最近又來研究研究有關飛行姿態解算的算法，看看當時沒搞懂的四元數算法。主要也是學了網上幾位大神的資源，才能小有所得。在這先附上參考一些大神心得的網址：
https://zh.wikipedia.org/wiki/四元數 四元數維基百科
https://zh.wikipedia.org/wiki/向量 向量維基百科
【Unity技巧】四元數（Quaternion）和旋轉
http://blog.csdn.net/candycat1992/article/details/41254799
學習筆記—四元數與歐拉角之間的轉換
http://blog.csdn.net/candycat1992/article/details/41254799
[小應用]GD32F103+MPU9150 四軸飛行器第一步：9DOF姿態融合-GigaDevice GD32 MCU論壇
http://bbs.21ic.com/icview-605405-1-1.html

很多解釋上面帖子裡都有了。我想從另一個方向來寫這篇文章。

先來說說我瞭解這個東西的經過吧。

說起對飛行姿態的解算，其實是相當複雜的一個過程，並不像我們一般認為的那麼簡單。首先，一開始（包括我自己）一般可能有個誤區：

姿態不就一個傾角嘛（類比斜面）？所以感覺只需要求一個傾斜角而已，感覺挺簡單的。仔細一想，發現其實有姿態有2個角度， 可以想像一個滑梯再像側面翻滾一下，這樣其實對於空間思維不太好的人已經有一點難度了。然後發現還需要再加一個是描述物體本身自旋的姿態。那麼最後的結論為，姿態總共有3個軸。

那麼以我本身的知識來說，如何能夠檢測出物體姿態的這3個變量呢？
總共需要3個傳感器。

第一步：+加速度傳感器。

說起測量傾斜角，一般能看見的都使用水泡，鉛垂線之類的東西。有傳感器使用經驗的第一感覺就會覺得使用重力傳感器就OK啦。

首先說說重力傳感器的原理，這裡說的重力傳感器又叫加速度傳感器。
加速度計和陀螺儀指南(很詳細的介紹)
http://blog.csdn.net/lovewubo/article/details/9084291

瞭解了這個東西以後，那麼問題來了，僅僅靠加速度計能否完成角度的測量？答案是在靜態情況下可以，在動態情況下不可以。

加速度傳感器，從這個名字（以及上面的原理）也可以看出，其測不是重力，而是重力引起的類似加速度帶來的效應。所以對於其他的加速度同樣會有讀數（運動狀態的改變），特別在震動的時候（震動狀態），該傳感器會有非常大的數據變化，此時的數據難以反應重力的實際值，所以結論是單靠加速度傳感器無法完成姿態解算。

第二步：+陀螺儀

既然單靠加速度傳感器無法完成姿態解算，那還需要添加哪些傳感器？通過上面資料 我們可以找到至少還需要的一個傳感器，陀螺儀。

陀螺儀測量出的數據為繞各個軸的旋轉角速度。

通過高等數學的知識可以得出，對角速度進行積分，可以得到旋轉角度。把旋轉角度加到之前測出的姿態上，會得到一個新的姿態，設為姿態A，通過加速度傳感器可以算出來一個姿態B，這樣將這兩個姿態一定融合，就可以得到一個比較準確的姿態，這也就是之後我們可以進行姿態融合的基礎。

第三步：+地磁場傳感器

其實我們還缺一個傳感器，地磁場傳感器，其實他有個通俗的名字：電子羅盤。

　　說到這，可能有人就已經知道為什麼需要這個傳感器了。當加速度傳感器完全水平的時候，可以預料，重力傳感器無法分辨出在水平面旋轉的角度即繞Z軸的旋轉無法顯示出來，此時只有陀螺儀可以檢測。

