



## **PROCESSAMENTO DIGITAL DE IMAGENS**

**CAMPUS BIRIGUI**

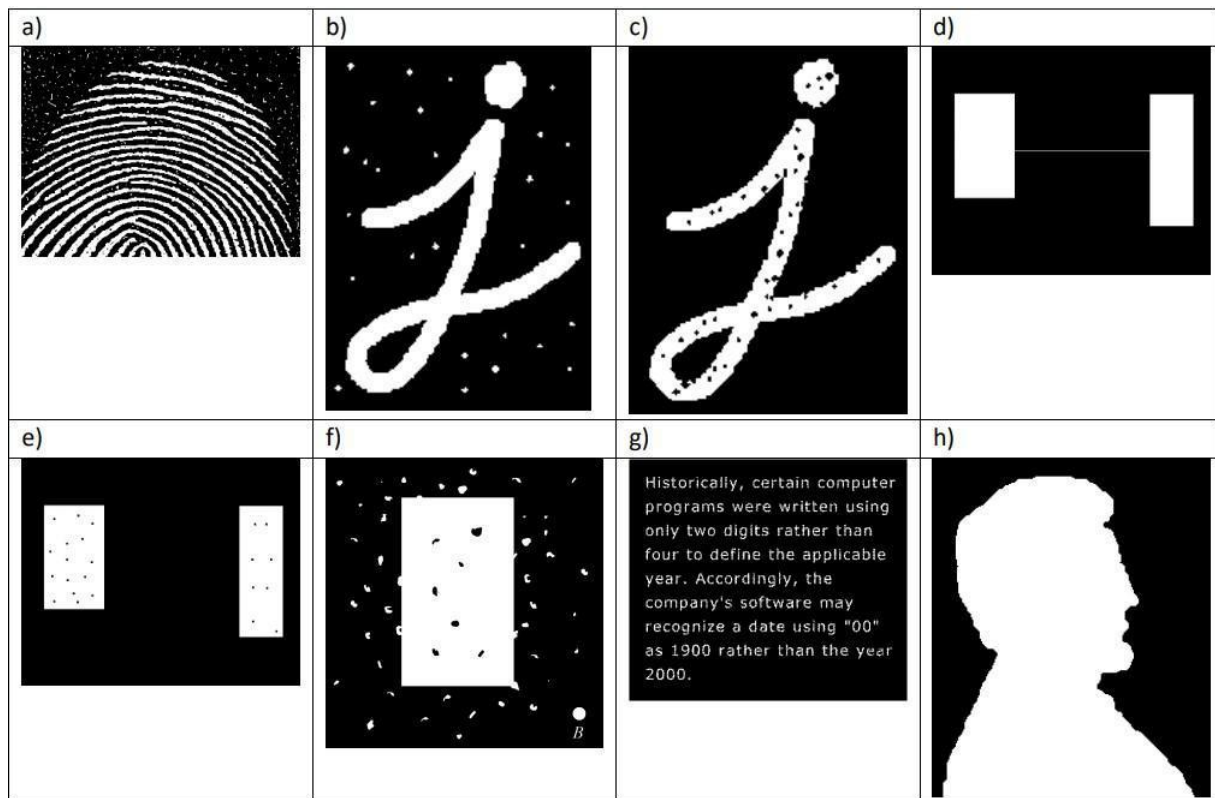
**MORFOLOGIA**

**Carlos Vinicius Baggio Savian**

**BI3002217**

**OUTUBRO DE 2023**

**1. IMPLEMENTE A EROÇÃO/DILATAÇÃO UTILIZANDO OS SEGUINTE ELEMENTOS ESTRUTURANTES E UTILIZE TODAS AS IMAGENS:**



*Figura 1: Imagens a serem usadas nos exercícios. Fonte: PDF do trabalho .*

**1.1. Erosão Código**

```
import numpy as np

import matplotlib.pyplot as

plt import cv2 as cv

img_fingerPrint = cv.imread('imgs/fingerprint.tif')

img_jay1 = cv.imread('imgs/Imagem1.tif')

img_jay2 = cv.imread('imgs/Imagem2.tif')

img_square1 = cv.imread('imgs/morfologia1.tif')

img_square2 = cv.imread('imgs/morfologia2.tif')

img_other =

cv.imread('imgs/noise_rectangle.tif') img_text =

cv.imread('imgs/text_gaps.tif') img_man =

cv.imread('imgs/rosto_perfil.tif')
```

```
# Estrutura 1
```

```
kernel_estrutura1 =
```

```
np.array((
```

```
    [0, 1, 0],
```

```
    [1, 1, 1],
```

```
    [0, 1, 0]
```

```
), np.uint8)
```

```
# Estrutura 2
```

```
kernel_estrutura2 = np.ones((3, 3), np.uint8)
```

```
# Estrutura 3
```

```
kernel_estrutura3 = np.ones((7, 1), np.uint8)
```

```
# Estrutura 4
```

```
kernel_estrutura4 =
```

```
np.array((
```

```
    [0, 0, 0, 1, 0, 0, 0],
```

```
    [0, 0, 1, 1, 1, 0, 0],
```

```
    [0, 1, 1, 1, 1, 1, 0],
```

```
    [1, 1, 1, 1, 1, 1, 1],
```

```
    [0, 1, 1, 1, 1, 1, 0],
```

```
    [0, 0, 1, 1, 1, 0, 0],
```

```
    [0, 0, 0, 1, 0, 0, 0]
```

```
), np.uint8)
```

```
# Apresentar na tela (Erosão)
```

```
fig, ax = plt.subplots(nrows = 2, ncols = 4)
```

```

ax[0, 0].imshow(cv.erode(img_fingerPrint, kernel_estrutura1, iterations =
1), cmap='gray')

ax[0, 0].set_title('Imagem 1')

ax[0, 1].imshow(cv.erode(img_jay1, kernel_estrutura1, iterations = 1),
cmap='gray') ax[0, 1].set_title('Imagem 2')

ax[0, 2].imshow(cv.erode(img_jay2, kernel_estrutura1, iterations = 1),
cmap='gray') ax[0, 2].set_title('Imagem 3')

ax[0, 3].imshow(cv.erode(img_square1, kernel_estrutura1, iterations =
1), cmap='gray')

ax[0, 3].set_title('Imagem 4')

ax[1, 0].imshow(cv.erode(img_square2, kernel_estrutura1, iterations =
1), cmap='gray')

ax[1, 0].set_title('Imagem 5')

ax[1, 1].imshow(cv.erode(img_other, kernel_estrutura1, iterations = 1),
cmap='gray') ax[1, 1].set_title('Imagem 6')

ax[1, 2].imshow(cv.erode(img_text, kernel_estrutura1, iterations = 1),
cmap='gray') ax[1, 2].set_title('Imagem 7')

ax[1, 3].imshow(cv.erode(img_man, kernel_estrutura1, iterations = 1),
cmap='gray') ax[1, 3].set_title('Imagem 8')

plt.show()

