相机标定

1 估计基础矩阵

基础矩阵原理：

根据针孔摄像机模型，我们可以知道，沿着三维点 X 和相机中心点之间的连线，可以在图像上找到对应的点 x。反过来，在三维空间中，与成像平面上的位置 x 对应的场景点可以位于这条线上的所有位置。这说明如果要根据图像中的一个点找到另一幅图像中对应的点，就需要在第二个成像平面上沿着这条线的投影搜索，这条线称为极线，在这里是 *l'*。另外，所有的对极线都通过同一个点，这个点成为极点，这是图中的 *e* 和 *e'*。那么这时，出来了一个矩阵 F，称为基础矩阵。

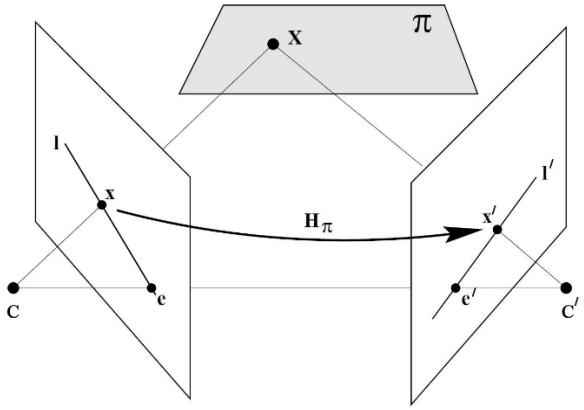

图 1 基础矩阵原理

一个场景中的一个空间点在不同视角下的像点存在一种约束关系，称为对极约束。基础矩阵就是这种约束关系的代数表示。它具体表示的是图像中的像点 p1 到另一幅图像对极线 12 的映射，有如下公式

$$l_2 = Fp_1$$

而和像点 p1 匹配的另一个像点 p2 必定在对集线 12 上，所以有

$$p_2^T l_2 = p_2^T F p_1 = 0$$

这样仅通过匹配的点对，就可以计算出两视图的基础矩阵 F。基础矩阵 F 是一个 3×3 的矩阵，有 9 个未知元素。然而，上面的公式中使用的齐次坐标，齐次坐标在相差一个常数因子下是相等，F 也就只有 8 个未知元素，也就是说，只需要 8 对匹配的点对就可以求解出两视图的基础矩阵 F。

计算基础矩阵的过程：

假设一对匹配的像点 p1=[u1,v1,1]T,p2=[u2,v2,1]T，带入式子中，得到：

$$[u_1, v_1, 1]\begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix}\begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = 0$$

把基础矩阵 F 的各个元素当作一个向量处理

$$f = [f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9]$$

那么上面式子可以写为，

$$[u_1u_2, u_1v_2, u_1, v_1u_2, v_1v_2, v_1, u_2, v_2, 1] \cdot f = 0$$

对于其他的点对也使用同样的表示方法。这样将得到的所有方程放到一起，得到一个线性方程组((ui,vi)表示第 i 个特征点）

$$\begin{bmatrix} u_1^1u_2^1 & u_1^1v_2^1 & u_1^1 & v_1^1u_2^1 & v_1^1v_2^1 & v_1^1 & u_2^1 & v_2^1 & 1 \\ u_1^2u_2^2 & u_1^2v_2^2 & u_1^2 & v_1^2u_2^2 & v_1^2v_2^2 & v_1^2 & u_2^2 & v_2^2 & 1 \\ u_1^3u_2^3 & u_1^3v_2^3 & u_1^3 & v_1^3u_2^1 & v_1^3v_2^1 & v_1^3 & u_2^3 & v_2^3 & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ u_1^8u_2^8 & u_1^8v_2^8 & u_1^8 & v_1^8u_2^8 & v_1^8v_2^8 & v_1^8 & u_2^8 & v_2^8 & 1 \end{bmatrix}\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = 0$$

求解上面的方程组就可以得到基础矩阵各个元素了。如果存在确定（非零）解，则系数矩阵 A 的自由度最多是 8。由于 f 是齐次矩阵，所以如果矩阵 A 的自由度为 8，则在差一个尺度因子的情况下解是唯一的。可以直接用线性算法解得。如果由于点坐标存在噪声则矩阵 A 的自由度可能大于 8（也就是等于 9，由于 A 是 n×9 的矩阵）。这时候就需要求最小二乘解，这里就可以用 SVD 来求解，F 的解就是系数矩阵 A 最小奇异值对应的奇异向量，也就是 A 奇异值分解后 A=UDVT 中矩阵 V 的最后一列矢量，这是在解矢量 f 在约$\|f\|$下取$\|Af\|$最小的解。以上算法是解基本矩阵的基本方法，称为 8 点算法。

