Inteligência Artificial

Instrutores
Ph.D. Professor Aluisio Igor Rego Fontes

Processamento de Imagens e Reconhecimento de Padrões

O que é processamento de imagens digitais?

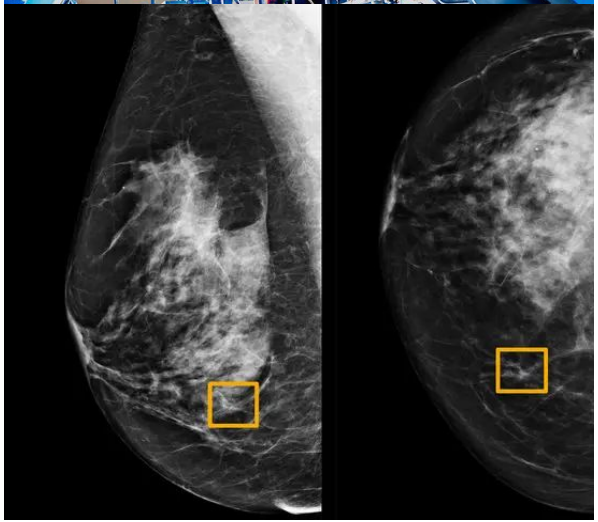
- O que é uma imagem digital?
  - Uma imagem pode ser definida como uma função bidimensional, onde *x* e *y* são coordenadas espaciais e a amplitude *f(x, y)* é a intensidade da imagem.

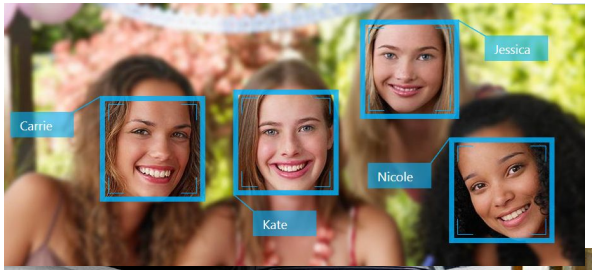
- Imagem e processamento
  - Uma imagem digital é composta por um número finito de elementos chamados *pixels*, sendo o termo mais amplamente utilizado.
  - O processamento de imagens digitais refere-se ao processamento de imagens por meio de um computador digital.

Motivação

- A visão é o nosso sentido mais avançado, imagens determinam um papel extremamente importante na percepção humana.
- Extrair informações valiosas e conseguir automatizar tarefas complexas junto com a melhora de eficiência em vários polos da sociedade.
- Já usamos processamento de imagens em diversos setores, já é uma realidade, a tendência é utilizarmos ainda mais.

Onde é utilizado





Introdução aos filtros

- Os filtros desempenham um papel fundamental no processamento de imagens, sendo ferramentas essenciais para realçar características específicas ou reduzir interferências indesejadas.

- Cada filtro, há uma função matemática associada, não iremos mostrar a função de todos pois foge do nosso escopo.

Introdução aos filtros

- Existem dois principais tipos de filtros:
  - Lineares
    - São úteis para realçar ou suavizar características em uma imagem. O filtro de média, por exemplo, é eficaz na redução de ruídos uniformes.
  - Não lineares
    - São ideais para preservar detalhes enquanto removem ruídos. O filtro de mediana, por exemplo, é eficiente na eliminação de ruídos impulsivos sem comprometer a nitidez da imagem.

Introdução aos filtros

- A principal biblioteca para tratamento de imagens em Python é o OpenCV (*Open Source Computer Vision Library)*.
  - Há outras bibliotecas, como Pillow, mas como nosso foco é o processamento de imagens, Opencv se torna mais eficiente pela liberdade que ela nos dá e por já ter integração com módulos de treinamento e aplicação dos modelos de aprendizado de máquina.
- OpenCV pode ser utilizado em Python, C++ e até MATLAB.
  - Nosso foco será na utilização dela em Python.

Filtros básicos

- Inversão de cores
  - Filtro de inverter as cores.

Imagem original

Imagem com filtro

Filtros básicos

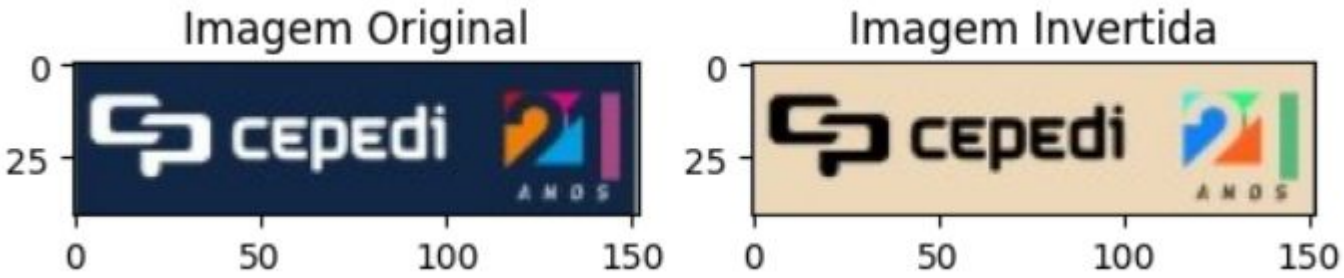
- Inversão de cores
  - Código em python utilizando o opencv

```
import cv2
from matplotlib import pyplot as plt
```

```
imagem = cv2.imread("cepedi_logo.jpg")

imagem_invertida = cv2.bitwise_not(imagem)

plt.subplot(121), plt.imshow(cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB)), plt.title("Imagem Original")
plt.subplot(122), plt.imshow(cv2.cvtColor(imagem_invertida, cv2.COLOR_BGR2RGB)), plt.title("Imagem Invertida")
plt.show()
```

Filtros básicos

- Inversão de cores
  -



Filtros básicos

- *Thresholding*
  - Transforma imagens coloridas em imagens preto e branco, onde cores acima de um determinado limite são mudadas para branco e abaixo são mudadas para preto.

