Robótica Industrial
Aula prática nº 3

Transformações geométricas em 3D
Uso de 'hipermatrizes' como sequência de transformações
Funções gerais para animação
Objetos poliédricos em Matlab
Ângulos de Euler

Vítor Santos

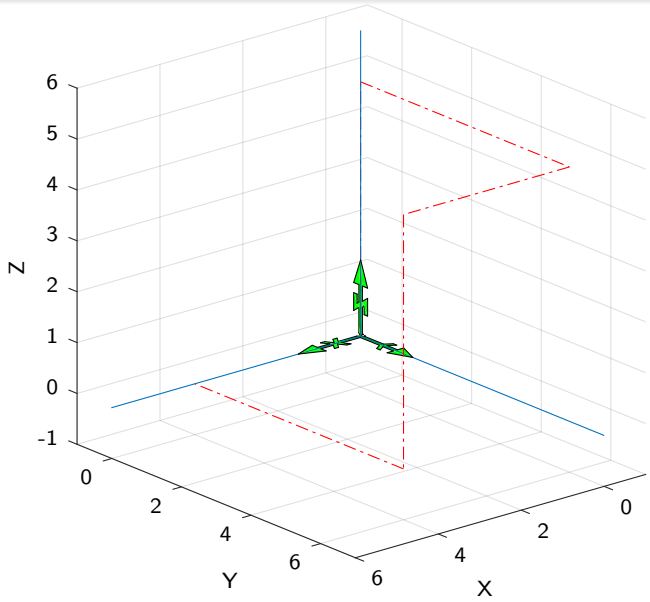
Universidade de Aveiro

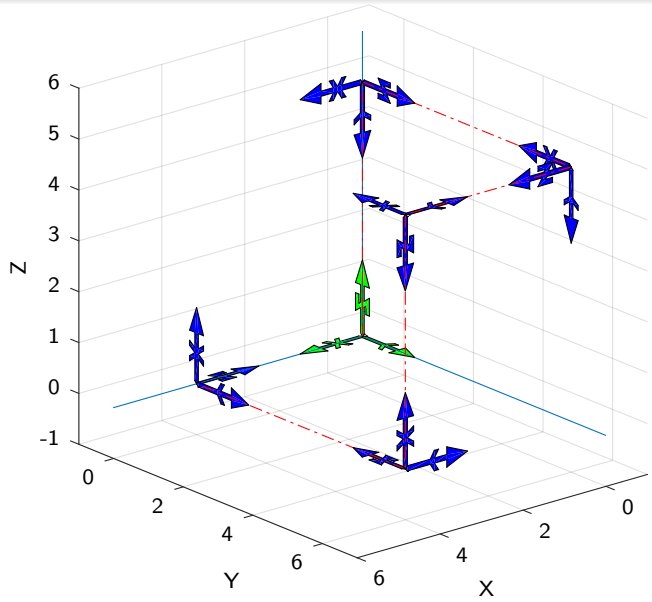
10 Out 2022

Exercício 1 - Representação de referenciais

Representar múltiplas configurações de um sistema de coordenadas

Usando a função fornecida `[P,F]=seixos3()`, representar o objeto `P` nas 6 configurações ilustradas mediante as transformações geométricas adequadas. As matrizes P e F podem ser usadas do seguinte modo: `patch('Vertices',P(1:3,:)','Faces',F,'FaceColor','b')`.
NB: Sugere-se começar com a representação base à esquerda!





As diversas posições podem ser obtidas de duas formas: por pré- ou pós-multiplicação. Sugere-se a pós-multiplicação porque aqui é mais fácil expressar as transformações no referencial local!

Exercício 2 - Funções vetoriais de rotação

Criar as novas funções

- `mrotx()`, `mroty()`, `mrotz()`

que aceitam vetores como argumentos e devolvem uma *hipermatriz* de transformações geométricas (uma matriz de transformação para cada valor do vetor na entrada; uma *hipermatriz* é, aqui, um tensor de ordem 3).

Exemplo para `mrotx()`

`A=mrotx(linspace(0,pi/2,2))` deverá devolver uma *hipermatriz* de 3 dimensões em que a terceira dimensão tem 2 folhas: uma para a rotação de 0 e outra para a rotação de pi/2:

$$A(:,:,1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A(:,:,2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Exercício 3 - Função vetorial de translação

Criar a nova função

- `mtrans(X,Y,Z)`

de filosofia similar às anteriores, mas que aceita 3 vetores como argumentos e devolve uma hipermatriz de transformações geométricas. Porém, esta função precisa de prever a situação em que os seus argumentos tenham dimensões diferentes.

