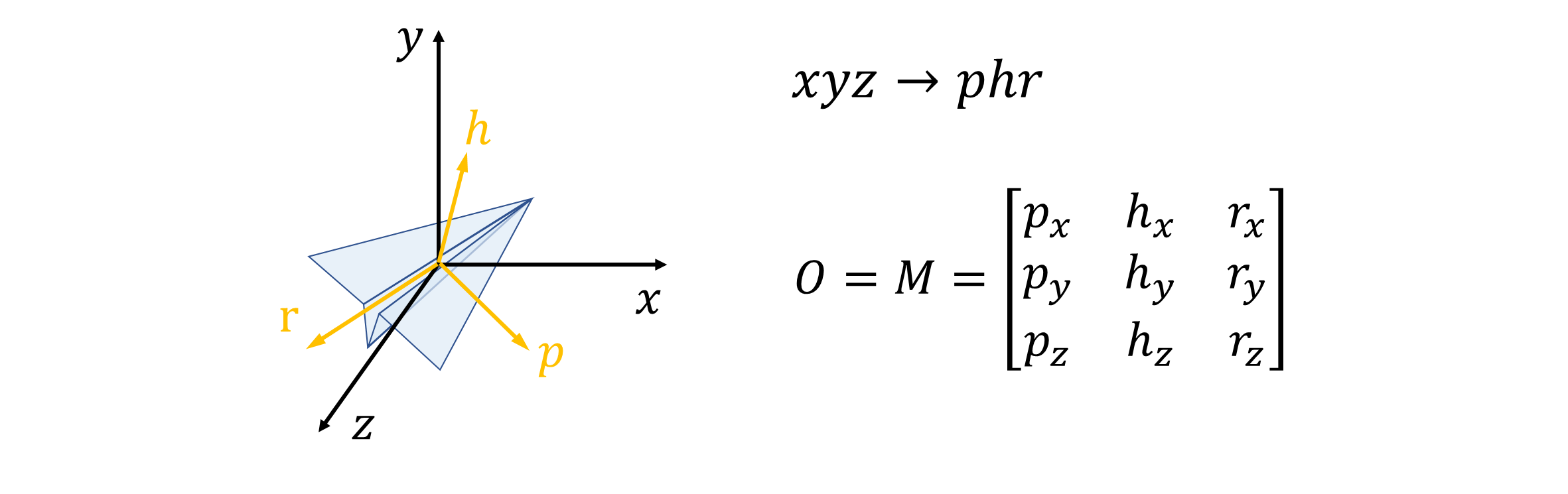
笔者最近在回顾一些图形学基础知识，遂整理在此，此文主要讲述图形学中的变换。

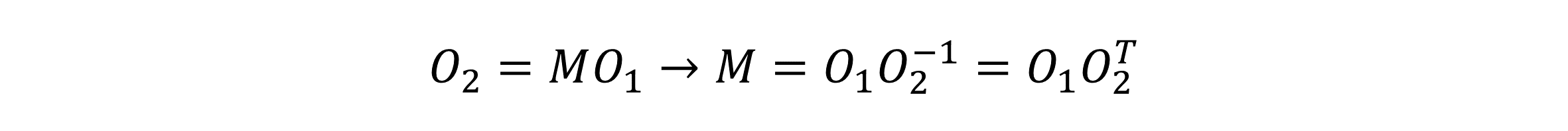
这是关于变换的第二篇文章。前面介绍了基础的变换矩阵，主要是对点向量和方向向量进行变换，此处引入定向(Orientation)的概念，进而介绍三维旋转的四种形式：矩阵形式、欧拉角、指数映射和四元数。并进行一些扩展：如关于三维旋转四种形式的对比、万向节死锁。本文同样使用列向量，且遵循右手定则。

前面我们讲到可以把点向量写作(x,y,z,1)表示一个三维点，把方向向量写作(x,y,z,0)表示一个三维空间的方向。而定向则表示：一个物体的三维姿态，如一架飞机在飞行过程中的：俯仰角、航向角、翻滚角，这三个角就是这架飞机的定向的欧拉角形式。除此之外，还有矩阵形式、指数映射形式、四元数形式。

1、矩阵形式，比如可以把物体坐标系的三个轴phr写进一个矩阵的列来表示纸飞机的定向，记作字母O，同时也是从xyz到phr的变换矩阵M（此处xyz与phr的对应关系只是一种惯例，当然也可以用其他的对应关系）