Imagem original

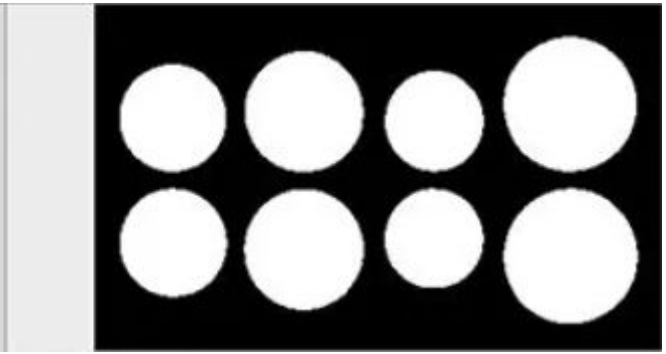


Imagem com filtro

Filtros básicos

- Thresholding
  - Código em python utilizando o opencv, para o threshold é importante que a imagem esteja em escala de cinza

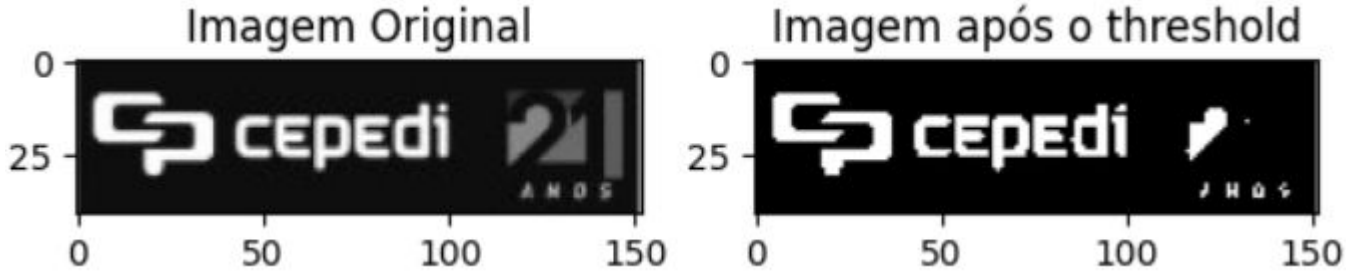
```
import cv2
from matplotlib import pyplot as plt


imagem = cv2.imread("cepedi_logo.jpg", cv2.IMREAD_GRAYSCALE)
```

```
_, imagem_threshold = cv2.threshold(imagem, 127, 255, cv2.THRESH_BINARY)
```

```
plt.subplot(121), plt.imshow(imagem, cmap="gray"), plt.title("Imagem Original")
plt.subplot(122), plt.imshow(imagem_threshold, cmap="gray"), plt.title("Imagem com Thresholding")
plt.show()
```

Filtros básicos

- Thresholding



Filtros básicos

- Rotação e espelhamento
  - Rotaciona a imagem em *X* graus ou espelha ela em algum eixo.

imagem original

90° graus sentido horário

90° graus sentido anti-horário

180° graus

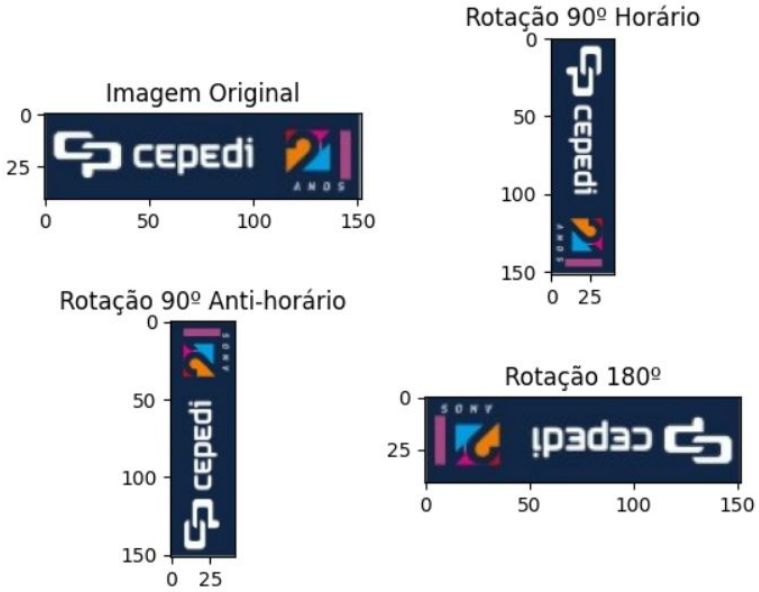
Filtros básicos

- Rotação e espelhamento
  - Código da rotação em python utilizando opencv

```
imagem = cv2.imread("cepedi_logo.jpg")

imagem_rotacionada_horario = cv2.rotate(imagem, cv2.ROTATE_90_CLOCKWISE)

imagem_rotacionada_antihorario = cv2.rotate(imagem, cv2.ROTATE_90_COUNTERCLOCKWISE)

imagem_rotacionada_180 = cv2.rotate(imagem, cv2.ROTATE_180)


plt.subplot(221), plt.imshow(cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB)), plt.title("Imagem Original")
plt.subplot(222), plt.imshow(cv2.cvtColor(imagem_rotacionada_horario, cv2.COLOR_BGR2RGB)), plt.title("Rotação 90º Horário")
plt.subplot(223), plt.imshow(cv2.cvtColor(imagem_rotacionada_antihorario, cv2.COLOR_BGR2RGB)), plt.title("Rotação 90º Anti-horário")
plt.subplot(224), plt.imshow(cv2.cvtColor(imagem_rotacionada_180, cv2.COLOR_BGR2RGB)), plt.title("Rotação 180º")
plt.show()
```

