



UNIVERSIDADE DO MINHO  
DEPARTAMENTO DE INFORMÁTICA  
VISÃO POR COMPUTADOR

---

## Tutorial 2

---

### Grupo 12

*Autores:*

Joel Gama (A82202)



Tiago Pinheiro (A82491)



3 de Janeiro de 2020

# **Conteúdo**

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b><i>Input</i></b>	<b>3</b>
<b>3</b>	<b>Resolução</b>	<b>4</b>
3.1	Algoritmo . . . . .	4
3.1.1	Tomada de decisões sobre técnicas a utilizar . . . . .	5
3.2	<i>Signal to Noise Ratio</i> . . . . .	6
<b>4</b>	<b>Análise de Resultados</b>	<b>7</b>
4.1	Resultados gerais . . . . .	7
4.2	Imagen <i>Coins</i> . . . . .	8
4.3	Imagen <i>Coins 2</i> . . . . .	10
4.4	Imagen <i>Coins 3</i> . . . . .	12
<b>5</b>	<b>Limitações</b>	<b>14</b>
<b>6</b>	<b>Conclusão</b>	<b>15</b>

## **1 Introdução**

No âmbito da unidade curricular de Visão por Computador, englobada no perfil de Computação Gráfica, é-nos proposto fazer mais um tutorial. Sendo este tutorial, o segundo a ser realizado neste semestre. Neste tutorial é apenas pedido para fazer um programa de reconhecimento de imagem.

O programa de reconhecimento de imagem é desenvolvido usando técnicas de aprendidas nas aula teóricas e práticas, é pedido que este consiga identificar, reconhecer e contar o número de moedas numa imagem. Para além do número de moedas, também é necessário que o programa consiga reconhecer o tipo das moedas.

Por fim, irá ser feita uma análise de todos os resultados obtidos.

## 2 Input

```
imagem = input('Select the image:\n      (1) coins\n      (2) coins2\n      (3) coins3\n');

if (imagem==1)
    image = imread('Images\coins.jpg','jpg');
elseif (imagem==2)
    image = imread('Images\coins2.jpg','jpg');
elseif (imagem==3)
    image = imread('Images\coins3.jpg','jpg');
else
    fprintf('Error\n');
end
```

Neste programa existem três imagens possíveis - *coins*, *coins2* e *coins3*. O utilizador pode escolher umas das três imagens introduzindo os valores 1, 2 ou 3, respetivamente. Se o utilizador introduzir algum valor ou carater diferente dos mencionados anteriormente o programa termina.

```
imageGreyScale = rgb2gray(image);

typeOfNoise = input('Select the type of noise:\n      (1) Salt-an-pepper\n      (2) Gaussian Noise\n');
if (typeOfNoise==1)
    typeOfNoise = "salt & pepper";
    noiseArg = input('What is the percentage of occurrence desired?\n      Type "-1" for default values\n:');
elseif (typeOfNoise==2)
    typeOfNoise = "gaussian noise";
    noiseArg = input('What is the variance desired?\n      Type "-1" for default values\n:');
else
    fprintf('Error');
    exit;
end

if(noiseArg == (-1))
    noiseArg = 0.05;
end

resultado = main_image_recognition(image,imagem,imageGreyScale
, typeOfNoise,noiseArg);
```

Depois da escolha imagem esta é convertida para *grayscale* através da função **rgb2gray**, pré-definida em *MATLAB*.

O programa dá a oportunidade ao utilizador para introduzir ruído na imagem. Se o utilizador introduzir o valor 1 escolhe o tipo de *noise salt & pepper*, caso introduza o valor 2 escolhe o ruído gaussiano. Por outro lado, se introduzir outro qualquer valor, o programa vai terminar.

Para cada tipo de ruído vai ser pedido ao utilizador para escolher um valor para usar como argumento no *noise*. Se o utilizador introduzir o valor -1, o programa assume um valor *default* (-1).

No final, é chamada a função que vai tratar do algoritmo com todos os parâmetros necessários.

## 3 Resolução

Neste capítulo iremos explicar o processo adotado para a deteção das moedas em cada imagem.

### 3.1 Algoritmo

O algoritmo utilizado para a deteção e contagem de moedas numa imagem é composto por cinco passos.