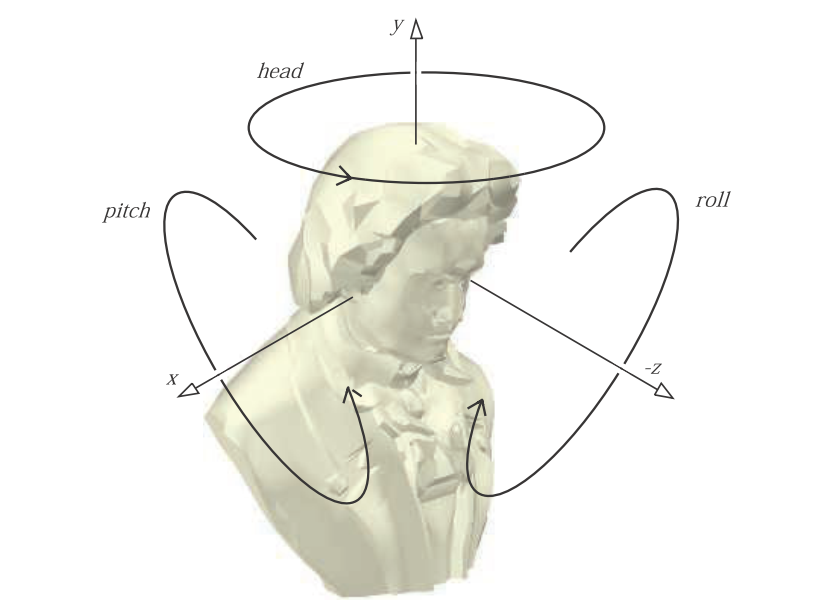


同样的，一个矩阵也可以表示从一个定向到另一个定向的变换：如把从定向O1到O2的变换记为M，则：



但是，矩阵形式对于定向的表示和对于定向的变换并不直观，无法直观想象出这个物体的空间姿态，下面的欧拉角形式解决了这个

2、欧拉角形式。前面说到定向可以理解为物体的三维姿态，有三个自由度，那么最少用三个变量表示即可。一种欧拉角的约定是：航向角（Head）、俯仰角（Pitch）和翻滚角（Roll），与旋转轴的对应关系如下：Head-y，Pitch-x，Roll-z，旋转正方向的定义遵循右手定则。以一个人体模型为例，则把前方当作-z轴，右方当作x轴，上方当作y轴，如下图所示：



在进行旋转的时候，另外两个轴也会跟着旋转（注意与固定轴方式不一样，网上有人以动态和静态区分）。欧拉角系统虽然最易于我们使用，但也会出现一些问题：

首先是别名问题，想象你转头90度和转头90+360n度会得到一样的结果，所以常常限制欧拉角的范围，将Head和Roll限制为 (-180,180]，将Pitch限制为[-90,90]。

其次，不同的旋转顺序和角度可能得到相同的结果，如：俯仰角向下旋转135与先航向角旋转180，再俯仰角向上旋转45，再翻滚角旋转180可得到一样的结果。所以常常要规定旋转的次序，如：x/y/z的次序，也有其他比如z/x/y，甚至是z/x/z.

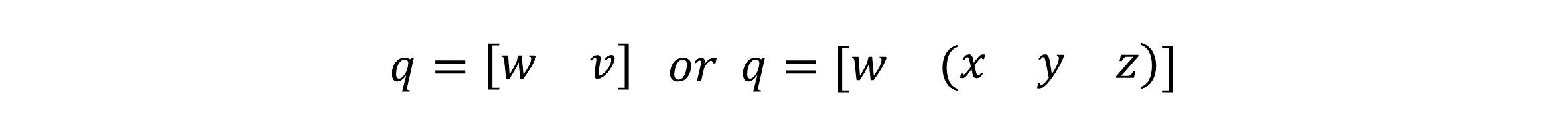
最后，欧拉角的插值会出现很多问题，比如把Roll从-170插值到170，是选择插值340度的范围还是插值20度的范围？以及困扰着我们的万向节死锁（Gimbal Lock）问题，我在本文扩展部分会深入探讨。

