SLAM14讲—03三维空间刚体运动

3.1旋转矩阵

3.1.1点和向量，坐标系

坐标：

$$\boldsymbol{a}=[\boldsymbol{e}_1,\boldsymbol{e}_2,\boldsymbol{e}_3]\begin{bmatrix}a_1\\a_2\\a_3\end{bmatrix}=a_1\boldsymbol{e}_1+a_2\boldsymbol{e}_2+a_3\boldsymbol{e}_3. \tag{3.1}$$

内积：

$$\boldsymbol{a}\cdot\boldsymbol{b}=\boldsymbol{a}^T\boldsymbol{b}=\sum_{i=1}^{3}a_ib_i=|\boldsymbol{a}|\,|\boldsymbol{b}|\cos\langle\boldsymbol{a},\boldsymbol{b}\rangle. \tag{3.2}$$

外积：

$$\boldsymbol{a}\times\boldsymbol{b}=\begin{bmatrix}\boldsymbol{i}&\boldsymbol{j}&\boldsymbol{k}\\a_1&a_2&a_3\\b_1&b_2&b_3\end{bmatrix}=\begin{bmatrix}a_2b_3-a_3b_2\\a_3b_1-a_1b_3\\a_1b_2-a_2b_1\end{bmatrix}=\begin{bmatrix}0&-a_3&a_2\\a_3&0&-a_1\\-a_2&a_1&0\end{bmatrix}\boldsymbol{b}\triangleq\boldsymbol{a}^{\wedge}\boldsymbol{b}. \tag{3.3}$$

- 外积还可以表示向量的旋转：
  考虑两个不平行的向量**a**，**b**，我们要描述从**a**到**b**之间是如何旋转的，如下图所示。我们可以用一个向量来描述三维空间中两个向量的旋转关系。在右手法则下，我们用右手的四个指头从**a**转向**b**，其大拇指朝向就是旋转向量的方向，事实上也是**a ✕ b**的
  方向。它的大小则由**a**和**b**的夹角决定。通过这种方式，我们构造了从**a**到**b**的一个旋转向量。这个向量同样位于三维空间中，在此坐标系下，可以用三个实数来描述它

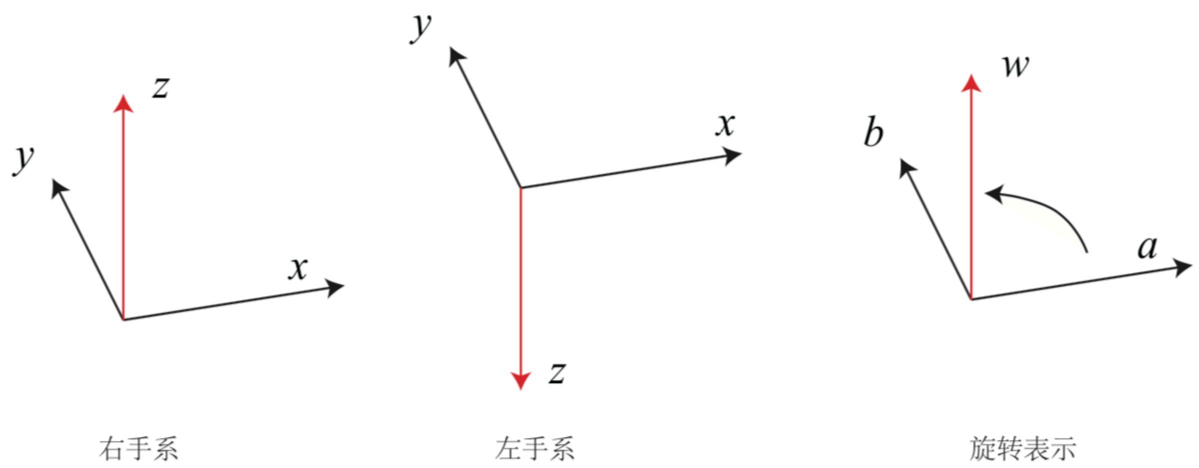

图 3-1　左右手系的区别与向量间的旋转。$\boldsymbol{a}$ 到 $\boldsymbol{b}$ 的旋转可以由向量 $\boldsymbol{w}$ 来描述。

