

Espaço de Escala

Resumo

O espaço de escala é uma das teorias utilizadas para a análise multi-escala de imagens e sinais. A técnica do espaço de escala linear gera as imagens em resoluções grossas fazendo convolução da imagem original com um núcleo gaussiano ou, equivalentemente, usando a imagem original como a condição inicial de um processo de difusão. Esta abordagem possui um defeito sério: é difícil obter a localização acurada das arestas importantes nas escalas grossas. A difusão anisotrópica foi proposta para superar esta dificuldade. Nela, os coeficientes da difusão são escolhidos de forma a encorajar a suavização intra-região e evitar a suavização inter-região. Com isso, os ruídos são eliminados e a imagem é simplificada ao mesmo tempo em que mantém as arestas nítidas.

Introdução

Percebemos os objetos no mundo como tendo estruturas em escalas grossas e finas. Uma floresta pode parecer simplesmente um amontoado verde quando vista de distância. À medida que nos aproximamos, começamos a distinguir as árvores individuais, os troncos, os galhos, as folhas, as nervuras das folhas, os orvalhos sobre as folhas, etc. Assim, a multi-escala constitui uma noção natural da percepção visual. A representação multi-escala de uma imagem em forma de pirâmide foi desenvolvida já na década de 70. Nesta estrutura, quanto mais grossa for a escala, menos pixels conterá a imagem.

Em 1983, Witkin [Witkin, 1983] propôs que a escala poderia ser considerada como um parâmetro contínuo, generalizando a noção de pirâmide. A idéia essencial desta abordagem é muito simples: dada uma imagem digital Q , essa imagem na escala σ é a convolução da Q com a máscara gaussiana de desvio-padrão σ . Esta teoria é denominada de espaço de escala gaussiano ou linear. A imagem Q na escala $\sigma=0$ é a própria imagem original. À medida que se vai da escala fina para a escala grossa, a imagem se torna cada vez mais “borrada”.

A convolução com a máscara gaussiana de desvio-padrão σ pode ser vista como a solução da equação de condução de calor, onde o valor da imagem original Q num ponto (x, y) é a temperatura inicial nesse ponto, o tempo decorrido é $t = \sigma^2 / 2$, e a imagem Q na escala σ representa as temperaturas no instante t . Assim, a convolução gaussiana é um processo de difusão isotrópica. Isotrópico significa “aquele que apresenta as mesmas propriedades físicas em todas as direções”, segundo [Aurélio, 1999].

As pesquisas subseqüentes levaram a diferentes formas de simplificar a imagem original, utilizando filtros diferentes da convolução gaussiana. Por exemplo, Jackway e Deriche [Jackway and Deriche, 1996] propuseram o uso de operadores morfológicos, resultando no espaço de escala morfológico.

Uma outra forma de simplificar imagens foi proposta por Perona e Malik [Perona and Malik, 1987; Perona and Malik, 1990], e teve um grande impacto científico. Eles propuseram o uso da difusão anisotrópica, substituindo a difusão isotrópica. No espaço de escala linear (que utiliza a difusão isotrópica para simplificar uma imagem), uma imagem em escala grossa torna-se borrada e as arestas deslocam-se espacialmente de uma escala para outra. Utilizando a difusão anisotrópica, as arestas continuam nítidas mesmo em escalas grossas e permanecem na mesma posição mesmo mudando de escala.

Na formulação da difusão anisotrópica de Perona-Malik, existe uma função chamada parada-na-aresta (edge stopping function) g , que controla a intensidade da difusão de acordo com o gradiente do ponto que deve sofrer difusão. A função parada-na-aresta possui um parâmetro de escala σ que, em conjunto com o gradiente, indica se a difusão deve ser forte ou fraca. A correta escolha da função parada-na-aresta e da escala afetam de forma decisiva o resultado da filtragem da imagem. Perona e Malik sugeriram duas funções parada-na-aresta, sem apresentar uma justificativa fundamentada para a escolha.

A difusão anisotrópica robusta (RAD) [Black et al., 1998] foi proposta como um melhoramento da difusão anisotrópica de Perona-Malik. Esta técnica assume que a entrada é uma imagem constante por regiões corrompida pelo ruído gaussiano aditivo com média zero e pequena variância. O objetivo é estimar a imagem original a partir dos dados ruidosos. Black et al. usaram a estatística robusta para resolver este problema, e propuseram o uso da função “Tukey’s biweight” como a função parada-na-aresta, de acordo com a teoria estatística adotada. Na prática, a RAD converge mais rapidamente e conserva ainda melhor as bordas do que a difusão de Perona-Malik.

A RAD mostra-se útil em diversas aplicações de Processamento e Análise de Imagens. Ela é um excelente detector de arestas. Também é um ótimo filtro de ruídos aditivos, que preserva as

bordas ao mesmo tempo em que elimina os ruídos. Quando utilizado como um filtro, a RAD procura estimar a imagem original constante por regiões a partir da sua versão corrompida pelo ruído gaussiano aditivo. Esta característica torna-a extremamente eficiente em diversas aplicações.

Espaço de Escala Linear

Programa Matlab usado para derivar a gaussiana 1-D:

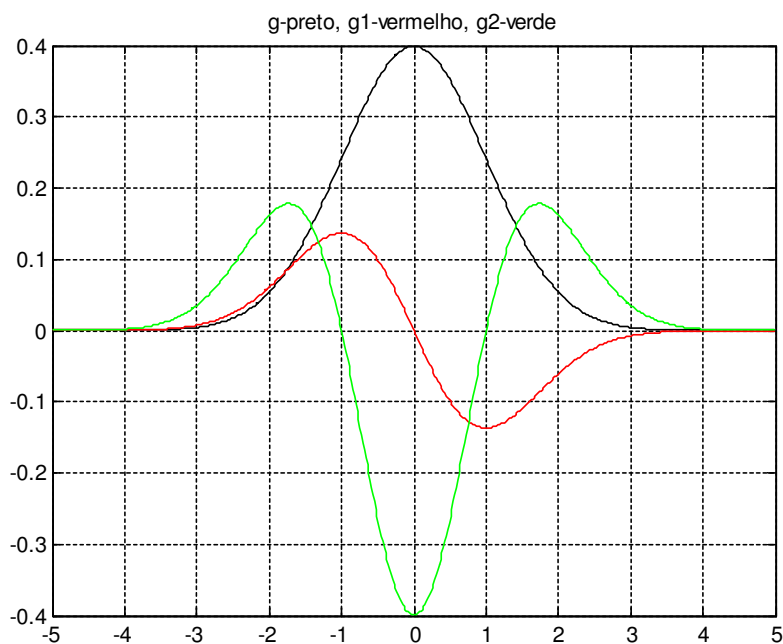
```
function dgauss1
g='1/(sigma*sqrt(2*pi))*exp((-x*x)/(2*sigma*sigma))'
g1=simplify(diff(sym(g)))
g2=simplify(diff(sym(g1)))

%g = 1/(sigma*sqrt(2*pi))*exp((-x*x)/(2*sigma*sigma))
%g1 = -1/2/sigma^3*2^(1/2)/pi^(1/2)*x*exp(-1/2*x^2/sigma^2)
%g2 = 1/2*2^(1/2)*exp(-1/2*x^2/sigma^2)*(-sigma^2+x^2)/sigma^5/pi^(1/2)
```

Programa Matlab usado para plotar a gaussiana e derivadas:

```
function grafico1

N=512;
s=1;
for i=0:N-1;
    T(ind(i))=10*i/N-5;
end;
for i=0:N-1;
    x=T(ind(i));
    v1(ind(i))=1/(s*(2*pi)^0.5)*exp((-x*x)/(2*s*s));
end;
for i=0:N-1;
    x=T(ind(i));
    v2(ind(i))=(-x)/(sqrt(2)*s^3*pi)*exp((-x*x)/(2*s*s));
end;
for i=0:N-1;
    x=T(ind(i));
    v3(ind(i)) = (x^2-s^2) / (s^5*sqrt(2*pi)) * exp(-x^2/(2*s^2));
end;
figure;
plot(T,v1,'k', T,v2,'r', T,v3,'g');
grid;
title('g-preto, g1-vermelho, g2-verde');
```

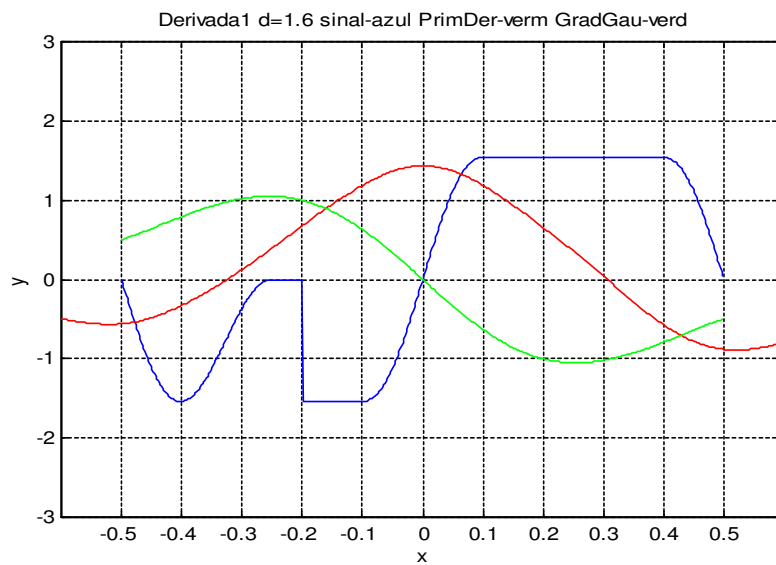
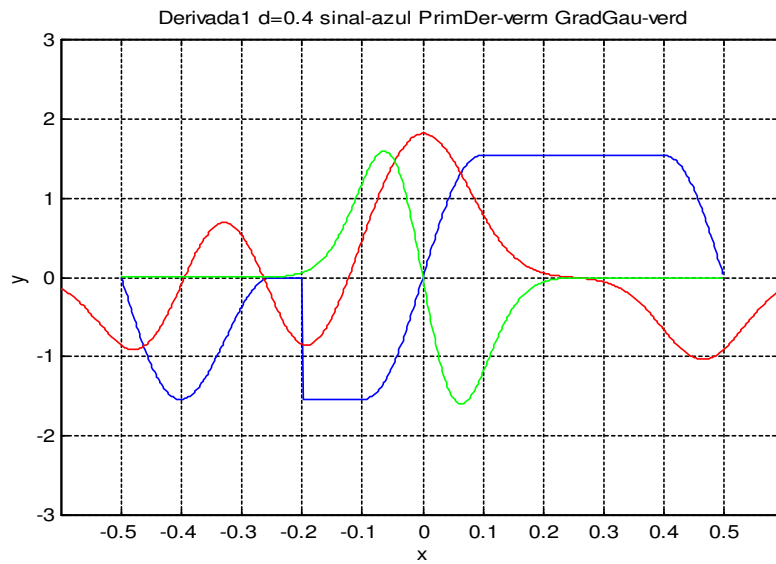
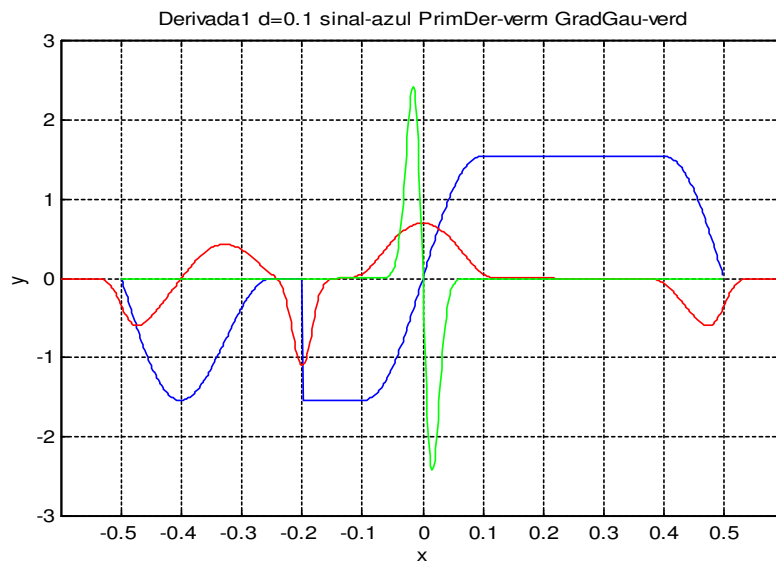


