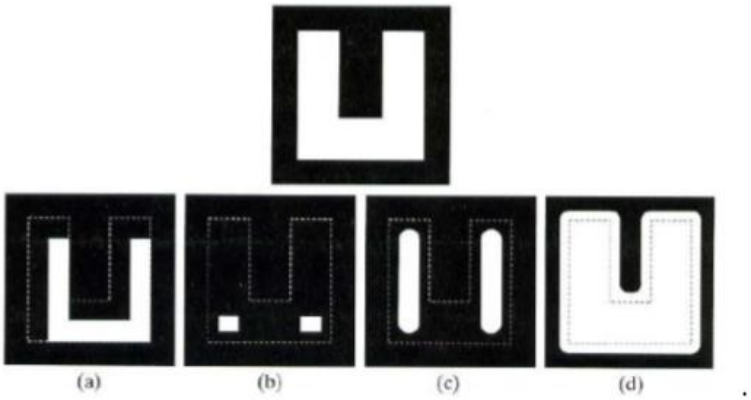
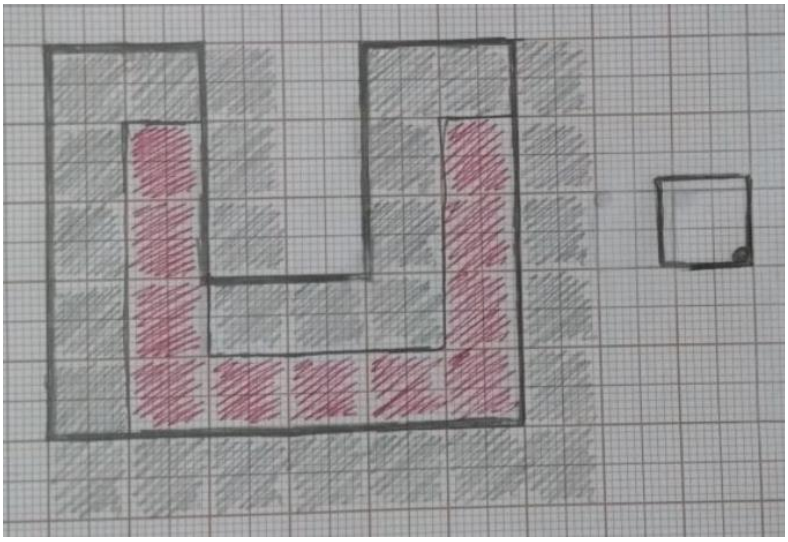
PROCESSAMENTO DIGITAL DE IMAGEM - LISTA DE EXERCÍCIOS 04

Problema 1.1: Considerando a imagem abaixo, onde a linha pontilhada indica o conjunto original, encontre o elemento estruturante e as operações morfológicas que produzem os resultados obtidos nas figuras (a) - (d). Mostre a origem de cada elemento estruturante.

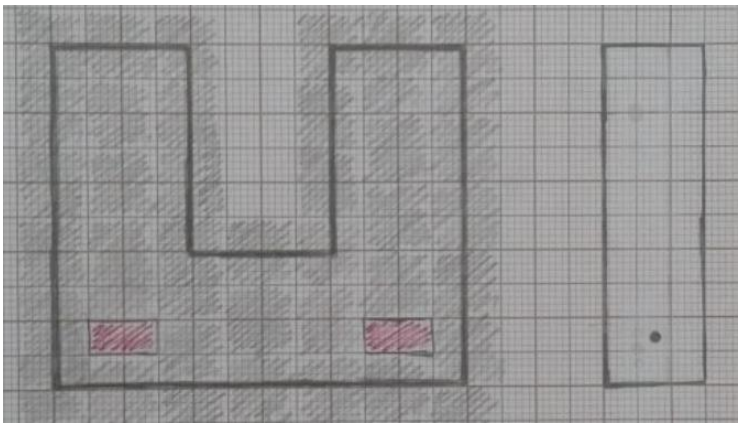


A)



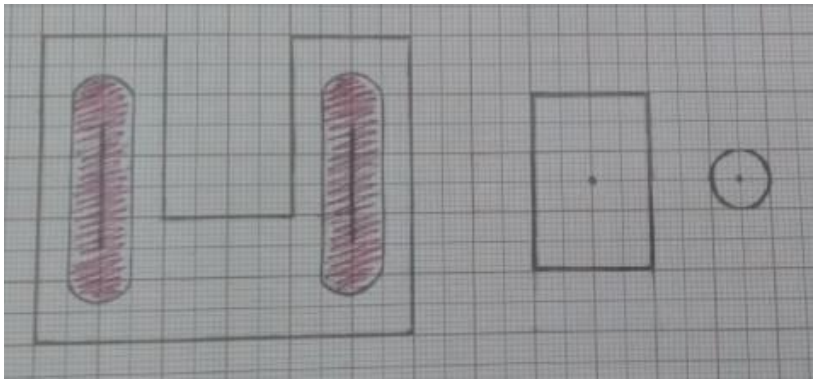
A operação morfológica utilizada é a EROSÃO.

