

# Limiarizacao e Operacoes Morfologicas

August 30, 2018

## 1 Limiarização de Imagens

### 1.1 Objetivos

- Aprender sobre limiarização simples e adaptativa.
- Utilizar as funções: `cv2.threshold` e `cv2.adaptiveThreshold`.

### 1.2 Teoria

A Limiarização é uma das técnicas mais simples de segmentação e consiste na classificação dos pixels de uma imagem de acordo com a especificação de um ou mais limiares, conhecidos como thresholds. É também chamada de binarização, pois a imagem resultante possui apenas dois valores de intensidade: 0 (preto) ou 1 (branco).


#### 1.2.1 1. Limiarização Simples

O método é realmente simples. Se o valor do pixel na imagem é maior que um valor de threshold, é atribuído a ele um valor (pode ser branco), caso contrário, é atribuído outro valor (pode ser preto). A função usada é `cv2.threshold`. O primeiro parâmetro é a imagem original, que deve ser em tons de cinza. O segundo parâmetro é um valor de threshold que é usado para classificar os valores do pixel. O terceiro parâmetro é um maxVal, que representa o valor a ser dado se o valor do pixel é maior que o valor do threshold. O OpenCV oferece diferentes estilos de limiarização, que é o quarto parâmetro da função (que podem ser melhor estudados na documentação). Os tipos são:

- `cv2.THRESH_BINARY`
- `cv2.THRESH_BINARY_INV`
- `cv2.THRESH_TRUNC`
- `cv2.THRESH_TOZERO`
- `cv2.THRESH_TOZERO_INV`

```
In [8]: import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('gradiente.jpg',0)


```

```

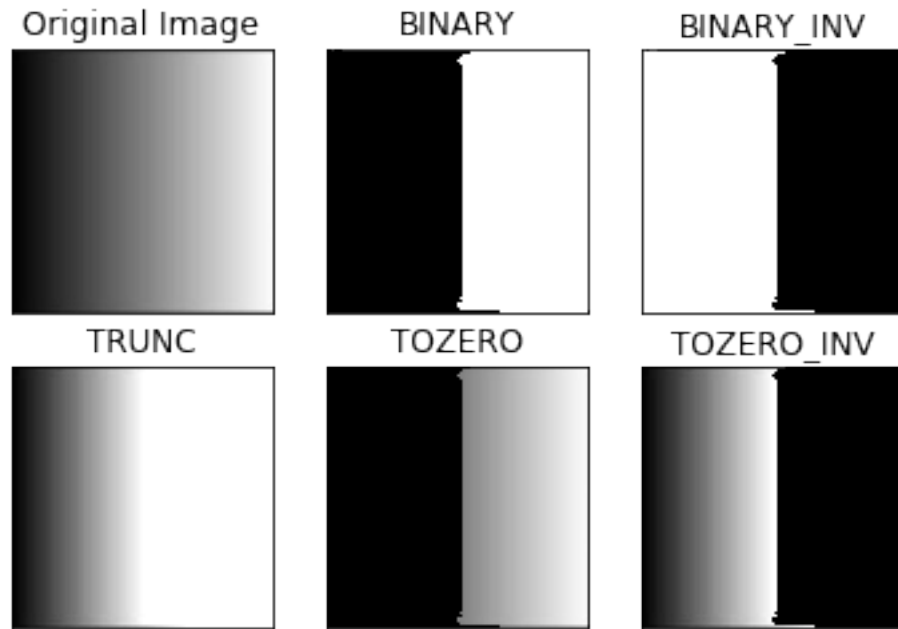
ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
ret,thresh2 = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
ret,thresh3 = cv2.threshold(img,127,255,cv2.THRESH_TRUNC)
ret,thresh4 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO)
ret,thresh5 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO_INV)

titles = ['Original Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]

for i in range(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([],plt.yticks([]))

plt.show()

```



### 1.2.2 2. Limiarização Adaptativa

No método simples, usamos um valor global como valor limite. Mas pode não ser bom quando a imagem tem diferentes condições de iluminação em diferentes áreas. Nesse caso, o limiar adaptativo apresenta melhores resultados. O algoritmo calcula o threshold para pequenas regiões da imagem. Assim, obtemos limiares diferentes para diferentes regiões da mesma imagem e isso nos dá melhores resultados para imagens com iluminação variável.

A limiarização adaptativa tem três parâmetros de entrada “especiais” e apenas um argumento de saída.

Método Adaptativo - Decide como o valor de threshold é calculado.