Filtros básicos

- Rotação e espelhamento



Filtros básicos

- Espelhamento

Imagem original

Imagem espelhada

Filtros básicos

- Espelhamento
  - Código em python do espelhamento utilizando opencv

```python
imagem = cv2.imread("cepedi_logo.jpg")

# Espelhar a imagem horizontalmente
imagem_espelhada_horizontal = cv2.flip(imagem, 1)

# Espelhar a imagem verticalmente
imagem_espelhada_vertical = cv2.flip(imagem, 0)

# Espelhar a imagem horizontalmente e verticalmente
imagem_espelhada_horizontal_vertical = cv2.flip(imagem, -1)
```

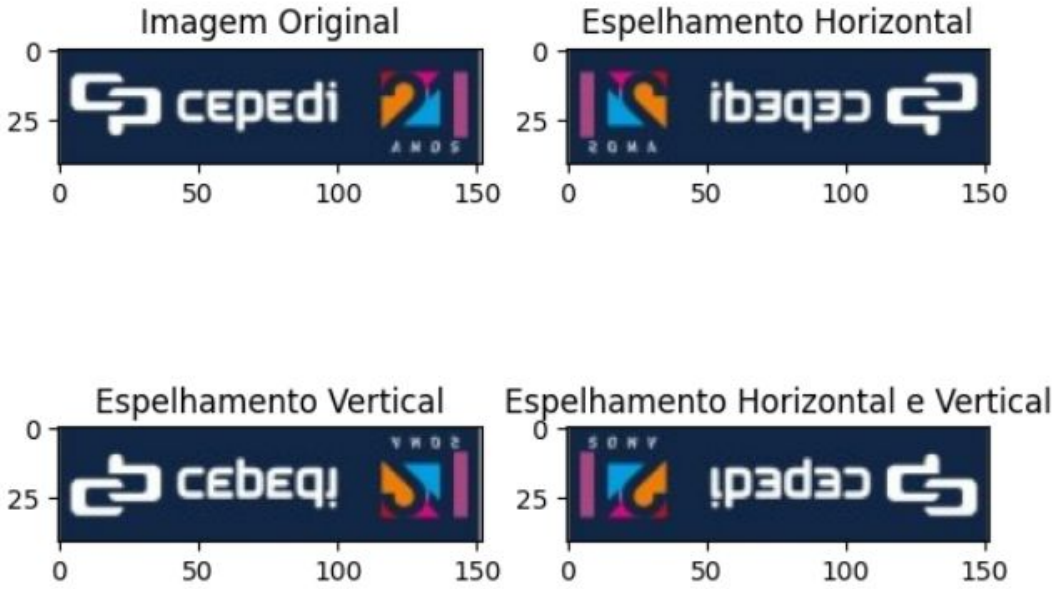
Filtros básicos

- Espelhamento
  - Código em python do espelhamento utilizando opencv

```
plt.subplot(221), plt.imshow(cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB)), plt.title("Imagem Original")
plt.subplot(222), plt.imshow(cv2.cvtColor(imagem_espelhada_horizontal, cv2.COLOR_BGR2RGB)), plt.title("Espelhamento Horizontal")
plt.subplot(223), plt.imshow(cv2.cvtColor(imagem_espelhada_vertical, cv2.COLOR_BGR2RGB)), plt.title("Espelhamento Vertical")
plt.subplot(224), plt.imshow(cv2.cvtColor(imagem_espelhada_horizontal_vertical, cv2.COLOR_BGR2RGB)), plt.title("Espelhamento Horizontal e Vertical")
plt.show()
```

Filtros básicos

- Espelhamento



Filtros lineares

- Em filtros lineares há dois que dominam
  - Filtro de média
    - Calcula a média dos valores dos pixels em uma região da imagem, resultando em um efeito de suavização.
  - Filtro gaussiano
    - Aplica uma função Gaussiana para ponderar os valores dos pixels, resultando em uma suavização, mas com um efeito mais controlado do que o filtro de média.

Filtros lineares

- Filtro de média

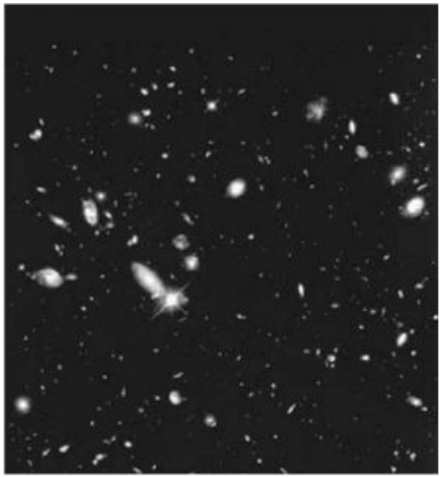


Imagem original

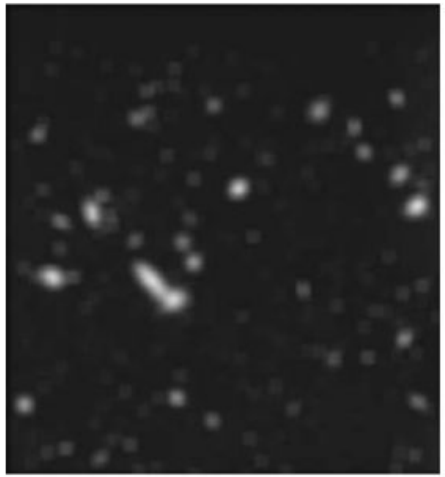


Imagem com filtro por média

Filtros lineares

- Filtro de média
  - Exemplo de código em python utilizando opencv

```
imagem = cv2.imread("cepedi_logo.jpg")


imagem_filtrada_media = cv2.blur(imagem, (3, 3))  # (3, 3) é o tamanho do kernel

# Exibir imagens usando matplotlib
plt.subplot(121), plt.imshow(cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB)), plt.title("Imagem Original")
plt.subplot(122), plt.imshow(cv2.cvtColor(imagem_filtrada_media, cv2.COLOR_BGR2RGB)), plt.title("Filtro por Média")
plt.show()
```