Derivadas da gaussiana 1D de média zero e desvio 1:

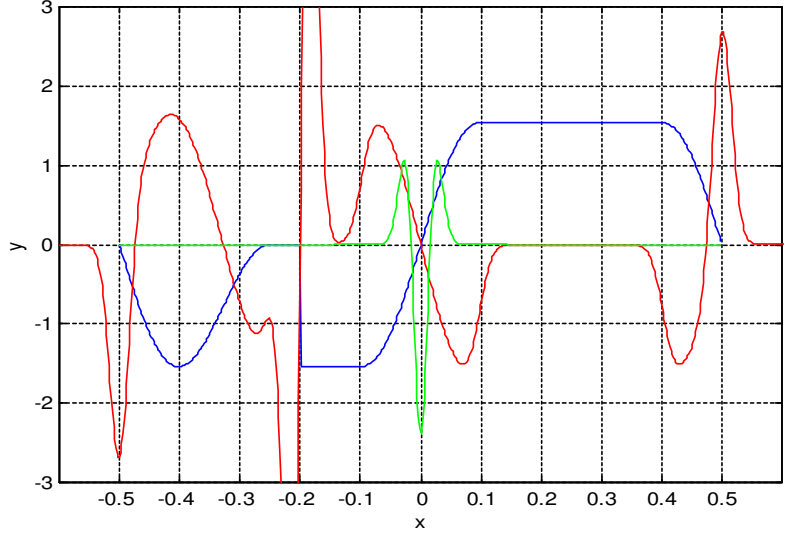
$$g(x, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left[\frac{-x^2}{2\sigma^2}\right]$$

$$g_x(x, \sigma) = \frac{-x}{\sigma^3 \sqrt{2\pi}} \exp\left[\frac{-x^2}{2\sigma^2}\right]$$

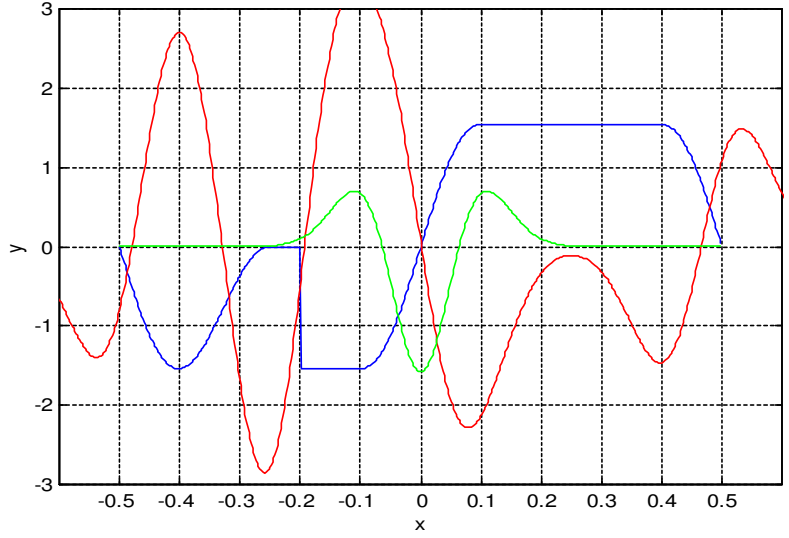
$$g_{xx}(x, \sigma) = \frac{x^2 - \sigma^2}{\sigma^5 \sqrt{2\pi}} \exp\left[\frac{-x^2}{2\sigma^2}\right]$$



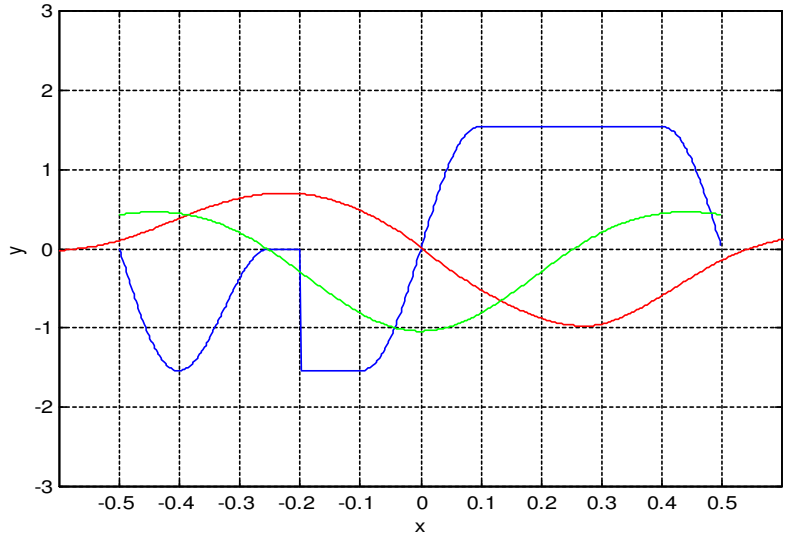
Derivada2 d=0.1 sinal-azul SegDef-verm LapGau-verd



Derivada2 d=0.4 sinal-azul SegDef-verm LapGau-verd



Derivada2 d=1.6 sinal-azul SegDef-verm LapGau-verd



Caso bidimensional

Definição (normal): A distribuição normal bidimensional $N(x_0, y_0, \sigma)$, onde (x_0, y_0) é a média e σ é o desvio-padrão, é definida através da função gaussiana:

$$g(x, y, x_0, y_0, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{(x-x_0)^2 - (y-y_0)^2}{2\sigma^2}\right]$$

