ADDETC – Área Departamental de Engenharia Eletrónica e Telecomunicações e de Computadores

LEIM -Licenciatura Engenharia informática e multimédia

Processamento Imagem e Visão

Trabalho prático 1

Turma:

LEIM-51D

Trabalho realizado por:

Miguel Silvestre Nº45101

Miguel Távora Nº45102

Docente:

Pedro Jorge

Data: 8/12/2020

Índice

Índice ilustrações

1. Introdução

Imagens nos dias correntes são utilizadas muito frequêntemente no cotidiano, tanto para redes sociais, conteúdos multimédia, filmes entre muitos outros.

O processamento de imagem tem vindo cada vez mais a ser explorado e expandido. O processamento imagem é utilizado na industria principamente

para deteções de falhas, reconhecimento de padrões entre muitos outros. Contudo também é utilizado pelo cidadão comum para criação de conteúdos nomeadamente em redes sociais, websites, filmes e muitos outros.

O primeiro trabalho prático tem como objetivo principal a introdução do desenvolvimento de algoritmos de visão por computador, capaz de contar automaticamente a quantia em dinheiro (moedas) em cima de uma mesa.

Para realizar este algoritmo será utilizado a linguagem de programação Python. Para realizar o trabalho será utilizada a biblioteca OpenCV (Open Source Computer Vision) que possui funções que auxiliam na obtenção dos resultados pretendidos.

O objetivo do trabalho consiste em contar a quantia em dinheiro do número de moedas colocadas em cima de uma mesa de superfície homogénia e clara, ajustando de modo a que o plano do sensor seja paralelo ao plano da mesa.

O algoritmo será testado de modo a que são fornecidos alguns exemplos de treino para desenvolver o algoritmo. O algoritmo será posteriormente testado com outras imagens diferentes das imagens de teste, adquiridas nas mesmas condições.

Um exemplo possível de elaboração do processamento seria:

	OpenCV
1. Leitura de imagens	imread
2. Conversão para níveis de cinzento	cvtColor
3. Binarização (cálculo automático de limiar)	threshold
4. Melhoramento da imagem	getStructuringElement, morphologyEx dilate erode
5. Extracção de componentes conexos	findContours, drawContours connectedComponents
6. Extracção de propriedades	contourArea, arcLength, moments connectedComponentsWithStats
7. Classificação de objectos	

Figura 1 - Esquema desenvolvimento do projeto

Do lado esquerdo está as ações a aplicar na imagem e do lado direito as funções disponíveis na biblioteca para obter os resultados pretendidos na imagem.

2. Desenvolvimento

1.1 Leitura da imagem e exibição de uma imagem

Para fazer a leitura da imagem é utilizada a função do OpenCV *imread*. A função retorna um numpy array com os valores da imagem correspondente. Esta função recebe como argumentos o caminho para a imagem em formato de string. O segundo argumento é o formato da leitura, onde a flag IMREAD_COLOR lê a imagem com os três planos de cor, nomeadamente o vermelho, verde e azul. A flag IMREAD_GRAYSCALE que converte a imagem para cinzento como um algoritmo próprio ficando somente com um plano de cor a variar de 0 a 255. Por fim a flag IMREAD_UNCHANGED que permite receber uma imagem com o canal alpha. Uma possível alternativa é utilizar o valor 1 no segundo argumento para obter com cor, 0 para obter em tons de cinzento e -1 para a imagem com o canal alpha.

Exemplo de utilização da função read: cv2.imread(‘img.jpg’,0)

Para exibir uma imagem é necessário utilizar a função *imshow* do OpenCV. Esta função recebe como argumento o nome da janela que irá ser criada e o numpy array da imagem. Contudo como o programa acaba a janela é também destruída com a conclusão do programa. Para prevenir essa situação utiliza-se a função *waitKey* que interrompe o interpretador de correr o resto do código python. Esta função recebe como argumento um valor que corresponde a uma dada tecla do teclado, quando se clica nessa tecla, o interpretador continua a interpretar o resto do código. Existe também uma função útil do OpenCV que é *destroyAllWindows* e que permite destruir todas as janelas abertas pela função imshow.

