Morfologia em gray scale

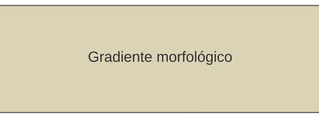
A morfologia matemática não se limita a imagens binárias e também pode ser aplicada em imagens de intensidade, como imagens em escala de cinza. Neste caso, os operadores lógicos AND e OR são substituídos pelos operadores aritméticos MIN e MAX, respectivamente.

Ao contrário da morfologia binária, os elementos estruturantes na morfologia em tons de cinza são definidos como funções 2D de valor real, em vez de conjuntos de pontos. Esses valores podem ser negativos ou zero, e, ao contrário da convolução linear, os elementos zero na morfologia em tons de cinza geralmente contribuem para o resultado.

```
img = cv2.imread('./images/street.jpeg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# kernel
kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
# opening
opening = cv2.morphologyEx(gray, cv2.MORPH_OPEN, kernel)

# plotting
fig, axs = plt.subplots(1, 3, figsize=(20, 20))
axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
axs[0].set_title('Original')
axs[1].imshow(gray, cmap='gray')
axs[1].set_title('Grayscale')
axs[2].imshow(opening, cmap='gray')
axs[2].set_title('Opening')
for ax in axs:
    ax.axis('off')
plt.show()
```

```
import cv2
import numpy as np
import matplotlib.pyplot as plt


img = cv2.imread('./images/street.jpeg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# kernel
kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
# opening
opening = cv2.morphologyEx(gray, cv2.MORPH_CLOSE, kernel)

# plotting
fig, axs = plt.subplots(1, 3, figsize=(20, 20))
axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
axs[0].set_title('Original')
axs[1].imshow(gray, cmap='gray')
axs[1].set_title('Grayscale')
axs[2].imshow(opening, cmap='gray')
axs[2].set_title('Closing')
for ax in axs:
    ax.axis('off')
plt.show()
```


```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('./images/street.jpeg', cv2.IMREAD_GRAYSCALE)
ret, thresh = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# kernel
kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
# opening
opened = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)

# plotting
fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].imshow(thresh, cmap='gray')
axs[0].set_title('Binary Image')
axs[1].imshow(opened, cmap='gray')
axs[1].set_title('Opened Image')
plt.show()
```



Abertura

Uma abertura binária é uma erosão seguida de uma dilatação com o mesmo elemento estruturante. O efeito principal é a eliminação de estruturas menores que o elemento estruturante na primeira etapa (erosão), seguida de uma suavização das estruturas restantes pela dilatação subsequente. Isso resulta em um encolhimento e posterior crescimento das estruturas, eliminando as pequenas.

Morfologia Matemática II

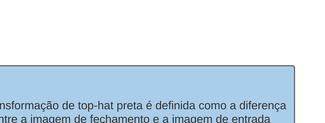
Dilatação e erosão são frequentemente combinadas em operações compostas devido à sua semidualidade. As operações morfológicas mais utilizadas na prática são "abertura" e "fechamento"

As operações de abertura e fechamento são idempotentes, ou seja, aplicá-las várias vezes não altera o resultado. Além disso, abrir e fechar são complementares, ou seja, aplicar uma operação em primeiro plano é equivalente a aplicar a outra operação no fundo.

Fechamento

Um fechamento binário é uma dilatação seguida de uma erosão com o mesmo elemento estruturante. Ele fecha buracos e fissuras nas estruturas de primeiro plano que são menores que o elemento estruturante.

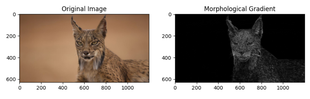
```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('./images/street.jpeg', cv2.IMREAD_GRAYSCALE)
ret, thresh = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# kernel
kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
# closing
closed = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)

# plotting
fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].imshow(thresh, cmap='gray')
axs[0].set_title('Binary Image')
axs[1].imshow(closed, cmap='gray')
axs[1].set_title('Closed Image')
plt.show()
```



Gradiente morfológico

Na morfologia matemática e no processamento digital de imagens, o gradiente morfológico é uma operação que consiste em calcular a diferença entre a imagem resultante da dilatação e a imagem resultante da erosão da imagem original.

O resultado é uma imagem que indica a intensidade do contraste na vizinhança próxima de cada pixel da imagem original. Essa informação é útil para detecção e segmentação de bordas.




