Universidade do Minho
Escola de Engenharia
Mestrado Integrado em Engenharia Informática

Unidade Curricular de Visão por Computador

Ano Letivo de 2019/2020

Tutorial 2 - Image Recognition

A78824 Mariana Lino Costa
A76387 André Ramalho

Janeiro, 2020

VC

Índice

Índice 2
Resumo 4
Introdução 5
Implementação do Código 6
Análise dos Resultados 15
Imagem coins1.jpg 15
Imagem coins2.jpg 17
Imagem coins3.jpg 20
Conclusão 22

Índice de Ilustrações

Figura 1 - imshow(background).....11
Figura 2 - imshow(I2).....12
Figura 3 - imshow(I3).....12
Figura 4 - imshow(bw).....13
Figura 5 - Imagem coins.jpg segmentada sem ruído.....15
Figura 6 - Imagem coins.jpg segmentada com ruído.....15
Figura 7 - SNR, TotalCircles e Types da imagem coins.jpg.....16
Figura 8 - Histograma com a distribuição do tamanho das moedas da imagem coins.jpg.....16
Figura 9 - Imagem coins2.jpg segmentada sem ruído.....17
Figura 10 - Imagem coins2.jpg segmentada com ruído.....17
Figura 11 - Deteção dos círculos na imagem binária (pré-processada).....18
Figura 12 - SNR, TotalCircles e Types da imagem coins2.jpg.....18
Figura 13 - Histograma com a distribuição do tamanho das moedas da imagem coins2.jpg. .19
Figura 14 - Imagem coins3.jpg segmentada sem ruído.....20
Figura 15 - Imagem coins3.jpg segmentada com ruído.....20
Figura 17 - Histograma com a distribuição do tamanho das moedas da imagem coins3.jpg. .21
Figura 16 - SNR, TotalCircles e Types da imagem coins3.jpg.....21

Resumo

Este relatório contém toda a explicação da implementação do código para a realização do programa solicitado, tanto como a análise dos resultados obtidos e a indicação de toda a dificuldade que se foi enfrentando ao longo da elaboração do problema.

Introdução

No âmbito da unidade curricular de Visão por Computador do Mestrado em Engenharia Informática na Universidade do Minho, foi proposto a elaboração de um programa que tem como principal objetivo receber uma imagem e detetar as moedas contidas, e ainda contar o número de moedas para cada tipo.

O tutorial proposto requer conhecimento adquirido nas aulas de visão computacional, com um foco no reconhecimento de imagens, incluindo a segmentação de imagens.

A segmentação de imagem é o processo de dividir uma imagem digital em múltiplas regiões (conjunto de pixels) ou objetos, com o objetivo de simplificar e/ou mudar a representação de uma imagem para facilitar a sua análise.

É de referir que detetar e contar objetos é um problema bastante comum na análise de imagens, e que por vezes a segmentação de uma imagem pode-se tornar difícil devido a sombras, desfocagem, reflexões ou até mesmo por os objetos estarem levemente sobrepostos na imagem.

Durante a elaboração deste projeto procurou-se arranjar métodos que solucionassem este problema de segmentação e também idealizou-se implementar um programa com a maior simplicidade possível e que fosse eficaz de forma a cumprir o objetivo principal supracitado.

Implementação do Código

Como já dito anteriormente, este trabalho foi focado no reconhecimento e segmentação de imagens e tem como objetivo identificar e reconhecer as moedas incluídas na imagem fornecida.

Começou-se na criação de um script image_recognition que chama a função principal *main_image_recognition,* passando lhe como parâmetros a imagem coins.jpg, coins2.jpg ou coins3.jpg (`I`), a imagem escolhida mas já a preto e branco (`IGray`), o tipo de ruído que o utilizador quer aplicar à sua imagem (`type_noise`) e o parâmetro de ruído (`noise_param`). No caso optou-se por aplicar o ruído *Gaussian* e aconselha-se a colocar o valor do `noise_param` não muito grande, pois o programa não funciona com imagens com um ruído muito elevado. Escolheu-se o valor 0.1 para a média do ruído gaussiano.

