# Sistemas de Visão e Percepção Industrial

## Aula Prática nº 6

Deteção de Arestas.

Universidade de Aveiro
Vitor Santos, 2022-2023

# Sumário

# Filtros para detetar “Arestas” em Matlab

A deteção de "arestas" (ou bordos) é feita pela aplicação de filtros sobre uma imagem. Em Matlab essa operação poder ser feita de várias formas recorrendo a diversas funções, das quais se destacam as seguintes:

- `filter2()`
  - Função genérica de aplicação de um filtro.
  - Nenhum processamento adicional é feito sobre o resultado.
- `edge()`
  - Função que aplica filtros especiais para deteção de "arestas" e dá uma imagem binária. Faz a operação toda (gradientes, absolutos, binarização). Pode aceitar parâmetros para o limiar de decisão.
- `imfilter()`
  - Função mais geral de filtro. Permite correlação [como o `filter2()`] mas também convolução. Permite matrizes multidimensionais (como as imagens a cores). Não é fundamental para estes exercícios!

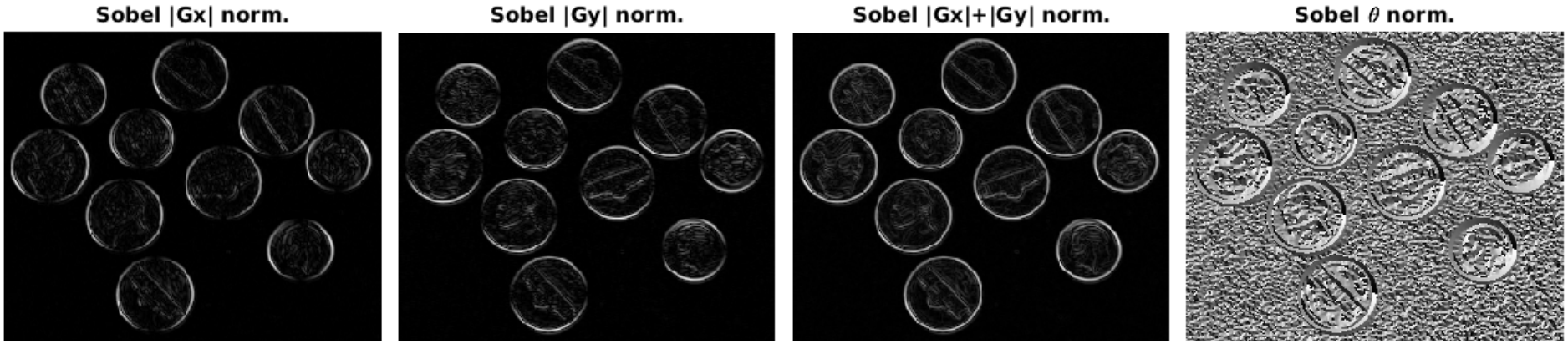
# Exercício 1)

- Carregar a imagem 'coins.png' (e usar `im2double()`)
- Aplicar os filtros de Sobel (Sx e Sy) usando a função `filter2()`.
  $Sx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$, $Sy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$.
- Usando `imshow()`, representar os gradientes absolutos, |Gx| e |Gy|, e o valor total: G=|Gx|+|Gy|

