BCM0505-22 – Processamento da Informação

Matrizes - Parte 2

Maycon Sambinelli
m.sambinelli@ufabc.edu.br
http://professor.ufabc.edu.br/~m.sambinelli/

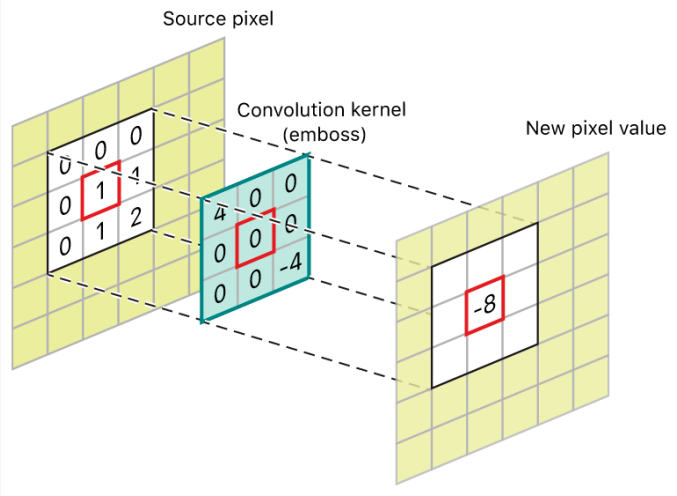
Outline

Convolução de Matriz

Convolução de matriz

Sejam $M \in \mathbb{R}^{m \times n}$ uma matriz e $K \in \mathbb{R}^{3 \times 3}$ uma matriz denomidada *kernel*. A **convolução** de K sobre M é a matriz obtida a partir do seguinte processo:

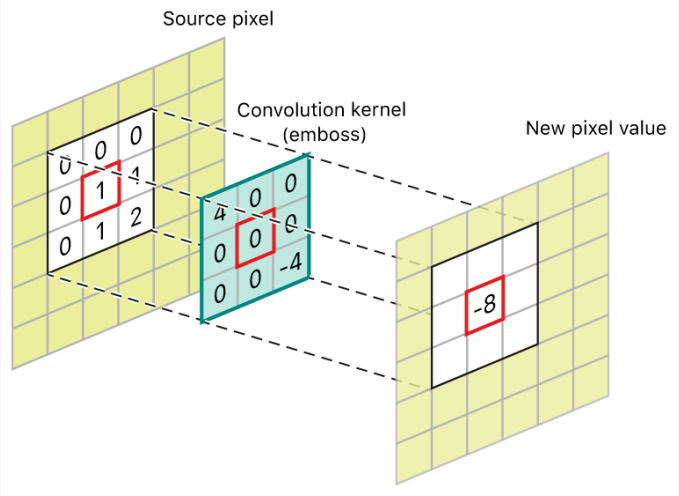
- Centralize o kernel em uma entrada (i, j) de M
- Multiplique cada valor do kernel pelo valor de M correspondente
- Some os resultados
- O resultado será gravado na entrada (i, j) da matriz resultante da convolução



Convolução de matriz

Sejam $M \in \mathbb{R}^{m \times n}$ uma matriz e $K \in \mathbb{R}^{3 \times 3}$ uma matriz denomidada *kernel*. A **convolução** de K sobre M é a matriz obtida a partir do seguinte processo:

- Centralize o kernel em uma entrada (i, j) de M
- Multiplique cada valor do kernel pelo valor de M correspondente
- Some os resultados
- O resultado será gravado na entrada (i, j) da matriz resultante da convolução



Escreva a função `matriz_convolucao(matriz, kernel)` que recebe duas matrizes e retorna a matriz de convolução de `kernel` sobre `matriz`.

Convolução de matriz

```
1  def matriz_valor_kernel_em(matriz, kernel, i, j):
2      valor = 0
3      for k in range(-1, 2):
4          for l in range(-1, 2):
5              valor += matriz[i + k][j + l] * kernel[1 + k][1 + l]
6      return valor
7
8  def matriz_convolucao(matriz, kernel):
9      matriz_com_borda = matriz_adiciona_borda(matriz, 0)
10     nlin = len(matriz)
11     ncol = len(matriz[0])
12     convolucao = matriz_nova(nlin, ncol, 0)
13     for i in range(nlin):
14         for j  in range(ncol):
15             convolucao[i][j] = matriz_valor_kernel_em(matriz_com_borda,
16                                                       kernel,
17                                                       i + 1,
18                                                       j + 1)
19     return convolucao
```

Convolução de matriz (funções auxiliares)