B)



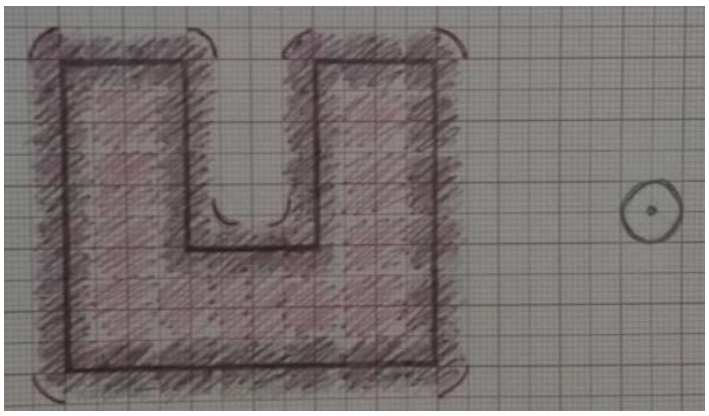
A operação morfológica utilizada é EROSÃO.

C)



A operação morfológica utilizada foi ABERTURA, é uma operação de erosão, seguida de dilatação.

D)



A operação morfológica utilizada foi a DILATAÇÃO.

Problema 1.2: Seja A o conjunto apresentado na figura abaixo. Considerando os elementos estruturantes da figura, esboce o resultado das seguintes operações morfológicas

a) $(A \ominus B^4) \oplus B^2$

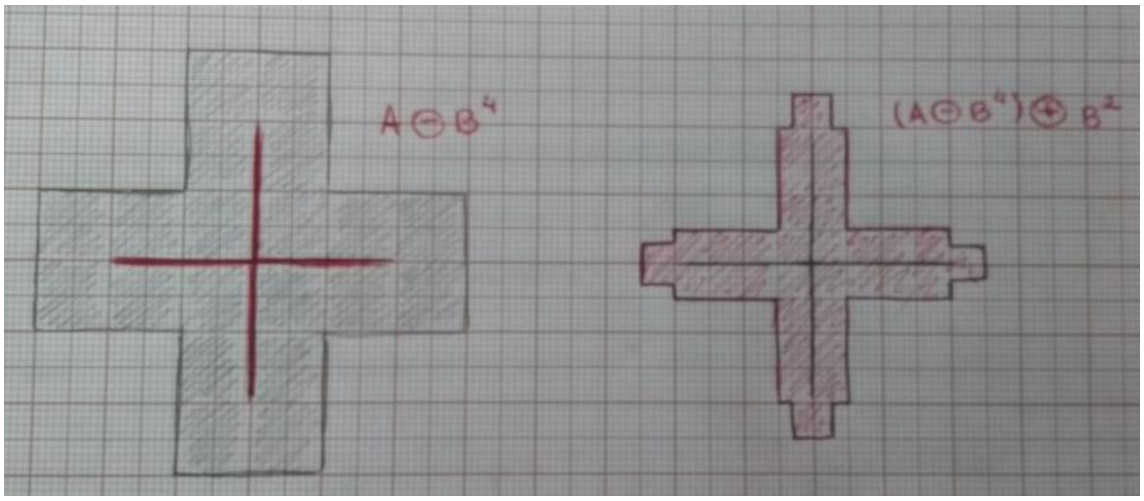
b) $(A \ominus B^1) \oplus B^3$

c) $(A \oplus B^1) \ominus B^3$

d) $(A \oplus B^3) \ominus B^2$

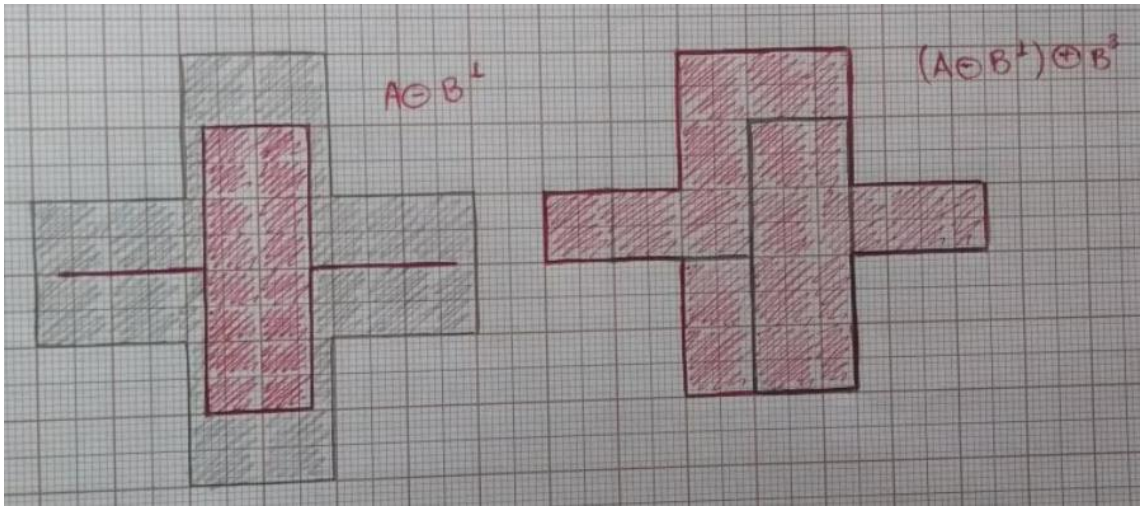
O resultado em vermelho é sempre a resposta da operação realizada.

