相机基础

以下内容以针孔相机[1]为参照模型。

1.1相机内部参数[2] ：焦距，图像传感器格式和主点

$$\mathrm{K}=\begin{bmatrix}\alpha_x & \gamma & u_x & 0\\ 0 & \alpha_y & v_y & 0\\ 0 & 0 & 1 & 0\end{bmatrix}$$

a. 参数$\alpha_x = f\cdot m_x$和$\alpha_y = f\cdot m_y$用表示像素级别的焦距 。通常也记作f_x和f_y。

对于f_x和f_y[3]：

通常针孔相机中只有一个透镜焦距。由相机拍摄得到的图像是遵从线性透视规律的。也就是说，一个物体的的宽和高会随着这个物体与相机间的距离增加而按比例变小。而对于一张矩形的图片，一个物体的宽和高则会根据物体与相机的距离按不同的比例变小。而这个比例，就是根据相机的焦距得到的。

当图像是正方形时的关系，即当$f_x = f_y$时的情况。其中f是相机的焦距，以像素为单位；d为物体到相机的距离，单位为米；x是物体在图像中的宽度，w为物体的实际宽度；y是物体在图像中的高度，而h是物体的实际高度。利用针孔模型推导该比例关系如下：

$$\frac{f}{d}=\frac{x}{w}=\frac{y}{h}$$

当图片为矩形时，则有：

$$\frac{f_x}{d}=\frac{x}{w}$$

$$\frac{f_y}{d}=\frac{y}{h}$$

b. γ：代表x和y轴之间的偏度系数，通常为 0

c. u_0和v_0：表示光学中心，通常是图像的中心。通常也记作(c_x, c_y)。表示相机光轴在图像坐标系中的偏移量，以像素为单位。

1.2相机外部参数[4]：相机在某个三维空间中的位置和朝向

a. T（平移矩阵）： 是以相机为中心的坐标系统的坐标表示的世界坐标系原点的位置

b. R（旋转矩阵）： 描述了世界坐标系相对于相机坐标轴的方向

c. 在世界坐标中表示的相机是 $\mathrm{C} = -\mathrm{R}^{-1}T = -\mathrm{R}^{T}\,\mathrm{T}$

/*正交矩阵（行列式=1）的逆等于其转置矩阵*/

1.3相机矩阵[5][6]：相机矩阵由四个矩阵相乘得来，通过相机矩阵M可以实现世界坐标系到图像坐标系的转换。

相机矩阵M的具体表示方式如下

$$M = \begin{bmatrix} a_x & s & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{3\times3} & 0_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} I_{3\times3} & T_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}$$

其中第一个矩阵和第二个矩阵[I|0]的乘积即为内参矩阵K，关于K的介绍在 1.1 中已叙述；第三个矩阵是旋转矩阵R；第四个矩阵是平移矩阵T。

2.将 3D 坐标的点转换到 2D 平面[7]

需要经历的操作如下：物体坐标系->世界坐标系->相机坐标系->投影坐标系->图像坐标系
其中：

物体坐标系->世界坐标系：涉及到旋转、平移、缩放

世界坐标->相机坐标转换：

$$\begin{bmatrix} X_{1c} \\ X_{2c} \\ X_{3c} \\ 1 \end{bmatrix} = [\mathrm{R}|\mathrm{t}] \begin{bmatrix} X_{1w} \\ X_{2w} \\ X_3 w \\ 1 \end{bmatrix}$$

$$[\mathrm{R}|\mathrm{t}] = \mathrm{R}[\mathrm{I}|-\mathrm{C}]$$

相机坐标系->投影坐标系：正交投影与透视投影（降维）

投影坐标系->屏幕坐标系：2D 坐标变换

已知物体坐标位置为$(\mathrm{X}_1, \mathrm{X}_2, \mathrm{X}_3)$，相机焦距$\mathrm{f_x}$和$\mathrm{f_y}$，光学中心$(\mathrm{c_x}, \mathrm{c_y})$，假设存在偏度参数s

