从零开始一起学习SLAM | 用四元数插值来对齐IMU和图像帧

视觉 Vs. IMU

小白：师兄，好久没见到你了啊，我最近在看IMU（Inertial Measurement Unit，惯性导航单元）相关的东西，正好有问题求助啊

师兄：又遇到啥问题啦？

小白：是这样的，现在VIO（Visual-Inertial Odometry，视觉惯性里程计）很火，我就想试试把IMU测量的信息和图像进行简单的融合，这样利用IMU测量的先验信息，可以给图像一个比较好的初值。。。

师兄：嗯嗯，这个思路没问题的啊，图像信息和 IMU 确实存在一定互补性，两者各有所长，取长补短。

小白：是滴，我也是这样想的，不过我采集了图像和IMU的数据后，发现IMU输出频率好高啊，远远大于图像帧率！

师兄：没错，IMU本身就是惯性传感器，用来测量角速度和加速度，对短时快速运动很敏感，因此帧率很高才能测量到，所以一般是100Hz以上。而我们图像传感器输出帧率一般比较低，15 - 60Hz 居多~

小白：那就有问题了啊，我想要把IMU测量的值和图像估计的值进行对齐，这样我就能根据当前IMU输出的旋转量来作为图像预测的初值了，现在帧率差这么多，这个怎么对齐呢？

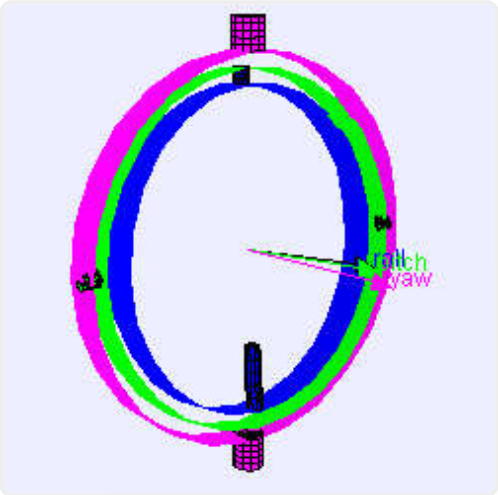
师兄：你是用什么来表达的旋转？

小白：四元数，我看网上都说用四元数好，不过不知道为啥（/尴尬）

四元数的优势

师兄：四元数确实在对姿态的描述具有独特的优势，非常适合用来表示空间中的旋转。这主要是因为几个原因：

1、四元数解决了其他3维空间旋转算法会遇到的恼人的问题，比如使用欧拉角来表示旋转操作时会遇到的万向节锁问题(Gimbal lock)。见下图



2、计算效率比旋转矩阵方法高，因为表达四元数只需要4个数，旋转矩阵需要9个。

3、其简单的数学表达方式可以被用来规划出高阶连续姿态运动以及在多姿态间插值。这里的插值就可以解决你说的对齐问题啦

小白：原来如此，看来我选择四元数表示是非常正确的！不过我有个疑问，师兄，什么是插值啊？

什么是插值？

师兄：插值对应的英文是**interpolation**，是数学上的一个常用术语。下面是维基百科的专业解释

数学的数值分析领域中，插值是一种通过已知的、离散的数据点，在一定范围内推求新数据点的过程或方法。求解科学和工程的问题时，通常有许多数据点借由采样、实验等方法获得，这些数据可能代表了有限个数值函数，其中自变量的值。而根据这些数据，我们往往希望得到一个连续的函数（也就是曲线）；或者更密集的离散方程与已知数据互相吻合，这个过程叫做拟合。插值是曲线必须通过已知点的拟合。