由于基本矩阵有一个重要的特点就是奇异性，F 矩阵的秩是 2。如果基本矩阵是非奇异的，那么所计算的对极线将不重合。所以在上述算法解得基本矩阵后，会增加一个奇异性约束。最简便的方法就是修正上述算法中求得的矩阵 F。设最终的解为 F′，令 detF′ =0 下求得 Frobenius 范数（二范数）$\|F - F'\|$最小的 F′ 。这种方法的实现还是使用了 SVD 分解，若 F=UDVT，此时的对角矩阵 D=diag(r,s,t)，满足 r≥s≥t，则 F′ =Udiag(r,s,0)VT 最小化范数$\|F - F'\|$，也就是最终的解。

完成代码如下：

```
M = np.zeros((matches.shape[0], 9))
x1, x2, y1, y2 = matches[:, 0].T, matches[:, 2].T, matches[:, 1].T, matches[:, 3].T
for i in range(matches.shape[0]):
    M[i] = [x1[i] * x2[i], x1[i] * y2[i], x1[i], y1[i] * x2[i], y1[i] * y2[i], y1[i], x2[i], y2[i], 1]
U, S, V = np.linalg.svd(M)
F_ori = V[-1].reshape(3, 3)
U_, S_, V_ = np.linalg.svd(F_ori)
S_[2] = 0
A = np.dot(np.diag(S_), V_)
F = np.dot(U_, A)
```

本次作业计算得到的基础矩阵为：

```
fundamental matrix
[[-5.36264198e-07  8.83539184e-06 -9.07382264e-04]
 [ 7.90364771e-06  1.21321685e-06 -2.64234650e-02]
 [-1.88600204e-03  1.72332901e-02  9.99500092e-01]]
```