设原 3D 坐标转换到 2D 平面上的坐标表示为（$\mathrm{W_x}, \mathrm{W_y}$），根据已知方法，则具体计算如下：

$$\begin{bmatrix} W_x \\ W_y \\ W \end{bmatrix} = \mathrm{M} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{bmatrix}$$

$$\mathrm{M} = \mathrm{K}[\mathrm{R}|\mathrm{t}] = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} [R|t]$$

3.将 2D 平面坐标还原到 3D 相机坐标[8]

已知图片坐标(u, v)，假设第二问基础上的已知条件仍然成立，设该图片 3D 相机坐标下的坐标为$(\mathrm{X}, \mathrm{Y}, \mathrm{Z})$，则具体推导如下：

a. 图像坐标系->投影坐标系

由于(u, v)仅代表图片像素的列数与行数，因此建立以物理单位（毫米）表示的图像坐标系$\mathrm{x}-\mathrm{y}$，定义$\mathrm{x}-\mathrm{y}$坐标系原点为O_1，且令x轴与u轴平行，y轴与v轴平行，假设$(\mathrm{u}_0, \mathrm{v}_0)$代表$\mathrm{O}_1$在$\mathrm{u}-\mathrm{v}$坐标系下的坐标，$\mathrm{d_x}$和$\mathrm{d_y}$为每个像素在 x 轴和 y 轴上的实际物理尺寸，则图像中每个像素在 $\mathrm{u}-\mathrm{v}$坐标系中的坐标和在$\mathrm{x}-\mathrm{y}$坐标系中的坐标对应关系如下：

$$\mathrm{u} = \frac{\mathrm{x}}{\mathrm{d_x}} + u_0$$

$$\mathrm{v} = \frac{\mathrm{y}}{\mathrm{d_y}} + \mathrm{v}_0$$

其逆关系即可表示投影坐标系下的图像坐标，表示关系如下：

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} d_x & 0 & -u_0 d_x \\ 0 & d_y & -v_0 d_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

b. 投影坐标系->相机坐标系

设相机焦距为f，相机和投影平面的距离为z_c。

由小孔成像原理，正向由相机投影到屏幕的过程可以表示为：

$$x = -f\left(\frac{x_c}{z_c}\right)$$

$$y = -f\left(\frac{y_c}{z_c}\right)$$

则其逆向过程即为由投影坐标系还原到相机坐标系的过程，可表示为：

$$\begin{cases} x_c = -\frac{1}{f} z_c x \\ y_c = -\frac{1}{f} z_c y \end{cases}$$

4. 相机畸变

4.1 相机畸变的概念[9][10][12]：透镜由于制造精度以及组装工艺的偏差会引入畸变，导致原始图像的失真。镜头的畸变分为径向畸变和切向畸变两类。

a. 径向畸变：

径向畸变就是沿着透镜半径方向分布的畸变，产生原因是光线在原理透镜中心的地方比靠近中心的地方更加弯曲，径向畸变主要包括桶形畸变和枕形畸变两种。

成像仪光轴中心的畸变为 0，沿着镜头半径方向向边缘移动，畸变越来越严重。畸变的数学模型可以用主点周围的泰勒级数展开式的前几项进行描述，通常使用前两项，即k_1和k_2。对于畸变更为严重的镜头，还可加入参数k_3。成像仪上某点根据其在径向方向上的分布位置，调节公式为：

$$x_0 = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_0 = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

式里(x_0, y_0)是畸变点在成像仪上的原始位置，(x, y)是畸变较真后新的位置。

b. 切向畸变：

切向畸变是由于透镜本身与相机传感器平面（成像平面）或图像平面不平行而产生的，这种情况多是由于透镜被粘贴到镜头模组上的安装偏差导致。畸变模型可以用两个额外的参数 p1 和 p2 来描述:

$$x_0 = x + [2p_1 y + p_2(r^2 + 2x^2)]$$
$$y_0 = y + [2p_2 x + p_1(r^2 + 2y^2)]$$