|Gx| |Gy| |Gx|+|Gy|

# Exercício 2)

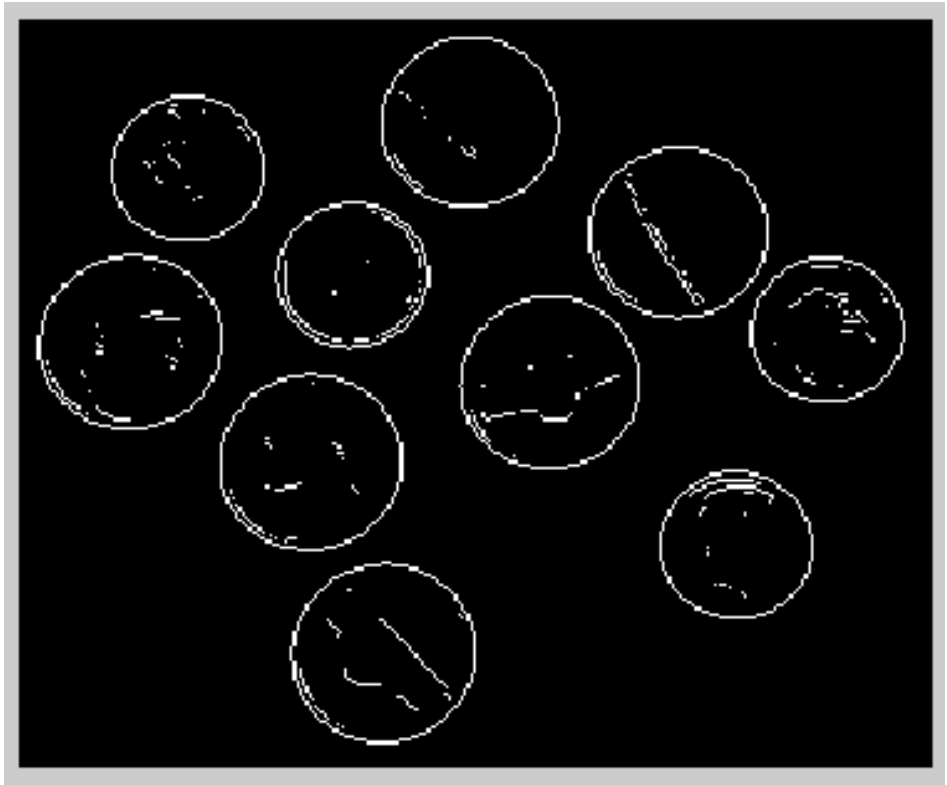
- Obter os mesmos resultados do exercício anterior usando em alternativa as funções `imgradientxy()` e `imgradient()`.
- Obter também a orientação $\theta$ dos gradientes.
  - Representar todos os resultados de forma normalizada [0;1] usando um segundo parâmetro no comando de visualização de uma imagem X: `imshow(X,[ ])`




- Comandos úteis:
  - `[Gx,Gy]=imgradientxy(A,'sobel');`
  - `[Gmag,Gdir]=imgradient(Gx,Gy);`
    - ou `[Gmag,Gdir]=imgradient(A,'sobel');`

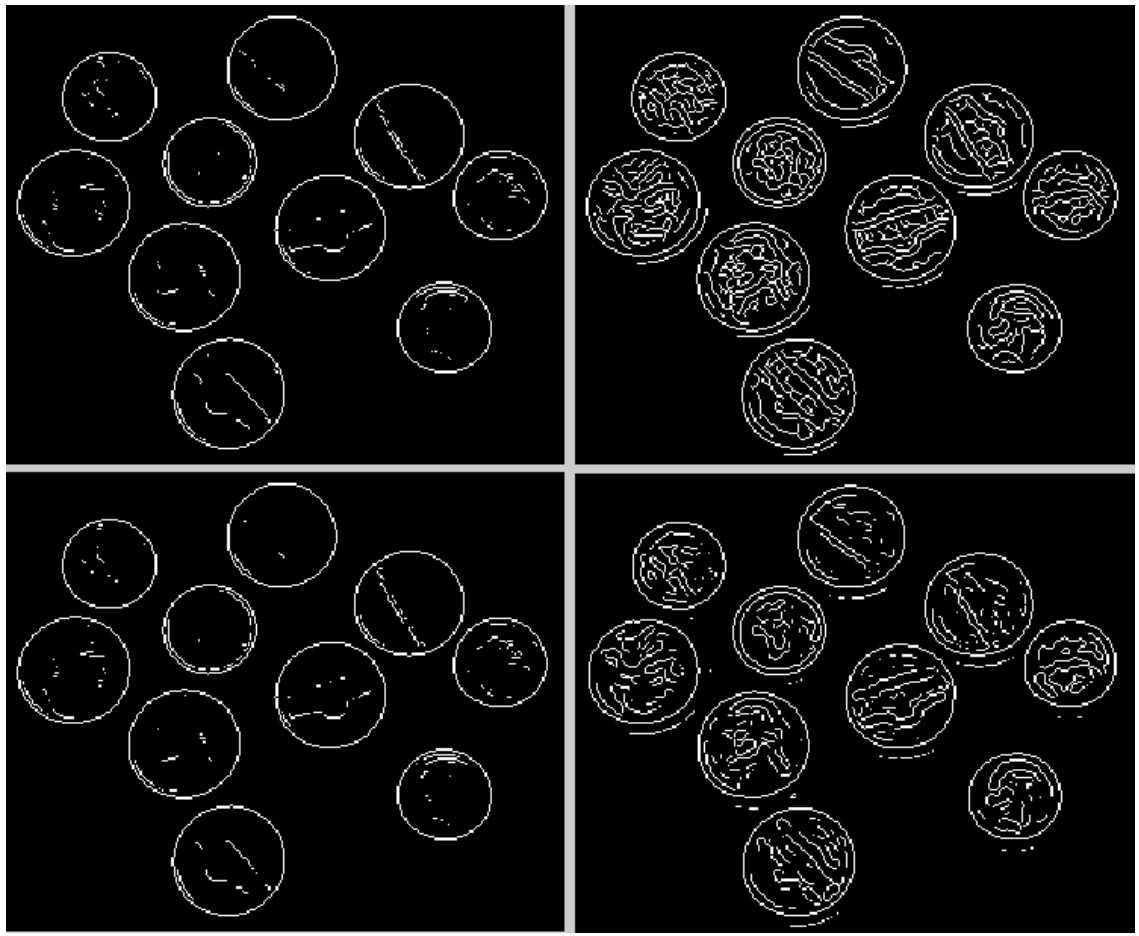
# Exercício 3)

