Introdução a Robótica

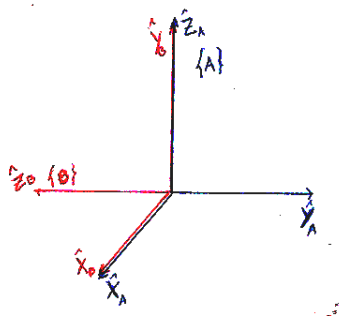
Jhonatan da Silva

Instituto Federal de Santa Catarina. Janeiro 21, 2017

Sumário

1 Matrizes de Rotação e Translação

Quando temos um frame de referência A e outro frame de referência B e queremos saber qual relação por exemplo do eixo x em A, $\hat{X}^A$ com o eixo x em B, $\hat{X}^B$, como na imagem abaixo,



podemos utilizar de matrizes de rotação, geralmente uma matriz de rotação R, tem a seguinte forma geral.

$$R := \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Assim com matrizes de rotação podemos escrever a relação entre $\hat{X}^A$ e $\hat{X}^B$ em termos de ${}^A_B R$, onde

$${}^A\hat{X}_B = {}^A_B R \, {}^B\hat{X}_B$$

e ${}^A_B R$ é definido como

$${}^A_B R = \begin{bmatrix} {}^A\hat{X}_B & {}^A\hat{Y}_B & {}^A\hat{Z}_B \end{bmatrix}$$

Nos resta determinar ${}^A\hat{X}_B \;\; {}^A\hat{Y}_B \;\; {}^A\hat{Z}_B$,

$${}^A\hat{X}_B = \begin{bmatrix} \hat{X}_B \hat{X}_A \\ \hat{X}_B \hat{Y}_A \\ \hat{X}_B \hat{Z}_A \end{bmatrix}$$

1.1 Computação simbólica com python

Como faríamos se quiséssemos definir uma matriz simbólica tal qual igual a R, Em uma linguagem de programação como python ? O python possui uma biblioteca em específico chamada sympy que permite que o usuário compute matemática simbólica, um simples código segue abaixo,

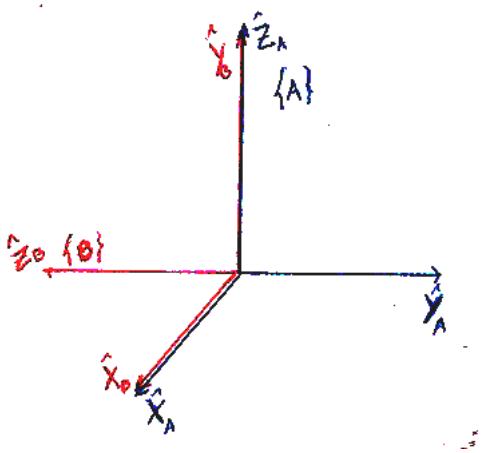
```
from sympy import symbols,Matrix

(r11,r12,r13,
r21,r22,r23,
r31,r32,r33) = symbols('r11 r12 r13 r21 r22 r23 r31 r32 r33')

R = Matrix([[r11,r12,r13],[r21,r22,r23],[r31,r32,r33]])
```

Que retornará

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

1.2 Definindo a matriz de rotação

Definindo os dois frames de referência pela imagem abaixo



Como poderiamos determinar ${}^A_B R$ tal que $\hat{X}^A \mapsto \hat{X}^B$ Pela definição anterior de que

$$ {}^A_B R = \begin{bmatrix} {}^A\hat{X}_B & {}^A\hat{Y}_B & {}^A\hat{Z}_B \end{bmatrix} $$