3.1.2坐标系间的欧式变换

在机器人的运动过程中，常见的做法是设定一个惯性坐标系（或叫*世界坐标系*），*可以认为它是固定不动*的，如图3-2中的x_W，y_W，z_W定义的坐标系。同时相机或机器人则是一个移动坐标系，例如x_C，y_C，z_C定义的坐标系。

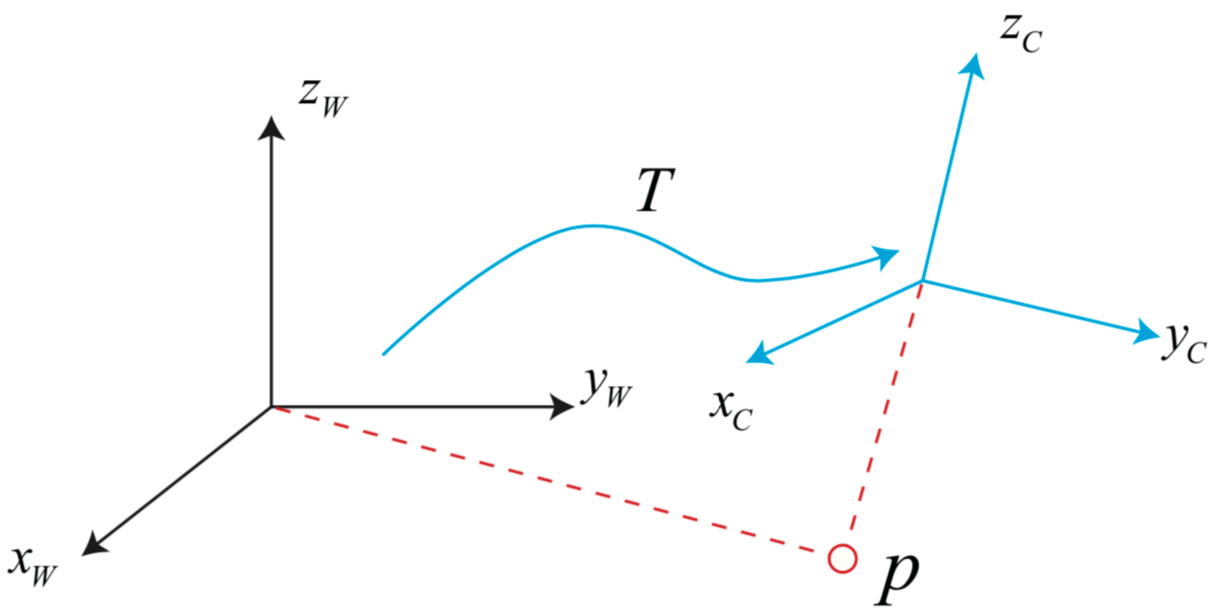

图 3-2　坐标变换。对于同一个向量 $\boldsymbol{p}$，它在世界坐标系下的坐标 $\boldsymbol{p}_w$ 和在相机坐标系下的 $\boldsymbol{p}_c$ 是不同的。这个变换关系由坐标系间的变换矩阵 $\boldsymbol{T}$ 来描述。

- 问：相机视野中某个向量$\mathbf{p}$，它的坐标为$\mathbf{p}_c$，与世界坐标系下的坐标$\mathbf{p}_w$如何进行转换？

相机运动是一个刚体运动，只有空间位置和姿态会发生变化。

1. 先考虑旋转
   设某个单位正交基（$\mathbf{e}_1$,$\mathbf{e}_2$,$\mathbf{e}_3$)经过一次旋转变成了($\mathbf{e}'_1$,$\mathbf{e}'_2$,$\mathbf{e}'_3$)。那么对于同一个向量$\mathbf{a}$，它在两个坐标系下的坐标为$[a_1,a_2,a_3]^T$和$[a'_1,a'_2,a'_3]^T$，根据坐标的定义，有：

$$[\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3] \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \left[\boldsymbol{e}_1^{'}, \boldsymbol{e}_2^{'}, \boldsymbol{e}_3^{'}\right] \begin{bmatrix} a_1^{'} \\ a_2^{'} \\ a_3^{'} \end{bmatrix}. \tag{3.4}$$