Risco de diferença de dimensões entre X, Y e Z

Para evitar erros de execução, dentro da função, deve-se fazer o *padding* dos vetores para corrigir a situação se os argumentos não forem todos da mesma dimensão. Há outras soluções, mas uma solução possível é ajustar os vetores do seguinte modo:

- `m = max([numel(X), numel(Y), numel(Z)]);`
- `X(end:m)=X(end);` % etc.

Exercício 4 - Função `manimate()`

Criar uma nova função de animação `manimate()`

- `Tlast=manimate(h, P, Tcurr, Tset, ord)`
  - **h** – *handle* gráfico do objeto a animar (movimentar);
  - **P** – matriz de pontos do objeto (no formato homogéneo);
  - **Tcurr** – Matriz de transformação da posição inicial do objeto (início da animação);
  - **Tset** – hipermatriz de transformações geométricas com o conjunto dos passos intermédios para a animação (sucessivas posições do objeto P).
  - **ord** – indicador se as transformações presentes na hipermatriz são para ser feitas no referencial local (ord=1) ou no referencial global (ord=0);
  - **Tlast** – última posição (matriz de transformação) onde foi deixado o objeto no fim da animação.

Diferenças entre `Animate()` e `manimate()`

Contrariamente à função `Animate()` criada da aula anterior, esta função espera as matrizes T (em `Tset`) e não o vetor com os 6 incrementos. Inclui uma opção para indicar se as transformações são pré- ou pós-multiplicadas.

Exercício 5 - Animação com a função `manimate`

Fazer a animação com a sequência apresentada no exercício 1

Invocando a função `manimate()`, a partir do programa principal, fazer a animação ilustrada no exercício 1. Propõe-se o uso de 'hipermatrizes' de 4 dimensões para facilitar a indexação dos diversos passos no ciclo `for`.

Excerto de código exemplo com pré- e pós-multiplicações

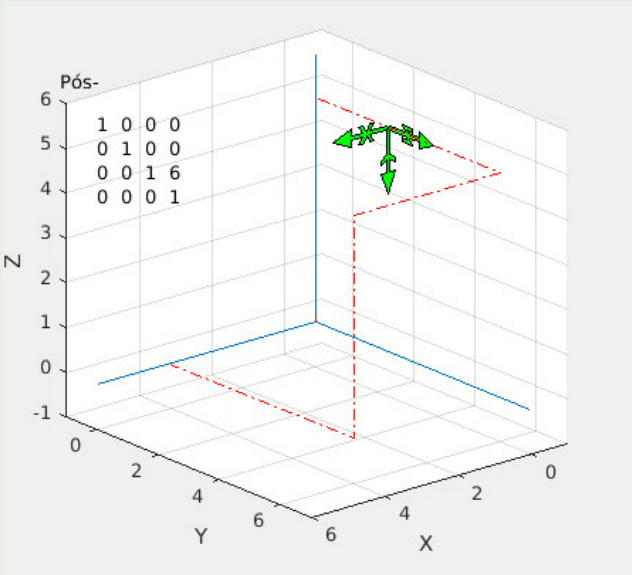
```
NN=10;
T(:,:,:,1)=mtrans(0,0,linspace(0,5,NN));
T(:,:,:,2)=mrotx(linspace(0,-pi/2,NN));
T(:,:,:,3)=mtrans(0,linspace(0,6,NN),0);
T(:,:,:,4)=mroty(linspace(0,pi/2,NN));
T(:,:,:,5)=mtrans(linspace(0,4,NN),0,0);
T(:,:,:,6)=mrotx(linspace(0,-pi/2,NN));
% ...
order=[0 1 0 1 0 1 ];  %1 -> pos-mult, 0 -> pre-mult
Tcurr=eye(4,4);  %Posição inicial (matriz identidade)
for n=1:size(T,4)
    Tcurr = manimate(h, P, Tcurr, T(:,:,:,n), order(n) );
    pause()
end
```

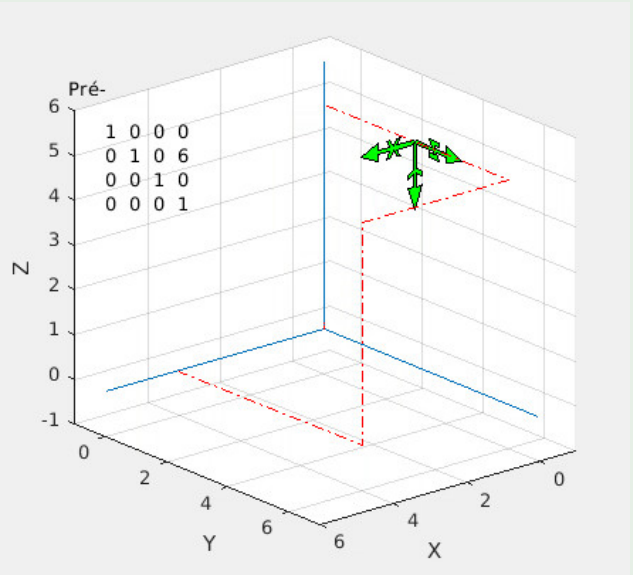
Ex. 5 - Duas formas de fazer as transformações