- Para obter as "arestas" deve-se binarizar a imagem de gradientes. Em Matlab, faz-se o procedimento todo (gradiente total + binarização) com o operador `edge()`
- Ou seja, da imagem de cinzentos A, obtém–se as "arestas" com:
  - `B=edge(A, 'sobel');`



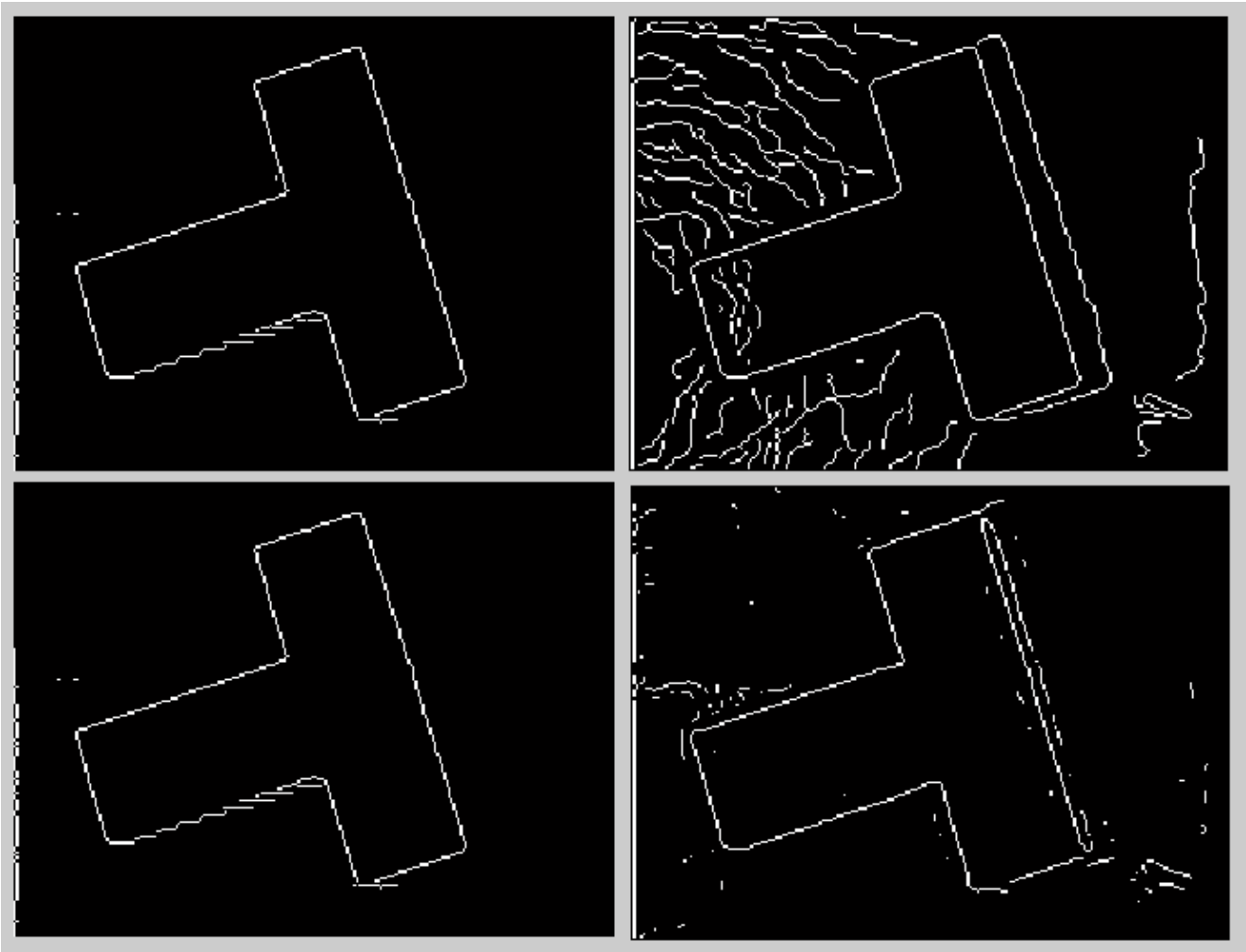
# Exercício 4)

