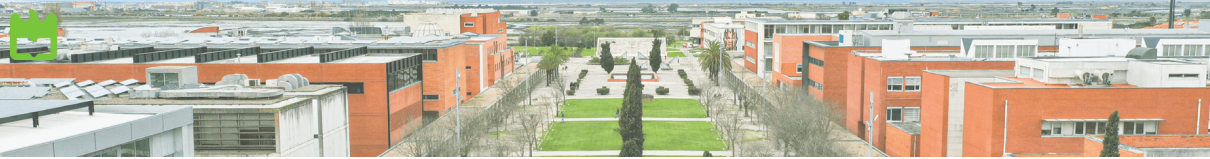


Robótica Espacial
Aula prática nº 11
Elementos de Visão Artificial e uma Aplicação na Deteção de Obstáculos

Vitor Santos

Universidade de Aveiro

15 maio 2025

Sumário

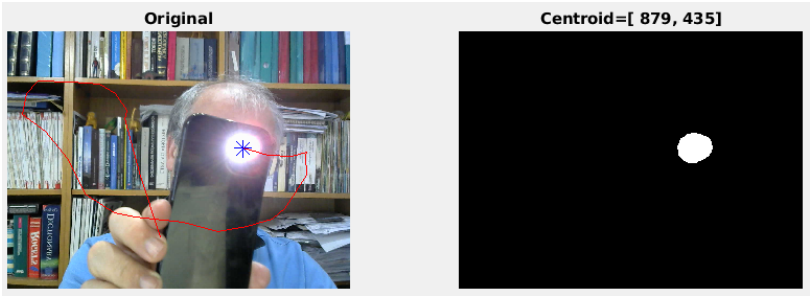
Preparação e configuração inicial

- Instalar as seguintes toolboxes em Matlab
  - Computer Vision Toolbox
  - Image Acquisition Toolbox
  - Image Processing Toolbox
- Adicionar os seguintes 'add-ons'
  - Image Acquisition Toolbox Support Package for OS Generic Video Interface
  - MATLAB Support Package for USB Webcams

Para fazer alguns dos exercícios é necessário ter uma webcam instalada: pode ser a integrada no computador, ou uma outra externa ligada, em geral, por USB.

Exercício 1 - Deteção e seguimento de um alvo na webcam

- O objetivo do exercício é o de mostrar em tempo real a deteção e seguimento de um alvo destacado numa imagem e de representar o seu caminho durante o movimento.
- Propõe-se seguir um ponto de luz intenso, mas o exercício pode ser adaptado para por exemplo seguir um objeto distintivo pela sua cor ou outra propriedade.
- A metodologia principal é a de ter duas imagens lado a lado: uma com a imagem que vem da câmara onde se marcará o movimento, e outra imagem a mostrar a segmentação do alvo a seguir.



Ex. 1 - Passos específicos - Parte 1

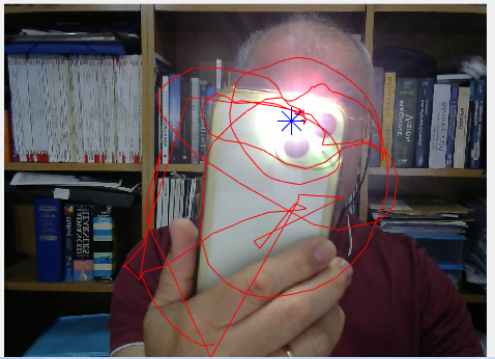
- Criar o dispositivo de acesso à camara (`webcam()`) e visualizar a imagem com `preview()` numa janela à parte. Essa janela pode ser fechada depois de se verificar que a aquisição de imagem está operacional.
- Criar um ciclo infinito (usando `while-end`) para adquirir e visualizar em contínuo uma imagem com `snapshot()` e `imshow()`. Usar também uma breve pausa dentro do ciclo, e.g. `pause(0.05)`.
- Criar num `subplot()` com duas imagens lado a lado com a imagem original e a imagem a níveis de cinzento convertida por `rgb2gray()` ou `im2gray()`.
- Passar um alvo muito brilhante (lanterna do smartphone ou similar) à frente da câmara e observar o movimento desse alvo, mas invertendo a imagem na horizontal para gerar um efeito de espelho da câmara usando o comando `fliplr()`.