上述径向畸变和切向畸变模型中一共有k_1，k_2，p_1，p_2这 4 个畸变参数。通常情况下，求得畸变参数后，就可以校正由于镜头畸变引起的图像的变形失真。

4.2 相机畸变的还原[9][11][14]

考虑存在径向畸变系数k_1和k_2，切向畸变系数p_1，p_2，设主点(c_x, c_y)，相机焦距为f_x, f_y，由畸变前图像坐标(u_0, v_0)推导畸变后图像坐标(u, v)过程如下：

$$取畸变前：\begin{cases} x' = \frac{u_0 - c_x}{f_x} \\ y' = \frac{v_0 - c_y}{f_y} \end{cases}$$

$$则，畸变后可表示为：\begin{cases} x'' = x'(1 + k_1 r^2 + k_2 r^4) + 2p_1 x' y' + p_2\left(r^2 + 2x'^2\right) \\ y'' = y'(1 + k_1 r^2 + k_2 r^4) + 2p_2 x' y' + p_1(r^2 + 2y'^2) \end{cases}$$

$$其中 \mathrm{r}^2 = \mathrm{x}'^2 + y'^2$$

$$故，畸变后的坐标可表示为：\begin{cases} u = f_x x'' + c_x \\ v = f_y y'' + c_y \end{cases}$$

故，已知畸变后图像坐标(u, v)，求畸变前图像坐标$(\mathrm{u}_0, \mathrm{v}_0)$的过程，需要首先找到整像素点畸变后的位置，然后插值（因为畸变后的图像坐标很可能不在整点位置上，而实际上像素值都是整数坐标）算出畸变后像素点的位置，并将这个位置的灰度值赋给原来的整像素点，实现畸变矫正。

5. 相机标定[13]

在图像测量过程以及机器视觉应用中，为确定空间物体表面某点的三维几何位置与其在图像中对应点之间的相互关系，必须建立相机成像的几何模型，这些几何模型参数就是相机参数。在大多数条件下这些参数必须通过实验与计算才能得到，这个求解参数的过程就称之为相机标定。

当前相机标定方法概述[15][16]：相机标定大致可以分为相机自标定法；基于主动视觉的标定方法；基于标定物的相机标定法。

a. 相机自标定法：

相机自标定法仅需图像对应点信息就可以直接完成相机标定。相机自标定法的基本思想是在相机做刚体运动时，绝对二次曲面或绝对二次曲线的像保持不变，它们的方程只跟相机的内部参数有关。因此可以把绝对二次曲线或二次曲面看作一个虚拟标定物，然后根据成像过程中不同形式的约束方程求解出标定参数。

b. 基于主动视觉标定法:

基于主动视觉的相机标定方法的基本思想是根据自动地控制相机来获取图像数据从而线性地求解出相机的模型参数，由于在标定过程中可以知道关于相机的运动信息，所以这种标定方法的主要优点是可以线性求解出相机的模型参数，并且求解的计算简单、鲁棒性较高。

c. 基于标定物的相机标定法：

基于标定物的相机标定方法需要使用具备先验知识的特定标定物，一般来说这些先验知识包括标定物的尺寸、形状以及标定物表面特征点的坐标值。通常使用的标定物可以分为平面标定物和立体标定物。

目前比较具有代表性的基于标定物的方法有：直接线性变换法，Tsai 两步法，张正友平面标定法等。

6.使用 OpenCV API 进行相机标定[9][17]

由于题目所提供的图片是棋盘，因此我使用函数cv2.findChessboardCorners()用于查找棋盘图案。该函数在查找成功的情况下该函数会返回图片从左到右、从上到下依次排列的角点坐标。对于已有数据集来说，数据集中的棋盘的内部角点为 7*6，因此传入图案的类型采用 7*6 比较稳定。left 文件夹下一共有 13 张图片，此时共有 11 张图片查找成功。在找到角点后采用cv2.cornerSubPix() 增加准确度。并使用函数 cv2.drawChessboardCorners() 绘制查找棋盘图案成功后的角点位置。