```

## 1.2. Dilatação Código

```

import numpy as np

import matplotlib.pyplot as

plt import cv2 as cv

img_fingerPrint = cv.imread('imgs/fingerprint.tif')

```

```
img_jay1 = cv.imread('imgs/Imagem1.tif')
img_jay2 = cv.imread('imgs/Imagem2.tif')
img_square1 = cv.imread('imgs/morfologia1.tif')
img_square2 = cv.imread('imgs/morfologia2.tif')
img_other =
cv.imread('imgs/noise_rectangle.tif') img_text =
cv.imread('imgs/text_gaps.tif') img_man =
cv.imread('imgs/rosto_perfil.tif')
```

```
# Estrutura 1
```

```
kernel_estrutura1 =
```

```
np.array((
```

```
    [0, 1, 0],
```

```
    [1, 1, 1],
```

```
    [0, 1, 0]
```

```
), np.uint8)
```

```
# Estrutura 2
```

```
kernel_estrutura2 = np.ones((3, 3), np.uint8)
```

```
# Estrutura 3
```

```
kernel_estrutura3 = np.ones((7, 1), np.uint8)
```

```
# Estrutura 4
```

```
kernel_estrutura4 =
```

```
np.array((
```

```
    [0, 0, 0, 1, 0, 0, 0],
```

```
    [0, 0, 1, 1, 1, 0, 0],
```

[0, 1, 1, 1, 1, 1, 0],

[1, 1, 1, 1, 1, 1, 1],

```

[0, 1, 1, 1, 1, 1, 0],
[0, 0, 1, 1, 1, 0, 0],
[0, 0, 0, 1, 0, 0, 0]
), np.uint8)

# Apresentar na tela (Dilatação)

fig, ax = plt.subplots(nrows = 2, ncols = 4)

ax[0, 0].imshow(cv.dilate(img_fingerPrint, kernel_estrutura1, iterations =
1), cmap='gray')
ax[0, 0].set_title('Imagem 1')
ax[0, 1].imshow(cv.dilate(img_jay1, kernel_estrutura1, iterations = 1),
cmap='gray') ax[0, 1].set_title('Imagem 2')
ax[0, 2].imshow(cv.dilate(img_jay2, kernel_estrutura1, iterations = 1),
cmap='gray') ax[0, 2].set_title('Imagem 3')
ax[0, 3].imshow(cv.dilate(img_square1, kernel_estrutura1, iterations =
1), cmap='gray')
ax[0, 3].set_title('Imagem 4')
ax[1, 0].imshow(cv.dilate(img_square2, kernel_estrutura1, iterations =
1), cmap='gray')
ax[1, 0].set_title('Imagem 5')
ax[1, 1].imshow(cv.dilate(img_other, kernel_estrutura1, iterations = 1),
cmap='gray') ax[1, 1].set_title('Imagem 6')
ax[1, 2].imshow(cv.dilate(img_text, kernel_estrutura1, iterations = 1),
cmap='gray') ax[1, 2].set_title('Imagem 7')
ax[1, 3].imshow(cv.dilate(img_man, kernel_estrutura1, iterations = 1),
cmap='gray') ax[1, 3].set_title('Imagem 8')
plt.show()

```



### 1.3. Resultado com a primeira estrutura

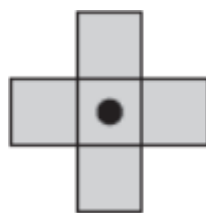


Figura 2: Estrutura 1. Fonte: Elaboração própria.