A)



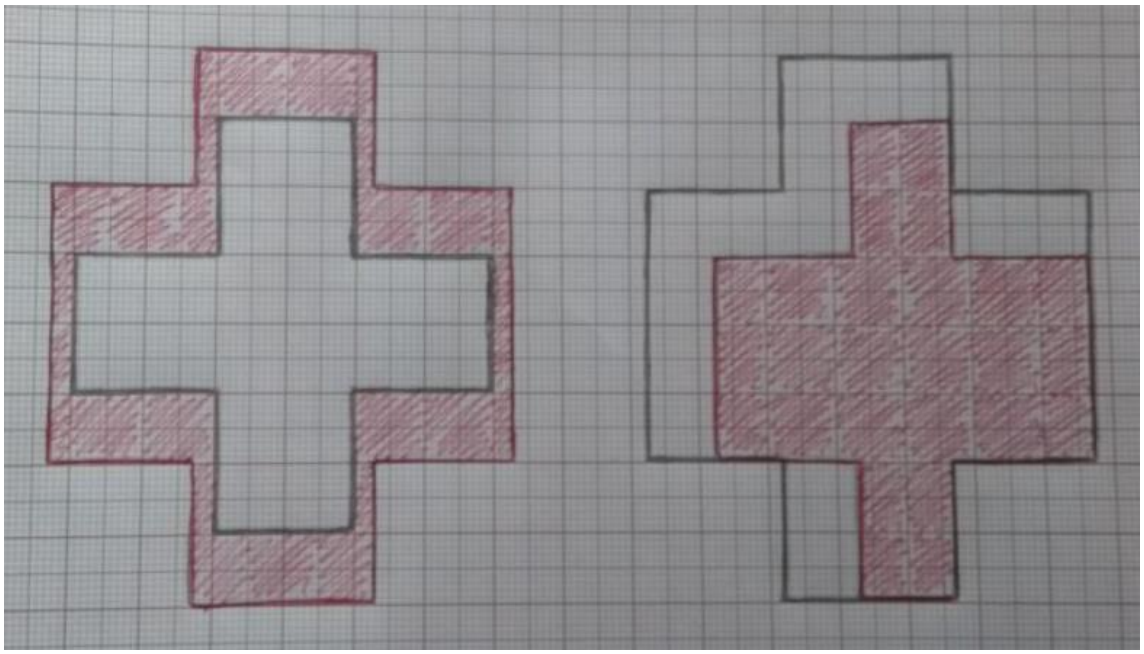
A primeira imagem mostra o resultado de uma EROSÃO entre a imagem A e o elemento estruturante B4 fornecido pelo professor. Já a segunda imagem resulta em uma DILATAÇÃO aplicada ao resultado da primeira imagem com o elemento estruturante B2.

B)



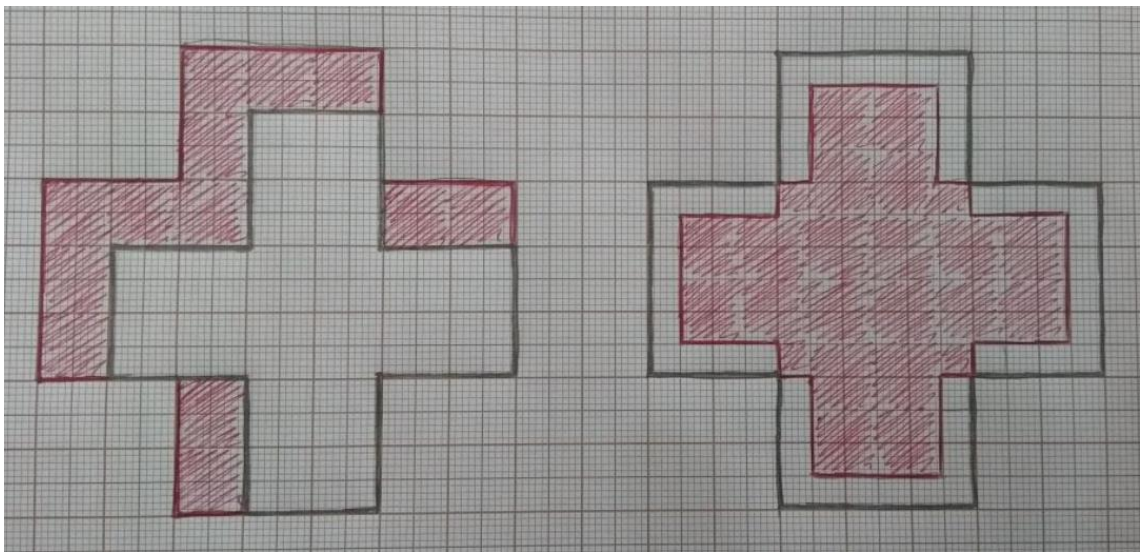
A primeira imagem mostra o resultado de uma EROSÃO entre a imagem A e o elemento estruturante B1 fornecido pelo professor. Já a segunda imagem resulta em uma DILATAÇÃO aplicada ao resultado da primeira imagem com o elemento estruturante B3.

