

Instituto Superior de Engenharia de Lisboa

Processamento de Imagem e Visão

Relatório do Trabalho Prático 1

Shun Wang nº45410 - Arman Freitas nº45414

Docente: Pedro Jorge

3 de Novembro de 2019

Conteúdo

1	Introdução	5
1.1	Binarização da imagem	7
1.2	Melhoramento da imagem	8
1.3	Extração de propriedades e Classificação dos objetos	9
2	Desenvolvimento	10
2.1	Conversão para níveis de cinzento e Binarização da imagem	10
2.2	Melhoramento de imagem	10
2.3	Extracção de componentes conexos	15
2.4	Extração de propriedades e Classificação de objetos	15
3	Resultados Experimentais	17
3.1	Antes da binarização	17
3.2	Após operadores morfológicos	18
3.3	Resultados finais	19
4	Conclusões	21
5	Bibliografia	22
6	Anexos	23

Listas de Figuras

1	Exemplo do limiar/ <i>threshold</i> com Binarização por Limiar	7
2	Exemplo do limiar/ <i>threshold</i> com o Método de Otsu	8
3	Imperfeição do fundo na imagem binarizada	11
4	Imagen binária após a erosão	12
5	Moeda de 1 euro mal binarizada	13
6	Moeda de 1 euro mal binarizada	14
7	Gráfico com média das moedas	15
8	Resultado final	16
9	<i>Grayscale</i> da imagem P1000699s	17
10	<i>Grayscale</i> da imagem P1000703s	17
11	<i>Grayscale</i> da imagem P1000705s	17
12	<i>Grayscale</i> da imagem P1000706s	17
13	<i>Grayscale</i> da imagem P1000709s	17
14	<i>Grayscale</i> da imagem P1000710s	17
15	<i>Grayscale</i> da imagem P1000713s	18
16	<i>Grayscale</i> da imagem P1000697s	18
17	<i>Grayscale</i> da imagem P1000698s	18
18	P1000699s com binarização melhorada	18
19	P1000703s com binarização melhorada	18
20	P1000705s com binarização melhorada	18
21	P1000706s com binarização melhorada	18
22	P1000709s com binarização melhorada	18
23	P1000710s com binarização melhorada	18
24	P1000713s com binarização melhorada	19
25	P1000697s com binarização melhorada	19

26	P1000698s com binarização melhorada	19
27	P1000699s <i>output</i>	19
28	P1000703s <i>output</i>	19
29	P1000705s <i>output</i>	19
30	P1000706s <i>output</i>	19
31	P1000709s <i>output</i>	19
32	P1000710s <i>output</i>	19
33	P1000713s <i>output</i>	20
34	P1000697s <i>output</i>	20
35	P1000698s <i>output</i>	20

1 Introdução

O presente trabalho tem o intuito de deteção e reconhecimento de moedas.

Recorremos à ferramenta *Python* e à biblioteca *OpenCV* encontrada no mesmo para o realizar.

O processamento de imagem é uma área da informática que tem vindo a crescer exponencialmente durante os últimos anos. Hoje em dia, está presente em diversos campos da tecnologia que usamos diariamente. Desde o uso de filtros no *Instagram* ou *Snapchat* até à segurança dos nossos dispositivos (reconhecimento facial). Aplicações como *Adobe Photoshop*, permitem-nos mais controlo sobre como queremos numa imagem.

No entanto, iremos-nos focar na manipulação de imagens binárias, de forma a extrair as componentes necessárias. Algoritmos mais precisos poderão fazer uso de *Machine Learning* para a deteção de padrões, contudo, apenas nos vamos incidir na manipulação de imagens e extração de características.

Como mencionado acima, o algoritmo será capaz contar o total da soma das moedas que se encontra numa determinada imagem.

De forma a chegarmos ao trabalho realizado, iremos seguir o seguinte esquema providenciado pelo docente da Unidade Curricular:

	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	

Tabela 1: Esquema do desenvolvimento do projeto

De seguida, aprofundaremos melhor os seguintes conceitos:

- **Binarização da imagem**
- **Melhoramento da imagem**
- **Extração de propriedades e Classificação dos objetos**

1.1 Binarização da imagem

A binarização da imagem é uma parte crucial do trabalho, o nosso objetivo nesta secção do trabalho é separar qualquer objeto do fundo. Conseguindo assim, que o fundo seja totalmente ignorado, e, apenas os objetos sejam visíveis. Este processo pode ser feito de diversas formas, no entanto, nós utilizaremos o **Método de Otsu**. É de notar que poderíamos utilizar uma **Binarização por Limiar** que, apenas detecta um limiar no histograma da imagem monocromática, como se pode ver na seguinte figura:

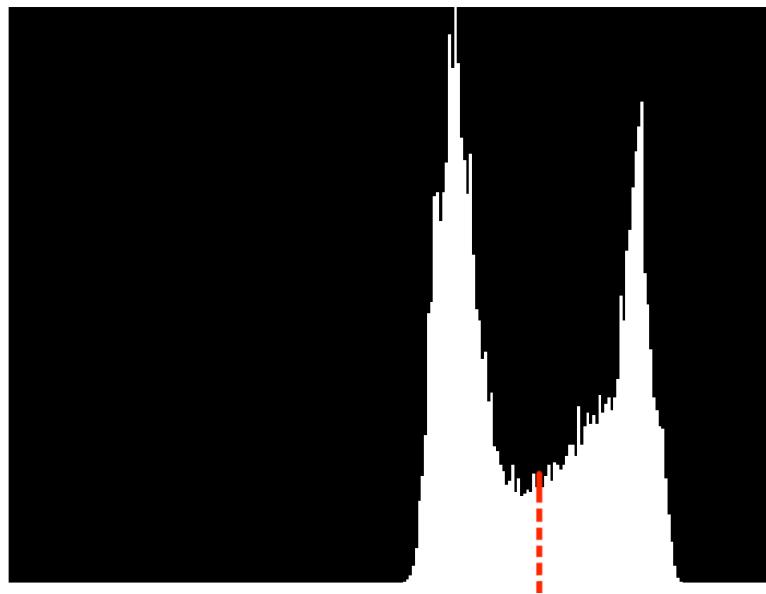


Figura 1: Exemplo do limiar/*threshold* com **Binarização por Limiar**

Já o **Método de Otsu**, pretende testar todos os possíveis limiares e, verificar o que afasta mais as médias, tentando também minimizar a variância intra-classe, ou, equivalentemente maximizar a variância inter-classe. A variância intra-classe é calculada da seguinte forma:

$$\sigma^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

Onde ω_0 e ω_1 são as probabilidades de o pixel faz parte do fundo ou é um objeto.