- cv2.ADAPTIVE\_THRESH\_MEAN\_C: valor limite é a média da área de vizinhança.
- cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C: valor limite é a soma ponderada dos valores de vizinhança onde os pesos são uma janela gaussiana.

Tamanho do bloco - Decide o tamanho da área do bairro.

C - É apenas uma constante que é subtraída da média ou média ponderada calculada.

Veja o código:

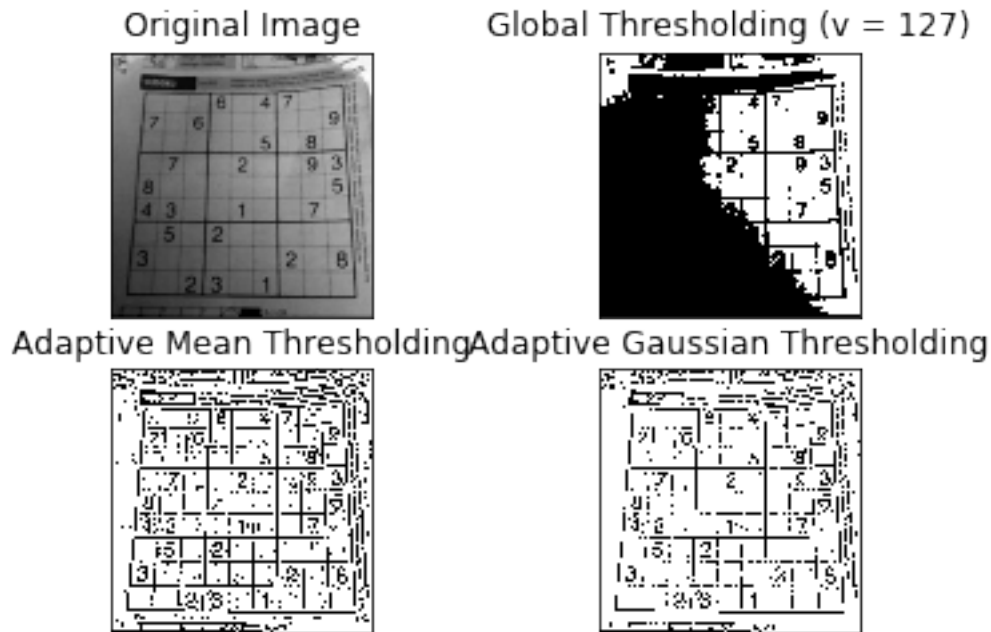
```
In [9]: import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('dave.jpg',0)
img = cv2.medianBlur(img,5)

ret,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,\
    cv2.THRESH_BINARY,11,2)
th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\
    cv2.THRESH_BINARY,11,2)

titles = ['Original Image', 'Global Thresholding (v = 127)',
    'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']
images = [img, th1, th2, th3]

for i in range(4):
    plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([],plt.yticks([]))
plt.show()
```



## 2 Transformações Morfológicas

### 2.1 Objetivos

- Aprender diferentes operações morfológicas como Erosão, Dilatação, Abertura e Fechamento.
- Utilizar as funções: `cv2.erode()`, `cv2.dilate()`, `cv2.morphologyEx()` etc.

### 2.2 Teoria

Transformações morfológicas são algumas operações simples baseadas no formato da imagem. É normalmente executado em imagens binárias. Ela precisa de duas entradas, uma é a imagem original, a segunda é chamada elemento estruturante ou kernel, que decide a natureza da operação. Dois operadores morfológicos básicos são erosão e dilatação. Em seguida, suas formas variantes, como Abertura, Fechamento, Gradiente, etc.

#### 2.2.1 1. Erosão

A idéia básica da erosão é eliminar os limites do objeto em primeiro plano (sempre tente manter o primeiro plano em branco), o que significa que o kernel desliza pela imagem (como na convolução 2D). Um pixel na imagem original (1 ou 0) será considerado 1 somente se todos os pixels sob o kernel forem 1, caso contrário, ele será erodido (reduzido a zero).

Então, o que acontece é que todos os pixels próximos ao limite serão descartados, dependendo do tamanho do kernel. Assim, a espessura ou o tamanho do objeto de primeiro plano diminui ou simplesmente a região branca diminui na imagem. É útil para remover pequenos ruídos brancos, destacar dois objetos conectados, etc.