Exemplo comum de leitura e visualização de uma imagem:

```
import cv2
import numpy as np
img = cv2.imread('img.jpg',1)
cv2.imshow('image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

1.2 Binarização da imagem

A binarização da imagem consiste essencialmente em transformar uma imagem em somente dois níveis possíveis preto ou branco, quer isto dizer que a imagem somente varia entre valor 0 (preto) e 255 (branco). A binarização é feita de tal maneira que é dado um determinado limiar, a partir desse limiar é feita a transformação onde valores inferiores ao limiar são postos com o valor 0 (preto) e valores superiores a esse valor são postos a 255 (branco).

Para realizar a binarização foi utilizado o plano de cor vermelho, foi utilizado este plano devido ao facto de ser o plano de cor que separa mais as médias. Desta forma é possível minizar a variação intra-classe e permite maximizar a variância inter-classe. Para saber essa informação foi feito um pequeno script em python obtendo o resultado que se segue:

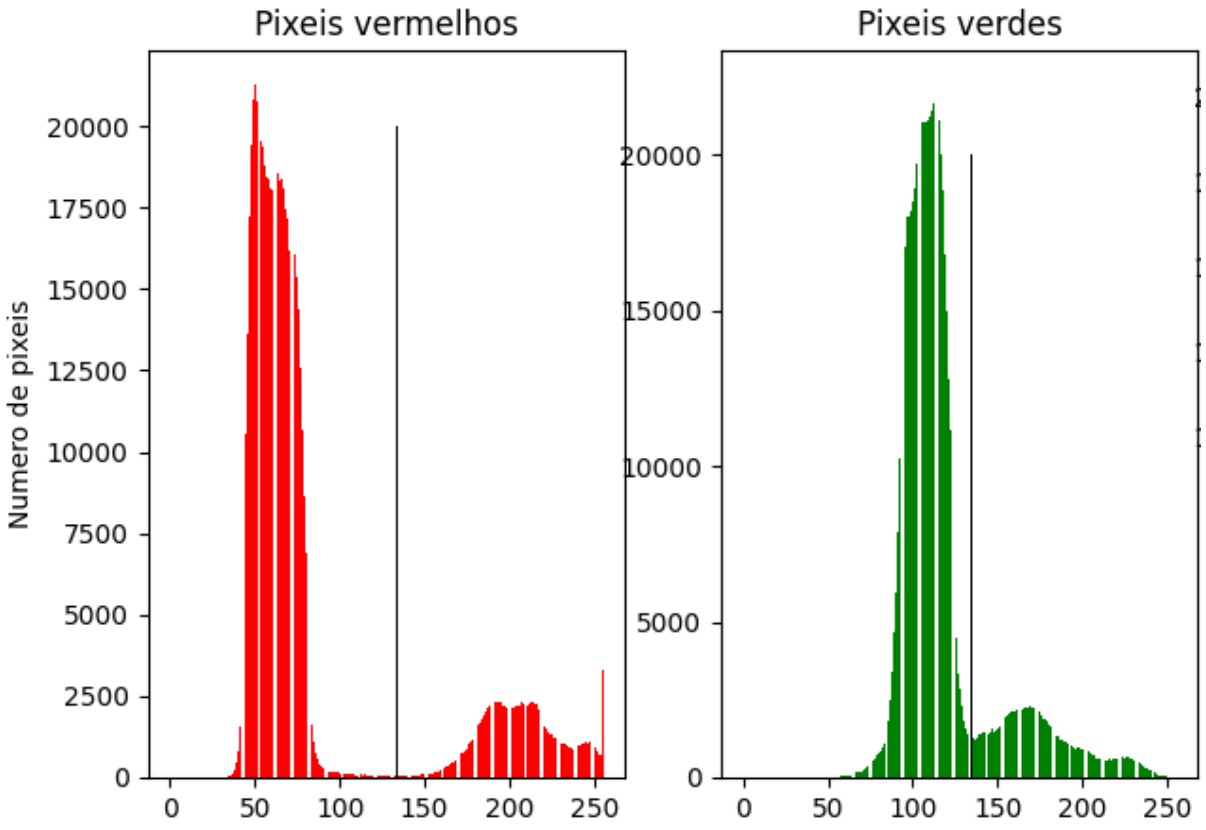

Figura 2 - Histogramas do plano cor vermelho e verde

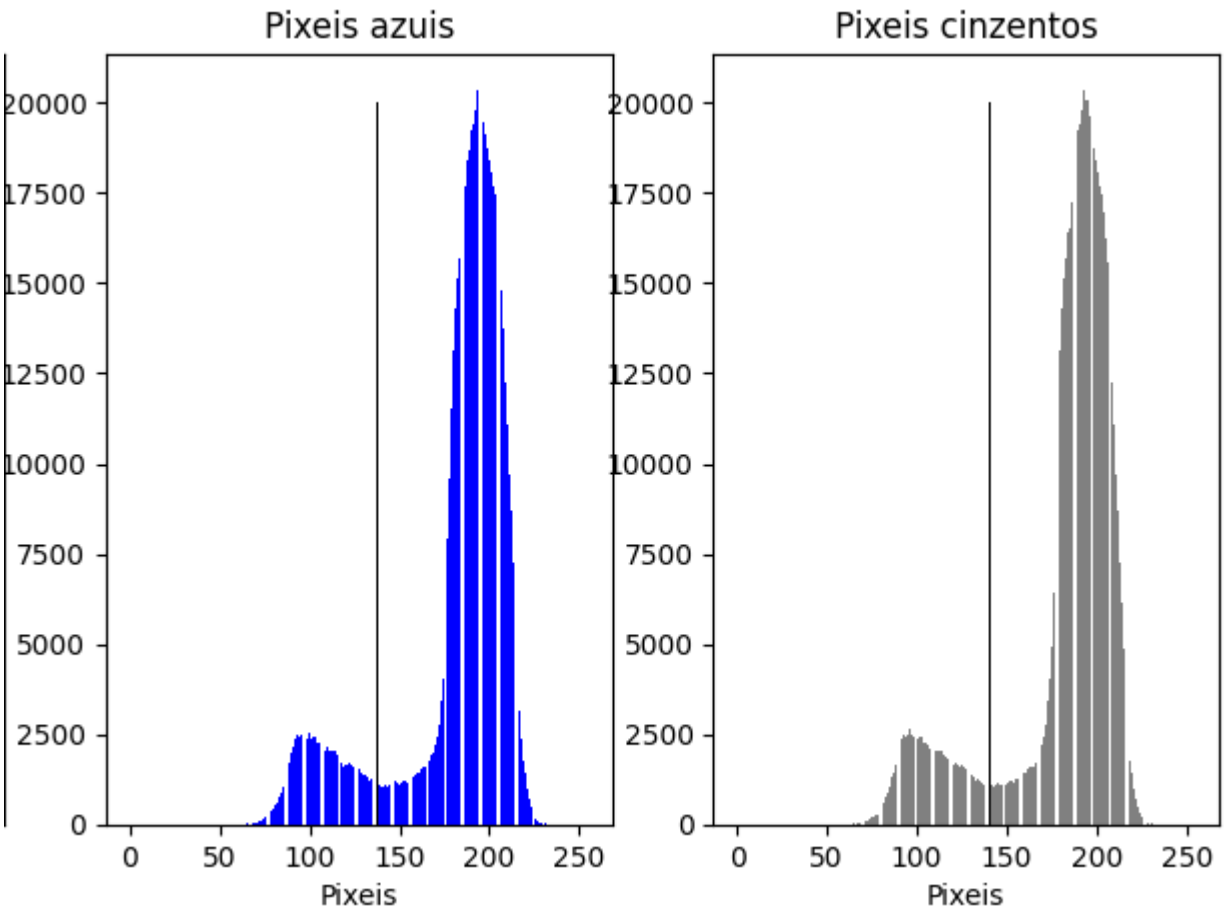

Figura 3 - histograma do plano cor azul e cinzento

Como se pode observar pelos histogramas, o histograma que possui as médias mais sepadas é o plano de cor vermelho. A barra preta é o valor que otimiza a separação da médias pelo método de Otsu, desta forma é possível visualizar de forma mais concreta qual é o melhor plano de cor.