As transformações podem ser com pré- e pós-multiplicações

- As rotações neste exercício são muito mais fáceis de expressar nos referenciais locais (portanto com pós-multiplicações).
- As translações também são mais fáceis no referencial local (neste caso sempre ao longo do eixo Z);
- Mas neste problema em particular também se conseguem fazer facilmente no referencial global (pré-multiplicações) porque os movimentos são ao longo dos eixos.

Ilustram-se animações com os dois casos.





Exercício 6 - Objetos poliédricos em Matlab

Criar um objeto poliédrico - uma pirâmide

1. Definir lista de vértices
2. Definir lista de faces
3. Definir lista de cores das faces

Código base

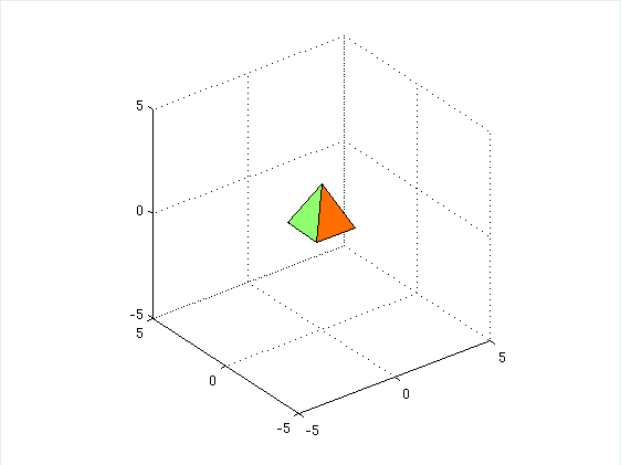
```
%Definition of vertices
points=[
         1 -1  0   %point 1
         1  1  0   %point 2
        -1  1  0   %point 3
        -1 -1  0   %point 4
         0  0  2   %point 5
       ];
% homogeneous version for
% transformations
hpoints=[points';ones(1,size(points,1))];
```

```
%definition of faces
Faces1 = [
            1 2 5 5   %face1
            2 3 5 5   %face2
            3 4 5 5   %face3
            4 1 5 5   %face4
            1 2 3 4   %face5
          ];
%simple color index to
%paint the faces
fColor= [ 1 2 3 4 5 ]';
```

Exercício 7 - Representação de objetos poliédricos

Representar a pirâmide e uma réplica

- Representar a pirâmide do exercício anterior com o comando:
  - `h=patch('Vertices', points, 'Faces', Faces1, 'FaceVertexCData', fColor, 'FaceColor', 'flat');`
- Criar e representar uma cópia da pirâmide anterior mas translacionada de 4 unidades em x em relação à original.



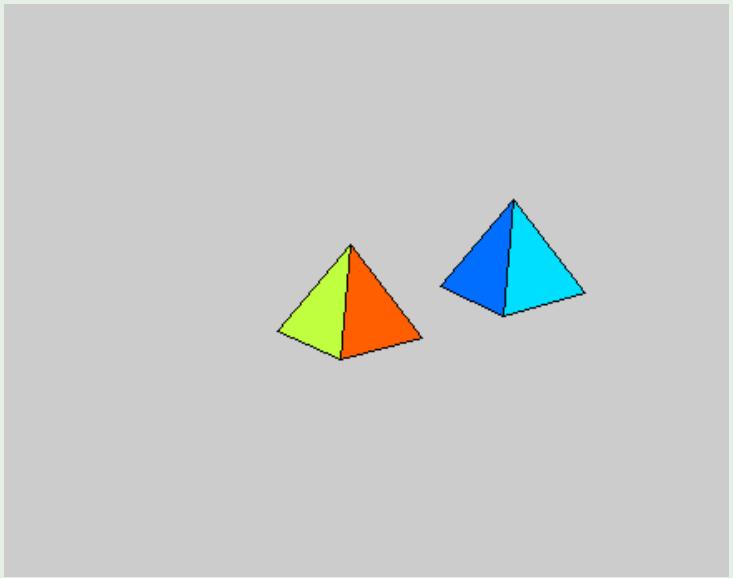
Tal como para polígonos, para criar cópias de poliedros não é necessário criar de raiz novos pontos ou faces. Por outro lado, as transformações geométricas são aplicadas só aos vértices.

