Robótica Industrial
Aula prática nº 1

Revisões de Matlab
Representação de objetos
Transformações geométricas elementares

Vitor Santos

Universidade de Aveiro

26 Set 2022

Recomendações para realizar os exercícios

- Criar uma pasta por aula (Aula1, Aula2, etc.)
- Criar uma pasta para colocar funções e scripts de MATLAB utilizáveis nas diversas aulas e trabalhos (por ex. `lib`) e esta pasta deverá ficar no *path* do MATLAB.
- Há duas metodologias principais para resolver os exercícios em cada aula:
  - Cada exercício está num ficheiro diferente:
    - `a1ex1.m` , `a1ex2.m`, etc.
  - Vários exercícios no mesmo ficheiro:
    - aula1.m, etc.
- Os dois métodos têm vantagens e desvantagens.

Caso dos exercícios num único ficheiro

- Na opção de usar todos os exercícios de uma aula no mesmo ficheiro, os exercícios devem ficar em secções distintas, demarcadas com comentários duplos, como no seguinte exemplo com dois exercícios:

```matlab
%% Ex1
Z=zeros(100,200);
imshow(Z)

%% Ex2
X=ones(100,200);
imshow(X)

```

- As secções podem ser executadas individualmente (CTRL+ENTER).
  - NB. Pode haver o risco de interferência de variáveis entre as diversas secções. Se não for para usar variáveis de umas secções para outras, recomenda-se um comando "clear" no início da secção para as limpar.
- Em muitas situações será preciso criar funções. Nesse caso, as funções deverão ficar cada uma num ficheiro separado, na pasta de trabalho da aula ou, no caso geral, na pasta **lib** criada atrás.
  - **N.B.** É possivel criar funções com nomes iguais a funções já existentes no MATLAB. Será executada a primeira que surgir no path. Este é um ponto a ter em atenção porque pode gerar resultados inesperados!

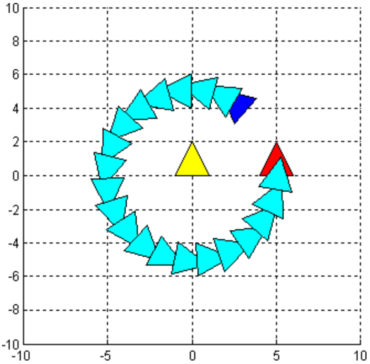
Exercício 1

Fazer um script onde:

- Se represente um triângulo $A1$ com os vértices:
  - $P1 = [-1 \quad 0]^\intercal$
  - $P2 = [1 \quad 0]^\intercal$
  - $P3 = [0 \quad 2]^\intercal$
  - $A1 = [P1 \quad P2 \quad P3]$
- Usar a função `fill` do MATLAB
- Criar e representar um novo triângulo $A2$ por translação de $A1$ com o vetor v: $A2 = A1 + v$
  - $v = [5 \quad 0]^\intercal$
- Usar uma janela entre -10 e $+10$ nos dois eixos

Exercício 2

- Criar o triângulo $A3$ por rotação de $A2$ de um ângulo de 50°.
  - Usar uma matriz de transformação geométrica apropriada (2×2) para obter $A3$. $A3 = Rot(50) \times A2$
- Gerar uma sequência de triângulos obtidos pela rotação sucessiva de $A2$ em N valores incrementais a começar em 60 e a terminar em 350. (Experimentar com $N = 10, 20, 50$).
- Modificar o programa de forma a fazer uma animação suave usando o comando `set` do MATLAB em vez de desenhar os triângulos todos.



Exercício 3

- Usando a representação **homogénea** de pontos, criar uma função de transformação geométrica que aceita três parâmetros e retorna a matriz de transformação M correspondendo a uma rotação alpha seguida de uma translação (x,y):
  - `M=TransGeom(x,y,alpha)`
- Invocando esta função, escrever um *script* que faça a animação do triângulo $A1$ na seguinte sequência:
  - Translação com o vetor $u_1 = [3 \quad 4]^T$
  - Rotação de 80° em torno da origem
  - Translação com o vetor $u_2 = [2 \quad -5]^T$
  - (todas estas etapas da animação devem ter 50 passos)

  Sugestões/Notas: fazer ciclos "for" para cada um dos 3 movimentos. Criar um vetor auxiliar de 50 valores entre 0 e 1 para facilitar a implementação dos passos da animação. Recordar que cada um dos 3 movimentos começa **após** a conclusão acumulada dos anteriores.

Exercício 4a - Opcional

Criar a função `AnimateSimple2D(h, P, M, N)`

Para fazer uma sequência de animações do objeto P com *handle* `h`

As transformações geométricas de cada etapa estão inseridas na matriz M, e N é o número de passos de cada etapa de animação.

- Cada coluna j de M tem o formato: $M_j = [t_x \quad t_y \quad r_\alpha]^\mathsf{T}$ e os valores reportam-se à origem do sistema de coordenadas.
- A cada elemento M_j corresponde a transformação: $\begin{bmatrix} \cos r_\alpha & -\sin r_\alpha & t_x \\ \sin r_\alpha & \cos r_\alpha & t_y \\ 0 & 0 & 1 \end{bmatrix}$

Sugestões

- Criar uma matriz MM com as interpolações entre todas as K colunas:

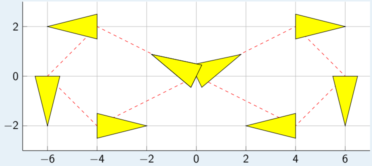
$$MM = \begin{bmatrix} \text{linspace}(t_{x_1}, t_{x_2}, N) & \text{linspace}(t_{x_2}, t_{x_3}, N) & \dots & \text{linspace}(t_{x_{K-1}}, t_{x_K}, N) \\ \text{linspace}(t_{y_1}, t_{y_2}, N) & \text{linspace}(t_{y_2}, t_{y_3}, N) & \dots & \text{linspace}(t_{y_{K-1}}, t_{y_K}, N) \\ \text{linspace}(r_{\alpha_1}, r_{\alpha_2}, N) & \text{linspace}(r_{\alpha_2}, r_{\alpha_3}, N) & \dots & \text{linspace}(r_{\alpha_{K-1}}, r_{\alpha_K}, N) \end{bmatrix}$$

- Fazer um ciclo "for" que usa a função `TransGeom()` definida no Ex. 3.

Exercício 4b - Opcional

Testar a função `AnimateSimple2D()` criando um script que:

- Desenha o objeto triangular com os vértices em $P1 = [0 \quad -1/2]^\intercal$ $P2 = [2 \quad 0]^\intercal$ e $P3 = [0 \quad 1/2]^\intercal$, e obter o *handle*.
- Faz um movimento aproximado ao longo das linhas ilustradas com os seguintes pontos de passagem em posição e orientação: (0,0,154°), (-4,2,180°), (-6,0,270°), (-4,-2,360°), etc.:



- Usa 50 pontos intermédios entre cada ponto de passagem.

Variante da função

Como se poderia adaptar/modificar a função para termos um número variável de iterações entre pontos de passagem?