Filtros lineares

- Filtro de média
  - Exemplo de código em python utilizando opencv (Kernel 3x3)

Filtros lineares

- Filtro de média
  - Exemplo de código em python utilizando opencv (Kernel 5x5)

Filtros lineares

- Filtro de Gauss

Imagem original

Imagem com filtro Gaussiano

Filtros lineares

- Filtro de Gauss
  - Código em python utilizando o Opencv

```
imagem = cv2.imread("cepedi_logo.jpg")

imagem_suavizada_gaussiano = cv2.GaussianBlur(imagem, (5, 5), 1)  # (5, 5) é o tamanho do kernel, 1 é o desvio padrão

plt.subplot(121), plt.imshow(cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB)), plt.title("Imagem Original")
plt.subplot(122), plt.imshow(cv2.cvtColor(imagem_suavizada_gaussiano, cv2.COLOR_BGR2RGB)), plt.title("Filtro Gaussiano")
plt.show()
```

Filtros lineares

- Filtro de Gauss

Filtros não lineares

- Nos filtros não lineares, o filtro por mediana se sobressai

- Filtro de mediana
  - Substitui o valor de cada pixel pela mediana dos valores em sua vizinhança, sendo eficaz na remoção de ruídos impulsivos.

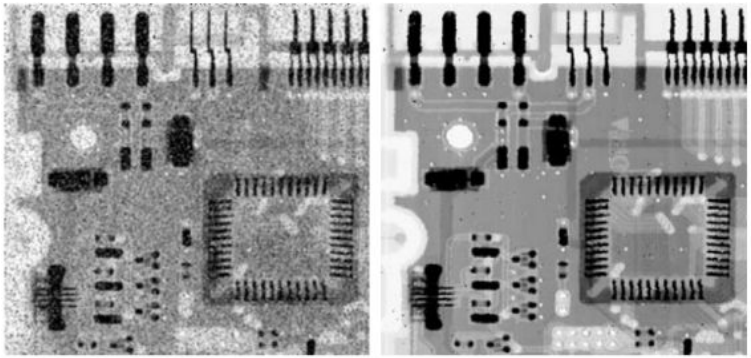


Imagem original Imagem com filtro de mediana

Filtros não lineares

- Filtro de mediana

Imagem original

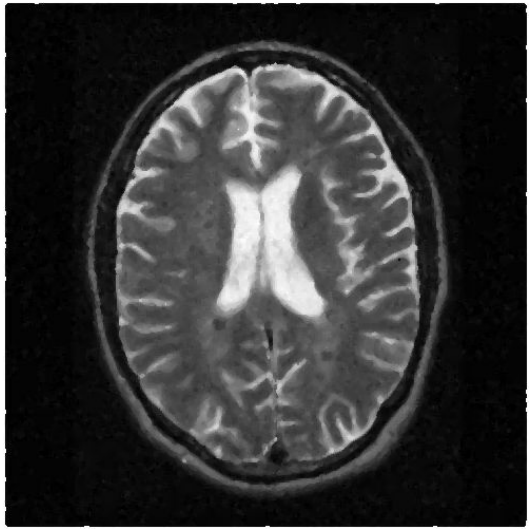


Imagem com filtro de mediana

Filtros não lineares

- Filtro de mediana
  - Código em python utilizando o opencv, ressaltando que o filtro por mediana é amplamente utilizado para retirar ruídos pontuais na imagem.

```python
import cv2
from matplotlib import pyplot as plt

imagem = cv2.imread("cepedi_logo.jpg")

imagem_suavizada_mediana = cv2.medianBlur(imagem, 3)  # O segundo parâmetro é o tamanho do kernel

plt.subplot(121), plt.imshow(cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB)), plt.title("Imagem Original")
plt.subplot(122), plt.imshow(cv2.cvtColor(imagem_suavizada_mediana, cv2.COLOR_BGR2RGB)), plt.title("Filtro de Mediana")
plt.show()
```

