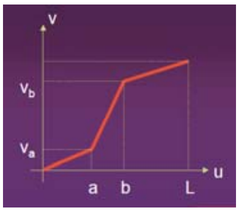
Filtragem de Imagens

Prof. Sérgio S Furuie

Filtros

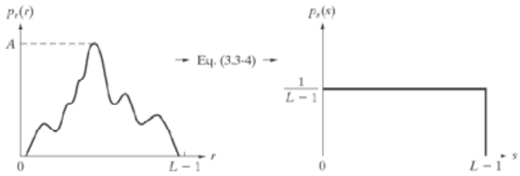
- Operadores pontuais (baseado apenas na intensidade)
  - Alteração de contraste
  - Equalização do histograma
- Operadores que consideram a vizinhança
  - Filtros globais
    - Filtros lineares
      - Convolução
      - Baseado em resposta em frequencia
    - Filtros não lineares: mediana, ad-hocs,
  - Filtros adaptativos

Alteração de contraste



$$v = \begin{cases} \alpha.u \\ v_a + \beta(u-a) \\ v_b + \gamma(u-b) \end{cases} \quad \text{se} \begin{bmatrix} 0 \le u < a \\ a \le u < b \\ b \le u < L \end{bmatrix}$$

Equalização de histograma



a b

FIGURE 3.18 (a) An arbitrary PDF. (b) Result of applying the transformation in Eq. (3.3-4) to all intensity levels, r. The resulting intensities, s, have a uniform PDF, independently of the form of the PDF of the r's.

$$p_r(r_k) = \frac{n_k}{M.N}$$

$$s_k = (L-1).\sum_{j=0}^{k} p_r(r_j)$$

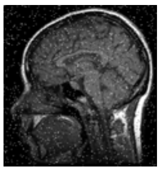
$$k = 0,1,\ldots,L-1$$

Exemplo: ImageJ Enhance Contrast (equalization). Use: fibroblast...jpg

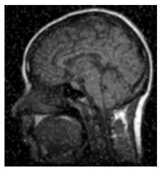
FILTRO NO DOMÍNIO DO ESPAÇO

Conceito: filtro e convolução

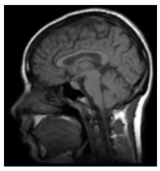
- Filtragem é uma operação feita sobre um sinal ou imagem para obter um outro sinal ou imagem
- Filtro linear e invariante pode ser implementado por convolução
- Convolução pode ser executado rapidamente no domínio da frequência por FFT



Ruído salt pepper



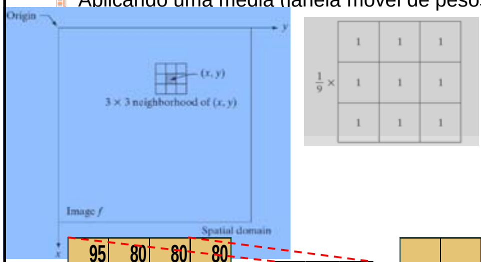
Filtro Média, r=1



Filtro Mediana,r=1 (Não-linear)

Filtragem de imagens

- Aplicando uma média (janela móvel de pesos)



95	80	80	80
95	90	90	105
150	20	100	130
220	150	160	150

*

1	1	1
1	1	1
1	1	1

=>

	89		

Filtros digitais: SIPSF

- Filtragem no domíno do espaço (Convolução)

$$g(x', y') = \iint h(x'-x, y'-y).f(x,y).dx.dy$$

g (x',y') = (h * f) (x',y')

$$g_{ij} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} h(i-m, j-n).f(m,n)$$

  - Exemplos
  - Características do filtro?
- Filtragem no domínio da frequencia (DFT)

Convolução 2D

$$g(x,y) = \sum_{r=-a}^{a} \sum_{c=-b}^{b} w(r,c).f(x-r, y-c)$$

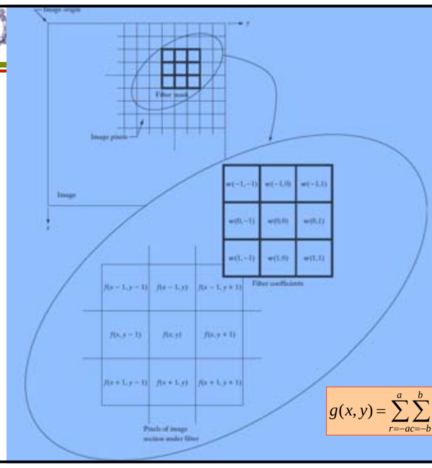
95	80	80	80
95	90	90	105
150	20	100	130
220	150	160	150

*

0	1	0
1	-4	1
0	1	0

=>

	-75		



$$g(x,y) = \sum_{r=-a}^{a} \sum_{c=-b}^{b} w(r,c).f(x-r, y-c)$$

Implementação em pseudo-código

$$g(x,y) = \sum_{r=-a}^{a} \sum_{c=-b}^{b} w(r,c).f(x-r, y-c)$$

Entradas:

f[R linhas][C colunas]; w[A linhas][B colunas] (A e B:ímpares)
Desconsidere as bordas

