Processamento Digital de Imagens

FILTRAGEM EM OPEN CV

CPGCG/UFPR

Prof. Dr. Jorge Centeno

descrição

- OpenCV possui soluções prontas para efetuar a filtragem de imagens.
- Uma opção é usar uma função de convolução 2D genérica, outra é aplicar os filtros previamente definidos.
- Em cada caso, é necessário conhecer os parâmetros de entrada.

Programa básico

```
import numpy as np
import cv2
from google.colab.patches import cv2_imshow

from google.colab import drive
drive.mount('/content/gdrive’)
pasta="/content/gdrive/My Drive/fotos/"
arquivo="dog.jpg"
nome=pasta+arquivo

img = cv2.imread(nome)
n,m,nb = img.shape
print(n,m,nb)
I=img[:,:,2]
cv2_imshow(I)
```

Convolução 2D

Um filtro 2D pode ser aplicado usando a função **filter2D,** que é uma função geral que aceita como entrada uma imagem e a matriz de pesos (kernel) definida pelo usuário

cv2.filter2D(I, ddepth=-1, kernel=P)

argumentos

- **I:** Matriz imagem
- **ddepth=-1**: indica que a saída tem a mesma profundidade (usar sempre -1)
- **kernel=P** : especifica o filtro, uma matriz

Exemplo: passa-baixas

```
# passa baixas
P = np.ones((3, 3), np.float32)
P=P/np.sum(P)
im = cv2.filter2D(I, ddepth=-1,kernel=P)
cv2_imshow(im)



P = np.ones((5, 5), np.float32) / 25
im = cv2.filter2D(I, ddepth=-1,kernel=P)
cv2_imshow(im)
```

BLUR

Open CV tem uma função específica para passa-baixas (BLUR=embaçar). Neste caso devemos especiticar apenas a imagem e as dimensões do filtro

cv2.blur(src=image, ksize=(5,5))

•

Gaussiano

- Também é possível aplicar um filtro passa-baixas Gaussiano usando a função pré-definida.
- No caso do filtro Gaussiano, a definição dos pesos depende do valor do desvio padrão, de pode ser especificado, mas também pode ser usado um valor “default”

GaussianBlur(src, ksize, sigmaX[, dst[, sigmaY[, borderType]]])

GaussianBlur(src, ksize, sigmaX, sigmaY, borderType)

- Src= Imagem de entrada
- Ksize: tamanho da janela do filtro (kernel)
- sigmaX e sigmaY: valores do desvio padrão que definem o filtro Gaussiano X (horizontal) Y (vertical).

escolhendo sigmaX=0, usa-se o valor default do desvio padrão, calculado em função do tamanho da janela. Mas também é permitido explicitar valores (positivos) de sigma.

```
ima = cv2.GaussianBlur(src=I, ksize=(5,5), sigmaX=0, sigmaY=0)
cv2_imshow(ima)
```

- borderType especifica como serão representadas as bordas (que não são filtradas):
- cv.BORDER_CONSTANT
- cv.BORDER_REPLICATE
- cv.BORDER_REFLECT
- cv.BORDER_WRAP
- cv.BORDER_REFLECT_101
- cv.BORDER_TRANSPARENT
- cv.BORDER_REFLECT101
- cv.BORDER_DEFAULT
- cv.BORDER_ISOLATED

outros filtros

Mediana

É um filtro passa baixas

medianBlur(src, ksize)

Mas não é linear.

Neste caso, apenas o tamanho da vizinhança deve ser especificado.

outros filtros

bilateralFilter

é basicamente um filtro Gaussiano, que varia seus pesos em função do contraste local. Com isto, ele se adapta a cada região, suavizando com maior ou menor intensidade a região com a finalidade de preservar as bordas e detalhes que são fortemente afetados pelo filtro Gaussiano.