O utilizador deve então escolher uma destas três imagens, colocando as outras duas sempre em comentário, como está demonstrado na figura abaixo.

```
%Escolha uma das 3 imagens retirando-a de comentário e colocando/mantendo as outras como comentário

I = 'coins.jpg';
%I = 'coins2.jpg';
%I = 'coins3.jpg';

J= imread(I);

IGray = rgb2gray(J);
type_noise = 'G';
noise_param = 0.1;

[IS, ISN, SNR, PreI, Histogram, Types] = main_image_recognition(I,IGray,type_noise,noise_param);
```

O código foi dividido em várias funções também com o objetivo de simplificar a sua análise e de forma a organizar todo o programa.

Para também facilitar todo o código, procurou-se sempre que possível usar funções já pré-definidas no Matlab.

- main_image_recognition.m

A função *main_image_recognition* é a principal função deste programa. O programa foi feito baseado no *Circle Hough Transform*, uma técnica básica de recursos usada no processamento de imagens digitais para detetar círculos em imagens imperfeitas.

Esta função principal recebe uma imagem a preto e branco e devolve-a segmentada, com e sem ruído (variáveis de output da função `ISCR` e `ISSR`).

```
%IMAGEM A PRETO E BRANCO
 IS= I;
 figure(1);
 imshow(IS);
 imwrite(IS,'Imagem_preto_e_branco.png');

 %IMAGEM COM RUÍDO APLICADO
 ISN = addNoise(IS,type_noise,noise_param);
 figure(2);
 imshow(ISN);
 imwrite(ISN,'Imagem_Noise.png');
```

De forma a ser possível saber a degradação da imagem após aplicar o ruído, também é calculado e devolvido o valor do SNR (Relação sinal-ruído). O SNR é um conceito de telecomunicações, usado em diversos campos como ciência e engenharia que envolvem medidas de um sinal em meio ruidoso, é geralmente expressa em Decibel. Quanto maior o valor do SNR melhor, ou seja, menos ruído na imagem.

```
% CÁLCULO DO SNR (em que o x é o sinal da imagem do input e o y a estimativa do ruido).
x=double(IS);
y=double(ISN);
SNR=snr(x,y-x);
SNR
```

Além disso, esta função ainda devolve a imagem que foi processada. O pré-processamento da imagem tem como propósito facilitar o reconhecimento de círculos. A função ***pre_processing*** recebe a imagem já com o ruido aplicado (`ISN`) ou a imagem original (`IS`) e utiliza filtragens, modificação no contraste ou até mesmo métodos como o *subtracting tresholding* para se retirar o fundo da imagem de forma a se identificar melhor os objetos pretendidos. Estes métodos são aplicados dependendo da imagem. Existem imagens mais fáceis de se detetar os objetos como a coins e coins3, em que basta aplicar uma filtragem, e outras, como a coins2, que devido a sobreposições dos objetos, sombras e reflexos é necessário se fazer mais técnicas de processamento.

```
%PRÉ-PROCESSAMENTO (filtragem e equalização de contraste).
PreI= pre_processing(filename,ISN);
PreISR= pre_processing(strcat('sr_',filename),IS);
figure(3); imshow(PreI);
imwrite(PreI,'Imagem_Processada.png');

%DETEÇÃO DE BORDAS (Canny Edge Detector).
e = edge(PreI,'Canny');
eSR = edge(PreISR, 'Canny');
figure(4); imshow(e);
imwrite(e,'Imagem_Canny_Edge.png');
```

O passo a seguir é a deteção das bordas da imagem através do *Canny Edge Detector* (onde se usou a função **`edge`** do Matlab) para finalmente passar-se à deteção dos círculos através da função também pré-definida **`imfindcircles`**.

