SLAM14讲—07视觉里程计1

7.1特征点法

前端即视觉里程计（VO），根据相邻图像的信息，估计出粗略的相机运动，给后端提供较好的初始值。基于特征点法的前端，运行稳定，对光照、动态物体不敏感，是目前比较成熟的解决方案。

7.1.1特征点

首先，从图像中选取比较**有代表性的点**。这些点在相机视角发生少量变化后会保持不变。然后，在这些点的基础上，讨论相机位姿估计问题，以及这些点的定位问题。在经典SLAM模型中，把它们称为**路标**。而在视觉SLAM中，路标则是图像特征（Features）。**特征是图像信息的另一种数字表达形式**。
学者研究设计了许多稳定的局部图像特征，如SIFT，SURF，ORB等，这些特征点拥有如下的性质：

- 1.可重复性：相同的“区域”可以在不同的图像中被找到
- 2.可区别性：不同的“区域”有不同的表达
- c. 高效率：同一图像中，特征点的数量应远小于像素的数量
- d. 本地性：特征仅与一小片图像区域有关

特征点由**关键点**和**描述子**两部分组成。关键点是指该特征点在图像李的位置；描述子通常是一个向量，按照某种人为设计的方式，描述了该关键点周围像素的信息。描述子是按照“**外观相似的特征应该有相似的描述子**”的原则设计的。故，只要两个特征点的描述子在向量空间上的距离相近，就可以认为它们是同样的特征点。

7.1.2ORB特征

提取ORB特征分为两个步骤：

- 1.FAST角点提取：找出图像中的“角点”。ORB中计算了特征点的主要方向，为后续的BRIEF描述子增加了旋转不变特性。
- 2.BRIEF描述子：对前一步提取出特征点的周围图像区域进行描述。

FAST关键点

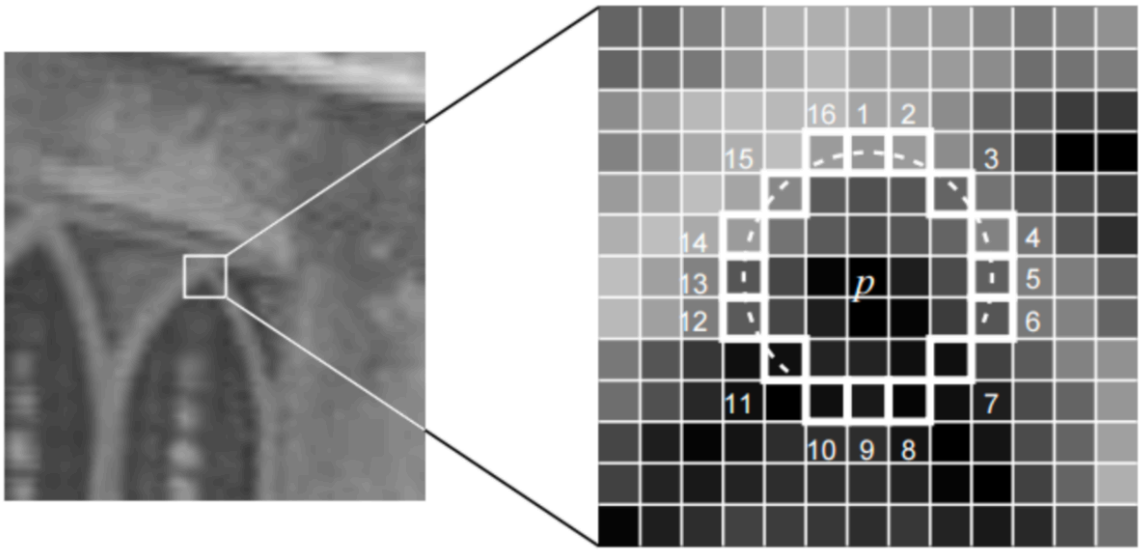


图 7-3　FAST 特征点 [33]。

FAST是一种角点，主要检测局部像素灰度变化明显的地方，以速度快著称。它的思想是：如果一个像素与它邻域的像素差别较大（过亮或过暗），那它更可能是角点。FAST只需要比较像素亮度的大小，检测过程如下：

- 1.在图像中选取像素p，假设它的亮度为I_p。
- 2.设置一个阈值T（比如I_p的20%）。
- 3.以像素p为中心，选取半径为3的圆上的16个像素点
- 4.假如选取的圆上，有连续的N个点的亮度大于I_p+T或小于I_p-T，那么像素p可以被认为是特征点（N的常用值为12，或9、11）
- 5.循环以上四步，对每一个像素执行相同的操作。

为了提高效率，可以增添一项与测试操作，以快速地排除绝大多数不是角点的像素。具体操作为：对于每个像素，直接检测邻域圆上的第1，5，9，13个像素的亮度。只有当这四个像素中有三个同事大于I_p+T或小于I_p-T时，当前像素才有可能是一个角点，否则直接排除。此外，原始的FAST角点经常出现“扎推”的现象。故第一遍检测之后，还需要用非极大值抑制（Non-maximal suppression），在一定区域内保留响应极大值的角点，避免角点集中的问题。