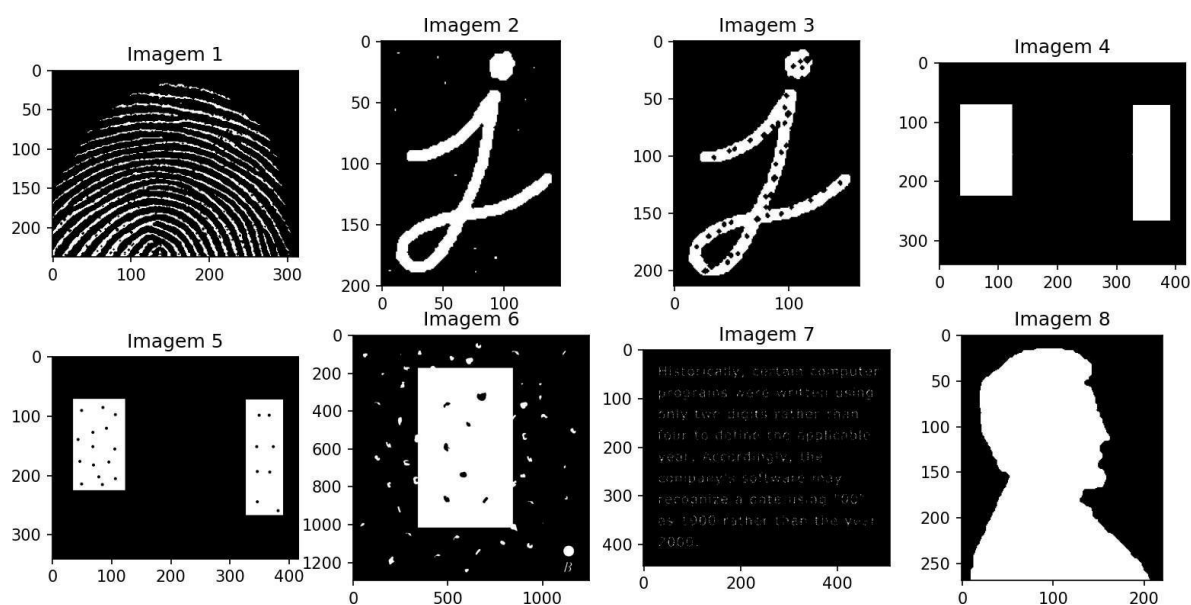


Figura 3: Imagens com o efeito de erosão (Estrutura 1). Fonte: Elaboração própria.

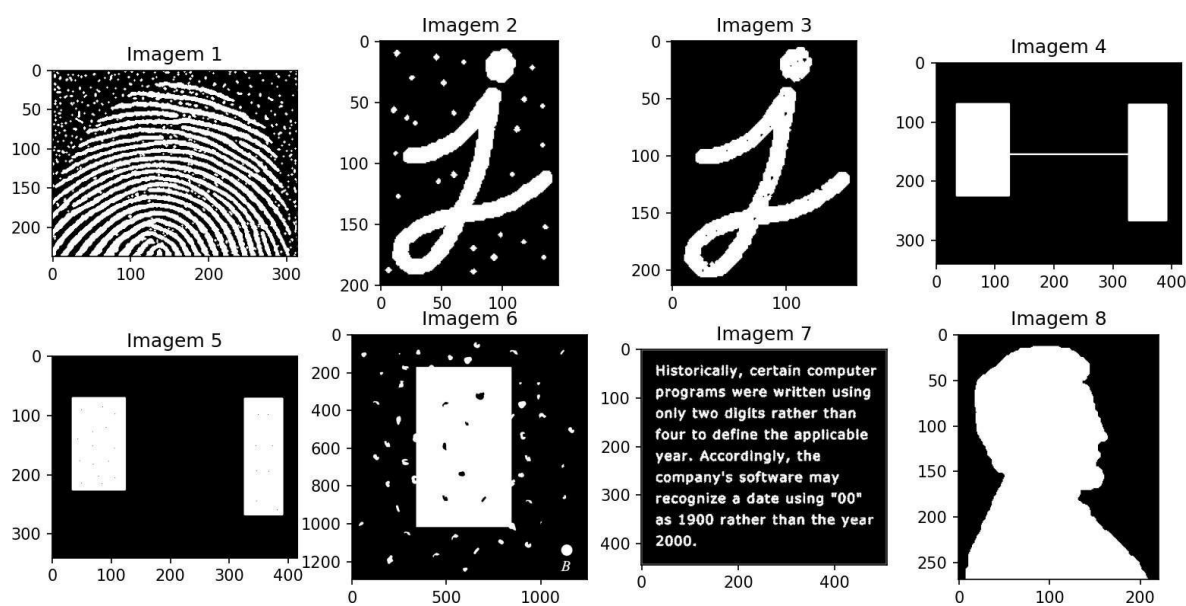


Figura 4: Imagens com o efeito de dilatação (Estrutura 1). Fonte: Elaboração própria.

#### 1.4. Resultado com a segunda estrutura

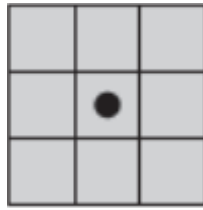


Figura 5: Estrutura 2. Fonte: Elaboração própria.