Houve a necessidade de se fazer a distinção de cada imagem recebida pois estas tinham características muito diferentes, o que levou à alteração dos parâmetros passados na função **`imfindcircles`** como:

1. os limites do raio dos círculos, estes valores vão variar consoante o tamanho das moedas na imagem.
2. *ObjectPolarity:* indica se os objetos circulares são mais brilhantes ou mais escuros que o plano de fundo. *Bright* se os objetos forem mais brilhantes que o fundo e *dark* se forem mais escuros.
3. *Sensitivity:* é um número no intervalo [0,1] que indica a sensibilidade para o conjunto de acumulados da transformada circular de *Hough*. À medida que se aumenta o fator de sensibilidade, os círculos detetam mais objetos circulares, incluindo círculos fracos e parcialmente obscuros.
4. ou até mesmo ainda acrescentar parâmetros como o *Method:* técnica usada para calcular a matriz do acumulador, *phasecode* é o método de codificação de fase e o TwoStage o método usado na transformação circular de *Hough* de dois estágios.

```
%DETEÇÃO DE CIRCULOS
% coins
if (strcmp(filename,'coins.jpg'))
 [centers, radii] = imfindcircles(e,[60 100],'ObjectPolarity','bright','Sensitivity',0.95);
 [centers_sr, radii_sr] = imfindcircles(eSR,[60 100],'ObjectPolarity','bright','Sensitivity',0.95);

% coins2
elseif(strcmp(filename,'coins2.jpg'))
 [centers, radii] = imfindcircles(e,[145 310],'ObjectPolarity','bright','Sensitivity',0.97,'Method','twostage');
 [centers_sr, radii_sr] = imfindcircles(eSR,[145 310],'ObjectPolarity','bright','Sensitivity',0.96,'Method','twostage');

% coins3
elseif(strcmp(filename,'coins3.jpg'))
 [centers, radii] = imfindcircles(e,[220 310],'ObjectPolarity','dark','Sensitivity',0.97);
 [centers_sr, radii_sr] = imfindcircles(eSR,[220 310],'ObjectPolarity','dark','Sensitivity',0.97);

%Para outras possiveis imagens
else
 [centers, radii] = imfindcircles(e,[60 100],'ObjectPolarity','bright','Sensitivity',0.95);
 [centers_sr, radii_sr] = imfindcircles(eSR,[60 100],'ObjectPolarity','bright','Sensitivity',0.95);

end
```

Para a coins2 houve a necessidade de se aumentar os limites do raio pois as moedas eram maiores, aumentar a sensibilidade ligeiramente em relação à imagem coins, e ainda se aplicar o método *twostage*, para se detetar os objetos que estavam levemente sobrepostos.

Contrariamente às imagens anteriores, a coins3 tem os objetos mais escuros que o seu fundo, por isso houve a modificação no parâmetro *ObjectPolarity* de *bright* para *dark*.

Após a deteção de todos os círculos, faz-se o desenho desses através da função **`viscircles`** do Matlab.

```
%CRIAÇÃO DOS CIRCULOS
%Imagem segmentada com ruído
figure(2);
ISCR = viscircles(centers, radii);
%Imagem segmentada sem ruído
figure(1);
ISSR =viscircles(centers_sr, radii_sr);

saveas(figure(2),sprintf('Imagem_segmentada_com_ruido.png'));
saveas(figure(1),sprintf('Imagem_segmentada_sem_ruido.png'));
```

A função *main_image_recognition* devolve ainda um histograma que mostra a distribuição do tipo de moedas consoante o tamanho, em que a medida do tamanho é o raio em pixéis, e uma tabela que indica os diferentes tipos de moedas na imagem e o número de moedas de cada tipo. Criou-se este programa de forma a também calcular o número total de moedas independentemente do seu tipo (variável `TotalCircles`).

```
%NÚMERO TOTAL DE CIRCULOS NA IMAGEM
TotalCircles = length(centers);
TotalCircles

radii

%NÚMERO DE MOEDAS DE CADA TIPO
Types = tiposMoeda(filename,radii);
Types
figure(4);

%HISTOGRAMA (com o número de moedas de cada tipo).
Histogram = histogram(radii);
title('Distribuição do Tamanho das Moedas', 'FontSize', 14);
xlabel('Raio (em pixeis)', 'FontSize', 11);
ylabel('Quantidade', 'FontSize', 11);
```

- addNoise.m

A função ***addNoise*** é a função responsável por aplicar o ruído à imagem recebida na função principal. Pode ser aplicado um ruído do tipo *Salt&Pepper* ou um do tipo *Gaussian*.