C)



A primeira imagem mostra o resultado de uma DILATAÇÃO entre a imagem A e o elemento estruturante B1 fornecido pelo professor. Já a segunda imagem resulta em uma EROSÃO aplicada ao resultado da primeira imagem com o elemento estruturante B3.

D)



A primeira imagem mostra o resultado de uma DILATAÇÃO entre a imagem A e o elemento estruturante B3 fornecido pelo professor. Já a segunda imagem resulta em uma EROSÃO aplicada ao resultado da primeira imagem com o elemento estruturante B2.

Problema 1.3: Esboce o resultado da aplicação da operação hit or miss na imagem à esquerda, utilizando o elemento estruturante da imagem à direita. Forneça todas as imagens intermediárias na sua resposta. Compare os resultados com os obtidos com uma operação de correlação (ver livro texto), discutindo as diferenças e similaridades entre os dois casos.

Utilizando o “script1.3”:

```
import cv2

import Utils

imagemOrignal = Utils.leImagem("utk.tif")
elementoEstruturante = Utils.leImagem("eleEs.tif")

imagemOrignal = cv2.cvtColor(imagemOrignal, cv2.COLOR_RGB2GRAY)
elementoEstruturante = cv2.cvtColor(elementoEstruturante,
cv2.COLOR_RGB2GRAY)

ret, A = cv2.threshold(imagemOrignal, 170, 255, cv2.THRESH_BINARY)
ret2, D = cv2.threshold(elementoEstruturante, 170, 255,
cv2.THRESH_BINARY)

imagemComErocao = cv2.erode(A, D)

ret, Ac = cv2.threshold(imagemOrignal, 170, 255,
cv2.THRESH_BINARY_INV)
ret, Dc = cv2.threshold(elementoEstruturante, 170, 255,
cv2.THRESH_BINARY_INV)

imagemComErocao2 = cv2.erode(Ac,Dc)
```

```
imagemFinal = cv2.bitwise_and(imagemComErocao,imagemComErocao2)

lista = []
lista.append(imagemComErocao)
lista.append(imagemComErocao2)
lista.append(Ac)
lista.append(Dc)
lista.append(imagemFinal)
Utils.mostraListaDeImagens(lista)
```

Inicialmente as imagens são convertidas pra cinza e em seguida para binária para facilitar a manipulação das mesmas.

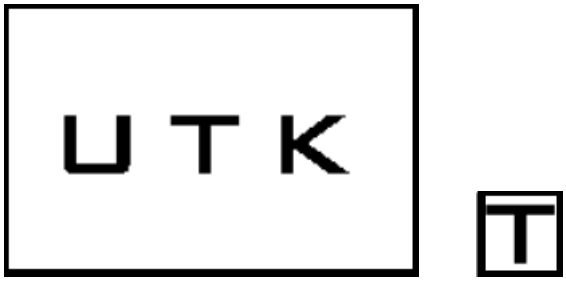
A transformada morfológica hit-or-miss é uma ferramenta básica para a detecção de formas. A aplicação da transformada é feita pela seguinte fórmula.

$$(A \ominus D) \cap (A^c \ominus [W - D])$$

1. Passo 1: faz a erosão entre a imagem original e o elemento estruturante.

Nota-se que com a utilização da erosão a letra “T” é encontrada na imagem.

