳* 3160101385　潘启璠　3170104344　吕科瑶　3170102472

2019 年 10 月 8 日

1 Introduction

- Finish intrinsic and extrinsic calibration
- Discuss the impact of different factors (e.g., number of images, coverage of the chessboard, view angle, etc) over the final reprojection error
- Output the estimated camera centers and the chessboard into a .ply file for 3D viewing in software like MeshLab
- Project some interesting 3D points on the input images (i.e., augmented reality)

2 Theory

2.1 Calibration

标定 (Calibration) 主要是解决从世界坐标系到相机坐标系，再到成像坐标之间的投影矩阵的关系，进而确定相机或投影仪的内参 (intrinsic calibration) 和外参 (extrinsic calibration)。其中涉及四个坐标系之间的变换，如图1所示：

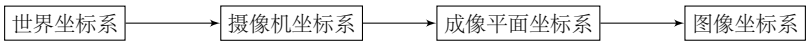


图 1: 摄像机标定过程中各坐标系转换流程

在不考虑图像畸变的条件下，坐标的变换可以写成矩阵相乘的形式：

$$sm' = \mathbf{A}[\mathbf{R}|\mathbf{T}]M' \tag{1}$$

*E-mail：3160101385@zju.edu.cn

即

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2}$$

其中 (X, Y, Z) 为物体在世界坐标系中的位置，(u, v) 为对应在图像坐标系中的位置，(f_x, f_y, c_x, c_y) 称为相机的内参。

另外，我们注意到旋转矩阵 $\mathbf{R}$ 可以写成 $\mathbf{R} = \mathbf{R}(\alpha, \beta, \gamma)$（其中 (α, β, γ) 为三个欧拉角），我们把 $\mathbf{R}$ 和 $\mathbf{T}$ 的 6 个参数 $(\alpha, \beta, \gamma), (T_x, T_y, T_z)$ 称为相机的外参。

如图2所示，我们考虑用古典黑白棋盘来完成相机的校准。

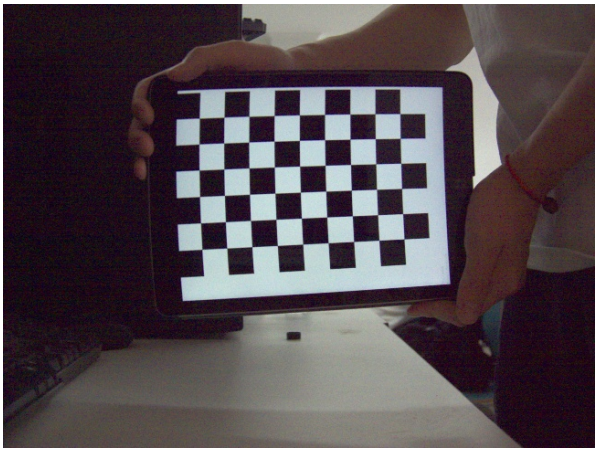


图 2: 古典黑白棋盘

2.2 Distortion

事实上，因为镜头并不完美，图像总会有畸变。对于畸变，OpenCV 考虑到径向和切向因素。对于径向因子，使用以下公式：

$$\begin{aligned} x_{\text{distorted}} &= x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\ y_{\text{distorted}} &= y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \end{aligned} \tag{3}$$

其中 $r^2 \equiv x^2 + y^2$。因此，对于坐标处的未失真像素点 (x, y)，其在失真图像上的位置将为 $(x_{\text{distorted}}, y_{\text{distorted}})$。

另外，由于摄像镜头不完全平行于成像平面，因此会发生切向畸变。它可以通过公式表示：

$$\begin{aligned} x_{\text{distorted}} &= x + \left[2p_1 xy + p_2\left(r^2 + 2x^2\right)\right] \\ y_{\text{distorted}} &= y + \left[p_1\left(r^2 + 2y^2\right) + 2p_2 xy\right] \end{aligned} \tag{4}$$