Algoritmo:

1. Sejam: a=A/2; b=B/2
2. for (x=a; x < (R-a); x++)
   for (y=b; y < (C-b); y++)
   - g[x][y]=0;
   - for (r=-a; r <= a; r++)
     - for (c=-b; c <= b; c++)
     - g[x][y]=g[x][y] + w[r+a][c+b] * f[x-r][y-c]
     - end for
   - end for
   end for
3. end for

Implementação em java

$$g(x,y) = \sum_{r=-a}^{a} \sum_{c=-b}^{b} w(r,c).f(x-r, y-c)$$

Entradas:

f[R linhas][C colunas]; w[A linhas][B colunas] (A e B:ímpares)
Desconsidere as bordas

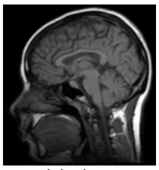
Algoritmo:

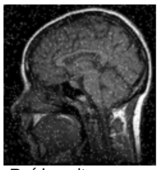
1. int a=A/2; int b=B/2
2. for (int x=a; x < (R-a); x++)
   for (int y=b; y < (C-b); y++)
   {g[x][y]=0;
   for (int r=-a; r <= a; r++)
   - for (int c=-b; c <= b; c++)
   - g[x][y]=g[x][y] + w[r+a][c+b] * f[x-r][y-c];
   }

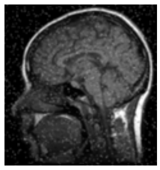
Exemplo no ImageJ: filtro

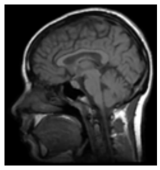
- Vamos implementar um plugin no ImageJ que:
  - Filtre os ruídos substituindo pela média dos 8-vizinhos (média entre 9 pixels) (ver plugin media_sf.java)
  - Filtre os ruídos substituindo pela mediana
- Como tratar as bordas?
  - Veja a classe ImageAccess.java
  - Ou tratar usando um modelo:
    - Supondo valores zero fora da imagem
    - Valores espelhados p/ pixels fora da imagem (ImageAccess)
    - Tratar a exceção considerando apenas os pixels da imagem. Necessário considerar os sub-kernels adaptando os valores dos pesos, pois a normalização fica diferente

Filtro adequado !


original

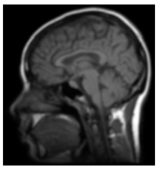

Ruído salt pepper

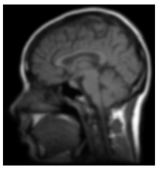

Filtro Média, r=1


Filtro Mediana, R=1

Imagens: efeito de filtro (suavização)


Original
238 x 253, 8 bits
58kB


Filtrado, média, r=2


Filtrado, média, 2 x (r=2)

Formalizando convolução: 2D, contínuo

Definição para funções contínuas:

$$g(x,y) = h(x,y) * f(x,y)$$

$$g(x,y) \stackrel{\text{def}}{=} \iint_{-\infty}^{\infty} f(x,y).h(x-x', y-y')\,dx'dy'$$

Pode-se provar:

$$g(x,y) = \iint_{-\infty}^{\infty} h(x,y).f(x-x', y-y')\,dx'dy'$$

$$G(u,v) = H(u,v).F(u,v)$$

Na qual G, H e F são as transformadas de Fourier 2D de g, h e f respectivamente.

Como faríamos a convolução se h e f fossem 1000 x 1000 cada ?

FILTROS NÃO-LINEARES E LOCAIS

Filtros não-lineares: mediana

$$g(n,m) = mediana\{f(m-k, n-l), (k,l) \in W\}$$

Filtragem de Imagens

Filtros adaptativos

- Wiener adaptativo (Lee)
  - Preserva bordas
  - Atenua ruído de regiões homogêneas

$g(x,y) = (1-\alpha).f(x,y) + \alpha.\bar{f}(x,y)$

$\alpha = \dfrac{\sigma^2_{ruído}}{\sigma^2_{local}}$

se borda => σ^2_{local} elevado => $\alpha \approx 0$ => g(x, y) = f(x, y)

se homogêneo => $\alpha \approx 1$ => g(x, y) = $\bar{f}$(x, y)

Filtro adaptativo por difusão

- Difusão (ex. no ImageJ)
  - Isotrópica, linear (difusão térmica)
  - Anisotrópica não-linear (Perona & Malik)
  - Anisotrópica direcional

Filtro por difusão anisotrópica

$$\frac{\partial I(x,y,t)}{\partial t} = \mathrm{div}[c(|\nabla I|).\nabla I]$$

$$c(s) = \frac{1}{1+\left(\frac{s}{K}\right)^2}$$

Na qual K é uma referência para o valor do módulo de gradiente Vamos detalhar uma implementação numérica por diferenças finitas de cada parte da equação 17:

$$\frac{\partial I}{\partial t} = \frac{I(x,y,t+1) - I(x,y,t)}{\Delta t}$$

$$\nabla I(x,y,t) = \frac{\partial I}{\partial x}.\vec{u}_x + \frac{\partial I}{\partial y}.\vec{u}_y$$

$$\frac{\partial I(x,y,t)}{\partial x} = \frac{I(x+1,y,t) - I(x-1,y,t)}{2}$$

$$\frac{\partial I(x,y,t)}{\partial y} = \frac{I(x,y+1,t) - I(x,y-1,t)}{2}$$

na qual $\vec{u}_x$ e $\vec{u}_y$ são os vetores unitários dos eixos x e y respectivamente.