Para realizar a binarização é utilizada a função threshold do OpenCV. Esta função recebe como argumentos a imagem para ser binarizada, o valor do limiar que pode ser alterado, caso seja o método de Otsu este valor é ignorado, o valor para o qual será convertido superior ao limiar e o tipo de algoritmo podendo ser THRESH_BINARY, THRESH_OTSU ou outro no caso do trabalho foi utilizado o método de Otsu.

Contudo com o método de Otsu existiam algumas imagens que ficavam com o interior da mesma cor do fundo e não era possível classificar assim as moedas com o método de Otsu. Como se pode observar na imagem que se segue:

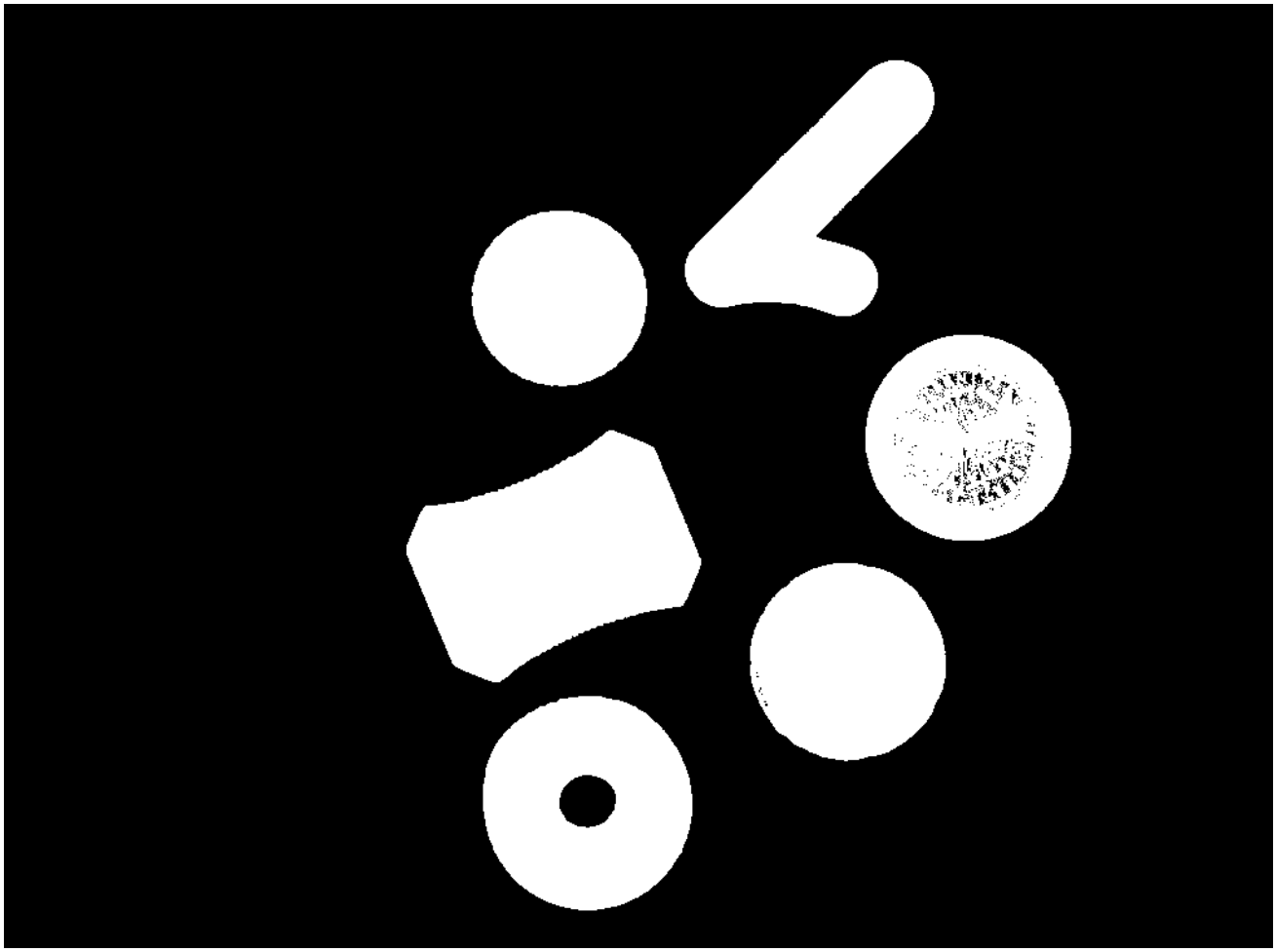

Figura 4 - Imagem binarizada pelo método Otsu

Para resolver esse problema foi utilizado a metodologia de aumentar a exposição da imagem em 127 desta forma existe uma movimentação no histograma e permite obter melhores resultados como se pode observar na imagem que se segue:

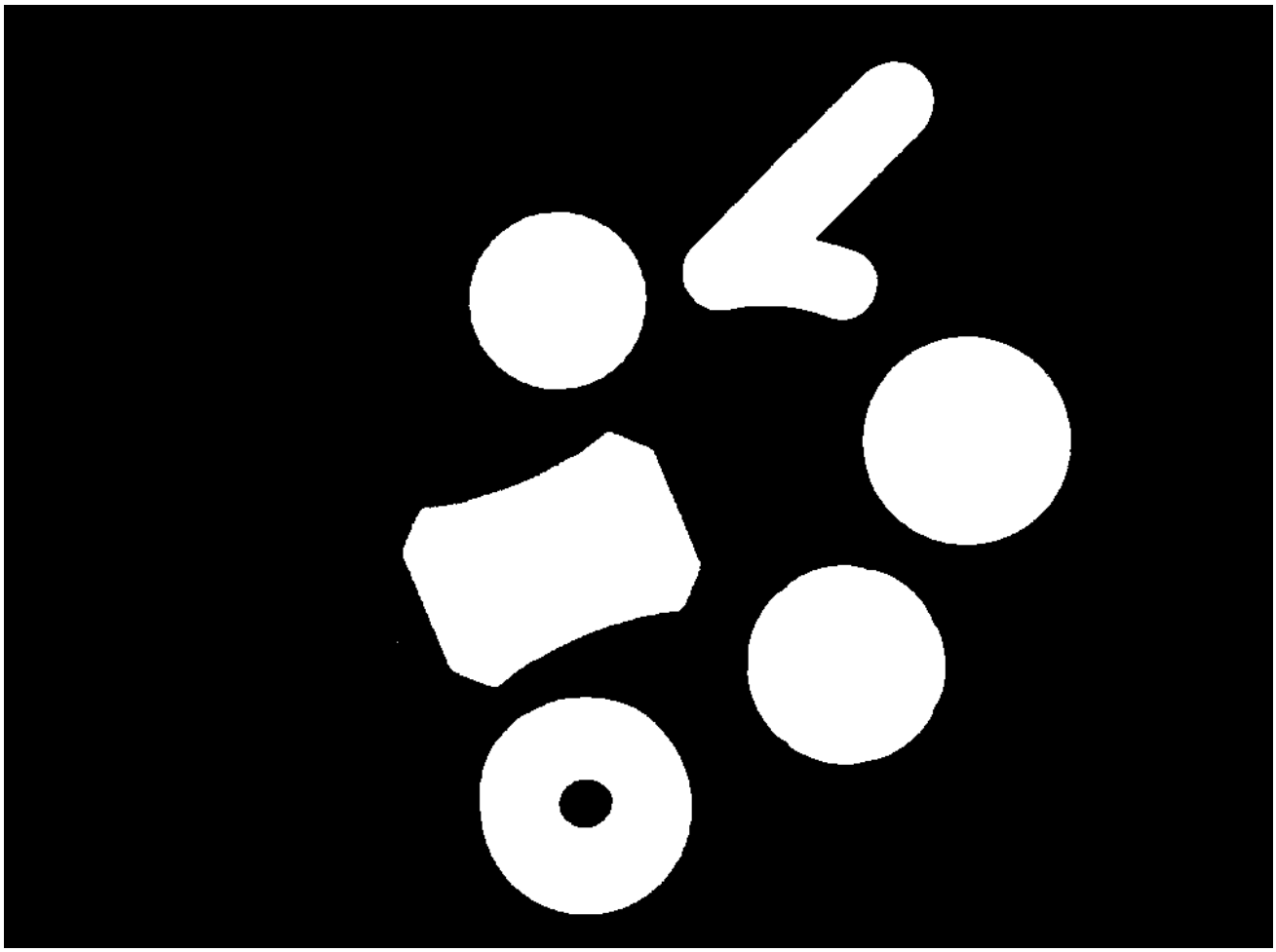

Figura 5 - Imagem binarizada aumentando exposição

1.3 Melhoramento da imagem

O melhoramento da imagem é feito através de operações morfológicas. Operações morfológicas são operações simples baseadas na forma da imagem. Estas operações permitem remover áreas indesejadas que tenham ficado após a binarização da imagem.

Para aplicar os operadores morfológicos é necessário um elemento estruturante, ou *kernel*, que dita a forma do operador que será aplicado na imagem. Este pode ser do tipo quadrado, elipse ou cruz. Os resultados para uma imagem para os diferentes elementos estruturantes são:

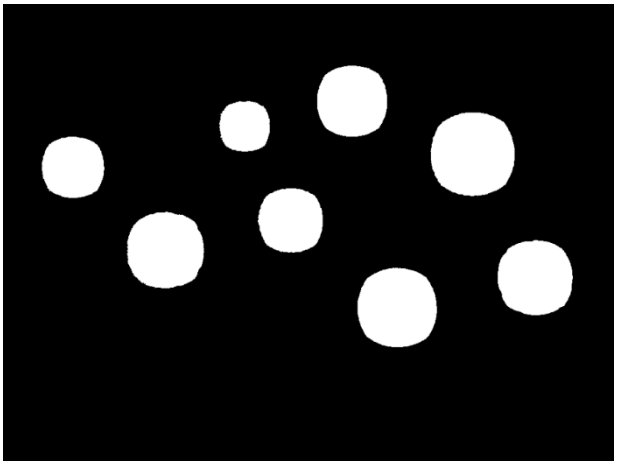

Figura 7 - Aplicação do elemento estruturante cruz

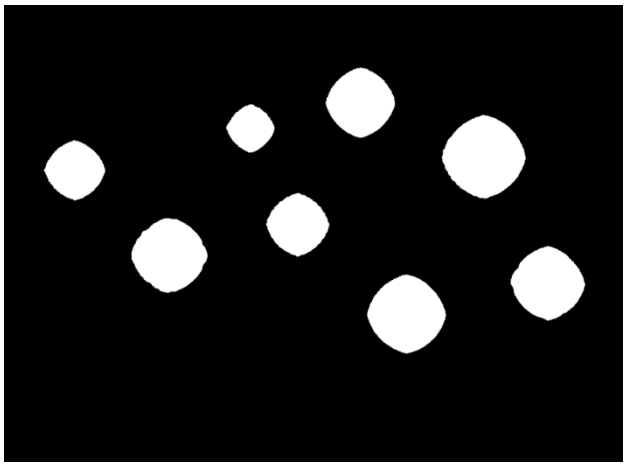

Figura 6 - Aplicação do elemento estruturante quadrado

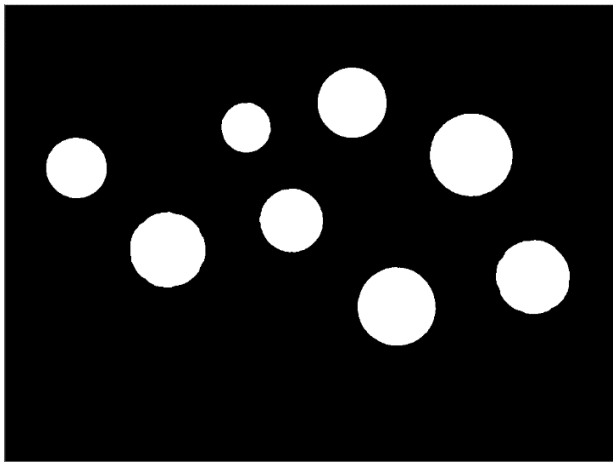

Figura 8 - Aplicação elemento estruturante elipse

Para aplicar o elemento estruturante no OpenCV é utilizada a função getStructuringElement onde se passa o tipo de elemento estruturante e o tamanho em x e y. Ex: kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(35,35))

Nesta fase irá ser usados operadores morfológicos com um elemento estruturante. Foram estudados quatro operadores morfológicos sendo eles:

- Dilatação: A dilatação consiste em tranformar em ‘1’ se pelo menos um valor for ‘1’ no *kernel*. Desta forma a dilatação aumenta a região da forma em primeiro plano. Por isso o píxel de saída é o valor máximo de todos os píxeis na vizinhança do píxel de entrada. É atribuído o valor mínimo aos píxeis exteriores.
- Erosão: A ideia base da erosão é que os píxeis próximos dos limites são descartados dependendo do tamanho do *kernel*. Desta forma a espessura do primeiro plano diminui, sendo útil para remover pequenos ruídos brancos. Por isso o píxel de saída é o valor mínimo de todos os píxeis de todos os píxeis na vizinhança do píxel de entrada. É atribuído o valor máximo aos píxeis exteriores.
- Abertura: Erosão seguida de uma dilatação.
- Fecho: Dilatação seguida de uma erosão.