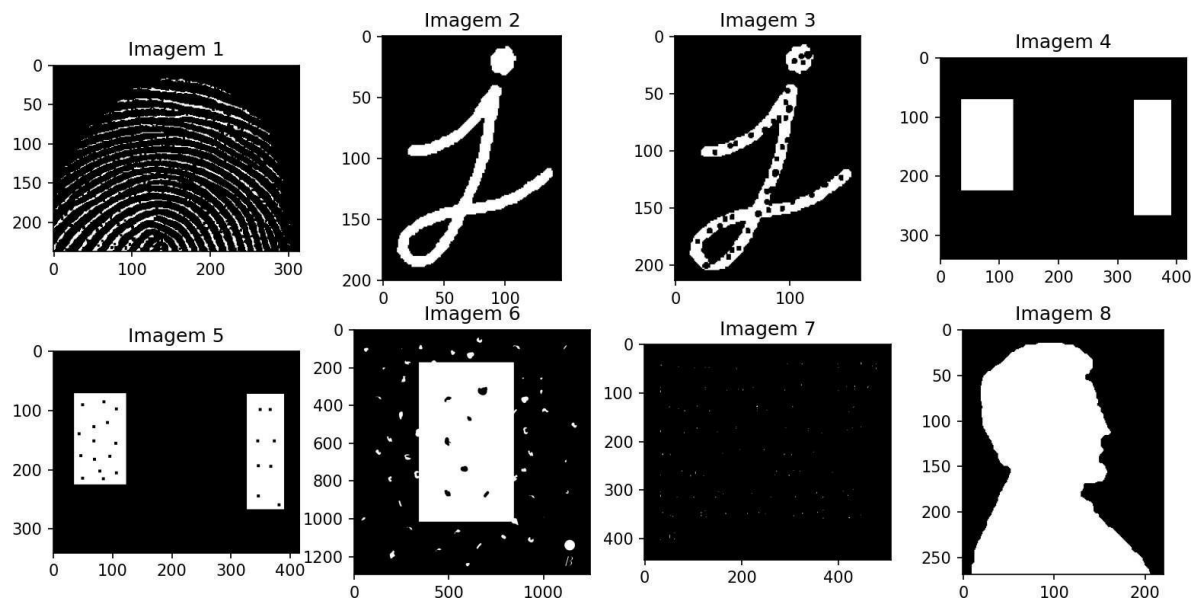


Figura 6: Imagens com o efeito de erosão (Estrutura 2). Fonte: Elaboração própria.

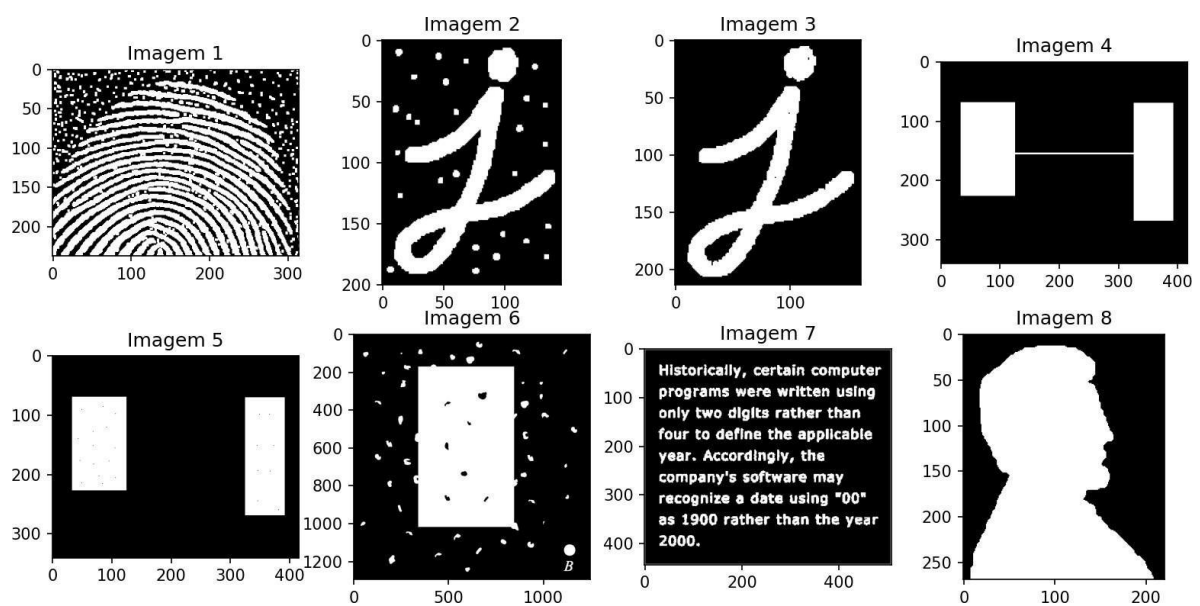


Figura 7: Imagens com o efeito de dilatação (Estrutura 2). Fonte: Elaboração própria.

## 1.5. Resultado com a terceira estrutura



Figura 8: Estrutura 3. Fonte: Elaboração própria.

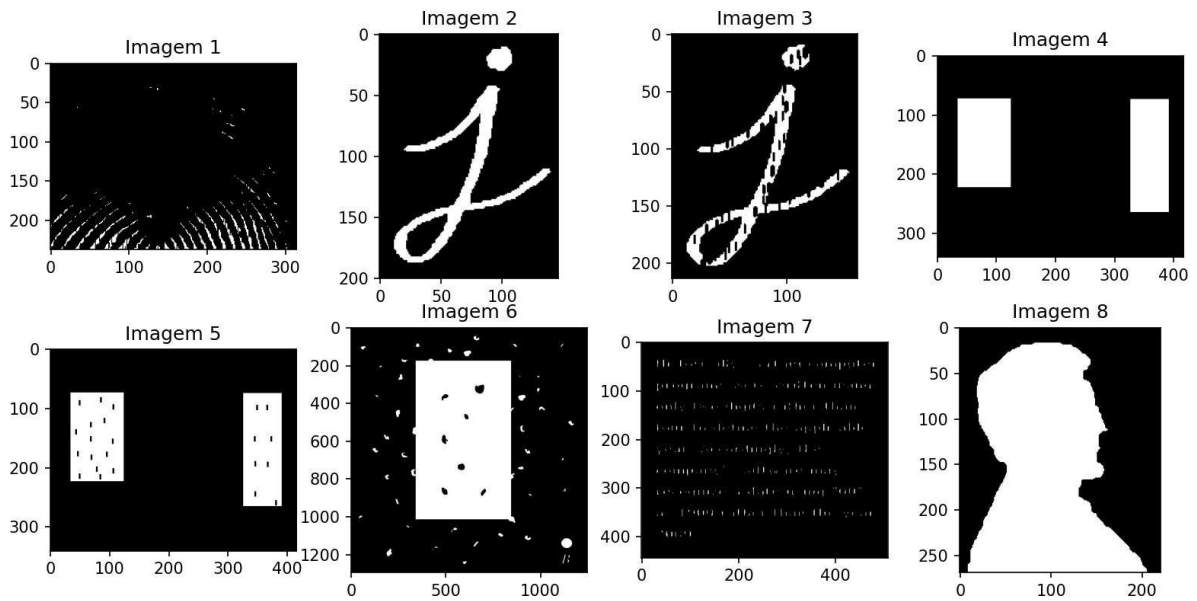


Figura 9: Imagens com o efeito de erosão (Estrutura 3). Fonte: Elaboração própria.

