第3章 几何标定

相机制造时的工艺限制，在未经处理前，其拍摄到的图像常具有形状畸变，如现实中笔直的物体在图像中变弯曲。因此，在进行各类图像处理操作之前通常需要计算出相机的相关参数，从而对拍摄到的图片进行畸变矫正，得到能反映物体真实形状的图像。计算相机各种几何参数的过程称为几何标定。此外，只有已知相机自身的几何参数之后，才能通过拍摄到的图像计算所拍摄物体的大小、位置和角度等信息，因此对于立体视觉和摄影测量等计算机视觉任务来说几何标定也是至关重要的。在讲解几何标定方法之前，3.1 节将介绍相机的基本成像原理、相关的几何知识及相机的几何参数；在此基础上，3.2 节将介绍几何标定的原理和具体算法步骤；作为实践，3.3 节将介绍如何使用 Python 语言和 OpenCV 工具包对相机进行几何标定及如何将拍摄到的照片去畸变。

3.1 相机几何

本节首先简要介绍两种经过简化的相机成像几何模型，即针孔相机模型与简单透镜模型；之后介绍与相机成像密切相关的几何变换，如平移、旋转、刚体变换、缩放、仿射变换与投影变换，并推导其在齐次坐标系统下的矩阵表示；最后介绍相机的内、外参数，为后续几何标定流程的讲解做铺垫。

3.1.1 针孔相机模型

文艺复兴初期的意大利建筑师布鲁内莱斯基(Brunelleschi)最早提出了几何透视模型[1]，研究了如何将三维空间中的物体在二维画布上表现出来，如图 3.1 所示。

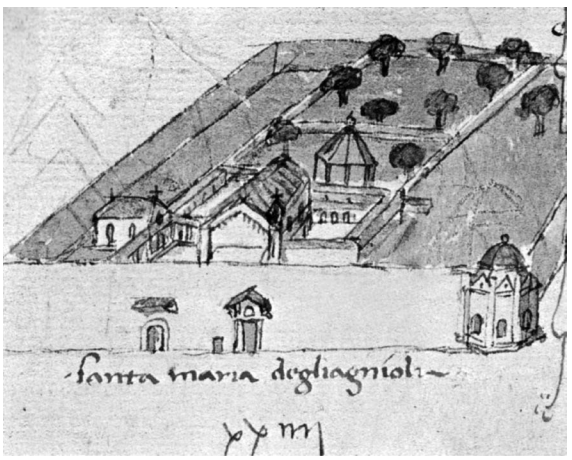


图 3.1 布鲁内莱斯基依据集合透视原理设计的《乡村法典》

早期的针孔照相机就是受几何透视模型的启发而发明的。如图 3.2 所示，在暗箱正前方开一个针孔，暗箱前方物体发出或反射来的光线穿过针孔照射到暗箱后端的物理成像平面上，由其上的光传感器或底片接收、记录影像。

根据光沿直线传播及小孔成像原理，图 3.2 中物理成像平面处得到的物体的实像是倒着的，更准确地说是与原物体关于针孔有缩放地中心对称。为了更直观、更方便地研

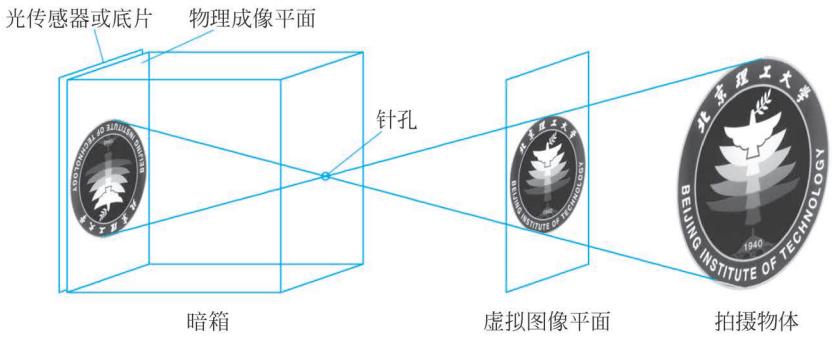


图 3.2 针孔相机模型

究物体的像与原物体间的几何关系，通常在与物理成像平面关于针孔中心对称的位置假想出一个虚拟图像平面，此平面上的图形其实是在针孔位置观察物体时得到的图像。注意这里所说的“虚拟图像平面”与镜子中的“虚像”含义不同，“虚像”是可以看到的，而“虚拟图像平面”是为了简化问题假想出来的。虚拟图像平面与物体在针孔的同一侧，且不再是倒像，因此讨论虚拟图像平面与物体之间的几何关系会容易很多。

图 3.2 中虚拟图像平面和物理成像平面到针孔的距离是相等的（通常称为焦距或像距，见 2.2.3 节），因此两个平面上的对应图形是全等的，研究虚拟图像平面上的图形就相当于研究了物理成像平面上的图形。3.2 节将根据针孔相机模型推导相机、物体及虚拟图像平面的几何关系，进而计算该相机的各种几何参数。

3.1.2 简单透镜模型

针孔相机原理简单，但实际使用时又存在矛盾：一方面为了应用小孔成像原理，即让入射光线照射到成像平面唯一的位置上，小孔应该越小越好；另一方面为了使像足够亮，应该允许更多的光线射入暗箱，即小孔越大越好。为解决这个问题，现代相机通常用一个或一组透镜代替小孔。理想的透镜如图 3.3 所示，从点 P 发出的光线可能从任意方向入射到透镜上，但经过透镜的两面时进行两次折射，最终都会聚到成像平面的像点 P'。

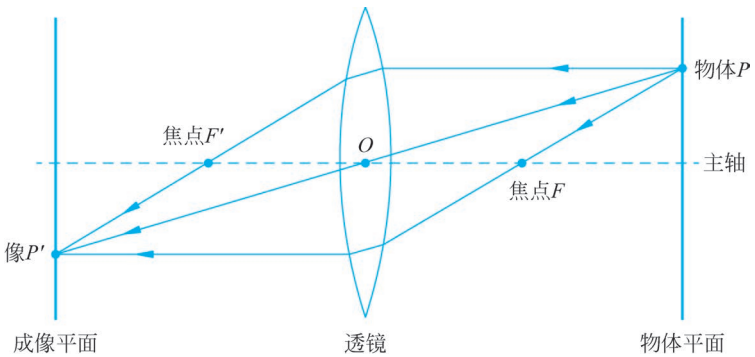


图 3.3 理想的透镜

图 3.3 中最上方光路的入射光线水平，根据焦点的定义，出射光线一定会经过透镜的左焦点 F'；中间的光路穿过透镜中心 O，由于透镜两个表面中心对称，出射光线方向不变；最下方光路的入射光线过透镜的右焦点 F，根据焦点的定义，出射光线一定会水平射出。由此可见，根据透镜的成像规律，点 P 射出的任意方向光都会被聚焦到像点 P'，这同时保证了光线充足和聚焦清晰，解决了针孔成像问题。然而上述机制假定了物体平面与成像平面与透镜的距离相等，当点 P 不在图 3.3 所示的物体平面时，其发出的各方向光线并不一定能严格地聚焦到同一点。下面证明当物体和折射点都距离主轴（如图 3.3 中虚线所示，穿过透镜中心且与透镜及两个平面都垂直的直线）足够近时，其像被严格聚焦到同一点。

为简化问题，证明中仅考虑透镜单侧表面上的折射，考虑双侧表面折射时严格聚焦的结论也同样成立，可参考几何光学或应用光学书籍中对共轴球面光学系统的介绍[2]。在图 3.4 中，仅考虑透镜的右表面，其距离主轴很近的部分可认为是标准的球面，球心为点 C。由于已经假设图中的所有点都是接近主轴的，因此图中所示的所有角度都很小，即对图中标记出的任一角 θ，有

$$\theta \approx \tan\theta \tag{3.1}$$

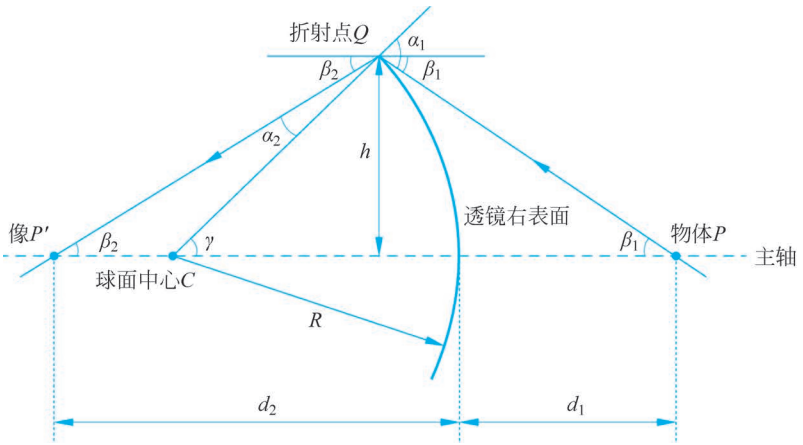


图 3.4　近轴折射聚焦原理

具体到图中的角 γ、β_1、β_2，有

$$\gamma \approx \tan\gamma = \frac{h}{R}, \quad \beta_1 \approx \tan\beta_1 = \frac{h}{d_1}, \quad \beta_2 \approx \tan\beta_2 = \frac{h}{d_2} \tag{3.2}$$

在△CPQ 中，外角 α_1 等于两内角 γ 和 β_1 之和，再结合式(3.2)可得

$$\alpha_1 = \gamma + \beta_1 \approx h\left(\frac{1}{R} + \frac{1}{d_1}\right) \tag{3.3}$$

另外，由内错角相等可知，$\gamma = \alpha_2 + \beta_2$，再结合式(3.2)可得

$$\alpha_2 = \gamma - \beta_2 \approx h\left(\frac{1}{R} - \frac{1}{d_2}\right) \tag{3.4}$$

根据折射定律[斯涅耳(Snell)法则]，光线经过两不同介质表面会发生折射，入射角与反

射角的正弦值之比等于两种介质折射率的反比。假设透镜外的介质折射率为 n_1，透镜折射率为 n_2，则根据折射定律、式(3.3)和式(3.4)可得

$$\frac{n_2}{n_1}=\frac{\sin\alpha_1}{\sin\alpha_2}\approx\frac{\alpha_1}{\alpha_2}=\frac{\frac{1}{R}+\frac{1}{d_1}}{\frac{1}{R}-\frac{1}{d_2}} \tag{3.5}$$

整理上式可得

$$d_2=1/\left[\frac{1}{R}-\frac{n_1}{n_2}\left(\frac{1}{R}+\frac{1}{d_1}\right)\right] \tag{3.6}$$

由此可见，当折射率 n_1、n_2 和透镜球面半径 R，以及透镜和物体间距 d_1 固定时，像点与透镜间距 d_2 也是固定的，即从图 3.4 中点 P 向各个方向发出的光线通过透镜聚焦到同一点 P'。

当物体或折射点距离主轴很远时，式(3.6)不再成立，因为此时式(3.1)不再成立。

第一种情况是物体在主轴附近，但折射点远离主轴，这造成单一透镜成像时出现图 3.5 所示的模糊圈。

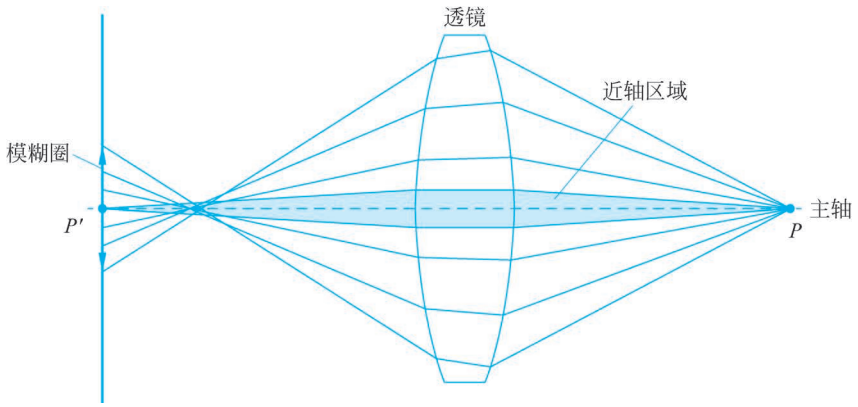


图 3.5　球形像差示意图

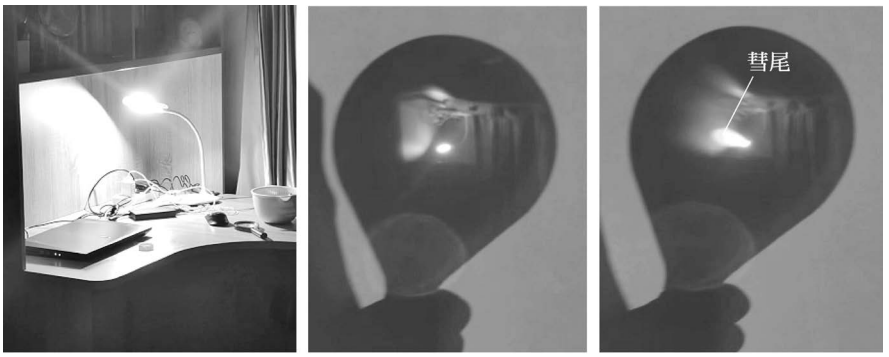
根据上述近轴折射聚焦原理，图 3.5 中灰色区域内的近轴光线(黑色折线)被严格聚焦到成像平面上的点 P'，而其他远离主轴的光线(其他折线)无法保证聚焦，经过实验发现它们会被投射到 P' 周围的圆形区域，形成一圈模糊的图像，这种现象被称为球形像差。

第二种情况是当物体偏离主轴时，其像会像彗星的尾部一样拖出一条光斑，如图 3.6 所示，这种现象称为彗形像差。

另外，透镜对不同波长光线的折射率是不一样的，因此具有混合色彩的光线通过透镜时会出现色散现象，它们聚焦的位置也不同。图 3.7 展示了两种情况下不同色光成像位置的差别——纵向色差和横向色差。