Optou-se por aplicar o ruído *Gaussian*, um ruído estatístico com uma função de densidade de probabilidade igual à da distribuição normal, também conhecida como distribuição gaussiana.

- pre_processing.m

No pré-processamento das imagens coins.jpg e coins3.jpg, começou-se por aplicar o filtro *Gaussian* no *spatial domain* com o intuito de remover o ruído, através da função do matlab `imgaussfilt`. No processamento de imagem digital, o ruido gaussiano pode ser reduzido com um filtro espacial. Optou-se por se escolher o filtro *Gaussian* pois a filtragem gaussiana é altamente eficaz na remoção do ruído gaussiano.

De seguida aplicou-se a equalização do histograma, permitindo melhor deteção das bordas, uma vez que algumas imagens têm pouco contraste, através da função `histeq`.

A filtragem na imagem *coins3.jpg* foi maior que na figura *coins.jpg*, como se pode verificar no código, os parâmetros passados na função `imgaussfilt` para a coins3 foram maiores que para a primeira imagem.

Uma vez que a imagem *coins2.jpg* foi mais complicada de segmentar, devido à quantidade de sombras, reflexos e mesmo à sobreposição das moedas, teve-se que usar um pré-processamento mais complexo:

1. Tal como nas outras imagens, aplicou-se o filtro *Gaussian* no spatial domain.
2. Removeu-se todo o primeiro plano (as moedas) usando abertura morfológica, através da função `strel`. A operação de abertura remove objetos pequenos que não podem conter completamente o elemento estruturador. Definiu-se o elemento estruturador em forma de disco com um raio de 15.

```
se=strel('disk',15);
background =imopen(I,se);
```

Figura 1 - imshow(background).

3. Para se realizar a abertura morfológica, usou se o operador morfológico `imopen` com o elemento estruturador.

4. Subtraiu-se uma aproximação da imagem de fundo à imagem original.

```
I2= I-background;
```

Figura 2 - imshow(I2).

5. Aumentou-se o contraste da imagem obtida.

```
I3 =imadjust(I2);
```

Figura 3 - imshow(I3).

6. Converteu-se para uma imagem binária e voltou-se a remover ruído de fundo.

```
bw =imbinarize(I3);
bw = bwareaopen(bw,50);

out=bw;
```

Figura 4 - imshow(bw).

- tiposMoeda.m

Na função *tiposMoeda* foi utilizado o tamanho do raio, em pixéis, dos círculos detetados para identificar o valor monetário das moedas reconhecidas na figura.

Para a *coins1* decidiu-se definir 2 tipos de moedas:

- 25 cent: a de raio entre 75 a 95;
- 1 cent: a de raio entre 55 e 75;

Para a *coins2* definiu-se também 2 tipos de moeda:

- 2 euros: a de raio entre 185 a 310;
- 1 euro: a de raio entre 145 e 185;

Para a *coins3* foi mais difícil se identificar o local e o valor da moeda, então optou-se por se considerar apenas 3 tipos de moeda:

- 50 cêntimos europeus: a de raio entre 270 e 300;
- 20 cêntimos europeus: a de raio entre 250 e 270;
- 10 cêntimos europeus: a de raio entre 200 e 250;

A estrutura utilizada foi uma tabela de 3 *arrays cell*, onde se guardou no primeiro *array* o tipo de moeda, no segundo a quantidade e no terceiro o intervalo de raios em pixéis.

```
tipos = table(tipo,quantidade,raio);
```

Análise dos Resultados

Imagem coins1.jpg

Apesar de a imagem ter uma luz não uniforme e de algumas moedas terem uma cor semelhante ao fundo, a mesa de madeira, conseguiu-se detetar facilmente todas as moedas corretamente e consequentemente calcular o número total de moedas e identificar cada tipo.