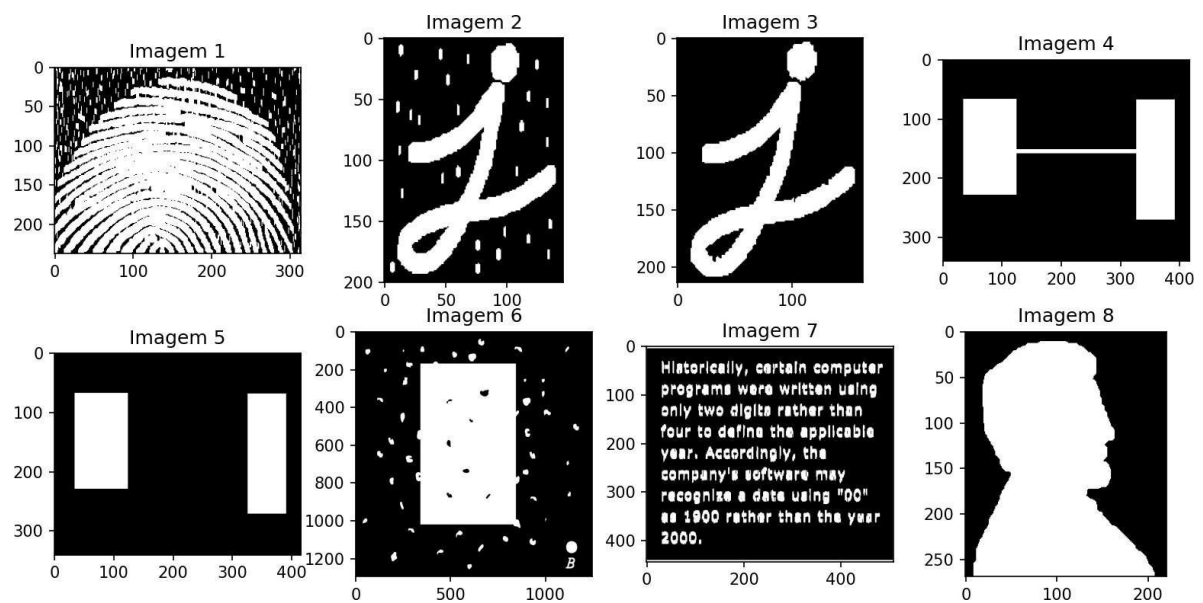


Figura 10: Imagens com o efeito de dilatação (Estrutura 3). Fonte: Elaboração própria.

## 1.6. Resultado com a quarta estrutura

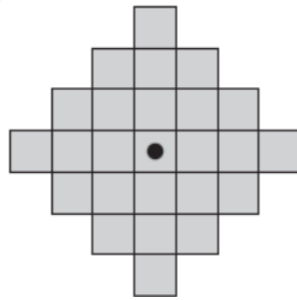


Figura 11: Estrutura 4. Fonte: Elaboração própria.

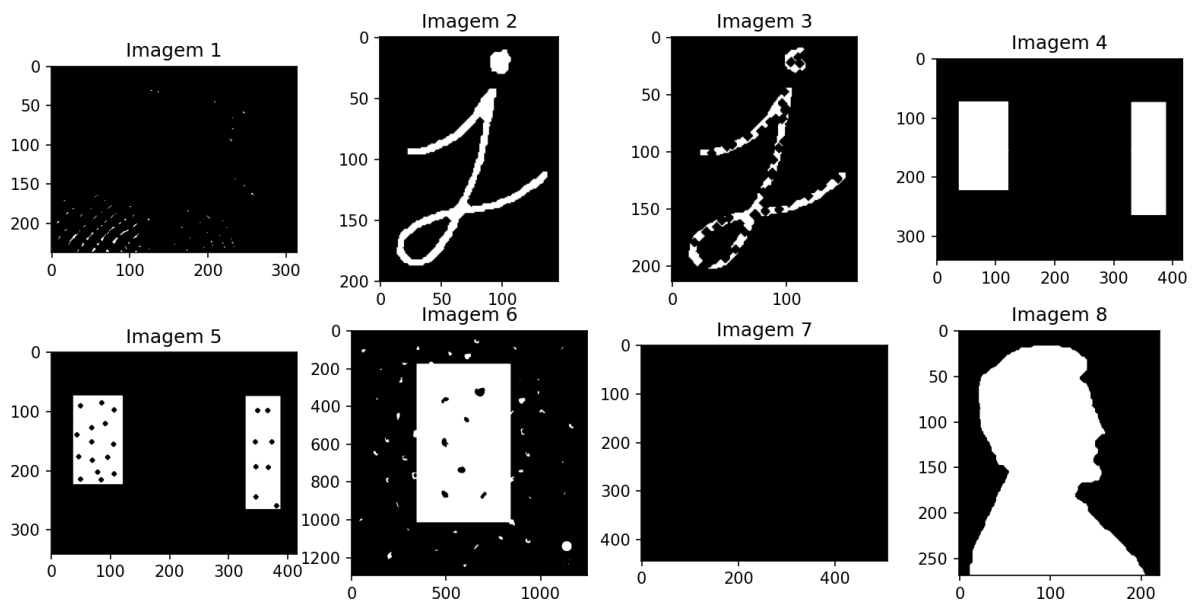


Figura 12: Imagens com o efeito de erosão (Estrutura 4). Fonte: Elaboração própria.