- Passo 1 - Para iniciar o processo de deteção das moedas é introduzido na imagem o ruído selecionado pelo utilizador.
- Passo 2 - Em função do tipo de ruído selecionado é utilizado um filtro na imagem com ruído. No caso de o ruído ser do tipo *salt & pepper* é utilizado um filtro *average*. Caso o ruído seja do tipo gaussiano o filtro também é gaussiano.
- Passo 3 - É feito *contrast equalization* como forma de ajustar o contraste das imagens, facilitando a deteção das moedas.
- Passo 4 - São calculadas as *edges* da imagem.
- Passo 5 - O passo final utiliza a função *imfindcircles*, pré definida no *MATLAB*, para encontrar os círculos (moedas) presentes na imagem. O resultado deste função são dois vetores com os centros e raios dos círculos. O único porem da função *imfindcircles* é a necessidade de fornecer um intervalo de valores para proceder à procura de círculos que estejam dentro desse intervalo.

#### Exemplo do algoritmo para a imagem *coins2*

```
% Aplicar ruído à imagem já em grey scale  
noiseImage = imnoise(imageGreyScale,'gaussian',noiseArg);  
  
% Aplicar filtro à imagem com ruído  
filteredImage =  
imgaussfilt(noiseImage,7,'FilterSize',11,'FilterDomain'  
,'spatial');  
  
% Contrast equalization  
eqImage = histeq(filteredImage);  
  
% Calculo das edges  
edgeImage = edge(eqImage, 'canny');  
  
% Deteção dos círculos  
[centers,radii] = imfindcircles(edgeImage,[230 300]  
,'Sensitivity', 0.98);  
  
% Mostrar imagem original com os círculos detetados  
imshow(original); title('Original image');  
viscircles(centers, radii,'EdgeColor','g');
```

### 3.1.1 Tomada de decisões sobre técnicas a utilizar

Como forma de melhorar os resultados obtidos na deteção das moedas foram tomadas algumas decisões em relação ao pré processamento das imagens e parâmetros das funções utilizadas.

A escolha do tipo de filtro utilizado para correção do ruído de cada imagem foi baseado na análise feita num projeto anterior. Dessa forma decidimos utilizar o filtro *average* para as imagens com ruído *salt & pepper*, uma vez que o filtro faz o cálculo do valor médio dos pixels dentro da área do *kernel* (em volta do pixel a corrigir) e sendo este ruído uma alteração em pixels aleatórios e dispersos o filtro *average* remove bastante bem o ruído. Já o filtro gaussiano (passa baixo) é o mais recomendado para as imagens com ruído gaussiano. Os valores utilizados nos dois filtros foi de 11 para o *kernel* e de 7 para o *sigma* no filtro gaussiano.

O *contrast equalization* é feito apenas para as imagens *coins 2* e *coins3*, pois a imagem *coins* não necessitava deste tipo de pré processamento e o resultado com a sua utilização era pior do que o obtido sem a técnica. Em relação às outras duas imagens o *contrast equalization* aumenta o contraste da imagem, fazendo com que as *edges* fiquem mais perceptíveis e melhora a deteção das moedas. O *contrast equalization* provoca uma dispersão uniforme dos valores de brilho da imagem, o que causa o aumento de contraste na imagem.

Por fim, na função *imfindcircles* são utilizados valores pré definidos para o intervalo que define o tamanho dos raios dos círculos a detetar. Como não conseguimos automatizar essa parte da função fizemos uma medição para verificar os valores entre os quais estavam os círculos das diferentes imagens. Em relação à sensibilidade utilizada na função, em cada imagem foram feitas várias observações para ver qual o melhor valor para cada uma.

### 3.2 Signal to Noise Ratio

```
function snr_noise = signal_to_noise_ratio(original, noise_img)
    image_var = 0.0;
    noise_var = 0.0;

    original = double(original);
    noise_img = double(noise_img);

    [l,c,~] = size(original);
    media = mean2(original);

    for i=1:l
        for j=1:c
            image_var = image_var + (noise_img(i,j)-media).^2;
            noise_var = noise_var + (noise_img(i,j)-original(i,j)).^2;
        end
    end

    snr_noise = 10*log10(image_var/noise_var);
end
```

Para calcular o *signal to noise ratio* usou-se uma variação da função fornecida pela docente nos slides da disciplina.

Neste caso, no código os dois ciclos *for* simulam o somatório presente na fórmula dos slides, uma vez que para obter um valor correto da variância esta teria de ser calculada em cada pixel. Os nomes *original* e *noise\_img* representam a imagem original e a imagem com ruído, respectivamente.

## 4 Análise de Resultados

Neste capítulo é feita a análise dos resultados obtidos na identificação de moedas nas diferentes imagens. Para cada imagem são feitas duas análises, uma com ruído *salt & pepper* e outra com ruído gaussiano.