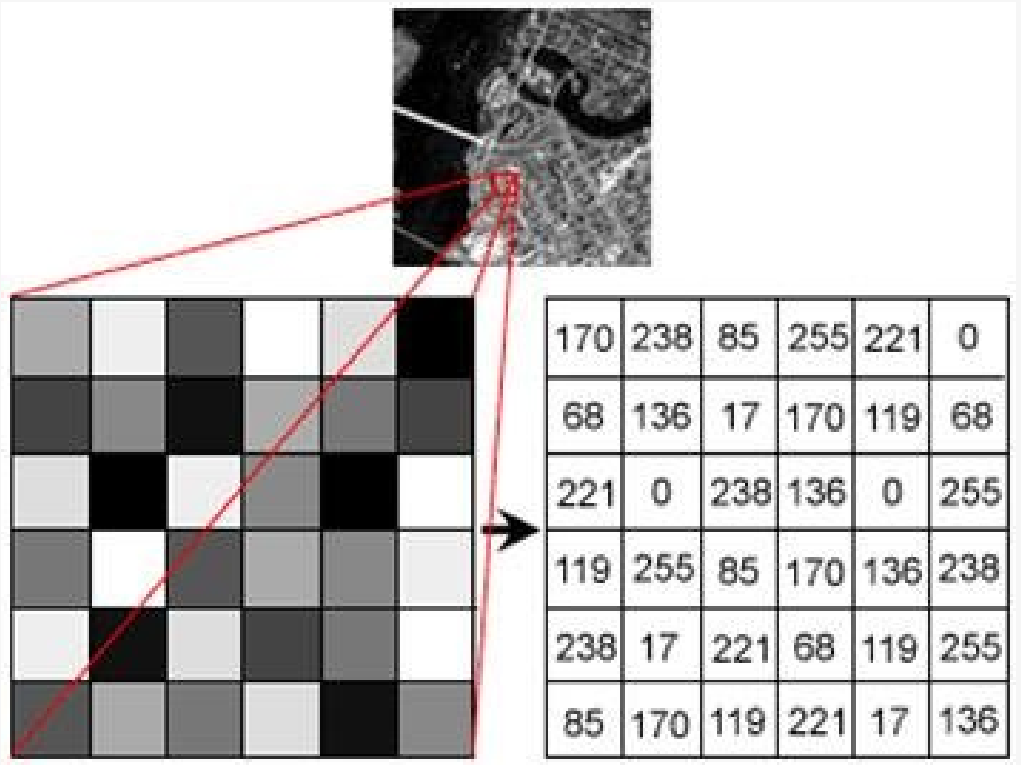
```python
def matriz_nova(nlins, ncols, value):
    M = []
    for i in range(nlins):
        M.append([value] * ncols)
    return M

def matriz_adiciona_borda(matriz, valor):
    nlins = len(matriz)
    ncols =  len(matriz[0])
    M = matriz_nova(nlins + 2, ncols + 2, valor)
    for  i in range(nlins):
        for j in range(ncols):
            M[i + 1][j + 1] = matriz[i][j]
    return M
```

Imagens

Imagens

Um uso importante de matrizes é na representação de imagens.



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

Uma imagem em escala de cinza, por exemplo, pode ser representada como uma matriz de inteiros, em que cada número representa o nível de brilho de um pixel, variando de 0 (preto) a 255 (branco).

Imagens

Um uso importante de matrizes é na representação de imagens. O *Portable Graymap Format* (PGM) é um formato simples de arquivo de imagem que representa imagens em escala de cinza.

```
P2
# exemplo simples reescalado
24 7
255
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 51 51 51 51 0 0 119 119 119 119 0 0 187 187 187 187 0 0 255 255 255 255 0
0 51 0 0 0 0 0 119 0 0 0 0 0 187 0 0 0 0 0 255 0 0 255 0
0 51 51 51 0 0 0 119 119 119 0 0 0 187 187 187 0 0 0 255 255 255 255 0
0 51 0 0 0 0 0 119 0 0 0 0 0 187 0 0 0 0 0 255 0 0 0 0
0 51 0 0 0 0 0 119 119 119 119 0 0 187 187 187 187 0 0 255 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

- P2 indica que é um arquivo PGM em texto (ASCII)

Imagens

Um uso importante de matrizes é na representação de imagens. O *Portable Graymap Format* (PGM) é um formato simples de arquivo de imagem que representa imagens em escala de cinza.