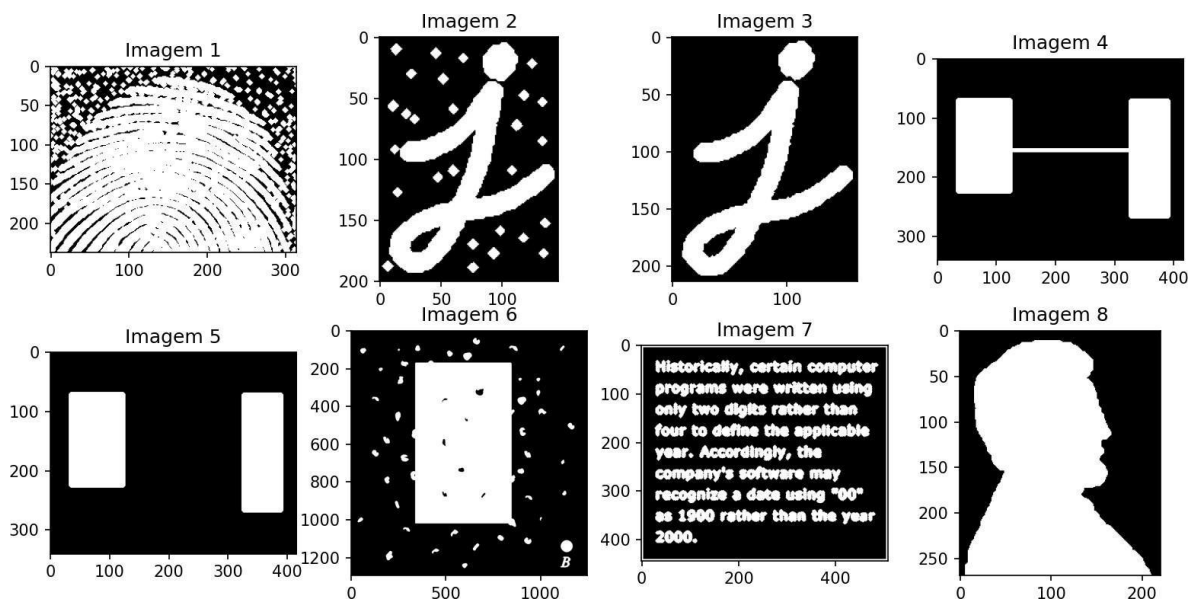


Figura 13: Imagens com o efeito de dilatação (Estrutura 4). Fonte: Elaboração própria.

2. **Implemente as operações de abertura e fechamento utilizando apenas o primeiro elemento estruturante do exercício acima. Considerando as imagens de b) a e) quais imagens seria mais interessante utilizar a abertura e quais o fechamento para remover os ruídos?**

**2.1. Código**

```
import numpy as np

import matplotlib.pyplot as

plt import cv2 as cv

def abertura(img, kernel):

    img = cv.erode(img, kernel, iterations =

    1) img = cv.dilate(img, kernel, iterations =

    1) return img

def fechamento(img, kernel):

    img = cv.dilate(img, kernel, iterations = 1)

    img = cv.erode(img, kernel, iterations =

    1) return img

img_fingerPrint = cv.imread('imgs/fingerprint.tif')

img_jay1 = cv.imread('imgs/Imagem1.tif')

img_jay2 = cv.imread('imgs/Imagem2.tif')

img_square1 = cv.imread('imgs/morfologia1.tif')

img_square2 = cv.imread('imgs/morfologia2.tif')

img_other =

cv.imread('imgs/noise_rectangle.tif') img_text =

cv.imread('imgs/text_gaps.tif') img_man =

cv.imread('imgs/rosto_perfil.tif')
```

```

kernel_estrutura1 =
    np.array(( [0, 1, 0],
               [1, 1, 1],
               [0, 1, 0]
    ), np.uint8)

# Apresentar na tela (Abertura)

fig, ax = plt.subplots(nrows = 2, ncols = 4)

ax[0, 0].imshow(abertura(img_fingerPrint, kernel_estrutura1), cmap='gray')
ax[0, 0].set_title('Imagem 1')
ax[0, 1].imshow(abertura(img_jay1, kernel_estrutura1),
cmap='gray') ax[0, 1].set_title('Imagem 2')
ax[0, 2].imshow(abertura(img_jay2, kernel_estrutura1),
cmap='gray') ax[0, 2].set_title('Imagem 3')
ax[0, 3].imshow(abertura(img_square1, kernel_estrutura1), cmap='gray')
ax[0, 3].set_title('Imagem 4')
ax[1, 0].imshow(abertura(img_square2, kernel_estrutura1), cmap='gray')
ax[1, 0].set_title('Imagem 5')
ax[1, 1].imshow(abertura(img_other, kernel_estrutura1),
cmap='gray') ax[1, 1].set_title('Imagem 6')
ax[1, 2].imshow(abertura(img_text, kernel_estrutura1), cmap='gray')
ax[1, 2].set_title('Imagem 7')
ax[1, 3].imshow(abertura(img_man, kernel_estrutura1),
cmap='gray') ax[1, 3].set_title('Imagem 8')

plt.show()