欧拉角虽然很直观，在描述一个物体定向的时候也很容易，但是要在比如动画的关键帧之间插值时几乎无能为力，这就需要指数映射或者四元数的辅助。

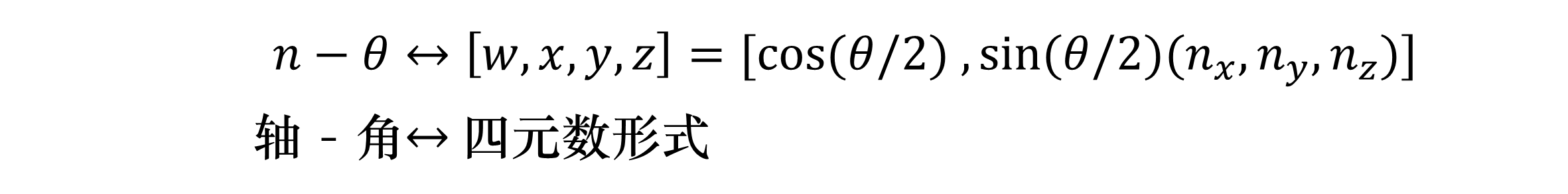
3、指数映射。首先讲述一个定理：任何三维角位移（可理解为从一个定向到另一个定向）都可以通过围绕一根特定轴且只经过一次旋转来完成。这种方式称之为轴-角方式，需要一个方向向量n和一个角a，一共四个值，此表示的一个改进就是指数映射，把角a的大小乘到单位矢量n上面去，得到：e = an. 在需要的时候求长度即可，所以指数映射更为紧凑。

指数映射也存在别名的现象，比如角度a和a+360之后的结果定向相同，但这也不总是一个缺点，比如可以帮助描述角速度，如每秒720的角速率和每秒1080的角速率还是有区别的，之后会说到四元数不能描述角速度。

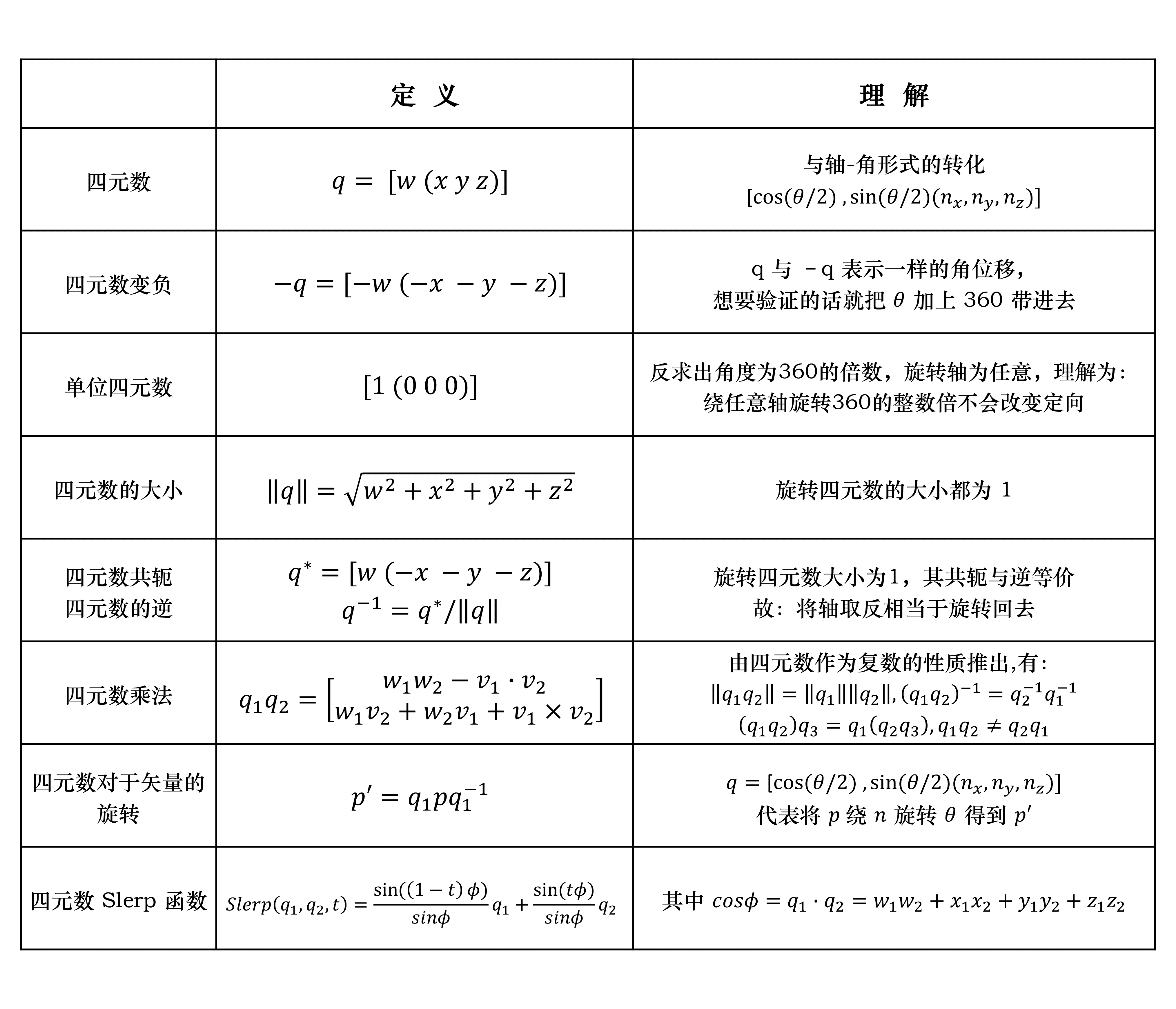
4、四元数。一些人会把四元数介绍为复数，但我在这里直接给出四元数的定义。四元数包括一个标量和一个三维矢量，一共四个数。可表示为：