在上式两边同乘$[\mathbf{e}^T{}_1, \mathbf{e}^T{}_2, \mathbf{e}^T{}_3]^T]^T$，则左边的系数变成了单位矩阵：

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{e}_1^T \boldsymbol{e}_1^{'} & \boldsymbol{e}_1^T \boldsymbol{e}_2^{'} & \boldsymbol{e}_1^T \boldsymbol{e}_3^{'} \\ \boldsymbol{e}_2^T \boldsymbol{e}_1^{'} & \boldsymbol{e}_2^T \boldsymbol{e}_2^{'} & \boldsymbol{e}_2^T \boldsymbol{e}_3^{'} \\ \boldsymbol{e}_3^T \boldsymbol{e}_1^{'} & \boldsymbol{e}_3^T \boldsymbol{e}_2^{'} & \boldsymbol{e}_3^T \boldsymbol{e}_3^{'} \end{bmatrix} \begin{bmatrix} a_1^{'} \\ a_2^{'} \\ a_3^{'} \end{bmatrix} \triangleq \boldsymbol{R}\boldsymbol{a}^{'}. \tag{3.5}$$

矩阵**R**即为旋转矩阵,可以描述相机的旋转。
按上面的定义方式，则有：

$$\boldsymbol{a}^{'} = \boldsymbol{R}^{-1}\boldsymbol{a} = \boldsymbol{R}^T\boldsymbol{a}. \tag{3.7}$$

- 旋转矩阵的性质
  旋转矩阵即为行列式为1的正交矩阵。
  可以定义旋转矩阵的集合如下：

  $$SO(n) = \{\boldsymbol{R} \in \mathbb{R}^{n \times n} | \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}, \det(\boldsymbol{R}) = 1\}. \tag{3.6}$$

2. 再考虑平移
   考虑世界坐标系中的向量**a**。经过一次旋转（用**R**描述）和一次平移**t**后，得到了**a**$^{'}$，即：

   $$\boldsymbol{a}^{'} = \boldsymbol{R}\boldsymbol{a} + \boldsymbol{t}. \tag{3.8}$$

3.1.3变换矩阵与齐次坐标

式（3.8）所表示的变换关系不是一个线性关系：

- 假设我们进行了两次变换：$\mathbf{R}_1$,$\mathbf{t}_1$和$\mathbf{R}_2$,$\mathbf{t}_2$，满足：

  $$\boldsymbol{b} = \boldsymbol{R}_1\boldsymbol{a} + \boldsymbol{t}_1, \quad \boldsymbol{c} = \boldsymbol{R}_2\boldsymbol{b} + \boldsymbol{t}_2.$$

  但是从**a**到**c**的变换为：

  $$\boldsymbol{c} = \boldsymbol{R}_2(\boldsymbol{R}_1\boldsymbol{a} + \boldsymbol{t}_1) + \boldsymbol{t}_2.$$

多次变换后会很复杂，故引入齐次坐标和变换矩阵重写式（3.8）：

$$\begin{bmatrix} \boldsymbol{a}^{'} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ 1 \end{bmatrix} \triangleq \boldsymbol{T} \begin{bmatrix} \boldsymbol{a} \\ 1 \end{bmatrix}. \tag{3.9}$$

把一个三维向量的末尾添加1，变成四维向量，称为齐次坐标。矩阵**T**称为变换矩阵。我们用 $\tilde{\boldsymbol{a}}$ 表示a**的齐次坐标。