### 4.1 Resultados gerais

Resultados				
Moeda	Ruído	Valor	Moedas Encontradas/ Totais	SNR
Coins	<i>Salt and Pepper</i>	0.05	4/4	3.37
	<i>Gaussian</i>	0.05	4/4	5.75
Coins2	<i>Salt and Pepper</i>	0.05	18/24	6.47
	<i>Gaussian</i>	0.05	20/24	10.54
Coins3	<i>Salt and Pepper</i>	0.05	7/7	3.94
	<i>Gaussian</i>	0.05	7/7	7.02

## 4.2 Imagem *Coins*

A imagem *coins* é a mais simples. Por essa razão o processo de identificar as moedas é mais rápido computacionalmente. Nesta imagem o tipo de ruído introduzido não altera muito os resultados quer a nível da deteção de *edges* quer a nível de pré processamento.

### *Salt & Pepper*

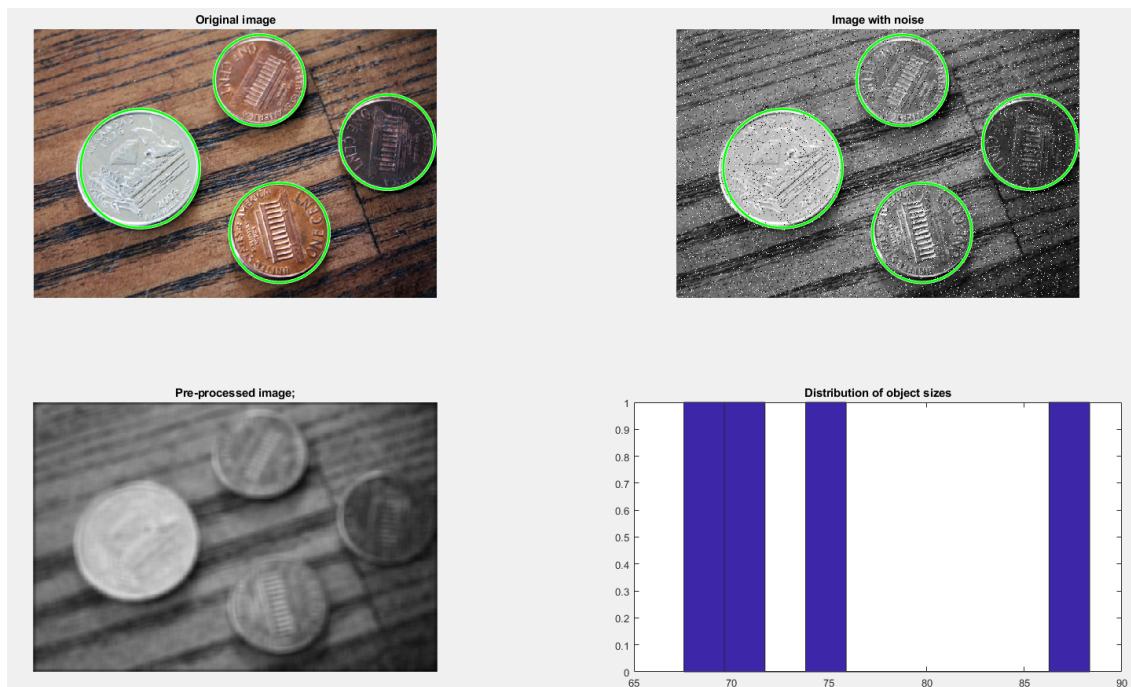


Figura 1: Resultado da imagem *coins* com ruído *salt & pepper*.

### Gaussian



Figura 2: Resultado da imagem *coins* com ruído gaussiano.

### 4.3 Imagem *Coins 2*

A imagem *coins* é a mais complexa, contendo o maior número de moedas e sombras. Por essa razão o processo de identificar as moedas é mais lento computacionalmente. Nesta imagem o tipo de ruído introduzido altera os resultados quer a nível da deteção de *edges* quer a nível do histograma da imagem. Em qualquer uma das situações, certas moedas não são detetadas pela função de deteção, pois este método não é 100% preciso.

#### *Salt & Pepper*

Com o ruído *salt & pepper* são detetadas menos moedas na imagem, mas são detetados mais círculos. Isto é, o ruído provoca alterações nas *edges* da imagem de forma a que a função de deteção deteta círculos onde eles não existem e em algumas moedas não são detetadas.