- Representar as "arestas" da mesma imagem 'coins.png' com os seguintes filtros:
  - Sobel, Canny, Prewitt, Log



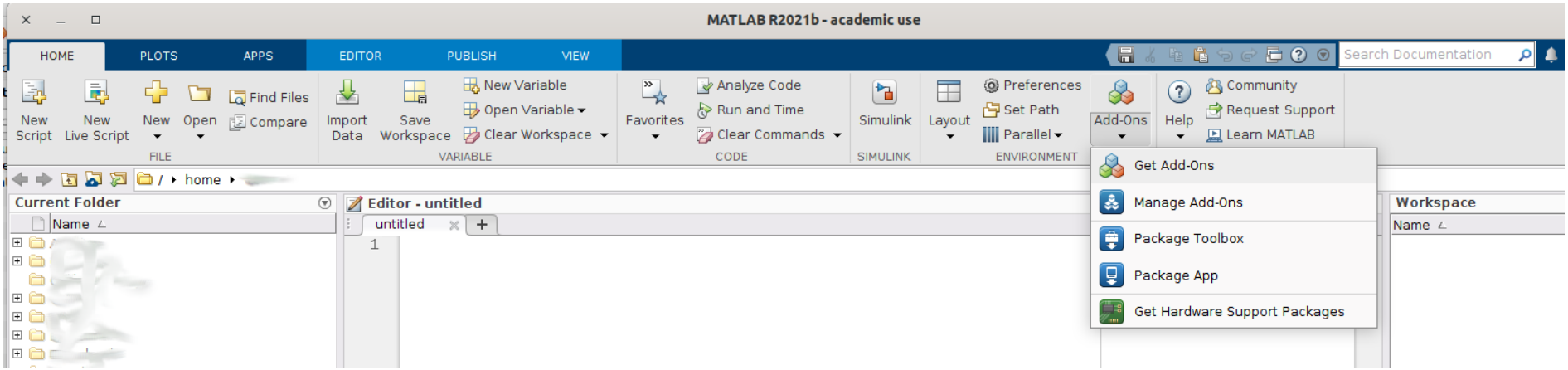
# Exercício 5)

- Repetir o exercício anterior para a imagem 'Tcomluz.jpg'.