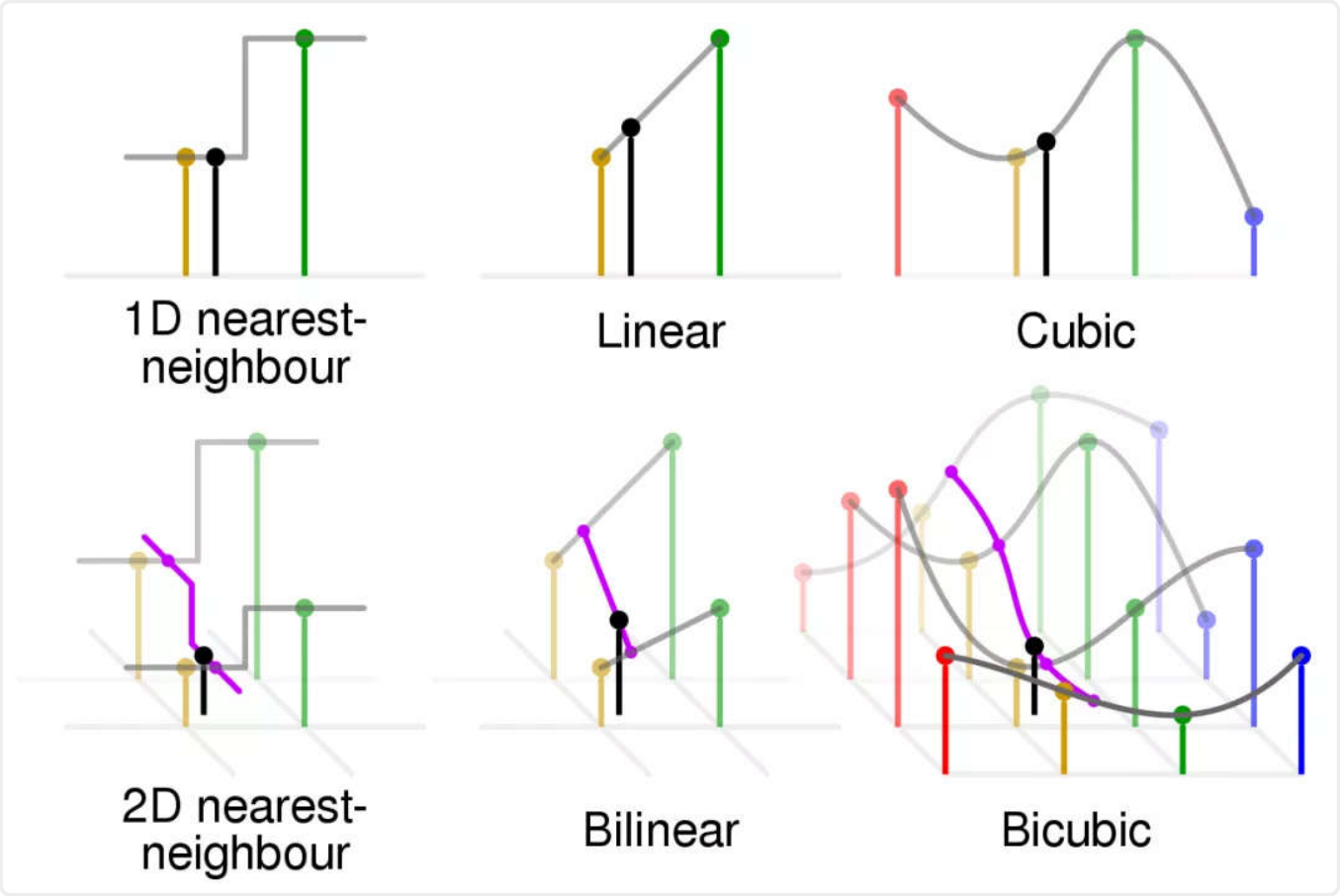
小白：师兄，你说的每个字我都认识，但是连在一起完全不知道啥意思啊！

师兄：没关系，为了严谨定义一般都比较晦涩难懂。插值，讲一个通俗但不严谨的例子，比如现在有10只大雁（对应已有的样本）排成一定的阵列在飞，让你在第5、6只大雁中间（原来没有样本的插值点）再插入一只大雁，但是要保证插队后的大雁在整体中不能太突兀，要显得比较“合群”（对应拟合曲线），如果其他大雁飞人字形，插入的大雁尽量要保持整体仍是人字形；如果其他大雁飞一字形，插入的大雁尽量要保持整体仍是一字形。