. **bilateralFilter(src, raio, sigmaColor, sigmaSpace)**

- **Raio:** define a vizinhança do pixel usando um raio em torno do pixel
- **sigmaSpace:** é o desvio padrão espacial usado no filtro Gaussiano. Pixels mais próximos (em espaço) recebem maior peso.
- **sigmaColor**: define o desvio padrão em termos de valores digitais. Com isto, pixels com valores digitais mais próximos recebem maior peso e pixels muito diferentes menores pesos

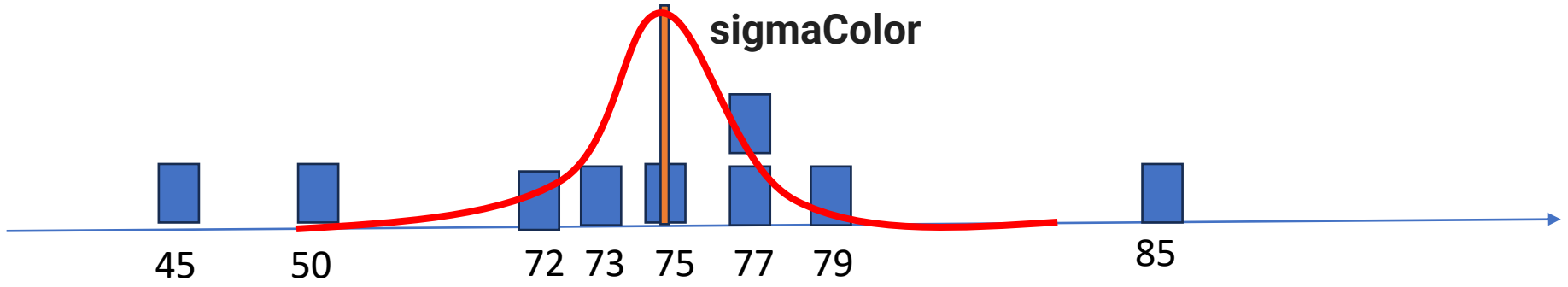
- Exemplo: Imagem

12	14	15	92	65
35	73	85	45	38
43	79	75	50	46
32	72	77	77	32
26	46	97	14	15

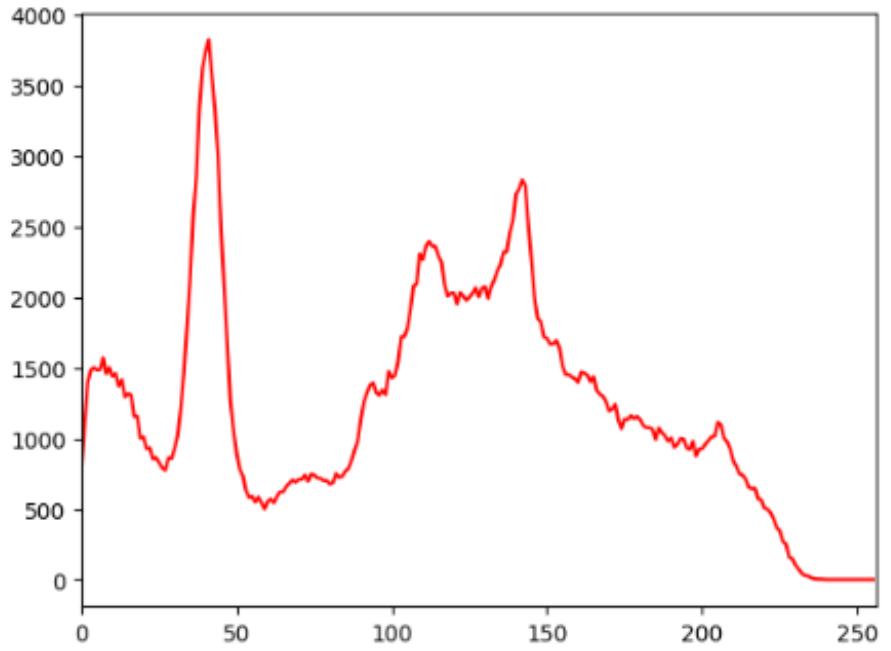
Kernel Gaussiano, depende de **sigmaSpace**

1	2	1
2	4	2
1	2	1

sigmaColor define peso em termos de similaridade de valor digital



- histograma



Histograma

cv.calcHist(imagem, banda, mask, histSize, faixa)