A figura 3.1b mostra a função $g(x, y, 0, 0, 1)$, e as figuras 3.1c-3.1f mostram o seu módulo do gradiente, as suas derivadas parciais, e o seu laplaciano. Costuma-se adotar $\sigma^2 = 2t$ e $\mu = 0$ para obter a notação:

$$G_t(x, y) = \frac{1}{4\pi t} \exp\left[-\frac{x^2 + y^2}{4t}\right].$$

Definição (espaço de escala): Seja $f: \mathbb{R}^2 \rightarrow \mathbb{R}$ uma imagem 2-D. O espaço de escala desta imagem é a função $F: \mathbb{R}^2 \times \mathbb{R}^+ \rightarrow \mathbb{R}$ (denotada $F_t(x, y)$) que satisfaz a seguinte equação diferencial parcial ou equação de calor bidimensional:

$$\begin{aligned} \frac{\partial F_t(x, y)}{\partial t} &= \nabla^2 F_t(x, y) = \frac{\partial^2 F_t(x, y)}{\partial x^2} + \frac{\partial^2 F_t(x, y)}{\partial y^2}, \\ F_0(x, y) &= f(x, y) \end{aligned} \quad (3.1)$$

Afirmção: A solução da equação diferencial parcial acima pode ser expressa como uma convolução com gaussianas bidimensionais:

$$F_t(x, y) = G_t(x, y) * f(x, y),$$

Proposição (separabilidade): A convolução acima pode ser calculada através de duas convoluções com gaussianas unidimensionais:

$$F_t(x, y) = G_t(y) *_{(y)} G_t(x) *_{(x)} f(x, y).$$

Esta propriedade permite acelerar a computação do espaço de escala gaussiano para as imagens.

Além da linearidade e da invariância por translações, o espaço de escala gaussiano bidimensional possui a invariância por rotações.

Proposição: Seja f uma imagem qualquer e $g = R_\theta f$ a rotação de f por ângulo θ . Então o espaço de escala G_t de g é a rotação por ângulo θ de F_t , isto é:

$$g = R_\theta f \Rightarrow G_t = R_\theta F_t.$$

Infelizmente, o princípio de causalidade não vale para as imagens 2-D. Velho et al. [Velho et al., 2000] afirmam: “Tentemos agora entender o que será o princípio da causalidade em 2-D. Note que não faz sentido falar em número de cruzamentos de zero de uma imagem, já que em geral os cruzamentos de zero de uma imagem formam um conjunto de curvas, não um conjunto discreto de pontos. Por outro lado, pode-se falar do número de máximos e mínimos locais de uma imagem genérica (ou de um sinal n -dimensional). No entanto, *não é verdade* que o número de pontos críticos diminua com a escala no espaço de escala de uma imagem qualquer.”

Na prática, qualquer imagem digital está definida em um subconjunto finito de Z^2 , em vez de R^2 . Assim, é necessário discretizar de alguma forma o espaço de escala espacialmente. Muitas técnicas de discretização têm sido utilizadas para esta tarefa, por exemplo, a gaussiana amostrada, a gaussiana integrada e a gaussiana verdadeiramente discreta (obtida utilizando a função modificada de Bessel). Por outro lado, não é estritamente necessário discretizar o espaço de escala no tempo, pois é possível calcular “sob encomenda” qualquer pixel em qualquer escala real no espaço de escala discretizado espacialmente $Z^2 \times R_+$. Porém, é computacionalmente vantajoso pré-calcular o espaço de escala para algumas escalas fixas, obtendo o espaço de escala discretizada espacial e temporalmente $Z^2 \times Z_+$. Veja [Velho et al., 2000; Lindeberg, 1994] para maiores detalhes.

A figura 3.2 mostra a detecção de arestas de uma imagem no espaço de escala linear. A imagem original sofre convoluções com as gaussianas de diferentes desvios-padrões, gerando as imagens em diferentes escalas (primeira coluna). Calculando a convolução da imagem original com o laplaciano da gaussiana com diferentes desvios-padrões, obtém-se a segunda coluna (onde está ilustrado somente o sinal das imagens resultantes: preto indica positivo e branco indica negativo). Aplicamos os operadores morfológicos (dilatação seguida pela operação ou-exclusivo) sobre as imagens da segunda coluna para obter as arestas em diferentes escalas (terceira coluna).

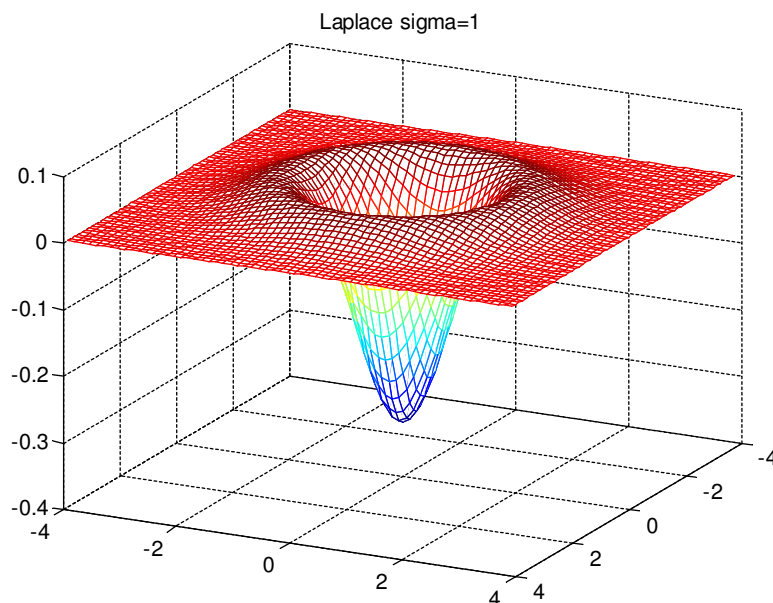
Programa Matlab usado para derivar a gaussiana 2-D:

```
function dgauss2
g='1/(2*pi*s^2) * exp((-x^2-y^2)/(2*s^2))'
gx=simplify(diff(sym(g), 'x'))
gy=simplify(diff(sym(g), 'y'))
gxx=simplify(diff(sym(gx), 'x'))
gyy=simplify(diff(sym(gy), 'y'))
laplace=simplify(plus(gxx, gyy))

%g = 1/(2*pi*s^2) * exp((-x^2-y^2)/(2*s^2))
%gx = -1/2/pi/s^4*x*exp(-1/2*(x^2+y^2)/s^2)
%gy = -1/2/pi/s^4*y*exp(-1/2*(x^2+y^2)/s^2)
%gxx = -1/2*exp(-1/2*(x^2+y^2)/s^2)*(s^2-x^2)/pi/s^6
%gyy = -1/2*exp(-1/2*(x^2+y^2)/s^2)*(s^2-y^2)/pi/s^6
%laplace = -1/2*exp(-1/2*(x^2+y^2)/s^2)*(2*s^2-x^2-y^2)/pi/s^6
```