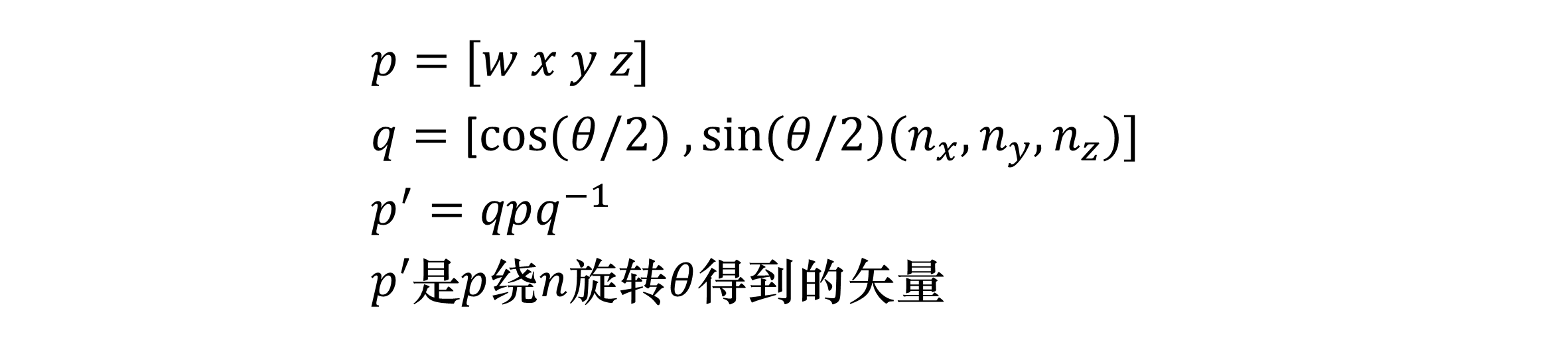
四元数与轴-角或者指数映射形式类似，都包含了轴和角，但需要进行以下转化：



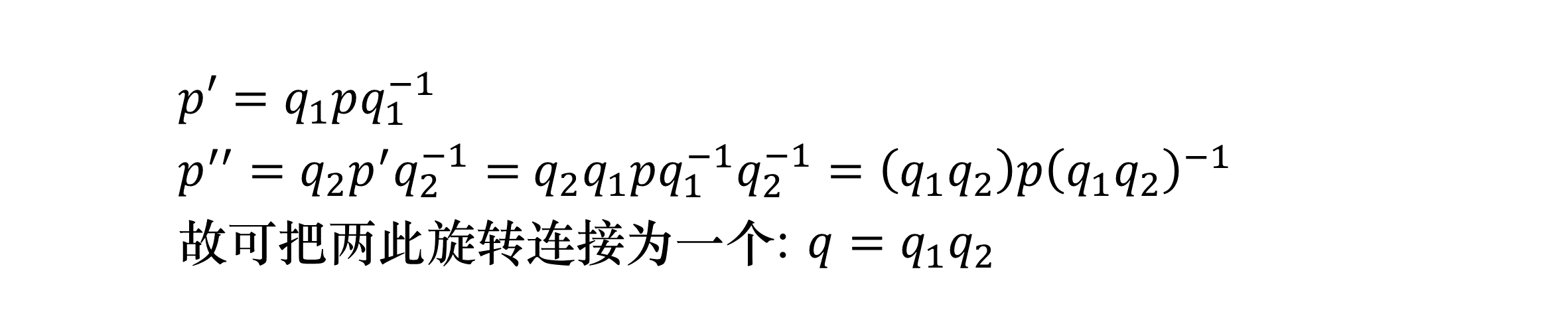
四元数的一些运算：



四元数对矢量的旋转：从一个矢量构建四元数 p = (w x y z)，其中xyz代表矢量的三维分量，w取0或1代表为点或者方向，可以证明的是，执行以下操作可以让 p 旋转得到 p’：