- 缺点：
    - 1.FAST特征点数量大且不确定。**因此在ORB中，对原始的FAST算法进行了改进。我们可以指定最终要提取的角点数量N，对原始FAST角点分别计算Harris响应值，选取前N个具有最大响应值的角点，作为最终的角点集合。**
    - 2.FAST角点不具有方向信息。且它固定取半径为3的圆，存在尺度问题：远处看着像是角点的地方，接近后看可能就不是角点了。**ORB添加了尺度和旋转的描述。尺度不变性由构建图像金字塔（*指对图像进行不同层次的降采样，以获得不同分辨率的图像*。），并在金字塔的每一层上检测角点来实现。而特征的旋转由灰度质心法实现。**
- 灰度质心法：
  质心是指以图像块灰度值作为权重的中心。具体操作步骤如下：
    - 1.在一个小的图像块B中，定义图像块的矩为：

$$m_{pq} = \sum_{x,y \in B} x^p y^q I(x,y), \quad p,q = \{0,1\}.$$

    - 2.通过矩可以找到图像块的质心

$$C = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}).$$

- 3.连接图像块的稽核中心O与质心C，得到一个方向向量OC，于是特征点的方向可以定义为：

$$\theta = \arctan(m_{01}/m_{10}).$$

通过上述方法，FAST角点便有了尺度与旋转的描述，提高了鲁棒性。改进后的FAST又称为Oriented FAST。

BRIEF描述子

BRIEF是一种二进制描述子，其描述向量由许多个0和1组成，这里的0和1编码了关键点附近两个像素（如p，q）的大小关系：如果p比q大，则取1，反之取0。p和q的选取大体上都是按照某种概率分布，随机地挑选p和q的位置，具体可见相关论文。**ORB在FAST特征点提取阶段计算了关键点的方向，所以可以利用方向信息，计算旋转之后的"Steer BRIEF"特征，使ORB的描述子具有较好的旋转不变性。**

7.1.3特征匹配

由于图像特征的局部特性，误匹配的情况广泛存在，而且长期以来一直没有得到有效解决，目前已经成为视觉SLAM中制约性能提升的一大瓶颈。部分原因是因为场景中经常存在大量的重复纹理，使得特征描述非常相似。考虑两个时刻的图像，如果在图像I_t中提取特征点x^m_t，m=1,2,...,M,在图像I_{t+1}中提取到特征点x^n_{t+1}，n=1,2,...,N，如何寻找对应关系？

- 暴力匹配
  对每一个特征点x^m_t，与所有的x^n_{t+1}测量描述子的距离，然后排序，去最近的一个作为匹配点。描述子距离表示了两个特征之间的相似程度。
  - 对于浮点类型的描述子，使用欧氏距离进行度量即可。
  - 杜宇二进制的描述子（如BRIEF），使用汉明距离作为度量——两个二进制串之间的汉明距离，指的是它们不同位数的个数。
- 快速近似最近邻（FLANN）
  适用于特征点数量很大时。该实现已集成到了OpenCV中。

7.2实践：特征提取和匹配

编译生成可运行文件后输入如下命令（**把图片都放在build文件夹下**

```
./feature_extraction 1.png 2.png
```

7.3 2D-2D：对极几何

7.3.1对极约束

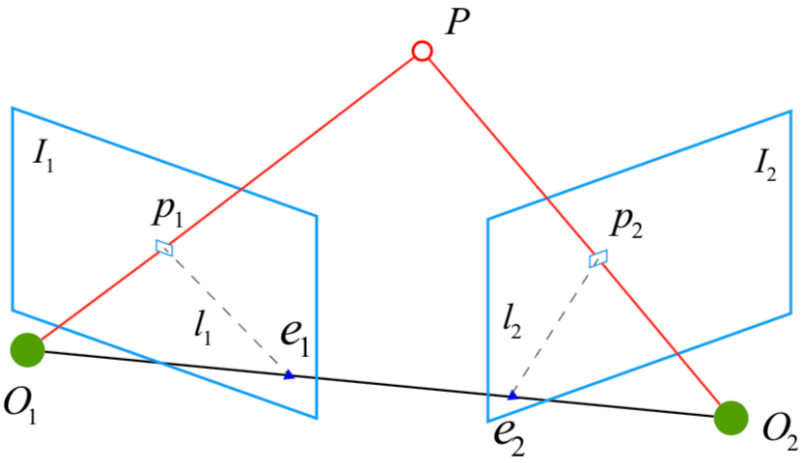


图 7-7　对极几何约束。

我们希望求取两帧图像I_1,I_2之间的运动，设第一帧到第二帧的运动为**R,t**。两个相机中心分别为O_1,O_2。考虑I_1中有一个特征点p_1，它在I_2中对应着特征点p_2。若匹配正确，则它们确实是同一个空间点在两个成像平面上的投影。点O_1，O_2，P三个点可以确定一个平面，称为极平面。O_1O_2连线与像平面I_1，I_2的交点分别为e_1，e_2。e_1，e_2称为极点，O_1O_2称为基线。称极平面与两个像平面I_1,I_2之间的相交线l_1,l_2为极线。

在第一帧坐标系下，设P的空间位置为：$\mathbf{P}=[X,Y,Z]^T$。根据针孔相机模型，像素点p_1，p_2的像素位置是：

$$s_1\boldsymbol{p}_1 = \boldsymbol{K}\boldsymbol{P}, \quad s_2\boldsymbol{p}_2 = \boldsymbol{K}(\boldsymbol{R}\boldsymbol{P}+\boldsymbol{t}). \tag{7.1}$$

。

如果使用齐次坐标，也可以把上式写成在乘以非零常数下成立的（up to a scale）等式：

$$\boldsymbol{p}_1 = \boldsymbol{K}\boldsymbol{P}, \quad \boldsymbol{p}_2 = \boldsymbol{K}(\boldsymbol{R}\boldsymbol{P}+\boldsymbol{t}). \tag{7.2}$$