Ex. 1 - Passos específicos - Parte 2

- Converter a imagem de cinzentos numa imagem binária onde só partes com brilho elevado ficam a branco (e.g. $> 95\%$) e todo o resto a preto. Usar o comando `imbinarize()`.
- Eliminar da imagem binarizada todos os objetos com dimensão menor do que um certo valor (por exemplo 5000 pixels, mas isso depende das condições e resolução da imagem em cada caso/câmara). Usar a função `bwareaopen()`.
- Obter certas propriedades dos objetos na imagem binarizada, em particular pretende-se o centróide, mas existem muitas outras. Usar a função `regionprops()` e visualizar no título da imagem as coordenadas do **primeiro** centróide se ele existir (usar `title()`)
- Sobre a imagem original fazer o `plot()` de um ponto no centro da zona brilhante. Deve usar-se as coordenadas do centróide obtidas antes. Para não apagar a imagem, deve fazer-se `hold on` antes da operação de `plot()`.

Exercício 2 - Representar o caminho do objeto luminoso

- Complementar o exercício com a representação do caminho percorrido pelo alvo durante todo o movimento.
- Sugestão:
  - Criar variáveis auxiliares fora do ciclo para acumular as coordenadas dos sucessivos centróides.
  - Atualizar o caminho do objeto e representar com um comando como `plot()` sobre a imagem original.



Exercício 3 - Interação remota com um servidor de imagem

- Ligar o computador à rede WIFI com os seguintes elementos:

  SSID: LARLAN
  Password: larlan2025
  Security: WPA-PSK/WPA2-PSK (ou similar)

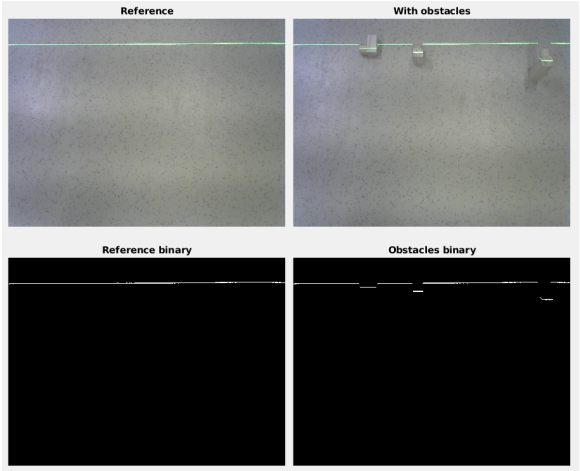
- Criar um programa para estabelecer uma ligação ao servidor remoto com o seguinte:

```
serverIP = '192.168.0.100'; % IP address of remote host
port = 9999;                % Port to establish the communication
t = tcpclient(serverIP, port, 'Timeout', 5);  % create the tcp/ip link
```

- Adquirir uma imagem com a função fornecida: `img = vsGetImageFromServer(t);`
- Visualizar a imagem com: `imshow(img)`
- Guardar essa imagem no ficheiro `imgref.jpeg` usando a função: `imwrite()`
- No fim do exercício recomenda-se fechar a ligação com: `clear t`

Exercício 4 - Imagem com linha laser irregular

Adaptando o exercício anterior obter uma nova imagem do servidor e fazer os passos seguintes:

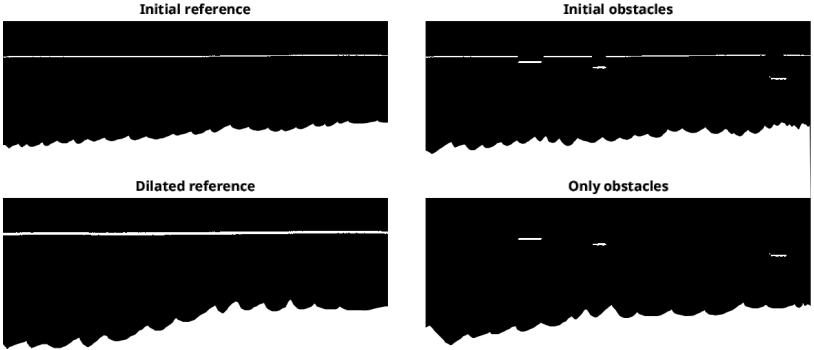


- Criar quatro janelas num `subplot`
- À esquerda representar a imagem do ficheiro `imgref.jpeg` obtida antes (e agora lida com `imread()`) e a sua variante binarizada
- À direita fazer o mesmo para a imagem adquirida do servidor de imagens.
- As versões **binarizadas** serão as usadas nos exercícios seguintes

N.B. - Se houver dificuldades no acesso ao servidor usar as imagens de defeito fornecidas.