Figura 5 - Imagem coins.jpg segmentada sem ruído.

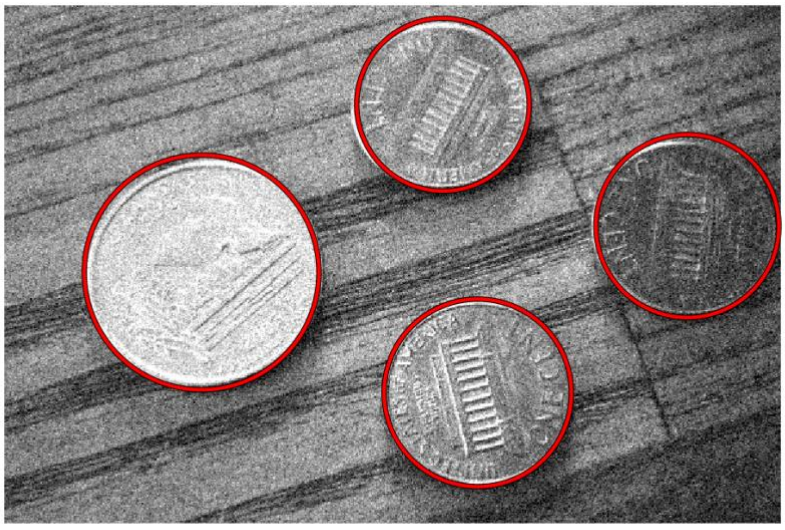

Figura 6 - Imagem coins.jpg segmentada com ruído.

O número total de moedas na imagem *coins,jpg* foi 4, em que três são de 1 cent e uma de 25 cent.

```
SNR =

   10.2819

TotalCircles =

     4
```

```
Types =

  2×3 table

      tipo          quantidade          raio
    ___________    ___________    ____________________

    {'1 cent' }      {[3]}        {'Entre 55 e 75 pixeis'}
    {'25 cent'}      {[1]}        {'Entre 75 e 95 pixeis'}
```

Figura 7 - SNR, *TotalCircles* e *Types* da imagem coins.jpg.

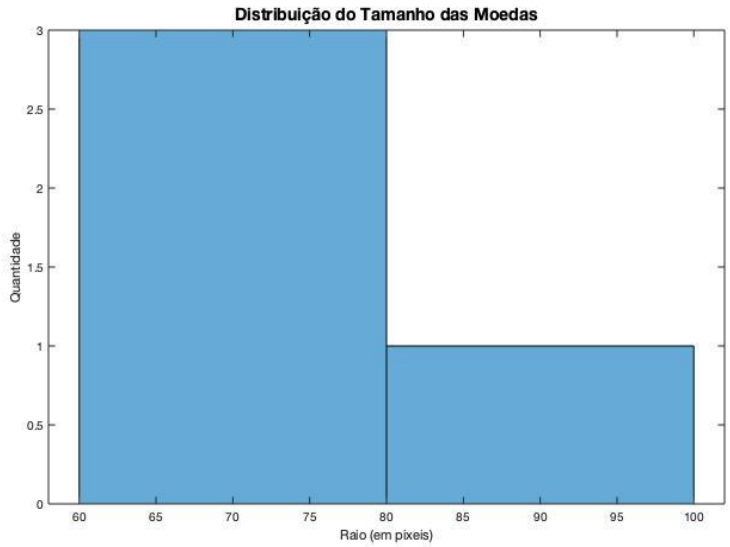

Figura 8 - Histograma com a distribuição do tamanho das moedas da imagem coins.jpg.

Imagem coins2.jpg

Esta foi a imagem onde houve uma maior dificuldade em identificar todos os círculos. As sombras, a inclinação da câmara e das moedas e a leve sobreposição entre elas foi o que mais complicou a sua deteção.