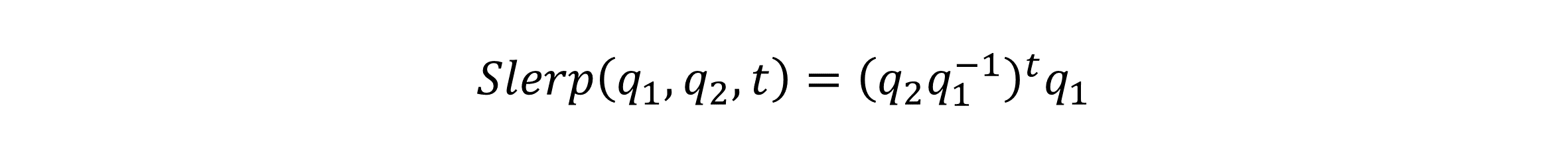


四元数和矩阵一样，可连接多个旋转：

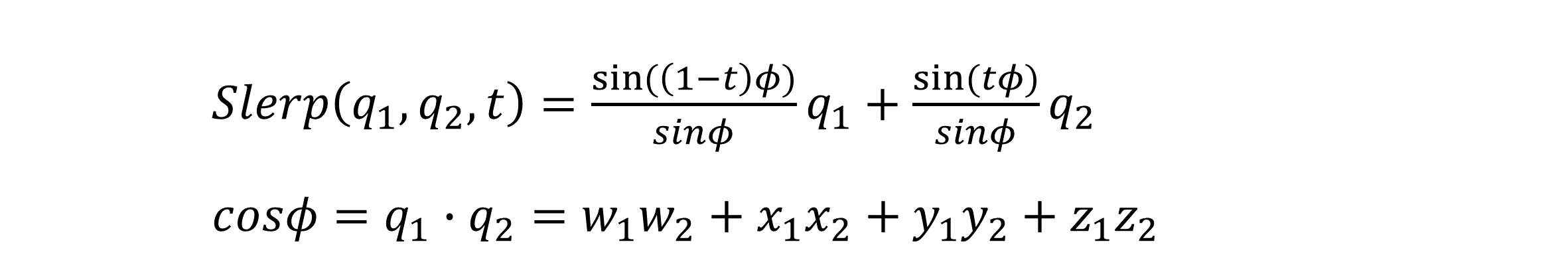


四元数的球面线性插值：

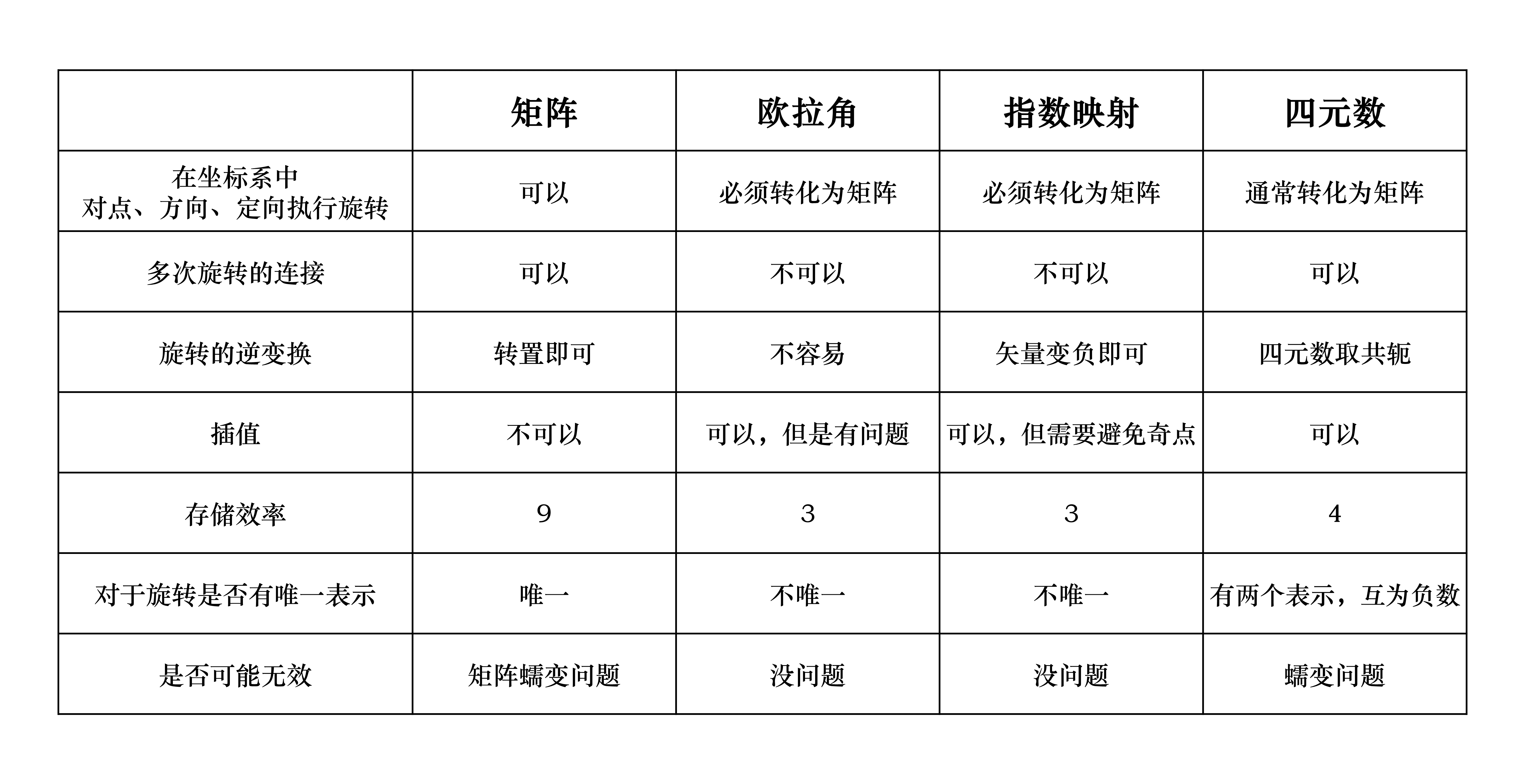
在骨骼动画里面，动画师们通常提供的是动画的关键帧，这些关键帧里面存储的就是各个骨骼的定向，如何将这些关键帧转化为连续的骨骼动画，就需要进行插值，即求出两个关键帧之间的任何一个定向。设两个定向为q1和q2，t的范围是0到1，这里直接给出四元数的理论球面线性插值公式：



和实际使用的球面线性插值公式，他们意义相同，但后者计算量更小：



扩展一：矩阵、欧拉角、指数映射、四元数四种形式的对比



此表对四种形式进行了对比，此外还有以下建议供参考：

1. 欧拉角最易于让人使用
2. 如果要在空间坐标系中执行变换，最终必须转化为矩阵形式
3. 需要进行大量存储时，考虑用欧拉角、指数映射和四元数
4. 只有四元数才能提供稳定可靠的插值
5. 如果需要表示旋转的角速度，可使用指数映射