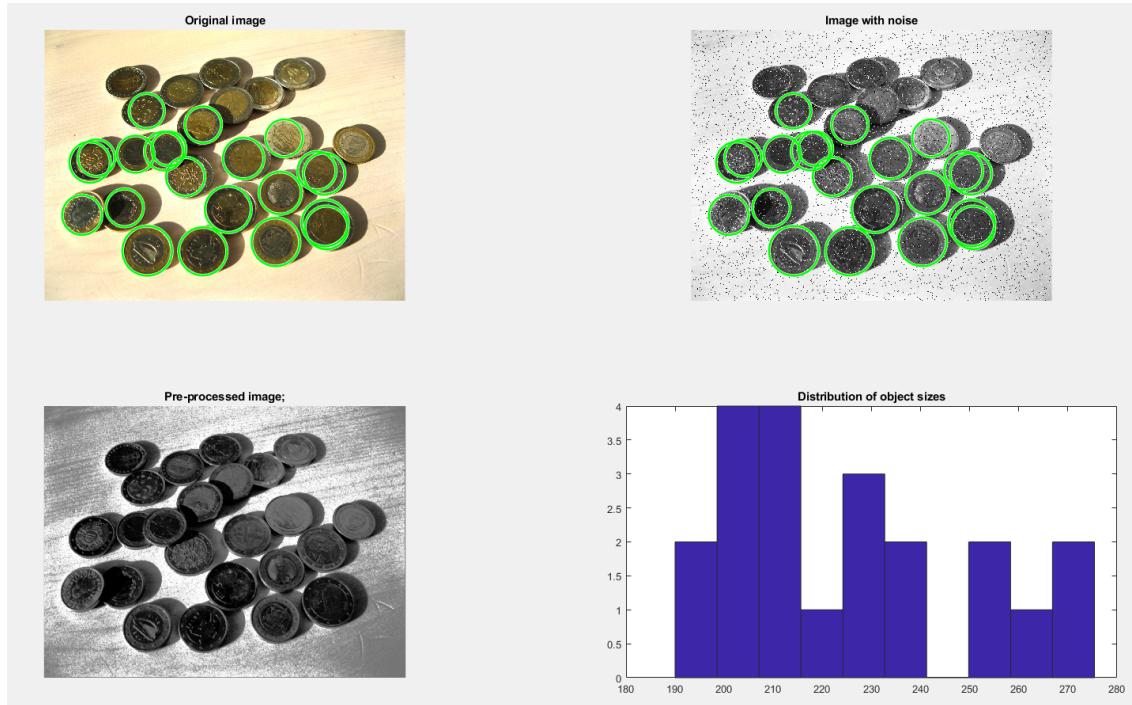


Figura 3: Resultado da imagem *coins2* com ruído *salt & pepper*.

### **Gaussian**

Com o ruído gaussiano são detetadas mais moedas na imagem e menos círculos. Isto é, o ruído não provoca tantas alterações nas *edges*. Sendo detetadas mais moedas e menos círculos não existentes.