小白：师兄，你早这么说，我不就明白啦！那一般怎样插值呢？

师兄：嗯，以后多举例子。插值方法有很多种，比如最简单的最邻近插值(nearest interpolation)、线性插值(linear interpolation)；常用的双线性插值（Bilinear interpolation）,还有保护图像细节效果较好的双三次插值(bicubic interpolation)、三次样条插值(cubic Spline Interpolation)等。

千言万语汇成一个图，如下图是一维和二维插值的比较。黑色表示待计算的插值点，其他颜色的点表示样本点。

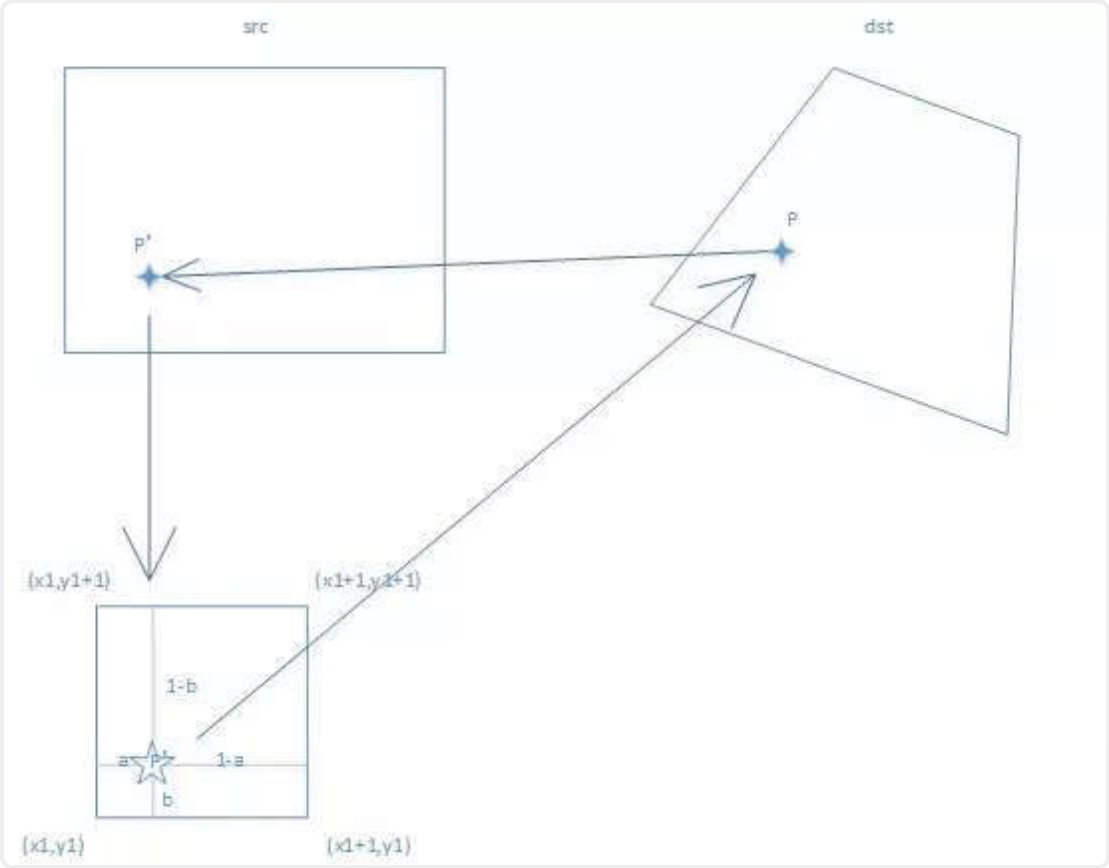


小白：看晕了都。那这么多插值方法，我们用哪种呢？

师兄：在图像处理和计算机视觉领域，应用比较多的双线性插值。双线性插值的效果不是最好的，但相较最邻近插值和线性插值的简单粗暴，其获得图像的效果还是更令人满意的，而且双线性插值的计算量和易于理解程度会优于双三次插值和三次样条插值等高阶插值方法。因此双线性插值还是最受广大图像研究者喜爱的。