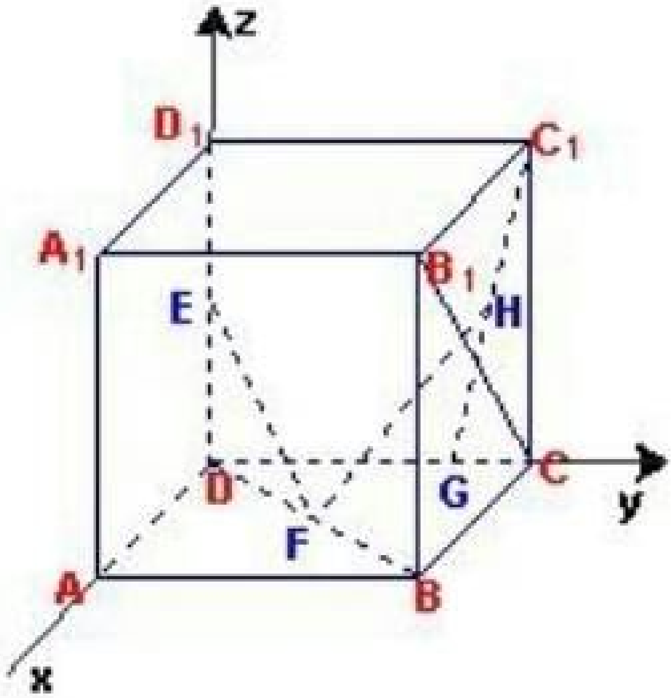

图 1　立方体

　　於是又回到第一步時的問題。陀螺儀雖然動態十分快速，但由於其工作原理是積分，所以在靜態會有累計誤差，表現為角度會一直增加或者一直減少。。
於是我們會需要一個在水平位置能確認朝向的傳感器，這就是如今IMU（慣性導航單元）必備的第三個傳感器，地磁場傳感器，通過這3個傳感器的相互校正，我們終於在大的理論上可以得到比較準確的姿態參數了。

總結一下前三步：
1、確定姿態我們需要三個軸上旋轉的數據。
2、我們需要三個傳感器，分別為3軸加速度傳感器，3軸陀螺儀，以及3軸地磁場傳感器。

　　完成這三步，可以說在大體姿態解算的框架上我已經有了概念，但具體怎麼做還是兩眼一抹黑啊。於是本著站在巨人的肩膀上做事的原則，我又開始漫長的資料搜索以及篩選。於是有了本文。