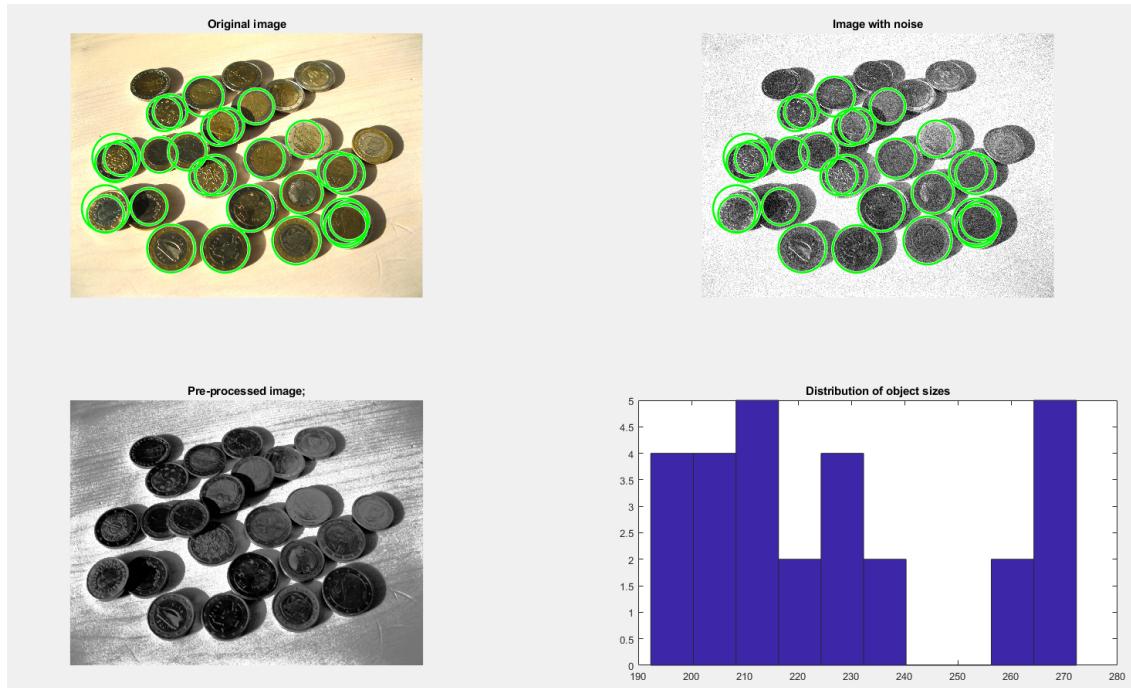


Figura 4: Resultado da imagem *coins2* com ruído gaussiano.

#### 4.4 Imagem *Coins 3*

Na imagem *Coins 3* é possível fazer uma deteção melhor do que nas imagens anteriores, tanto das moedas (faz uma deteção total) como do tamanho das mesmas. Isto é possível graças à cor de fundo da imagem. Uma vez que a imagem tem um fundo a tender para o branco e as moedas têm uma tonalidade complementar oposta, é mais fácil detetar as *edges* e a própria moeda (uma grande variação da cor).

##### *Salt & Pepper*

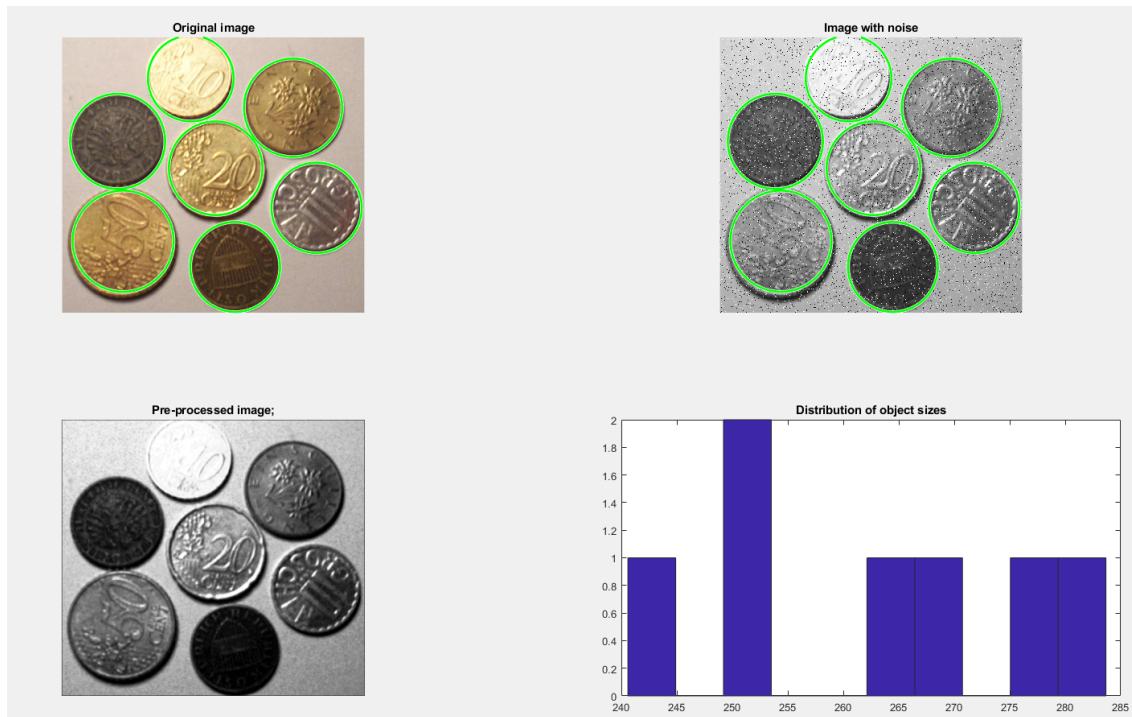


Figura 5: Resultado da imagem *coins3* com ruído *salt & pepper*.

### Gaussian

Neste caso, é feita uma melhor deteção das edges das moedas com ruído *Salt and Pepper* do que com ruído gaussiano mas é apenas uma pequena diferença.