小白：师兄，可以举个具体的例子吗？还是不太明白插值的具体应用呢！

师兄：嗯，那就再举个栗子吧，比如我们常见的针孔相机成像就是一种射影变换，下图中一个矩形src经过相机拍摄后成像变为了dst，此时我们拿到了dst图像中的像素点，如果想要用得到的dst图像来恢复原始的src图像，就需要用到射影变换和插值。



当我们要对图像进行插值操作的时候，通常需要遍历dst中的每一个像素点，假设dst中某像素点为p(x0,y0)，对像素点p进行相应变换，使其对应到原图src中的p’（x0’,y0’）点。在我们遍历dst像素的时候， p点的像素值(x0,y0)都是整数，然而变换后对应到原图src中的p’ 点的像素值(x0’,y0’)就不一定是整数了。

小白：图中的p'点就是插值点吧?

师兄：对！将src中p’ 附近的内容放大，我们可以发现p’ （x0’,y0’）点落在了(x1,y1), (x1+1,y1), (x1+1,y1+1), (x1,y1+1)四个相邻点中间。我们要做的，就是要利用(x1,y1), (x1+1,y1), (x1+1,y1+1), (x1,y1+1)这几个整数点的像素值来计算p’ （x0’,y0’）这个非整数点的像素值，再用src中p’ （x0’,y0’）的像素值表示dst中p(x0,y0)的像素值。这个就是插值啦！关于这部分内容网上很多资料，也不是今天的重点，这里就不详细介绍了，今天重点是介绍四元数插值~

小白：嗯嗯，我回头去查查看双线性插值。有点跑偏了，我们还是回到四元数插值的讨论吧~

四元数有哪些插值方法?

师兄：好，其实四元数插值的思路也和上面类似，常见的有线性插值、球面线性插值等。我们从简单的说起。

先说说最简单的线性插值（**L**inear Int**erp**olation，简称Lerp）

假设有两个四元数 q0,q1，想要在位置 t 处求插值 qt，用线性插值可以这样计算，是不是很熟悉?

$$q_t = Lerp(q_0, q_1, t) = (1-t)q_0 + tq_1$$

小白：是啊，感觉这个非常简单啊，那我就用这个插值好了!

师兄：四元数的线性插值是非常简单，但是是有代价的。如下图所示，四元数表示旋转时是单位四元数，这种插值方式，相当于我们是沿着一条直线（也就是圆上的一个弦）进行插值的，这样插值出来的四元数不是单位四元数，而且还有其他问题（后面会说）。

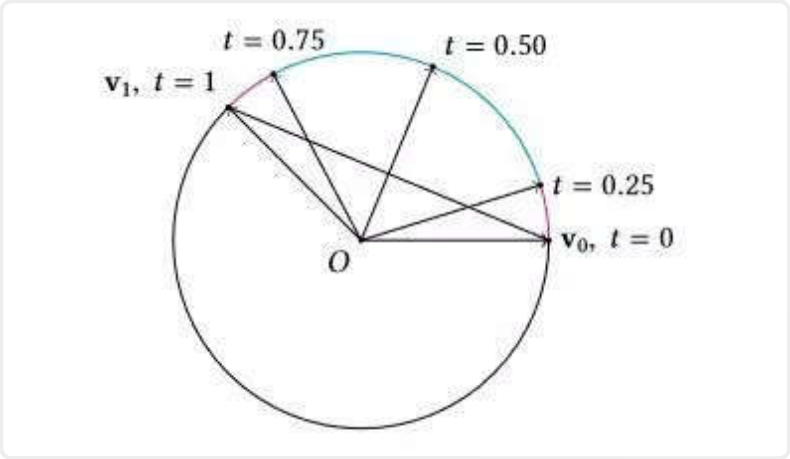
小白：那我归一化一下就行了吧?

师兄：你说的就是归一化线性插值（Normalized LinearInterpolation，简称Nlerp)，前面说过Lerp这样插值出来的并不是单位四元数，但如你所说，只要将 qt 除以它的模 ||qt||就能够 将其转化为一个单位四元数了:

$$q_t = Nlerp(q_0, q_1, t) = \frac{(1-t)q_0 + tq_1}{\|(1-t)q_0 + tq_1\|}$$

小白：嗯，那就这样进行四元数插值吧，看起来也不是很复杂哈!