第四步 瞭解什麼叫姿態
　　請各位自帶筆和紙，複習或者預習大學高等數學，線性代數，複變函數，等數學知識，聽我慢慢回憶的學習的過程。。。

　　既然我一直說姿態解算，姿態解算，那到底什麼是姿態。

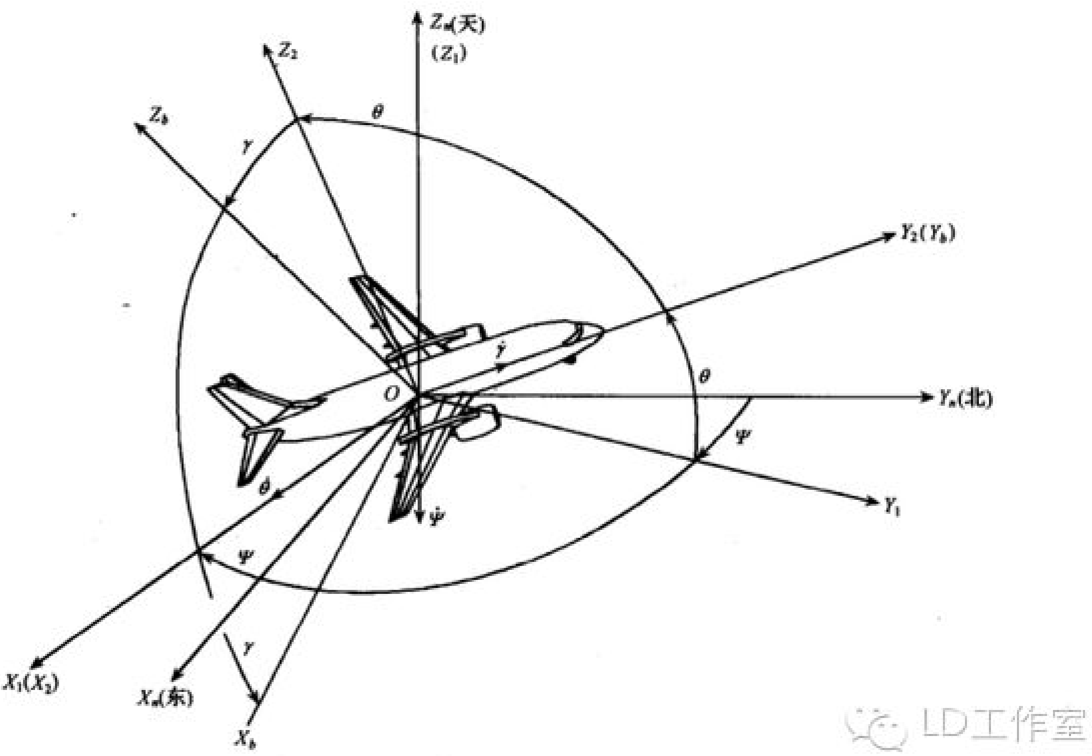

图 1.2.3 飞机空间角位置的确定

上圖，我們想像一個飛機在天空飛行，他可能側傾，可能仰俯，也可能旋轉，這就是我們平時所說的姿態，那麼這個姿態抽象出來意味著什麼呢？這裡是重點啦：

姿態，就是一個坐標系與另一個坐標系的轉換關係。

這個可能比較抽象哈。

首先在飛機上建立一個坐標系，X軸為機翼的方向，Y軸為機頭的方向，Z軸垂直於飛機，這個坐標系是隨著飛機姿態改變而變動的，此時就要求飛機的姿態，就等價於求出這個坐標系，那麼如何得到這個坐標系？要得到一個新的坐標系，首先要有一個參考坐標系，一般選取Y軸正向為正北，x正向為正東，z軸垂直於地平面了，給他取個名字，就叫地理坐標系吧。