O elemento estruturante usado foi uma elipse, já que queríamos manter os contornos o mais circulares possíveis, tornando-se assim mais fácil mais a frente detectar os contornos que são de facto moedas.

Na realização do trabalho foi usado somente uma erosão, visto que era necessário eliminar algumas imperfeições no fundo da imagem, nomeadamente píxeis ruidosos, e também para separar as moedas que se encontravam juntas umas as outras.

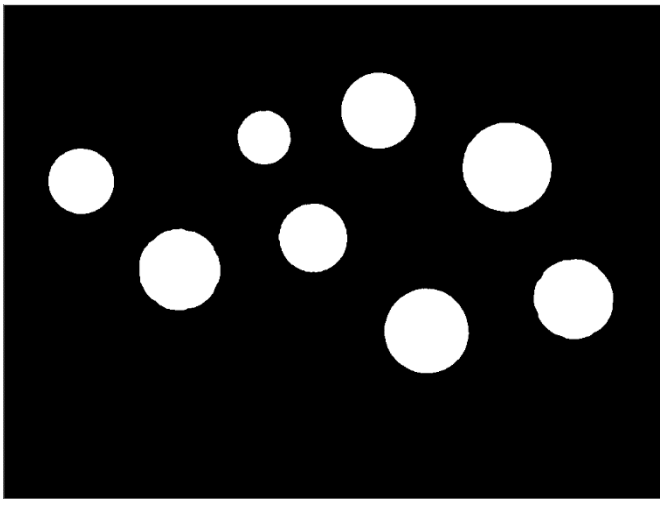

Figura 9 - Resultado dos operados morfológicos na imagem

1.4 Extração de componentes conexos

A extração de componentes conexos pode ser feita através de encontrar os contornos ou através dos componentes conectados. Nesta fase iremos abordar a forma de encontrar os contornos. Os contornos são uma curva que une todos os pontos contínuos tendo a mesma cor ou intensidade, são ferramentas úteis para analisar a forma, deteção e reconhecimento de objetos. Os contornos foram extraídos após o melhoramento da imagem binarizada. Para encontrar os contornos utilizou-se o método *findContours* do OpenCV. Esta função retorna uma lista de contornos encontrados na imagem e também as suas hierarquias. A hierarquia dita se o contorno tem contornos interiores ou exteriores.

Com os contornos encontrados é possível extrair as suas características, sendo elas:

- Área
- Perímetro
- Circularidade
- Centroide

Para verificar se os contornos estavam a ser encontrados corretamente usou-se o método *drawContours*, método que desenha os contornos encontrados na imagem.