现在，取：

$$\boldsymbol{x}_1 = \boldsymbol{K}^{-1}\boldsymbol{p}_1, \quad \boldsymbol{x}_2 = \boldsymbol{K}^{-1}\boldsymbol{p}_2. \tag{7.3}$$

这里的x_1，x_2是两个像素点的归一化平面上的坐标。代入上式，得：

$$\boldsymbol{x}_2 = \boldsymbol{R}\boldsymbol{x}_1 + \boldsymbol{t}. \tag{7.4}$$

两边同时左乘$\mathbf{t}^\wedge$，相当于两侧同时与**t**做外积:

$$\boldsymbol{t}^\wedge \boldsymbol{x}_2 = \boldsymbol{t}^\wedge \boldsymbol{R}\boldsymbol{x}_1. \tag{7.5}$$

然后，两侧同时左乘 $\boldsymbol{x}_2^T$：

$$\boldsymbol{x}_2^T \boldsymbol{t}^\wedge \boldsymbol{x}_2 = \boldsymbol{x}_2^T \boldsymbol{t}^\wedge \boldsymbol{R}\boldsymbol{x}_1. \tag{7.6}$$

观察等式左侧，$\boldsymbol{t}^\wedge \boldsymbol{x}_2$ 是一个与 $\boldsymbol{t}$ 和 $\boldsymbol{x}_2$ 都垂直的向量。把它再和 $\boldsymbol{x}_2$ 做内积时，将得到 0。因此，我们就得到了一个简洁的式子：

$$\boldsymbol{x}_2^T \boldsymbol{t}^\wedge \boldsymbol{R}\boldsymbol{x}_1 = 0. \tag{7.7}$$

重新代入 $\boldsymbol{p}_1, \boldsymbol{p}_2$，有：

$$\boldsymbol{p}_2^T \boldsymbol{K}^{-T} \boldsymbol{t}^\wedge \boldsymbol{R}\boldsymbol{K}^{-1} \boldsymbol{p}_1 = 0. \tag{7.8}$$

这两个式子都称为**对极约束**，它以形式简洁著名。它的几何意义是 O_1, P, O_2 三者共面。对极约束中同时包含了平移和旋转。我们把中间部分记作两个矩阵：基础矩阵（Fundamental Matrix）$\boldsymbol{F}$ 和本质矩阵（Essential Matrix）$\boldsymbol{E}$，可以进一步简化对极约束：

$$\boldsymbol{E} = \boldsymbol{t}^\wedge \boldsymbol{R}, \quad \boldsymbol{F} = \boldsymbol{K}^{-T} \boldsymbol{E} \boldsymbol{K}^{-1}, \quad \boldsymbol{x}_2^T \boldsymbol{E} \boldsymbol{x}_1 = \boldsymbol{p}_2^T \boldsymbol{F} \boldsymbol{p}_1 = 0. \tag{7.9}$$

对极约束简洁地给出了两个匹配点的空间位置关系。相机位姿估计问题变为以下两步：

- 1.根据配对点的像素位置，求出**E**或者**F**
- 2.根据**E**或**F**，求出**R，t**

7.3.2本质矩阵E（基础矩阵F）

- 八点法——使用八对匹配好的特征点来估计**E**

同理，对于其它点对也有相同的表示。我们把所有点都放到一个方程中，变成线性方程组 (u^i, v^i 表示第 i 个特征点，以此类推)：

$$\begin{pmatrix} u_1^1 u_2^1 & u_1^1 v_2^1 & u_1^1 & v_1^1 u_2^1 & v_1^1 v_2^1 & v_1^1 & u_2^1 & v_2^1 & 1 \\ u_1^2 u_2^2 & u_1^2 v_2^2 & u_1^2 & v_1^2 u_2^2 & v_1^2 v_2^2 & v_1^2 & u_2^2 & v_2^2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ u_1^8 u_2^8 & u_1^8 v_2^8 & u_1^8 & v_1^8 u_2^8 & v_1^8 v_2^8 & v_1^8 & u_2^8 & v_2^8 & 1 \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \\ e_9 \end{pmatrix} = 0. \tag{7.12}$$

这八个方程构成了一个线性方程组。它的系数矩阵由特征点位置构成，大小为 8×9。$\boldsymbol{e}$ 位于该矩阵的零空间中。如果系数矩阵是满秩的（即秩为 8），那么它的零空间维数为 1，也就是 $\boldsymbol{e}$ 构成一条线。这与 $\boldsymbol{e}$ 的尺度等价性是一致的。如果八对匹配点组成的矩阵满足秩为 8 的条件，那么 $\boldsymbol{E}$ 的各元素就可由上述方程解得。

- 奇异值分解（SVD）——根据估计到的本质矩阵**E**恢复相机的运动**R，t**

是由奇异值分解（SVD）得到的。设 $\boldsymbol{E}$ 的 SVD 分解为：

$$\boldsymbol{E} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T, \tag{7.13}$$