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


img = cv2.imread('./images/cat.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# box kernel
kernel = np.ones((3,3), np.uint8)
# gradient operation
gradient = cv2.morphologyEx(gray, cv2.MORPH_GRADIENT, kernel)

# plotting
fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
axs[0].set_title('Original Image')
axs[1].imshow(gradient, cmap='gray')
axs[1].set_title('Morphological Gradient')
plt.show()
```

TOP-HAT

A transformação top-hat é uma operação na morfologia matemática e processamento digital de imagens que extrai pequenos elementos e detalhes de imagens. Existem dois tipos de transformação top-hat.

O tamanho ou largura dos elementos extraídos pelas transformações top-hat podem ser controlados pela escolha da função estruturante. Quanto maior o elemento estruturante, maiores serão os elementos extraídos.

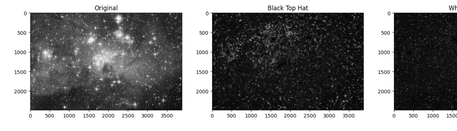
A transformação top-hat branca é definida como a diferença entre a imagem de entrada e sua abertura por algum elemento estruturante.

A transformação top-hat branca retorna uma imagem contendo objetos ou elementos de uma imagem de entrada que são menores que o elemento estruturante e mais brilhantes que seus arredores, ou seja, locais onde o elemento estruturante não se encaixa. Essa operação é útil para realçar pequenos detalhes ou estruturas em uma imagem que possam ter sido perdidos durante a etapa de pré-processamento ou que sejam de interesse específico para análise.

A transformação de top-hat preta é definida como a diferença entre a imagem de fechamento e a imagem de entrada

A transformação top-hat preta retorna uma imagem contendo objetos ou elementos de uma imagem de entrada que são menores que o elemento estruturante e mais escuros que seus arredores.

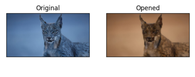
```
import cv2
import matplotlib.pyplot as plt


img = cv2.imread('./images/image.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# kernel size defines size of the objects being recovered
kernel_size = 25
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (kernel_size, kernel_size))

# black top hat operation
black_tophat = cv2.morphologyEx(img, cv2.MORPH_BLACKHAT, kernel)

# white top hat operation
white_tophat = cv2.morphologyEx(img, cv2.MORPH_TOPHAT, kernel)

# plotting images
fig, axs = plt.subplots(1, 3, figsize=(10, 10))
axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
axs[0].set_title('Original')
axs[1].imshow(black_tophat, cmap='gray')
axs[1].set_title('Black Top Hat')
axs[2].imshow(white_tophat, cmap='gray')
axs[2].set_title('White Top Hat')
plt.show()
```



Imagens Coloridas

A morfologia matemática também pode ser aplicada em imagens coloridas, mas é tratada como um caso especial da morfologia em níveis de cinza. Os operadores são aplicados separadamente nas componentes R, G e B (no modelo RGB) e recombinados posteriormente. A dificuldade de ordenação de cores é o principal desafio nesse contexto. É importante definir uma métrica para o espaço de cor usado e escolher o espaço de cor apropriado.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt


img = cv2.imread('./images/cat.jpg')

# splitting the BGR image into three channels
b, g, r = cv2.split(img)

# kernel
kernel = np.ones((15,15),np.uint8)

# opening on each color channel
r_open = cv2.morphologyEx(r, cv2.MORPH_OPEN, kernel)
g_open = cv2.morphologyEx(g, cv2.MORPH_OPEN, kernel)
b_open = cv2.morphologyEx(b, cv2.MORPH_OPEN, kernel)

# merging channels on image
opened_img = cv2.merge([r_open, g_open, b_open])

# plotting
plt.subplot(121),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(opened_img),plt.title('Opened')
plt.xticks([]), plt.yticks([])
plt.show()
```



Esqueletização

Esqueletização (ou "skeletonization", em inglês) de uma imagem é um processo que extrai a forma estrutural ou topológica de um objeto em uma imagem, representando-a como uma linha fina que representa o "esqueleto" do objeto. O resultado da esqueletização é uma imagem binária, em que cada pixel representa uma parte do esqueleto do objeto. Basicamente se erode a imagem com base em um kernel até que cada objeto se torne o menor possível antes de sumir, formando uma espécie de esqueleto

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

#
img = cv2.imread('./images/cat.jpg', 0) # Load as grayscale
_, thresh = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

# cross kernel will be used
kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))

# skeletonizing
skel = np.zeros(img.shape, np.uint8)
done = False

while not done:
    eroded = cv2.erode(thresh, kernel)
    temp = cv2.dilate(eroded, kernel)
    temp = cv2.subtract(thresh, temp)
    skel = cv2.bitwise_or(skel, temp)
    thresh = eroded.copy()

    zeros = img.size - cv2.countNonZero(thresh)
    if zeros == img.size:
        done = True

# plotting
plt.subplot(121),plt.imshow(img, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(skel, cmap='gray'),plt.title('Skeletonized')
plt.xticks([]), plt.yticks([])
plt.show()
```