# Instalação de uma Webcam

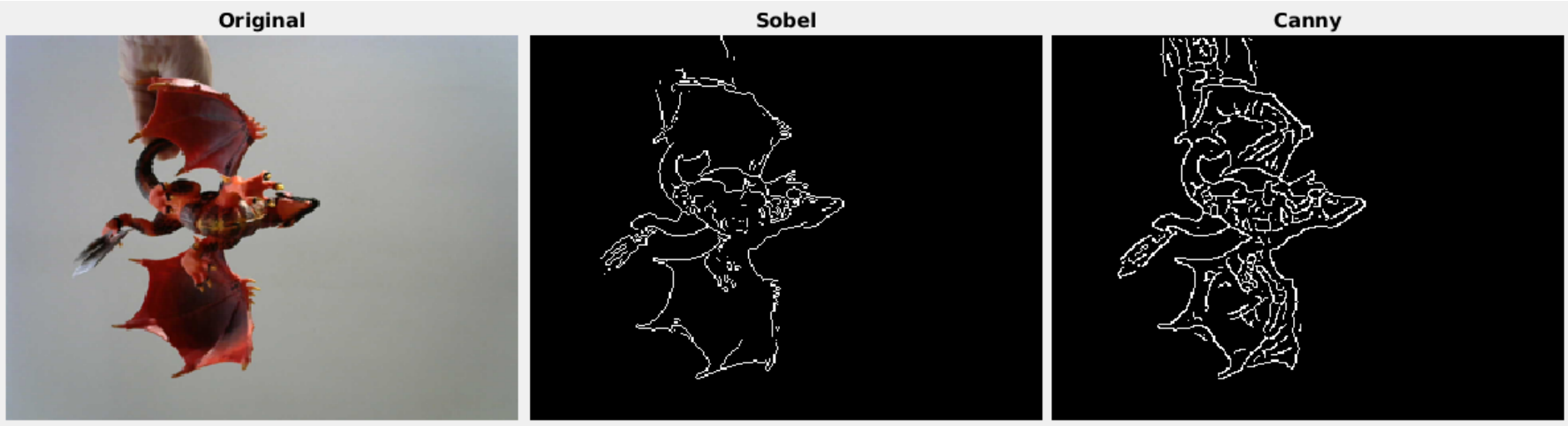
- Para testar os algoritmos de deteção de arestas em ambientes mais realistas propõe-se usar imagens diretamente de uma webcam que esteja instalada no computador
- Assim, além da toolbox de "Image Acquisition Toolbox" é necessário ter instalado os seguintes *add-ons* do Matlab:
  - Image Acquisition Toolbox Support Package for OS Generic Video Interface
  - MATLAB Support Package for USB Webcams
- O acesso aos add-ons pode ser encontrado nos menus ou na barra de ferramentas, como ilustrado na seguinte imagem:



# Exercício 6) Deteção de arestas em tempo real

- Criar um programa que abre o device da webcam e adquire a imagem em tempo real, a converte para níveis de cinzento e mostra as arestas detetadas por dois algoritmos diferentes, nomeadamente "Sobel" e "Canny".
- Exemplo de janelas a vizualisar em tempo real

```
close all
cam=webcam();
h=figure();
while isvalid(h)
    A=snapshot(cam);
    % complete the missing code
    pause(0.05)
end
clear cam % to close the device
```