$|\nabla I(x,y,t)| =$

$$\sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} = \frac{1}{2}\sqrt{(I(x+1,y,t) - I(x-1,y,t))^2 + (I(x,y+1,t) - I(x,y-1,t))^2}$$

$$c_d(x,y,t) = c(|\nabla I|) = \frac{1}{1+\left(\frac{|\nabla I|}{K}\right)^2} = \frac{1}{1+\left[(I(x+1,y,t) - I(x-1,y,t))^2 + (I(x,y+1,t) - I(x,y-1,t))^2\right]/(2K^2)}$$

$$\frac{\partial I}{\partial t} = \mathrm{div}[c(|\nabla I|).\nabla I] =$$

$$= \mathrm{div}[a(x,y,t).\vec{u}_x + b(x,y,t).\vec{u}_y] = \frac{\partial a(x,y,t)}{\partial x} + \frac{\partial b(x,y,t)}{\partial y} =$$

$$= \frac{1}{2}[a(x+1,y,t) - a(x-1,y,t) + b(x,y+1,t) - b(x,y-1,t)]$$

Na qual,

$a(x,y,t) = c_d(x,y,t).\frac{\partial I(x,y,t)}{\partial x} = c_d(x,y,t).\frac{1}{2}[I(x+1,y,t) - I(x-1,y,t)]$, donde:

$$a(x+1,y,t) = \frac{1}{2}c_d(x+1,y,t).[I(x+2,y,t) - I(x+1,y,t)]$$

$$a(x-1,y,t) = \frac{1}{2}c_d(x-1,y,t).[I(x,y,t) - I(x-2,y,t)]$$

$$b(x,y,t) = c_d(x,y,t).\frac{\partial I(x,y,t)}{\partial y} = c_d(x,y,t).\frac{1}{2}[I(x,y+1,t) - I(x,y-1,t)]$$

$$b(x,y+1,t) = \frac{1}{2}c_d(x,y+1,t).[I(x,y+2,t) - I(x,y,t)]$$

$$b(x,y-1,t) = \frac{1}{2}c_d(x,y-1,t).[I(x,y,t) - I(x,y-2,t)]$$

Portanto,

$$4\frac{\partial I}{\partial t} = 4\frac{I(x,y,t+1) - I(x,y,t)}{\Delta t}$$
$$= c_d(x+1,y,t).[I(x+2,y,t) - I(x,y,t)]$$
$$- c_d(x-1,y,t).[I(x,y,t) - I(x-2,y,t)]$$
$$+ c_d(x,y+1,t).[I(x,y+2,t) - I(x,y,t)] - c_d(x,y-1,t).[I(x,y,t) - I(x,y-2,t)]$$

E o algoritmo com relaxação seria:

$$\begin{aligned} I(x,y,t+1) &\\ &= I(x,y,t) \\ &+ \frac{\lambda}{4}.\Delta t.\{c_d(x+1,y,t).[I(x+2,y,t) - I(x,y,t)] \\ &+ c_d(x-1,y,t).[I(x-2,y,t) - I(x,y,t)] \\ &+ c_d(x,y+1,t).[I(x,y+2,t) - I(x,y,t)] \\ &+ c_d(x,y-1,t).[I(x,y-2,t) - I(x,y,t)]\} \end{aligned}$$

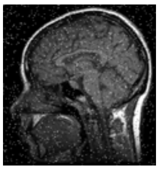
Na qual:

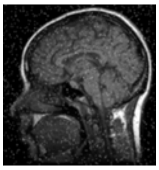
$$c_d(x,y,t) = \frac{1}{1+\left[(I(x+1,y,t) - I(x-1,y,t))^2 + (I(x,y+1,t) - I(x,y-1,t))^2\right]/(2K^2)}$$

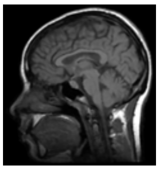
e $I(x,y,0) = I(x,y)$ (*imagem inicial*)

Conceito: filtro, convolução e Fourier

- Filtragem é uma operação feita sobre um sinal ou imagem para obter um outro sinal ou imagem
- Filtro linear e invariante pode ser implementado por convolução
- Convolução pode ser executado rapidamente no domínio da frequência por FFT


Ruído salt pepper


Filtro Média, r=1


Filtro Mediana, r=1

FILTROS LINEARES INVARIANTES NO DOMÍNIO DA FREQUÊNCIA=>