所以我们有五个畸变参数，它们在 OpenCV 中表示为五维的向量：

$$\text{distortion\_coefficients} = \begin{pmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{pmatrix} \tag{5}$$

3 Calibration

对于相机的校准，我们取 50 张图像，我们直接调用 OpenCV 的`cv2.calibrateCamera()`函数（它会返回摄像机矩阵，畸变系数，旋转和变换向量），即可完成标定。标定结果如下所示：

- 内参：

表 1: 相机矩阵 **A**

897.08	0	291.85
0	903.50	231.07
0	0	1

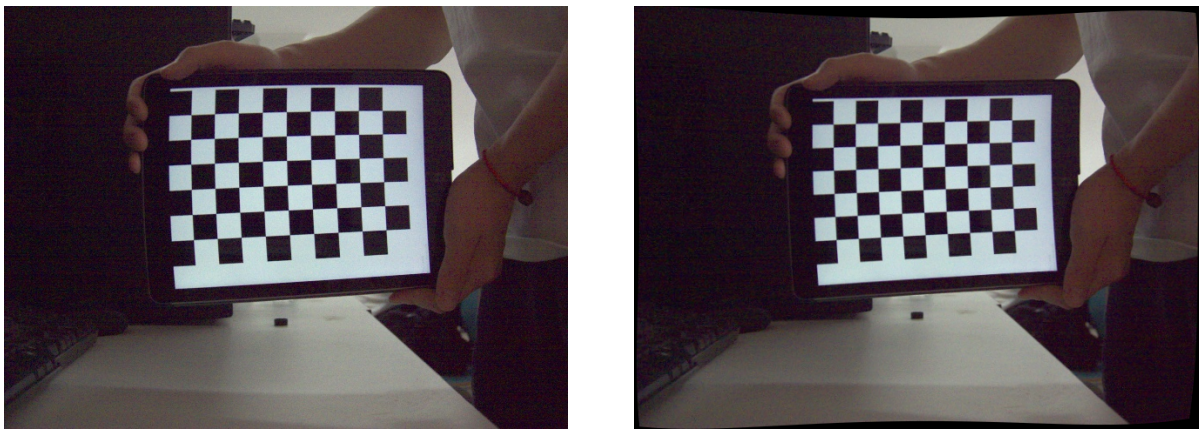
表 2: 畸变系数

-0.468	-2.271	0.0068	0.0173	15.432

详细数据可见`camera_matrix.txt`和`dist_coefs.txt`

- 外参：由于每张图像对应的外参不同，详见`rvec.txt`和`tvec.txt`

我们可以用`cv.undistort()`校正图像的畸变，效果如下图所示：



(a) 原图像　　(b) 畸变校正后的图像

图 3: 畸变校正前后的对比

4 Impact of Different Factors

4.1 Reprojection Error

反向投影误差 (reprojection error) 可以表征我们所找到的参数的准确性，它越接近 0 则说明找到的参数越好。有了内参，畸变参数和旋转变换矩阵，我们就可以使用`cv.projectPoints()`将对象点转换到图像点，然后就可以计算变换得到图像与角点检测算法的绝对误差，最后计算所有标定图像的误差平均值，即为反向投影误差。

我们考虑不同因素对它的影响。

4.2 Number of Images

首先考虑图像数量对反向投影误差的影响，我们一共取 80 张图像，每次随机地取不同数量的图像经过校准后计算反向投影误差（每个相同数量的图像做三组，取平均值），得到的结果如图4所示：