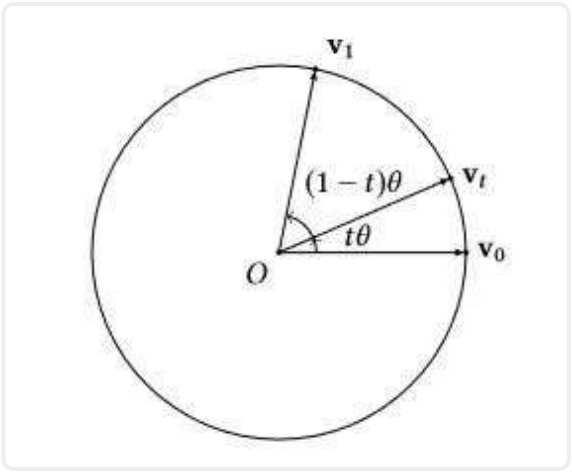
师兄：且慢！还是有其他问题的。如下图所示，在同等时间内， vt 扫过的角度是不同的, vt 扫过的速度（或者说角速度）首先会不断地增加，到t = 0.50之后会开始减速，所以Nlerp插值不能保证均匀的角速度。



小白：那怎么办呢？

师兄：为了解决这个问题，我们可以转而对角度进行线性插值。这就要使用更复杂一些的插值方法了，比如常用的球面线性插值(Spherical Linear Interpolation)，简称Slerp。Slerp插值可以解决前面的均匀角速度问题，它能够保证 每两个四元数之间的角速度是固定的，这就从原理上保证了插值的效果。如下图所示，如果 v1 和 v2 之间的夹角为 θ，那么：

$$\theta_t = (1-t)\cdot 0 + t\theta = t\theta$$



小白：那这个四元数怎么计算呢？

师兄：计算也不复杂，主要是利用三角形、三角函数性质。证明过程我们就不推导了，直接给出以下结论。当然如果你对结果有疑问，也可以自己推导一遍~

$$q_t = Slerp(q_0, q_1, t) = \frac{sin((1-t)\theta)}{sin(\theta)} q_0 + \frac{sin(t\theta)}{sin(\theta)} q_1$$

$$\theta = acos(q_0 \cdot q_1)$$

小白：不，不，我相信这个结论，推导的事情前人已经做过了，我就不重复造轮子了，哈哈，用过的时候直接套公式就行了吧！

师兄：理论上是这样的，不过，在编程实现Slerp插值的时候还是有几个问题需要注意一下。

1、如果单位四元数之间的夹角θ非常小，那么sin(θ)可能会由于浮点数的误差被近似为0.0，从而导致除以0的错误．所以，我们在实施 Slerp 之前，需要检查两个四元数的夹角是否过小（或者完全相同）。一旦发现这种问题，我们就必须改用 Nlerp 对两个四元数进行插值，这时候 Nlerp 的误差非常小，所以基本不会与真正的 Slerp 有什么区别。

2、在对两个单位四元数进行插值之前，我们需要先检测q0与q1之间是否是钝角，即检测它们点积的结果q0·q1 是否为负数。如果 q0·q1<0，那么我们就反转其中的一个四元数，比如说将q1改为−q1 ，并使用q0与−q1之间新的夹角来进行插值，这样才能保证插值的路径是最短的．