Já a variância inter-classes possuí a seguinte fórmula:

$$\sigma_B^2 = \sigma^2 - \sigma_W^2$$

Após estes cálculos (até $t =$ valor máximo), os resultados terão o aspetto da seguinte tabela, onde apenas é verificado o máximo valor para a variância inter-classes e o mínimo para a variância intra-classes, escolhendo-se assim esse limiar:

Threshold	T=0	T=1	T=2	T=3	T=4	T=5
Within Class Variance	$\sigma_W^2 = 3.1196$	$\sigma_W^2 = 1.5268$	$\sigma_W^2 = 0.5561$	$\sigma_W^2 = 0.4909$	$\sigma_W^2 = 0.9779$	$\sigma_W^2 = 2.2491$
Between Class Variance	$\sigma_B^2 = 0$	$\sigma_B^2 = 1.5928$	$\sigma_B^2 = 2.5635$	$\sigma_B^2 = 2.6287$	$\sigma_B^2 = 2.1417$	$\sigma_B^2 = 0.8705$

Figura 2: Exemplo do limiar/*threshold* com o **Método de Otsu**

1.2 Melhoramento da imagem

De seguida, o melhoramento de imagem permite que com a imagem binarizada seja possível remover áreas que possam não ter ficado de forma desejada.

Aqui usaremos operadores morfológicos como:

- **Dilatação**

O valor do pixel de saída é o valor máximo de todos os pixels na vizinhança do pixel de entrada.

É atribuído o valor mínimo aos pixels exteriores

- **Erosão**

O valor do pixel de saída é o valor mínimo de todos os pixels na vizinhança do pixel de entrada.

É atribuído o valor máximo (1 ou 255) aos pixels exteriores

- **Abertura**

Erosão seguida de uma dilatação

- **Fecho**

Dilatação seguida de uma erosão

1.3 Extração de propriedades e Classificação dos objetos

Finalmente, após a imagem binária se encontrar melhorada e, tenham sido encontradas as regiões (**Extração de componentes conexos**), passamos à **extração de propriedades e classificação de objetos**.

Apartir das regiões etiquetadas, é possível calcular diversos tipos de propriedades, tais como:

- Área

$$A = \sum_{(r,c) \in R}$$

- Centróide

$$r = 1/A \sum_{(r,c) \in R} \times r$$

$$c = 1/A \sum_{(r,c) \in R} \times c$$

- Pixels de perímetro

$$P_4 = \{(r, c) \in R | N_8(r, c) - R \neq \emptyset\}$$

$$P_8 = \{(r, c) \in R | N_4(r, c) - R \neq \emptyset\}$$

- Circularidade

$$C_1 = |P|^2/A$$

Assim, utilizando estas propriedades conseguimos chegar a uma classificação de objeto para cada região da imagem.

2 Desenvolvimento

2.1 Conversão para níveis de cinzento e Binarização da imagem

Como explicado anteriormente, foi utilizado o **Método de Otsu** como técnica de *threshold*, pois como descrito anteriormente, possuí uma maior exatidão a escolher o *threshold*. No entanto antes de o fazer, fizemos um *slice* na imagem para apenas obter os seus tons de cinzento.

2.2 Melhoramento de imagem

Já no melhoramento de imagem, apenas utilizámos uma operação morfológica, sendo este a erosão. A razão do uso do mesmo, foi para que fosse possível apagar completamente alguns pequenos artefactos que se encontravam no fundo. Sendo estes artefactos mínimos, após uma erosão são facilmente removidos, como demonstra a seguinte figura:

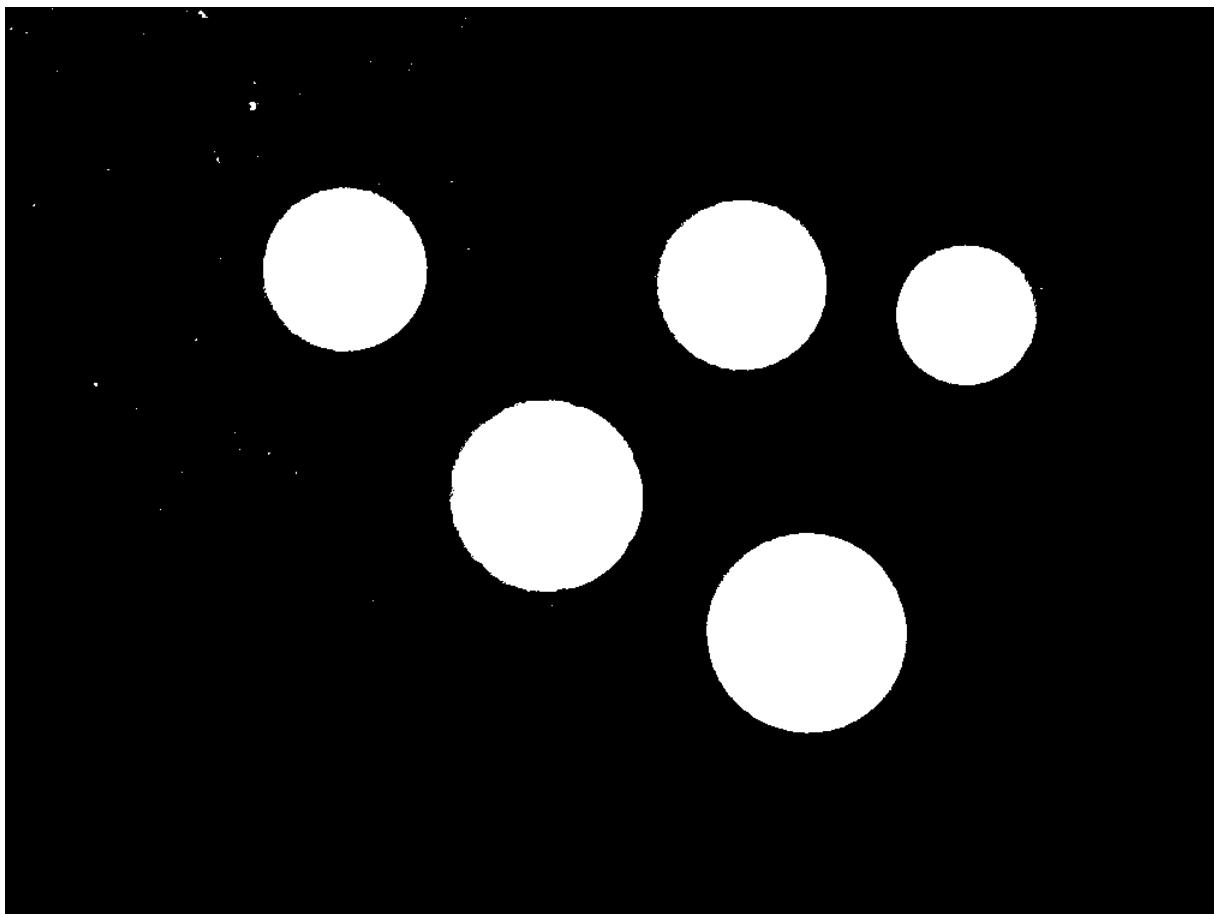


Figura 3: Imperfeição do fundo na imagem binarizada

O resultado da aplicação da erosão foi o seguinte:

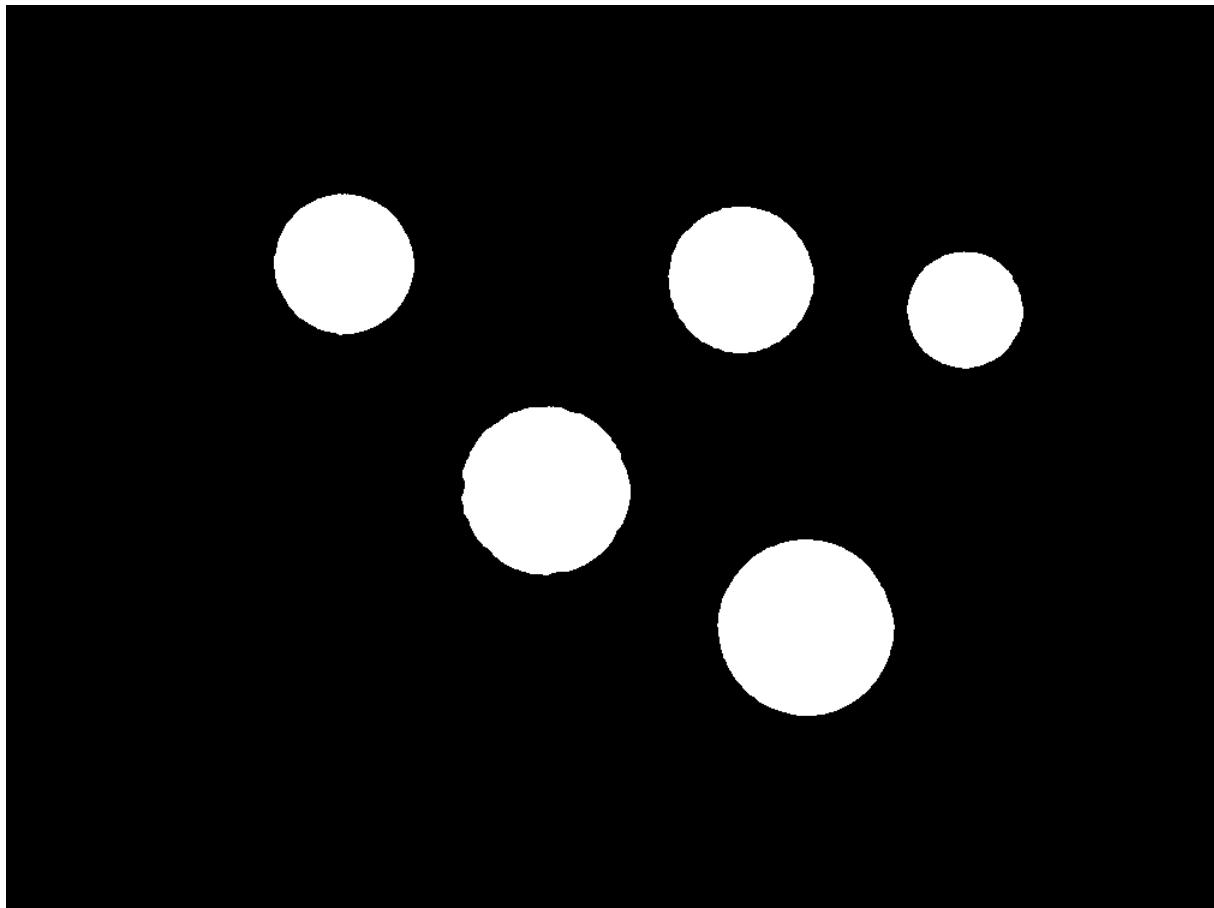


Figura 4: Imagem binária após a erosão

Apesar de termos sucesso nesta imagem, mais em frente deparámos-nos com umas moedas que continham áreas não cobertas:

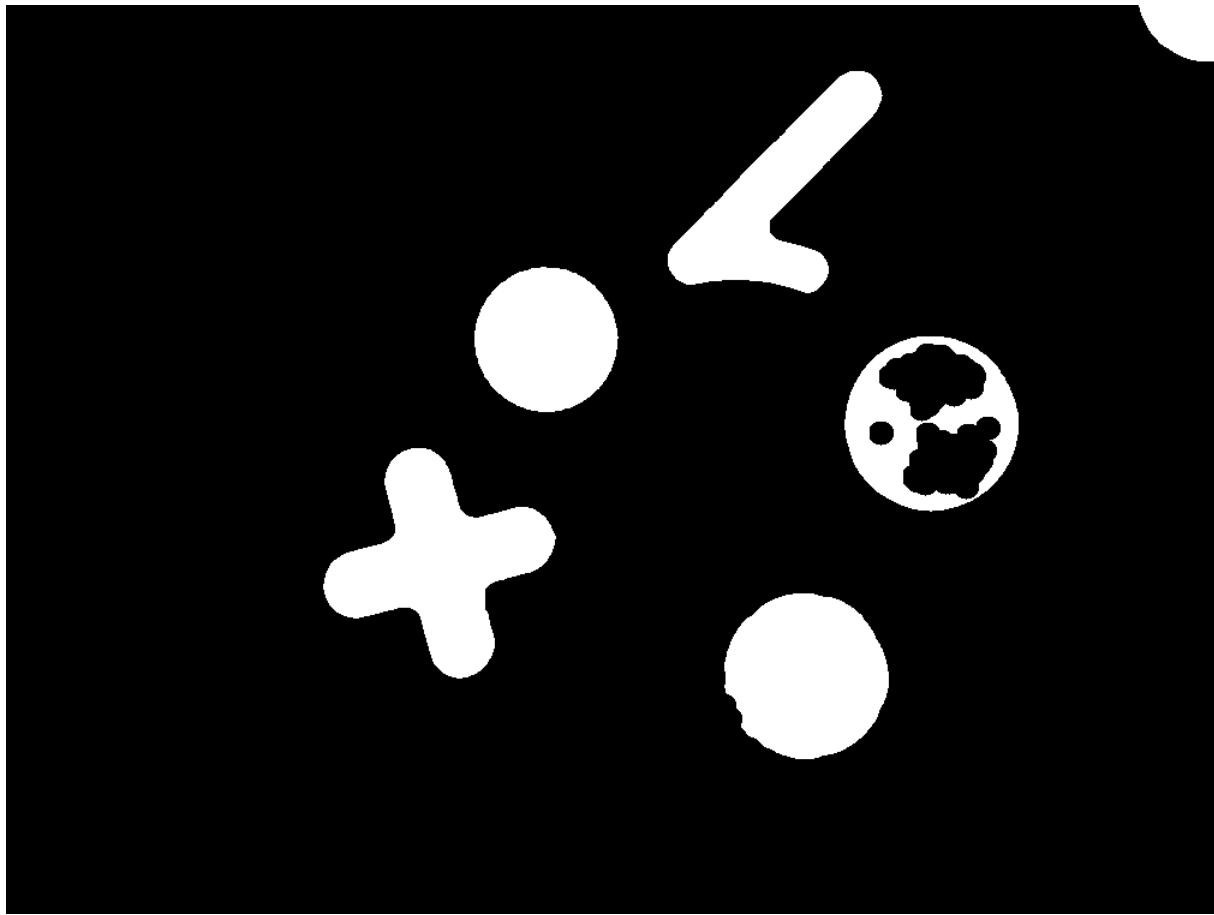


Figura 5: Moeda de 1 euro mal binarizada

Assim, prosseguimos com a utilização de uma dilatação seguida de uma erosão (fecho). Isto para que, ao ser dilatada, a região iria fechar e, para que não fizesse noutras imagens com que duas moedas se juntassem erodíamos a região logo de seguida.

Após algum tempo a mexer com os valores do elemento estruturante utilizado e mesmo a sua forma, tanto na dilatação como na nova erosão. Não conseguimos encontrar valores que nos satisfizessem. Essa dilatação e erosão foram apagadas.

Apenas permanecemos com a primeira erosão. Todavia, ocorreu-nos a ideia de que se aumentássemos a exposição da imagem, a face das moedas perdia o detalhe, não sendo possível a formação de áreas abertas.

Resolvemos aumentar a exposição da imagem antes de a binarizar e, após apenas uma erosão, o nosso problema foi resolvido.

O resultado final foi o seguinte:

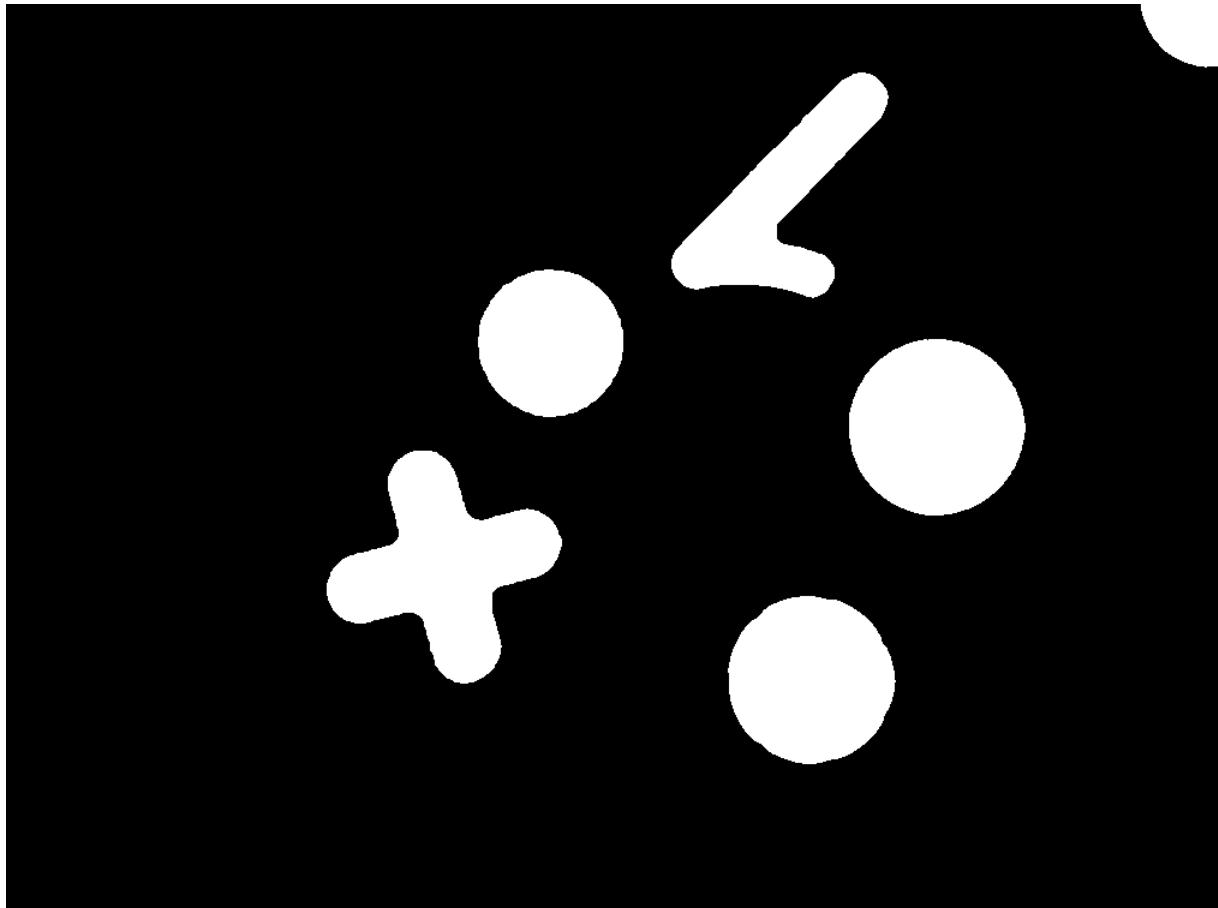


Figura 6: Moeda de 1 euro mal binarizada

2.3 Extracção de componentes conexos

Nesta parte do trabalho, apenas utilizámos a função *findContours* do *OpenCV* para recolher tanto os contornos da nossa imagem binarizada, tanto como as suas hierarquias.

Os **contornos** contêm todas as regiões encontradas e, as **hierarquias** maioritariamente nos dizem se o contorno contém algum contorno *child*. Ou seja qualquer objeto que contenha um contorno *child* não é definitivamente uma moeda, pois uma moeda não tem buracos.

2.4 Extração de propriedades e Classificação de objetos

De modo a poder distinguir se um objeto é uma moeda ou não e, se for qual é, utilizámos propriedades tais como, a circularidade, a área e o perímetro.

Com apenas estas propriedades conseguimos reconhecer se um objeto é circular. Assim, colocámos como regra: qualquer objeto com uma certa circularidade (neste caso entre 13 e 14) e, sem áreas abertas é uma moeda.

A partir daí fizemos uma média das áreas de cada moeda, com apenas uma pequena porção das imagens. Conseguimos um rascunho de um gráfico 1D com um ponto para cada moeda, como sugere o seguinte gráfico:

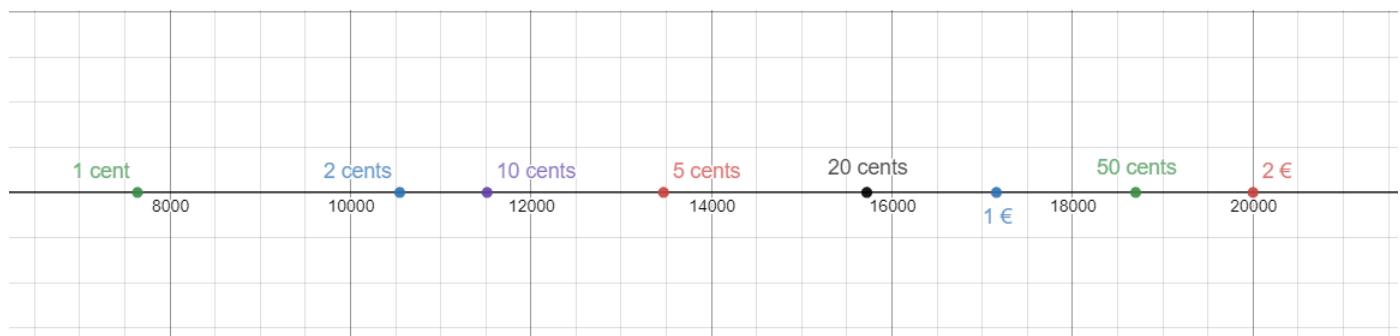


Figura 7: Gráfico com média das moedas

A ideia é, uma determinada área corresponde à moeda cujo ponto está mais próximo. E assim, concluímos a fase final do trabalho, detetamos que moeda é e vamos somando ao contador, o *output* final de cada imagem tem a seguinte aparência:



Figura 8: Resultado final

3 Resultados Experimentais

Esta secção contém todos os resultados que obtivemos no trabalho.