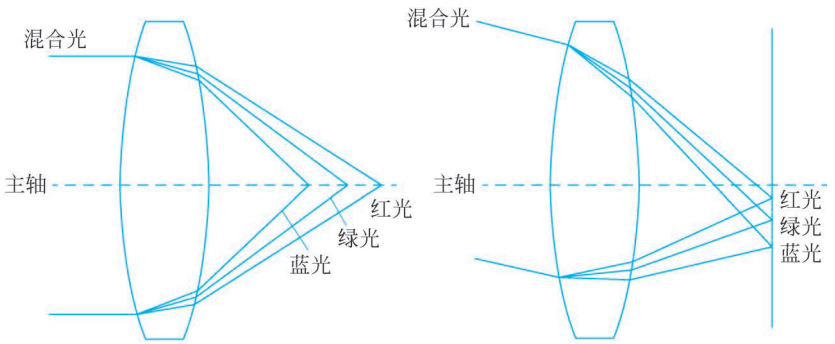
以上介绍的这些像差问题通常可以用多个透镜及光圈组成的复合镜头解决。例如，可以使用消色差透镜减少部分色光的色差。

其他像差问题不再一一介绍，读者可以查找几何光学和像差的相关书籍[3]了解各类



(a) 光源　　(b) 正常成像　　(c) 彗形像差

图 3.6　使用放大镜在一张白纸上投射台灯附近的发光场景。当光线垂直穿过放大镜镜片时，成清晰的倒像；当光线与镜片不垂直时，光源在原本的清晰成像点左侧拖出像“彗尾”一样的模糊光带，产生彗形像差



(a) 纵向色差（主轴方向）　　(b) 横向色差（成像平面方向）

图 3.7　纵向色差和横向色差

像差的形成原因与对应的解决方案。尽管现实中的相机包含各类复杂的光学元件以减少像差问题的影响，但是从宏观效果来看用相机拍照时物体与像之间的关系仍然大致满足针孔相机模型。3.2 节和 3.3 节中相机标定实验的结果将验证这样简化的合理性。为几何标定做铺垫，3.1.3 节～3.1.5 节将讲解一些后续需要用到的几何知识，3.1.6 节与 3.1.7 节会利用这些几何知识建立物体从世界坐标系投影到图像坐标系的几何变换模型。

3.1.3　平移、旋转与刚体变换

本节首先讨论平面上点的平移和旋转。将平面上的点 $\boldsymbol{p}$ 平移到另一点 $\boldsymbol{p}'$ 的过程写成向量形式为

$$\boldsymbol{p}' = \boldsymbol{p} + \boldsymbol{t}$$

即

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{3.7}$$

式中：t_x 和 t_y 分别是 x 方向和 y 方向的平移量；$\boldsymbol{t}$ 是 t_x 和 t_y 组成的平移向量。

点的旋转比平移更复杂一些。考虑平面上点 $\boldsymbol{p}$ 绕原点沿逆时针方向旋转 θ 角的过程，如图 3.8 所示。

设向量 $\boldsymbol{Op}$ 与 x 轴夹角为 α，向量 $\boldsymbol{Op}'$ 与 x 轴夹角为 α'，则有角度关系：

$$\alpha' = \alpha + \theta \tag{3.8}$$

对式(3.8)用正弦函数和角公式可得

$$\sin\alpha' = \sin\alpha\cos\theta + \cos\alpha\sin\theta \tag{3.9}$$

另外，将点 $\boldsymbol{p}$ 和 $\boldsymbol{p}'$ 表示成极坐标形式可得

$$\begin{cases} x = r\cos\alpha \\ y = r\sin\alpha \end{cases}, \quad \begin{cases} x' = r\cos\alpha' \\ y' = r\sin\alpha' \end{cases} \tag{3.10}$$

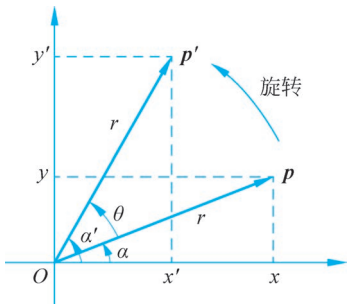


图 3.8　平面点的旋转

因此有

$$\begin{cases} \cos\alpha = x/r \\ \sin\alpha = y/r \end{cases}, \quad \begin{cases} \cos\alpha' = x'/r \\ \sin\alpha' = y'/r \end{cases} \tag{3.11}$$

将式(3.11)代入式(3.9)可得

$$\frac{y'}{r} = \frac{y}{r}\cos\theta + \frac{x}{r}\sin\theta \Rightarrow y' = y\cos\theta + x\sin\theta \tag{3.12}$$

同理，对式(3.8)用余弦函数和角公式可得

$$\cos\alpha' = \cos\alpha\cos\theta - \sin\alpha\sin\theta \tag{3.13}$$

将式(3.11)代入式(3.13)可得

$$\frac{x'}{r} = \frac{x}{r}\cos\theta - \frac{y}{r}\sin\theta \Rightarrow x' = x\cos\theta - y\sin\theta \tag{3.14}$$

根据式(3.12)和式(3.14)，旋转变换的矩阵与向量形式可写为

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \quad \boldsymbol{p}' = \boldsymbol{R}\boldsymbol{p} \tag{3.15}$$

式中：$\boldsymbol{R}$ 为旋转矩阵，只与旋转角度 θ 有关。

容易看出，当 θ 角为负数(顺时针旋转)时上式仍然成立。

平移变换和旋转变换的组合变换通常称为刚体变换。刚体变换不改变物体的形状和大小，只改变物体的位置和朝向。由式(3.7)和式(3.15)易知，刚体变换过程可以分解为一次旋转和一次平移，写成向量与矩阵形式为

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \quad \boldsymbol{p}' = \boldsymbol{R}\boldsymbol{p} + \boldsymbol{t} \tag{3.16}$$

可以看出，上式包含一个矩阵乘法和一个向量加法，分别代表旋转操作 $\boldsymbol{R}$ 和平移操作 $\boldsymbol{t}$。为了更简单地描述刚体变换及更复杂的几何变换，最好能用矩阵乘法的形式统一表示所有变换，即 $\boldsymbol{p}' = \boldsymbol{A}\boldsymbol{p}$，其中变换矩阵 $\boldsymbol{A}$ 包括所有要对 $\boldsymbol{p}$ 进行的变换操作。从式(3.16)可以看出，由于普通坐标系统下向量加法的限制，平移变换是无法放到矩阵 $\boldsymbol{A}$ 中的，因此需要引入一个更完美的坐标系统——齐次坐标系统。

3.1.4 齐次坐标系统

简单地说，齐次坐标系统通过增加额外的一维坐标 w，将各种常见几何变换的表示形式统一为一个矩阵乘法。以二维情况为例，点 $\boldsymbol{p}(x,y)$ 的齐次坐标表示为 (xw,yw,w)，其中 w 可以是任意非零实数，其并不影响 $\boldsymbol{p}$ 点的位置。例如，点(2,3)的齐次坐标可以是(2,3,1)、(6,9,3)、(−2/7,−3/7,−1/7)等。可以发现，平面上任何一个点的齐次坐标不是唯一的。同样地，原点(0,0)的齐次坐标可以是(0,0,1)、(0,0,−3)等。

当最后一个分量为 0 时，齐次坐标 $(x,y,0)$ 表示向量 (x,y) 方向的无穷远的点。这一点也很好理解：当不全为零的坐标 x 和 y 固定时，(x,y,w) 表示的点等于 $(x/w,y/w,1)$ 所表示的点，也就是普通坐标点 $(x/w,y/w)$，其一定在向量 (x,y) 所在的直线上。w 为正数时该点在向量正方向上，此时 w 越小，表示的点离原点越远。当 w 接近 0 时，所表示的点与原点距离接近无穷大，因此 w 等于 0，正好表示此方向上的无穷远的点。显然表示无穷远的点时，x 和 y 不能同时为 0，否则就不知道这个向量是哪个方向，因此齐次坐标不允许所有分量同时为 0，即不存在点(0,0,0)。二维平面的齐次坐标系下的原点是 $(0,0,w)$，其中 w 是任意非零实数，不能是 0。

除了能表示无穷远点外，齐次坐标系统的优势是可以将各种常见几何变换的表示形式统一为一个矩阵乘法。首先看平移变换，将式(3.7)中平移前的点 $\boldsymbol{p}$ 和平移后的点 $\boldsymbol{p}'$ 都用齐次坐标表示可得

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x+t_x \\ y+t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \boldsymbol{p}' = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{t} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \boldsymbol{p} \tag{3.17}$$

上式右侧的等式是使用了分块矩阵的形式来简化表示，其中 $\boldsymbol{I}$ 是二阶单位矩阵，$\boldsymbol{0}^{\mathrm{T}}$ 是一行两列的零向量，$\boldsymbol{t}$ 仍然是两行一列的平移向量。通过这种方式，平移变换也被写作在原来点 $\boldsymbol{p}$ 左边乘以一个变换矩阵的形式。

此外，将式(3.15)中的旋转矩阵 $\boldsymbol{R}$ 增加一行一列，即可得到旋转变换的表达式：

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ y\cos\theta + x\sin\theta \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \boldsymbol{p}' = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{0} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \boldsymbol{p} \tag{3.18}$$

对比式(3.17)和式(3.18)可以发现，n 维齐次坐标系统下的变换矩阵最后一行通常由 n 个 0 和 1 个 1 组成，这是为了保证变换后的结果 $\boldsymbol{p}'$ 的最后一个齐次分量仍为 1。而变换矩阵上方的左右两部分分别嵌入了旋转矩阵和平移向量。式(3.17)中旋转矩阵部分为单位阵，即表示没有旋转；而式(3.18)中平移向量部分为零向量，即表示没有平移。因此，如果既有旋转又有平移，那么变换矩阵上方应该既有旋转矩阵 $\boldsymbol{R}$，又有平移向量 $\boldsymbol{t}$：

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta + t_x \\ y\cos\theta + x\sin\theta + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \boldsymbol{p}' = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \boldsymbol{p} \tag{3.19}$$

式(3.19)即为刚体变换在齐次坐标系统下的矩阵表示，这将在3.1.6节中推导相机外参数时用到。

3.1.5 缩放、仿射与投影变换

图3.9展示了三种基本几何变换对图形的影响。3.1.3节和3.1.4节讨论了平移与旋转及二者的组合(刚体变换)，它们在变换中都不改变图形的大小和形状。本节将介绍一些可能会改变图形形状的缩放、仿射与投影变换。

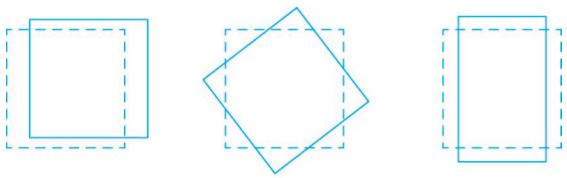


图3.9 三种基本几何变换对图形的影响

缩放变换将原图形在水平或竖直方向上拉长或缩短，相当于将原图形中所有点的横、纵坐标分别乘以一个缩放因子：

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \boldsymbol{p}' = \begin{bmatrix} \boldsymbol{S} & \boldsymbol{0} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \boldsymbol{p} \tag{3.20}$$

式中：s_x 为水平缩放因子；s_y 为竖直缩放因子。

可以想象，若 $s_x = s_y$，则相当于对原图形整体放大或缩小；若 s_x 或 s_y 为负数，则横坐标或纵坐标符号改变，对应图形会水平或竖直翻转。式(3.20)将在3.1.6节中推导相机内参数时用到。

式(3.20)仅包括沿两个轴向的缩放变换。更一般的缩放变换允许将图形沿任意方向缩放。缩放变换一般可由旋转变换和轴向缩放变换组成，如图3.10所示。下面提到

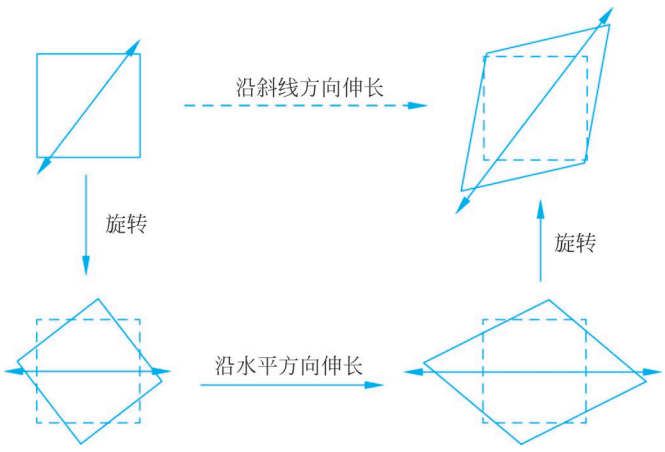


图3.10 缩放变换示意图

缩放变换时仍然指沿轴向的缩放变换。

平移、旋转、缩放这 3 种几何变换之所以称为简单几何变换，是因为一旦确定了图形中一个点的变化轨迹(起点与终点)，其他点的变化轨迹也会确定。也就是说，这 3 种变换各自可用两个对应的坐标点(变换前位置和变换后位置)表示。平移和旋转操作组成的刚体变换需要两对对应点表示。而平移、旋转、缩放 3 种变换组成的复合变换需要 3 对对应点确定，这种复合变换称为仿射变换。

一个正方形在仿射变换的作用下会变成一个任意形状的平行四边形；一个坐标系在仿射变换的作用下会变成另一个新的坐标系，这个新的坐标系不一定是直角坐标系，但每个坐标轴仍然是直线。仿射变换可能改变物体的形状，如两点间的距离、两条直线夹角大小，但两个平行线经过仿射变换后仍然是平行的。

平移、旋转、缩放这 3 种基本变换的任意组合都可以形成一个仿射变换，也就是说，任意一个仿射变换都可分解为这 3 种基本变换不限制次数和顺序的某种组合。显然，仿射变换在刚体变换的基础上进一步放宽了限制条件，即不再保证长度和角度不变。这意味着，仿射变换的变换矩阵具有比刚体变换更一般的形式：

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \boldsymbol{p}' = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{T} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \boldsymbol{p} \tag{3.21}$$

与式(3.19)比较可以发现，仿射变换与刚体变换的不同在于不再要求变换矩阵左上角为旋转矩阵 $\boldsymbol{R}$，而变成了一个任意矩阵 $\boldsymbol{A}$。这样左上角的矩阵就从 1 个自由度(转角 θ)变为 4 个自由度，即多出了缩放的信息。假设平面上的坐标系也经过仿射变换而形成一个新的坐标系，则矩阵 $\boldsymbol{A}$ 描述了新、旧两个坐标系对应轴之间的长短和夹角关系，而平移向量 $\boldsymbol{t}$ 描述了两坐标系原点间的位置关系。当 $\boldsymbol{A}$ 为正交矩阵(列向量都是单位向量且彼此正交)时，两个坐标系的对应轴之间的长短和夹角相等，相当于只进行了旋转，因此 $\boldsymbol{A}$ 退化为旋转矩阵 $\boldsymbol{R}$，此时整个仿射变换退化为一个刚体变换。