所以此時，我們所說的姿態其實就是飛機坐標系和地理坐標系（是固定的）的一種關係。

現在我們終於可以進入第五步啦。

第五步：

如何表示這種關係。找到了表示這種關係的方法，就可以利用這個方法隨意轉換這2個坐標系。先上結論吧。

有三種關係表示方式：

1、歐拉角
2、餘弦矩陣
3、四元數

1、歐拉角

歐拉角是很直觀的，一個物體的旋轉，可以分解到三個坐標軸上的旋轉。這三個旋轉角度就是歐拉角。

在慣性系統中一般把這三個角度分別稱為roll，pitch，yaw。上個直觀的圖，很容易理解。

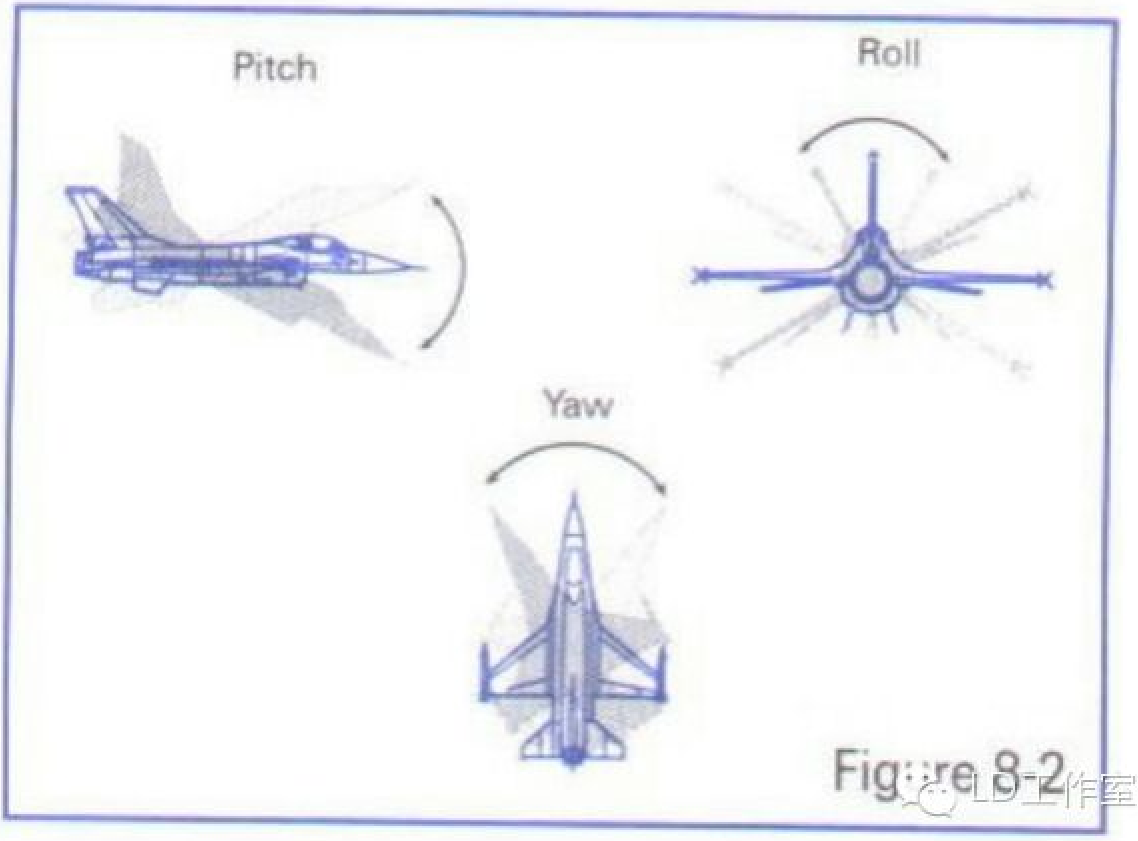

Fig::re 8-2

用歐拉角來描述物體的旋轉不光需要有角度，還需要有旋轉順序，一般旋轉順序是先yaw後pitch，再roll反應到坐標軸上就是先繞Z軸旋轉，再繞X軸旋轉，最後繞Y軸旋轉。為什麼有順序呢？是否可以沒順序？如果身邊有東西可以轉轉看，這個問題之後在理論上會進行說明。

需要注意的是 yaw pitch roll 都是對應的固定的參考系 也就是上面說的地理坐標系而言，每次新的姿態坐標系都是由地理坐標系通過歐拉角旋轉得到的。
這樣我們就用歐拉角表示了物體的姿態。

歐拉角是有很多優點的。但是也有致命的缺點，那就是 Gimbal Lock（萬向節死鎖），要 理解 Gimbal Lock所說的情況（可能有點難）讓我們看個現實中的場景。

假如我們有一個望遠鏡和一個用來放望遠鏡的三腳架，（我們將）三腳架放在地面上，使支撐望遠鏡的三腳架的頂部是平行於地平面（參考平面）的，以便使得豎向的旋轉軸（記 為 x 軸）是完全地垂直於地平面的。現在，我們就可以將望遠鏡饒 x 軸旋轉 360度，從而觀 察 （以望遠鏡為中心的） 水平包圍圈的所有方向。 通常將正北朝向方位角度記為 0 度方位角。 第二個坐標軸，即平行於地平面的橫向的坐標軸（記為y 軸）使得望遠鏡可以饒著它上下旋轉，通常將地平面朝向的仰角記為 0 度，這樣，望遠鏡可以向上仰+90 度指向天頂，或者向 下-90 度指向腳底。好了，萬事俱備。現在，天空中（包括地面上）的每個點只需要唯一的 一對 x 和 y 度數就可以確定。比如 x=90 度,y=45 度指向的點是位於正東方向的半天空上。 現在，看看萬向節死鎖是怎麼發生的。一次，我們探測到有一個飛行器貼地飛行，位於望遠 鏡的正東方向（x=90 度，y=10 度），朝著我們直飛過來，我們跟蹤它。飛行器飛行方向是 保持 x 軸角度 90 度不變，而 y 向的角度在慢慢增大。隨著飛行器的臨近，y 軸角增長的越 來越快且當 y向的角度達到 90 度時（即將超越），突然它急轉彎朝南飛去。這時，我們發現 我們不能將望遠鏡朝向南方（在只繞一個軸旋轉的情況下），因為此時 y向已經是 90度！造成我們失去跟蹤目標。這就是 萬向節死鎖！

為什麼說不能將望遠鏡朝向南方呢，讓我們看看坐標變化，從開始的（x=90度，y=10 度）到（x=90 度，y=90 度），這個過程沒有問題，望遠鏡慢慢轉動跟蹤飛行器。當飛行器 到達（x=90