其中 $\boldsymbol{U}, \boldsymbol{V}$ 为正交阵，$\boldsymbol{\Sigma}$ 为奇异值矩阵。根据 $\boldsymbol{E}$ 的内在性质，我们知道 $\boldsymbol{\Sigma} = diag(\sigma, \sigma, 0)$。在 SVD 分解中，对于任意一个 $\boldsymbol{E}$，存在两个可能的 $\boldsymbol{t}, \boldsymbol{R}$ 与它对应：

$$\begin{aligned} \boldsymbol{t}_1^\wedge &= \boldsymbol{U}\boldsymbol{R}_Z(\tfrac{\pi}{2})\boldsymbol{\Sigma}\boldsymbol{U}^T, \quad \boldsymbol{R}_1 = \boldsymbol{U}\boldsymbol{R}_Z^T(\tfrac{\pi}{2})\boldsymbol{V}^T \\ \boldsymbol{t}_2^\wedge &= \boldsymbol{U}\boldsymbol{R}_Z(-\tfrac{\pi}{2})\boldsymbol{\Sigma}\boldsymbol{U}^T, \quad \boldsymbol{R}_2 = \boldsymbol{U}\boldsymbol{R}_Z^T(-\tfrac{\pi}{2})\boldsymbol{V}^T. \end{aligned} \tag{7.14}$$

其中 $\boldsymbol{R}_Z(\frac{\pi}{2})$ 表示沿 Z 轴旋转 90 度得到的旋转矩阵。同时，由于 $-\boldsymbol{E}$ 和 $\boldsymbol{E}$ 等价，所以对任意一个 $\boldsymbol{t}$ 取负号，也会得到同样的结果。因此，从 $\boldsymbol{E}$ 分解到 $\boldsymbol{t}, \boldsymbol{R}$ 时，一共存在四

一共有四种情况，但只有第一种是正确的，此时P在两个相机中都具有正的深度。故把任意一点代入四种解中，检测该点在两个相机下的深度，就可以确定哪个解是正确的了。

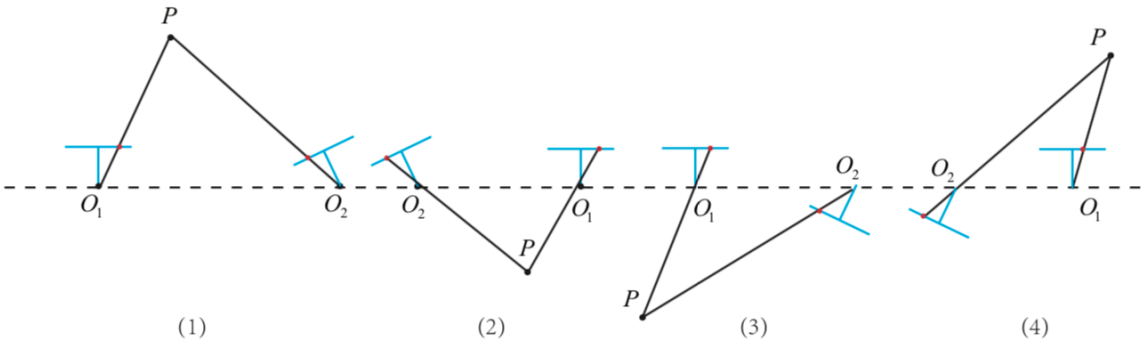


图 7-8　分解本质矩阵得到的四个解。在保持投影点（红点）不变的情况下，两个相机以及空间点一共有四种可能的情况。

剩下的问题还有一个：根据线性方程解出的 $\boldsymbol{E}$，可能不满足 $\boldsymbol{E}$ 的内在性质——它的奇异值不一定为 $\sigma,\sigma,0$ 的形式。这时，在做 SVD 时，我们会刻意地把 $\boldsymbol{\Sigma}$ 矩阵调整成上面的样子。通常的做法是，对八点法求得的 $\boldsymbol{E}$ 进行 SVD 分解后，会得到奇异值矩阵 $\boldsymbol{\Sigma}=\mathrm{diag}(\sigma_1,\sigma_2,\sigma_3)$，不妨设 $\sigma_1\geq\sigma_2\geq\sigma_3$。取：

$$\boldsymbol{E}=\boldsymbol{U}\mathrm{diag}(\frac{\sigma_1+\sigma_2}{2},\frac{\sigma_1+\sigma_2}{2},0)\boldsymbol{V}^T. \tag{7.15}$$

这相当于是把求出来的矩阵投影到了 $\boldsymbol{E}$ 所在的流形上。当然，更简单的做法是将奇异值矩阵取成 $\mathrm{diag}(1,1,0)$，因为 $\boldsymbol{E}$ 具有尺度等价性，这样做也是合理的。

7.3.3单应矩阵H

描述两个平面之间的映射关系。若场景中的特征点都落在同一平面上，则可以通过单应性来进行运动估计。

- 单应性
  单应矩阵描述处于共同平面上的一些点，在两张图像之间的变换关系。考虑在图像I_1和I_2有一对匹配好的特征点p_1和p_2。这些特征点落在某平面上，设这个平面满足方程：

$$\boldsymbol{n}^T\boldsymbol{P} + d = 0. \tag{7.16}$$

稍加整理，得：

$$-\frac{\boldsymbol{n}^T\boldsymbol{P}}{d} = 1. \tag{7.17}$$

然后，回顾本开头的式 (7.1)，得：

$$\begin{aligned}
\boldsymbol{p}_2 &= \boldsymbol{K}(\boldsymbol{R}\boldsymbol{P} + \boldsymbol{t}) \\
&= \boldsymbol{K}\left(\boldsymbol{R}\boldsymbol{P} + \boldsymbol{t}\cdot(-\frac{\boldsymbol{n}^T\boldsymbol{P}}{d})\right) \\
&= \boldsymbol{K}\left(\boldsymbol{R} - \frac{\boldsymbol{t}\boldsymbol{n}^T}{d}\right)\boldsymbol{P} \\
&= \boldsymbol{K}\left(\boldsymbol{R} - \frac{\boldsymbol{t}\boldsymbol{n}^T}{d}\right)\boldsymbol{K}^{-1}\boldsymbol{p}_1.
\end{aligned}$$

