

Filtros Digitais

Estude a classe ImageAccess

A classe de ImageAccess contém métodos úteis como getRow, getNeighborhood, putColumn.

Por exemplo, o método getNeighborhood devolve um arranjo n*n centrada em uma determinada posição (x,y).

ImageAccess também contém operações aritméticas básicas:

Operações	Sintaxe	Significado
Valor absoluto	image.abs();	Image <- abs(image)
Raíz quadrada	image.sqrt();	Image <- sqrt(image)
Quadrado	image.pow(2.0);	Image <- image ^ 2
Adição	image.add(image1, image2);	Image <- image1 + image2
Subtração	image.subtract(image1, image2);	Image <- image1 - image2
Multiplicação	image.multiply(image1, image2);	Image <- image1 * image2
Duplicar	image = image1.duplicate();	Image <- image1
Mostra a imagem	image.show("title of the window");	Mortra uma imagem em uma nova janela

1. Filtro Detecção de bordas

A máscara de detecção de bordas vertical é definida abaixo:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

1.1 Entendendo o filtro de detecção de bordas verticais

Leia e entenda o código de um detector de bordas vertical em FilteringSession.java.

Há duas versões:

- detectEdgeVertical_NonSeparable (): versão Não-separável;

- `detectEdgeVertical_Separable ()`: versão Separável.

Aplique esta função às imagens `mit.tif` e `octagon.tif` selecionando a operação certa.

1.2 Escrevendo o detector de bordas horizontais

Escreva um método que implementa a versão separável do detector de bordas horizontal. O nome do método é `detectEdgeHorizontal_Separable()` em `FilteringSession.java`.

Aplique esta função às imagens `mit.tif` e `octagon.tif` selecionando a operação certa e clicando em "Run Student Solution".

1.3 Comparação

Compare os resultados das duas versões em termos de tempo de computação e conteúdo de imagem. Eles diferem ou eles são exatamente o mesmo? Complete o `relatorio.doc`. Aplique este detector de bordas vertical e horizontal em `africa.tif`. Converta os resultados em 8-bits e insira no `relatorio.doc`.

2. Médias-móveis (5x5)

O filtro de médias-móveis 5x5 substitui um pixel por sua média em uma 5x5 janela centrada. Há vários modos para implementar um filtro de médias-móveis; todos dão os mesmos resultados, mas alguns são mais eficientes que outros. Aqui, nós analisaremos três versões da implementação que nós chamamos:

- Não-separável: A janela 5x5 é corrida para cada pixel da imagem;
- Separável: A rotina de média-móvel é implementada em 1D. É aplicado a todas as linhas e colunas;
- Recursivo: Como o prévio, este método é também separável, mas a rotina computa média 1D usando um algoritmo recursivo mais eficiente.

2.1 Escrevendo a versão não-separável 5x5

Escreva um método que implementa os 5x5, versão não-separável de um

filtro de médias-móveis.

O nome do método é `doMovingAverage5_NonSeparable()`.

2.2 Escrevendo a versão separável 5x5

Escreva um método que implementa a versão separável de um filtro de médias-móveis 5x5. O nome do método é `doMovingAverage5_Separable()`.

Crie uma rotina 1D chamada `doAverage5()`.

Preste atenção à própria implementação das condições de fronteira reflexiva na rotina 1D.

2.3 Escrevendo a versão separável 5x5 recursiva

Escreva um método que implementa a versão 5x5, separável e de maneira recursiva de um filtro de médias-móveis. O nome do método é `doMovingAverage5_Recursive()`.

Crie uma rotina 1D chamada `doAverage5_recursive()` que implementa uma média-móvel, de maneira recursiva.

Condições de fronteiras reflexivas são aplicadas.

3. Aplicações do operador de suavização

O filtro de médias-móveis 5x5 substitui um pixel por sua média em uma janela centrada 5x5. Há vários modos para implementar um filtro de médias-móveis; todos dão os mesmos resultados, mas alguns são mais rápidos que outros.

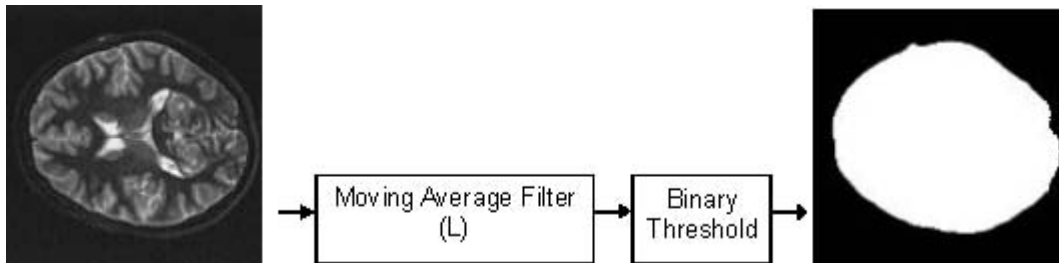
Aqui, nós analisaremos três versões da implementação que nós chamamos:

- Não-separável: A janela 5x5 é transcorrida para cada pixel da imagem;
- Separável: A rotina de mudança-média é implementada em 1D. É aplicado a todas as linhas e todas as colunas;
- Recursivo: Como o anterior, este método é também separável, mas a 1D rotina computa o usando comum um algoritmo mais eficiente.

3.1 Segmentando

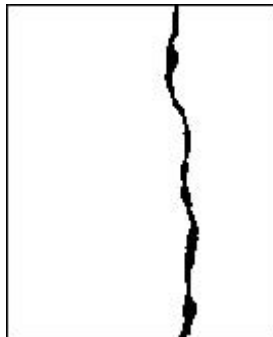
Usando o diagrama descrito abaixo, segmente a imagem `brain14.tif` em

áreas claras e escuras. Selecione um comprimento de janela apropriado (L) para o filtro de médias-móveis, e escolha um limiar binário apropriado (T) usando o comando “Threshold” do menu de Image/Adjust. Complete o relatorio.doc.



3.2. Supressão de fundo

Encontre um procedimento simples para suprimir o fundo que usa o operador suavizador sem imagem de fundo. A meta é extrair só o dendrito através de “thresholding”. Aplique o procedimento ao dendrite.tif e complete o relatorio.doc.



4. Detector de bordas Sobel

Escreva o método doSobel() que produz a seguinte saída:

$$c_s = \sqrt{(G_x\{f(x,y)\})^2 + (G_y\{f(x,y)\})^2}$$

onde:

- $f(x,y)$ é a imagem de entrada
- G_x , G_y são os gradientes vertical e horizontal respectivamente, suas máscaras são:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \text{ e } G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Se é um filtro separável, o filtro se torna mais eficiente, então implemente de modo separável. Sugestão: confira a imagem de G_x , G_y usando o método `show("título")` de `ImageAccess`. Aplique esta função às imagens `mit.tif` e `octagon.tif`. Complete o `relatorio.doc`.

5. Desafio: escrevendo filtro de médias-móveis recursivo com o tamanho da janela variável.

Escreva um método que implementa um filtro de médias-móveis $L \times L$ que é mais rápido que a versão separável que você desenvolveu previamente (em `média1D`, e para uma imagem suficientemente grande, você deveria ser capaz de acessar não mais que dois pixels de entrada por pixel de saída).