Filtragem

Aulas práticas

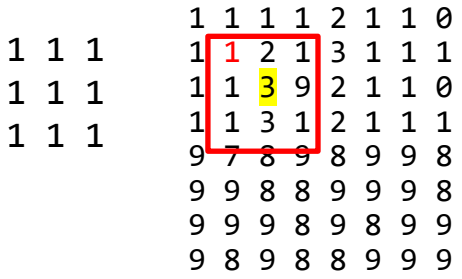
Processamento de imagens

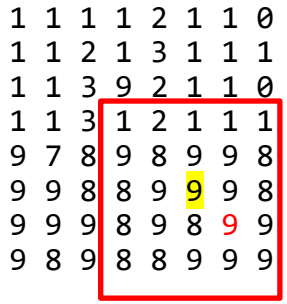
Prof. Dr. Jorge Centeno

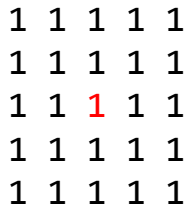
- No ambiente Google Colab

- Desenvolva uma função de filtro de média com tamanho variável (3,5,7,...)
- Um filtro passa-altas genérico

Programa (filtro linear)







Considere, se o filtro tem tamanho dim x dim...

Vizinhos antes e depois

lado=(dim-1)/2

Verifique para dim=3, ou 5

No 3x3, a varredura não pode ser feita para o primeiro pixel, devemos começar no elemento (lado+1)

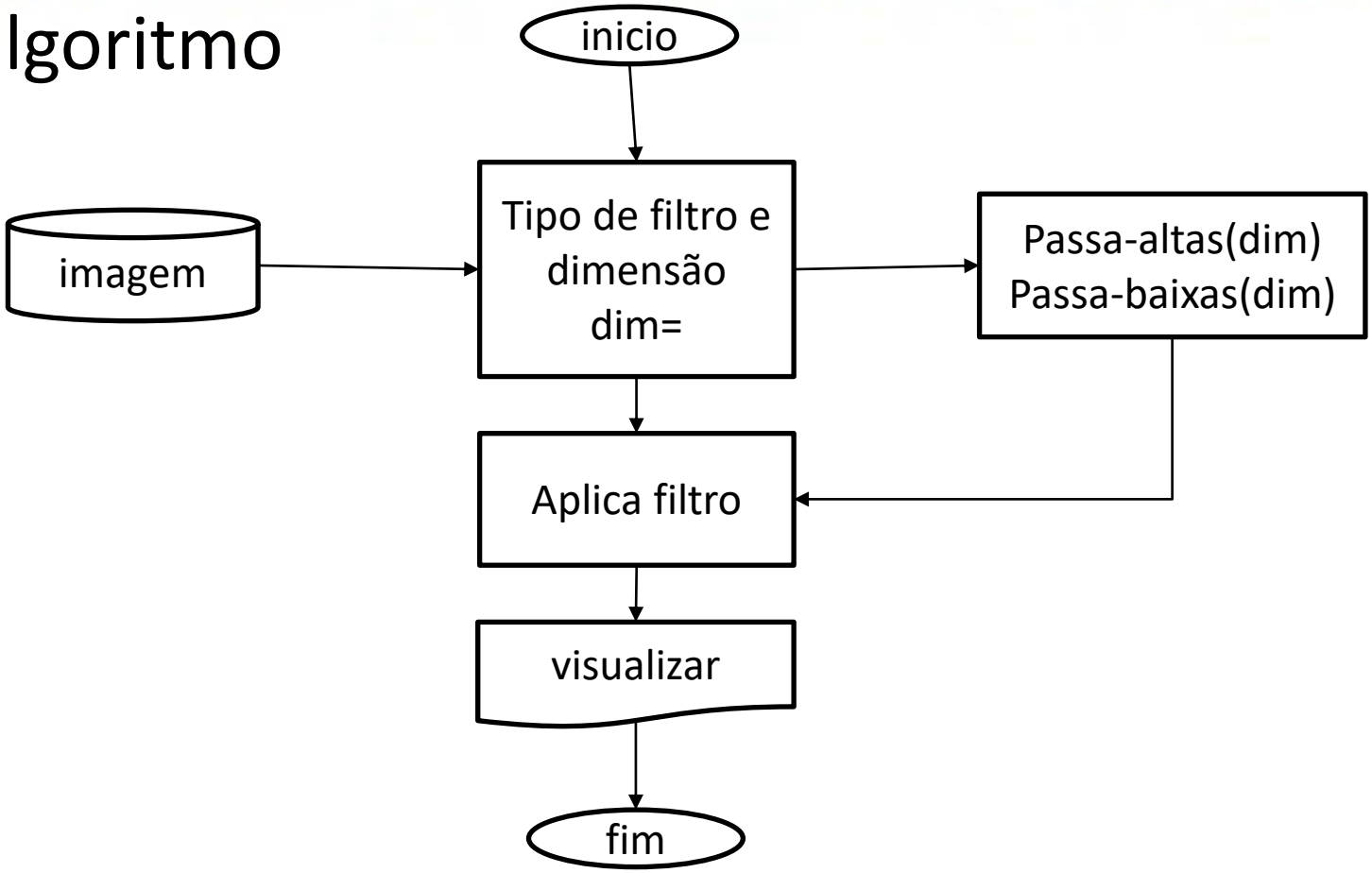
e não podemos terminar na ultima linha (n) mas devemos terminar em (n-[lado-1])

E no caso 5x5 ?