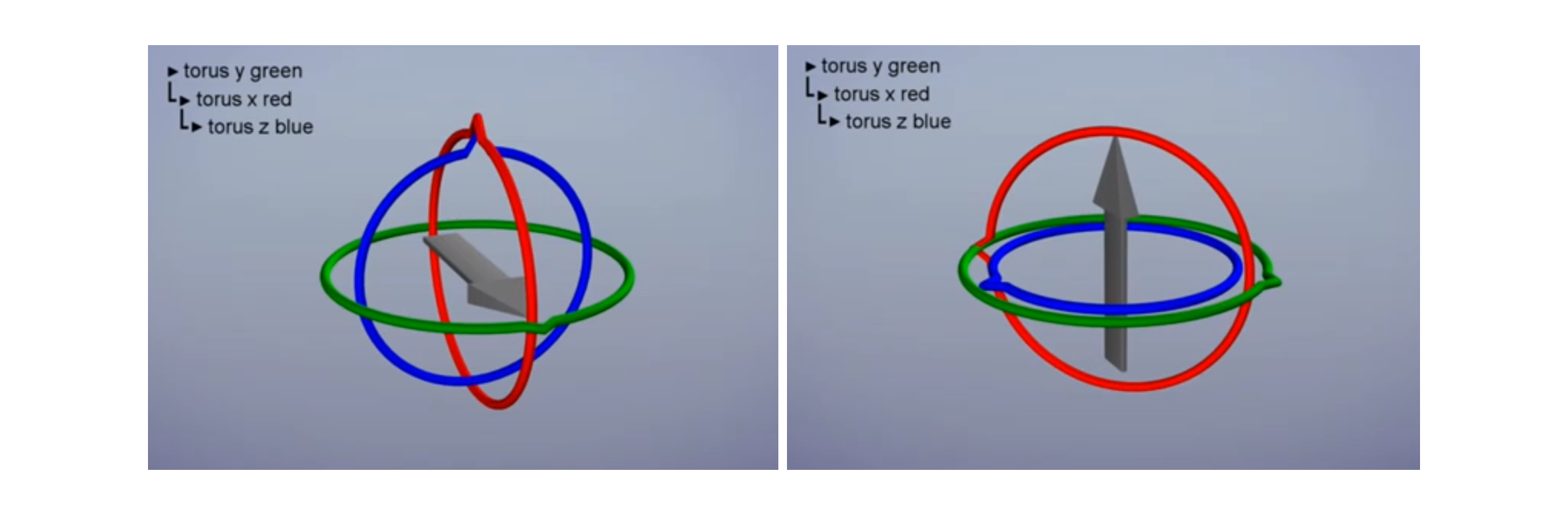
扩展二：万向节死锁

在网上看到Niklas Frykholm写的一篇博客，我认为比较客观地讲解万向节死锁（Gimbal Lock）在图形学中是怎样一回事：

<https://jahej.com/alt/2013_03_15_what-is-gimbal-lock-and-why-do-we-still-have-to-worry-about-it.html>