```
P2
# exemplo simples reescalado
24 7
255
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 51 51 51 51 0 0 119 119 119 119 0 0 187 187 187 187 0 0 255 255 255 255 0
0 51 0 0 0 0 0 119 0 0 0 0 0 187 0 0 0 0 0 255 0 0 255 0
0 51 51 51 0 0 0 119 119 119 0 0 0 187 187 187 0 0 0 255 255 255 255 0
0 51 0 0 0 0 0 119 0 0 0 0 0 187 0 0 0 0 0 255 0 0 0 0
0 51 0 0 0 0 0 119 119 119 119 0 0 187 187 187 187 0 0 255 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

- P2 indica que é um arquivo PGM em texto (ASCII)
- A linha com # é um comentário (opcional)

Imagens

Um uso importante de matrizes é na representação de imagens. O *Portable Graymap Format* (PGM) é um formato simples de arquivo de imagem que representa imagens em escala de cinza.

```
P2
# exemplo simples reescalado
24 7
255
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 51 51 51 51 0 0 119 119 119 119 0 0 187 187 187 187 0 0 255 255 255 255 0
0 51 0 0 0 0 0 119 0 0 0 0 0 187 0 0 0 0 0 255 0 0 255 0
0 51 51 51 0 0 0 119 119 119 0 0 0 187 187 187 0 0 0 255 255 255 255 0
0 51 0 0 0 0 0 119 0 0 0 0 0 187 0 0 0 0 0 255 0 0 0 0
0 51 0 0 0 0 0 119 119 119 119 0 0 187 187 187 187 0 0 255 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

- P2 indica que é um arquivo PGM em texto (ASCII)
- A linha com # é um comentário (opcional)
- 24 7 são as dimensões da imagem: 24 colunas e 7 linhas

Imagens

Um uso importante de matrizes é na representação de imagens. O *Portable Graymap Format* (PGM) é um formato simples de arquivo de imagem que representa imagens em escala de cinza.

```
P2
# exemplo simples reescalado
24 7
255
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 51 51 51 51 0 0 119 119 119 119 0 0 187 187 187 187 0 0 255 255 255 255 0
0 51 0 0 0 0 0 119 0 0 0 0 0 187 0 0 0 0 0 255 0 0 255 0
0 51 51 51 0 0 0 119 119 119 0 0 0 187 187 187 0 0 0 255 255 255 255 0
0 51 0 0 0 0 0 119 0 0 0 0 0 187 0 0 0 0 0 255 0 0 0 0
0 51 0 0 0 0 0 119 119 119 119 0 0 187 187 187 187 0 0 255 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

- P2 indica que é um arquivo PGM em texto (ASCII)
- A linha com # é um comentário (opcional)
- 24 7 são as dimensões da imagem: 24 colunas e 7 linhas
- 255 é o valor máximo de cinza (0 = preto, 255 = branco)

Imagens

Um uso importante de matrizes é na representação de imagens. O *Portable Graymap Format* (PGM) é um formato simples de arquivo de imagem que representa imagens em escala de cinza.

```
P2
# exemplo simples reescalado
24 7
255
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 51 51 51 51 0 0 119 119 119 119 0 0 187 187 187 187 0 0 255 255 255 255 0
0 51 0 0 0 0 0 119 0 0 0 0 0 187 0 0 0 0 0 255 0 0 255 0
0 51 51 51 0 0 0 119 119 119 0 0 0 187 187 187 0 0 0 255 255 255 255 0
0 51 0 0 0 0 0 119 0 0 0 0 0 187 0 0 0 0 0 255 0 0 0 0
0 51 0 0 0 0 0 119 119 119 119 0 0 187 187 187 187 0 0 255 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

- P2 indica que é um arquivo PGM em texto (ASCII)
- A linha com # é um comentário (opcional)
- 24 7 são as dimensões da imagem: 24 colunas e 7 linhas
- 255 é o valor máximo de cinza (0 = preto, 255 = branco)
- Os valores seguintes são os tons de cinza de cada pixel (escritos linha por linha)

Imagens

Um uso importante de matrizes é na representação de imagens. O *Portable Graymap Format* (PGM) é um formato simples de arquivo de imagem que representa imagens em escala de cinza.

```
P2
# exemplo simples reescalado
24 7
255
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 51 51 51 51 0 0 119 119 119 119 0 0 187 187 187 187 0 0 255 255 255 255 0
0 51 0 0 0 0 0 119 0 0 0 0 0 187 0 0 0 0 0 255 0 0 255 0
0 51 51 51 0 0 0 119 119 119 0 0 0 187 187 187 0 0 0 255 255 255 255 0
0 51 0 0 0 0 0 119 0 0 0 0 0 187 0 0 0 0 0 255 0 0 0 0
0 51 0 0 0 0 0 119 119 119 119 0 0 187 187 187 187 0 0 255 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

- P2 indica que é um arquivo PGM em texto (ASCII)
- A linha com # é um comentário (opcional)
- 24 7 são as dimensões da imagem: 24 colunas e 7 linhas
- 255 é o valor máximo de cinza (0 = preto, 255 = branco)
- Os valores seguintes são os tons de cinza de cada pixel (escritos linha por linha)