度，y=90 度）後，坐標突然變成（x=180 度，y=90 度）（因為朝南），x 由 90 突變成 180 度，所以望遠鏡需要饒垂直軸向 x 軸旋轉 180-90=90 度以便追上飛行器,但此時，望遠鏡已經是平行於 x 軸，我們知道饒平行於自身的中軸線的的旋轉改變不了朝向，就像擰 螺絲一樣，螺絲頭的指向不變。所以望遠鏡的指向還是天頂。而後由於飛行器飛遠，坐標變 成（x=180 度，y<90 度）時，y 向角減小，望遠鏡只能又轉回到正東指向，望'器'興嘆。

這 說明用 x,y旋轉角（又稱歐拉角）來定向物體有時並不能按照你想像的那樣工作，象上面的例子中從（x=90度，y=10度）到（x=90 度，y=90度），按照歐拉角旋轉確實可以正確地定 向，但從（x=90度，y=90度）到（x=180 度，y=90 度），再到（x=180度，y<90 度）,按照 歐拉角旋轉後的定向並非正確。我的理解是坐標值的變化和飛行器空間的位置變化一一對 應，但是從（x=90 度，y=90 度）到（x=180 度，y=90 度），再到（x=180 度，y<90 度）這個變化，飛行器位置是連續的變化，但坐標值的變化卻不是連續的（從 90 突變到 180），其原因在於（x=90 度，y=90 度）和（x=180 度，y=90 度）甚至和（x=任意 度，y=90 度）這些不同的坐標值對應空間同一個位置，這種多個坐標值對應同一個位置的 不一致性是造成死鎖的根源。

上面是 2 維坐標系中的例子， 同樣， 對於 3 維的也一樣。 比如有一個平行於 x 軸的向量，我們先將它饒 y旋轉直到它平行於 z 軸，這時，我們會發現任何饒 z的旋轉都改變不了向量 的方向，即萬向節死鎖，所以說傳統的歐拉角是不能做到全姿態解析的。

2.方向餘弦矩陣。

說方向餘弦矩陣之前，先討論方向餘弦。

一個向量的方向（姿態）我們可以用他在參考坐標系（地理坐標系）各個軸向的夾角的餘弦來表示（及在各個軸的投影）。

類似的 一個坐標系 可以看成是3個向量組成，所以三個向量分別在坐標軸上的投影可以用來表示一個坐標系與參考坐標系的關係。這總共9個方向餘弦組成了一個三階矩陣，其對應方式如下圖。