小白：哇塞，太中肯的建议了！可以少踩好多坑，谢谢师兄，我要去编程啦！

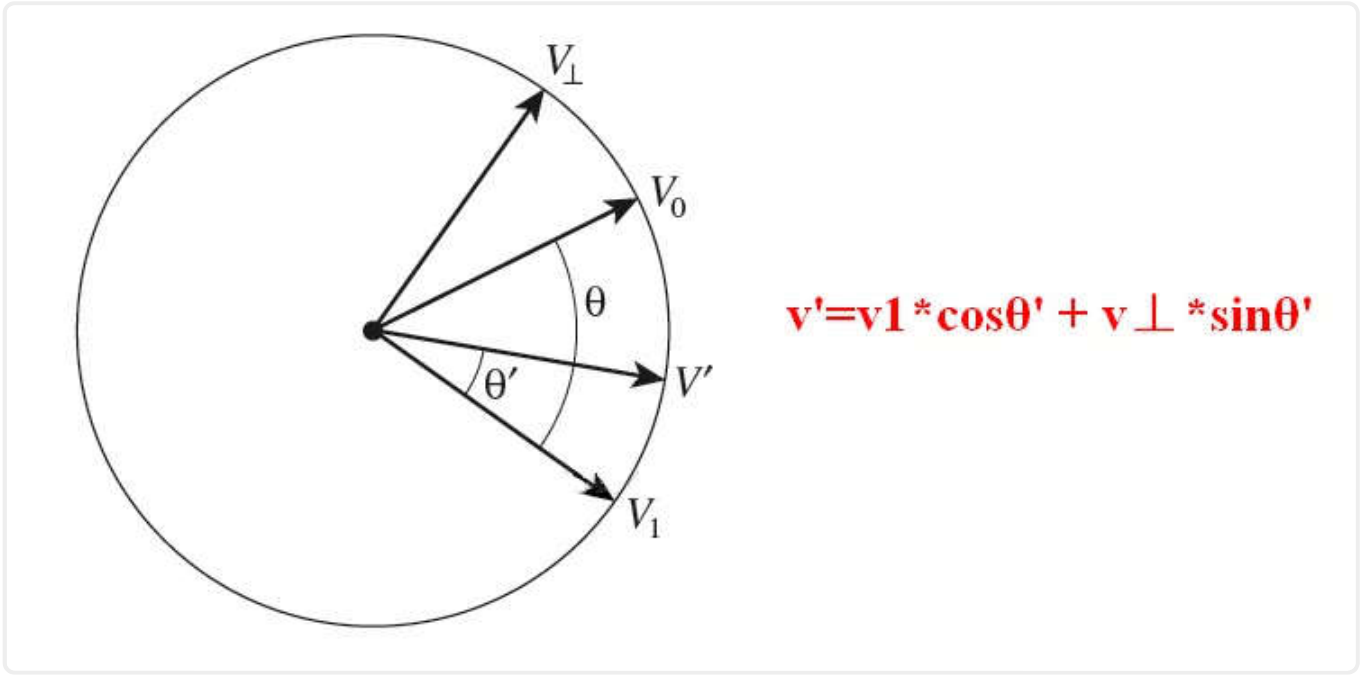
师兄：哈哈，别着急，这个方法可行，但是编程稍微复杂点，计算量也大，还有一种实现四元数的球面插值计算方式，要简单很多，留给你当做作业练习啦，搞定作业，你就可以直接用来做Slerp插值啦！

编程练习

作业练习1：前面四元数球面线性插值方法比较复杂，下面是它的简化版求解方法，请证明。

假设v0, v1是两个四元数，其夹角为θ，假设在它们中间进行四元数插值结果为v'，v'和v1之间夹角为θ ‘ < θ，记v⊥是垂直于v1的四元数向量，证明：

$v' = v1\cos\theta' + v\perp\sin\theta'$



作业练习2：编程实现四元数球面线性插值。

我们用智能手机采集了图像序列和IMU数据，由于IMU帧率远大于图像帧率，需要你用Slerp方法进行四元数插值，使得插值后的IMU和图像帧对齐。

已知某帧图像的时间戳为：t =700901880170406

离该图像帧最近的前后两个时刻IMU时间戳为：t1 = 700901879318945，t2 = 700901884127851

IMU在t1, t2时刻测量得的旋转四元数为：

q1x=0.509339, q1y=0.019188, q1z=0.049596, q1w=0.858921

q2x=0.509443, q2y=0.018806, q2z=0.048944,q2w=0.858905

根据上述信息求IMU对齐到图像帧的插值后的四元数。

参考结果已经给出。

本文参考：

高翔《视觉SLAM十四讲》

https://zhuanlan.zhihu.com/p/47396001