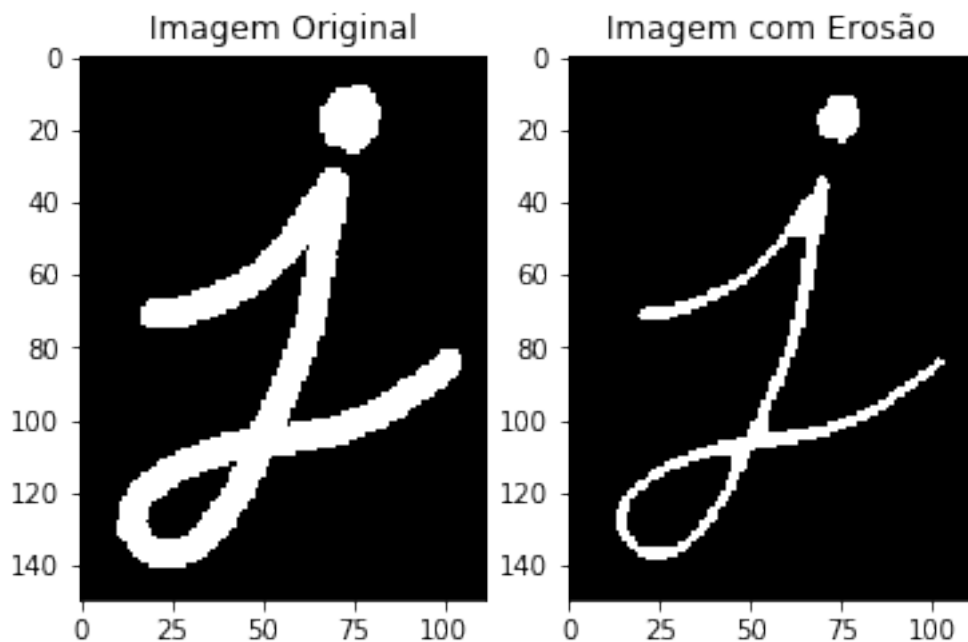
Vamos ver como funciona com um kernel 5x5:

```
In [16]: import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('j.png')

kernel = np.ones((5,5),np.uint8)
erosion = cv2.erode(img,kernel,iterations = 1)

fig, (o, e) = plt.subplots(1,2)
o.imshow(img)
o.set(title="Imagem Original")
e.imshow(erosion)
e.set(title="Imagem com Erosão")
plt.show()
```



### 2.2.2 2. Dilatação

É o oposto da erosão. Um elemento de pixel é “1”, se pelo menos um pixel abaixo do núcleo for “1”. Por isso, aumenta a região branca na imagem ou o tamanho do objeto em primeiro plano aumenta. Normalmente, em casos como a remoção de ruídos, a erosão é seguida por dilatação. Como a erosão remove ruídos brancos, ela também encolhe o objeto. Então nós dilatamos isso. Como o ruído se foi, eles não voltarão, mas nossa área de objeto aumenta. Também é útil para unir partes quebradas de um objeto.

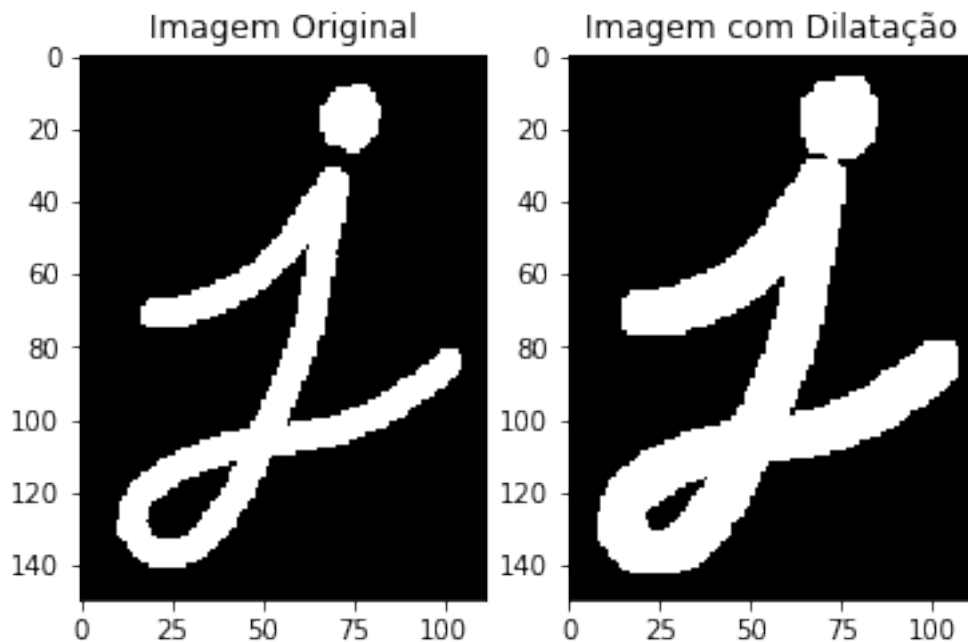
Vejamos:

```
In [17]: import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('j.png')

kernel = np.ones((5,5),np.uint8)
dilation = cv2.dilate(img,kernel,iterations = 1)

fig, (o, d) = plt.subplots(1,2)
o.imshow(img)
o.set(title="Imagem Original")
d.imshow(dilation)
d.set(title="Imagem com Dilatação")
plt.show()
```