还有一种比仿射变换更一般的平面几何变换——二维投影变换(简称投影变换)：

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \propto \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \boldsymbol{p}' \propto \boldsymbol{H}\boldsymbol{p} \tag{3.22}$$

其变换矩阵 $\boldsymbol{H}$ 也称单应性矩阵。与式(3.21)比较可以发现，投影变换矩阵 $\boldsymbol{H}$ 在仿射变换的基础上又多了 h_{31} 和 h_{32} 两个自由度。此时矩阵 $\boldsymbol{H}$ 最后一行与 $\boldsymbol{p}$ 相乘的结果不一定为 1，因此式(3.22)中将等号改为了正比符号“$\propto$”。由于齐次坐标系统中 $\boldsymbol{p}'$ 整体乘以一个非零常数仍然表示点 $\boldsymbol{p}'$，因此“$\propto$”符号两端所差的常数并不重要。

投影变换不再保证平行线经过变换后仍然平行，它只能保证直线经过变换后仍然是直线。平面投影变换与相机对平面图像的成像过程密切相关：用相机从某一角度拍摄一个现实中的平面图片，得到的新图片与原图片之间的关系就是投影变换。或者说，用相机从不同位置、不同角度拍摄同一平面图片，得到的两幅图像之间的关系就是投影变换。

如图 3.11 所示，假设原图片是挂在墙上的一幅山峰的风景图，分别从左上方俯视拍摄和从下方仰视拍摄这幅图可以得到不同角度的两幅图像。这两幅图像之间及与原图片之间的变换都可以用投影变换描述。投影变换最大的特点是近大远小，可以看出距离相机越近的位置在拍出的照片中相对越大。另外，投影变换之后原本平行的两条线不一定平行，但原有的所有直线变换后仍然是直线。同时投影变换是保交点的，即原图片中的山峰位于矩形画框对角线交点处，拍摄得到的两幅图像中的山峰仍然位于四边形对角线的交点处。容易看出，一个投影变换可由 4 对对应点确定，如图片的 4 个角。3.2 节将根据从不同角度拍摄的同一平面图形，利用投影变换矩阵 $\boldsymbol{H}$ 计算相机参数。

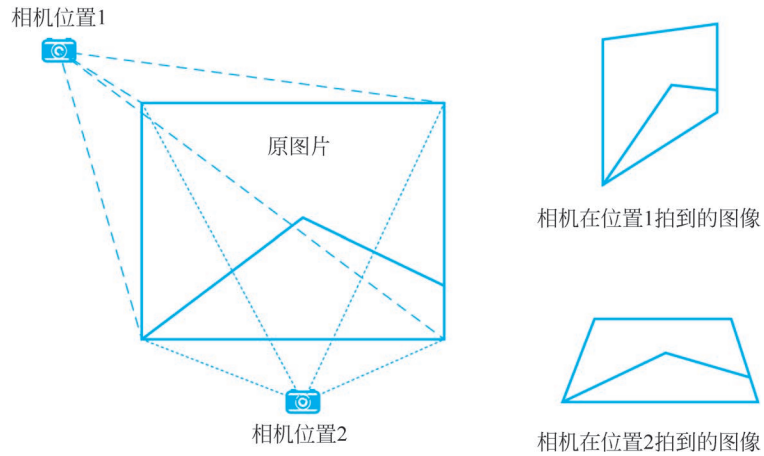


图 3.11　投影变换

3.1.6　相机参数

本节将推导世界坐标系、相机坐标系与图像坐标系之间的几何关系，为相机的几何标定做准备。根据 3.1.1 节介绍的针孔相机模型，三个坐标系之间的位置关系如图 3.12 所示。其中世界坐标系 $x_w y_w z_w$ 和相机坐标系 xyz 是三维空间直角坐标系，而图像坐标系 uv 是二维平面直角坐标系。相机坐标系的 z 轴表示物体的深度，也就是针孔相机模型中的主轴。

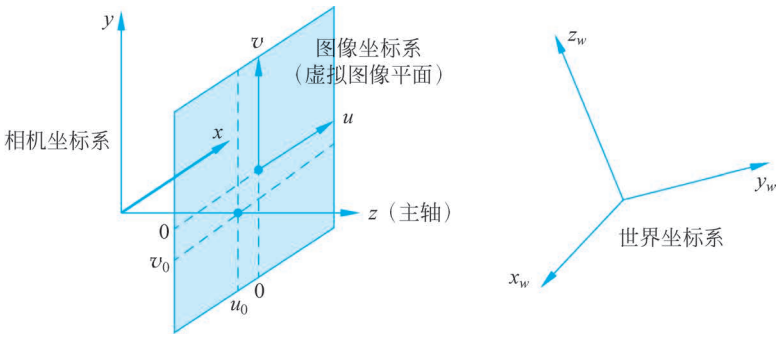


图 3.12　相机坐标系、图像坐标系与世界坐标系

注意，通常假定图像坐标系的 u、v 轴分别与相机坐标系的 x、y 轴平行。因为制造时存在的误差，图像坐标系的原点不一定位于相机坐标系的 z 轴上，这里将 z 轴在图像坐标系下的横纵坐标分别记为 u_0 和 v_0。另外，图像的像素不一定是严格的正方形，也就是说，u、v 两轴不一定等长。因此，需要两个参数分别描述同一点的相机坐标与图像坐标在横、纵两个方向上的比例关系。这里将图像平面上一点的 x、y 坐标相对于 u、v 坐标的缩放比例分别记为 s_x 和 s_y。

下面计算物点 $\boldsymbol{p}(x,y,z)$ 的像点 $\boldsymbol{p}'(x',y',z')$ 在图像坐标系下的坐标 (u,v)。

首先需要求像点 $\boldsymbol{p}'$ 点在相机坐标系下的坐标 (x',y',z')。如图 3.13 所示，已知 $\boldsymbol{p}'$ 点和 $\boldsymbol{p}$ 点在相机坐标系下的深度坐标分别为 z' 和 z，则根据三角形相似关系可得

$$y'=y\frac{z'}{z},\quad x'=x\frac{z'}{z} \tag{3.23}$$

式(3.23)是通过 $\boldsymbol{p}$ 点坐标计算 $\boldsymbol{p}'$ 点坐标，它们都是相机坐标系下的坐标。

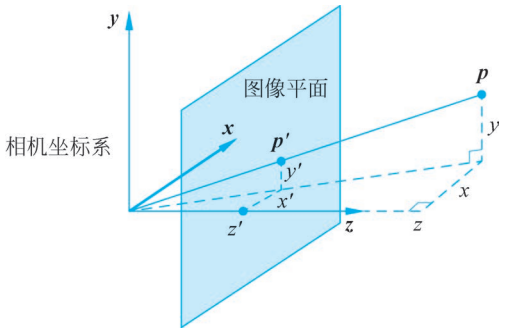


图 3.13　物点 p 与其像点 p' 坐标的相似关系

接下来需要根据 $\boldsymbol{p}'$ 点的相机坐标计算它的图像坐标，需要知道这两个坐标系间的变换关系。参考图 3.12，根据相机坐标系的 x、y 轴相对于图像坐标系的 u、v 轴有水平和竖直偏移 u_0 和 v_0，以及水平和竖直缩放比例 s_x 和 s_y，再仿照齐次坐标系统下的平移变换公式(3.17)和缩放变换公式(3.20)，容易写出 $\boldsymbol{p}'$ 点在两坐标系中的坐标转换关系：

$$\begin{bmatrix}u\\v\\1\end{bmatrix}=\begin{bmatrix}s_xx'+u_0\\s_yy'+v_0\\1\end{bmatrix}=\begin{bmatrix}s_x&0&u_0\\0&s_y&v_0\\0&0&1\end{bmatrix}\begin{bmatrix}x'\\y'\\1\end{bmatrix}=\begin{bmatrix}s_x&0&u_0\\0&s_y&v_0\\0&0&1\end{bmatrix}\begin{bmatrix}x'\\y'\\1\end{bmatrix} \tag{3.24}$$

可以发现，缩放矩阵和平移向量互不干扰，它们各自占据总体变换矩阵上方一部分。

式(3.23)与式(3.24)实际上是分两步将 $\boldsymbol{p}$ 点的相机坐标转化为 $\boldsymbol{p}'$ 点的图像坐标。观察式(3.23)可以发现，它实际上也是对 $\boldsymbol{p}$ 点的相机坐标进行一次横纵缩放比例均为 z'/z 的缩放变换，因此这个缩放变换可以一起放到式(3.24)变换矩阵的左上角：

$$\begin{bmatrix}u\\v\\1\end{bmatrix}=\begin{bmatrix}s_x\dfrac{z'}{z}&0&u_0\\0&s_y\dfrac{z'}{z}&v_0\\0&0&1\end{bmatrix}\begin{bmatrix}x\\y\\1\end{bmatrix}=\frac{1}{z}\begin{bmatrix}s_xz'&0&u_0\\0&s_yz'&v_0\\0&0&1\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix}$$

$$=\frac{1}{z}\begin{bmatrix} s_x z' & 0 & u_0 & 0 \\ 0 & s_y z' & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.25}$$

$\boldsymbol{p}$ 点实际上是三维坐标，因此在上式中变换矩阵右侧补充了一列，以便在右端向量中加入 z。由于齐次坐标整体乘以一个非零常数不改变其表示的点，因此整个变换矩阵与前面提出的系数 $1/z$ 无关，即与相机外的物点 $\boldsymbol{p}$ 的深度 z 无关。变换矩阵内的参数称为相机的内参数，意思是这些参数仅由相机内部构造决定。每台相机的内参数是固定的：u_0 和 v_0 是相机坐标系的 x、y 轴相对于图像坐标系的 u、v 轴的水平和竖直偏移，s_x 和 s_y 是相机坐标系 x、y 轴相对于图像坐标系 u、v 轴的水平和竖直缩放比例。z'是相机坐标系原点到物理成像平面或虚拟图像平面的距离，也称焦距（或像距），记为 f。由于 s_x 和 s_y 在变换矩阵中是与 z'乘在一起的，因此也可以将 $s_x z'$和 $s_y z'$当作两个相机内参数，分别记为 f_x 和 f_y。这样变换矩阵中只有 4 个内参数，即 f_x、f_y、u_0 和 v_0：

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \propto \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.26}$$

此外，相机内参数还包括镜头的畸变参数，具体细节将在 3.1.7 节介绍。

除内参数外，外参数也是指示相机状态的重要参数，其描述的是相机在现实世界中的位置与拍摄角度。由于相机坐标系是依据相机的位置和方向建立的，外参数也代表了相机坐标系与世界坐标系的关系。为简单起见，不妨认为这两个坐标系是全等的（各坐标轴长度相等的直角坐标系），这样两个坐标系之间就是一个简单的刚体变换。参考式(3.19)，很容易写出三维空间的刚体变换公式：

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}, \quad \boldsymbol{p} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix}\boldsymbol{p}_w \tag{3.27}$$

式中：$\boldsymbol{p}_w$ 为 $\boldsymbol{p}$ 点在世界坐标系下的坐标；$\boldsymbol{p}$ 为 $\boldsymbol{p}$ 点在相机坐标系下的坐标；$\boldsymbol{R}$ 为三维空间中的旋转矩阵；$\boldsymbol{t}$ 为三维空间中的平移向量。

旋转矩阵中的 9 个 r 和平移向量中的 3 个 t 组成了相机的外参数。

实际上三维旋转矩阵 $\boldsymbol{R}$ 中的 9 个 r 并不是互不相关的，它还受到单位正交性的约束，即旋转矩阵 $\boldsymbol{R}$ 的列向量组必须是单位正交向量组，才能表示一个旋转变换。如果把任意一个三维旋转看成依次绕 x、y、z 轴旋转一次，每一次旋转相当于给向量左乘一个旋转矩阵，就可以将其改写为只有 3 个自由度的形式：

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \boldsymbol{R}_z(\alpha)\boldsymbol{R}_y(\beta)\boldsymbol{R}_x(\gamma)$$

$$
\begin{aligned}
&= \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \\
&= \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix}
\end{aligned} \tag{3.28}
$$

式中：α、β 和 γ 是三个相互独立的角度参数，分别表示绕 z 轴、y 轴和 x 轴的旋转角度。

读者可以根据式(3.28)最后一行验证 $\boldsymbol{R}$ 的每一列是相互正交的单位向量。

将式(3.26)与式(3.27)连在一起即可得到所有参数和总体坐标转换关系：

$$
\begin{aligned}
\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &\propto \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}
\end{aligned} \tag{3.29}
$$

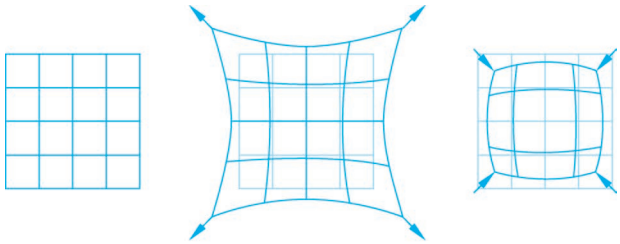
其中第二行是把第一行两个矩阵中无关紧要的一列和一行删掉得到的。第二行左侧的变换矩阵称为相机内参矩阵，右侧的变换矩阵称为相机外参矩阵。3.2 节将展示如何计算这两个矩阵中的所有参数。

3.1.7 镜头畸变

镜头在实际使用中并非理想的透视成像，现实世界中的物体变为图像后带有不同程度的畸变。理论上镜头的畸变主要包括径向畸变(极坐标系下极径 r 方向上的畸变)和切向畸变(极坐标系下转角 θ 方向上的畸变)，切向畸变影响较小，因此通常只考虑径向畸变。径向畸变是镜头的形状缺陷造成的，其主要由镜头径向曲率产生(光线在远离透镜中心的地方比靠近中心的地方更加弯曲)，导致真实成像点向内或向外偏离理想成像点。其中畸变像点相对于理想像点沿径向向外偏移，远离中心的，称为枕形畸变，常出现在远摄(长焦)镜头中；径向畸点相对于理想点沿径向向中心靠拢的，称为桶形畸变，常出现在广角镜头和鱼眼镜头中。这两种径向畸变的整体效果如图 3.14 所示。

图像径向畸变的存在不仅影响图像视觉效果，而且增加后续识别、跟踪等下游视觉任务的处理难度。从图 3.14 可以看出，枕形畸变和桶形畸变是互逆的过程，要想去除一幅图像中的畸变效果，可以再将此图像进行一次逆向畸变。