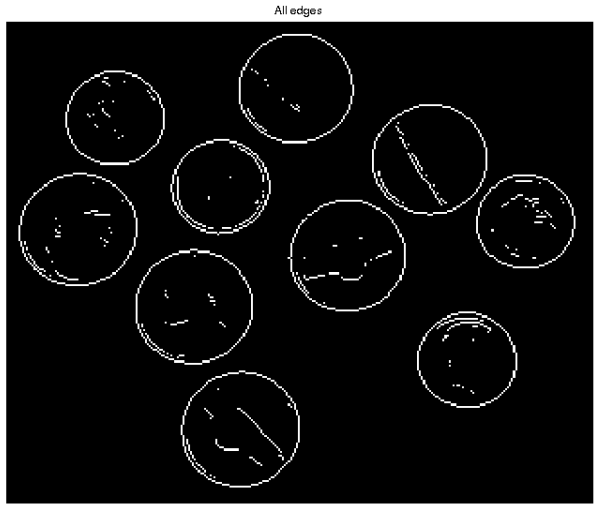
# Separar “arestas” individuais

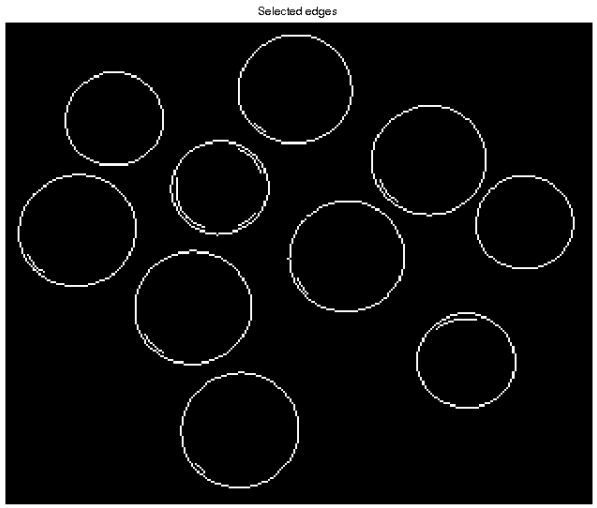
- Princípio
  - Obtidas as “arestas”, é necessário isolá-las ou distingui-las umas das outras para separar objetos. A abordagem pode ser feita de duas formas usando a conectividade ...
    - ... de regiões (função `bwlabel`);
    - ... de contornos (função `bwboundaries`);
- Função `[L,N]=bwlabel(BW)`
  - Devolve uma matriz L de números inteiros em que os pontos de cada região têm o mesmo valor (entre 1 e N). Os pontos do fundo são 0.
- Função `[B]=bwboundaries(BW,'noholes')`
  - Devolve uma lista `{cell array}` de conjuntos de coordenadas para os pontos dos diversos contornos da imagem BW. A opção 'noholes' impede que devolva contornos interiores de outros contornos fechados, devolvendo, portanto, apenas os contornos ditos de "hierarquia 1" da imagem BW.
    - NB. Se a imagem BW for uma região única (contígua), então B só terá um único contorno (um único conjunto de coordenadas) e obtém-se com: `B{1}`

# Exercício 7)

- Numa imagem de "arestas" de 'coins.png' (por Sobel), e usando bwlabel, separar as "arestas" que tenham 100 ou mais pixels, e colocá-as numa nova imagem.

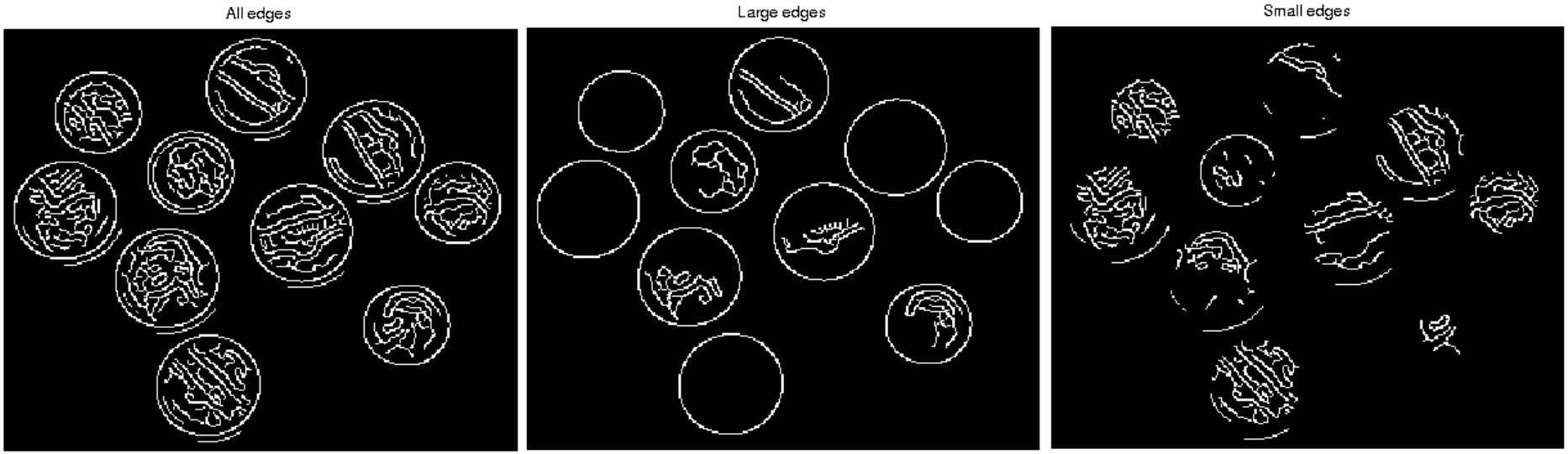
```
Z=edge(A,'Sobel');
X=false(size(Z)); %explain!
%window with all edges
subplot(1,2,1), imshow(Z);
title('All edges');
%window for the selected edges. Empty at the begin.
subplot(1,2,2), imshow(X), hold on;
title('Selected edges');

minSize=100; % minimum size of individual edges

[L, N]=bwlabel(Z); %explain!

for k=1:N
    C = (L==k);   %explain!
    if ( sum(sum(C)) < minSize ), continue; end %explain
    X = X | C;
    subplot(1,2,2), imshow(X)
    pause(0.2)   %why?
end
```