$$\mathbf{C}_b^n = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

第 i 行、 j 列的元素表示參考坐標系 i 軸和姿態坐標系 j 軸夾角的餘弦。

事實上 方向餘弦和歐拉角沒有本質區別，因為方向餘弦實際上就是用歐拉角表示的。

下面附上推倒具體表達式的方法

先從二維坐標系轉換開始。

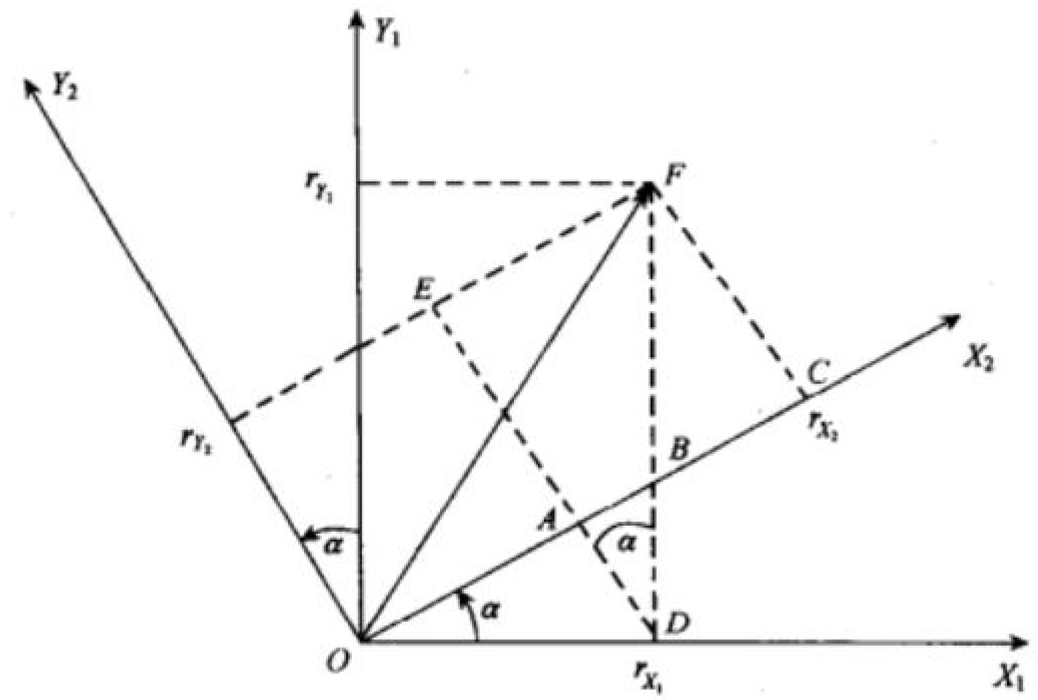

图 1.2.1　坐标系间的变换关系

推廣到三軸的單次旋轉，我們用矩陣表示為（繞Z軸旋轉）：

将上述三式写成矩阵形式：

$$\begin{bmatrix} r_{X_2} \\ r_{Y_2} \\ r_{Z_2} \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{X_1} \\ r_{Y_1} \\ r_{Z_1} \end{bmatrix} \tag{1.2.1}$$

這裡要說一下矩陣的含義，C2 1表示坐標系 1 到坐標系 2 的變換矩陣，那麼有

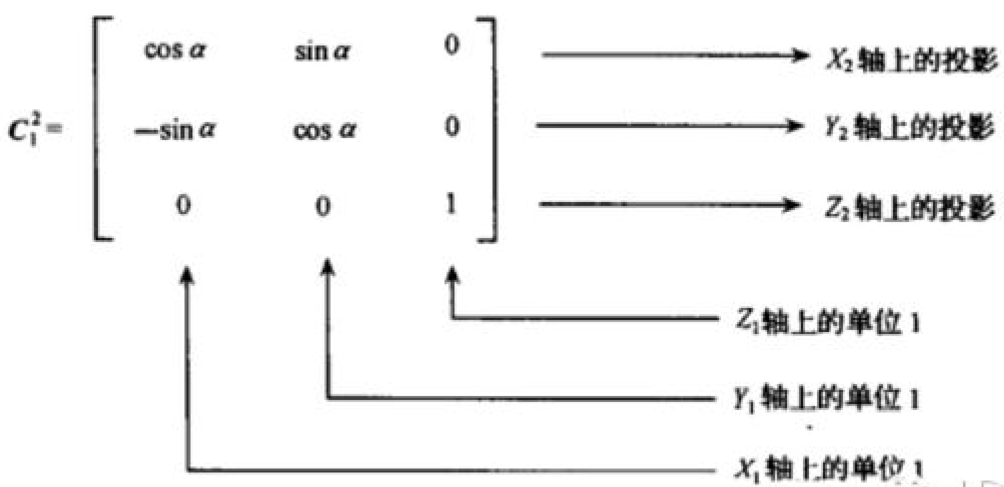

图 1.2.2　坐标系 1 和坐标系 2 之间的投影关系

這樣我們可以得到3個變換矩陣

$$C_n^1=\begin{bmatrix}\cos\Psi & -\sin\Psi & 0\\ \sin\Psi & \cos\Psi & 0\\ 0 & 0 & 1\end{bmatrix}\quad C_1^2=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta & \sin\theta\\ 0 & -\sin\theta & \cos\theta\end{bmatrix}\quad C_2^b=\begin{bmatrix}\cos\gamma & 0 & -\sin\gamma\\ 0 & 1 & 0\\ \sin\gamma & 0 & \cos\gamma\end{bmatrix}$$