下面推导径向畸变过程的坐标变换公式。如图 3.15 所示，理想成像点 P 经过径向畸变后移动到 P'，其到图像坐标原点的距离从 r 变为 r'。在极坐标系下，从 r 到 r'的畸变过程可表示为



(a) 正常图像　(b) 枕形畸变　(c) 桶形畸变

图 3.14　径向畸变

$$r' = rf(r) \tag{3.30}$$

式中：$f(r)$为畸变造成的径向偏移系数。

由于径向畸变造成偏移的程度只与点 P 到原点的长度 r 有关而与方向无关，因此式(3.30)中的未知函数 f 是一个复杂的偶函数。可以用一个简单的多项式偶函数来近似表示复杂函数：

$$r' = r(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.31}$$

多项式函数 $1 + k_1 r^2 + k_2 r^4 + k_3 r^6$ 取到 6 次方是因为更高次数的项一旦有误差对结果的影响会很大；另外，目前这三项一般能满足函数拟合的要求。

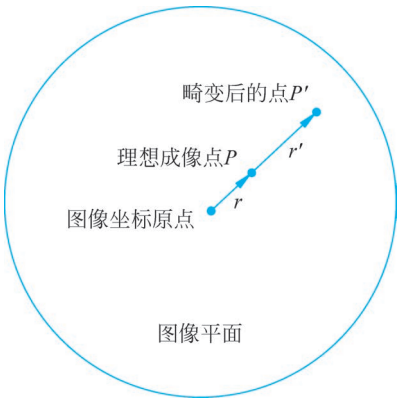


图 3.15　理想成像点 P 经过径向畸变后移动到 P'

在考虑镜头畸变的情况下，式(3.31)中的 $k_1 \sim k_3$ 也算作相机内参数。

以上是在极坐标下考虑，如果回到图像坐标系(平面直角坐标系 uv)下，那么式(3.31)可改写为

$$\begin{cases} u' = u(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ v' = v(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{cases} \tag{3.32}$$

式中：u、v 为畸变前的理想成像点的图像坐标；u'、v'为畸变后的成像点的实际图像坐标。

3.2 几何标定

本节将介绍相机的几何标定方法，即通过拍摄到的多个图像数据估计某个相机的内参数和外参数，包括式(3.29)中两个参数矩阵及式(3.32)中的三个镜头畸变系数。

观察式(3.29)可以发现，其中的所有相机参数都在两个紧挨着的参数矩阵内，而左右两侧的图像坐标和世界坐标可以通过实际拍摄一个物体计算得到。因此，很容易想到一种参数求解方法，即用同一个相机从不同位置、以不同角度拍摄同一个物体，得到许多个形如式(3.29)的等式，其中左右两端是已知的坐标，联立这些等式就可以求出中间的两个参数矩阵里的所有参数。

目前，针对普通相机最常用的几何标定方法是张正友标定法[4-6]，它是张正友博士提

出的一种利用平面棋盘格进行相机标定的实用方法。本节将详细讲解这种标定方法的步骤和原理。

3.2.1 张正友标定法

在张正友标定法提出之前，传统的相机标定方法主要分为主动标定法和自标定法两大类。

主动标定法（基于主动视觉的标定方法）是通过主动系统控制相机做特定运动，拍摄多组图像，依据图像信息和已知位移变化来求解相机内外参数。有两种典型的方法：一是使相机在三维空间稳定平移[7]；二是使相机做参数固定的旋转运动[8]。基于主动视觉的标定方法可以简化计算过程，并得到线性结果。这种方法的缺点是需要配备精准的控制平台，系统的成本高，而且在一些运动参数未知的场合和不能控制的场合不适用。

20 世纪 90 年代初，Luong 等首先提出自标定概念。自标定法并不需要知道图像点的三维坐标，该方法仅利用图像点之间的对应关系或约束关系而不需要标定物就可以得出标定系统的内、外参数，这就使得在相机任意运动或者一些复杂未知场景下的相机标定成为现实。目前，自标定的主要方法有直接求解 Kruppa 方程的自标定、分层逐步标定和基于绝对二次曲面的自标定等[9]。相机自标定方法不需要特殊制作的标定物，只是通过相机在无约束运动过程中建立多幅图像之间对应的关系直接进行标定。它克服了传统标定方法的缺点，灵活性较强；它的缺点是标定过程复杂，使用非线性方法标定，鲁棒性差，精度也不高。

张正友标定法介于主动标定法和自标定法之间，既克服了主动标定法需要结构较为复杂的高精度三维标定物的缺点，又解决了自标定法鲁棒性差的难题。标定过程仅需使用一个打印出来的棋盘格，并从不同方向拍摄几组图片即可，任何人都可以自己制作标定图案纸板（甚至直接用显示器显示），不仅灵活方便，而且精度高、鲁棒性好。因此，张正友标定法很快被全世界广泛采用，极大地促进了三维计算机视觉从实验室走向真实世界的进程。

张正友标定法所需要的图像数据是用待标定相机拍摄得到的不同位置、不同角度的多张平面棋盘格照片。为提高精度和鲁棒性，拍摄棋盘格照片越多越好，通常需要 10 张以上。同样为了提高计算精度，棋盘格需要占图片 1/4 以上的面积，同时需要保证棋盘格图像完整显示在照片中。棋盘格中的格子通常是四角相连、黑白交错的正方形（类似国际象棋的棋盘图案）。

最简单的方法是用平整的电脑显示器显示棋盘格图片，用待标定相机从不同角度进行拍照，图像采集结果如图 3.16 所示。

得到图像数据后，参数的计算方法可分为四大步骤，包括角点检测与配对、单应性矩阵求解、相机参数求解和相机参数细调。下面将按顺序依次介绍这些步骤。

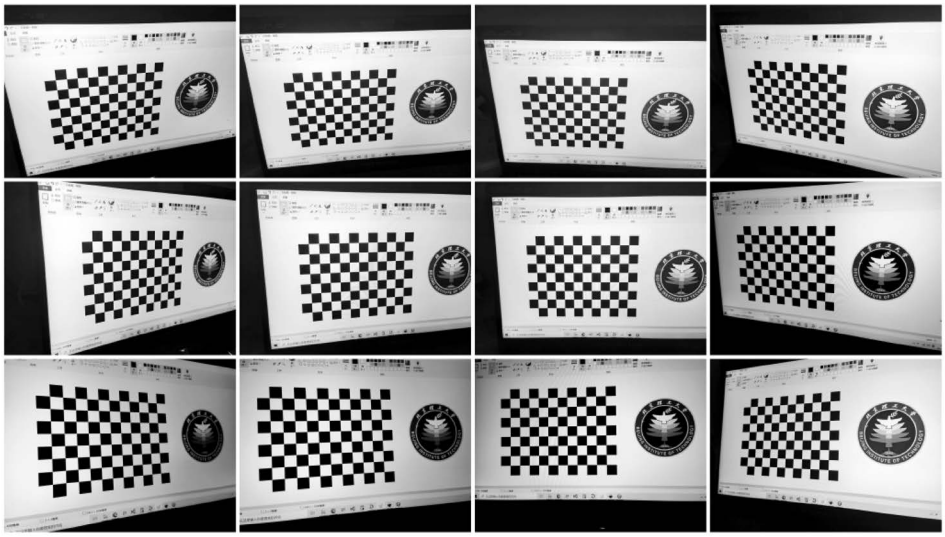


图 3.16　张正友标定法图像采集结果

3.2.2　角点检测与配对

角点检测与配对是指计算图像拍到的棋盘格中所有角点(黑白格的边界交点)的图像坐标及世界坐标,它们需要一一对应。世界坐标系并不是以长度定义的,而是人为选取的,因此可以将其单位长度设为与棋盘格的边长相同,以简化问题。由于现实世界中的棋盘格是一个平面图形,不妨设其上所有角点的 z_w 坐标为 0。因此,只需要所有角点在图像上的 u、v 坐标和世界中的 x_w、y_w 坐标。一旦找出了图像中所有角点,由于它们近似排布成一个方阵,很容易求出它是棋盘格图案上第几行第几列的角点,即其世界坐标 x_w、y_w。因此,该步骤的主要难度在于如何准确无误地从拍摄图像中提取棋盘格图案中的角点。张正友标定法相关论文中假定已完成角点配对,因此没有相关介绍。下面主要介绍 OpenCV 库中的实现方法。

为方便说明角点检测的原理,图 3.17 中对棋盘格做了简化,缩小了格子数量。需要注意的是,实际标定中使用的棋盘格行数与列数通常都大于 8,这样可以增加数据量,保证标定结果的精度。下面结合图 3.17 详细介绍棋盘格角点检测与配对算法。

图 3.17(a)为拍摄得到的原始图像。可以看出,由于拍摄时的光线问题,其中的棋盘格的白色格在图像上表现为灰色。为了更明显地区分黑白格并且减少数据量和计算量,需要把整个图像二值化,即把应该是黑色的地方变为全黑(如用 0 表示),把应该是白色的地方变为全白(如用 1 表示)。由于黑白图像的像素点亮度范围是 0～255,一个最简单的二值化方法就是超过 128 的像素点认为是白色,而不超过 128 的像素点认为是黑色,这样得到的结果如图 3.17(b)所示。可以看出,用简单的绝对阈值 128 分割得到的结果并不理想,下半部分的白格也被识别成黑格,因为其亮度恰好略低于 128。可以看出,由于光线问题,无法统一地给出一个合理的固定阈值来区分黑白,需要采用相对阈值的方

法。整幅图像很容易有光线充足以及不足的地方，但在一个局部范围内可以认为光线是相同的，因此在此范围内统计出所有像素的最大值，记为 v_{max}，再统计出所有像素的最小值，记为 v_{min}，最后把纯白与纯黑的中间值$(v_{max}+v_{min})/2$作为相对阈值来判别此区域所有像素应该是白色还是黑色。这样就可以得到最终的二值化图像，如图 3.17(c)所示。

接下来为了套用已有的多边形检测算法来识别棋盘格的所有黑格，需要把所有黑格之间的距离加大，这样每个黑格是一个独立的四边形，可以用多边形检测算法识别出来。这个过程称为黑格的“腐蚀”或者白格的“膨胀”。具体方法：针对图 3.17(c)中的每个像素点，判断其相邻的几个像素点中有没有白色，如果有白色，则其自身也变为白色。这样图像中所有白色像素点都向四周扩散，原本黑白格的交界处也变为白色，相当于黑色格子都缩小了。图 3.17(d)展示了白色膨胀操作的结果，现在所有黑格都是独立的四边形，相互之间没有交点。

然后对整张二值化图 3.17(d)进行多边形检测，即识别所有的黑白交界线，找到所有可以围成封闭多边形的边界线，得到图 3.17(e)。首先使用轮廓跟踪算法[10]对已二值化的图像进行轮廓提取(找到棋盘格边缘上的所有点)，再用 Douglas-Peucker 算法[11]对所有轮廓点进行多边形逼近。

再去除图 3.17(e)中所有非棋盘格的干扰多边形，得到图 3.17(f)。为简化算法，通常约定拍摄棋盘格照片时倾斜角度不能过大，这样棋盘格照片中的所有黑格应该近似为正方形(图 3.16)。由此可以删除掉一些各边长度与夹角差距过大或边数不是 4 的多边形，如图 3.17(e)中的五角星和五角星下面的不规则四边形(含有小于 60°的角和差距过大的边)。另外，在棋盘格以外的部分还会有一些无关的四边形留下来，如图 3.17(e)中右下角的正方形，接下来需要去除这些很像棋盘格的无关四边形。对于棋盘格中的四边形，根据图 3.17(e)中左半部分可以找到这样一些规律，每个四边形周围至少有另一个大小相近的四边形，且两个相邻四边形的距离远小于四边形的任意一个边长。将这个规律作为限定条件几乎可以再次去掉图 3.17(e)中所有无关四边形，最终得到图 3.17(f)。如果仍有遗漏，那么可以加入更复杂的几何限定条件，或重新选择更简单的场景拍摄照片。

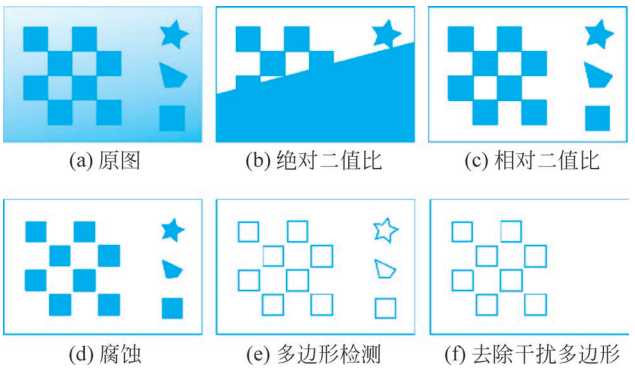


图 3.17　棋盘格角点检测算法的部分步骤

最后将图 3.17(f)中每两个相邻四边形的相邻顶点连上线段，则该线段的中点即为棋盘格的一个角点，如图 3.18 所示。

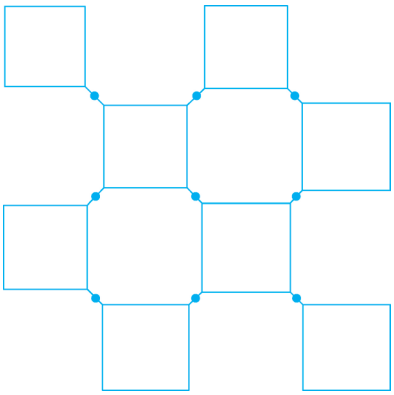


图 3.18　在图 3.17(f)基础上的角点检测结果(图中的每个圆点为一个棋盘格角点)

至此角点检测已经结束，但还需要确定棋盘格照片上的每个角点对应的世界坐标，即角点配对。由于拍照时有一定倾斜，棋盘上同一行的角点不一定在图像上的同一行，因此不能直接根据纵坐标判断一个角点在棋盘的哪一行。例如，图 3.19 中的情况，棋盘格上第二行的角点(1,6)在第一行的角点(0,0)的上方。

目前效果较好的棋盘格角点配对算法是 Geiger 等在 2012 年提出的方法[12]。该方法允许棋盘有很大倾斜且允许图片中有多张棋盘，但过程较为复杂、计算量很大。下面结合图 3.19 讲解一种简单的、适用于仅包含一个棋盘的图像且仅允许棋盘有小角度倾斜和扭曲的角点配对方法。