极线的绘制：

首先用其次坐标表示点。一个点坐标换成齐次坐标表示可以简单理解为在最后加一个维度，该维度值为 1。例如，设有一个坐标

$$(x,y)->(x,y,1)$$
$$(x,y,z)->(x,y,z,1)$$

在二维平面上，一条直线 l 可以用直线方程 ax+by+c=0 来表示（这里 x，y 是变量），我们还可以用向量来表示一条直线：

$$l=(a,b,c)^T$$

已知一个具体位置的点 p(x0,y0)，若该点在直线 l 上，则有

$$ax_0+by_0+c=0$$

当改用齐次坐标表示点 p(x0,y0,1)，那么该点在直线 l 上的表示可以用向量表述：

$$pl=0$$

因此，改用齐次坐标表示一个点后，判断该点在不在一条直线上的方式，变成判断等式 pl=0 是否成立，或者说点 p 与 l 的内积是否为 0。

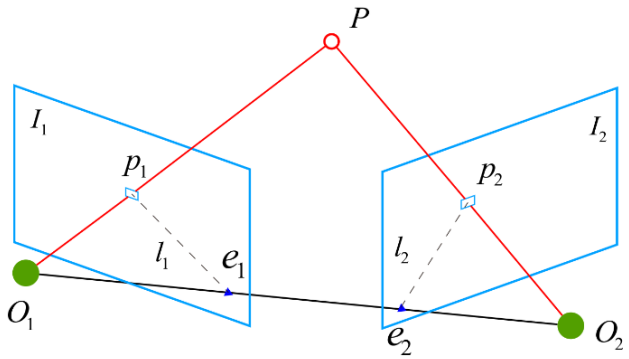

图 2 极线的绘制

对于

$$p_1 = (u_1, v_1, 1)^T, p_2 = (u_2, v_2, 1)^T$$

基本矩阵 F ，根据对极约束有

$$p_2^T F p_1 = 0$$

利用齐次坐标中点在直线上的表示方法，可以知道点 p2 在直线 Fp1 上，因此直线 Fp1 就是极线，且该直线为上图中的极线 12 。通过类比可得另一条极线为

$$l_1 = F^T p_2$$

通过上面推倒的公式计算出齐次坐标下的极线，再将几个特殊点带入极线方程中，得到截距坐标，通过其将极线绘制出来。

极线计算和绘制实现代码如下（以极线 12 为例）：

```
one_ = np.array([[1.]])
for i in range(len(pt_2d)):
    p_i = np.r_[pt_2d[i].reshape(2, 1), one_]
    if if_pt2:line_homogeneous = np.dot(F.T, p_i)
    else:line_homogeneous = np.dot(F, p_i)
    y = -line_homogeneous[2] / line_homogeneous[1]
    x = (-line_homogeneous[2]) / line_homogeneous[0]
    x1, y2 = 0, 0
    if x < 0:
        y2 = 712
        x = (-line_homogeneous[2] - line_homogeneous[1] * 712) / line_homogeneous[0]
    if y < 0:
        x1 = 1072
        y = (-line_homogeneous[2] - line_homogeneous[0] * 1072) / line_homogeneous[1]
    ax.plot([x1, x], [y, y2], 'g')
```

绘制出的极线效果：

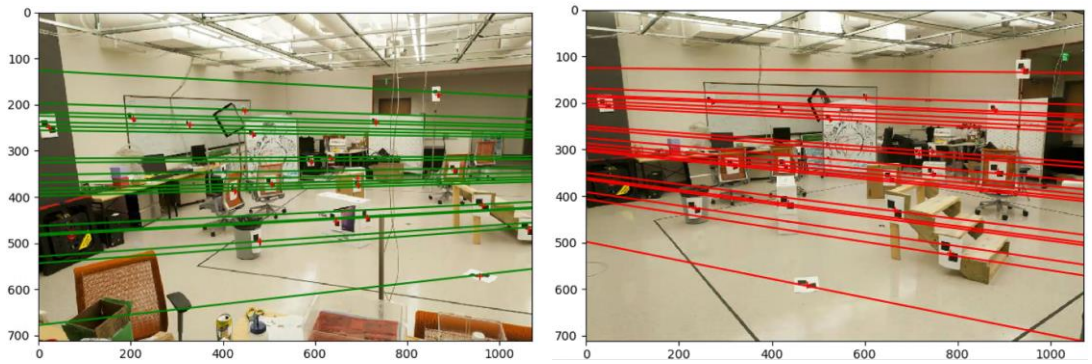

图 3 极线绘制效果

2 相机标定

概念和基本原理：

摄像机标定(Camera calibration)简单来说是从世界坐标系换到图像坐标系的过程，也就是求最终的投影矩阵 P 和相关参数的过程。
基本的坐标系有：世界坐标系(world coordinate system)；相机坐标系(camera coordinate system)；图像坐标系(image coordinate system)。一般来说，标定的过程分为两个部分：
1.从世界坐标系转换为相机坐标系，这一步是三维点到三维点的转换，包括 R，t（相机外参）等参数；
2.从相机坐标系转为图像坐标系，这一步是三维点到二维点的转换，包括 K（相机内参）等参数。

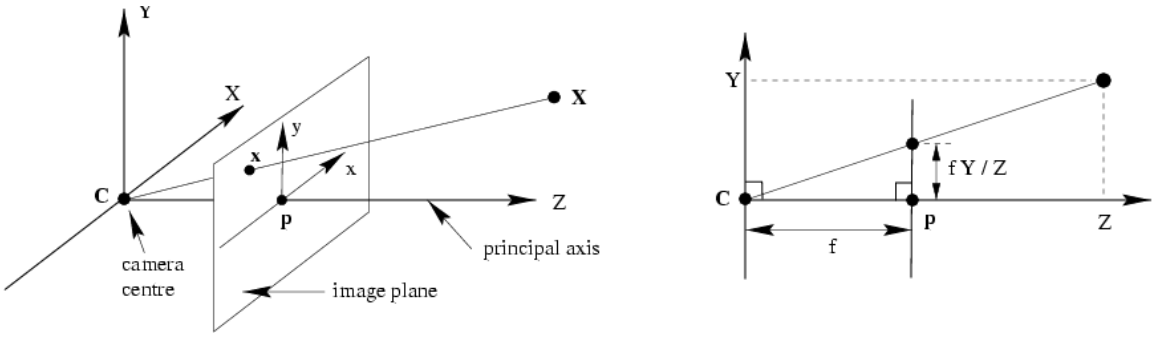

图 4 相机标定坐标系

相机坐标系转换到图像坐标系：
如果知道相机坐标系中的一个点 X（现实三维世界中的点），在像平面坐标系对应的点是 x,要求求从相机坐标系转为像平面坐标系的转换,也就是从 X 点的(X,Y,Z)通过一定的转换变为 x 点的(x,y)。注意：(X,Y,Z)（大写）是在相机坐标系，而(x,y)（小写）是在像平面坐标系（还不是图像坐标系，原点不同。）观察第二个图，很简单的可以得到这个转换：