分別為單獨繞Z軸旋轉，繞X軸旋轉，繞Y軸旋轉。

實際上，兩坐標系任何複雜的角位置關係都可以看做有限次基本旋轉的組合，變換矩陣 等於基本旋轉確定的變換矩陣的連乘（線性代數）， 連 乘的基本順序依據基本旋轉的順序向右排列。

之所以有順序是因為矩陣有「左乘」和「右乘」 之分（還是線性代數）。那麼我們得到：

$$C_n^b=C_2^bC_1^2C_n^1=\begin{bmatrix}\cos\gamma & 0 & -\sin\gamma\\ 0 & 1 & 0\\ \sin\gamma & 0 & \cos\gamma\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta & \sin\theta\\ 0 & -\sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}\cos\Psi & -\sin\Psi & 0\\ \sin\Psi & \cos\Psi & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$=\begin{bmatrix}\cos\gamma\cos\Psi+\sin\gamma\sin\Psi\sin\theta & -\cos\gamma\sin\Psi+\sin\gamma\cos\Psi\sin\theta & -\sin\gamma\cos\theta\\ \sin\Psi\cos\theta & \cos\Psi\cos\theta & \sin\theta\\ \sin\gamma\cos\Psi-\cos\gamma\sin\Psi\sin\theta & -\sin\gamma\sin\Psi-\cos\gamma\cos\Psi\sin\theta & \cos\gamma\cos\theta\end{bmatrix}$$

最後的矩陣就是完整的餘弦矩陣。γ、θ、ψ就是歐拉角啦。

至此我們解釋了為什麼歐拉角會有旋轉的順序之分。從以上數學計算可以看出不同的旋轉次序會帶來不同的結果。

四元數：

四元數要介紹的太多了。。因為他優點有很多，利用起來也很方便，但是理解起來太抽象了。

百度四元數，一開始看到的就是四元數來歷，還有就是四元數的基本計算。

對於 來歷，還是想說一下，四元數（Quaternions）是由威廉·盧雲·哈密爾頓(William Rowan Hamilton,1805-1865）在 1843 年愛爾蘭發現的數學概念（百度百科）。

將實數域擴充到複數域，並用複數來表示平面向量，用複數的加、乘運算表示平面向量 的合成、伸縮和旋，這就是我們熟知的複數的二維空間含義，所以人們會繼續猜想，利用三 維複數不就可以表達三維空間的變換了嗎，歷史上有很多數學家試圖尋找過三維的複數， 但 後來證明這樣的三維複數是不存在的。

有關這個結論的證明，我沒有查到更明確的版本， 據 《古今數學思想》中的一個理由，三維空間中的伸縮旋轉變換需要四個變量來決定：兩個變 量決定軸的方向，一個變量決定旋轉角度，一個變量決定伸縮比例。這樣，只有三個變量的 三維複數無法滿足這樣的要求。但是歷史上得到的應該是比這個更強的結論，即使不考慮空 間旋轉，只從代數角度來說，三維的複數域作為普通複數域的擴張域是不存在的。

並且， 據 《古今數學思想》敘述，即使像哈密爾頓後來引入四元數那樣，犧牲乘法交換律，這樣的三 維複數也得不到。經過一些年的努力之後， Hamilton 發現自己被迫應作兩個讓步，第一個 是他的新數包含四個份量，而第二個是他必須犧牲乘法交換律。 （ 《古今數學思想》第三冊 177 頁 ）但是四元數用作旋轉的作用明顯，簡化了運算，而且避免了 Gimbal Lock。

四元數 是最簡單的超複數，我們不能把四元數簡單的理解為 3D 空間的矢量，它是 4 維空間中的 的矢量，也是非常不容易想像的。

那什麼是四元數呢？

在維基百科中截取一段：

複數是由實數加上虛數單位 i 組成，其中

$$i^2 = -1$$

相似地，四元數都是由實數加上三個元素 i、j、k 組成，而且它們有如下的關係：

$$i^2 = j^2 = k^2 = ijk = -1$$

每個四元數都是 1、i、j 和 k 的線性組合，即是四元數一般可表示為

$$a + bi + cj + dk$$

要把兩個四元數相加只需將同類的係數加起來就可以，就像複數一樣。

那麼四元數如何表示旋轉呢？

再截取維基百科上的一段話

群旋轉

主條目：四元數與空間旋轉

非零四元數的乘法群在R3的實部為零的部分上的共軛作用可以實現轉動。單位四元數（絕對值為1的四元數）若實部為cos(t)，它的共軛作用是一個角度為2t的轉動，轉軸為虛部的方向。