Programa Matlab usado para plotar a gaussiana e derivadas:

```
function grafico2
N=64;
s=1;
for i=0:N-1;
    X(ind(i))=8*i/N-4;
    Y(ind(i))=8*i/N-4;
end;
for i=0:N-1;
    x=X(ind(i));
    for j=0:N-1;
        y=Y(ind(j));
        Z(ind(i),ind(j))=-1/2*exp(-1/2*(x^2+y^2)/s^2)*(2*s^2-x^2-y^2)/pi/s^6;
    end;
end;
mesh(X,Y,Z);
view([1 0.5 0.5]);
title('Laplace sigma=1');
```



Derivadas da gaussiana 2D de média zero e desvio 1.

$$g(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right]$$

Nota: O volume embaixo da curva dá um.

$$g_x(x, y, \sigma) = \frac{-x}{2\pi\sigma^4} \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right]$$

$$g_y(x, y, \sigma) = \frac{-y}{2\pi\sigma^4} \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right]$$

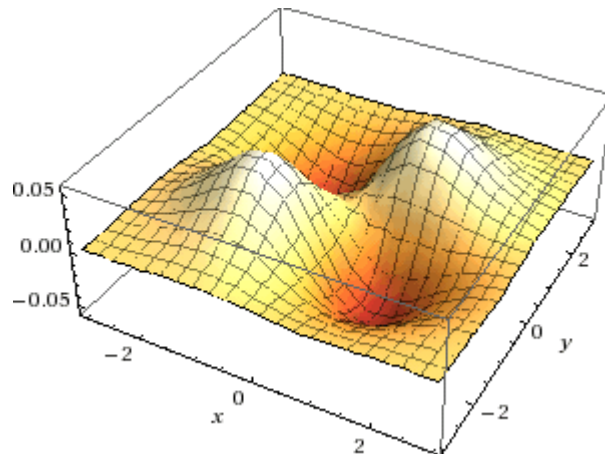
$$\text{gradiente}(g) = \nabla g(x, y, \sigma) = [g_x, g_y] = \left[\frac{-x}{2\pi\sigma^4} \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right], \frac{-y}{2\pi\sigma^4} \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right] \right]$$

Nota: A somatória absoluta não dá um e depende do desvio.

$$g_{xx}(x, y, \sigma) = \frac{x^2 - \sigma^2}{2\pi\sigma^6} \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right]$$

$$g_{yy}(x, y, \sigma) = \frac{y^2 - \sigma^2}{2\pi\sigma^6} \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right]$$

$$g_{xy}(x, y, \sigma) = \frac{xy}{2\pi\sigma^6} \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right]$$



$$\text{laplace}(g) = \nabla^2 g(x, y, \sigma) = g_{xx} + g_{yy} = \frac{x^2 + y^2 - 2\sigma^2}{2\pi\sigma^6} \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right]$$

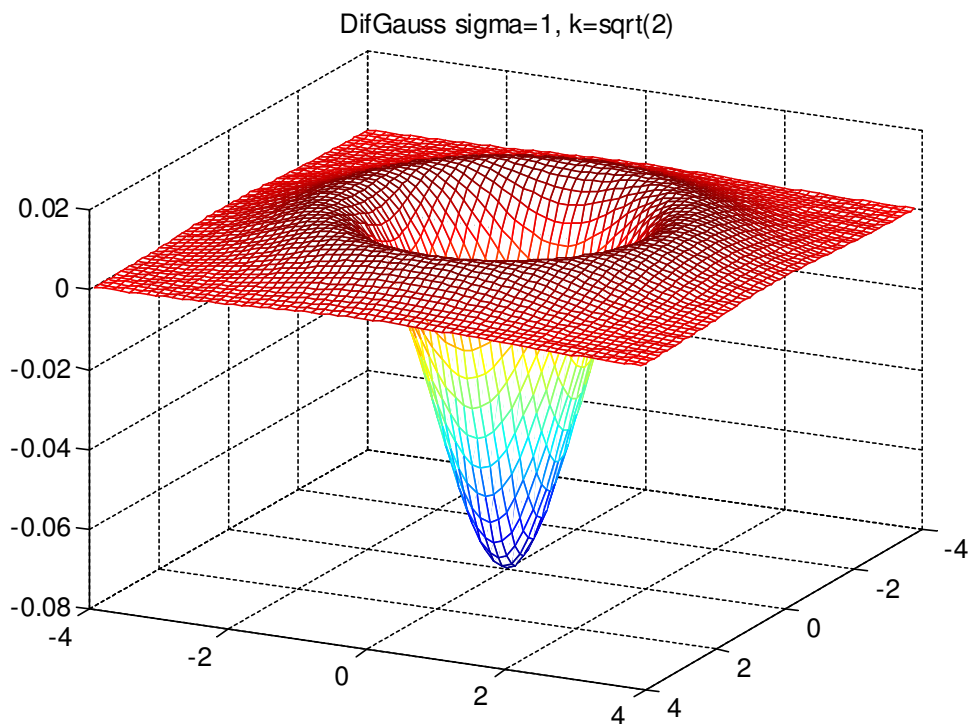
Nota: A somatória absoluta (ou integral absoluta) não dá um e depende do desvio.