2. Passo 2: faz o mesmo do passo 1, porém utilizando o complemento das imagens.

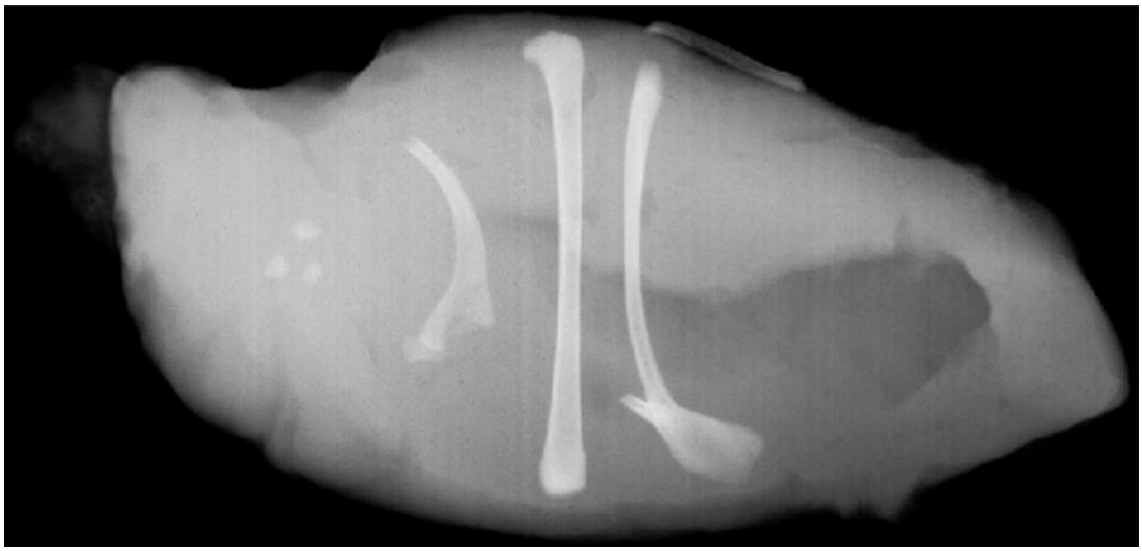


O resultado da erosão é ilustrado a seguir:

Para concluir a transformada morfológica hit-or-miss, faz uma operação and entre as duas imagens, lembrando que essa operação mantém a cor branca onde as duas imagens se coincidem, se não altera pra preto.

O resultado dessa operação nos torna a mesma imagem da primeira erosão feita, indicando que foi encontrado o elemento estruturante na imagem tanto com as imagens originais e o complemento das mesmas.

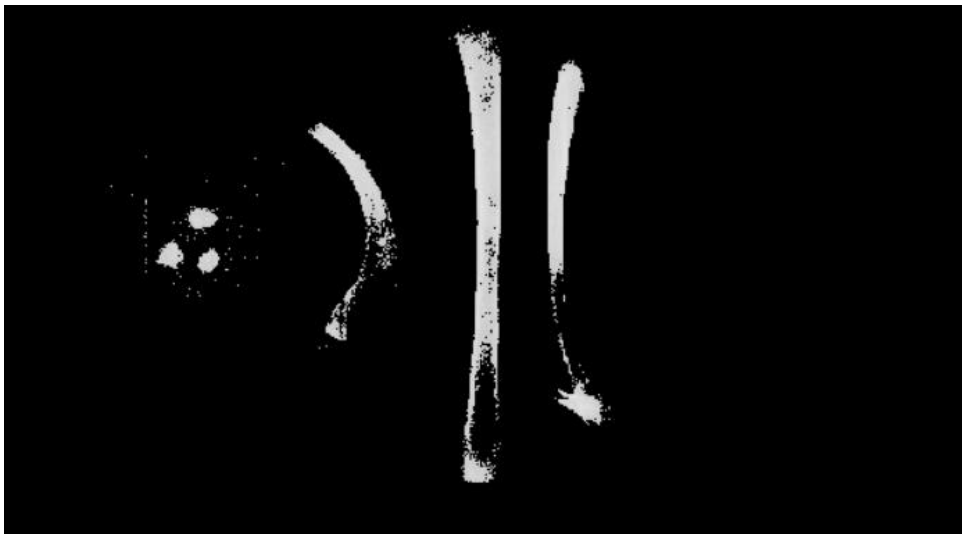
Problema 2.1: Escreva um programa capaz de extrair (e contar) as componentes conectadas de uma imagem binária. Repita os resultados do exemplo 9.7 (3ª edição internacional).



Utilizando o “script2.1”

```
import cv2
import numpy

import Utils

imagemOriginal = Utils.leImagem("frango.tif")
imagemRuidosa = cv2.cvtColor(imagemOriginal, cv2.COLOR_RGB2GRAY)

dimensoes = numpy.shape(imagemRuidosa)
linhas = dimensoes[0]
colunas = dimensoes[1]

for linha in range(linhas):
    for coluna in range(colunas):
        pixel = imagemRuidosa[linha, coluna]
        if (pixel >= 0 and pixel <= 200):
            imagemRuidosa[linha, coluna] = 0

kernel = numpy.ones((5, 5), numpy.uint8)
imagemErosao = cv2.erode(imagemRuidosa, kernel)
imageResultado, contours, hierarchy = cv2.findContours(imagemErosao,
cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
print("Contador: " + str(len(contours)))

lista = []
lista.append(imagemOriginal)
lista.append(imagemRuidosa)
lista.append(imageResultado)
Utils.mostraListaDeImagens(lista)
```

O algoritmo filtra os pixels com valor menor que 200, alterando os mesmos para zero (preto), destacando apenas os ossos da imagem.