- image : entrada uint8 ou float32. Deve ser escrito em colchetes, "[I]".
- banda : No caso de imagens coloridas, especifica qual banda será processada. Em imagens em nível de cinza, deve-se usar [0], sempre entre colchetes.
- mask : opção de processor apenas uma parte da imagem (mascara). Por default se processa toda a imagem com a opção "None".
- histSize : número de elementos do histograma. Em imagens de 8 bits o correto é usar [256].
- faixa : a faixa a ser representada, em colchetes. Geralmente [0,256]

```
from matplotlib import pyplot as plt

hist = cv2.calcHist([I],[0],None,[256],[0,256])
plt.plot(hist,color = 'red')
plt.xlim([0,256])
plt.show()
```

- Limiarização

- Consiste em verificar se o valor do pixel supera um limiar. Neste caso (TRUE) o valor é mudado para 255 (ou 1), caso contrário o valor é anulado.

```
True=255
Limiar=100
if I[x,y] > Limiar:
  J[x,y] = True
else:
  J[x,y] = 0
```

Podemos especificar o limiar ou achar o limiar ótimo. Vejamos...

opções

th, K = cv2.threshold(IMA, limiar, maximo, MODO)

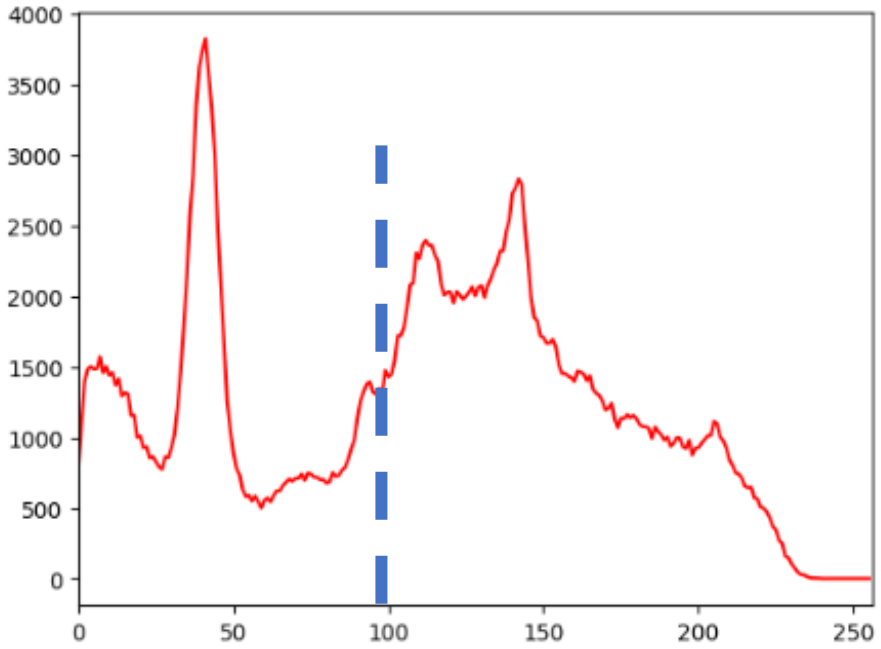
- th: limiar (redundante)
- K: imagem de saída
- IMA: imagem de entrada
- Limiar: Limiar especificado pelo usuário. Deve estar dentro d afaixa de valores da imagem. Vale a pena visualizer o histograma para esolher um valor
- Maximo: o valor que será atribído aos pixels que superem o limiar
- MODO: opção de binarização, pode ser simples ou preservando valores originais. veja a lista …

Exemplo cv2.THRESH_BINARY

cv2.THRESH_BINARY: opção de binarização simples

```
maximo=255
Limiar=100
if I[x,y] > Limiar:
  J[x,y] = maximo
else:
  J[x,y] = 0
```

```
limiar=100
maximo=255
th, K = cv2.threshold(I, limiar, maximo, cv2.THRESH_BINARY)
cv2_imshow(K)
print(th)
```



Limiar=100

- MODO: THRESH_BINARY_INV

Neste caso, o resultado é o negative do anterior. Ou seja, os valores que superam o limiar são marcados como zero.

```
maximo=255
Limiar=100
if I(x,y) > Limiar:
  J(x,y) =0
else:
  J(x,y) = maximo
```