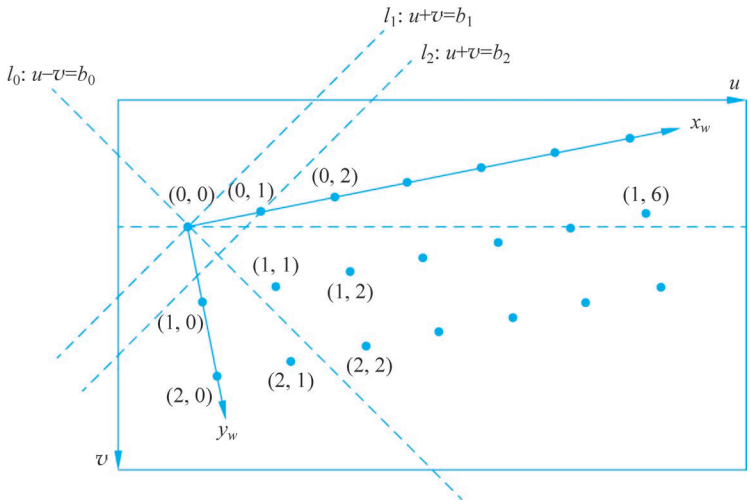


图 3.19　棋盘格角点配对简单算法示意

首先找到最左上角的角点作为世界坐标系的原点(0,0)。如图 3.19 所示，(x_w, y_w) 为世界坐标系，(u, v) 为图像坐标系(一般认为其原点在照片左上角，单位为像素)。根据解析几何知识，照片上越左上方的点其图像坐标之和越小，即 $u+v$ 的值越小。例如，图中角点(0,0)在直线 l_1 上，其 $u+v$ 值为 b_1；角点(0,1)在直线 l_2 上，其 $u+v$ 值为 b_2。由于 $b_1<b_2$，可以断定角点(0,0)在角点(0,1)的左上方。据此可以遍历所有角点，计算对应的 $u+v$ 值，具有最小 $u+v$ 值的角点定为世界坐标系的原点。

接下来寻找棋盘格第一行中的其他角点。与“左上方的点具有更小的 $u+v$ 值”类似，右上方的点具有更大的 $u-v$ 值。如图 3.19 所示，棋盘格上的角点(0,1)是在角点(0,0)

右上方的所有角点里最左上方的那个，因此可以遍历所有 $u-v$ 值大于 b_0 的角点，在其中选取 $u+v$ 值最小的即为角点(0,1)。

从左往右寻找第一行的每一个角点时，都将其前一个点所在的左上—右下方向上的直线作为 $u-v$ 值的分界线，在这条线右上方的点里找最左上方的点作为下一个角点，同时按顺序赋给世界坐标值。当这条分界线右上方没有其他点时，说明第一行已经遍历完。每一行遍历完后将其上所有点删除，继续以相同方法遍历下一行，直至没有其他剩余的角点。这样就完成了所有角点的“图像坐标-世界坐标”配对。

假设棋盘格的规格是 9 行 12 列，则一幅图像可以得到 $8\times11=88$ 个角点的图像坐标 u、v 和对应的世界坐标 x_w、y_w。根据这些对应坐标可以联立方程组来求解相机几何参数。

3.2.3 单应性矩阵求解

为了便于分析，将式(3.29)可写为

$$\begin{bmatrix}u\\v\\1\end{bmatrix}\propto\begin{bmatrix}f_x&0&u_0\\0&f_y&v_0\\0&0&1\end{bmatrix}\begin{bmatrix}r_{11}&r_{12}&r_{13}&t_x\\r_{21}&r_{22}&r_{23}&t_y\\r_{31}&r_{32}&r_{33}&t_z\end{bmatrix}\begin{bmatrix}x_w\\y_w\\z_w\\1\end{bmatrix}=\boldsymbol{A}\begin{bmatrix}\boldsymbol{R}&\boldsymbol{t}\end{bmatrix}\begin{bmatrix}x_w\\y_w\\z_w\\1\end{bmatrix}\tag{3.33}$$

式中：$\boldsymbol{A}$ 为相机内参矩阵；$[\boldsymbol{R}\quad\boldsymbol{t}]$ 是外参矩阵。

为化简问题，上一节中是以棋盘格平面作为 x_wy_w 平面建立世界坐标系的，因此棋盘格角点的 z_w 坐标均为 0。根据 z_w 为 0 可以将式(3.33)做简化：

$$\begin{aligned}\begin{bmatrix}u\\v\\1\end{bmatrix}&\propto\begin{bmatrix}f_x&0&u_0\\0&f_y&v_0\\0&0&1\end{bmatrix}\begin{bmatrix}r_{11}&r_{12}&r_{13}&t_x\\r_{21}&r_{22}&r_{23}&t_y\\r_{31}&r_{32}&r_{33}&t_z\end{bmatrix}\begin{bmatrix}x_w\\y_w\\0\\1\end{bmatrix}\\&=\begin{bmatrix}f_x&0&u_0\\0&f_y&v_0\\0&0&1\end{bmatrix}\begin{bmatrix}r_{11}&r_{12}&t_x\\r_{21}&r_{22}&t_y\\r_{31}&r_{32}&t_z\end{bmatrix}\begin{bmatrix}x_w\\y_w\\1\end{bmatrix}\\&=\boldsymbol{A}\begin{bmatrix}\boldsymbol{r}_1&\boldsymbol{r}_2&\boldsymbol{t}\end{bmatrix}\begin{bmatrix}x_w\\y_w\\1\end{bmatrix}\end{aligned}\tag{3.34}$$

上式删去了外参矩阵的第 3 列(旋转矩阵 $\boldsymbol{R}$ 的最后一列 $\boldsymbol{r}_3$)和世界坐标的第 3 行($z_w=0$)。设上式“$\propto$”符号两侧相差的比例系数为 λ，则上式可转化为

$$\begin{bmatrix}u\\v\\1\end{bmatrix}=\lambda\boldsymbol{A}\begin{bmatrix}\boldsymbol{r}_1&\boldsymbol{r}_2&\boldsymbol{t}\end{bmatrix}\begin{bmatrix}x_w\\y_w\\1\end{bmatrix}=\boldsymbol{H}\begin{bmatrix}x_w\\y_w\\1\end{bmatrix}=\begin{bmatrix}h_{11}&h_{12}&h_{13}\\h_{21}&h_{22}&h_{23}\\h_{31}&h_{32}&1\end{bmatrix}\begin{bmatrix}x_w\\y_w\\1\end{bmatrix}\tag{3.35}$$

可以发现，上式中的变换矩阵 $\boldsymbol{H}=\lambda\boldsymbol{A}\begin{bmatrix}\boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t}\end{bmatrix}$ 就是一个 2.2.3 节中介绍的投影变换的单应性矩阵(参见式(3.22))，其中包含了 $h_{11}\sim h_{32}$ 8 个未知数。选取合适的参数 λ 可以保证矩阵 $\boldsymbol{H}$ 右下角元素是 1。可以先根据角点配对步骤中得到的角点图像与世界坐标计算出单应性矩阵 $\boldsymbol{H}$，再利用单应性矩阵 $\boldsymbol{H}$ 求出相机内外参数。

将式(3.35)展开，可得

$$\begin{cases}(h_{31}x_w+h_{32}y_w+1)u=h_{11}x_w+h_{12}y_w+h_{13}\\(h_{31}x_w+h_{32}y_w+1)v=h_{21}x_w+h_{22}y_w+h_{23}\end{cases}\tag{3.36}$$

如果把 $h_{11}\sim h_{32}$ 当作未知数，那么将上式变形可得到关于 $h_{11}\sim h_{32}$ 的线性方程组：

$$\begin{cases}x_wh_{11}+y_wh_{12}+h_{13}-ux_wh_{31}-uy_wh_{32}=u\\x_wh_{21}+y_wh_{22}+h_{23}-vx_wh_{31}-vy_wh_{32}=v\end{cases}\tag{3.37}$$

上一步中得到了许多角点的图像坐标 u、v 和对应的世界坐标 x_w、y_w。对每个角点都可以写出如式(3.37)的一对方程。如果一共有 n 对角点，那么可以写出由 $2n$ 个方程组成的线性方程组：

$$\begin{bmatrix}x_{w1} & y_{w1} & 1 & 0 & 0 & 0 & -u_1x_{w1} & -u_1y_{w1}\\0 & 0 & 0 & x_{w1} & y_{w1} & 1 & -v_1x_{w1} & -v_1y_{w1}\\x_{w2} & y_{w2} & 1 & 0 & 0 & 0 & -u_2x_{w2} & -u_2y_{w2}\\0 & 0 & 0 & x_{w2} & y_{w2} & 1 & -v_2x_{w2} & -v_2y_{w2}\\\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\x_{wn} & y_{wn} & 1 & 0 & 0 & 0 & -u_nx_{wn} & -u_ny_{wn}\\0 & 0 & 0 & x_{wn} & y_{wn} & 1 & -v_nx_{wn} & -v_ny_{wn}\end{bmatrix}\begin{bmatrix}h_{11}\\h_{12}\\h_{13}\\h_{21}\\h_{22}\\h_{23}\\h_{31}\\h_{32}\end{bmatrix}=\begin{bmatrix}u_1\\v_1\\u_2\\v_2\\\vdots\\u_n\\v_n\end{bmatrix},\quad \boldsymbol{Ah}=\boldsymbol{b}\tag{3.38}$$

式中：$\boldsymbol{A}$ 为 $2n$ 行 8 列的矩阵；$\boldsymbol{h}$ 为 $\boldsymbol{h}_{11}\sim\boldsymbol{h}_{32}$ 组成的 8 行的列向量；$\boldsymbol{b}$ 为 $2n$ 行的列向量。

若 $n=4$，则 $\boldsymbol{A}$ 是一个 8 阶方阵，若此时 $\boldsymbol{A}$ 可逆，则有解 $\boldsymbol{h}=\boldsymbol{A}^{-1}\boldsymbol{b}$，这就是通过 4 组对应角点求解投影变换的方法。

然而，角点检测过程中难免有误差，故只能通过增加方程的个数来减少单个方程造成的误差，因此一个棋盘格上通常有 40 个以上的角点(远大于 4)。这样这个系数矩阵 $\boldsymbol{A}$ 的行数大于列数，方程个数大于未知量个数，此时这个方程称为超定方程。超定方程显然不存在精确解，因为方程比未知数还多，各方程之间通常有矛盾，超定方程显然不存在精确解。例如，方程 $x=0.99$ 与方程 $x=1.01$ 准确来说是相互矛盾的，但可以认为 $x=1$ 是同时满足这两个方程的近似最优解，因为它使得两个方程的误差之和最小。对于式(3.38)而言，希望找到一个 $\boldsymbol{h}$ 使得 $\boldsymbol{Ah}$ 与 $\boldsymbol{b}$ 的差距最小，也就是求解下面优化问题：

$$\arg\min_{\boldsymbol{h}}\|\boldsymbol{Ah}-\boldsymbol{b}\|\tag{3.39}$$

上式中的 $\|\boldsymbol{Ah}-\boldsymbol{b}\|$ 可以看成 $\boldsymbol{h}$ 的函数，为求它的最小值点，需要对这个函数求导。$\|\boldsymbol{Ah}-\boldsymbol{b}\|$ 是向量 $\boldsymbol{Ah}-\boldsymbol{b}$ 的长度，其内部包含一个开根号的操作，求导会非常麻烦，因此

将其简化。将求其本身的最小值点简化为求它的平方的最小值点：

$$\arg\min_{\boldsymbol{h}} \| \boldsymbol{Ah}-\boldsymbol{b} \|^2 \tag{3.40}$$

由于平方函数对于正变量是单调递增的，式(3.39)和式(3.40)的结果是一样的。式(3.40)是带有乘方的最小化问题，因此其结果称为超定方程 $\boldsymbol{Ah}=\boldsymbol{b}$ 的最小二乘解。

令 $f(\boldsymbol{h})=\| \boldsymbol{Ah}-\boldsymbol{b} \|^2$，令其导数为 $\boldsymbol{0}$ 得

$$f'(\boldsymbol{h})=\frac{\partial(\boldsymbol{Ah}-\boldsymbol{b})}{\partial \boldsymbol{h}}\frac{\partial \| \boldsymbol{Ah}-\boldsymbol{b} \|^2}{\partial(\boldsymbol{Ah}-\boldsymbol{b})}=\boldsymbol{A}^{\mathrm{T}}[2(\boldsymbol{Ah}-\boldsymbol{b})]=\boldsymbol{0} \tag{3.41}$$

其中用到了矩阵和向量形式的链式求导法则$\left(\frac{\partial g(f(\boldsymbol{x}))}{\partial \boldsymbol{x}}=\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}\frac{\partial g(f(\boldsymbol{x}))}{\partial f(\boldsymbol{x})}\right)$、线性函数求导公式$\left(\frac{\partial(\boldsymbol{Ax}-\boldsymbol{b})}{\partial \boldsymbol{x}}=\boldsymbol{A}^{\mathrm{T}}\right)$和平方函数求导公式$\left(\frac{\partial \| \boldsymbol{x} \|^2}{\partial \boldsymbol{x}}=2\boldsymbol{x}\right)$。可以发现，这些求导法则与标量函数的求导法则十分类似，只是需要注意乘法顺序(矩阵乘法不可交换)和转置的问题。更多矩阵求导的相关知识可以参考矩阵微分相关书籍[13]。

将式(3.41)继续整理可得

$$\begin{aligned}
&\boldsymbol{A}^{\mathrm{T}}(\boldsymbol{Ah}-\boldsymbol{b})=\boldsymbol{0}\\
&\boldsymbol{A}^{\mathrm{T}}\boldsymbol{Ah}-\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}=\boldsymbol{0}\\
&\boldsymbol{A}^{\mathrm{T}}\boldsymbol{Ah}=\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}\\
&\boldsymbol{h}=(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}
\end{aligned} \tag{3.42}$$

式(3.42)即为超定方程 $\boldsymbol{Ah}=\boldsymbol{b}$ 的最小二乘解。需要注意该式假定 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 可逆，即 $\boldsymbol{A}$ 列满秩。由于 $\boldsymbol{A}$ 的行数远大于列数，一般是可以保证 $\boldsymbol{A}$ 列满秩的。

当 $\boldsymbol{A}$ 不列满秩时，由于 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 不可逆，无法根据式(3.42)计算最小二乘解。此时需要用伪逆矩阵等方法来求解，读者可以通过查阅矩阵论或线性代数的相关书籍[14]来了解。