Filtros não lineares

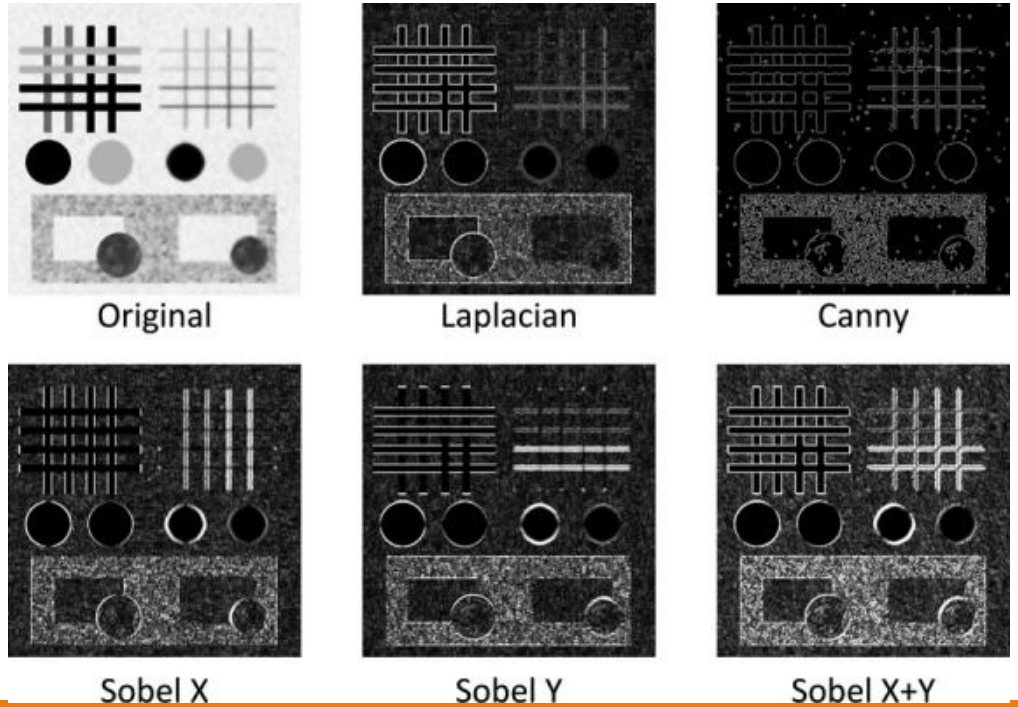
- Filtro de mediana

Outros tipos de filtros

- Filtros de borda
  - Filtro laplaciano
    - Calcula a segunda derivada da imagem, realçando áreas de rápida mudança de intensidade.
  - Filtro sobel
    - Utiliza operadores de convolução para calcular as derivadas em diferentes direções (horizontal e vertical), destacando bordas em várias orientações.
    - Podemos calcular, também, a combinação vertical e horizontal do filtro, conhecido como magnitude do gradiente.
  - Filtro canny
    - Utiliza um método de múltiplos estágios para identificar as bordas, combinando suavização de imagem, cálculo de gradientes e aplicação de limiares.

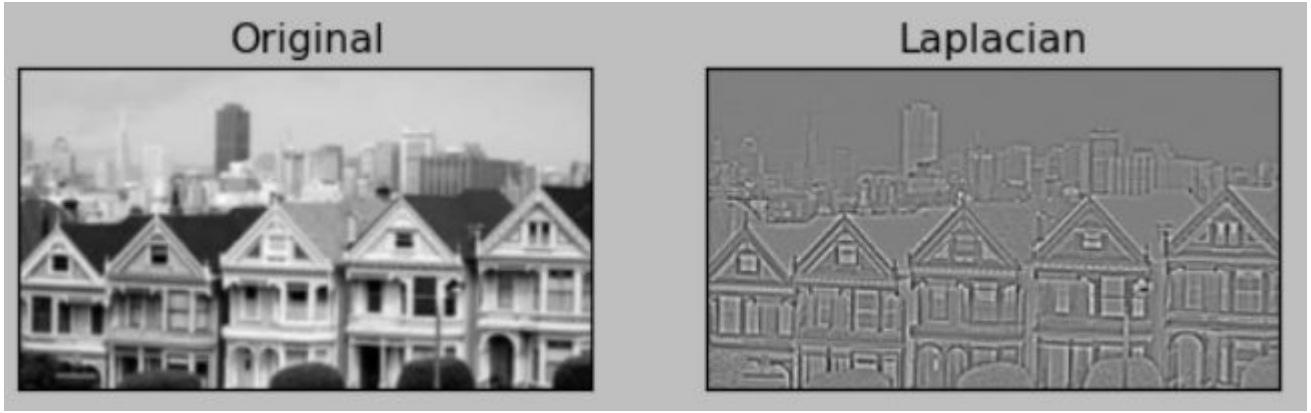
Filtros de borda

- Filtros



Filtros de borda

- Filtro laplaciano



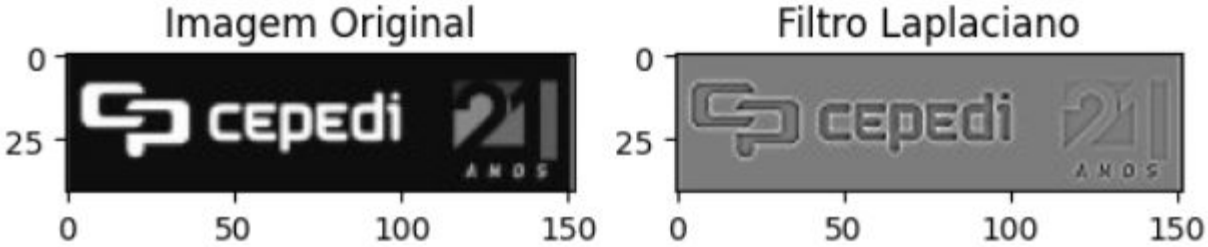
Filtros de borda

- Filtro laplaciano
  - Código em python utilizando o opencv

```
imagem = cv2.imread("cepedi_logo.jpg", cv2.IMREAD_GRAYSCALE)

imagem_realcada_laplaciano = cv2.Laplacian(imagem, cv2.CV_64F)

# Exibir imagens usando matplotlib
plt.subplot(121), plt.imshow(imagem, cmap="gray"), plt.title("Imagem Original")
plt.subplot(122), plt.imshow(imagem_realcada_laplaciano, cmap="gray"), plt.title("Filtro Laplaciano")
plt.show()
```