Como parametrizar isto?

- Podemos criar um filtro genérico?
- Digamos um passa-baixas com tamanho desejado
- Ou um passa-altas com tamanho especificado?

- algoritmo


inicio
imagem
Tipo de filtro e dimensão dim=
Passa-altas(dim)
Passa-baixas(dim)
Aplica filtro
visualizar
fim

Criar uma janela de médias

- Dada a dimensão, crie a matriz com os respectivos pesos para o cálculo da média

Python function

```
def low_pass(dim):

  f= np.zeros((dim,dim),dtype =
  float)


  f=f+1


  f=f/(dim*dim )

   return(f)
```

```
0 0 0
0 0 0
0 0 0

1 1 1
1 1 1
1 1 1

0.11 0.11 0.11
0.11 0.11 0.11
0.11 0.11 0.11
```

Entrada, o tamanho do filtro (dim), retorna o filtro f

Criar uma janela passa-altas

- Dada a dimensão, crie a matriz com os pesos para o cálculo do filtro passa-altas

python

```
def high_pass(dim):

    f= np.zeros((dim,dim),dtype = float)

    f= f-1

    f[lado,lado]=dim*dim

     return(f)
```

```
 0 0 0
 0 0 0
 0 0 0

-1 -1 -1
-1 -1 -1
-1 -1 -1

-1 -1 -1
-1  9 -1
-1 -1 -1
```

Entrada, o tamanho do filtro (dim), retorna o filtro f

Dois filtros em python

```
## ################# LOW_pass #################
def low_pass(dim):
   f= np.zeros((dim,dim),dtype = float) # matriz vazia
   f=f+1 # cria matriz com tudo igual a 1, zeros+1=1
   f=f/(dim*dim ) # dividir para calcular a media
   return(f)


## ################# HIGH_pass #################
def high_pass(dim):
   f= np.zeros((dim,dim),dtype = float) # matriz vazia
   f=f-1 # cria matriz com tudo igual a -1, zeros-1=-1
   f[lado,lado]=dim*dim # e muda o central
   return(f)
```

inicio

```
## ################cria janela de filtro tamanho DIM ################
dim=5                 # dimensao do filtro exe 3x3 dim=3, 5x5 dim=5...
lado=(dim-1)/2    # numero de vizinhos antes ou depois do central
lado=np.uint8(lado)
#_________________ escolha o filtro aqui
f=low_pass(dim)  # escolhe passa baixas (3x3)
# f=high_pass(dim)
print('dim=',dim, 'vizinho=',lado)
print('F=',f)

# ler imagem de entrada e recuperar dimensoes
X1= plt.imread('small.tif')
nl,nc = X1.shape
X=np.array(X1, dtype=float) #transforma em float para facilitar
multiplicacoes
Y=np.zeros((nl,nc),dtype = float) # replica imagem para gerar uma saida
Y=np.uint8(Y) # em uint 8 para armazenar em byte
```

Filtragem

```
for L in range(lado,nl-lado): # varrer em linhas
    for C in range (lado, nc-lado): # varrer em colunas
        s=0. # zerar soma para convolucao
        for i in range(dim): #varrer filtro, linhas e colunas
            for j in range(dim):
                # determina o pixel na imagem
                p=(L-lado)+i
                q=(C-lado)+j
                s=s+X[p,q]*f[i,j] #pixel*peso
            if s>255: #truncar se ficar fora da faixa
                s=255
            if s<0:
                s=0
            v=np.uint8( np.round( s ) ) # muda a uint8
            Y[L,C]=v # salva na posicao do central
plt.imsave('saida.png',Y,cmap='gray')
```

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE CIÊNCIAS DA TERRA
Departamento de Geomática

Disciplina: PROCESSAMENTO DIGITAL DE IMAGENS II
Código: GA144

CH Total:45 h
CH Semanal 03 h

FILTRAGEM EM OPEN CV

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(IMAGEM, ddepth=-1, kernel=P)

argumentos

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

Exemplo: passa-baixas

Em nosso programa, temos a imagem de nível de cinza (1 banda) como “I”

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

- Use este filtro na imagem do exercício de filtros

BLUR

BLUR = embaçar

Open CV tem uma função específica para passa-baixas (BLUR).

Neste caso devemos especificar apenas a imagem e as dimensões do filtro

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

a biblioteca se encarrega de construir o filtro no tamanho especificado

Gaussiano

- **GaussianBlur**
- 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”

- Parâmetros...

GaussianBlur(IMA, ksize, sigmaX, sigmaY)

- IMA= Imagem de entrada
- ksize: tamanho da janela do filtro (kernel size, exe (5,5), (7,7) (3,3))
- 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(I, ksize=(5,5), sigmaX=0, sigmaY=0)
cv2_imshow(ima)
```

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

- **borderType** especifica como serão representadas as bordas (que não são filtradas devido ao tamanho da janela móvel, pois nas bordas a vizinhança não está definida. Pode ser:
- cv.BORDER_CONSTANT : espeficicar um valor digital, ex:255 para branco
- cv.BORDER_REPLICATE : repete o valor do pixel da imagem original
- 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.

Seletivo

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	45	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