$$x = fX/Z$$

$$y = fY/Z$$

$$(X,\ \ Y,\ \ Z) \mapsto (fX/Z,\ \ fY/Z)$$

可以表示为矩阵计算为

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

简写为

$$x = PX$$

通过上式，可以把相机坐标系转换到像平面坐标系，但是像平面坐标系和图像坐标系虽然在同一个平面上,但是原点并不是同一个,而目标是要转换到图像坐标系下，所以还需要一步操作，如下图

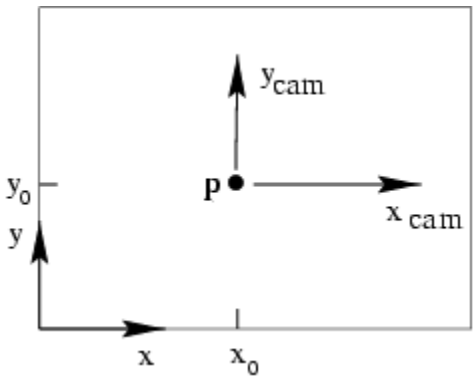

图 5 偏移量

其中主点 p 是像平面坐标系的原点,但在图像坐标系中的位置为(px,py),在这里，图形坐标系的原点是图片的左下角，所以可以得到：

$$(X,\ \ Y,\ \ Z) \mapsto (fX/Z + p_x,\ \ fY/Z + p_y)$$

相当于在上面的基础上加了一个 p 点坐标的偏移量，同时可以表示为矩阵计算为

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

整理得

$$\begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

所以最后可以得到 K，也就是相机内参（Intrinsic parameters）：

$$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}$$

以及投影矩阵 P

$$P = K \begin{bmatrix} I \mid 0 \end{bmatrix}$$

转换到像素坐标系统

$$K = \begin{bmatrix} m_x & & \\ & m_y & \\ & & 1 \end{bmatrix} \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & & \beta_x \\ & \alpha_y & \beta_y \\ & & 1 \end{bmatrix}$$

其中，m_x 表示在水平方向 1m 的长度包含的像素的个数，m_y 表示在竖直方向 1m 的长度包含的像素的个数。一般来说，在使用相机内参 K 计算坐标系转换时，提供的

都是已经变换后的值，例如会提供 f_x, f_y , c_x, c_y 四个值代表相机内参 K, 其实 f_x 就是这里的 α_x, 同理 f_y 是 α_y, c_x 是 β_x, c_y 是 β_y。

世界坐标系转换到图像坐标系：

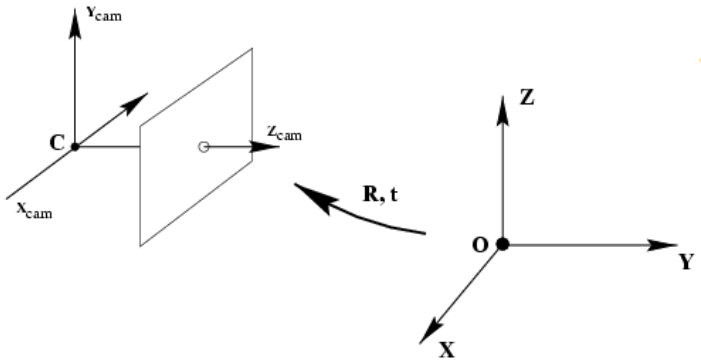

图 6 世界坐标系转换到图像坐标系

从世界坐标系转换到相机坐标系是三维空间到三维空间的变换，一般来说需要一个平移操作和一个旋转操作就可以完成这个转换，用公式表示如下

$$\widetilde{X}_{cam} = R(\widetilde{X} - \widetilde{C})$$

其中，R 表示旋转矩阵，$\widetilde{X}$表示 X 点在世界坐标系中的位置，$\widetilde{C}$表示相机原点 C 在世界坐标系中的位置，$\widetilde{X}_{cam}$ 表示 X 点在相机坐标系中的位置。根据上面的公式可以得到从一个三维点从世界坐标系转换到相机坐标的变换公式如下

$$X_{cam} = \begin{pmatrix} \widetilde{X}_{cam} \\ 1 \end{pmatrix} = \begin{bmatrix} R & -R\widetilde{C} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} \widetilde{X} \\ 1 \end{pmatrix} = \begin{bmatrix} R & -R\widetilde{C} \\ 0 & 1 \end{bmatrix} X$$