于是，我们得到了一个直接描述图像坐标 $\boldsymbol{p}_1$ 和 $\boldsymbol{p}_2$ 之间的变换，把中间这部分记为 $\boldsymbol{H}$，于是

$$\boldsymbol{p}_2 = \boldsymbol{H}\boldsymbol{p}_1. \tag{7.18}$$

单应矩阵H的求解：

这样一组匹配点对就可以构造出两项约束（事实上有三个约束，但是因为线性相关，只取前两个），于是自由度为 8 的单应矩阵可以通过 4 对匹配特征点算出（注意：这些特征点不能有三点共线的情况），即求解以下的线性方程组（当 $h_9 = 0$ 时，右侧为零）：

$$\begin{pmatrix}
u_1^1 & v_1^1 & 1 & 0 & 0 & 0 & -u_1^1u_2^1 & -v_1^1u_2^1 \\
0 & 0 & 0 & u_1^1 & v_1^1 & 1 & -u_1^1v_2^1 & -v_1^1v_2^1 \\
u_1^2 & v_1^2 & 1 & 0 & 0 & 0 & -u_1^2u_2^2 & -v_1^2u_2^2 \\
0 & 0 & 0 & u_1^2 & v_1^2 & 1 & -u_1^2v_2^2 & -v_1^2v_2^2 \\
u_1^3 & v_1^3 & 1 & 0 & 0 & 0 & -u_1^3u_2^3 & -v_1^3u_2^3 \\
0 & 0 & 0 & u_1^3 & v_1^3 & 1 & -u_1^3v_2^3 & -v_1^3v_2^3 \\
u_1^4 & v_1^4 & 1 & 0 & 0 & 0 & -u_1^4u_2^4 & -v_1^4u_2^4 \\
0 & 0 & 0 & u_1^4 & v_1^4 & 1 & -u_1^4v_2^4 & -v_1^4v_2^4
\end{pmatrix}
\begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \end{pmatrix}
=
\begin{pmatrix} u_2^1 \\ v_2^1 \\ u_2^2 \\ v_2^2 \\ u_2^3 \\ v_2^3 \\ u_2^4 \\ v_2^4 \end{pmatrix}. \tag{7.20}$$

这种做法把 $\boldsymbol{H}$ 矩阵看成了向量，通过解该向量的线性方程来恢复 $\boldsymbol{H}$，又称直接线性变换法（Direct Linear Transform）。与本质矩阵相似，求出单应矩阵以后需要对其进行分解，才可以得到相应的旋转矩阵 $\boldsymbol{R}$ 和平移向量 $\boldsymbol{t}$。分解的方法包括数值法 [42, 43] 与解析法 [44]。与本质矩阵的分解类似，单应矩阵的分解同样会返回四组旋转矩阵与平移向量，并且同时可以计算出它们分别对应的场景点所在平面的法向量。如果已知成像的地图点的深度全为正值（即在相机前方），则又可以排除两组解。最后仅剩两组解，这时需要通过更多的先验信息进行判断。通常我们可以通过假设已知场景平面的法向量来解决，如场景平面与相机平面平行，那么法向量 $\boldsymbol{n}$ 的理论值为 $\boldsymbol{1}^T$。

- 应用意义：当特征点共面，或者相机发生纯旋转的时候，基础矩阵的自由度下降，即退化。现实中的数据总包含一些噪声，这时候若继续使用八点法求解基础矩阵，基础矩阵多余出来的自由度将会主要由噪声决定。为了避免退化现象造成的影响，通常我们会同时估计基础矩阵**F**和单应矩阵**H**，选择重投影误差比较小的那个作为最终的运动估计矩阵。