- MODO THRESH_TRUNC

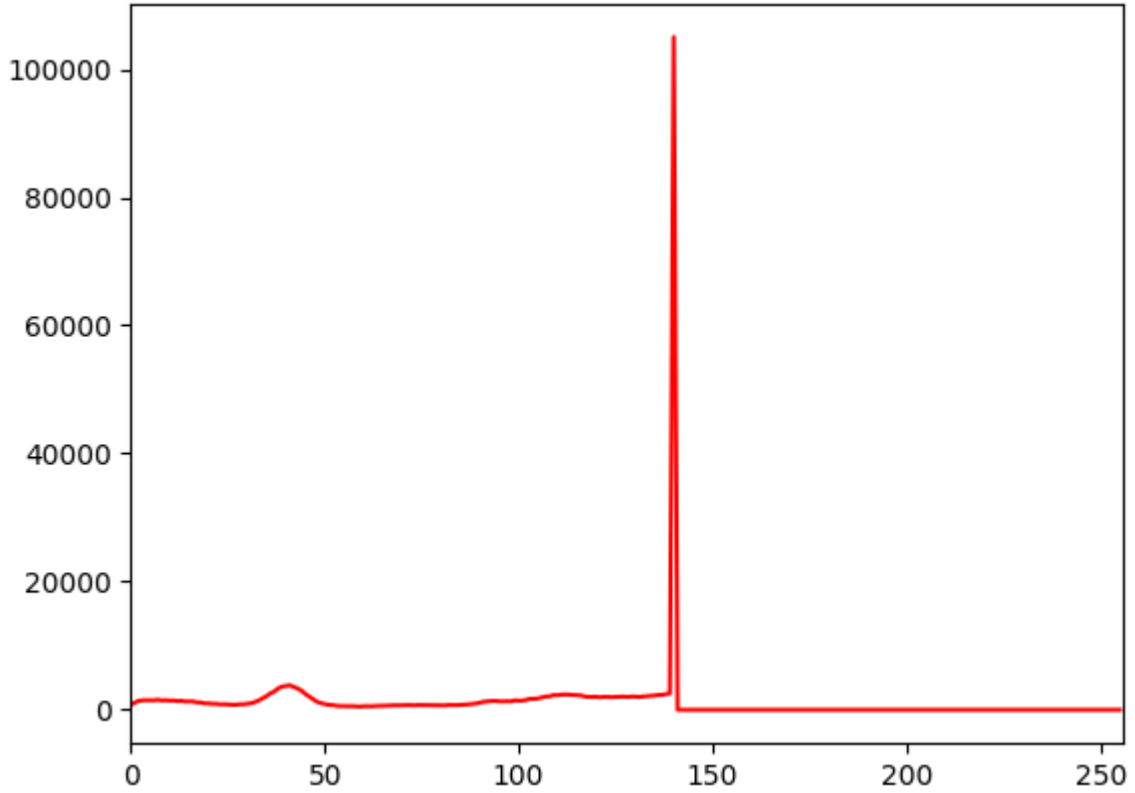
É usado para salientar os valores acima do limiar e preserver aqueles abaixo do limiar. Neste caso, os valores que seriam anulados(abaixo do limiar) preservam seu valor original e aqueles acima do limiar assumem o valor do limiar. Equivale a truncar a imagem com o limiar.

```
th, K = cv2.threshold(I, limiar, maximo, cv2.THRESH_TRUNC)
```

Ignora-se máximo

```
maximo=255
Limiar=100
if I[x,y] > Limiar:
  J[x,y] = maximo
else:
  J[x,y] =I[i,j]
```

```
limiar=140
th, K = cv2.threshold(I, limiar, maximo, cv2.THRESH_TRUNC)
cv2_imshow(K)
```



THRESH_TOZERO

- Similar ao anterior, mas neste caso o valor original é representado se o limiar for superado. Os pixels abaixo do limiar são anulados (zero)

THRESH_TOZERO_INV

- Inverted Threshold to Zero,
- Se o valor do limiar for superado, o pixel recebe zero. Caso contrário o valor original é preservado.