Para poder contar os ossos é necessário retirar os pontos “brancos soltos” na imagem, pois os mesmo serão contados como grupos quando utilizado a conectividade por vizinhança. Esses “pontos brancos” são retirados utilizando erosão na imagem com o kernel 5x5 de “uns”.

Através da conectividade entre vizinha-8 de pixels são contados 15 grupos, devido alguns pontos brancos que ainda permaneceram soltos na imagem.

Problema 2.2: Escreva um programa capaz de contar a quantidade de dinheiro (moedas americanas) na imagem abaixo:

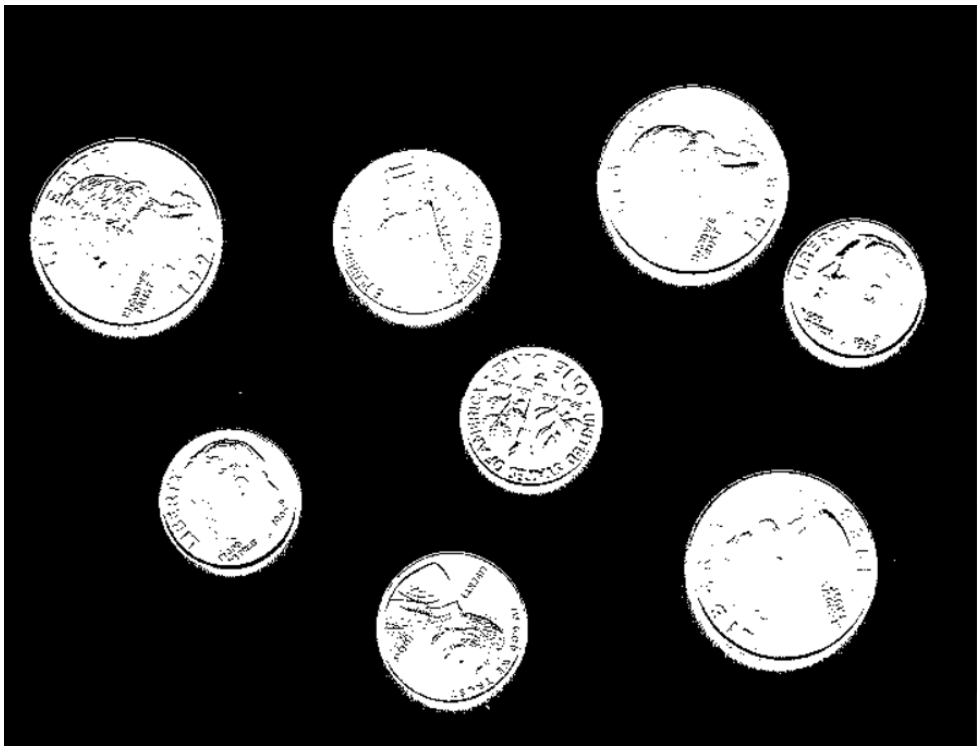
Utilizando o script2.2:

```
import cv2
import numpy

import Utils

imagemOriginal = Utils.leImagem("coins1.png")
imagemCinza = cv2.cvtColor(imagemOriginal, cv2.COLOR_RGB2GRAY)
ret, imagemBinaria = cv2.threshold(imagemCinza, 170, 255,
cv2.THRESH_BINARY_INV)

kernelDilatacao = numpy.ones((5, 5), numpy.uint8)
imagemComDilatacao = cv2.dilate(imagemBinaria, kernelDilatacao)

kernelErosao = numpy.ones((10, 10), numpy.uint8)
imagemComErocao = cv2.erode(imagemComDilatacao, kernelErosao)

imagemResultado = cv2.medianBlur(imagemComErocao, 5)
imagemResultado, contours, hierarchy =
cv2.findContours(imagemResultado, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

cents25 = 0
cents10 = 0
cents5 = 0
cents1 = 0
for i in range(8):
    tamanho = len(contours[i])
    if (tamanho >= 195 and tamanho <= 210):
        cents25 += 1
    elif (int(tamanho) >= 170 and int(tamanho) <= 180):
        cents10 += 1
```

```
    elif (int(tamanho) >= 150 and int(tamanho) <= 160):
        cents1 += 1
    elif (int(tamanho) >= 140 and int(tamanho) <= 149):
        cents5 += 1
    else:
        print("Tamanho de moeda nao definido")

print("25 cents = " + str(cents25))
print("10 cents = " + str(cents10))
print("5 cents = " + str(cents5))
print("1 cents = " + str(cents1))

moedas = [cents25, cents10, cents5, cents1]
Utils.calculaMoedas(moedas)

lista = []
lista.append(imagemOriginal)
lista.append(imagemCinza)
lista.append(imagemBinaria)
lista.append(imagemComDilatacao)
lista.append(imagemComErocao)
lista.append(imagemResultado)
Utils.mostraListaDeImagens(lista)
```