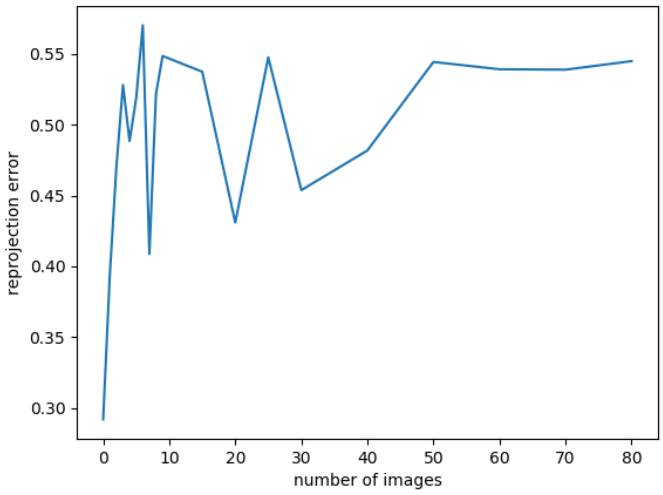


图 4

Reprojection error 和图像数量之间的关系大致表现为：图像数量极少时，error 值非常低；图像数量略微增加后，error 迅速上升，呈现无规则的大幅度波动。图像数量在 30 张以上时，error 趋于稳定。

4.3 Coverage of The Chessboard

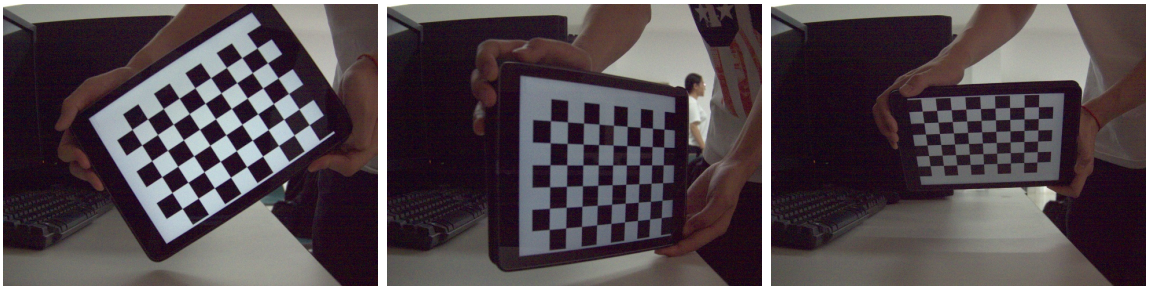
接下来考虑棋盘在图像中所占的大小对反向投影误差的影响，结果如下表所示：

big	0.517
small	0.295

事实上，因为反向投影误差计算的是变换得到图像与角点检测算法的绝对误差，若棋盘所占的像素数目较少，则绝对误差显然会较小。

4.4 View Angle

最后考虑棋盘的偏转角对反向投影误差的影响：我们考虑棋盘分别绕 x, y, z 轴旋转：



(a) 棋盘绕 x 轴旋转　(b) 棋盘绕 y 轴旋转　(c) 棋盘绕 z 轴旋转

绕 x, y 轴的图像各取 50 张，绕 z 轴的图像取 35 张，最终计算得到的反向投影误差如表3所示：

0	0.517
+45°	0.295
+90°	0.128

(a) 棋盘绕 x 轴旋转

-60°	0.199
-30°	0.120
0	0.294
+30°	0.118
+60°	0.432

(b) 棋盘绕 y 轴旋转

-45°	0.129
0	0.309
+45°	0.188

(c) 棋盘绕 z 轴旋转

表 3: Reprojection Error

事实上，棋盘沿 y, z 轴旋转和沿 x 轴旋转的情况不同，前者会因为旋转角度改变正对镜头的面积，并且对亮度也有影响；后者保持正对镜头，亮度基本一致。至于具体 error 随角度变化的关系有待进一步探究。