Imagens

Um uso importante de matrizes é na representação de imagens. O *Portable Graymap Format* (PGM) é um formato simples de arquivo de imagem que representa imagens em escala de cinza.

```
P2
# exemplo simples reescalado
24 7
255
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 51 51 51 51 0 0 119 119 119 119 0 0 187 187 187 187 0 0 255 255 255 255 0
0 51 0 0 0 0 0 119 0 0 0 0 0 187 0 0 0 0 0 255 0 0 255 0
0 51 51 51 0 0 0 119 119 119 0 0 0 187 187 187 0 0 0 255 255 255 255 0
0 51 0 0 0 0 0 119 0 0 0 0 0 187 0 0 0 0 0 255 0 0 0 0
0 51 0 0 0 0 0 119 119 119 119 0 0 187 187 187 187 0 0 255 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

- P2 indica que é um arquivo PGM em texto (ASCII)
- A linha com # é um comentário (opcional)
- 24 7 são as dimensões da imagem: 24 colunas e 7 linhas
- 255 é o valor máximo de cinza (0 = preto, 255 = branco)
- Os valores seguintes são os tons de cinza de cada pixel (escritos linha por linha)

Você pode usar o seguinte site para converter uma imagens no formato png, jpg e jpeg para o formato pgm: `https://to.imagestool.com/image-to-pgm`

Tratando imagens

A aplicação de um **filtro** em uma imagem é uma técnica usada para manipular ou realçar características da imagem (como suavizar, destacar bordas, borrar, etc).

- Filtros formam a base para operações em visão computacional, aprendizado de máquina, processamento de imagens, etc.