Filtros de borda

- Filtro laplaciano



Filtros de borda

- Filtro Sobel

Imagem original

Imagem com filtro sobel (x e y)

Filtros de borda

- Filtro Sobel
  - Código em python utilizando Opencv

```
imagem = cv2.imread("cepedi_logo.jpg", cv2.IMREAD_GRAYSCALE)

sobel_x = cv2.Sobel(imagem, cv2.CV_64F, 1, 0, ksize=3)
sobel_y = cv2.Sobel(imagem, cv2.CV_64F, 0, 1, ksize=3)
sobel_xy = cv2.Sobel(imagem, cv2.CV_64F, 1, 1, ksize=3)

magnitude_gradiente = np.sqrt(sobel_x**2 + sobel_y**2)
```

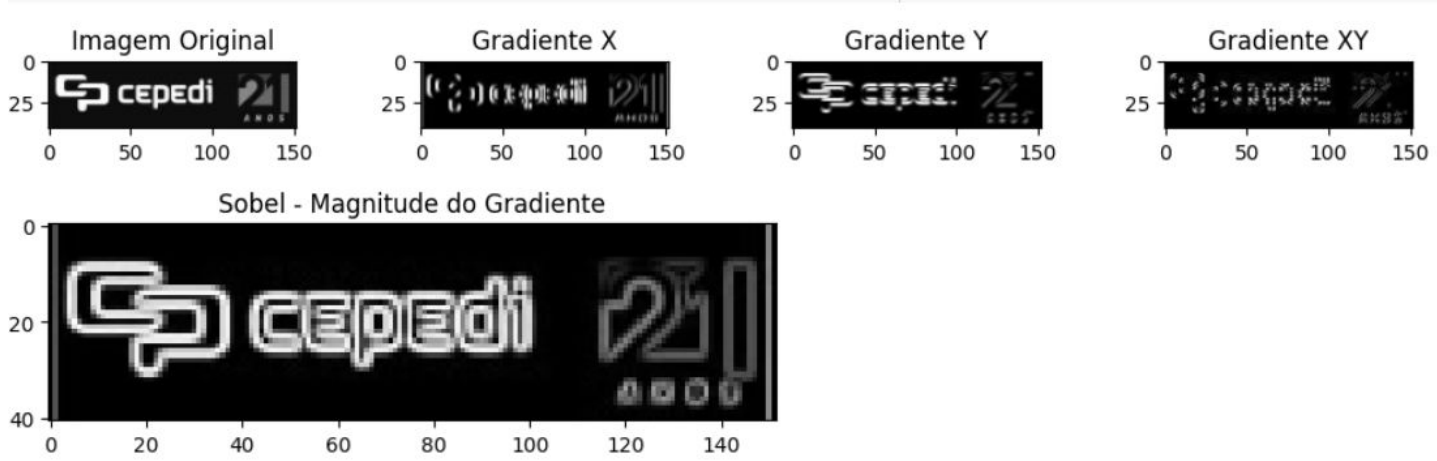
Filtros de borda

- Filtro Sobel
  - Código em python utilizando Opencv

```
plt.subplots_adjust(wspace=0.5, hspace=0.5)

plt.subplot(141), plt.imshow(imagem, cmap="gray"), plt.title("Imagem Original")
plt.subplot(142), plt.imshow(np.abs(sobel_x), cmap="gray"), plt.title("Gradiente X")
plt.subplot(143), plt.imshow(np.abs(sobel_y), cmap="gray"), plt.title("Gradiente Y")
plt.subplot(144), plt.imshow(np.abs(sobel_xy), cmap="gray"), plt.title("Gradiente XY")

plt.show()

plt.imshow(magnitude_gradiente, cmap="gray")
plt.title("Sobel - Magnitude do Gradiente")
plt.show()
```

Filtros de borda

- Filtro Sobel



Filtros de borda

- Filtro Canny

Imagem original

Imagem com filtro canny

Filtros de borda

- Filtro Canny
  - Código em python utilizando opencv

```
import cv2
from matplotlib import pyplot as plt

imagem = cv2.imread("cepedi_logo.jpg", cv2.IMREAD_GRAYSCALE)

bordas_canny = cv2.Canny(imagem, 50, 150)  # 50 é o limite inferior e 150 é o limite superior

plt.subplot(121), plt.imshow(imagem, cmap="gray"), plt.title("Imagem Original")
plt.subplot(122), plt.imshow(bordas_canny, cmap="gray"), plt.title("Filtro Canny")
plt.show()
```

Filtros de borda

- Filtro Canny
  - Código em python utilizando opencv

Aplicações práticas

- Melhoramento de Imagens
  - O intuito do exemplo não é para entender tudo o que está acontecendo, mas apenas para exemplificar um caso.
  - A imagem original é um raio-x do corpo inteiro.

- Reconhecimento de padrões (Nosso foco)
  - No contexto do reconhecimento de padrões, o uso de filtros é crucial no pré-processamento de imagens. Eles ajudam a destacar características relevantes, facilitando a identificação de padrões por algoritmos de aprendizado de máquina.
  - Iremos falar sobre esse assunto em aulas posteriores.