- 齐次坐标
  在齐次坐标中，某个点**x**的每个分量同乘一个非零常数k后，**仍然表示的是同一个点**。故，一个点的具体坐标值不是唯一的。但若最后一项不为0时，我们总可以把所有坐标除以最后一项，强制最后一项为1，从而得到一个点唯一的坐标表示（即转换成非齐次坐标）：

  $$\tilde{\boldsymbol{x}} = [x, y, z, w]^T = [x/w, y/w, z/w, 1]^T. \tag{3.10}$$

  这时，忽略掉最后一项，改点的坐标和欧式空间是一样的。

据此，我们可将两次变换的累加变为如下形式：

$$\tilde{\boldsymbol{b}} = \boldsymbol{T}_1\tilde{\boldsymbol{a}},\ \tilde{\boldsymbol{c}} = \boldsymbol{T}_2\tilde{\boldsymbol{b}} \quad \Rightarrow \tilde{\boldsymbol{c}} = \boldsymbol{T}_2\boldsymbol{T}_1\tilde{\boldsymbol{a}}. \tag{3.11}$$

在不引起歧义的情况下，之后都直接写成b=Ta，默认T为齐次坐标。

- 特殊欧式群

$$SE(3) = \left\{ \boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} | \boldsymbol{R} \in SO(3), \boldsymbol{t} \in \mathbb{R}^3 \right\}. \tag{3.12}$$

与 $SO(3)$ 一样，求解该矩阵的逆表示一个反向的变换：

$$\boldsymbol{T}^{-1} = \begin{bmatrix} \boldsymbol{R}^T & -\boldsymbol{R}^T\boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix}. \tag{3.13}$$

3.2实践：Eigen（具体代码详见/slambook/ch3/）

调用eigen直接在cpp文件开头include即可，并且在CMakeLists文件中指定头文件位置

```
include_directories("/usr/local/eigen3")
```

注意usr前面的“/”不要忘记！！
也可以使用以下代码：

```
fina_package(Eigen3)
INCLUDE_DIRECTORIES(${EIGEN3_INCLUDE_DIR})
```

3.3旋转向量和欧拉角

3.3.1旋转向量

矩阵表示方式的几个缺点：

1. SO(3)的旋转矩阵有九个量，但是一次旋转只有三个自由度。因此这种表达方式是冗余的。同理，变换矩阵用十六个量表达了六自由度的变换，可以由更紧凑的表示。
2. 旋转矩阵自身带有约束：它必须是个正交矩阵，且行列式为1。变换矩阵也是如此。故在估计或优化一个旋转矩阵/变换矩阵时，这些约束会使得求解变得更困难。

旋转向量（或轴角，**Axis-Angle）**

一个向量，其方向与旋转轴一致，而长度等于旋转角。这样只需一个三维向量即可描述旋转。对于变换矩阵，我们使用一个旋转向量和一个平移向量即可表达一次变换，此时维数正好是六维。

旋转向量与旋转矩阵之间的转换

假设有一个旋转轴$\mathbf{n}$，角度为θ的旋转，其对应的旋转向量为θ$\mathbf{n}$。

- 旋转向量到旋转矩阵之间的转换结果由罗德里格斯公式表明：

  $$\boldsymbol{R} = \cos\theta \boldsymbol{I} + (1-\cos\theta)\boldsymbol{n}\boldsymbol{n}^T + \sin\theta \boldsymbol{n}^\wedge. \tag{3.14}$$

  符号^是向量到反对称的转换符，见式（3.3）。
- 旋转矩阵到旋转向量的转换

  $$\begin{aligned}\mathrm{tr}(\boldsymbol{R}) &= \cos\theta\,\mathrm{tr}(\boldsymbol{I}) + (1-\cos\theta)\,\mathrm{tr}\left(\boldsymbol{n}\boldsymbol{n}^T\right) + \sin\theta\,\mathrm{tr}(\boldsymbol{n}^\wedge) \\ &= 3\cos\theta + (1-\cos\theta) \\ &= 1 + 2\cos\theta.\end{aligned} \tag{3.15}$$