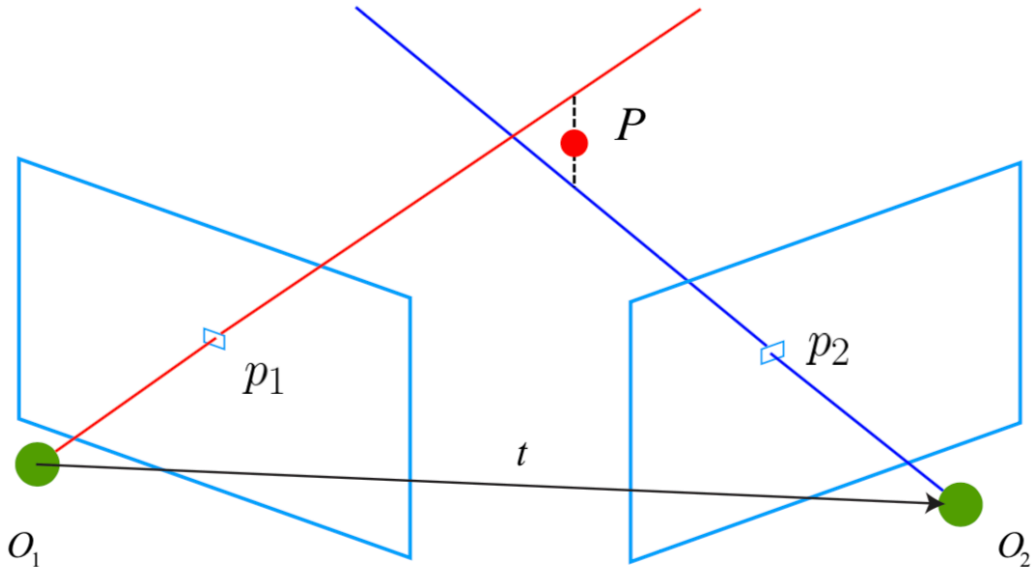
7.4实践：对极约束求解相机运动

编译生成可运行文件后输入如下命令（把图片都放在**build**文件夹下

```
./pose_estimation_2d2d 1.png 2.png
```

7.5三角测量（估计地图点的深度）

三角测量是指，通过在两处观察同一个点的夹角，确定该点的距离。



和上一节类似，考虑图像 I_1 和 I_2，以左图为参考，右图的变换矩阵为 $\boldsymbol{T}$。相机光心为 O_1 和 O_2。在 I_1 中有特征点 p_1，对应 I_2 中有特征点 p_2。理论上直线 O_1p_1 与 O_2p_2 在场景中会相交于一点 P，该点即是两个特征点所对应的地图点在三维场景中的位置。然而由于噪声的影响，这两条直线往往无法相交。因此，（又）可以通过最二小乘去求解。

按照对极几何中的定义，设 $\boldsymbol{x}_1, \boldsymbol{x}_2$ 为两个特征点的归一化坐标，那么它们满足：

$$s_1\boldsymbol{x}_1 = s_2\boldsymbol{R}\boldsymbol{x}_2 + \boldsymbol{t}. \tag{7.24}$$

现在我们已经知道了 $\boldsymbol{R}, \boldsymbol{t}$，想要求解的是两个特征点的深度 s_1, s_2。当然这两个深度是可以分开求的，比方说先来看 s_2。如果我要算 s_2，那么先对上式两侧左乘一个 $\boldsymbol{x}_1^\wedge$，得：

$$s_1\boldsymbol{x}_1^\wedge\boldsymbol{x}_1 = 0 = s_2\boldsymbol{x}_1^\wedge\boldsymbol{R}\boldsymbol{x}_2 + \boldsymbol{x}_1^\wedge\boldsymbol{t}. \tag{7.25}$$

该式左侧为零，右侧可看成 s_2 的一个方程，可以根据它直接求得 s_2。有了 s_2，s_1 也非常容易求出。于是，我们就得到了两个帧下的点的深度，确定了它们的空间坐标。当然，由于噪声的存在，我们估得的 $\boldsymbol{R}, \boldsymbol{t}$，不一定精确使式（7.24）为零，所以更常见的做法求最小二乘解而不是零解。

7.6实践：三角测量（未实践）

7.73D-2D：PnP

PnP描述了当我们知道n个3D空间点以及它们的投影位置时，如何估计相机所在的位姿。**在双目或RGB-D的视觉里程计中，我们可以直接使用PnP估计相机运动。而在单目视觉里程计中，必须先进行初始化，然后才能使用PnP。**
常用的求解方法：用三对点估计位姿的P3P，直接线性变化（DLT），EPnP，UPnP等；还有非线性优化的方式，构建最小二乘问题并迭代求解，即**Bundle Adjustment**。

7.7.1直接线性变换

7.7.2P3P

7.7.3Bundle Adjustment

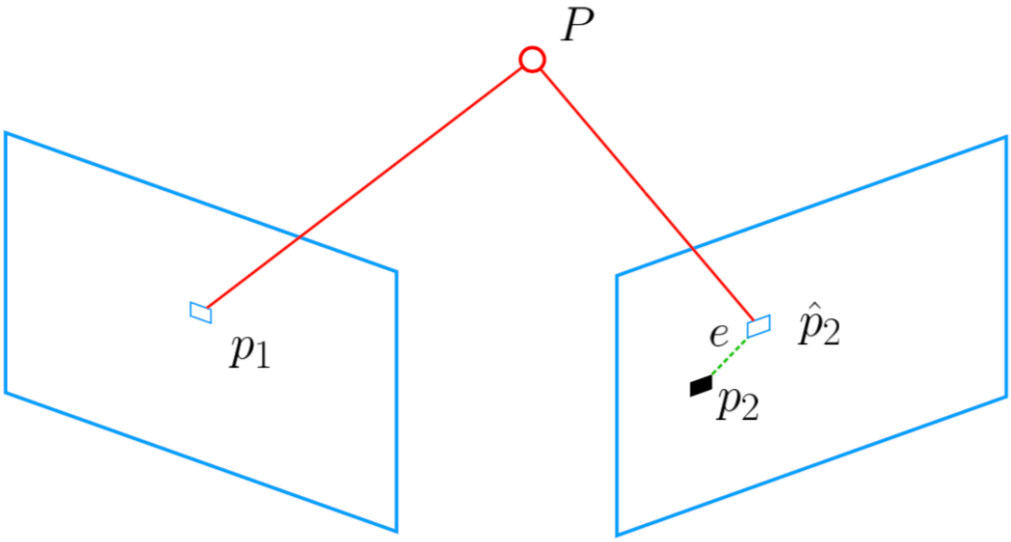


图 7-12　重投影误差示意图。

- 前面提到的线性优化，往往是**先求相机位姿，再求空间点位置**，而非线性优化则是把它们都看出优化变量，放在一起优化。在PnP中，这个Bundle Adjustment问题，是一个最小化重投影误差的问题。本节给出此问题在两个视图下的基本形式。