# Exercício 8)

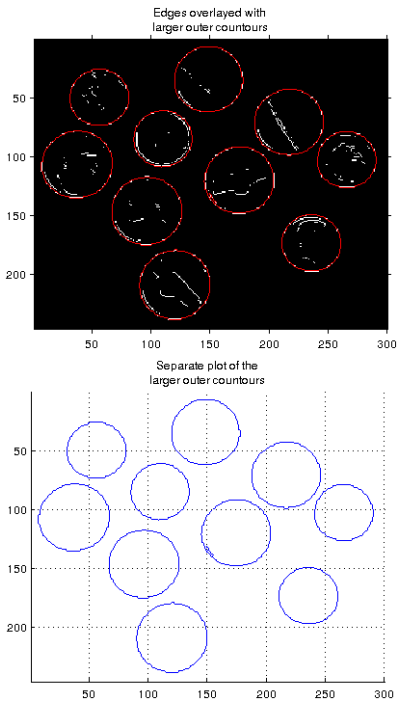
- Repetir o exercício anterior, mas usando filtro de Canny para detetar as "arestas", e criando duas imagens separadas: uma para "arestas" com 160 ou mais pontos, e outra para "arestas" com menos de 160 pontos. Continuar a usar bwlabel para determinar a dimensão das "arestas".