Tratando imagens

A aplicação de um **filtro** em uma imagem é uma técnica usada para manipular ou realçar características da imagem (como suavizar, destacar bordas, borrar, etc).

- Filtros formam a base para operações em visão computacional, aprendizado de máquina, processamento de imagens, etc.
- Um filtro é uma pequena matriz chamada **máscara** (ou **kernel**).

Tratando imagens

A aplicação de um **filtro** em uma imagem é uma técnica usada para manipular ou realçar características da imagem (como suavizar, destacar bordas, borrar, etc).

- Filtros formam a base para operações em visão computacional, aprendizado de máquina, processamento de imagens, etc.
- Um filtro é uma pequena matriz chamada **máscara** (ou **kernel**).

Tratando imagens

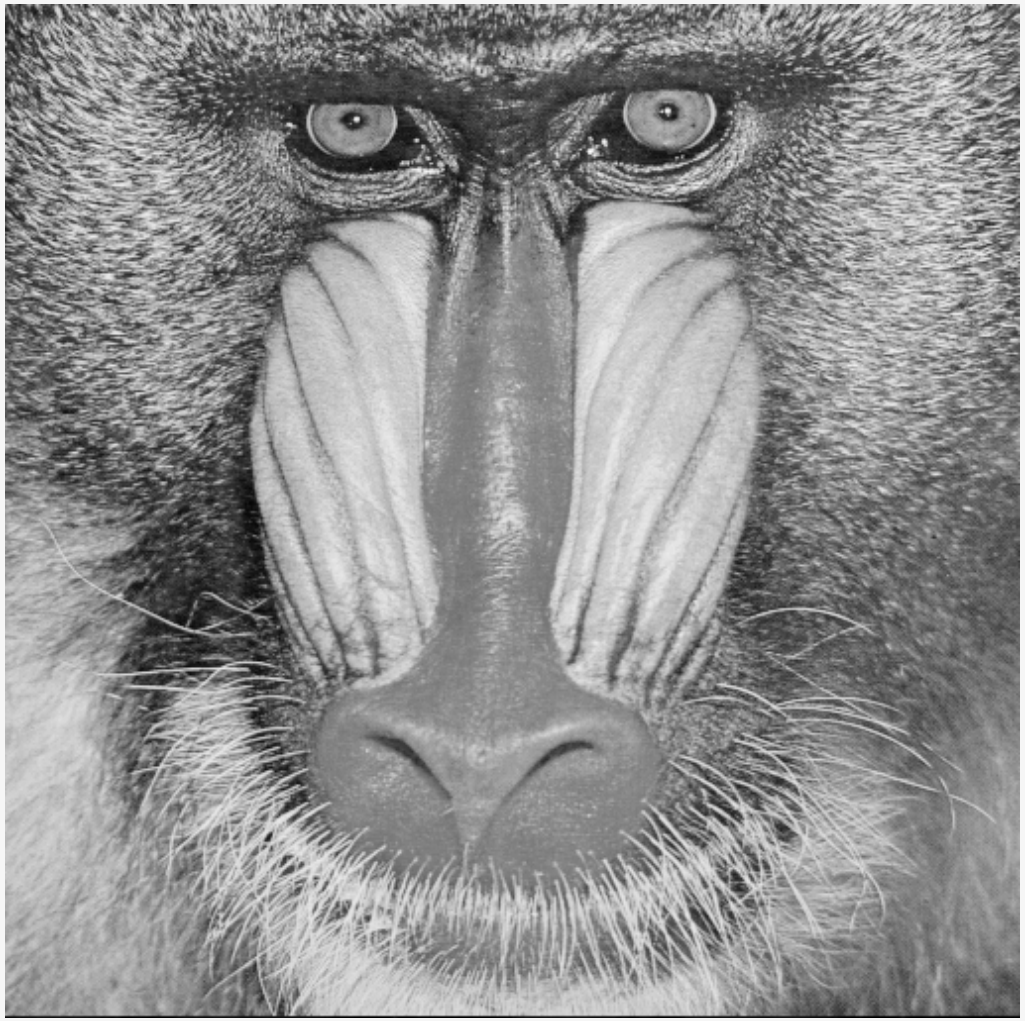
A aplicação de um **filtro** em uma imagem é uma técnica usada para manipular ou realçar características da imagem (como suavizar, destacar bordas, borrar, etc).

- Filtros formam a base para operações em visão computacional, aprendizado de máquina, processamento de imagens, etc.
- Um filtro é uma pequena matriz chamada **máscara** (ou **kernel**).

Para filtrar uma imagem, fazemos a convolução do kernel sobre a imagem (matriz)

Imagens: Principais Filtros

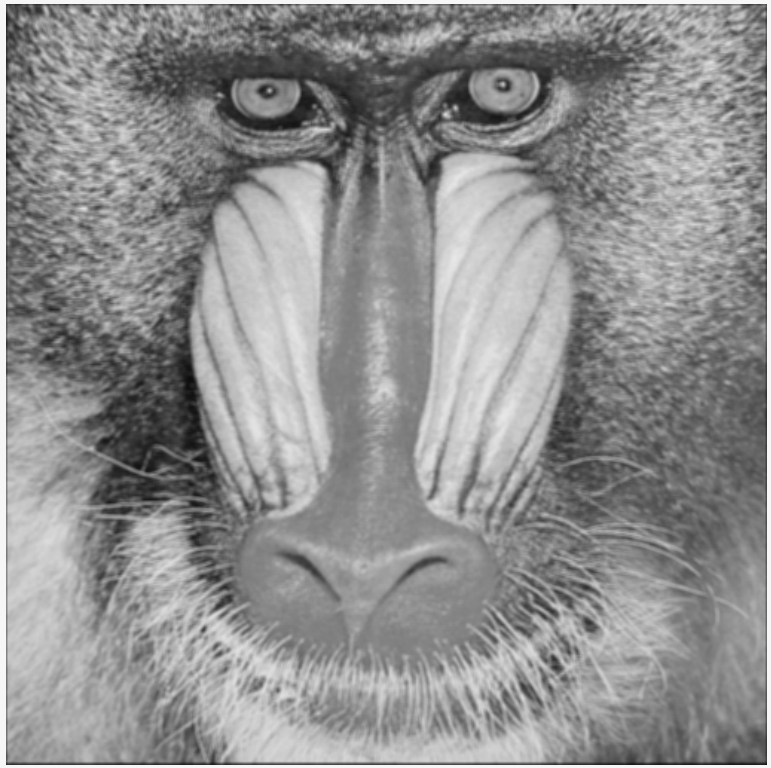