3.1 Antes da binarização

Antes da binarização, as imagens em *grayscale* com maior exposição, para remover detalhes indesejáveis:

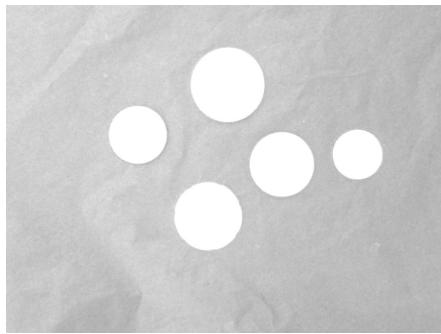


Figura 9: *Grayscale* da imagem
P1000699s

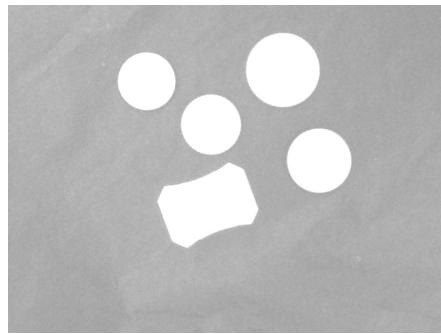


Figura 10: *Grayscale* da imagem
P1000703s

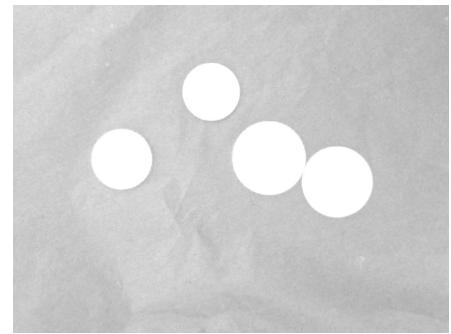


Figura 11: *Grayscale* da imagem
P1000705s

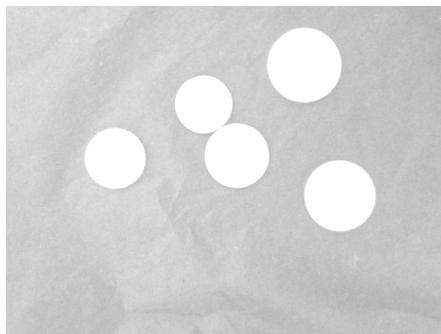


Figura 12: *Grayscale* da imagem
P1000706s

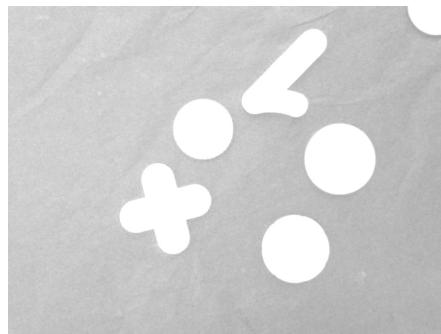


Figura 13: *Grayscale* da imagem
P1000709s

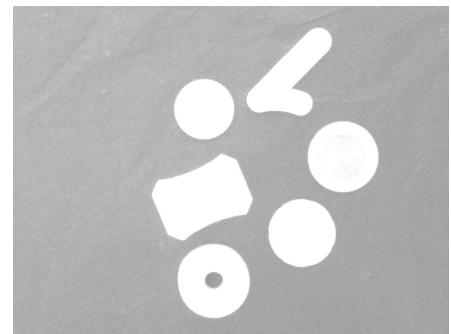


Figura 14: *Grayscale* da imagem
P1000710s

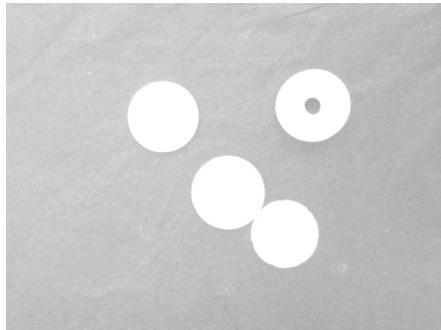


Figura 15: *Grayscale* da imagem P1000713s

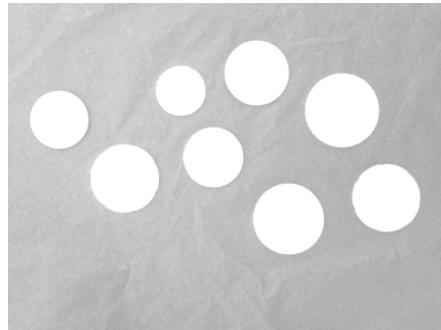


Figura 16: *Grayscale* da imagem P1000697s

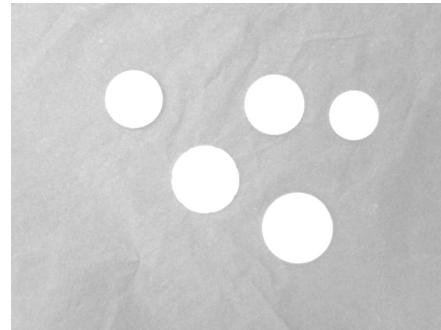


Figura 17: *Grayscale* da imagem P1000698s

3.2 Após operadores morfológicos

Depois de binarizar com o **método de Otsu** e, aplicar a erosão a aparência é a seguinte:

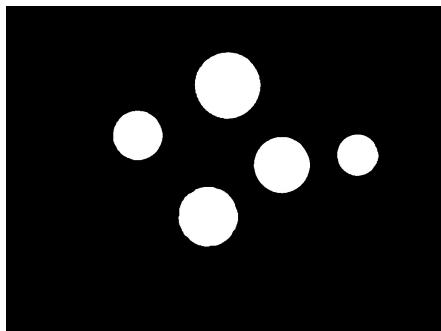


Figura 18: P1000699s com binarização melhorada

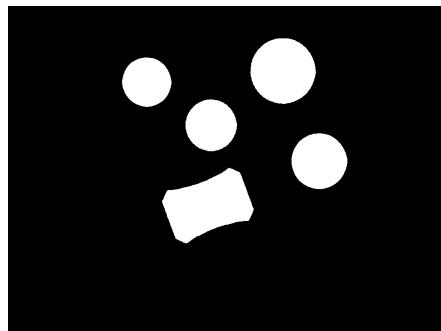


Figura 19: P1000703s com binarização melhorada

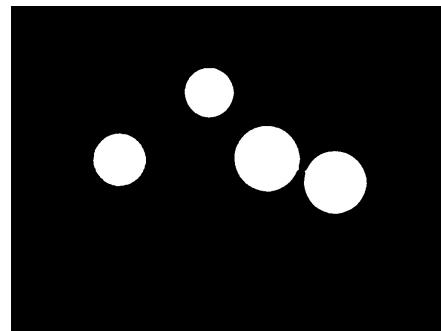


Figura 20: P1000705s com binarização melhorada

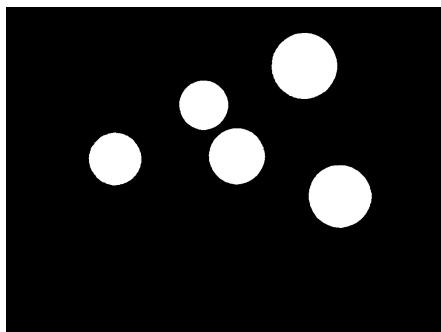


Figura 21: P1000706s com binarização melhorada

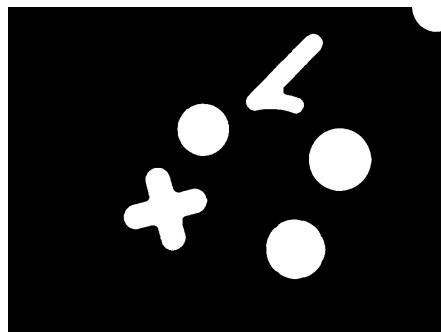


Figura 22: P1000709s com binarização melhorada

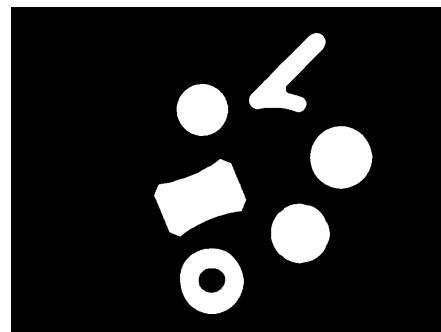


Figura 23: P1000710s com binarização melhorada

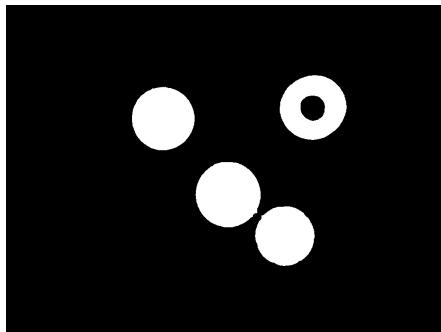


Figura 24: P1000713s com binarização melhorada

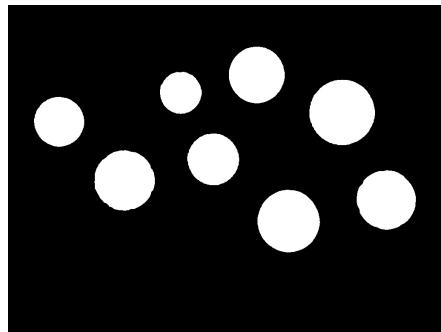


Figura 25: P1000697s com binarização melhorada

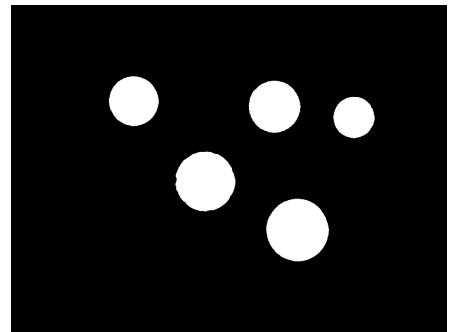


Figura 26: P1000698s com binarização melhorada

3.3 Resultados finais

Os respetivos *outputs* finais foram os seguintes:

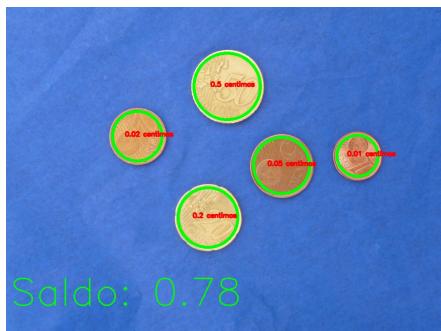


Figura 27: P1000699s *output*



Figura 28: P1000703s *output*

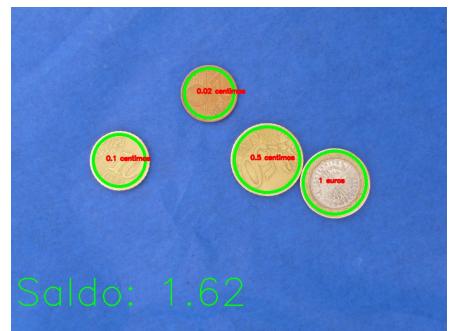


Figura 29: P1000705s *output*

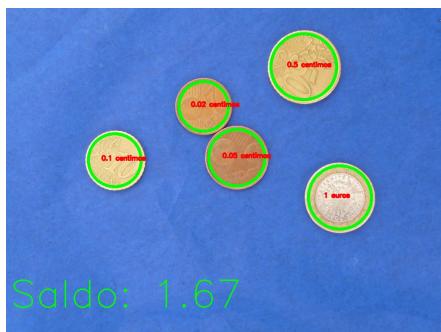


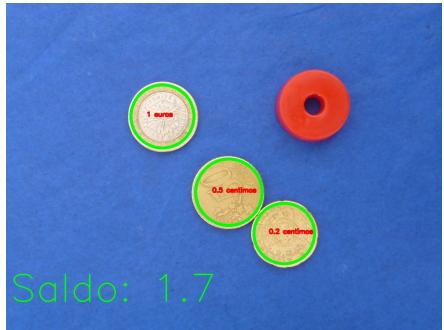
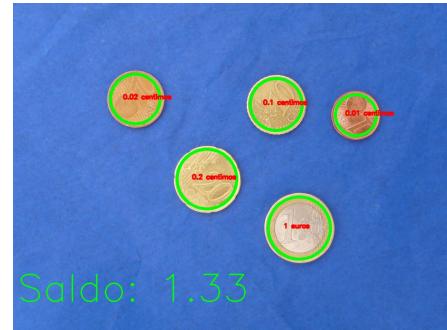
Figura 30: P1000706s *output*



Figura 31: P1000709s *output*



Figura 32: P1000710s *output*

Figura 33: P1000713s *output*Figura 34: P1000697s *output*Figura 35: P1000698s *output*

4 Conclusões

Em boa verdade, este projeto permitiu-nos consolidar a matéria dada nas aulas teóricas. Conseguimos perceber a importância que têm conceitos como por exemplo: a binarização, regiões, operações morfológicas etc.

Viemos a encontrar alguns problemas no desenvolvimento do trabalho, nomeadamente na escolha dos elementos estruturantes para cada operador morfológico. No entanto, conseguimos terminar todas as tarefas pedidas no enunciado.

É de notar também, que foi adicionada compatibilidade para moedas de 2 euros, apesar da mesma não se apresentar em nenhuma das imagens providenciadas pelo docente. De forma a testar o funcionamento da deteção da moeda de 2 euros, utilizámos o *Adobe Photoshop* para duplicar uma das moedas de 1 euro e, aumentar ligeiramente o tamanho, o que resultou na perfeição.

5 Bibliografia

Computer Vision. (2000) Linda G. Shapiro, George C. Stockman

6 Anexos

```

import numpy as np
import cv2

def detectCoin (area, debug=False):
    moedas = [ 2, 1, 0.5, 0.2, 0.1, 0.05, 0.02, 0.01]
    medias = [20000, 17156.75, 18700, 15721, 11513, 13468, 10546, 7640]
    medias = np.asarray(medias)
    distance = np.abs(area-medias)
    # print("Distancia: ", distance)
    return moedas[np.argmin(distance)]


names = ["P1000697s", "P1000698s", "P1000699s", "P1000703s", "P1000705s", "P1000706s", "P1000709s", "P1000710s", "P1000713s"]

font = cv2.FONT_HERSHEY_SIMPLEX

enable = False

for j in range (len(names)):
    img = cv2.imread(str("imgs/" + names[j] + ".jpg"))

    hsv = img[:, :, 2]

    hsv = cv2.add(hsv, np.array([130.0]))

    cv2.imwrite("imgs/Grayscale/" + names[j] + ".jpg", hsv)

    ret2,th2 = cv2.threshold(hsv,127,255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)

    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (20, 20))
    erode = cv2.morphologyEx(th2, cv2.MORPH_ERODE, kernel)

    cv2.imwrite("imgs/Binarizado/" + names[j] + ".jpg", erode)

    contours, hierarchy = cv2.findContours(erode, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    # cv2.imshow('Closing', erode)
    # cv2.waitKey(0)
    # cv2.destroyAllWindows()

    cash = 0
    helper = img.copy()
    for i in range(len(contours)):

        approx = cv2.approxPolyDP(contours[i], 0.01 * cv2.arcLength(contours[i], True), False)
        perimeter = cv2.arcLength(contours[i], True)
        area = cv2.contourArea(contours[i])
        circularity = 0
        if area != 0: circularity = (perimeter**2)/(area)
        a = True
        # helper = img.copy()
        if a: #(7 < len(approx) < 18) and area >= 7400:
            print("Perimeter:", perimeter, "\nArea:", area, "\nCircularity:", circularity)

```

```
if area <= 7000: continue
if circularity >= 13 and circularity <= 14.9:
    if hierarchy[0][i][2] == -1:#detectCoin(area) != 0:
        # print (hierarchy[0][i][2])
        center, radius = cv2.minEnclosingCircle(contours[i])
        c = detectCoin(area)
        if c > 0.5:
            txt = str(detectCoin(area)) + " euros"
        else:
            txt = str(detectCoin(area)) + " centimos"
        helper = cv2.circle(img, (int(center[0]), int(center[1])), int(radius), (0, 255, 0), thickness=5)
        cv2.putText(helper, txt, (int((center[0]) - (radius/2)), int(center[1])), font, 0.5, (0, 0, 255), 2, cv2.LINE_AA)
        # cv2.drawContours(helper, contours[i], -1, (0, 255, 0), thickness=5)
        print (cash, "DETECT", c)
        # cv2.imshow(names[j], helper)
        # cv2.waitKey(0)
        # cv2.destroyAllWindows()
        cash = cash + detectCoin(area)
        print ("-----")

txt = "Saldo:" + str(round(cash, 2))
cv2.putText(helper, txt, (10, 700), font, 2, (0, 255, 0), 2, cv2.LINE_AA)
cv2.imwrite("imgs/Output/" + names[j] + ".jpg", helper)
cv2.imshow(names[j], helper)
cv2.waitKey(0)
cv2.destroyAllWindows()
print ("Cash_Amount=", round(cash, 2))
print ("#" * 50)
if enable: break
```