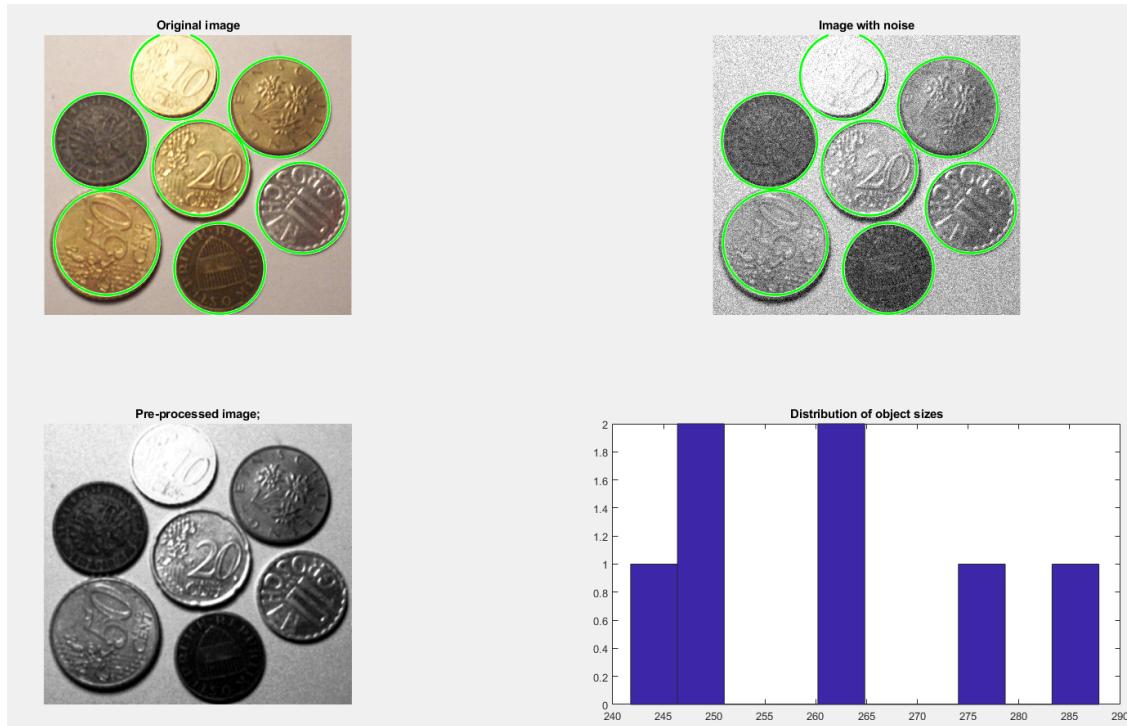


Figura 6: Resultado da imagem *coins3* com ruído gaussiano.

## 5 Limitações

O programa, com os algoritmos atuais, não consegue detetar o tipo das moedas.

Apesar de o programa fazer uma boa deteção das moedas nas imagens 1 e 3, que são imagens pequenas e sem moedas sobrepostas, o programa não consegue detetar corretamente todas as moedas na imagem 2, porque para além de uma grande quantidade de moedas, tem moedas sobrepostas.

O programa nem sempre consegue fazer uma deteção correta da limitação da moeda porque não consegue extrair as *edges* totalmente.

Outra das limitações do programa, é não conseguir fazer a deteção automática das moedas, ou seja, o programa não consegue aplicar as técnicas necessárias automaticamente. Estas tiveram de ser aplicadas pelos elementos do grupo num método de tentativa e erro, sendo isto, um dos aspetos negativos do programa porque, basicamente, para cada imagem é necessária aplicar técnicas diferentes da anterior.

## 6 Conclusão

A realização deste trabalho permitiu ao grupo estabelecer e alargar o conhecimento sobre reconhecimento de imagem e deteção de objetos. Devido a um conjunto de fatores, que foram explicados anteriormente, a deteção de objetos não é algo cem por cento preciso, portanto, apesar de não termos conseguido obter resultados ótimos ficamos satisfeitos com os resultados obtidos.

Mais uma vez, tanto os slides das aulas teóricas como o *MATLAB*, que dispõe de uma documentação muito detalhada, ajudaram na realização do trabalho, que seria impossível sem estes.

Em suma, este trabalho prático ajudou a aumentar o conhecimento do grupo sobre reconhecimento de imagem. O grupo espera poder aplicar estes conhecimentos obtidos em trabalhos futuros.