3.2.4 相机参数求解

3.2.3 节中得到了单应性矩阵 $\boldsymbol{H}$，本节将利用单应性矩阵 $\boldsymbol{H}$ 与相机内、外参数矩阵的关系求出相机的内参和外参。根据式(3.35)可得

$$\boldsymbol{H}=[\boldsymbol{h}_1 \quad \boldsymbol{h}_2 \quad \boldsymbol{h}_3]=\lambda\boldsymbol{A}[\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{t}] \tag{3.43}$$

式中：$\boldsymbol{h}_1$、$\boldsymbol{h}_2$ 和 $\boldsymbol{h}_3$ 为已经求出的单应性矩阵 $\boldsymbol{H}$ 的三个列向量。

将此式按列展开可得 $\boldsymbol{h}_1$、$\boldsymbol{h}_2$ 和 $\boldsymbol{r}_1$、$\boldsymbol{r}_2$ 之间的关系式：

$$\begin{cases}\boldsymbol{h}_1=\lambda\boldsymbol{A}\boldsymbol{r}_1\\ \boldsymbol{h}_2=\lambda\boldsymbol{A}\boldsymbol{r}_2\end{cases}\Rightarrow\begin{cases}\boldsymbol{r}_1=\frac{1}{\lambda}\boldsymbol{A}^{-1}\boldsymbol{h}_1\\ \boldsymbol{r}_2=\frac{1}{\lambda}\boldsymbol{A}^{-1}\boldsymbol{h}_2\end{cases} \tag{3.44}$$

旋转矩阵中各列向量彼此正交，即内积为 0；同时，各向量长度都是 1，即自己和自己的内积都等于 1。由此可以得到两个限制等式：

$$\begin{cases}\boldsymbol{r}_1^{\mathrm{T}}\boldsymbol{r}_2=0\\ \boldsymbol{r}_1^{\mathrm{T}}\boldsymbol{r}_1=1=\boldsymbol{r}_2^{\mathrm{T}}\boldsymbol{r}_2\end{cases} \tag{3.45}$$

将式(3.44)代入式(3.45)可得

$$\begin{cases} \boldsymbol{h}_1^{\mathrm{T}}(\boldsymbol{A}^{-1})^{\mathrm{T}}\boldsymbol{A}^{-1}\boldsymbol{h}_2 = 0 \\ \boldsymbol{h}_1^{\mathrm{T}}(\boldsymbol{A}^{-1})^{\mathrm{T}}\boldsymbol{A}^{-1}\boldsymbol{h}_1 = \boldsymbol{h}_2^{\mathrm{T}}(\boldsymbol{A}^{-1})^{\mathrm{T}}\boldsymbol{A}^{-1}\boldsymbol{h}_2 \end{cases} \tag{3.46}$$

此式是关于相机内参矩阵 $\boldsymbol{A}$ 的方程，$\boldsymbol{h}_1$ 和 $\boldsymbol{h}_2$ 是目前已知的单应性矩阵中的向量。每拍一个棋盘格照片就可以获得一个单应性矩阵 $\boldsymbol{H}$，为了求解内参矩阵 $\boldsymbol{A}$ 中的 4 个未知数，至少需要 2 张照片来得到 4 个方程。在实际的标定流程中，收集到的实验图像通常超过 10 张，可以通过足够多的方程来保证解的精确性。可以发现，式(3.46)也可以理解为是未知量为$(\boldsymbol{A}^{-1})^{\mathrm{T}}\boldsymbol{A}^{-1}$ 的方程组，可以先求出$(\boldsymbol{A}^{-1})^{\mathrm{T}}\boldsymbol{A}^{-1}$，再求出 $\boldsymbol{A}$。记$(\boldsymbol{A}^{-1})^{\mathrm{T}}\boldsymbol{A}^{-1}$ 为 $\boldsymbol{B}$，则有

$$\begin{aligned} \boldsymbol{B} = (\boldsymbol{A}^{-1})^{\mathrm{T}}\boldsymbol{A}^{-1} &= \begin{bmatrix} \frac{1}{f_x} & 0 & 0 \\ 0 & \frac{1}{f_y} & 0 \\ -\frac{u_0}{f_x} & -\frac{v_0}{f_y} & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{f_x} & 0 & -\frac{u_0}{f_x} \\ 0 & \frac{1}{f_y} & -\frac{v_0}{f_y} \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{{f_x}^2} & 0 & -\frac{u_0}{{f_x}^2} \\ 0 & \frac{1}{{f_y}^2} & -\frac{v_0}{{f_y}^2} \\ -\frac{u_0}{{f_x}^2} & -\frac{v_0}{{f_y}^2} & 1+\frac{{u_0}^2}{{f_x}^2}+\frac{{v_0}^2}{{f_y}^2} \end{bmatrix} \end{aligned} \tag{3.47}$$

$\boldsymbol{B}$ 是对称矩阵，可表示为

$$\boldsymbol{B} = \begin{bmatrix} b_{11} & 0 & b_{13} \\ 0 & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{bmatrix} \tag{3.48}$$

对任意的 $i \in \{1,2\}$和 $j \in \{1,2\}$，将二次型 $\boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{B}\boldsymbol{h}_j$ 展开可得

$$\boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{B}\boldsymbol{h}_j = \begin{bmatrix} h_{i1}h_{j1} \\ h_{i1}h_{j3}+h_{i3}h_{j1} \\ h_{i2}h_{j2} \\ h_{i2}h_{j3}+h_{i3}h_{j2} \\ h_{i3}h_{j3} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} b_{11} \\ b_{13} \\ b_{22} \\ b_{23} \\ b_{33} \end{bmatrix} \tag{3.49}$$

将式(3.49)右端记为$\boldsymbol{v}_{ij}^{\mathrm{T}}\boldsymbol{b}$，其中$\boldsymbol{v}_{ij}$ 和 $\boldsymbol{b}$ 都是 5 个元素的列向量，$\boldsymbol{v}_{ij}$ 可以由单应性矩阵的前两个列向量 $\boldsymbol{h}_1$、$\boldsymbol{h}_2$ 算得，$\boldsymbol{b}$ 是需要计算的对称矩阵 $\boldsymbol{B}$ 中所有未知量组成的列向量。这样就可以将式(3.49)简化成线性方程组：

$$\begin{bmatrix} \boldsymbol{v}_{12}^{\mathrm{T}} \\ (\boldsymbol{v}_{11}-\boldsymbol{v}_{22})^{\mathrm{T}} \end{bmatrix} \boldsymbol{b}=\boldsymbol{0} \tag{3.50}$$

由每张照片都可以得到如式(3.50)所示的一个线性方程,将所有照片对应的方程拼在一起可以得到超定方程组,采用求解单应性矩阵 $\boldsymbol{H}$ 时的最小二乘法即可求出最佳的 $\boldsymbol{b}$,从而得到只和相机内参相关的矩阵 $\boldsymbol{B}$。

得到 $\boldsymbol{B}$ 之后,根据下式可以逆向推出相机内参数 f_x、f_y、u_0、v_0:

$$\begin{cases} f_x=1/\sqrt{b_{11}} \\ f_y=1/\sqrt{b_{22}} \\ u_0=-b_{13}/b_{11} \\ v_0=-b_{23}/b_{21} \end{cases} \tag{3.51}$$

根据式(3.43)可计算得到相机外参矩阵:

$$\begin{cases} \boldsymbol{r}_1=\dfrac{1}{\lambda}\boldsymbol{A}^{-1}\boldsymbol{h}_1 \\ \boldsymbol{r}_2=\dfrac{1}{\lambda}\boldsymbol{A}^{-1}\boldsymbol{h}_2 \\ \boldsymbol{t}=\dfrac{1}{\lambda}\boldsymbol{A}^{-1}\boldsymbol{h}_3 \end{cases} \tag{3.52}$$

式中:$\lambda=\|\boldsymbol{A}^{-1}\boldsymbol{h}_1\|=\|\boldsymbol{A}^{-1}\boldsymbol{h}_2\|$,是由 $\boldsymbol{r}_1$ 和 $\boldsymbol{r}_2$ 为单位向量得到的。

最后,由于 $\boldsymbol{r}_3$ 是与 $\boldsymbol{r}_1$ 和 $\boldsymbol{r}_2$ 都正交的单位向量,其可以表示为 $\boldsymbol{r}_1$ 和 $\boldsymbol{r}_2$ 的外积(叉乘):

$$\boldsymbol{r}_3=\boldsymbol{r}_1\times\boldsymbol{r}_2 \tag{3.53}$$

至此已经求出了相机的内参数和外参数。注意式(3.51)是由所有标定照片共同决定的,一个相机的内参数是固定的;而式(3.52)和式(3.53)是针对其中每一张标定照片分别有一个解(每张照片的单应性矩阵 $\boldsymbol{H}$ 不一样),代表了拍这个照片时相机的位置和姿态。

相机几何标定的主要任务是获得相机的内参数(包括镜头畸变系数),上述过程中并没有考虑镜头的畸变,因此现在还需要考虑镜头的畸变(参见式(3.32)),整体求解一个非线性优化问题(自变量与因变量不是线性函数关系),其损失函数为

$$L=\sum_{i=1}^{n}\sum_{j=1}^{m}\|\boldsymbol{p}(i,j)-\boldsymbol{p}'(i,j)\|^2 \tag{3.54}$$

式中:$\boldsymbol{p}(i,j)$为第 i 张照片中第 j 个角点的真实图像坐标(畸变后的);$\boldsymbol{p}'(i,j)$为根据相机内、外参数计算出的第 i 个照片中第 j 个角点的图像坐标(畸变后的),它与第 j 个角点的世界坐标 $\boldsymbol{p}_{\mathrm{w}}(j)$和相机的一系列内、外参数相关,即

$$\boldsymbol{p}'(i,j)=F[\boldsymbol{p}_{\mathrm{w}}(j);f_x,f_y,u_0,v_0,k_1,k_2,k_3,\alpha(i),\beta(i),\gamma(i),t_x(i),t_y(i),t_z(i)] \tag{3.55}$$

式中的函数 F 根据相机的内、外参数将世界坐标 $\boldsymbol{p}_{\mathrm{w}}(j)$转化为预测的图像坐标 $\boldsymbol{p}'(i,j)$,其

具体形式可以根据式(3.29)和式(3.32)得到。

f_x、f_y、u_0、v_0、k_1、k_2、k_3 是固定的相机内参数，它们分别表示相机坐标系相对于图像坐标系的水平和垂直缩放系数，水平和垂直偏移系数，以及决定相机径向畸变的三个系数。

$\alpha(i)$、$\beta(i)$、$\gamma(i)$、$t_x(i)$、$t_y(i)$、$t_z(i)$是相机外参数，在每次拍照过程中是不同的，因此用 i 来表示它是哪张照片对应的参数。$\alpha(i)$、$\beta(i)$、$\gamma(i)$表示世界坐标系相对于相机坐标系的绕三个坐标轴的转角(见式(3.28))，而 $t_x(i)$、$t_y(i)$、$t_z(i)$表示世界坐标系相对于相机坐标系在三个坐标轴方向上的偏移。

为使估计出的参数更加接近真实情况，需要所有点的预测坐标图像 $\boldsymbol{p}'(i,j)$和真实图像坐标 $\boldsymbol{p}(i,j)$之间的总差距最小，即最小化损失函数 L。优化求解函数 L 的优化对象是所有的相机参数(式(3.55)分号右侧的所有参数)，而所有坐标点 $\boldsymbol{p}_w(j)$和 $\boldsymbol{p}(i,j)$是已知的常量。

这个非线性优化问题通常使用 Levenberg-Marquardt(LM)算法[15]进行迭代求解，它是非线性回归参数最小二乘估计的一种较好的方法。其首先需要设定所求参数的初始值，然后一步步迭代到能使损失函数 L 更小的参数值。所有参数的初始值的设定也非常重要，它决定了能否快速收敛到全局最优解，因此需要先用线性代数的方法粗略求出不考虑畸变时相机的内外参数。这些参数可作为迭代的初始值，而畸变系数 k_1、k_2、k_3 可以都初始化为 0。下一节将介绍 LM 算法，根据该算法的步骤可以求解相机内、外参数(包括畸变系数)使得总误差最小的、更为精确的解。

3.2.5 相机参数细调

本小节介绍相机参数细调常用的几种非线性优化方法的发展历程和具体计算步骤，包括梯度下降法、牛顿法、高斯-牛顿法和信赖域(Levenberg-Marquardt，LM)算法的发展历程，并介绍它们的具体计算步骤。

根据可微函数的最优化理论，求函数 $f(x)=0$ 的极小值点，可以令其导函数为 0，即 $f'(x)=0$，从而解出极小值点。通常情况下，这个方程难于求解或没有解析解(一个能用各种初等函数表示的公式形式的解)，因此通常采用迭代方法不断缩短当前点和极小值点的距离，从而一步步逼近极小值点。这种求解方法称为迭代法。

最简单的方法是梯度下降法，也称为最速下降法。对于多元函数 $g(\boldsymbol{x})=g(x_1,x_2,\cdots,x_n)$，其所有变量的偏导数组成的向量称为梯度：

$$\boldsymbol{G}(\boldsymbol{x})=\begin{bmatrix}\dfrac{\partial g}{\partial x_1}\\ \vdots\\ \dfrac{\partial g}{\partial x_n}\end{bmatrix} \tag{3.56}$$

式中：$\boldsymbol{G}$ 表示梯度(Gradient)。

梯度的几何意义是函数在某一点处斜率最大的方向，也就是说函数在某一点处沿梯

度方向是上升最快的。因此，为了从一个点更快地下降到一个使函数值更小的点，可以从当前点沿着负梯度方向走一定距离。这样可以写出梯度下降法的迭代公式：

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}-\mu\boldsymbol{G}(\boldsymbol{x}^{(k)}) \tag{3.57}$$

式中：带括号的上角标(k)表示该值是第 k 次迭代时的值；μ 为移动的步长(正实数)。

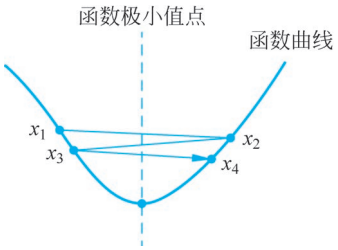


图 3.20　迭代步长选取过大时梯度下降法出现振荡现象(x_1、x_2、x_3 和 x_4 依次为四次迭代选取的最优点)