考虑n个三维空间点P和它们的投影p，我们希望计算相机的位姿**R**，**t**，它的李代数表示为**ξ**。假设某空间点坐标为$\mathbf{P}_i=[X_i,Y_i,Z_i]^T$，其投影的像素坐标为$\mathbf{u}_i=[u_i,v_i]^T$。根据第五章内容，像素位置与空间点的位置关系如下（其中隐含着齐次坐标到非齐次的转换）：

$$s_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \boldsymbol{K} \exp(\boldsymbol{\xi}^\wedge) \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}. \tag{7.34}$$

除了用 $\boldsymbol{\xi}$ 为李代数表示的相机姿态之外，别的都和前面的定义保持一致。写成矩阵形式就是：

$$s_i \boldsymbol{u}_i = \boldsymbol{K} \exp(\boldsymbol{\xi}^\wedge) \boldsymbol{P}_i.$$

- 位姿优化
  由于相机位姿未知及观测点的噪声，该等式存在一个误差，因此，我们把误差求和，构建最小二乘问题，然后寻找最好的相机位姿，使其最小化：

$$\boldsymbol{\xi}^* = \arg\min_{\boldsymbol{\xi}} \frac{1}{2} \sum_{i=1}^{n} \left\| \boldsymbol{u}_i - \frac{1}{s_i} \boldsymbol{K} \exp(\boldsymbol{\xi}^\wedge) \boldsymbol{P}_i \right\|_2^2. \tag{7.35}$$

  在进行优化之前，我们需要知道每个误差项关于优化变量的导数，也就是线性化：

$$\boldsymbol{e}(\boldsymbol{x} + \Delta\boldsymbol{x}) \approx \boldsymbol{e}(\boldsymbol{x}) + \boldsymbol{J}\Delta\boldsymbol{x}. \tag{7.36}$$

  我们可以推得其中**J**的形式如下：
  将这两项相乘，就得到了 2×6 的雅可比矩阵：

$$\frac{\partial \boldsymbol{e}}{\partial \delta\boldsymbol{\xi}} = - \begin{bmatrix} \frac{f_x}{Z'} & 0 & -\frac{f_x X'}{Z'^2} & -\frac{f_x X' Y'}{Z'^2} & f_x + \frac{f_x X'^2}{Z'^2} & -\frac{f_x Y'}{Z'} \\ 0 & \frac{f_y}{Z'} & -\frac{f_y Y'}{Z'^2} & -f_y - \frac{f_y Y'^2}{Z'^2} & \frac{f_y X' Y'}{Z'^2} & \frac{f_y X'}{Z'} \end{bmatrix}. \tag{7.45}$$

  这个雅可比矩阵描述了重投影误差关于相机位姿李代数的一节变化关系。前面的负号是由于误差是由观测值减预测值定义的。若SE(3)的定义方式是旋转在前，平移在后，只要把这个矩阵的前三列与后三列对调即可。
- 优化特征点的空间位置
  需要讨论**e**关于空间点**P**的导数。

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{P}} = - \begin{bmatrix} \frac{f_x}{Z'} & 0 & -\frac{f_x X'}{Z'^2} \\ 0 & \frac{f_y}{Z'} & -\frac{f_y Y'}{Z'^2} \end{bmatrix} \boldsymbol{R}.$$

7.8实践：求解PnP

pose_estimation_3d2d中的g2o初始化部分需要修改代码：

```
// 初始化g2o
    typedef g2o::BlockSolver< g2o::BlockSolverTraits<6,3> > Block;  // pose 维度为 6, landmark 维度为 3

    Block::LinearSolverType* linearSolver = new
```

```
g2o::LinearSolverCSparse<Block::PoseMatrixType>(); // 线性方程求解器
    Block* solver_ptr = new Block ( linearSolver ); // 矩阵块求解器
    g2o::OptimizationAlgorithmLevenberg* solver = new g2o::OptimizationAlgorithmLevenberg ( solver_ptr);
    g2o::SparseOptimizer optimizer;
    optimizer.setAlgorithm ( solver );
```

正确代码如下所示：

```
// 初始化g2o
    typedef g2o::BlockSolver< g2o::BlockSolverTraits<6,3> > Block;  // pose 维度为 6, landmark 维度为 3

    Block::LinearSolverType* linearSolver = new g2o::LinearSolverCSparse<Block::PoseMatrixType>(); // 线性方程求解器
    Block* solver_ptr = new Block (std::unique<Block::LinearSolverType>( linearSolver)); // 矩阵块求解器
    g2o::OptimizationAlgorithmLevenberg* solver = new g2o::OptimizationAlgorithmLevenberg (std::unique<Block>(solver_ptr));
    g2o::SparseOptimizer optimizer;
    optimizer.setAlgorithm ( solver );
```

3d3d部分的代码同样进行修改即可。

7.93D-3D：ICP

假设我们有一组配对好的3D点：

$$\boldsymbol{P}=\{\boldsymbol{p}_1,\ldots,\boldsymbol{p}_n\},\quad \boldsymbol{P}'=\{\boldsymbol{p}'_1,\ldots,\boldsymbol{p}'_n\},$$