  因此：

  $$\theta = \arccos\left(\frac{\mathrm{tr}(\boldsymbol{R})-1}{2}\right). \tag{3.16}$$

关于转轴$\mathbf{n}$，由于旋转轴上的向量在旋转后不发生改变，说明：**Rn=n**。
因此，转轴**n**是矩阵**R**特征值1对应的特征向量。求解此方程，再归一化，就得到了旋转轴。

3.3.2欧拉角

欧拉角使用了三个分离的转角，把一个旋转分解成三次绕不同轴的旋转。由于分解方式的不同，欧拉角也存在着不同的定义方法，例如XYZ、ZYZ、ZYX等旋转方式均可。还需要区分每次旋转是绕固定轴旋转的，还是绕旋转之后的轴旋转的。
欧拉角中比较常用的一种，便是用“偏航-俯仰-滚转“（**yaw-pitch-roll**）三个角度来描述一个旋转的。

- 由于它等价于ZYX轴的旋转，我们就以ZYX为例。假设一个刚体的前方（朝向我们的方向）为X轴，右侧为Y轴，上方为Z轴。那么ZYX转角相当于把任意旋转分解成以下三个轴上的转角：
  - 1.绕物体的Z轴旋转，得到偏航角yaw；
  - 2.绕旋转之后的Y轴旋转，得到俯仰角pitch;
  - 3.绕旋转之后的X轴旋转，得到滚转角roll。

    此时，可以用$[r,p,y]^T$这样一个三维的向量描述任意旋转。
- 欧拉角的一个*重大缺点*是碰到会碰到著名的万向锁问题：在俯仰角为±90°时，第一次旋转与第三次旋转将使用同一个轴，使得系统丢失了一个自由度。理论上可以证明，只要我们想用三个实数来表示三维旋转时，都会不可避免地碰到奇异性问题。由于这种原理，欧拉角不适于插值和迭代，往往只用于人机交互中。在SLAM中也很少直接使用欧拉角表示姿态。