Melhoramento de imagens

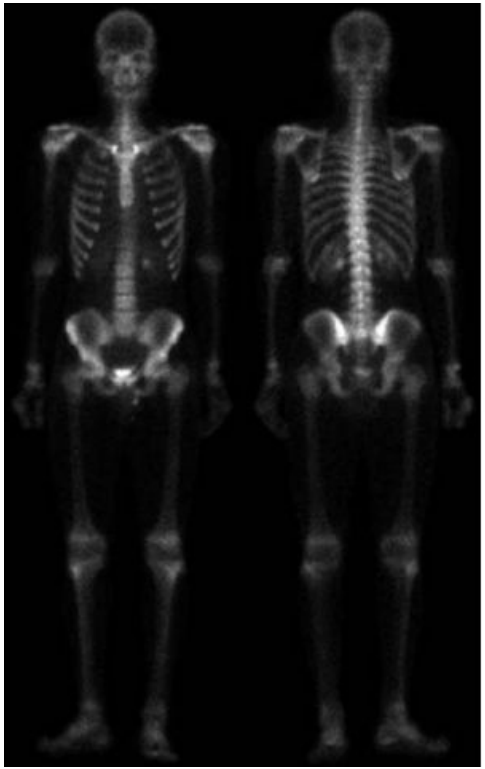


Imagem original

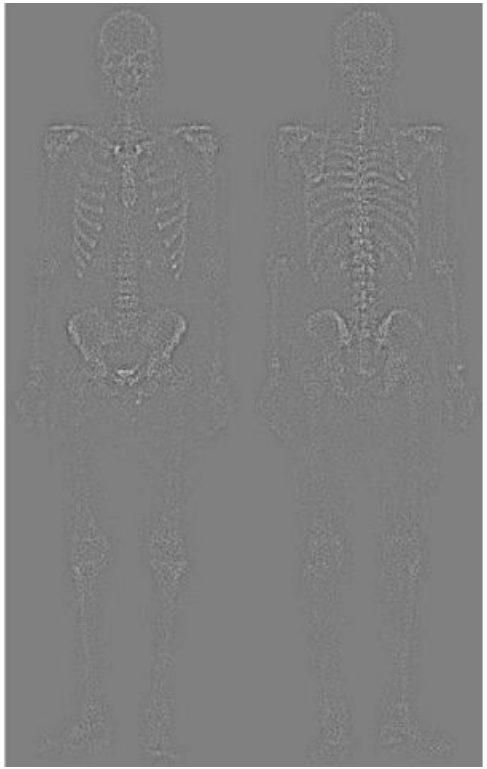


Imagem com filtro laplaciano

Melhoramento de imagens

Imagem original + imagem com filtro laplaciano. (Imagem 1)

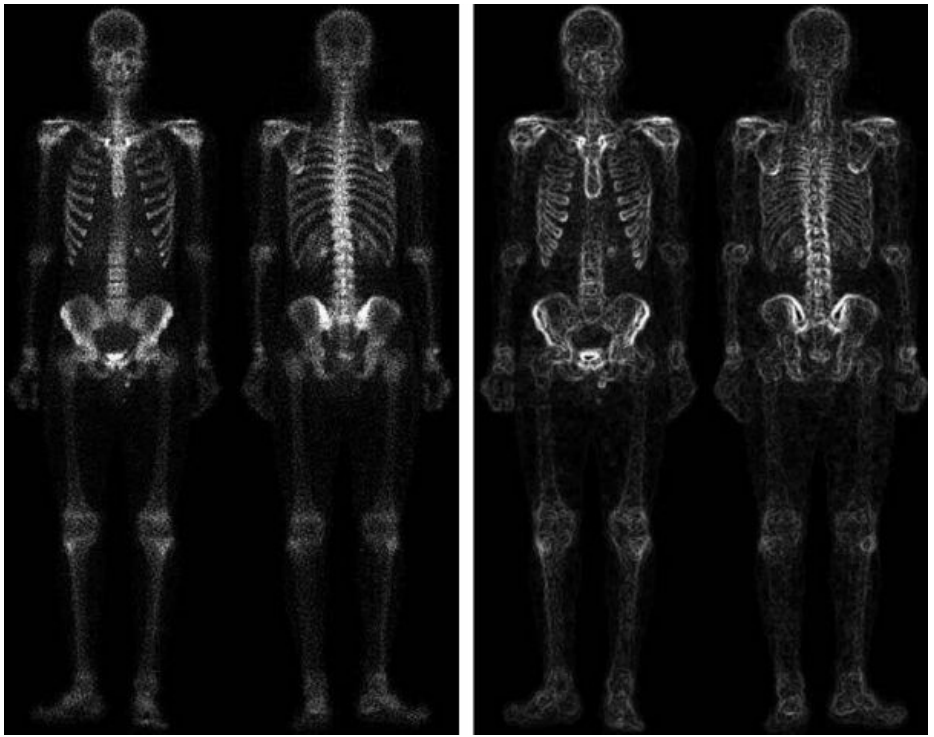


Imagem original com filtro sobel

Melhoramento de imagens

Imagem do sobel com filtro de média (Imagem 2)