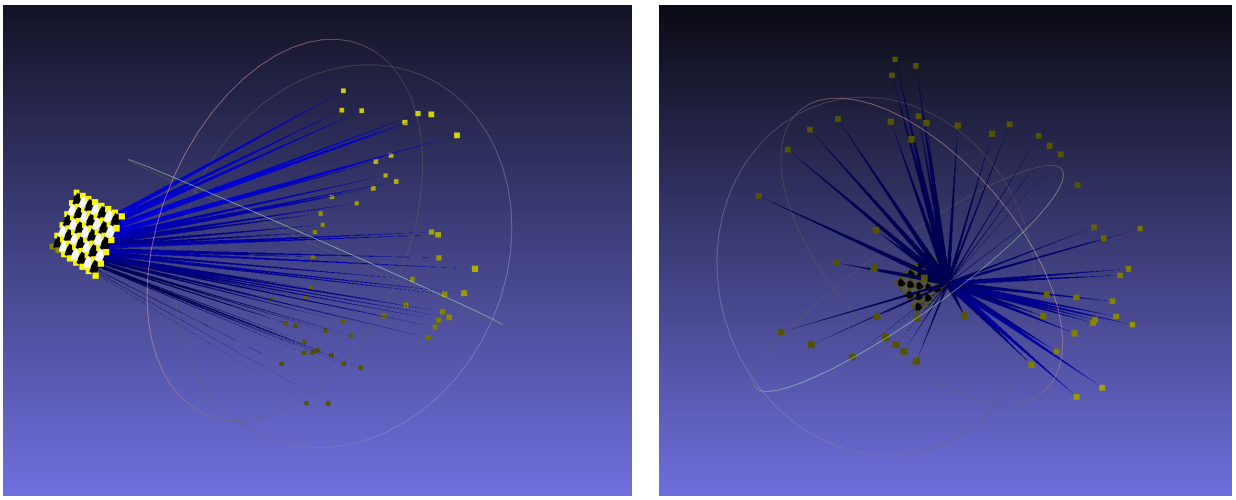
4.5 Discussion

因为在拍摄图像的时候，因为用人手来控制棋盘的移动非常粗糙，实验过程中难以控制变量，所以实验做出来的结果并不是很理想，如果想要进一步探究的话，可能需要铺设滑轨来控制棋盘移动。

5 3D Viewing

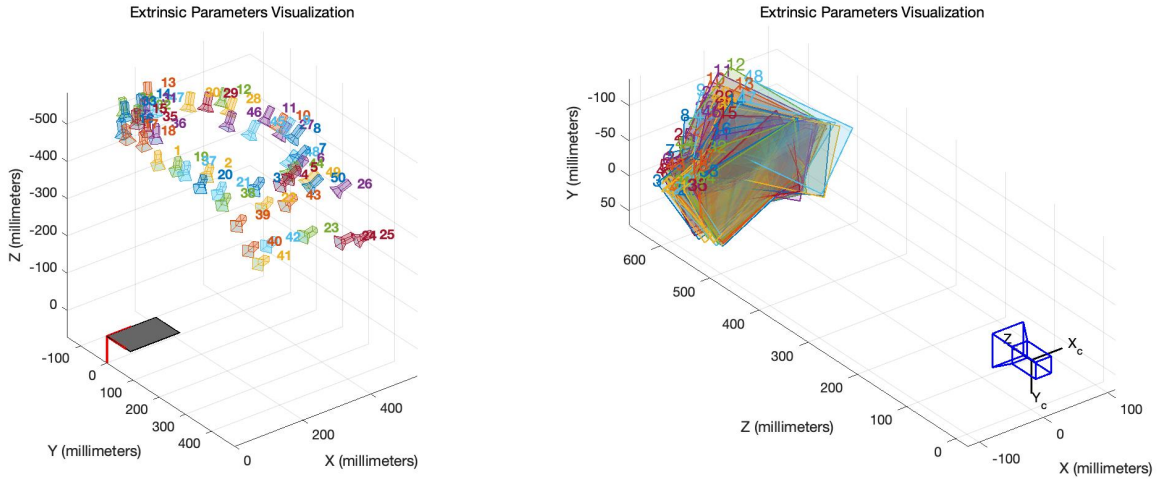
我们取 50 张图像来计算，因为 MeshLab 不能准确地显示点云，所以我们考虑把点连到了棋盘的一条边上来显示棋盘与相机的相对位置，如图6所示：