Figura 10 - Contornos desenhos na imagem

1.5 Extração de propriedades

A extração de propriedades permite saber informação útil sobre a imagem nomeadamente área do contorno, perímetro do contorno entre outros. Estas características são úteis para distinguir os contornos que são moedas dos que não são. Para isso foram feitos dois métodos discriminantes.

O primeiro método, *circle_contours*, vê a hierarquia de cada contorno e verifica se o elemento pai de cada contorno é igual a -1(valor definido para quando um contorno não tem um elemento pai). Caso o contorno possua um valor diferente de -1 esse contorno e o respetivo contorno pai irão ser removidos da lista de contornos.

O segundo método, *discriminant_contours*, faz um círculo á volta do contorno. Para fazer este círculo foi usado o método do OpenCV, *minEnclosingCircle*. Este método retorna o raio e o centro para desenhar um círculo com a menor área possível a envolver o contorno. Como as moedas são contornos circulares a área obtida com o *drawContour*, método que retorna a área do contorno, irá ser semelhante á área obtida com o raio obtido do *minEnclosingCircle* e caso isso aconteça o contorno irá ser mantido, caso contrário o contorno irá ser removido.

Por fim foi usado novamente o *drawContour* para testar se os contornos mantidos estavam corretos.

Figura 11 - Contornos mantidos após métodos discriminates

1.6 Classificador