Apesar de toda a complicação, em todos os testes feitos foram reconhecidas sempre entre 21 a 24 (todas) moedas.

Figura 9 - Imagem coins2.jpg segmentada sem ruído.

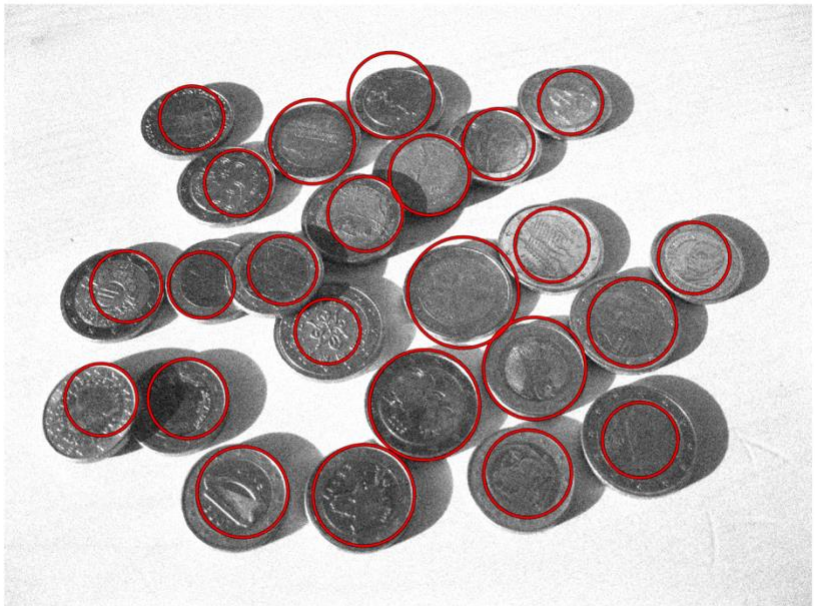

Figura 10 - Imagem coins2.jpg segmentada com ruído.

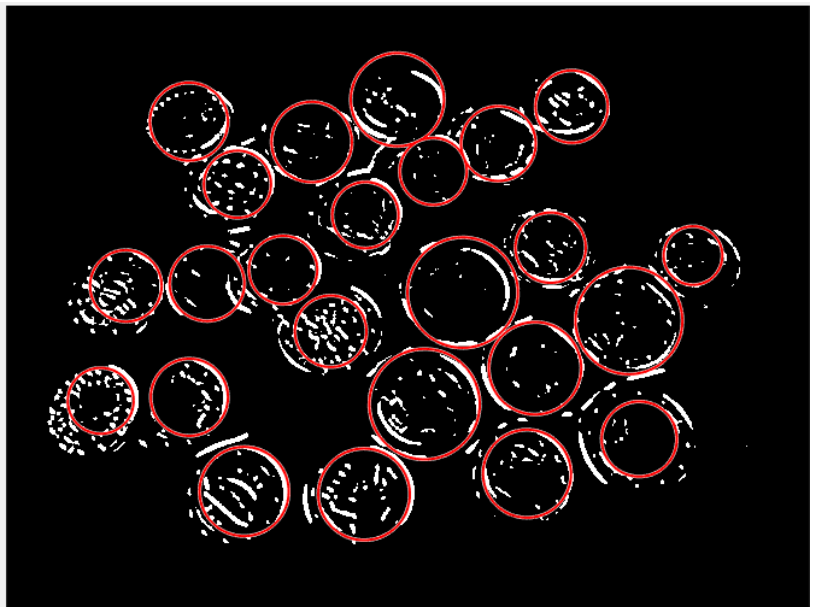

Figura 11 - Deteção dos círculos na imagem binária (pré-processada).