### 2.2.3 3. Abertura

A abertura é apenas outro nome de erosão seguido de dilatação. É útil na remoção de ruído. Utiliza-se a função, `cv2.morphologyEx()`.

```
In [21]: import cv2
import numpy as np
from matplotlib import pyplot as plt
```

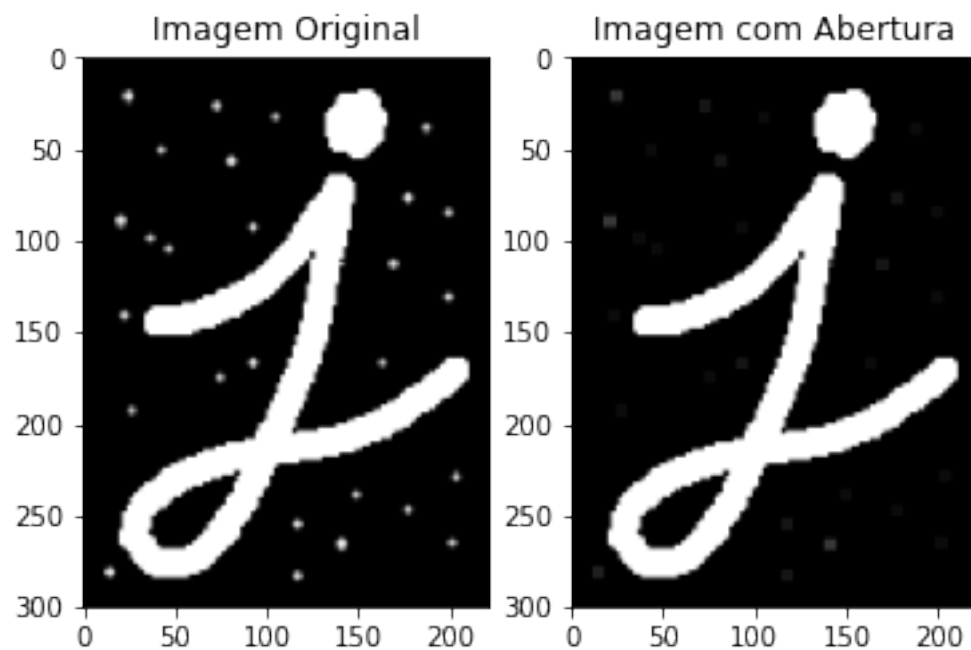
```

img = cv2.imread('abertura.jpg')

kernel = np.ones((5,5),np.uint8)
opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)

fig, (o, op) = plt.subplots(1,2)
o.imshow(img)
o.set(title="Imagem Original")
op.imshow(opening)
op.set(title="Imagem com Abertura")
plt.show()

```



#### 2.2.4 4. Fechamento

O fechamento é inverso da abertura, dilatação seguida de erosão. É útil para fechar pequenos furos dentro dos objetos em primeiro plano, ou pequenos pontos pretos no objeto.