```

```
# Apresentar na tela (Fechamento)

# fig, ax = plt.subplots(nrows = 2, ncols = 4)

# ax[0, 0].imshow(fechamento(img_fingerPrint, kernel_estrutura1), cmap='gray')
# ax[0, 0].set_title('Imagem 1')
# ax[0, 1].imshow(fechamento(img_jay1, kernel_estrutura1),
cmap='gray') # ax[0, 1].set_title('Imagem 2')
# ax[0, 2].imshow(fechamento(img_jay2, kernel_estrutura1),
cmap='gray') # ax[0, 2].set_title('Imagem 3')
# ax[0, 3].imshow(fechamento(img_square1, kernel_estrutura1), cmap='gray')
# ax[0, 3].set_title('Imagem 4')
# ax[1, 0].imshow(fechamento(img_square2, kernel_estrutura1), cmap='gray')
# ax[1, 0].set_title('Imagem 5')
# ax[1, 1].imshow(fechamento(img_other, kernel_estrutura1),
cmap='gray') # ax[1, 1].set_title('Imagem 6')
# ax[1, 2].imshow(fechamento(img_text, kernel_estrutura1), cmap='gray')
# ax[1, 2].set_title('Imagem 7')
# ax[1, 3].imshow(fechamento(img_man, kernel_estrutura1),
cmap='gray') # ax[1, 3].set_title('Imagem 8')

# plt.show()
```

## 2.2. Imagens

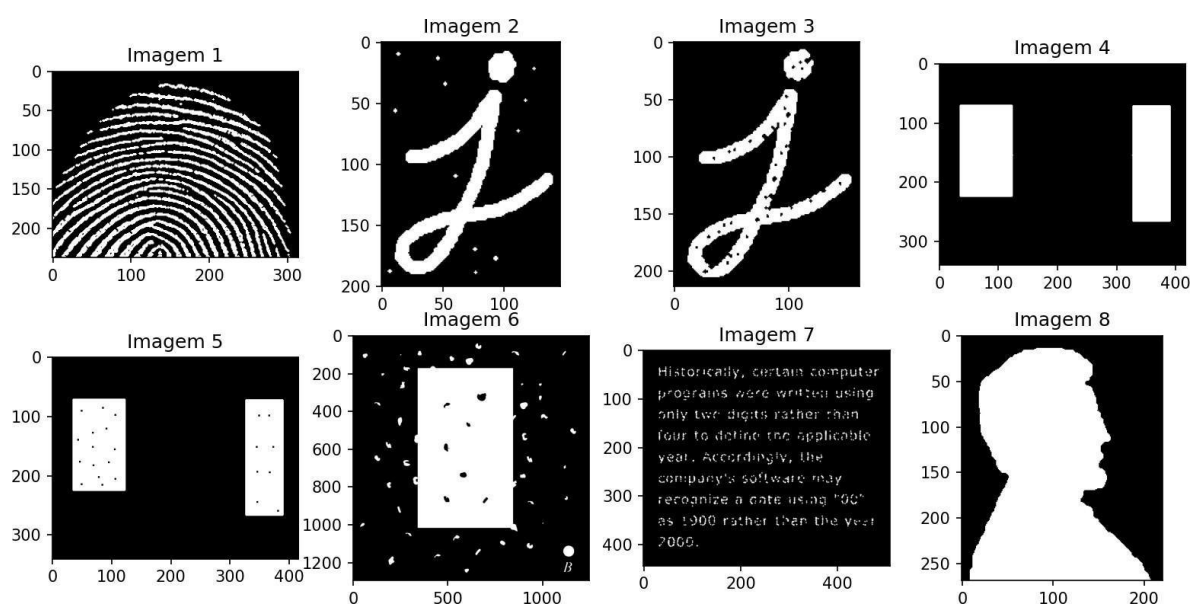


Figura 14: Imagens com o efeito de abertura (Estrutura 1). Fonte: Elaboração própria.

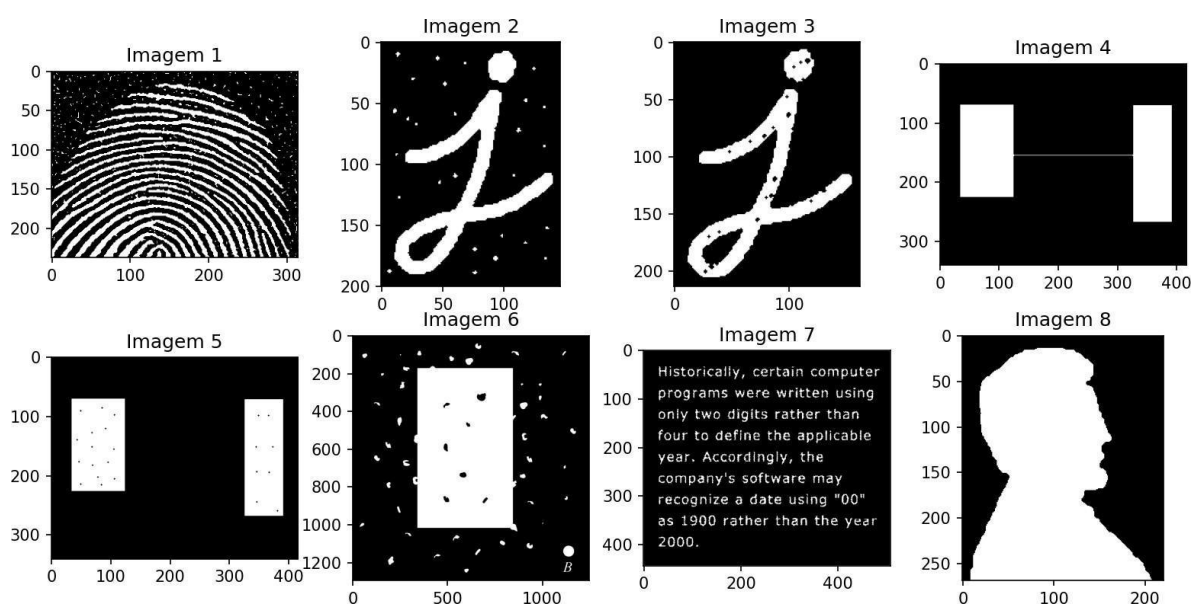


Figura 15: Imagens com o efeito de fechamento (Estrutura 1). Fonte: Elaboração própria.