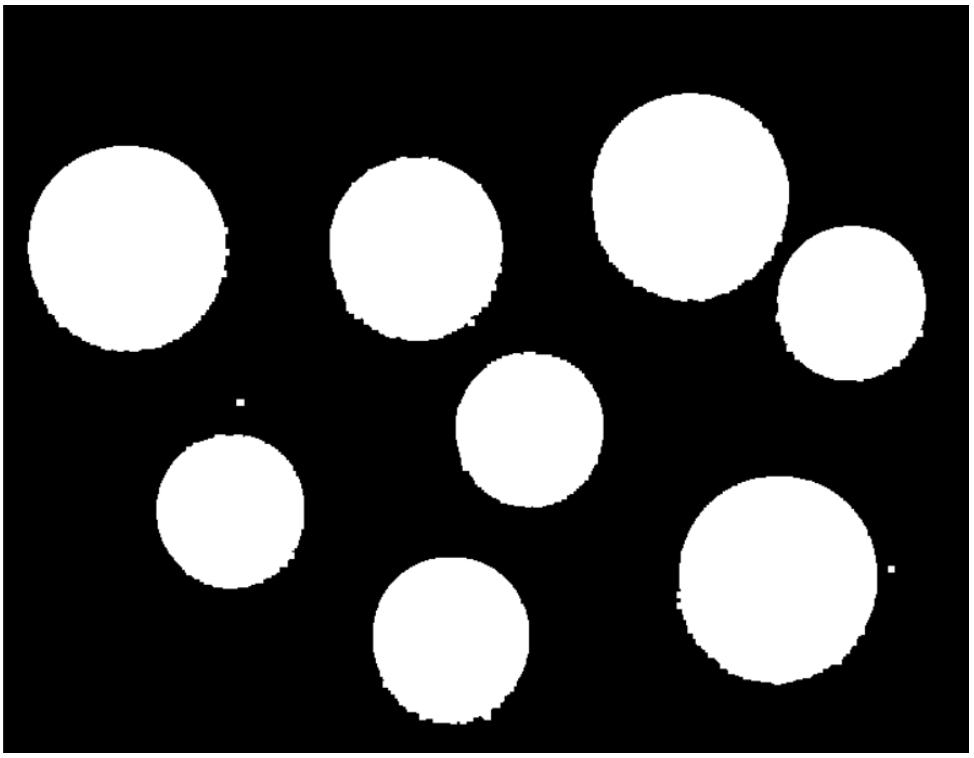
Para resolução do problema, inicialmente o algoritmo converte a imagem original para cinza, facilitando a manipulação da mesma, trabalhando apenas com a linha e coluna da imagem.

Em seguida a mesma é convertida para binário, retirando os tons de cinza e, assim trabalhando apenas com as cores preta e branca.

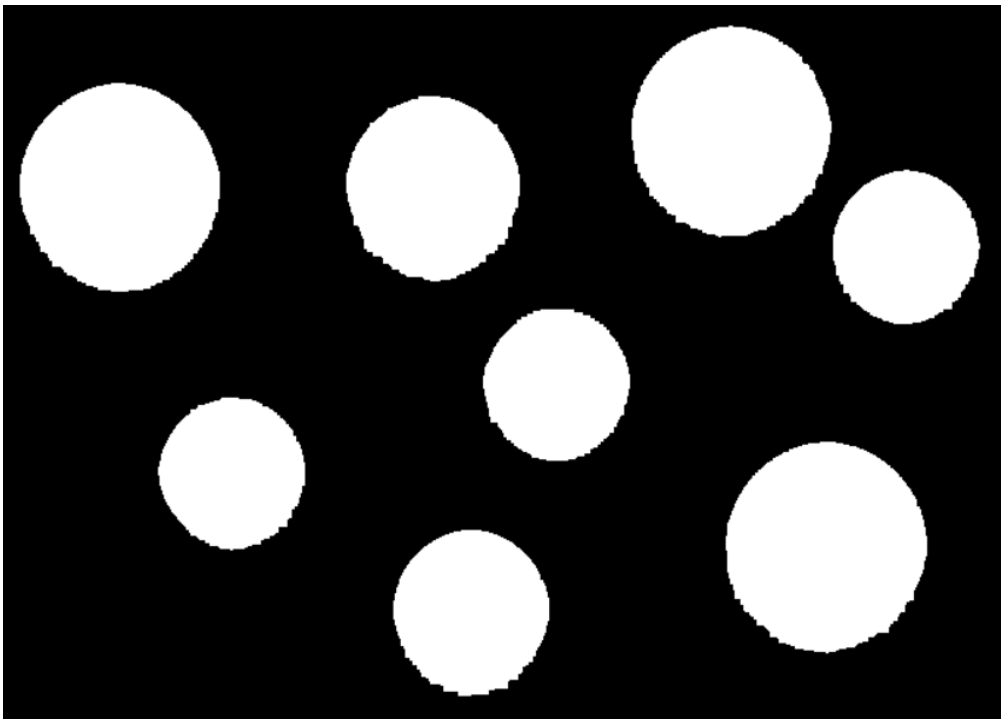


Nota-se que na imagem a cima demonstra vestígios da moeda o que pode interferir na conexão dos pixels para formar os grupos para reconhecê-las.