Por fim, foi feito o classificador. Foi escolhido um classificador do tipo de distância ao centroide com a distância de *manhattan*. Para tal foram tiradas as áreas das moedas e fez-se a média delas, como se pode observar na imagem que se segue:

	2 Euros	1 Euro	50 Cêntimos	20 Cêntimos	10 Cêntimos	5 Cêntimos	2 Cêntimos	1 Cêntimo
		13509	15113	11989	8855	10732	8209	5460
		13632	15517	12645	8955	10706	8155	5260
		13699	14914	12408	8675	10413	8109	5278
		13852	15200	12423	9377	10762	7841	
		13661	15538	12088	9469		7986	
		13388	15135	11813	9113		8016	
		14019		12187	8788			
Média	17500	13680	15236,16667	12221,85714	9033,14286	10653,25	8052,66667	5332,666667

Figura 12 - Valores centroides das áreas das moedas

No caso da moeda de dois euros foi feita um ponto médio entre 20000 e 15000, uma vez que não existia nenhum caso no conjunto de treino com essa moeda. No entanto sabe-se que área da moeda de dois euros irá ser ligeiramente maior que a de cinquenta cêntimos. O grupo supõe que o classificador irá funcionar para moedas de dois euros, porém não é possível testar.

Após cada moeda estar classificada, vai ser desenhado um círculo a volta delas, mais uma vez com ajuda do método minEnclosingCircle e escrito o seu valor com ajuda do método *putText*.

No processo de classificação cada moeda classificada vai incrementar um contador, contador este que vai servir para contar a quantia de dinheiro na imagem para ser posteriormente exibido na imagem através do *putText* no canto inferior esquerdo da imagem.

Figura 13 - Resultado final da classificação

3. Conclusão