梯度下降法的主要缺点是每次移动的步长μ不好选取，取小了收敛得太慢，取大了又会跳过极小值点，出现迭代点在极小值点两侧来回振荡的现象，如图 3.20 所示。

为解决上述问题，牛顿提出使用函数二阶导数的信息在当前迭代点处拟合一个抛物面(原函数的二阶近似)，将抛物面的最小值点作为下一步最优点，这样就解决了步长该如何选取的问题。下面首先设 $\boldsymbol{x}$ 为当前最优点，$\boldsymbol{x}+\Delta\boldsymbol{x}$ 为能使损失函数 L 值更小的下一步最优点，可以把要极小化的目标函数 g(最小二乘法中损失向量函数 $\boldsymbol{f}$ 各分量的平方和)在 $\boldsymbol{x}$ 处泰勒展开出前三项：

$$\|\boldsymbol{f}(\boldsymbol{x}+\Delta\boldsymbol{x})\|^2=g(\boldsymbol{x}+\Delta\boldsymbol{x})\approx g(\boldsymbol{x})+\boldsymbol{G}(\boldsymbol{x})^{\mathrm{T}}\Delta\boldsymbol{x}+\frac{1}{2}\Delta\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}(\boldsymbol{x})\Delta\boldsymbol{x} \tag{3.58}$$

式中：$\boldsymbol{G}(\boldsymbol{x})$为目标函数 $g(\boldsymbol{x})$的梯度，即一阶导数；$\boldsymbol{H}(\boldsymbol{x})$为目标函数 $g(\boldsymbol{x})$的海森(Hessian)矩阵，即

$$\boldsymbol{H}(\boldsymbol{x})=\begin{bmatrix}\dfrac{\partial^2 g}{\partial x_1\partial x_1} & \dfrac{\partial^2 g}{\partial x_1\partial x_2} & \cdots & \dfrac{\partial^2 g}{\partial x_1\partial x_n}\\ \dfrac{\partial^2 g}{\partial x_2\partial x_1} & \dfrac{\partial^2 g}{\partial x_2\partial x_2} & \cdots & \dfrac{\partial^2 g}{\partial x_2\partial x_n}\\ \vdots & \vdots & \ddots & \vdots\\ \dfrac{\partial^2 g}{\partial x_n\partial x_1} & \dfrac{\partial^2 g}{\partial x_n\partial x_2} & \cdots & \dfrac{\partial^2 g}{\partial x_n\partial x_n}\end{bmatrix} \tag{3.59}$$

将式(3.58)最右端对 $\Delta\boldsymbol{x}$ 求导并令导数为 0，可得

$$\boldsymbol{G}(\boldsymbol{x})+\boldsymbol{H}(\boldsymbol{x})\Delta\boldsymbol{x}=\boldsymbol{0}\Rightarrow\Delta\boldsymbol{x}=-\boldsymbol{H}^{-1}(\boldsymbol{x})\boldsymbol{G}(\boldsymbol{x}) \tag{3.60}$$

由此得到牛顿法公式：

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}-\boldsymbol{H}^{-1}(\boldsymbol{x}^{(k)})\boldsymbol{G}(\boldsymbol{x}^{(k)}) \tag{3.61}$$

图 3.21 是在一维情形下的示意图，它展示了牛顿法的优势。图中 $\boldsymbol{x}_1$ 与 $\boldsymbol{x}_2$ 处的一阶导数相同，若用梯度下降法，则步长是相同的，就会导致 $\boldsymbol{x}_1$ 的下一步走过极值点或 $\boldsymbol{x}_2$ 的下一步走得不够远。而牛顿法不但用了一阶导数 $\boldsymbol{G}(\boldsymbol{x})$，而且利用了二阶导数 $\boldsymbol{H}(\boldsymbol{x})$的信息，其用一个与原函数有相同一阶导和二阶导的抛物线拟合当前所处的局部区域，并在一次迭代中直接下降到这个抛物线的最小值点，同时也接近原函数的极小值点。当在 $\boldsymbol{x}_1$ 处二阶导较大时，说明当前所在波谷的宽度较窄，因此选取较小的步长 $\Delta\boldsymbol{x}_1$；当在 $\boldsymbol{x}_2$

处二阶导较小时，说明当前所在波谷的宽度较宽，因此选取较大的步长 $\Delta \boldsymbol{x}_2$。在高维情形下也是类似的，牛顿法会将原函数曲面局部拟合成一个抛物面，并选取其最小值点作为下一步迭代点。

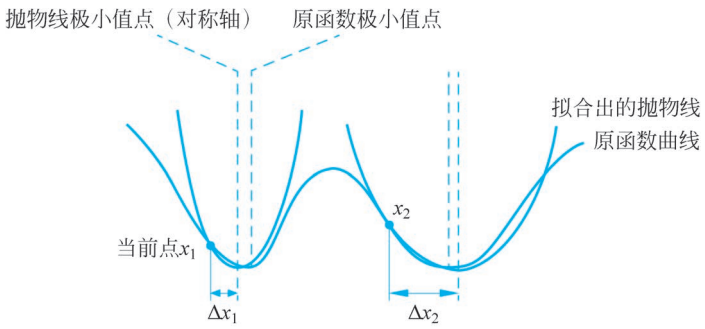


图 3.21　牛顿法根据二阶导合理选择步长

注：$f''(x_1)>f''(x_2)$，因此在 x_1 处拟合出的抛物线更窄，选取步长更小。

牛顿法收敛速度很快，但仍然存在两个问题：一是 Hessian 矩阵 $\boldsymbol{H}(\boldsymbol{x})$ 计算复杂，二是只有当 $\Delta \boldsymbol{x}$ 很小时泰勒展开式(3.58) 才有效，也就是说当步长很大时，根据式(3.61)计算出的 $\boldsymbol{x}^{(k+1)}$ 不一定是更好的点。

高斯针对此问题提出了高斯-牛顿法。针对第一个问题，高斯-牛顿法用向量函数 $\boldsymbol{f}$ 的雅可比矩阵的乘积 $\boldsymbol{J}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{x})$ 来近似代替 Hessian 矩阵 $\boldsymbol{H}$：

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}-[\boldsymbol{J}(\boldsymbol{x}^{(k)})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{x}^{(k)})]^{-1}\boldsymbol{J}(\boldsymbol{x}^{(k)})^{\mathrm{T}}\boldsymbol{f}(\boldsymbol{x}^{(k)}) \tag{3.62}$$

具体推导过程：首先将 $\boldsymbol{f}(\boldsymbol{x}+\Delta \boldsymbol{x})$ 在 $\boldsymbol{x}$ 处一阶泰勒展开，可得

$$\boldsymbol{f}(\boldsymbol{x}+\Delta \boldsymbol{x})\approx \boldsymbol{f}(\boldsymbol{x})+\boldsymbol{J}(\boldsymbol{x})\Delta \boldsymbol{x} \tag{3.63}$$

式中：$\boldsymbol{J}$ 为向量函数 $\boldsymbol{f}$ 的 m 行 n 列的雅可比矩阵，具有：

$$\boldsymbol{J}(\boldsymbol{x})=\begin{bmatrix}\dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} & \cdots & \dfrac{\partial f_1}{\partial x_n}\\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} & \cdots & \dfrac{\partial f_2}{\partial x_n}\\ \vdots & \vdots & \ddots & \vdots\\ \dfrac{\partial f_m}{\partial x_1} & \dfrac{\partial f_m}{\partial x_2} & \cdots & \dfrac{\partial f_m}{\partial x_n}\end{bmatrix} \tag{3.64}$$

将式(3.63)代入目标函数得

$$\begin{aligned}\|\boldsymbol{f}(\boldsymbol{x}+\Delta \boldsymbol{x})\|^2 &\approx \|\boldsymbol{f}(\boldsymbol{x})+\boldsymbol{J}(\boldsymbol{x})\Delta \boldsymbol{x}\|^2\\ &=[\boldsymbol{f}(\boldsymbol{x})+\boldsymbol{J}(\boldsymbol{x})\Delta \boldsymbol{x}]^{\mathrm{T}}[\boldsymbol{f}(\boldsymbol{x})+\boldsymbol{J}(\boldsymbol{x})\Delta \boldsymbol{x}]\\ &=\|\boldsymbol{f}(\boldsymbol{x})\|^2+2\boldsymbol{f}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{x})\Delta \boldsymbol{x}+\|\boldsymbol{J}(\boldsymbol{x})\Delta \boldsymbol{x}\|^2\end{aligned} \tag{3.65}$$

将等式最右端对 $\Delta \boldsymbol{x}$ 求导并令导数为 $\boldsymbol{0}$，可得

$$\begin{aligned}&2\boldsymbol{J}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{f}(\boldsymbol{x})+2\boldsymbol{J}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{x})\Delta \boldsymbol{x}=\boldsymbol{0}\\ \Rightarrow &\Delta \boldsymbol{x}=-[\boldsymbol{J}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{x})]^{-1}\boldsymbol{J}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{f}(\boldsymbol{x})\end{aligned} \tag{3.66}$$

因此，可以得到高斯-牛顿法迭代公式(3.62)。

高斯-牛顿法与牛顿法相比，简化了计算，但多了一个限制条件，即近似矩阵 $\boldsymbol{J}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{x})$ 必须可逆。针对此问题，Levenberg 和 Marquardt 提出了 LM 算法(也称为衰减最小二乘法)，其在近似矩阵后面加了一个修正项 $\mu\boldsymbol{I}$：

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}-[\boldsymbol{J}(\boldsymbol{x}^{(k)})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{x}^{(k)})+\mu\boldsymbol{I}]^{-1}\boldsymbol{J}(\boldsymbol{x}^{(k)})^{\mathrm{T}}\boldsymbol{f}(\boldsymbol{x}^{(k)}) \tag{3.67}$$

式中：μ 是每次迭代时可变的正实数，称为信赖域半径；$\boldsymbol{I}$ 为单位阵。

如果 μ 很小(接近 0)，那么 LM 方法的迭代公式与高斯-牛顿法的迭代公式(3.62)相同；如果 μ 很大，那么 $\boldsymbol{J}(\boldsymbol{x}^{(k)})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{x}^{(k)})$ 可以忽略，此时 LM 方法的迭代公式与梯度下降法的迭代公式(3.57)等效(证明留作习题)，而此时 μ 相当于每次梯度下降的步长。由此可见，LM 算法是梯度下降法和高斯-牛顿法的折中，它通过动态地改变参数 μ 使两种方法各自发挥其长处。

参数 μ 不仅影响了迭代方向，而且影响了步长。从图 3.21 可以看出，当牛顿法中拟合的抛物面很接近真实函数曲面时，牛顿法的表现很好。因此，可以定义一个信赖参数 ρ，用于表示拟合的抛物面与真实函数曲面的相似程度：

$$\rho=\frac{\boldsymbol{f}(\boldsymbol{x}+\Delta\boldsymbol{x})-\boldsymbol{f}(\boldsymbol{x})}{\boldsymbol{J}(\boldsymbol{x})\Delta\boldsymbol{x}} \tag{3.68}$$

其中分子是真实曲面上两点处的函数值之差，分母是拟合的曲面上对应的两点的函数值之差。显然，当 ρ 接近于 1 时，认为近似比较准确，此时可以把信赖域半径 μ 调小，让迭代公式(3.67)更接近高斯-牛顿公式；当 ρ 接近于 0 时，认为近似不准确，此时可以把信赖域半径 μ 调大，让迭代公式(3.67)更接近梯度下降公式。

另外，可以使用经验公式[16]选取 μ 的初始值：

$$\boldsymbol{A}^{(0)}=\boldsymbol{J}(\boldsymbol{x}^{(0)})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{x}^{(0)}),\quad \mu^{(0)}=\tau\max_i a_{ii}^{(0)} \tag{3.69}$$

式中：τ 为很小的正实数，如可以取 10^{-5}；$a_{ii}^{(0)}$ 为 $\boldsymbol{A}^{(0)}$ 的第 i 个对角线元素，也就是说 $\mu^{(0)}$ 通常取为初始近似 Hessian 矩阵 $\boldsymbol{J}(\boldsymbol{x}^{(0)})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{x}^{(0)})$ 的最大对角元素的 τ 倍。

具体来说，LM 算法步骤如下：

(1) 给定初始点 $\boldsymbol{x}^{(0)}$，根据式(3.69)计算初始信赖域半径 $\mu^{(0)}$；

(2) 在第 k 次迭代中，根据当前点 $\boldsymbol{x}^{(k)}$ 和当前信赖域半径 $\mu^{(k)}$ 以及迭代公式(3.67)计算出新的点 $\boldsymbol{x}^{(k+1)}$；

(3) 根据当前点 $\boldsymbol{x}^{(k)}$ 和新的点 $\boldsymbol{x}^{(k+1)}$ 以及式(3.68)求信赖参数 $\rho^{(k)}$；

(4) 根据 $\rho^{(k)}$ 的大小不同，调大或调小信赖域半径 $\mu^{(k)}$，得到新的信赖域半径 $\mu^{(k+1)}$；

(5) 重复步骤(2)～(4)直到达到最大迭代步数，或相邻两次迭代的点 $\boldsymbol{x}^{(k)}$ 和 $\boldsymbol{x}^{(k+1)}$ 足够接近，或 $\boldsymbol{J}(\boldsymbol{x}^{(k)})^{\mathrm{T}}\boldsymbol{f}(\boldsymbol{x}^{(k)})$ 足够小为止。

其中步骤(4)更新信赖域半径的具体方法目前较好的是 Nielsen(1999)策略[16]：

$$\mu^{(k+1)}=\begin{cases}\mu^{(k)}\max\left\{\dfrac{1}{3},1-(2\rho^{(k)}-1)^3\right\}, & \rho^{(k)}>0\\ 2\mu^{(k)}, & \rho^{(k)}\leqslant 0\end{cases} \tag{3.70}$$

在实际工程应用中各平台通常有对应的科学计算函数库，它们提供了求解各种非线性优化问题的高效函数，其中包含 LM 算法对应的函数。

如果使用 C++语言编程，那么可以使用谷歌开源的非线性优化库 Ceres Solver，它能够解决有约束或无约束条件下的非线性最小二乘问题。它被命名为 Ceres，是为了纪念高斯使用最小二乘法成功地预测了绕行至太阳背后的小行星 Ceres 的位置。

如果使用 Python 语言编程，那么可以使用开源数学工具包 SciPy，它是基于 Numpy 的科学计算库，用于数学、科学、工程学等领域，其涵盖了最优化、线性代数、数值积分、插值、特殊函数、快速傅里叶变换、信号处理和图像处理、常微分方程求解和其他科学与工程中常用的方法。

3.3 几何标定实践

几何标定是大部分计算机视觉任务的预备步骤，只有标定出相机的内、外参数和畸变系数后，才能对相机拍摄到的照片进行畸变矫正，并在此基础上进行其他操作。本节将讲解如何使用 Python 语言和 OpenCV 工具包进行相机标定和畸变矫正。