Imagem de Entrada



Filtro de Média

suaviza a imagem, reduzindo ruído e detalhes.

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

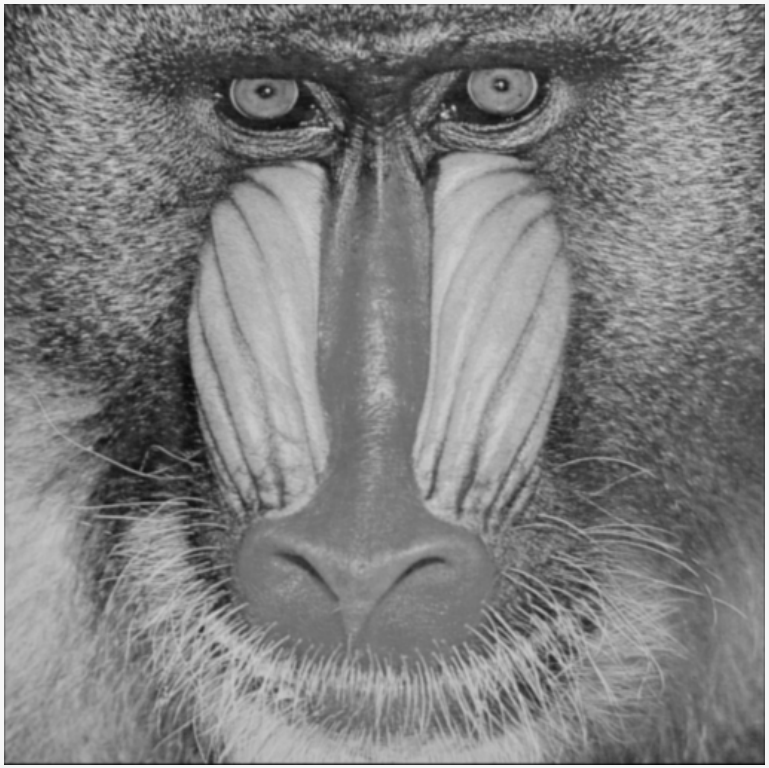


Cuidado com valores não inteiros!

Filtro Gaussiano

suaviza a imagem, mas com pesos que valorizam o centro.

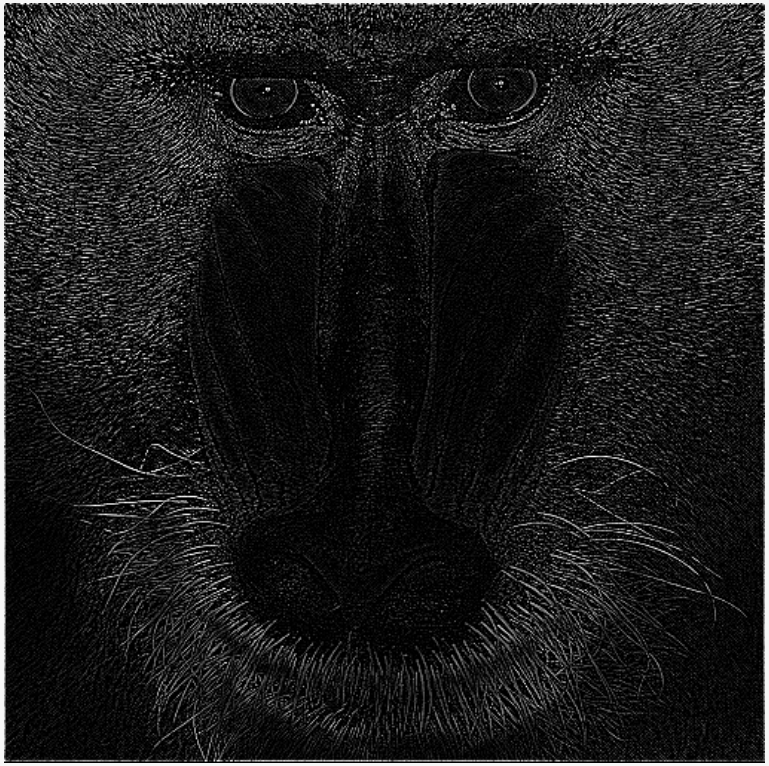
$$\begin{bmatrix} 1/16 & 2/16 & 1/16 \\ 2/16 & 4/16 & 2/16 \\ 1/16 & 2/16 & 1/16 \end{bmatrix}$$



Detecção de Bordas

destaca áreas com transições bruscas de intensidade.

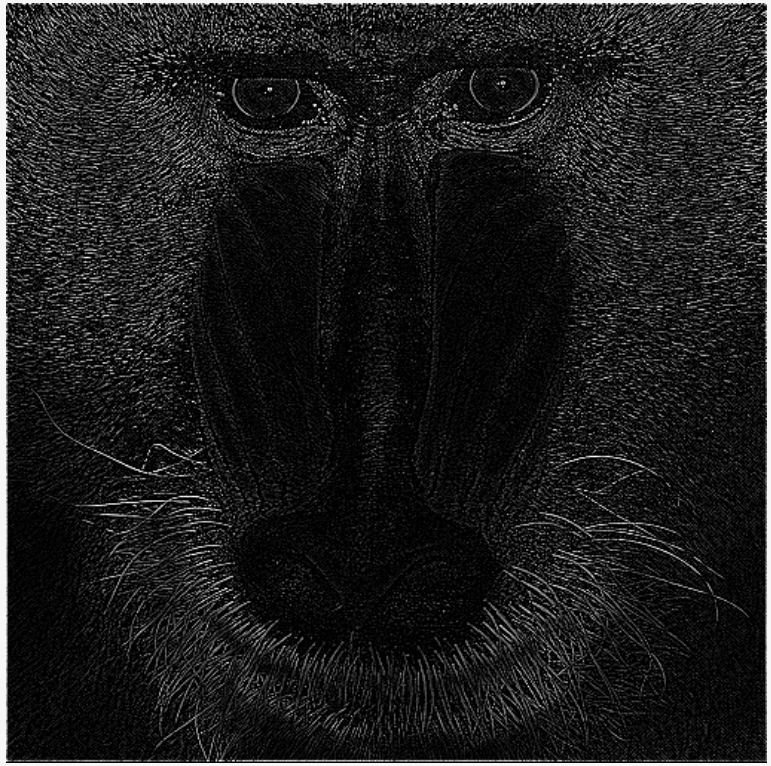
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



Detecção de Bordas

destaca áreas com transições bruscas de intensidade.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



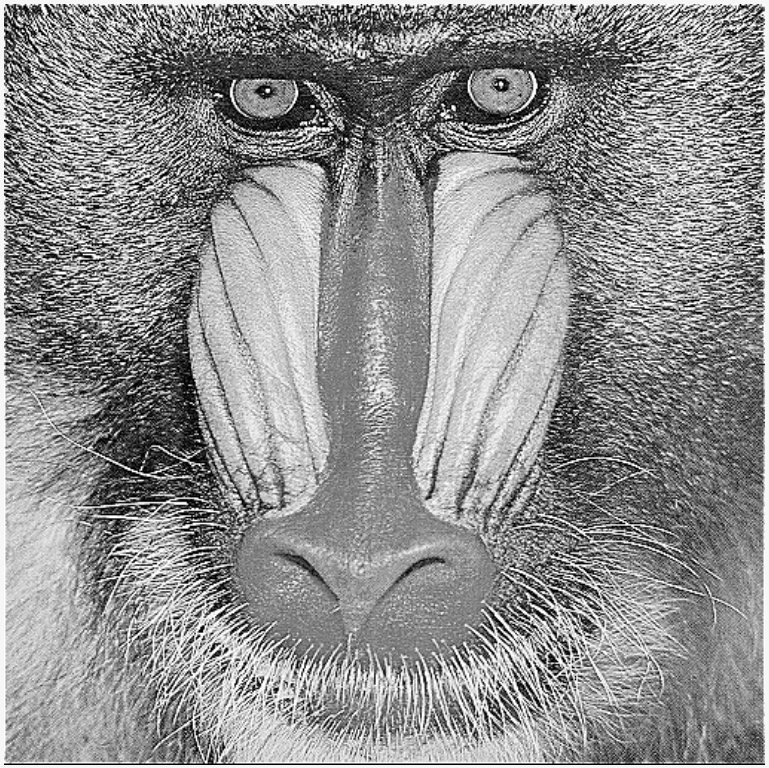
OBS: é possível que o valor do pixel fique fora do intervalo [0-255]. Você pode usar algumas estratégias para tratar esse problema:

- **Truncamento**: Se o resultador for menor que 0, transforme em 0, e se for maior que 255, transforme em 255
- **Normalização**: Normaliza os valores para que fiquem dentro do intervalo [0-255]

Realce

contrasta regiões com pouco relevo.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



Cuidado com valores fora dos limites da imagem!

App de Filtro

Programa que aplica um filtro em uma imagem fornecida