Com o desenvolvimento deste projeto o grupo adquiriu os seguintes conhecimentos:

- Utilização dos diferentes planos de cor para obtenção do resultado pretendido, baseado nos histogramas das diferentes componentes
- Binarização de uma imagem para obter as formas separadas da imagem de fundo
- Utilização de elementos estruturantes para melhorar a qualidade da imagem
- Encontrar os contornos de uma imagem para ir buscar os píxeis onde estão as formas geométricas
- Extrair propriedades para verificar se é a forma geométrica pretendida de obter
- Classificar baseado nas propriedades obtidas

Durante a realização do trabalho prático foi compreendida a importância de conceitos como:

- Visualização dos histogramas para determinar qual o melhor espaço de cor para binarizar
- Escolha do melhor elemento estruturante
- Escolha da melhor operação morfológica a aplicar

As maiores dificuldades foram na fase de binarizaçao da imagem, pelo facto de estar a binarizar a imagem sem ver previamente os histogramas de cada cor. Após ultrapassar essa dificuldade o trabalho foi realizado com fluidez, não surgindo nenhuma dificuldade em concreto.

Em relação aos resultados todas as imagens foram bem classificadas em relação ao dinheiro presente em cada imagem e em cada moeda individualmente em cada imagem.

4. Bibliografia

https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html

https://docs.opencv.org/master/d4/d73/tutorial_py_contours_begin.html

https://docs.opencv.org/master/dd/d49/tutorial_py_contour_features.html