实验结果展示 Figure 1：

Figure 2 **角点查找**

（从左至右分别为 left 文件夹下图片 01，03，05，13）

之后使用准备好的对象点和图像点，调用cv2.calibrateCamera()，即可返回摄像机矩阵，畸变系数，旋转和变换向量等所需参数。具体计算结果如下：

相机矩阵：

$$K = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 534.07 & 0 & 341.53 & 0 \\ 0 & 534.12 & 232.95 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

畸变系数：

$$[k_1, k_2, k_3, p_1, p_2] = [-2.93e-01, 1.08e-01, 1.31e-03, -3.11e-05, 4.35e-02]$$

7.畸变校正[9][17]

在第六步相机标定完成的基础上，使用函数 cv2.getOptimalNewCameraMatrix() 得到的自由缩放系数对摄像机矩阵进行优化。之后进行畸变校正。

最简单的方式是采用cv2.undistort()对图像进行畸变校正。

也可使用cv2.remap()，先找到从畸变图像到非畸变图像的映射方程，再使用重映射方程。

两种方法的结果相同，具体展示见 Figure 3：

Figure 4 **畸变校正与原图对比**

（从左至右分别为 left 文件夹下图片 01，03，05，13）

8.实现张正友相机标定方法[15][20]

2018.3.30 – 2018.4.4实验大致实现思路如下：

1) 根据文章附录 A 中提示，求解H矩阵。将所有图片分成三份，这样可以获得三个 H 矩阵。具体求解方法为直接使用cv2. findChessboardCorners()方法查找 7*6 的棋盘角点坐标，并自己定义一个世界坐标系坐标数组，运用两个数组中对应的点，通过文章中给m_i与m_i^*出方差和最小的关系进行拟合，求解 H 矩阵。
2) 在单应性矩阵H求解的基础上，直接对矩阵 V 赋值。其中:

$$v_{ij} = \left[h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j2}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}\right]^T$$

3) 调用np. linalg. svd()方法求解齐次线性方程组$Vb = 0$，求解得到矩阵b。
4) 根据附录 B 中等式关系计算出相机内参矩阵。

实验结果：

由于计算H矩阵是应用到随机数生成初始解，并通过拟合的方式求解H矩阵中的具体参数，导致矩阵中参数的计算结果始终不够稳定。

在利用H矩阵求解齐次线性方程组的过程中，求解的结果不是一个稳定的值，且由于解的符号问题，导致有较高概率内参矩阵计算失败。

2018.4.7 – 2018.4.11实验大致实现思路如下：

1) 直接调用 cv2. findHomography() 求解 H 矩阵。对于每一张图片首先使用 cv2. findChessboardCorners()方法查找 7*6 的棋盘角点坐标，并自己定义一个世界坐标系坐标数组，利用两个数组中点的映射关系，向函数cv2. findHomography()中传入参数，求解 H 矩阵。此时将获得对应图片数量个 H 矩阵。
2) 在单应性矩阵 H 求解的基础上，直接对矩阵 V 赋值。其中:

$$v_{ij} = \left[h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j2}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}\right]^T$$

3) 调用np. linalg. svd()方法求解线性方程组$Vb = 0$，由于此时有 2n 行等式，而未知数的个数为 6，属于超定方程组，因此首先利用等式$V^T Vb = V^T 0$，将等式转换成齐次线性方程组后再求解得到矩阵b
4) 根据附录 B 中等式关系计算出相机内参矩阵

实验结果：

本周实验直接调用 OpenCV 方法求解H矩阵，求解结果稳定，且内参矩阵的结果计算稳定，不会出现上一周实验中的计算失败现象。

在计算内参矩阵时，其计算结果和第 6 问中的计算结果有数量级上的差异，暂时还未分析出导致该结果出现的原因。

当前结果仅仅是一个初始解，还未进行优化操作。

计算结果为：

$$K = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0.0473 & -0.0029 & -0.0209 & 0 \\ 0 & 0.0647 & 0.0066 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