O número total de moedas na imagem *coins2.jpg* é 24, em que 9 são de 1 euro e 15 são de 2 euros. Infelizmente estes resultados variavam ligeiramente sempre que o programa era corrido, o que indica que não é perfeito e existem algumas limitações.

```
                          SNR =              TotalCircles =

Types =                   16.5561            24

  2×3 table

      tipo          quantidade                raio
    ___________     __________    ____________________________

    {'1 euro' }       {[ 9]}      {'Entre 145 e 185 pixeis'}
    {'2 euros'}       {[15]}      {'Entre 185 e 310 pixeis'}
```

Figura 12 - SNR, *TotalCircles* e *Types* da imagem coins2.jpg.

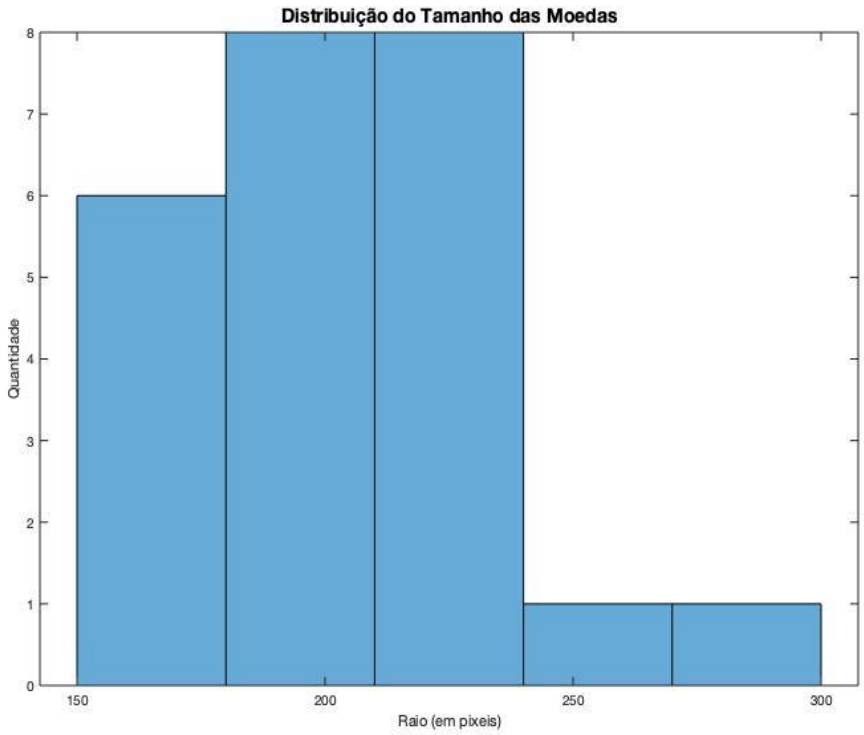

Figura 13 - Histograma com a distribuição do tamanho das moedas da imagem coins2.jpg.

Imagem coins3.jpg

Embora esta imagem esteja bastante desfocada, também se conseguiu identificar todas as moedas.

Figura 14 - Imagem coins3.jpg segmentada sem ruído.

Figura 15 - Imagem coins3.jpg segmentada com ruído.

O número total de moedas na imagem *coins3.jpg* é 7, em que duas foram consideradas de 10 cêntimos, três de 20 cêntimos e duas de 50 cêntimos. Como já foi referido anteriormente não se conseguiu identificar bem quais os valores de algumas das moedas da imagem, pois não são moedas utilizadas em Portugal. Então foi considerado que o tipo de moeda dependia do tamanho, caso fossem do tamanho da moeda de 10 cêntimos da imagem, 20 cêntimos ou 50 cêntimos, valiam também o mesmo valor e eram integradas nesse tipo.

```
SNR =

   13.3729
```

```
TotalCircles =

     7
```

```
Types =

  3×3 table

          tipo          quantidade              raio
    ________________    __________    ____________________________

    {'10 centimos'}       {[2]}       {'Entre 200 e 250 pixeis'}
    {'20 centimos'}       {[3]}       {'Entre 250 e 270 pixeis'}
    {'50 centimos'}       {[2]}       {'Entre 270 e 300 pixeis'}
```

Figura 17 - SNR, *TotalCircles* e *Types* da imagem coins3.jpg.