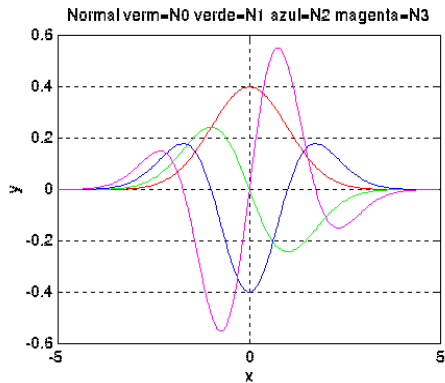
Aproximar laplace pela diferença de duas Gaussianas [Lowe, 2004]:

$$g(x, y, k\sigma) - g(x, y, \sigma) \approx (k - 1)\sigma^2 \nabla^2 g(x, y, \sigma)$$

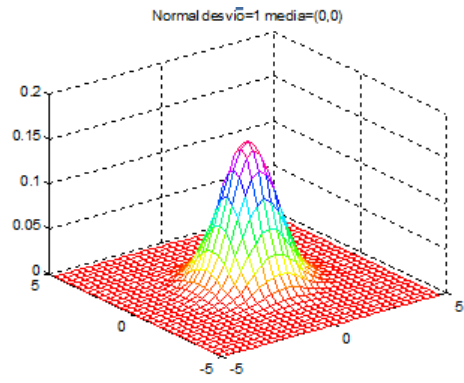
onde $k > 1$ é a relação entre duas gaussianas. Por exemplo, $k = \sqrt{2}$. Se tiver n escalas por oitava, $k = \sqrt[n]{2}$. O erro de aproximação tende a zero quando k tende a 1, mas a aproximação funciona mesmo para k razoavelmente grande como $k = \sqrt{2}$.

A somatória absoluta da diferença de Gaussianas dá aparentemente $1/n$, independente do desvio. A média dá zero.

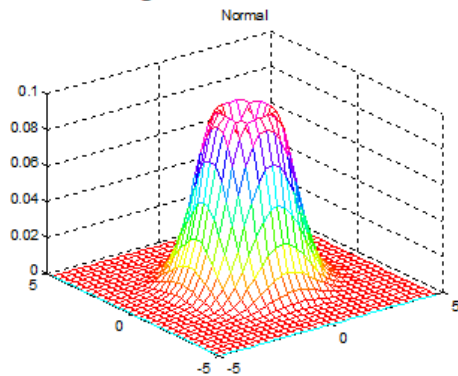




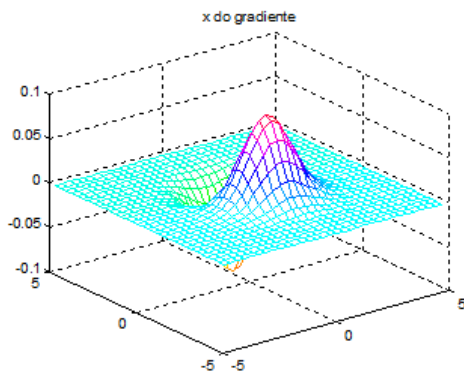
(3.1a) Função gaussiana com $\sigma=1$ (vermelho) e suas 1ª, 2ª e 3ª derivadas (respectivamente em verde, azul e magenta).



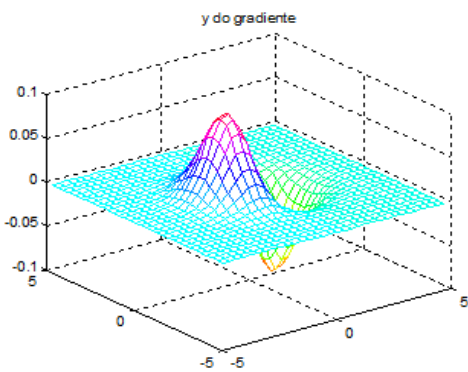
(3.1b) Função gaussiana G bidimensional com $\sigma=1$.



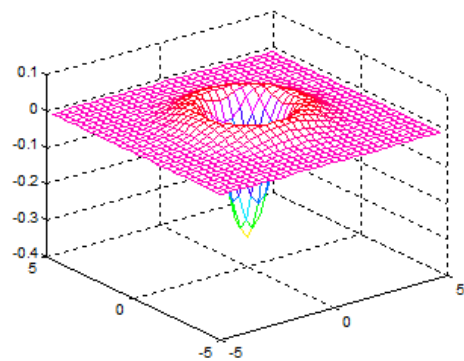
(3.1c) Módulo do gradiente da gaussiana $\|\nabla G(x, y)\|$.



(3.1d) Derivada parcial x da gaussiana $\partial G(x, y) / \partial x$.



(3.1e) Derivada parcial y da gaussiana $\partial G(x, y) / \partial y$.



(3.1f) Laplaciano da gaussiana $\nabla^2 G(x, y)$.

Fig. 3.1: Funções gaussianas unidimensional, bidimensional e suas derivadas.




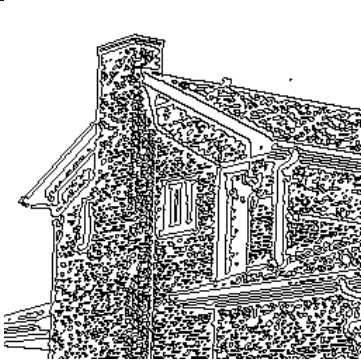


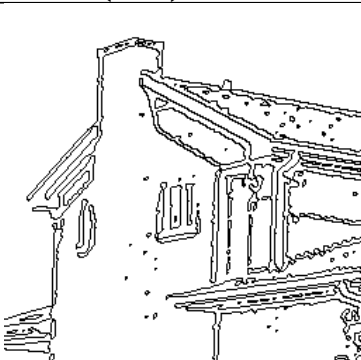
Imagem “casa.tga” no espaço de escala linear.	Sinal do laplaciano da imagem. Preto=negativo.	Cruzamentos de zero do laplaciano (ou arestas).
 <p>(3.2a) Imagem original ($\sigma=0$)</p>		
 <p>(3.2b) $\sigma=1,0$</p>	 <p>(3.2c) $\sigma=1,0$</p>	 <p>(3.2d) $\sigma=1,0$</p>
 <p>(3.2e) $\sigma=1,5$</p>	 <p>(3.2f) $\sigma=1,5$</p>	 <p>(3.2g) $\sigma=1,5$</p>

Fig. 3.2: Continua na próxima página.