```
1  nome_imagem = "baboon.pgm"
2  nome_kernel = "media.txt"
3  nome_imagem_tratada = "baboon_filtrada.pgm"
4
5  imagem = carrega_imagem_pgm(nome_imagem)
6  kernel = carrega_matriz_arquivo(nome_kernel, float)
7  imagem_processada = matriz_convolucao(imagem, kernel)
8  imagem_pos_processamento(imagem_processada)
9  salva_imagem_pgm(nome_imagem_tratada, imagem_processada)
```

Filtro: carregando a imagem

Lê a imagem pgm e retorna a matriz correspondente

```
1  def carrega_imagem_pgm(nome_arquivo):
2      f = open(nome_arquivo, 'r') # abre o arquivo para leitura (r)
3      header = f.readline()       # lê linha do arquivo
4      header = f.readline()
5      largura, altura = list(map(int, f.readline().split()))
6      maior = f.readline()
7
8      matriz = []
9      for i in range(altura):
10         linha = list(map(int, f.readline().split()))
11         matriz.append(linha)
12     return matriz
```

Filtro: criando imagem a partir de matriz

Cria um arquivo pgm a partir de uma matriz

```
1  def salva_imagem_pgm(nome_arquivo, matriz):
2      altura = len(matriz)
3      largura = len(matriz[0])
4
5      f = open(nome_arquivo, 'w')  # abre o arquivo para escrita (w)
6      f.write("P2\n")
7      f.write("# Created in PI\n")
8      f.write(f"{largura} {altura}\n")
9      f.write("255\n")
10     for i in range(altura):
11         for j in range(largura):
12             f.write(f"{matriz[i][j]} ") # Escreva no arquivo
13         f.write("\n")
14     f.close()
```

Filtro: Carrega matriz a partir de um arquivo

Função para carregar o kernel