2018.4.12 – 2018.4.15实验大致实现思路如下：

1) 本周实验过程中，我参考了github[21]上开源的代码，发现在内参矩阵的计算过程中，我

和作者的实现思路基本一致，唯一差别在于H矩阵坐标的使用方式，于是我联想到曾经使用 OpenCV 处理数据时了解到，OpenCV 整体使用的图像的长宽方向的坐标系和我们日常习惯的坐标系维度恰好相反，即 OpenCV 中矩阵的第一个维度表示宽，第二个维度表示高，因此分析得出 :OpenCV 函数库直接计算的结果 H 其实是 Zhang 的论文中H^{T}。

我先继续直接调用cv2. findHomography()求解 H 矩阵，并更正了矩阵V 的赋值操作，得出和第六问直接调用函数的相似结果。

计算结果为：

$$K=\begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}=\begin{bmatrix} 545.995 & -1.989 & 355.192 & 0 \\ 0 & 546.556 & 225.643 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

2) 之后我利用论文中附录 A 中提供的方法，并参考知乎专栏[22]，使用等式：

$$\mathrm{Au}=\mathrm{v}$$

$$\mathrm{A}=\begin{bmatrix} \mathrm{x}_1 & y_1 & 1 & 0 & 0 & 0 & -\mathrm{x}_1 x_2 & -x_2 y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -\mathrm{x}_1 y_2 & -y_1 y_2 \end{bmatrix}$$

$$\mathrm{u}=[\mathrm{h}_{11}\ \mathrm{h}_{12}\ \mathrm{h}_{13}\ \mathrm{h}_{21}\ \mathrm{h}_{22}\ \mathrm{h}_{23}\ \mathrm{h}_{31}\ \mathrm{h}_{32}]^{\mathrm{T}}$$

$$\mathrm{v}=[\mathrm{x}_2\ \mathrm{y}_2]^{\mathrm{T}}$$

手工计算单应性矩阵H，替换之前直接调用函数的部分。单应性矩阵的计算在函数dlt. py中。

此时计算结果为：

$$K=\begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}=\begin{bmatrix} 532.806 & 1.521 & 350.359 & 0 \\ 0 & 529.066 & 225.710 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

3）根据论文中提供方法计算外参矩阵，每一幅图片都有其对应的一个外参矩阵。计算函数在extrinsics. py中。

实验总结：

到此为止，我已基本实现 Zhang 论文中的相机标定方法。

此次实验共使用半个月的时间，从前期的阅读论文、分析论文中的推导，到实现过程中不断遇到各类问题并逐一解决。在整个过程中，由于日常课业关系，虽然我并不是每天都在调试代码，但是对当前遇到的问题的思考和问题解决后的反思贯穿在整个过程中。

在此次实验过程，我根据题目的引导，从最简单的直接调用函数操作，到分部分用自己编写的代码替换 OpenCV 提供的函数，再到完全使用自己的代码实现相机标定操作。我个人认为这种分模块实现并时刻进行编码结果检验方式有利于我解决规模较大的问题。

相机基础部分所有代码：https://github.com/SelinaFelton/Camera-Calibration

9.Summary

到目前为之，我掌握的传统方法仍然无法通过单目相机进行深度估计。

对于第三问的解决方案来说，最终能实现的仅仅是将图像坐标系下的图像还原到相机坐标系下，且由投影坐标系还原到相机坐标系的前提则是自己设定一个投影平面到相机位置的距离z_{c}，由此并不能实现真正估计出每个像素到图像平面的距离的操作。