Pela matriz podemos ver que cada coluna na verdade é a cordenada de B em relação a A, como vemos na imagem as coordenadas de $\hat{X}^A$ e $\hat{X}^B$ são as mesmas, já, $\hat{Y}^B$ tem coordenadas em $\hat{Z}^A$ enquanto $\hat{Z}^B$ tem coordenadas em $\hat{Y}^A$. Então podemos escrever ${}^A_B R$ como

$$ {}^A_B R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} $$

E como R é ortonormal sabemos que ${}^A_B R^T = {}^B_A R = {}^A_B R^{-1}$

1.3 Exemplo de rotação com python

Quando lida-se com computação numérica ao contrário da simbólica usa-se o numpy, usualmente importa-se o numpy como np como se vê no código abaixo

```
import numpy as np

R = np.matrix('1 0 0; 0 0 -1;0 1 0')
```

Que retornará

$$ R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} $$

1.4 Usando C++

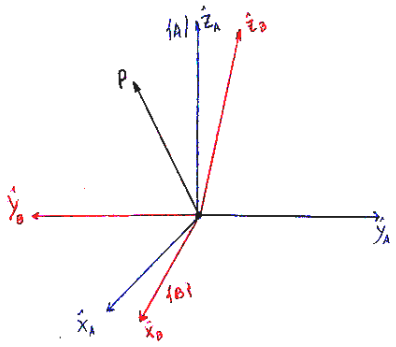
Como ficaria a sintaxe do programa escrito em python em C++ ? Para o código em questão foi usado a biblioteca Eigen que permite trabalhar com algebra linear em C++

```
#include <iostream>
#include <Eigen/Dense>

using Eigen::MatrixXd;
using namespace std;

int main()
{
    MatrixXd R(3,3);
    R << 1, 0, 0,
         0, 0, -1,
         0, 1, 0;
    cout << R;
}
```

Esse código tem o mesmo retorno que o anterior.

1.5 Rotação geral

E quando temos um ponto P em termos de B e queremos saber como definir o mesmo em termos de A, por exemplo na imagem

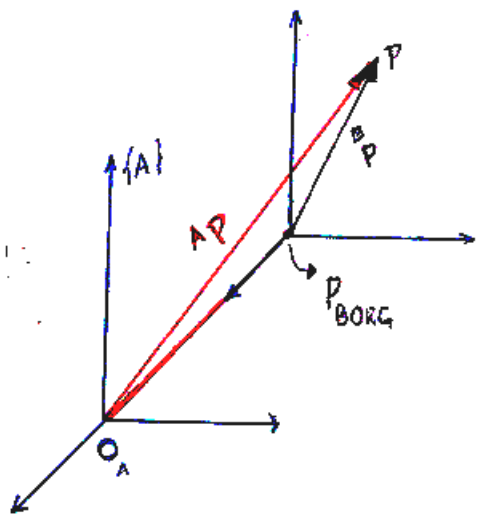


Podemos definir o ponto P em A como ${}^{A}P$ assim temos

$$ {}^{A}P = {}^{A}_{B}R\,{}^{B}P $$

1.6 Translações

Quando temos um ponto P em um frame A e queremos transladá-lo sem rotacionar ou mudar de frame como mostra a imagem a seguir

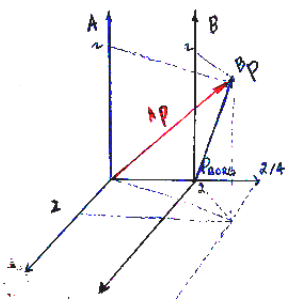


Podemos definir ${}^{A}P$ como

$$ {}^{A}P = {}^{B}P + {}^{A}P_{BORG} $$

1.7 Exemplo de translação em python

Como nenhum exemplo prático foi definido na seção anterior antes de fazer o código precisamos de algumas definições, segundo a imagem



Então temos o primeiro frame de referência A e o segundo B, que está deslocado duas casas no eixo y, então nosso ponto de origem para B é $(0,2,0)$, pode-se notar que em relação ao frame B, a posição y é 2 mas em relação a A é 4, agora temos algumas definições

$$^{B}P = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}$$

$$P_{BORG} = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}$$

$$^{A}P = Pb + Pborg$$

E agora podemos fazer o código para computar ^{A}P

```
import numpy as np

Pborg = np.matrix('0; 2; 0')
Pb = np.matrix('2; 2; 2')

Pa = Pb + Pborg
```