3.3.1 使用 Python 和 OpenCV 标定相机

以下是在 Python 中调用 OpenCV 库标定相机的完整代码，其中需要根据实际情况设置角点阵列的大小及图片存放路径。将使用待标定相机拍摄得到的 10 个以上的棋盘格照片(图 3.16)保存到同一个文件夹后，运行下面代码将弹出窗口显示角点检测结果，并输出相机内外参数和畸变系数。

```
import cv2                                  # 用于角点检测和标定(需安装 opencv-python 包)
import numpy as np                          # 用于矩阵运算(需安装 numpy 包)
import glob                                 # 用于从文件夹读取所有图片

# 只有以下 3 行设置需要根据情况修改
cx = 11                                     # 一行有多少角点
cy = 8                                      # 一列有多少角点
images = glob.glob("D:/calib/*.jpg")        # 图片路径("*"表示所有 jpg 文件名)

# 以下构造相机标定需要的"3D-2D"点对数据
# 每个照片里的 3D 点集 objp 中共 cx*cy 个角点,
# 每个角点有 x、y、z 共 3 个坐标,初始化为 0
objp = np.zeros((cx*cy, 3), np.float32)
# 将前两维坐标(x、y)赋值为棋盘格角点对应的列号和行号
objp[:, :2] = np.mgrid[0:cx, 0:cy].T.reshape(-1, 2)      # 存储所有图的 3D 点
# 将通过循环把所有图的 objp 连接起来存到 obj_points
obj_points = []

# 将通过角点检测函数 findChessboardCorners 依据照片自动计算 img_points
img_points = []                             # 用于存储所有图的 2D 点
```

```
    # 以下开始对每张图提取棋盘格角点、亚像素优化、显示结果
    # 用于设置亚像素角点优化的迭代结束条件,最大循环次数 30 和最大误差容限 0.001
criteria = (cv2.TERM_CRITERIA_MAX_ITER |
            cv2.TERM_CRITERIA_EPS, 30, 0.001)
for fname in images:                    # 遍历所有图片
    # 读取图片
    img = cv2.imread(fname)
    # 转灰度图("gray"是单通道图)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # 将图片的宽度和高度存到 size
    # 由于 gray.shape 是先行数后列数,需要倒过来才是先宽度再高度
    size = gray.shape[::-1]
    # 角点检测
    ret, corners = cv2.findChessboardCorners(gray, (cx, cy))
    if ret:                             # 如果此图角点检测成功
        print(
'图片', fname, '角点检测成功!')
        # 将此图的 3D 坐标数组 objp 数组存入 obj_points
        obj_points.append(objp)
        # 在原角点的基础上寻找亚像素角点
        corners2 = cv2.cornerSubPix(gray, corners, (5, 5), (-1, -1), criteria)
        # 将此图的 2D 亚像素角点数组存入 obj_points
        img_points.append(corners2)
        # 在图上绘制角点检测结果
        cv2.drawChessboardCorners(img, (cx, cy), corners, ret)
        # 小窗口预览角点检测结果(可更改窗口标题和宽高)
        cv2.imshow('Corner detection result',
                   cv2.resize(img, [1000, 600]))
        # 窗口停留 1s(1000ms)展示结果
        cv2.waitKey(1000)
# 关掉所有窗口
cv2.destroyAllWindows()

# 标定并显示结果
loss, A, dist, r, t = \
    cv2.calibrateCamera(obj_points, img_points, size, None, None)
print("\n 重投影误差:", loss)
print("相机内参矩阵 A: \n", A)
print("径向畸变系数 k1、k2、k3: \n", dist[0][0], dist[0][1], dist[0][4])
print("切向畸变系数 p1、p2: \n", dist[0][2], dist[0][3])
for i in range(len(r)):
    print('\n 第', i+1, '张照片对应的相机外参: ')
    print("绕 x、y、z 轴的旋转角度 α、β、γ: ",r[i][0][0], r[i][1][0], r[i][2][0])
    print("沿 x、y、z 轴的平移量 tx、ty、tz: ",t[i][0][0], t[i][1][0], t[i][2][0])
```

将图 3.16 所示的 12 张照片作为标定数据,运行以上的标定代码,输出的重投影误差、相机内参、畸变系数和第 1 幅图对应的相机外参如图 3.22 所示。

```
重投影误差: 0.9927880881911118
相机内参矩阵A:
 [[3.33309697e+03 0.00000000e+00 2.06352926e+03]
 [0.00000000e+00 3.34407116e+03 1.52283170e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
径向畸变系数k1、k2、k3:
 -0.0881655340406312 0.7193585052301302 -0.8422687340476115
切向畸变系数p1、p2:
 0.0013119130497566537 0.00012094318691297564

第 1 张照片对应的相机外参:
绕x、y、z轴的旋转角度α、β、γ: -0.39913431344443046 -0.5715875678253 79 3.0071220506100005
沿x、y、z轴的平移量tx、ty、tz: 3.599713380033281 4.531175874092325 22.639591729381902
```

图 3.22 运行标定代码得到的部分输出结果

图 3.22 中第一行输出的重投影误差是指按照计算出的内、外参数，将每个棋盘格角点从世界坐标系转换到图像坐标系之后，其与真实图像坐标的平均差距。接下来的内参矩阵模型与本章所讲的相同，其后的畸变模型不止考虑了径向畸变 k_1、k_2 和 k_3，还考虑了切向畸变 p_1 和 p_2。这是由于 OpenCV 的 calibrateCamera 函数求出的畸变系数共有 5 个，其在代码里 dist 数组中存储的顺序依次是 k_1、k_2、p_1、p_2 和 k_3。从结果可以看出，切向畸变通常很小，甚至可以忽略，因此本章前面的讲解过程中没有考虑切向畸变。

在畸变系数之后，代码会输出所有成功完成角点检测的图片所对应的相机外参数。例如，图 3.22 中给出的第 1 张照片对应的相机外参数，第一行是式(3.28)中的 α、β 和 γ，第二行是式(3.27)中的 t_x、t_y 和 t_z。它们确定了相机拍摄对应照片时在世界坐标系中的位置和角度，也就是相机相对于标定板的位置和角度。

此外，立体视觉相关应用中需要使用并排的两个相机模拟人眼同时观察目标物体，从而根据视差确定物体的距离，这就需要对双相机系统(双目相机)同时进行标定。一旦双相机系统中的两个相机的相对位置固定下来，它们之间的刚体变换也就确定下来，从而可以用两个相机的外参数计算出它们之间的刚体变换，对其中一个相机的位置和姿态进行矫正，最后使得两相机视线平行，方便对拍到的两幅图像进行双目匹配和深度估计。这些内容将在第 8 章“立体视觉”中介绍。

3.3.2 畸变矫正

得到相机的内参矩阵和畸变系数后，将其分别复制到下面代码的 A 和 dist 变量中，可以对原图片进行畸变矫正(包括径向畸变和切向畸变)，保存去畸变后的图像并在窗口中预览。

```
import cv2
import numpy as np

# 读取待去畸变图片
img = cv2.imread("D:/calib/input.jpg")
# 相机内参矩阵(使用标定程序的结果)
A = np.array([
        [3.33309697e+03, 0, 2.06352926e+03],
```

```
        [0, 3.34407116e+03, 1.52283170e+03],
        [0, 0, 1]], dtype=np.float32)
# 相机畸变系数(使用标定程序的结果)
dist = np.array(
    [-8.81655340e-02, 7.19358505e-01,          # 径向畸变系数 k1,k2
     1.31191305e-03, 1.20943187e-04,           # 切向畸变系数 p1,p2
     -8.42268734e-01],                         # 径向畸变系数 k3
    dtype=np.float32)
# 去畸变
img2 = cv2.undistort(img, mtx, dist)
# 保存去畸变后的图片
cv2.imwrite("D:/calib/output.jpg", img2)
# 小窗口预览去畸变结果(可更改窗口标题和宽高)
cv2.imshow("Undistorted result", cv2.resize(img2, [1000, 600]))
# 窗口停留,直到按下某按键或关闭窗口
cv2.waitKey()
```

只需要用 3.3.1 节给出的代码计算一次内参矩阵,之后就可以用以上代码对该相机拍到的所有照片进行畸变矫正,其效果如图 3.23 所示。可以看出,经过畸变矫正后的图像已经符合投影规律,即现实中的直线在图像中也是直线。

图 3.23　图像去畸变效果

本章小结

本章主要介绍了相机成像的几何模型以及使用相机进行摄影测量等计算机视觉任务的第一步——几何标定。只有知道了相机自身的几何参数之后,才能根据所拍摄到的图像来计算所拍摄物体的长度、位置和角度等信息。此外,相机直接拍摄到的图像通常具有畸变,为了得到正常的图像需要对原始图像进行畸变矫正,这同样需要对相机进行几何标定,得到相机镜头的各类畸变系数。

3.1.1 节首先介绍了针孔相机模型,光线经过小孔会成一个倒立的实像,像和物体之间呈现相似关系。3.1.2 节提到为了避免小孔成像亮度低、清晰度差的问题,之后发明出的相机使用透镜代替小孔,这样光源射向一定范围内不同方向的光线都可以会聚到成像平面上的同一点,形成更亮、更清晰的像。为了解决单一透镜成像形成的球形像差、彗形

像差和色差等各种像差，现代相机中使用组合镜头等更复杂的光学元件来代替单个透镜。

3.1.3节～3.1.5节介绍了二维平面中的一些常见几何变换，包括平移、旋转、缩放、仿射和投影变换。3.1.5节中引入了齐次坐标系统，将各种变换统一写成矩阵乘法的形式。可以发现使用矩阵不仅简化了表示形式，而且提供了一系列求解线性问题的基本定理和算法。读者可以借此机会熟悉用矩阵表示线性变换的思维方式，理解矩阵理论对现代科学技术的重要推进作用。

3.1.6节、3.1.7节依次推出了刚体变换、投影变换和径向畸变的具体公式，并引出了相机的内、外参数。3次变换整体可以看成一个从世界坐标映射到畸变后的图像坐标的函数，而这3次变换中依次包含了6个决定相机位置和角度的相机外参、4个与畸变无关的相机内参和3个与畸变相关的相机内参。要想求出这13个参数，就需要采集这个函数的多组输入和输出，列出关于这些参数的方程组进行求解。由于这个函数包含了径向畸变等非线性运算，难以写出公式形式的精确解，因此考虑将求解方程组的问题转化为一个最小二乘优化问题，通过迭代一步步找到能使函数值与实验值差距更小的13个参数。

3.2节以目前最常用的张正友标定法为例详细讲解了这个过程：

(1) 用待标定相机从不同角度拍摄一个棋盘格标定板，得到多张带有棋盘格角点的照片；

(2) 对每一张照片，使用相对二值化、腐蚀、多边形检测、无关多边形剔除等一系列操作提取其中的角点像素坐标，然后根据角点间的相对位置关系计算其对应的世界坐标；

(3) 先不考虑畸变，分别以各角点的世界坐标和图像坐标为输入和输出，联立线性方程组，求解它们之间的投影变换对应的单应性矩阵，也就是相机内参矩阵和外参矩阵的乘积；

(4) 根据乘积反推出相机的内、外参数(不包括畸变系数)的粗略解；

(5) 此时考虑畸变，以上一步求出的粗略解为相机内、外参数的初始值，并将各畸变系数的初值设为0，使用LM算法求解非线性优化问题(式(3.54))，得到相机内、外参数(包括畸变系数)的精确解。

在了解几何标定的原理后，读者可以尝试根据所讲内容编写代码实现相机的几何标定全流程。当然，开源计算机视觉库OpenCV中提供了现成的角点检测函数和相机标定函数，可以在Python、C++等编程语言中直接调用。3.3节以Python语言为例给出了使用OpenCV提供的函数实现相机标定的代码，之后给出了根据标定得到的各个参数实现照片畸变矫正的代码。

经过本章的学习，读者掌握了对可见光相机进行几何标定，以及对其拍到的图像进行畸变矫正的方法，这为之后的图像处理操作提供了合理的输入图像。此外，了解相机成像的几何模型同样会对立体视觉(第8章)等计算机视觉任务的学习有很大帮助。

习题

1. 依据针孔相机成像模型，若用 720×480 像素、光传感器阵列总大小为 3cm×2cm、焦距为 4cm 的相机拍摄距离镜头 5m、高度为 1m 的物体，该物体在照片上占多少像素高度？

2. 平面上某一点的齐次坐标可表示为(1,2,3)，写出其普通坐标表示。

3. 写出平面图形以原点为中心逆时针旋转 30°的齐次坐标变换公式。

4. 写出平面图形以 y 轴为对称轴水平翻转的齐次坐标变换公式。

5. 在二维平面中，保证 y 轴上的点位置不变，点(1,0)被移动到(1,k)，平面上任意一点向 y 轴正方向移动距离正比于该点的横坐标的几何变换称为错切比例为 k 的垂直错切变换。试推导其在齐次坐标系统下的变换矩阵。

6. 设二维平面上的四个点 $A(0,0)$、$B(2,0)$、$C(2,2)$、$D(0,2)$经过同一个投影变换分别被映射到 $A'(2,2)$、$B'(3,3)$、$C'(3,4)$、$D'(1,3)$。

(1) 求该投影变换对应的单应性矩阵。

(2) 求点 $P(1,1)$被该投影变换映射到的点 P'。

(3) 在平面直角坐标系中画出两个四边形 $ABCD$、$A'B'C'D'$和两个点 P、P'，观察计算结果是否合理。

7. 使用 OpenCV 标定某红外相机，得到径向畸变系数 k_1、k_2、k_3 分别为 −0.48、0.32、−0.13，求以原点为中心、边长为 0.8(图像宽度的 0.8)的正方形的四个顶点及各边中点经过畸变分别到了什么位置，并以此判断此畸变是属于枕形畸变，还是属于桶形畸变(假设主轴和畸变中心都在图像坐标原点)。

8. 证明当信赖域半径 μ 很大时，LM 算法迭代公式(3.57)与梯度下降法迭代公式(3.67)等效。

9. 参考 3.3 节的 Python 程序标定自己的手机摄像头，并将拍到的其他照片去畸变(建议在广角模式下采集标定数据和待矫正照片，去畸变的效果会比较明显)。

10. 本章所述的标定方法需要使用待标定相机拍摄棋盘格图案并识别其中的角点。红外相机拍摄到的图案只能区分不同温度的物体，而无法区分不同颜色的物体，因此无法从红外照片中锁定棋盘格角点。那么红外相机该如何标定？给出一个思路。

参考文献