所以带入上面的矩阵计算，可以得到

$$x = K \left[\; I \; \middle| \; 0 \; \right] X_{cam} = K \left[\; R \; \middle| \; -R\widetilde{C} \; \right] X$$

这样就得到了最终的投影矩阵 P ：

$$P = K \left[\; R \; \middle| \; t \; \right]$$

其中，

$$t = -R\widetilde{C}$$

K 称为相机内参（intrinsic parameters），描述了相机的内部参数，包括焦距 f、主点 p 的位置、以及像素与真实环境的大小比例等，这个是固有属性，是提供好的；R 和 t 称为相机外参（extrinsic parameters），R 在这里是旋转矩阵，可以转换为三维的旋转向量，分别表示绕 x, y, z 三个轴的旋转角度，t 目前就是一个平移向量，分别表示在 x, y, z 三个方向上的平移量。

利用投影矩阵计算相机内参和外参的代码实现如下：

```
f = []
for n in range(matches.shape[0]):
    x, y, z, u, v = pt_3d[n,0], pt_3d[n,1], pt_3d[n,2], matches[n,0], matches[n,1]
    f.append([x,y,z,1,0,0,0,0, -u*x, -u*y, -u*z, -u])
    f.append([0, 0, 0, 0, x, y, z, 1, -v * x, -v * y, -v * z, -v])

w, v = np.linalg.eig(np.dot(np.mat(f).T, np.mat(f)))
P = np.reshape(v[:, np.argsort(w)[0]],(3,4))
c = np.dot(np.linalg.inv(-P[:, 0:3]), P[:, 3])
R, K = np.linalg.qr(P[:,:3])
```

计算结果中，lab1 视角的投影矩阵 P， 内参矩阵 K、旋转矩阵 R 与相机中心c 分别为：

```
lab 1 camera projection P
 [[ 3.09963986e-03  1.46205641e-04 -4.48500867e-04 -9.78930667e-01]
 [ 3.87018218e-04  6.37193851e-04 -2.77356157e-03 -2.04144461e-01]
 [ 1.67933489e-06  2.74767742e-06 -6.83967566e-07 -1.32882924e-03]]

lab 1 intrinsic matrix K
 [[-3.11480743e-03 -2.08301626e-04  7.19698877e-04]
 [ 0.00000000e+00 -6.19685465e-04  2.71582463e-03]
 [ 0.00000000e+00  0.00000000e+00  1.12539041e-05]]

lab 1 rotation matrix R
 [[-9.95130239e-01  9.85687239e-02 -1.17332262e-04]
 [-9.85673192e-02 -9.95121159e-01 -4.28519128e-03]
 [-5.39145653e-04 -4.25275830e-03  9.99990812e-01]]

lab 1 camera center c
 [[305.83277023]
 [304.28104019]
 [ 30.13699218]]
```

Lab2 视角的投影矩阵 P， 内参矩阵 K、旋转矩阵 R 与相机中心c 分别为：

```
lab 2 camera projection P
 [[ 6.93155458e-03 -4.01685454e-03 -1.32601857e-03 -8.26700168e-01]
 [ 1.54768962e-03  1.02452778e-03 -7.27441321e-03 -5.62523823e-01]
 [ 7.60946956e-06  3.70953803e-06 -1.90282064e-06 -3.38807962e-03]]

lab 2 intrinsic matrix K
 [[-7.10224260e-03  3.69705300e-03  2.87936071e-03]
 [ 0.00000000e+00 -1.87525745e-03  6.81056935e-03]
 [ 0.00000000e+00  0.00000000e+00  2.90414574e-05]]

lab 2 rotation matrix R
 [[-9.75967025e-01  2.17918202e-01 -1.54288721e-04]
 [-2.17915622e-01 -9.75958465e-01 -4.22561524e-03]
 [-1.07141786e-03 -4.09043922e-03  9.99991060e-01]]

lab 2 camera center c
 [[303.10003555]
 [307.18428059]
 [ 30.42166819]]
```

与真实的相机投影点比较，平均距离均小于 20：

```
residuals between the observed 2D points and the projected 3D points:
residual in lab1: 13.5457449560554
residual in lab2: 15.545064148671402
```