Que irá retornar

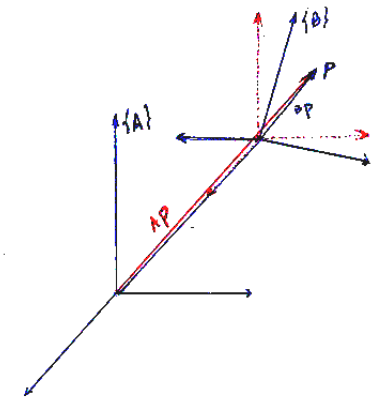
$$^{A}P = \begin{bmatrix} 2 \\ 4 \\ 2 \end{bmatrix}$$

1.8 Exemplo de translação em C++

```
#include <iostream>
#include <Eigen/Dense>

using Eigen::MatrixXd;
using namespace std;

int main()
{
    MatrixXd Pborg(3,1);
    Pborg << 0,
             2,
             0;
    cout << Pborg << endl;
    MatrixXd Pb(3,1);
    Pb << 2,
          2,
          2;
    cout << Pb << endl;
    cout << "\n\n Ponto final" << endl;
    MatrixXd Pa(3,1);
    Pa << Pborg + Pb;
    cout << Pa;
}
```

1.9 Transformação Geral

Se temos um caso onde foi feita uma translação e uma rotação em relação a um novo frame, e queremos saber qual é o novo ponto em relação ao frame original, como na imagem abaixo



Primeiro temos que fazer a matriz de rotação ${}^A_B R$ pois o novo ponto está em um frame diferente do original, então não podemos apenas fazer a soma, depois de fazer ${}^A_B R$ aplicado em ${}^B P$ podemos fazer a translação somando com ${}^A P_{BORG}$, em uma forma geral podemos escrever

$$ {}^A P = {}^A_B R {}^B P + {}^A P_{BORG} $$

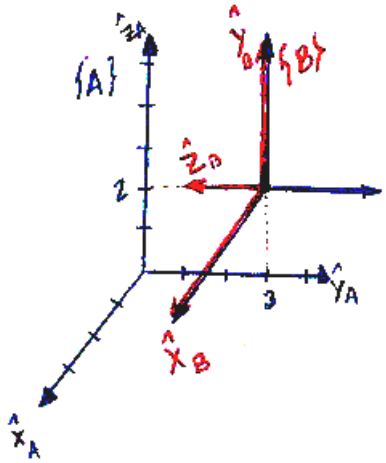
1.9.1 A Transformação Homogênea

Seria interessante ter uma matriz que resolvesse o problema geral, onde temos a rotação e a translação, porque computacionalmente falando, é mais interessante ter uma multiplicação de uma matriz com um vetor que ter várias operações. A transformação homogênea resolve esse problema, expandindo para uma matriz $(4,4)$ podemos reescrever a expressão ${}^A P = {}^A_B R {}^B P + {}^A P_{BORG}$ em forma matricial

$$ \begin{bmatrix} {}^A P \\ 1 \end{bmatrix} = \begin{bmatrix} {}^A_B R & {}^A P_{BORG} \\ 00 & 0 \end{bmatrix} \begin{bmatrix} {}^B P \\ 1 \end{bmatrix} $$

1.9.2 Exemplo

Um exemplo prático de como funciona a transformação homogênea, pela imagem abaixo,