Exercício 8 - Animação de poliedros

Animar o segundo poliedro em órbita em torno do primeiro

Invocando a função `manimate()`, a partir do programa principal, fazer a animação ilustrada no exercício 1. Propõe-se o uso de hipermatrizes de 4 dimensões para facilitar a indexação dos diversos passos no ciclo `for`.

Exemplo de animação a implementar



Componente de orientação e ângulos de Euler

Funções do Matlab `eul2tform()` e `tform2eul()`

- Funções disponiveis na robotics toolbox do MATLAB.
- Se não estiver instalada pode-se usar o cálculo direto (para RPY):
  - $T = eul2tform([\phi, \theta, \psi]) = rotz(\phi) \times roty(\theta) \times rotx(\psi)$
  - $[e_z, e_y, e_x] = [\phi, \theta, \psi] = tform2eul(T)$
    - com $T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$ virá: $\begin{cases} \phi = \arctan(r_{21}, r_{11}) \\ \theta = \arctan(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}) \\ \psi = \arctan(r_{32}, r_{33}) \end{cases}$

`atan()` vs. `atan2()`

Em MATLAB, no cálculo anterior, deve-se usar da função `atan2()` para realizar as operações com arctan().

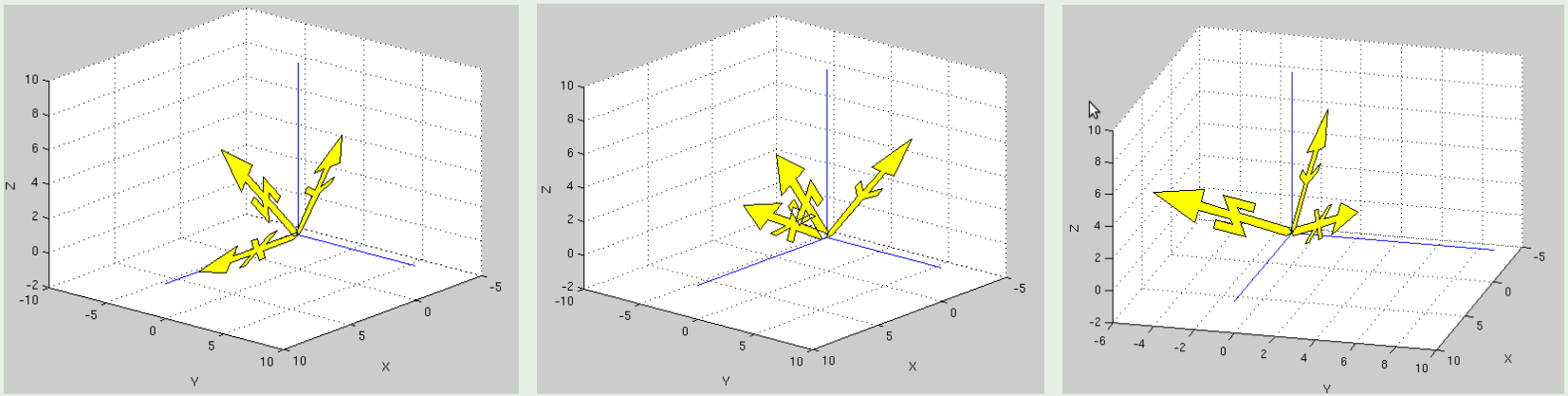
Nestas funções, a componente de translação é irrelevante: em `eul2tform()` resulta em 3 valores nulos nas componentes de translação, e em `tform2eul()` a coluna dos 3 valores da translação é simplesmente ignorada.

Exercício 9 - Operações com ângulos de Euler

Representar um objeto orientado pelos ângulos de Euler

- Usar a função `eul2tform()` para ilustrar o objeto devolvido pela função `seixos3()` com as orientações $[\phi, \theta, \psi] = [45^\circ, -30^\circ, 60^\circ]$, mas representando em três etapas com 3 gráficos separados como ilustrado: $[0^\circ, 0^\circ, 60^\circ]$, $[0^\circ, -30^\circ, 60^\circ]$ e $[45^\circ, -30^\circ, 60^\circ]$

Decomposição da orientaçao final em 3 etapas



- Usando a função `tform2eul()` verificar que a matriz final dada por $T = eul2tform([45^\circ, 0^\circ, 0^\circ])eul2tform([0^\circ, -30^\circ, 0^\circ])eul2tform([0^\circ, 0^\circ, 60^\circ])$ corresponde aos ângulos impostos para a orientação ($[45^\circ, -30^\circ, 60^\circ]$).