提取幾個關鍵詞：

四元數表示轉動，首先其模應該為1，如下

$$|h| = \sqrt{h \cdot h^*} = \sqrt{a^2 + b^2 + c^2 + d^2}$$

這個值應該為1。

此時實部，也就是a若等於cos(t)則他的共軛作用為一個2t的轉動，轉軸方向為向量(b,c,d)的方向。

那什麼叫共軛作用？實際上就是四元數表示旋轉的方式。

那麼利用四元數代表旋轉是如何實現的，在載體系定義的一個矢量 r b 可以直接 利用四元數將其在參考系中表示為 rn 。首先定義一個四元數 rb'，它的虛部等於 rb的相應份量，標量份量為零：

$$\boldsymbol{r}^{\mathrm{b}} = \mathrm{i}x + \mathrm{j}y + \mathrm{k}z$$

$$\boldsymbol{r}^{\mathrm{b}'} = 0 + \mathrm{i}x + \mathrm{j}y + \mathrm{k}z$$

參考系中的 r n' 表示為

$$\boldsymbol{r}^{\mathrm{n}'} = q\boldsymbol{r}^{\mathrm{b}'}q^*$$

這也就是共軛作用。，其中 q = a + bi + cj +dk，q* 為q復共軛，q = a - bi - cj -dk及因此有：

$$\begin{aligned}\boldsymbol{r}^{\mathrm{n}'} = (a + \mathrm{i}b + \mathrm{j}c + \mathrm{k}d)(0 + \mathrm{i}x + \mathrm{j}y + \mathrm{k}z)(a - \mathrm{i}b - \mathrm{j}c - \mathrm{k}d) = \\ 0 + \{(a^2 + b^2 - c^2 - d^2)x + 2(bc - ad)y + 2(bd + ac)z\}\mathrm{i} + \\ \{2(bc + ad)x + (a^2 - b^2 + c^2 - d^2)y + 2(cd - ab)z\}\mathrm{j} + \\ \{2(bd - ac)x + 2(cd + ab)y + (a^2 - b^2 - c^2 + d^2)z\}\mathrm{k}\end{aligned}$$

寫成矩陣式：

$$C = \begin{bmatrix} (a^2+b^2-c^2-d^2) & 2(bc-ad) & 2(bd+ac) \\ 2(bc+ad) & (a^2-b^2+c^2-d^2) & 2(cd-ab) \\ 2(bd-ac) & 2(cd+ab) & (a^2-b^2-c^2+d^2) \end{bmatrix} \quad (3.59)$$

因為都表示旋轉，所以這個矩陣理論上應該和餘弦矩陣是等效的，從而就能計算歐拉角了。

$$C_b^n = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} =$$

$$\begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi+\sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi+\cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi+\sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi+\cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} =$$

$$\begin{bmatrix} (a^2+b^2-c^2-d^2) & 2(bc-ad) & 2(bd+ac) \\ 2(bc+ad) & (a^2-b^2+c^2-d^2) & 2(cd-ab) \\ 2(bd-ac) & 2(cd+ab) & (a^2-b^2-c^2+d^2) \end{bmatrix} \quad (3.63)$$

上圖顯示了四元數和餘弦矩陣的關係。
這樣反求出歐拉角如下圖：

$$\phi = \arctan\left[\frac{c_{32}}{c_{33}}\right]$$

$$\theta = \arcsin[-c_{31}]$$

$$\psi = \arctan\left[\frac{c_{21}}{c_{11}}\right]$$

此時就可以根據四元數反求出歐拉角了。

如果喜歡觀看類似科技新奇事物，以及瞭解創客圈最新資訊，或者您對Arduino有所耳聞，可以關注我們微信公眾號，一定會帶給您最新的資訊，最實用的教程，以及創客最新的玩意。