Veja o código:

```

In [22]: import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('fechamento.jpg')

kernel = np.ones((5,5),np.uint8)

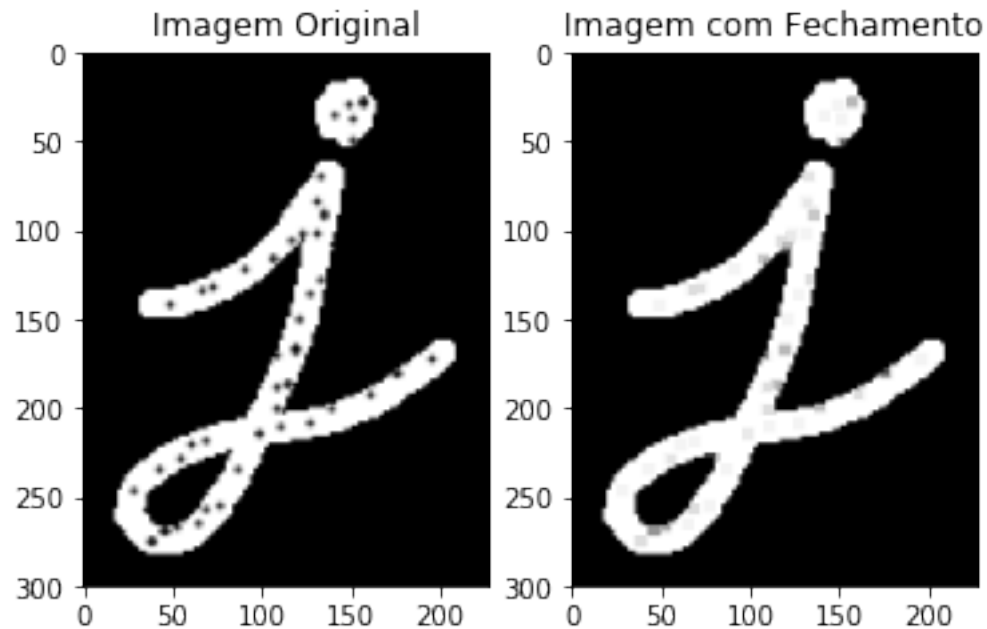
```

```

closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)

fig, (o, c) = plt.subplots(1,2)
o.imshow(img)
o.set(title="Imagem Original")
c.imshow(closing)
c.set(title="Imagem com Fechamento")
plt.show()

```



### 2.2.5 5. Gradiente Morfológico

É a diferença entre dilatação e erosão de uma imagem. O resultado será parecido com o contorno do objeto.

```

In [25]: import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('j.png')

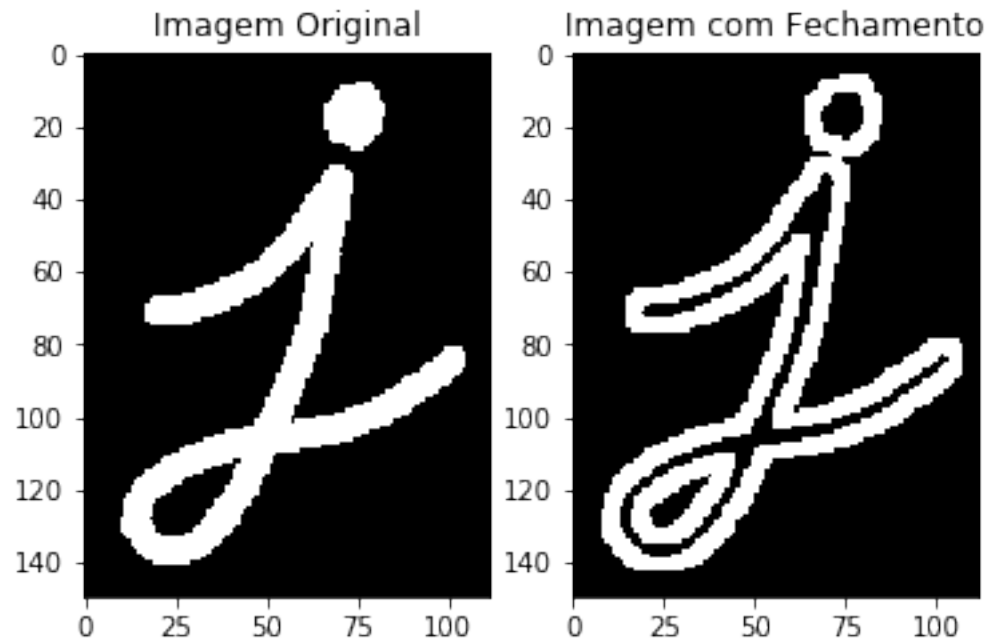
kernel = np.ones((5,5),np.uint8)
gradient = cv2.morphologyEx(img, cv2.MORPH_GRADIENT, kernel)

fig, (o, g) = plt.subplots(1,2)
o.imshow(img)
o.set(title="Imagem Original")

```



```
g.imshow(gradient)
g.set(title="Imagem com Fechamento")
plt.show()
```



## 2.3 Material Complementar

### 1. Operações Morfológicas