De acordo com os resultados das aplicações dos efeitos de abertura e fechamento apresentados nas figuras 14 e 15, seria mais interessante utilizar o efeito de abertura nas imagens *b* e *d*. E nas imagens *c* e *e*, seria mais interessante utilizar o efeito de fechamento para ajudar a remover os ruídos.



**3. Qual sequência de operações poderiam ser realizadas para que a imagem f) ficasse apenas com um retângulo branco ao centro? Implemente essas operações.**

Para isso é necessário seguir 3 passos:

- Primeiro: é necessário aplicar a erosão 10 *vezes* com o kernel em  $1 \times 7$ ;
- Segundo: é necessário aplicar a erosão 5 *vezes* com o kernel em  $7 \times 1$ ;
- Terceiro: é necessário aplicar a dilatação 10 *vezes* com o kernel em  $9 \times 9$ .

**3.1. Código**

# Exercício 3

```
kernel_estrutura7emPe = np.ones((1, 7), np.uint8)
```

```
kernel_estrutura7deitado = np.ones((7, 1), np.uint8)
```

```
kernel_estrutura9 = np.ones((9, 9), np.uint8)
```

```
img1 = cv.erode(img_other, kernel_estrutura7emPe, iterations =
```

```
10) img2 = cv.erode(img1, kernel_estrutura7deitado, iterations = 5)
```

```
img3 = cv.dilate(img2, kernel_estrutura9, iterations = 10)
```

# Apresentar na tela (Abertura)

```
fig, ax = plt.subplots(nrows = 1, ncols = 3)
```

```
ax[0].imshow(img1,
```

```
cmap='gray')
```

```
ax[0].set_title('Imagem 1')
```

```
ax[1].imshow(img2,
```

```
cmap='gray')
```

```
ax[1].set_title('Imagem 2')
```

```
ax[2].imshow(img3,
```

```
cmap='gray')
```

```
ax[2].set_title('Imagem 3')
```

```
plt.show()
```

### 3.2. Resultado

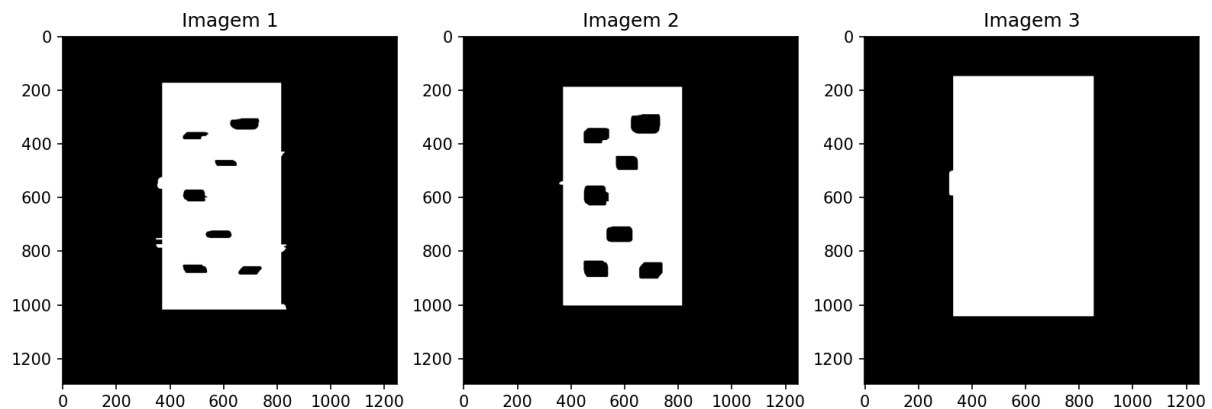


Figura 16: Imagem sem ruído com retângulo no centro. Fonte: Elaboração própria.

4. Qual(is) operações seriam necessárias para melhorar a imagem g)?  
Implemente essa(s) operação(ões).

#### 4.1. Código

```
kernel_estrutura3 = np.ones((3, 2), np.uint8)

cv.imshow('Imagem texto melhor', cv.dilate(img_text, kernel_estrutura3, iterations =
1))

cv.waitKey(0)
```

#### 4.2. Resultado

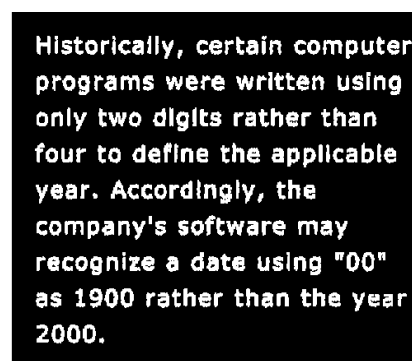


Figura 17: Imagem texto de forma melhorada. Fonte: Elaboração própria.

5. Quais operações seriam necessárias para extrair apenas a borda da imagem h)? Implemente essas operações.

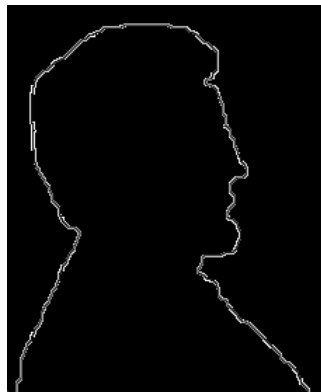
Com a imagem original menos a imagem com somente uma dilatação e o *kernel* igual a 3, é possível obter a borda da imagem *h*.

### 5.1. Código

```
kernel_estrutura2 = np.ones((3, 3), np.uint8)
```

```
cv.imshow('Imagem H com borda', img_man - cv.dilate(img_man, kernel_estrutura2,  
iterations = 1))
```

### 5.2. Resultado



*Figura 18: Imagem perfil com somente a borda. Fonte: Elaboração própria.*