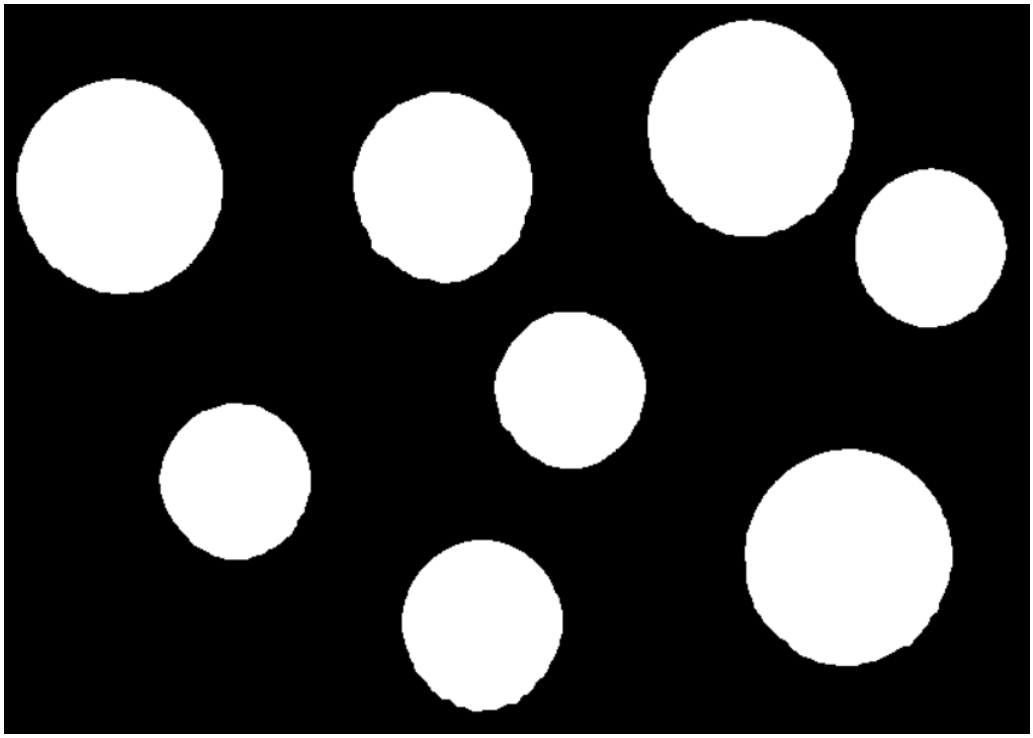
Para tirar os vestígios das mesmas foi aplica dilatação na imagem utilizando um elemento estruturante com kernel 5x5 formada de “uns”.



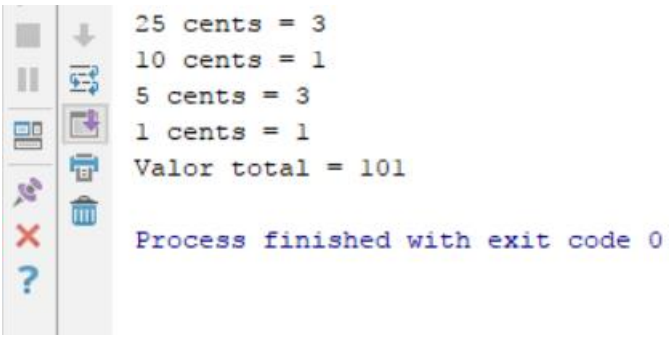
Aplica-se erosão com uma kernel 10x10 formada de “uns” para retirar os pontos brancos soltos na imagem, ou seja, que não fazem parte da moeda.



Para melhorar, “arredondando” os conjuntos de pixels que formam as moedas foi aplicado um filtro de mediana.



O algoritmo conta os grupos através de vizinhança-8 entre os pixels. No final do algoritmo fornece a contagem das moedas e o valor total delas.



Apêndice A – Utils

```
import cv2
import numpy


def mostraImagem(imagem):
    cv2.imshow('imagem', imagem)
    cv2.waitKey(0)
    pass


def leImagem(imagem):
    path = "imagens/"
    return cv2.imread(path + imagem)

def passaFiltroDeMedia(imagem, tamanho):
    return cv2.blur(imagem, (tamanho, tamanho))


def mostraListaDeImagens(list):
    posicao = 0
    for i in range(len(list)):
        cv2.imshow('Imagem ' + str(posicao), list[i])
        posicao += 1
    cv2.waitKey(0)
    pass


def calculaMoedas(moedas):
    soma = 0
    soma += moedas[0] * 25
    soma += moedas[1] * 10
    soma += moedas[2] * 5
    soma += moedas[3]
    print("Valor total = " + str(soma))
    pass
```