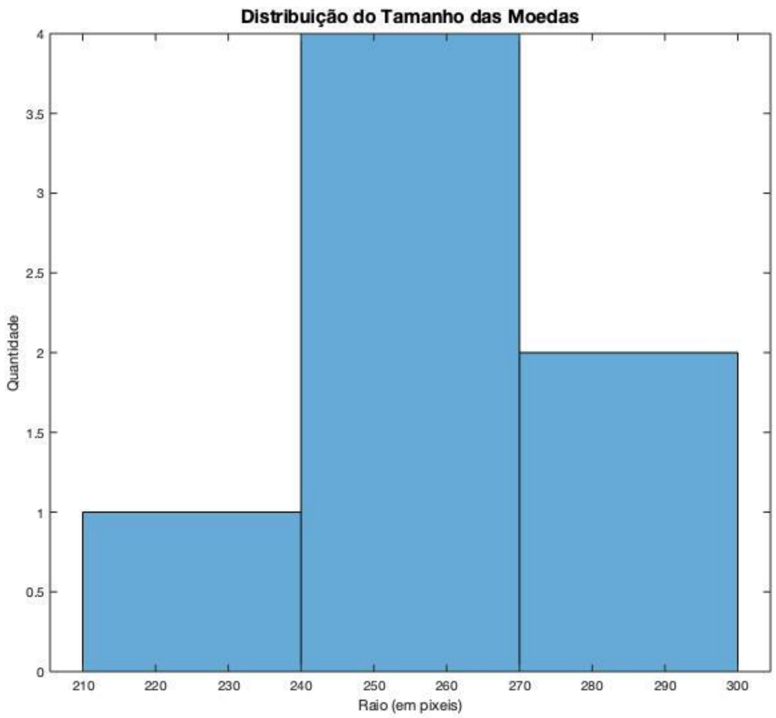

Figura 16 - Histograma com a distribuição do tamanho das moedas da imagem coins3.jpg.

Conclusão

Com o desenvolvimento de todo o trabalho houve um maior aprofundamento na aprendizagem da matéria lecionada nas aulas de Visão por Computador, principalmente à cerca de segmentação de imagens e deteção de objetos.

Apesar de todas as dificuldades encontradas, conseguiu-se um programa simples e capaz de corresponder a quase todos os objetivos do que era pretendido.

Como todos os programas computacionais, dificilmente se implementa algo perfeito, existe sempre algumas limitações e erros. O programa final provavelmente só vai funcionar para estas três imagens especificas (*coins*, *coins2* e *coins3*). Apesar de se ter tido o cuidado de colocar uma opção de segmentação mais geral para outras possíveis imagens, os parâmetros nunca vão estar perfeitos e adequados, pois cada imagem tem um contraste, luz, sombras e tipos de moedas diferentes. Para além que o programa fica ligeiramente mais lento com imagens mais pesadas.

Este método também deixa de funcionar se o parâmetro de ruído escolhido pelo utilizador for muito alto, ou seja, o ruído na imagem for muito elevado.

Todo o programa foi feito baseado no *Circle Hough Transform* e no caso da segunda imagem (*coins2.jpg*), existe por vezes uma falha e não se consegue detetar 1 ou 2 moedas. A inclinação das moedas na imagem faz com que elas não sejam círculos perfeitos e não sejam detetadas pelo detetor de círculos implementado. No entanto, maior parte das vezes as 24 moedas são detetadas.

Apesar de a contagem do tipo de moedas estar a ser feita corretamente nestes três exemplos, contar o valor da moeda pelo seu raio pode ser arriscado e levar a um erro. Por exemplo, uma moeda de 1 euro num plano de fundo da imagem não vai ter o mesmo raio que uma moeda de 1 euro em primeiro plano, pois estão a distâncias diferentes da lente da câmara.

Independentemente das limitações e erros que o programa poderia possuir, considerou-se que este projeto foi feito da melhor forma possível e aproximado ao que se pretendia, cumpriu todos os objetivos, detetar círculos e contar moedas pelo seu tipo.