图 6: 使用 meshlab 查看生成的.ply 文件



另外，我们也可以用 Matlab 的`cameraCalibrator`工具箱，同样可以直接导出三维的视角，如图7所示：

图 7: 三维视角



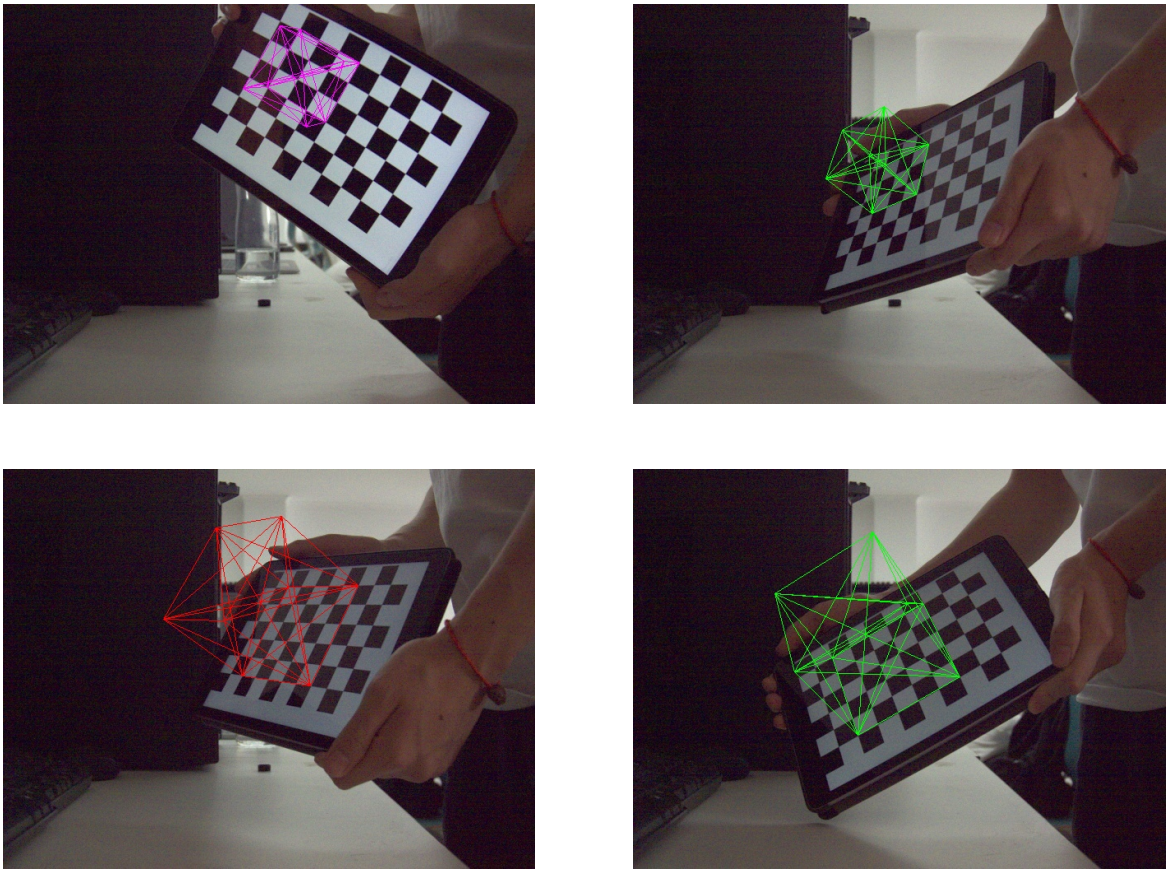
(a) 固定棋盘的三维视角

(b) 固定相机的三维视角

6 Augmented Reality

我们考虑用函数`cv.projectPoints()`将一个立方体的三维顶点投影到二维图像平面上，最终的效果如图8所示：

图 8: Augmented Reality



7 Real-Time Demo

我们考虑固定相机的内参，具体详见`Real-Time.avi`