Exercício 5 - Separação dos obstáculos na linha laser

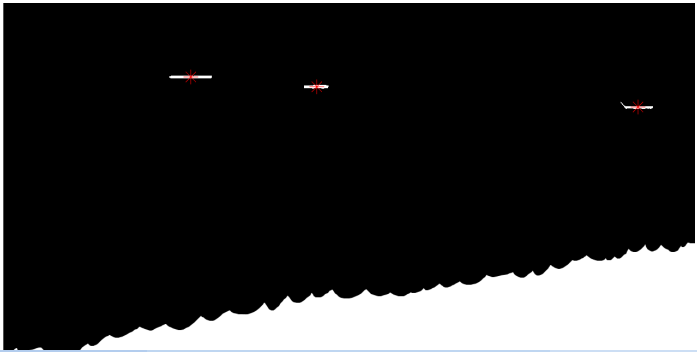
- Na imagem binarizada de referência (linha sem obstáculos) eliminar pequenos objetos (menores que 30 pixels) usando a função `bwareaopen()`
- Dilatar a imagem resultante usando `imdilate()` para a alargar 2 ou 3 pixels na vertical.
- Usar a imagem diladata para eliminar as zonas que não são obstáculos na outra imagem.
  - Usar indexação de imagem com outra imagem e.g., `obstB(refBdil) = 0;`
  - Limpar da imagem resultante pequenos objetos que possam ter ficado (< 10 pixels)



Exercício 6 - Localizar os obstáculos e os seus centros

O objetivo é obter as coordenadas dos centros de todos os obstáculos para depois procurar a zona com mais espaço entre eles para por exemplo levar um robô nessa direção.

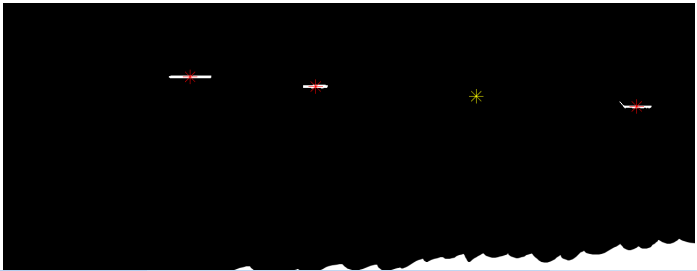
- Obter o número de objetos e a matriz de "labels"com `[L,N]=bwlabel(obsB);`
- Obter os centróides desses objetos: `r=regionprops(L,'Centroid')`
  - Concatenar os centróides todos numa matriz: `cc=cat(1,r.Centroid)'`
- Assinalar os centróides na imagem dos obstáculos (usar `plot()` depois de um `hold on`)



Exercício 7 - Determinar o ponto de passagem entre obstáculos

Assumido que os obstáculos são enumerados da esquerda para a direita, determinar o par de obstáculos sucessivos com os centros mais afastados entre si

- Fazer um ciclo para calcular as `N-1` distâncias entre pares sucessivos de centróides e guardar os resultados num vetor `DD`.
- Procurar em DD o índice da maior distância: `idx=find(DD==max(DD));`
- Determinar o ponto médio entre os dois centróides mais afastados: `dest=(cc(:,***)+cc(:,***))/2`
- Assinalar esse ponto na imagem com `plot()` com um marcador amarelo.



Exercício 8 - Integração numa aplicação em tempo real

Integrar os exercícios anteriores numa aplicação que lê continuamente (ciclo infinito) do servidor imagens novas, que podem estar a variar, e calcular a cada instante o destino mais seguro para um robô que parte do centro da linha de baixo da imagem em direção ao destino.

- O robô pode ser um simples ponto que inicia o movimento no meio da linha de baixo da imagem
- Simular um movimento do ponto corrente (*Curr*) em direção ao ponto de passagem (*Dest*) com uma velocidade sugerida de 5 pixels por *frame*, mas que pode ser ajustada para verificar outras situações:
  - $\Delta r = (\Delta x, \Delta y) = 5 \times \frac{Dest - Curr}{||Dest - Curr||}$