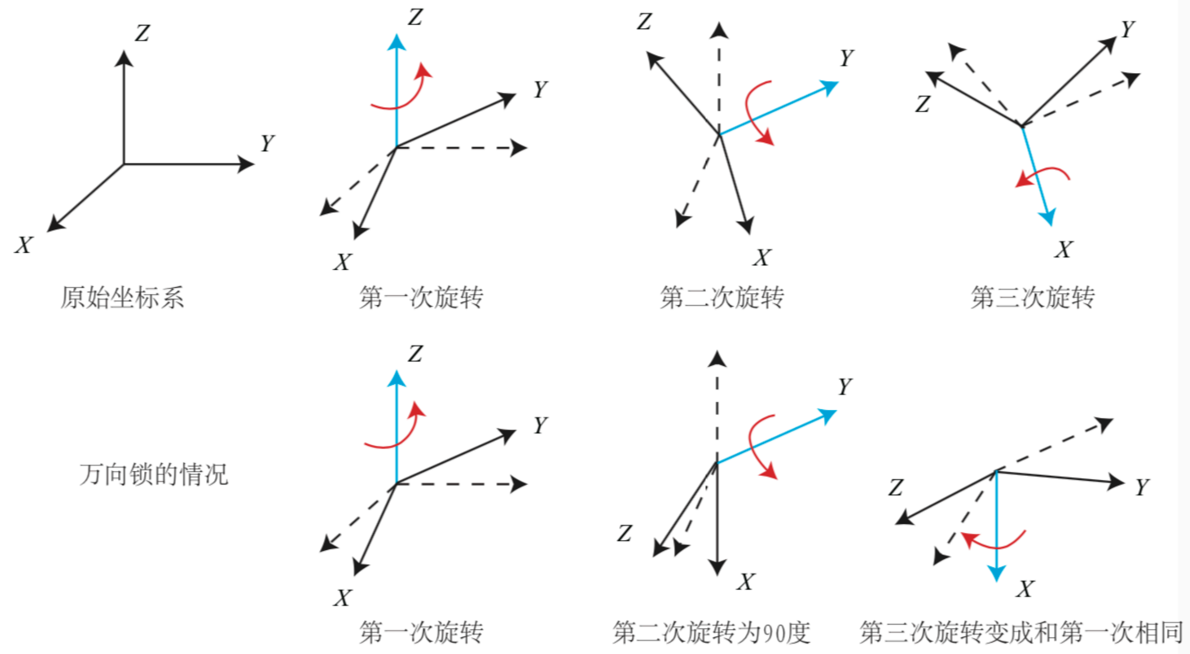

图 3-3 欧拉角的旋转示意图。上方为 ZYX 角定义。下方为 pitch=90 度时，第三次旋转与第一次滚转角相同，使得系统丢失了一个自由度。如果你还没有理解万向锁，可以看看相关视频，理解起来会更方便。

3.4四元数

3.4.1四元数的定义

四元数是Hamilton找到的一种扩展的复数，**既是紧凑的，也没有奇异性**，缺点为不够直观且运算稍复杂。
一个四元数**q**拥有一个实部和三个虚部，如下：

$$\boldsymbol{q} = q_0 + q_1 i + q_2 j + q_3 k, \tag{3.17}$$

其中i,j,k为四元数的三个虚部，满足如下关系式：

$$\begin{cases} i^2 = j^2 = k^2 = -1 \\ ij = k, ji = -k \\ jk = i, kj = -i \\ ki = j, ik = -j \end{cases}. \tag{3.18}$$

也可以用一个标量和一个向量来表示四元数：

$$\boldsymbol{q} = [s, \boldsymbol{v}], \quad s = q_0 \in \mathbb{R}, \boldsymbol{v} = [q_1, q_2, q_3]^T \in \mathbb{R}^3,$$

其中，s为四元数的实部，而**v**为它的虚部。
我们可以用单位四元数表示三维空间中任意一个旋转。

- 假设某个旋转是绕单位向量**n**=$[n_x,n_y,n_z]^T$进行了角度为θ的旋转，那么这个旋转的四元数形式为：

$$\boldsymbol{q} = \left[\cos\frac{\theta}{2}, n_x \sin\frac{\theta}{2}, n_y \sin\frac{\theta}{2}, n_z \sin\frac{\theta}{2}\right]^T. \tag{3.19}$$

我们可以从单位四元数中计算出对应旋转轴与夹角（由3.17和3.19联合得到）：

$$\begin{cases} \theta = 2\arccos q_0 \\ [n_x, n_y, n_z]^T = [q_1, q_2, q_3]^T / \sin\frac{\theta}{2} \end{cases}. \tag{3.20}$$

若（3.19）式的θ加上2π，则此时的四元数变成-q。因此，在四元数中，**任意的旋转都可以由两个互为相反数的四元数表示**。θ取0时，得到一个没有任何旋转的实四元数:$\mathbf{q}_0$=$[\pm1,0,0,0]^T$.

3.4.2四元数的运算

3.4.3用四元数表示旋转

假设一个空间三维点**p**=[x,y,z]，以及一个由轴角**n**，θ指定的旋转。三维点**p**经过旋转之后变成**p**'。如果使用矩阵描述，则有**p**'=**Rp**。

- 四元数描述如下：
  p = [0,x,y,z] = [0,**v**]
  q = [cos(θ/2),**n**sin(θ/2)]
  则旋转后的点**p**'可以表示为这样的乘积：

$$\boldsymbol{p}' = \boldsymbol{q}\boldsymbol{p}\boldsymbol{q}^{-1}. \tag{3.34}$$

3.4.4四元数到旋转矩阵的转换

设四元数 $\boldsymbol{q} = q_0 + q_1 i + q_2 j + q_3 k$，对应的旋转矩阵 $\boldsymbol{R}$ 为：

$$\boldsymbol{R} = \begin{bmatrix} 1 - 2q_2^2 - 2q_3^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & 1 - 2q_1^2 - 2q_3^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 1 - 2q_1^2 - 2q_2^2. \end{bmatrix} \tag{3.35}$$