Temos um frame de referência A e outro B, pelo que se ve na image o frame de referêcia continuou no mesmo plano xy, deslocou 3 casas em direção ao eixo y e 2 em relação ao eixo z, definindo a translação como q temos que

$$ q = \begin{bmatrix} 0 \\ 3 \\ 2 \end{bmatrix} $$

Agora precisamos definir ${}^A_B R$, que tem a mesma rotação do primeiro exemplo de rotações

$$ {}^A_B R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} $$

Agora podemos definir uma transformação homogênea ${}^A_B T$ de tal modo

$$ {}^A_B T = \begin{bmatrix} {}^A_B R & {}^A P_{BORG} \\ 00 & 0 \end{bmatrix} $$

$$ {}^A_B T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Se definirmos um ponto ${}^B P$

$$ {}^B P = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} $$

Onde a quarta cordenada de ${}^B P$ não é levada em conta no final. Podemos saber o novo ponto ${}^A P$ fazendo

$$ {}^A P = {}^A_B T {}^B P $$

$$ {}^A P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad {}^A P = \begin{bmatrix} 0 \\ 2 \\ 3 \\ 1 \end{bmatrix} $$

$$ \therefore \ {}^A P = \begin{bmatrix} 0 \\ 2 \\ 3 \end{bmatrix} $$

1.10 Exemplo de transformação homogênea com python

Precisamos definir a matriz de transformação ${}^A_B T$ o ponto ${}^B P$ e agora que temos essas duas informações podemos computar ${}^A P$ com a multiplicação dos dois com $np.dot({}^A_B T, {}^B P)$, e do resultado final pegar apenas as 3 primeiras coordenadas que resultam no ponto desejado.

```python
import numpy as np

T = np.matrix('1 0 0 0; 0 0 -1 3; 0 1 0 2; 0 0 0 1')

Pb = np.matrix('0;1;1;1')

Pa = np.dot(T,Pb)

Pa = Pa[0:3]
```

O código retornará

$$ {}^A P = \begin{bmatrix} 0 \\ 2 \\ 3 \end{bmatrix} $$

1.11 Exemplo de transformação homogênea com C++

```cpp
#include <iostream>
#include <Eigen/Dense>

using Eigen::MatrixXd;
using namespace std;

int main()
{
    MatrixXd T(4,4);
    T << 1, 0, 0, 0,
         0, 0, -1, 3,
         0, 1,  0, 2,
         0, 0,  0, 1;
    MatrixXd Pb(4,1);
    Pb << 0,
          1,
```

```
            1,
            1;
    MatrixXd Pa(4,1);
    Pa << T * Pb;
    MatrixXd P(3,1);
    int i = 0;
    for(i = 0; i < 3; i++)
    {
        P(i) =  Pa(i);
    }
    cout << P;
}
```

1.12 Ângulos de Euler

Antes de apresentar os ângulos de euler algumas definições devem ser feitas. Abaixo temos 3 frames de referência, como poderíamos achar a relação entre C e A ? Tendo em vista ${}^{C}P$, poderíamos fazer a transformação homogênea para o frame B de tal maneira,

$$ {}^{B}P = {}^{B}_{C}T\,{}^{C}P $$

Agora que temos ${}^{B}P$, podemos fazer a mesma transformação para obter ${}^{A}P$,

$$ {}^{A}P = {}^{A}_{B}T\,{}^{B}P $$

e como ${}^{B}P = {}^{B}_{C}T\,{}^{C}P$, podemos substituir na equação acima e obter dessa maneira

$$ {}^{A}P = {}^{A}_{B}T\,{}^{B}_{C}T\,{}^{C}P $$

Se usarmos as 3 transformações obtemos a Identidade

$$ {}^{A}_{B}T\,{}^{B}_{C}T\,{}^{C}_{A}T = I $$

1.12.1 Representação da posição

Temos 3 principais maneiras de representar a posição de um vetor Cartesiana = (x, y, z), Cilindrica = (ρ, θ, z), Esférica = (r, θ, ϕ)

1.12.2 Direction Cosine

Definimos um vetor x de tal maneira

$$ x_r = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} $$

onde r_1,r_2 e r_3 são as colunas da matriz de rotação.