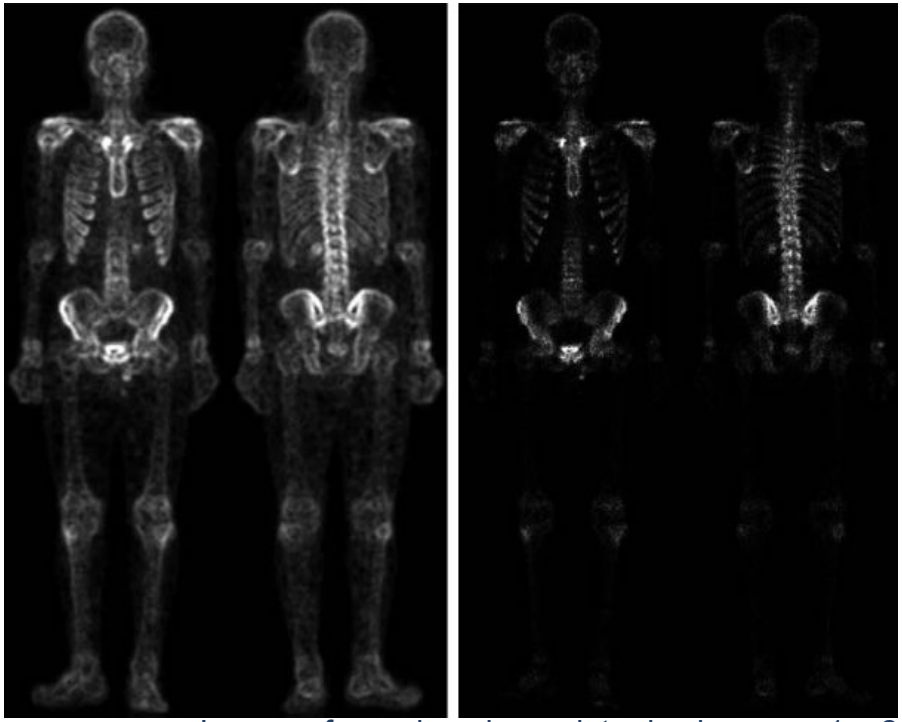
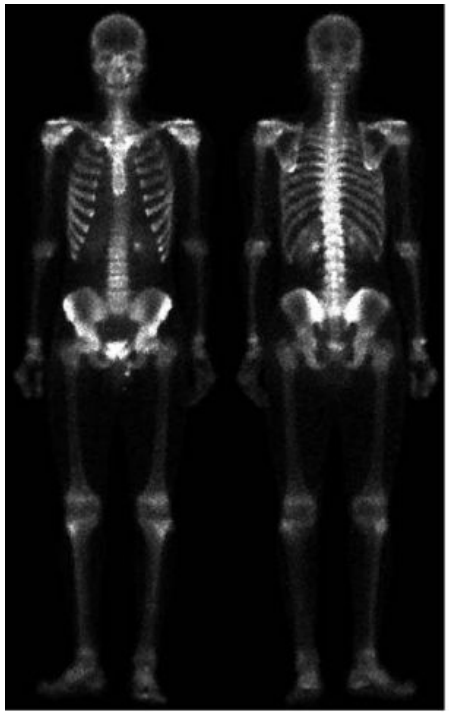


Imagem formada pelo produto das imagens 1 e 2 (máscara)

Melhoramento de imagens

Imagem original + máscara (Imagem 3)



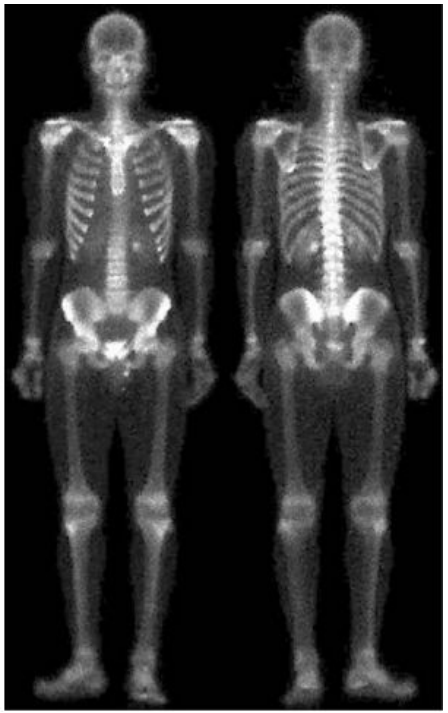


Imagem obtida após exponenciação da imagem 3.

Melhoramento de imagens

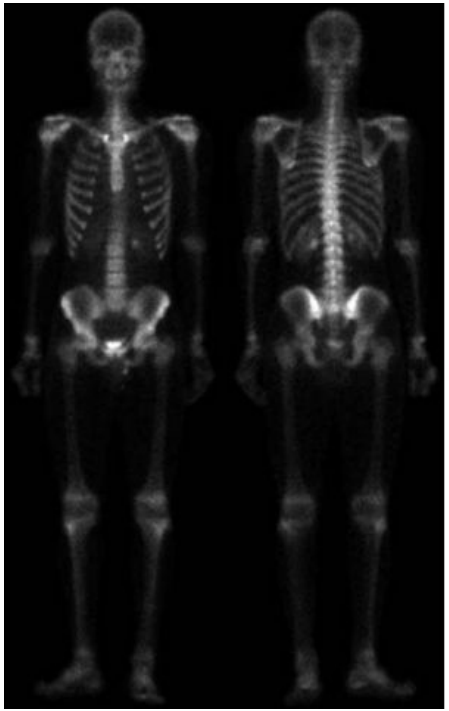


Imagem original

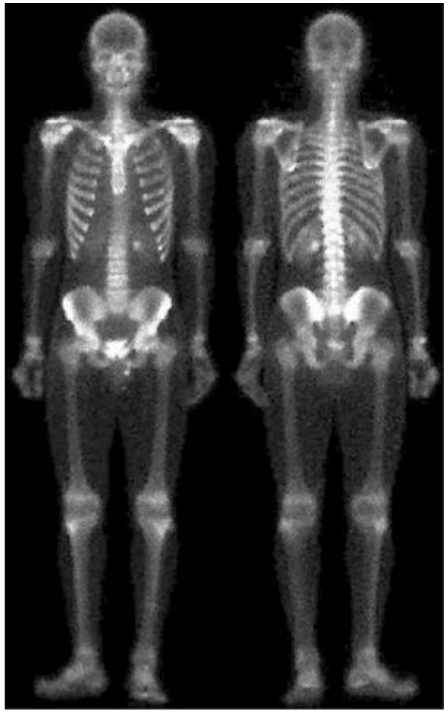


Imagem final


OBRIGADO!