现在，要寻找一个欧式变换**R，t**，使得：

$$\forall i,\boldsymbol{p}_i=\boldsymbol{R}\boldsymbol{p}'_i+\boldsymbol{t}.$$

仅考虑两组3D点之间的变换时，和相机并没有关系。在激光SLAM中也会碰到ICP，不过由于激光数据特征不够丰富，我们无从知道两个点集之间的匹配关系，只能认为距离最近的两个点为同一个，即迭代最近点。在RGB-D SLAM中，可以用这种方式估计相机位姿。下文中，ICP指代匹配好的两组点间的运动估计问题。

7.9.1SVD方法

首先我们看以 SVD 为代表的代数方法。根据前面描述的 ICP 问题，我们先定义第 i 对点的误差项：

$$\boldsymbol{e}_i=\boldsymbol{p}_i-(\boldsymbol{R}\boldsymbol{p}'_i+\boldsymbol{t}). \tag{7.48}$$

然后，构建最小二乘问题，求使误差平方和达到极小的 $\boldsymbol{R},\boldsymbol{t}$：

$$\min_{\boldsymbol{R},\boldsymbol{t}}J=\frac{1}{2}\sum_{i=1}^{n}\|(\boldsymbol{p}_i-(\boldsymbol{R}\boldsymbol{p}_i{}'+\boldsymbol{t}))\|_2^2. \tag{7.49}$$

优化目标函数在质心的作用下可简化为：

$$\min_{\boldsymbol{R},\boldsymbol{t}}J=\frac{1}{2}\sum_{i=1}^{n}\left\|\boldsymbol{p}_i-\boldsymbol{p}-\boldsymbol{R}\left(\boldsymbol{p}_i{}'-\boldsymbol{p}'\right)\right\|^2+\left\|\boldsymbol{p}-\boldsymbol{R}\boldsymbol{p}'-\boldsymbol{t}\right\|^2.$$

故，只要获得了**R**,令第二项为0就可以得到**t**。
而**R**使用SVD求解：

接下来，我们介绍怎样通过 SVD 解出上述问题中最优的 $\boldsymbol{R}$，但是关于最优性的证明较为复杂，感兴趣的读者请参考 [50, 51]。为了解 $\boldsymbol{R}$，先定义矩阵：

$$\boldsymbol{W} = \sum_{i=1}^{n} \boldsymbol{q}_i \boldsymbol{q}_i'^T. \tag{7.56}$$

$\boldsymbol{W}$ 是一个 3×3 的矩阵，对 $\boldsymbol{W}$ 进行 SVD 分解，得：

$$\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T. \tag{7.57}$$

其中，$\boldsymbol{\Sigma}$ 为奇异值组成的对角矩阵，对角线元素从大到小排列，而 $\boldsymbol{U}$ 和 $\boldsymbol{V}$ 为正交矩阵。当 $\boldsymbol{W}$ 满秩时，$\boldsymbol{R}$ 为：

$$\boldsymbol{R} = \boldsymbol{U}\boldsymbol{V}^T. \tag{7.58}$$

解得 $\boldsymbol{R}$ 后，按式（7.53）求解 $\boldsymbol{t}$ 即可。

总结：用SVD方法求解ICP可以分为以下三个步骤：

1. 计算两组点的质心位置 $\boldsymbol{p}, \boldsymbol{p}'$，然后计算每个点的**去质心坐标**：

$$\boldsymbol{q}_i = \boldsymbol{p}_i - \boldsymbol{p}, \quad \boldsymbol{q}_i' = \boldsymbol{p}_i' - \boldsymbol{p}'.$$

2. 根据以下优化问题计算旋转矩阵：

$$\boldsymbol{R}^* = \arg\min_{\boldsymbol{R}} \frac{1}{2} \sum_{i=1}^{n} \left\| \boldsymbol{q}_i - \boldsymbol{R}\boldsymbol{q}_i' \right\|^2. \tag{7.52}$$

3. 根据第二步的 $\boldsymbol{R}$，计算 $\boldsymbol{t}$：

$$\boldsymbol{t}^* = \boldsymbol{p} - \boldsymbol{R}\boldsymbol{p}'. \tag{7.53}$$

7.9.2非线性优化方法

以李代数表示位姿时，目标函数可以写成：

$$\min_{\boldsymbol{\xi}} = \frac{1}{2} \sum_{i=1}^{n} \left\| \left(\boldsymbol{p}_i - \exp\left(\boldsymbol{\xi}^\wedge\right) \boldsymbol{p}_i' \right) \right\|_2^2. \tag{7.59}$$

单个误差项关于位姿导数已经在前面推导过了，使用李代数扰动模型即可：

$$\frac{\partial \boldsymbol{e}}{\partial \delta \boldsymbol{\xi}} = -(\exp(\boldsymbol{\xi}^\wedge)\boldsymbol{p}_i')^\odot. \tag{7.60}$$

于是，只要不断迭代，就可以找到极小值。ICP问题存在唯一解或无穷多解的情况。在唯一解的情况下，只要能找到极小值解，那么这个极小值就是全局最优值。故ICP求解可以任意选定初始值。
这里的ICP是指已经由图像特征给定了匹配的情况下，进行位姿估计的问题。在匹配已知的情况下，这个最小二乘问题实际上具有解析解，故没必要进行优化迭代。在RGB-D SLAM中，由于一个像素的深度数据可能测量不到，故我们可以混合使用PnP和ICP优化：对于深度已知的特征点，用建模它们的3D-3D误差；对于深度未知的特征点，则建模3D-2D的重投影误差。

7.10实践：求解ICP

编译生成可运行文件后输入如下命令（把图片都放在**build**文件夹下

```
./pose_estimation_2d2d 1.png 2.png 1_depth.png 2_depth.png
```