对于估计像素深度，我个人认为在单目的条件下可以通过图像中景物的遮挡关系、“近

大远小”的透视关系、在拍摄中往往会对焦于前景而背景呈现模糊状态这几种条件大致判断各个部分之间的远近关系，但是仍然无法获得具体的数值。

我也查找了部分资料[18][19]，了解了单目深度估计的做法。我了解到的相关解决方案是基于分割的方法。首先将图像分割成几个部分，并且假设被分割的区域内所有景物的深度相同且忽略其中的遮挡关系；之后获取已有的分割区域之间的深度线索，深度线索包括遮挡信息、运动信息、消失点等；最后进行全局推理，获得深度图。

参考文献

[1].*针孔相机概念见维基百科:* [*Pinhole camera model*], https://en.wikipedia.org/wiki/Pinhole_camera_model

[2].*参考维基百科:* [*Camera resectioning*], https://en.wikipedia.org/wiki/Camera_resectioning

[3].*参考博客:*[*关于相机内参中的焦距 fx 和 fy*],*http://www.cnblogs.com/zipeilu/p/6658177.html*

[4].*参考维基百科:* [*Camera resectioning*], https://en.wikipedia.org/wiki/Camera_resectioning

[5]. *参考知乎，并根据维基百科对应内容，已做相应修改：*[*相机矩阵*], https://zhuanlan.zhihu.com/p/33406665

[6]. *相机矩阵具体分解过程，同时参考博客：*[*摄像机矩阵详解*], *http://haiyangxu.github.io/posts/2014/2014-06-12-camera-matrix.html*

[7]. *以上步骤详细推导过程见博客:*[*从世界坐标系到相机坐标系（3D 物体到 2D 图像的转变*], http://blog.sina.com.cn/s/blog_8c388a3a0101dd15.html

[8]. *参考知乎，并根据题目 2 正向坐标转换过程进行反向推导:* [*相机的那些事儿（二)成像模型*], *https://zhuanlan.zhihu.com/p/23090593*

[9].*OpenCV 文档:* [*Camera Calibration and 3D Reconstruction*], https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

[10].*参考博客:* [*透镜畸变及校正模型*], https://blog.csdn.net/dcrmg/article/details/52950141

[11].*参考博客:* [*[图像]畸变校正详解*], https://blog.csdn.net/humanking7/article/details/45037239

[12].*参考论文:冯焕飞，“三维重建中的相机标定方法研究”，重庆交通大学硕士学位论文，2013.5.26*

[13].*参考博客:* [*相机标定（Camera calibration）原理、步骤*], https://blog.csdn.net/lql0716/article/details/71973318?locationNum=8&fps=1

[14].*参考博客:* [*镜头畸变矫正*], https://blog.csdn.net/waeceo/article/details/50580808

[15].*参考论文: Z. Zhang, “A flexible new technique for camera calibration,” IEEE Transactions on pattern analysis and machine intelligence, vol. 22, no. 11, pp. 1330–1334, 2000.*

[16].*参考论文:吴丹，“计算机视觉中相机标定算法研究”，华中科技大学硕士学位论文，2014.5.24*

[17].*参考博客:* [*摄像机标定和 3D 重构*], https://www.cnblogs.com/Undo-self-blog/p/8448500.html

[18].*参考论文:牛玉婷，“单目图像中的场景深度估计理论与方法，合肥工业大学硕士学位论文，2016.4*

[19].*参考论文: Gould S，Fulton R Koller D. “Decomposing a scene into geometric and semantically consistent regions” IEEE International Conference on Computer Vision．Kyoto，Japan：IEEE，2009，30(2)：1-8．*

[20].*参考博客:* [*张正友标定算法原理详解*], https://blog.csdn.net/u010128736/article/details/52860364

[21].*参考代码:* [*Python implementation of Z. Zhang's camera calibration algorithm*] zhang/steps/intrinsics.py, https://github.com/onurtemizkan/zhang/blob/master/steps/intrinsics.py

[22]. *参考知乎，王小龙回答：*[*推导四对对应点单应矩阵的计算公式?*], https://www.zhihu.com/question/23310855