```
1  def carrega_matriz_arquivo(nome_arquivo, func_conversao):
2
3      f = open(nome_arquivo, 'r')
4      nlinhas = int(f.readline())
5
6      matriz = []
7      for i in range(nlinhas):
8          linha = list(map(func_conversao, f.readline().split()))
9          matriz.append(linha)
10
11     return matriz
```

Poderíamos ter especificado o kernel como uma imagem pgm e carregá-lo com a função `carrega_imagem_pgm`

Filtro: pós processamento

```
1  def imagem_pos_processamento(matriz):
2      for i in range(len(matriz)):
3          for j in range(len(matriz[0])):
4              matriz[i][j] = max(0, min(255, round(matriz[i][j])))
```

Quadrado mágico

Quadrado mágico

Dizemos que uma matriz quadrada de ordem n é um quadrado mágico se todos os números de 1 até n^2 aparecem exatamente uma vez e se a soma das linhas, colunas e diagonais é a mesma.

Por exemplo, se $n = 3$, a seguinte matriz é um quadrado mágico:

```
8  1  6
3  5  7
4  9  2
```

Faça um programa que determine se uma dada matriz é um quadrado mágico.

Quadrado mágico

Comecemos verificando se a matriz só possui números de 1 até n^2. Uma forma é usar um vetor cuja i-ésima posição armazena `True` quando o número i aparece na matriz.

```
1  def verifica_conteudo(A):
2      n = len(A)
3      numeros_encontrados = [False] * n**2
4      for lin in range(n):
5          for col in range(n):
6              if A[lin][col] >= 1 and A[lin][col] <= n**2:
7                  numeros_encontrados[A[lin][col]-1] = True
8
9      for elem in numeros_encontrados:
10         # elem é True ou False
11         if not elem:
12             return False
13
14     return True
```

Quadrado mágico

Outra forma de verificar se a matriz só possui números de 1 até n^2 é procurando por cada um desses números nela.

```
1  def verifica_conteudo(A):
2      n = len(A)
3      for i in range(1, n**2+1):
4          # procura por i em alguma posição da matriz
5          existe = False
6          for lin in range(n):
7              for col in range(n):
8                  if A[lin][col] == i:
9                      existe = True
10         if not existe:
11             return False
12     return True
```

Quadrado mágico

```python
def eh_quadrado_magico(A):
    assert(len(A) == len(A[0]))
    n = len(A)

    # Primeiro verificar se os números de 1 a n² estão
    if verifica_conteudo(A) == False:
        return False

    soma_base = 0           # soma da primeira linha
    for col in range(n):
        soma_base += A[0][col]

    for lin in range(1, n):         # Todas as outras linhas devem ter essa soma
        soma = 0
        for col in range(n):
            soma += A[lin][col]
        if soma != soma_base:
            return False

    for col in range(n):            # Todas as colunas devem ter essa soma
        soma = 0
        for lin in range(n):
            soma += A[lin][col]
        if soma != soma_base:
            return False

    soma_princ = 0
    soma_sec = 0
    for i in range(n):              # As diagonais devem ter essa soma
        soma_princ += A[i][i]
        soma_sec += A[i][n-i-1]
    if soma_princ != soma_base or soma_sec != soma_base:
        return False

    return True
```

Exercícios

Escalonamento

Adapte o código da função `imagem_pos_processamento` para que a mesma escalone os valores da matriz para o intervalo [0-255] após o arredondamento.

Convolução

Adapte o código da função `matriz_convolucao` para que a mesma seja capaz de lidar com kernels de tamanho 5

- Você consegue adaptar essa função para que ela seja capaz de fazer a convolução com qualquer kernel de tamanho ímpar?

Preencher uma Matriz em Espiral

Escreva um programa que leia um número inteiro positivo n e preencha uma matriz quadrada de tamanho $n \times n$ com os números de 1 até n^2 em ordem crescente, seguindo o formato espiral, no sentido horário, começando do canto superior esquerdo.

Assim, se $n = 3$, o resultado deve ser

```
1  1  2  3
2  8  9  4
3  7  6  5
```

E se $n = 4$, o resultado deve ser

```
1  1   2   3   4
2  12  13  14  5
3  11  16  15  6
4  10   9   8  7
```

Detectar pessoas famosas

Em um conjunto de n pessoas, nomeadas de 0 a $n-1$, sabemos quem conhece quem.

Essa informação é dada por meio de uma matriz de conhecimento `conhece`, $n \times n$, em que `conhece[i][j]` é igual a `1` se a pessoa `i` conhece a pessoa `j`, e `0` caso contrário.

Faça um programa que identifique se existe alguma pessoa famosa no grupo. Uma pessoa famosa é aquela que é conhecida por todos os outros e não conhece ninguém.