在此，我简单传达一些他的观点或者表达一些我自己的想法。

首先，要将机械工程学中的Gimbal Lock和图形学中的区分开，我相信你们也经常在网上看到这种万向架模型：



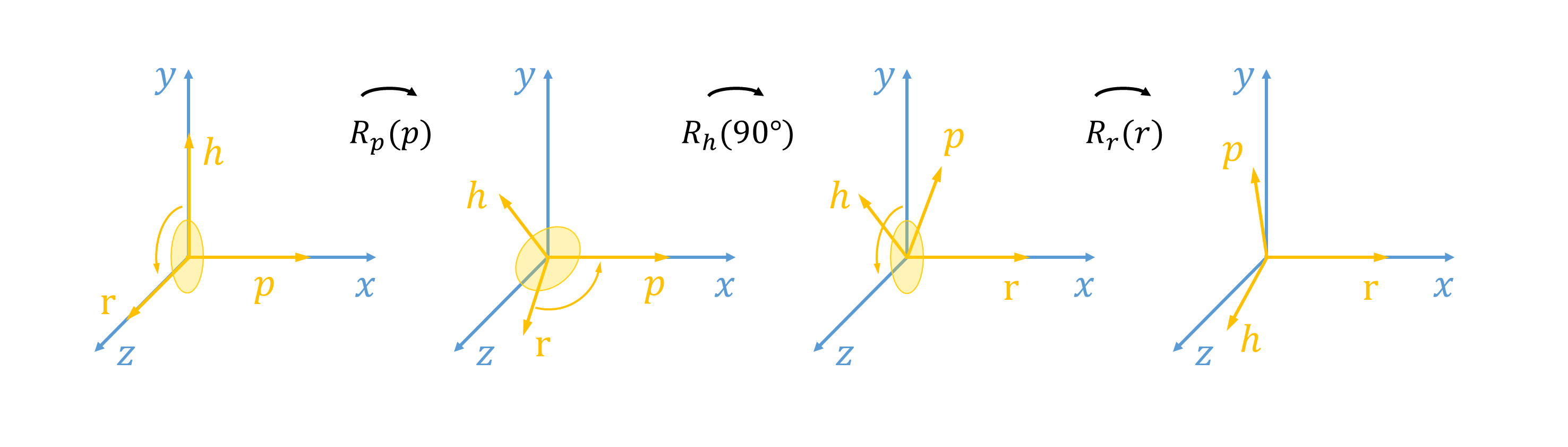
但我认为这种模型只存在于机械学中而不存在于图形学中，仔细观察这个模型：

在第一次旋转绿色的y，这时候里面的红色x和蓝色z是跟着转的，确实符合绕体轴旋转（也叫物体空间旋转、动态旋转）的规律，但是第二次旋转红色的x，此时绿色的y已经没转了，只有蓝色的z在和x一起转，然后第三次旋转蓝色的z，此时也没有带动绿色的y和红色的x一起旋转。

所以后面两次旋转不符合绕体轴旋转的规律，所以才会出现上右图这种两根轴重合的情况，而在图形学中，不管是绕固定轴旋转还是绕体轴旋转（即有人称之动态和静态两种旋转），都是不可能让坐标轴重合的，所以始终不会“锁住”。

所以我很赞同Niklas Frykholm在他的博客里面将Gimbal Lock（万向节死锁）称之为Euler angle flip（欧拉角翻转） 或者coordinate singularity（坐标奇异性），我将图形学里面的“万向节死锁”带来的影响分为两点：

1、在绕体轴旋转时，如果第二次的旋转角度为90度，就会导致第三次的旋转轴与第一次的重合，产生旋转冗余。比如按照x/y/z的顺序，如y轴旋转角度为正90度，那么会发现第三次绕z轴旋转的时候，这时的z轴和第一次绕x轴旋转那时x轴的方向一致，所以才会有“绕z轴旋转是冗余的”这一说法



如图所示，当第二次绕体轴h轴（即y）旋转90度后，第三幅图中的r轴和第一幅图中的p轴处于同一位置，所以说第三次绕r（z）轴旋转与第一次绕p（z）轴旋转冗余了。要想规避这种冗余，我们可以根据不同的场景，选择不同的旋转次序，比如x/y/z、z/x/y和z/x/z都是比较常用的，但从理论上我们是无法完全规避掉这种现象。这是第一个问题，我认为这种冗余并不会带来多严重的后果，无非是在进行欧拉角旋转的时候，仔细考虑旋转次序和旋转角度，但是这种麻烦程度和本身绕体轴的旋转矩阵的推导是差不多的。

2、第二个就是插值问题，我觉得更像是欧拉角显而易见的别名带来的，而不是万向节死锁。由于欧拉角的别名问题，在限制范围之前，如将0度和360度插值，是选择从0~360旋转一圈，还是不插值？如果限制了范围，即制定了“规范欧拉角”的规则，在插值时还是会有问题，比如从-170度插值到170，是选择340度的范围还是选择20度的范围？

的确，对于这些问题，也出现了诸如wrapPi这类工具函数（在两个角度之间选择最短弧度）来尽量避免插值错误，但是由于插值期间的角速度不恒定，对欧拉角进行插值总会出现一些问题，所以插值还是选择四元数

此文首先引入了定向的概念，介绍了描述定向和角位移的四种方式：矩阵、欧拉角、指数映射、四元数，并对进行了对比，最后简单分享了关于万向节死锁的想法，另外关于欧拉角、矩阵和四元数之间的转换方法，放到下一篇文章作为补充。

注意：此处所列举的各种表达式并不一定就能直接使用，一定要先考察所处的坐标系统、行列向量、左右手法则。三维旋转变幻多端，但是相信只要理解了其中的思想，就能一通百通。笔者对此理解可能不够深刻，有不妥之处，请大家多多指教！