反之，由旋转矩阵到四元数的转换如下。假设矩阵为 $\boldsymbol{R} = \{m_{ij}\}, i, j \in [1, 2, 3]$，其对应的四元数 $\boldsymbol{q}$ 由下式给出：

$$q_0 = \frac{\sqrt{\mathrm{tr}(R) + 1}}{2}, q_1 = \frac{m_{23} - m_{32}}{4q_0}, q_2 = \frac{m_{31} - m_{13}}{4q_0}, q_3 = \frac{m_{12} - m_{21}}{4q_0}. \tag{3.36}$$

值得一提的是，由于 $\boldsymbol{q}$ 和 $-\boldsymbol{q}$ 表示同一个旋转，事实上一个 $\boldsymbol{R}$ 对应的四元数表示并不是惟一的。同时，除了上面给出的转换方式之外，还存在其他几种计算方法，而本书都省略了。实际编程中，当 q_0 接近 0 时，其余三个分量会非常大，导致解不稳定，此时我们再考虑使用其他的方式进行转换。

3.6实践:Eigen几何模块（具体见slambook/ch3/useGeometry/useGeometry.cpp）

表 3-1　常见变换性质比较

变换名称	矩阵形式	自由度	不变性质
欧氏变换	$\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix}$	6 自由度	长度、夹角、体积
相似变换	$\begin{bmatrix} s\boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix}$	7 自由度	体积比
仿射变换	$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix}$	12 自由度	平行性、体积比
射影变换	$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{a}^T & v \end{bmatrix}$	15 自由度	接触平面的相交和相切

Eigen 中对各种形式的表达方式总结如下。请注意每种类型都有单精度和双精度两种数据类型，而且和之前一样，不能由编译器自动转换。下面以双精度为例，你可以把最后的 d 改成 f，即得到单精度的数据结构。

- 旋转矩阵（3×3）：Eigen::Matrix3d。
- 旋转向量（3×1）：Eigen::AngleAxisd。
- 欧拉角（3×1）：Eigen::Vector3d。
- 四元数（4×1）：Eigen::Quaterniond。
- 欧氏变换矩阵（4×4）：Eigen::Isometry3d。
- 仿射变换（4×4）：Eigen::Affine3d。
- 射影变换（4×4）：Eigen::Projective3d。

```
#include<Eigen/Core>
#include<Eigen/Geometry>
```

3.7可视化演示（具体见slambook/ch3/visualizeGeometry）

世界坐标系和相机坐标系

设某个点在世界坐标系中坐标为$\mathbf{p}_w$，在相机坐标系下为$\mathbf{p}_c$，那么：

$$\boldsymbol{p}_c = \boldsymbol{T}_{cw}\boldsymbol{p}_w, \tag{3.40}$$

这里$\mathbf{T}_{cw}$表示世界坐标系到相机坐标系间的变换。或者我们可以反过来用$\mathbf{T}_{wc}$</sub表示：

$$\boldsymbol{p}_w = \boldsymbol{T}_{wc}\boldsymbol{p}_c = \boldsymbol{T}_{cw}^{-1}\boldsymbol{p}_c. \tag{3.41}$$

原则上，两者都可以表示相机的位姿，但一般用$\mathbf{T}_{\mathrm{cw}}$。如把$\mathbf{p}_{\mathrm{c}}$取成零向量，即相机坐标系中的原点，则此时$\mathbf{p}_{\mathrm{w}}$就是相机原点在世界坐标系下的坐标：

$$\boldsymbol{p}_w = \boldsymbol{T}_{wc}\boldsymbol{0} = \boldsymbol{t}_{wc}. \tag{3.42}$$

正是$\mathbf{T}_{\mathrm{wc}}$的平移部分。因此，可以从$\mathbf{T}_{\mathrm{wc}}$中直接看出相机在何处。