# Exercício 9) Opcional

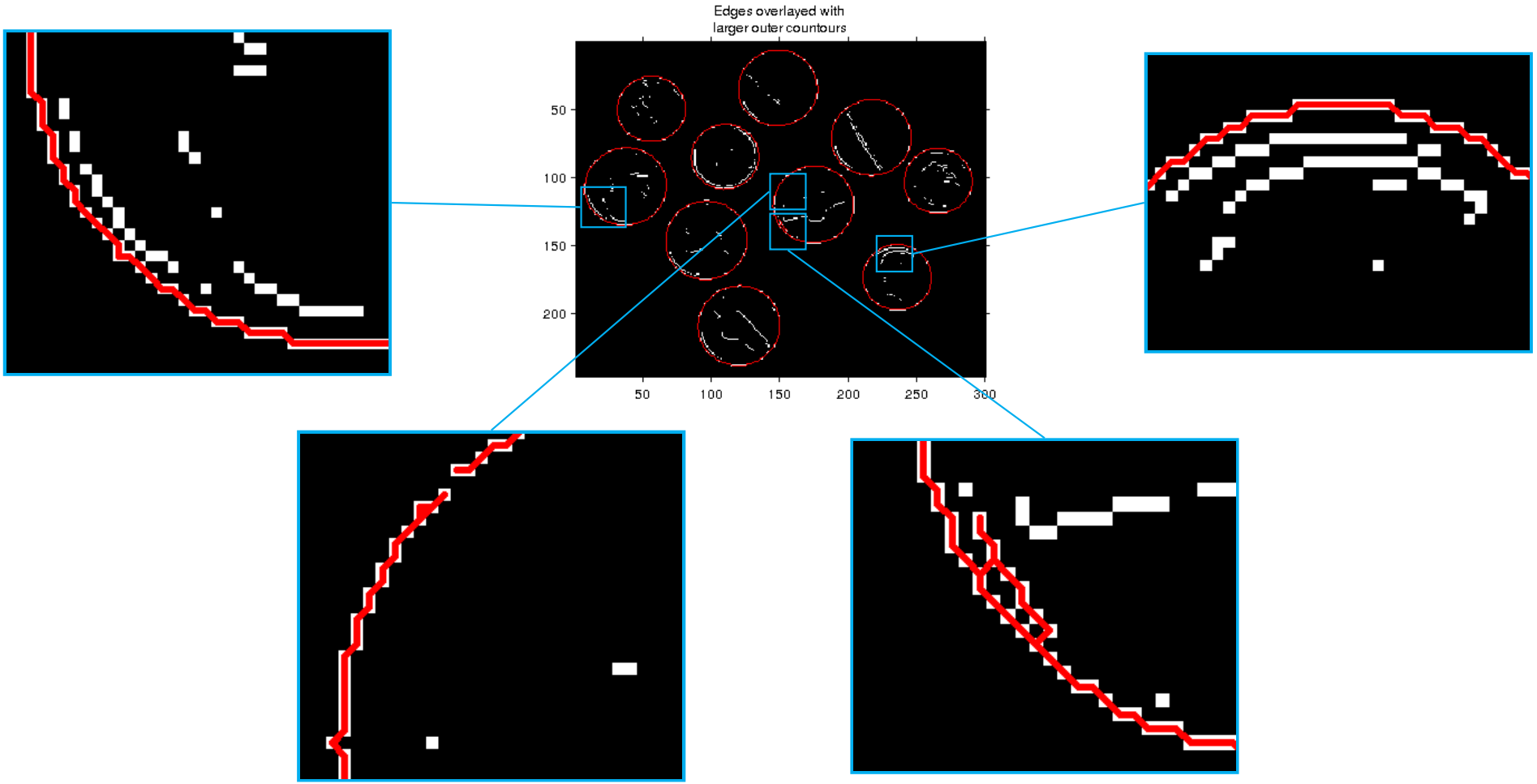
- A partir das "arestas" de 'coins.png' (por Sobel), obter as **coordenadas** dos pixels dos contornos e representá-las com `plot()` sobrepostas na imagem original das arestas, e também sobre um gráfico novo.
- Como precisamos das coordenadas dos pixels das arestas (contornos), o comando `bwlabel` não serve; usar o comando `bwboundaries`.

```
Z=edge(A,'Sobel');
X=false(size(Z));
%Create image with overlayed edges
subplot(1,2,1), imshow(Z); hold on;  axis on;
title({'Edges overlayed with', 'larger outer countours'});
%Create a plot without the image. Check the axis instructions...
myAxis=axis;
subplot(1,2,2), hold on, axis ij, axis equal, axis(myAxis), grid on
title({'Separate plot of the','larger outer countours'});
minSize=100;
[L, N]=bwlabel(Z);
for k=1:N
    C = (L==k);
    if ( nnz(C) < minSize ), continue; end

    BB = bwboundaries(C, 'noholes'); %skip inner boundaries
    boundary = BB{1}; %extract the first (outtermost) boundary

    %overlay boundary points in image
    subplot(1,2,1)
    plot(boundary(:,2), boundary(:,1), 'r','LineWidth',4);
    %create a new plot with lines
    subplot(1,2,2)
    plot(boundary(:,2), boundary(:,1), 'b');

    pause(0.5)
end
```



# Sobre os contornos

Os "contornos" ("contours") são uma representação de nível superior obtida a partir de imagens binárias (imagens de "arestas"). A sua estrutura pode ser complexa, podendo incluir hierarquias de contornos dentro de outros contornos. Do ponto de vista prático, são conjuntos ordenados de coordenadas de pixels. Abaixo representam-se detalhes em escala aumentada de algumas situações presentes no exercício anterior onde só se usam os contornos de nível 1 da hierarquia. Os pixels estão a branco e os contornos de nível 1 estão a vermelho.



# Exercício 10) Opcional

- Determinar e sobrepor na imagem 'Tcomluz.jpg' a sua maior "aresta" detetada por Canny, e usando os parâmetros de defeito do Matlab. Representar todas as N arestas numa imagem auxiliar, mas as as N-1 arestas menores devem ter a cor verde.
- Procurar encontrar os parâmetros, th1, th2 e $\sigma$ (do filtro gaussiano) para se obterem resultados similares aos ilustrados nas imagens da parte inferior da figura. NB: O valor de $\sigma$ afeta a dimensão do filtro gaussiano aplicado na imagem (o valor de defeito é $\sqrt{2}$.)