1.12.3 Ângulos de Euler, Z-Y-X

Quando temos uma rotação no frame de referência como podemos saber como voltar ao original ? Seguindo a imagem abaixo Temos primeiro uma rotação sobre $\hat{Z}^B$ com um ângulo α essa rotação às vezes é chamada de *yaw*, depois uma rotação sobre $\hat{Y}^B$ com um ângulo β essa às vezes chamada como *pitch* e finalmente em $\hat{X}^B$ com γ às vezes chamada como *roll*. Temos que

$$ {}^{A}_{B'''}R = {}^{A}_{B'}R\,{}^{B'}_{B''}R\,{}^{B''}_{B'''}R = R_z(\alpha)R_y(\beta)R_x(\gamma) $$

1.12.4 Matrizes Z-Y-X

Quando falamos de ângulos de euler nos referimos a α β e γ, mas também às matrizes de rotação $R_z(\alpha)$,$R_y(\beta)$, $R_x(\gamma)$, que são a rotação sobre o eixo z, y e x respectivamente, definidas como, tendo em vista a declaração que $c\alpha := \cos(\alpha)$ e $s\alpha := \sin(\alpha)$

$$ R_z = \begin{bmatrix} c\alpha & -s\alpha & 0 \\ s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ R_y = \begin{bmatrix} c\alpha & 0 & s\alpha \\ 0 & 1 & 0 \\ s\alpha & 0 & c\alpha \end{bmatrix} $$

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\alpha & -s\alpha \\ 0 & s\alpha & c\alpha \end{bmatrix}$$

E obtemos R_{zyx} como

$$R_{zyx} = \begin{bmatrix} c\alpha c\beta & X & X \\ s\alpha c\beta & X & X \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}$$

1.13 Problema Inverso

Dada a matriz ${}^A_B R$ como podemos obter os ângulos de euler ? Temos que uma típica matrix de rotação tem essa cara

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Podemos igualar as matrizes

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} c\alpha c\beta & X & X \\ s\alpha c\beta & X & X \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}$$

Podemos começar com β se igualarmos temos que $r_{11} = c\alpha c\beta$, $r_{21} = s\alpha c\beta$ e $r_{31} = -s\beta$ como temos $c\beta$ como fator comum nas duas primeiras expressões podemos fazer

$$r_{11}^2 = c^2\alpha c^2\beta \,, \quad r_{21}^2 = s^2\alpha c^2\beta \quad , -r_{31} = s\beta$$

$$c^2\beta(c^2\alpha + s^2\alpha) = r_{11}^2 + r_{21}^2$$

$$c\beta = \sqrt{r_{11}^2 + r_{21}^2}$$

Como temos duas funções em sin e cos que defininem β podemos usar

$$\text{arctan2}(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2})$$

1.14 Parâmetros de Euler

Pela imagem temos W um vetor unitário define o eixo de rotação, junto com os ângulos de euler podemos definir seus parâmetros, que tentar diminuir os problemas com singularidade, fazendo com que sempre haja um valor maior que zero para ser computado, definimos seus 4 parâmetros ϵ_1 , ϵ_2 , ϵ_3 , ϵ_4 como

$$\epsilon_1 = W_x \sin(\theta/2)$$

$$\epsilon_2 = W_y \sin(\theta/2)$$

$$\epsilon_3 = W_z \sin(\theta/2)$$

$$\epsilon_4 = \cos(\theta/2)$$

Então definimos um novo vetor com os parâmetros de euler

$$x_e = \begin{bmatrix} W_x \sin(\theta/2) \\ W_y \sin(\theta/2) \\ W_z \sin(\theta/2) \\ \cos(\theta/2) \end{bmatrix}$$

No exemplo da imagem temos

$$W = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \quad \theta = 60^\circ$$

$$x_e = \begin{bmatrix} 1/2 \\ 0 \\ 0 \\ \sqrt{3}/2 \end{bmatrix}$$

e como estamos rotacionando sobre x

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1/2 & -\sqrt{3}/2 \\ 0 & \sqrt{3}/2 & 1/2 \end{bmatrix}$$