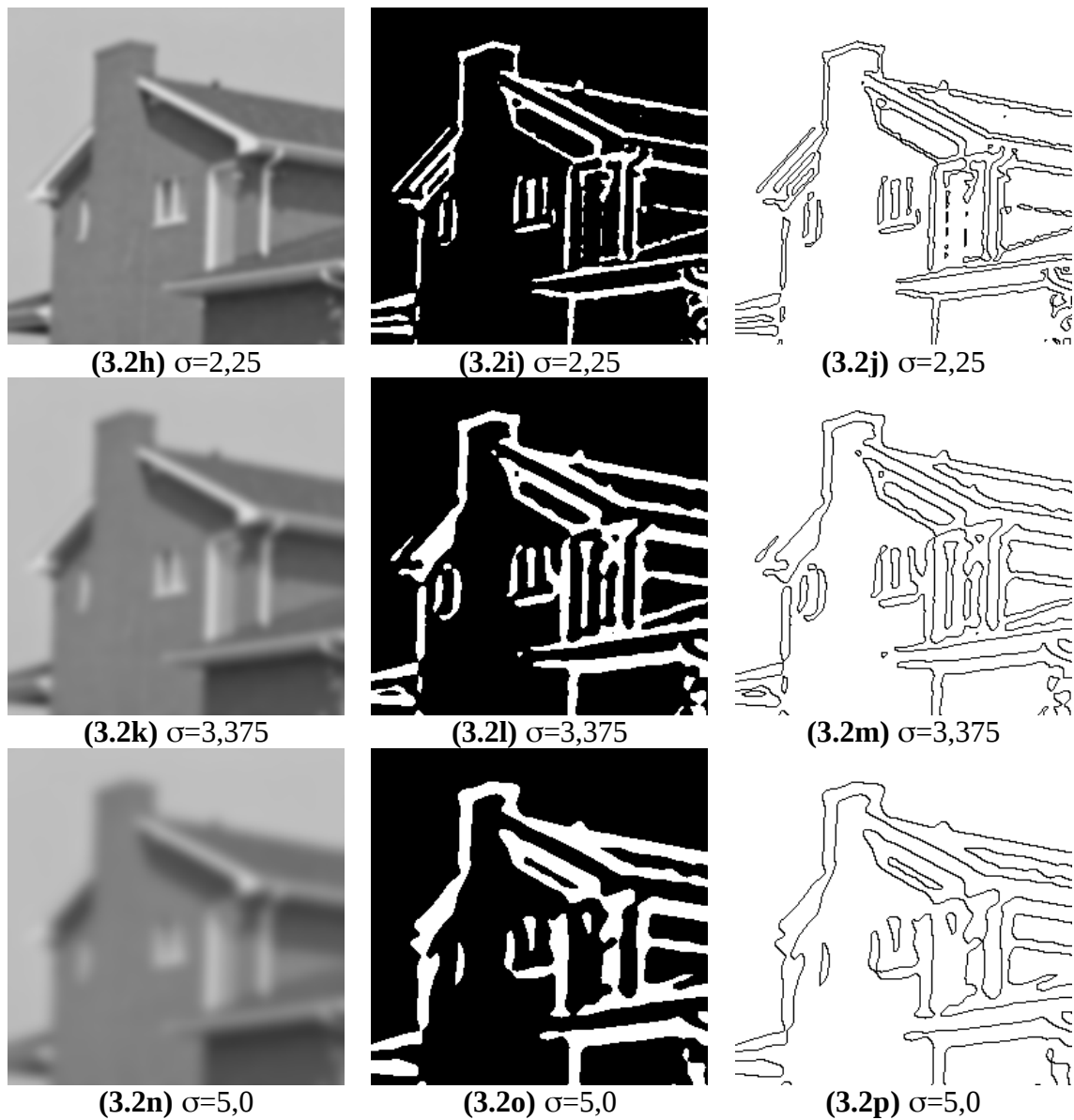


Fig. 3.2: Detecção de arestas no espaço de escala linear. A imagem “casa.tga” no espaço de escala (primeira coluna), o sinal do laplaciano da imagem (segunda coluna) e as arestas ou os cruzamentos de zero do laplaciano da imagem (terceira coluna). Quanto σ cresce, as arestas menos importantes deixam de ser detectadas. Note que as arestas deslocam-se espacialmente à medida que σ cresce.

```
img gaussg casag.tga g1000.tga 1
img laplaceg casag.tga l1000.tga 1 1
img threshg l1000.tga l1000.bmp 128
img findedb l1000.bmp e1000.bmp
```

Convolução com Gaussiana:

Convolução com gaussiana em Proeikon:

```
IMGFLT ConvGaussHV(IMGFLT b, double desvio, char borda='x');  
img gaussg
```

Convolução com gaussiana usando Proeikon+OpenCV:

```
IMGFLT pvConvGauss(IMGFLT b, double desvio);  
pv gaussg
```

Ambas as funções dão o mesmo resultado. Ambas assumem a borda estendida ('x').

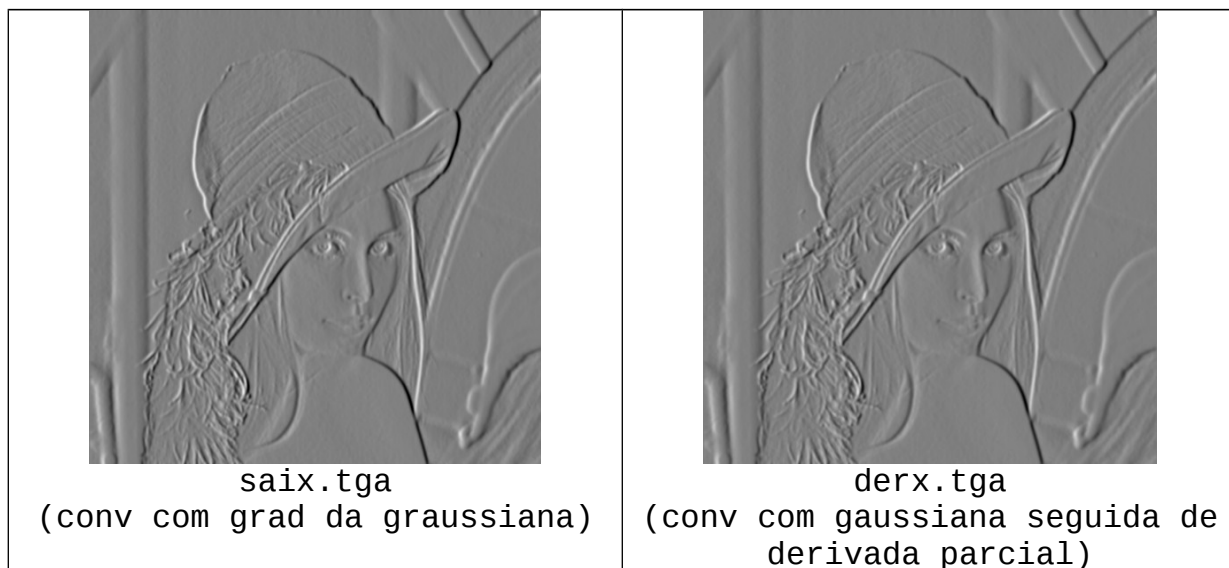
Convolução com gradiente da Gaussiana:

Fazer convolução com gradientes da Gaussiana é igual a fazer convolução com Gaussiana seguida de cálculo de derivadas parciais?

```
img gradieng lennag.tga saix.tga saiy.tga 1.3 1  
:: Convolucao com gradientes da gaussiana desvio=1.3,  
:: amplitude=1
```

```
img gaussg lennag.tga gauss.tga 1.3  
:: Convolucao com gaussiana desvio=1.3  
img masgradg gauss.tga derx.tga dery.tga d 1.3  
:: Derivadas parciais [convolucao com (1 0 -1) e (-1 0 1)t]  
:: amplitude=1.3
```

Os resultados são quase idênticos. MAE=0.4%. Note que na segunda opção, a saída tem que ser multiplicada pelo desvio-padrão para que as saídas fiquem semelhantes.



Para calcular convolução com gradientes das gaussianas:

```
IMGFLT gradientex(IMGFLT ent, double desvio, double rotacao=0.0);  
IMGFLT gradientey(IMGFLT ent, double desvio, double rotacao=0.0);
```

Para calcular derivadas parciais:

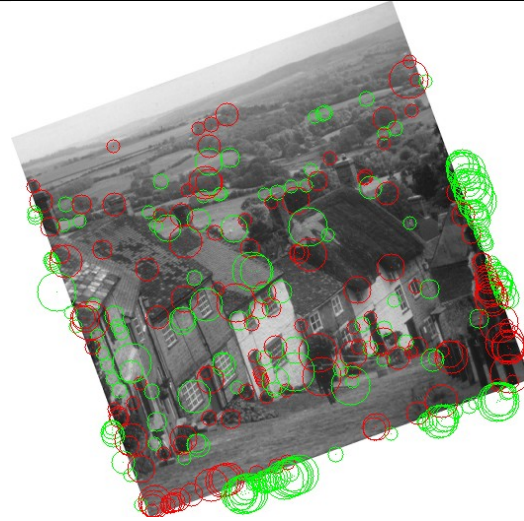
```
IMGFLT ent; le(ent, "????????");  
IMGFLT mx=derivparc('x'); IMGFLT my=derivparc('y');  
// ou IMGFLT mx(1, 3, 1.0, 0.0, -1.0);  
// ou IMGFLT my(3, 1, -1.0, 0.0, 1.0);  
SomatoriaUm(mx); SomatoriaUm(my);  
saix=convolucao(ent,mx); saiy=convolucao(ent,my);
```


Detecção de blob com extremos de DoG (diferença de gaussianas).

Os extremos de DoG fornecem key-points invariantes à rotação e à escala.
kcek dogextr goldhill.tga goldhill.ppm 10 3 4 0.05
kcek dogextr z.pgm z.ppm 10 3 4 0.05



goldhill.ppm



z.ppm

Note que o programa achou os mesmos keypoints (que não encostam na borda), mesmo após rotação e mudança de escala. Os círculos encostados nas bordas das imagens devem ser desprezados.

```

#include <cekeikon.h>

void dogextr(int argc, char** argv)
{ if (argc!=6 && argc!=7) {
    printf("DoGExtr: Difference of Gaussians extrema\n");
    printf("DoGExtr ent.pgm sai.pgm NEsc EscInic escOitava [limiar=0]\n");
    printf(" NEsc=#escalas(>=4) EscInic=escala inicial escOitava=#escalas por oitava\n");
    printf(" So picos acima de limiar e vales abaixo de -limiar sao mostrados\n");
    printf(" Ex: DoGExtr ent.pgm sai.ppm 9 1.0 3 0.1\n");
    erro("Erro: Numero de argumentos invalido");
}

int ns;
if (sscanf(argv[3], "%d", &ns)!=1) erro("Erro: Leitura NEsc");
if (ns<4) erro("Erro: NEsc<4");

double scale;
if (sscanf(argv[4], "%lf", &scale)!=1) erro("Erro: Leitura EscInic");
if (scale<0.3) erro("Erro: EscInic<0.3");

double escOitava;
if (sscanf(argv[5], "%lf", &escOitava)!=1) erro("Erro: Leitura escOitava");
if (escOitava<=0.0) erro("Erro: escOitava<=0.0");

double limiar=0;
if (argc==7) convArg(limiar, argv[6]);

Mat_<GRY> g; le(g, argv[1]);
Mat_<FLT> a; converte(g, a);
int tam1[]={ns, a.rows, a.cols}; M3d_<FLT> f(3, tam1);
VETOR<double> sca(ns);

double passo=pow(2.0, 1.0/escOitava);
Mat_<FLT> b;
for (int s=0; s<ns; s++) {
    sca(s)=scale;
    GaussianBlur(a, b, Size(0,0), scale);
    b.copyTo(fatia(f, s));
    scale=scale*passo;
}
// impAvi(f, "espaco.avi");

int tam2[]={ns-1, a.rows, a.cols}; M3d_<FLT> dog(3, tam2);
for (int s=0; s<dog.size[0]; s++) {
    b = FLT(escOitava)*(fatia(f, s+1)-fatia(f, s));
    b.copyTo(fatia(dog, s));
}
// impAvi(dog, "dog.avi");

Mat_<COR> sai; converte(g, sai);
for (int s=1; s<dog.size[0]-1; s++)
    for (int l=1; l<dog.size[1]-1; l++)
        for (int c=1; c<dog.size[2]-1; c++) {
            double scale=sca(s);
            FLT valor=dog(s, l, c);

            bool omaior=true;
            for (int s2=-1; s2<=1; s2++)
                for (int l2=-1; l2<=1; l2++)
                    for (int c2=-1; c2<=1; c2++)
                        if (valor<dog(s+s2, l+l2, c+c2)) { omaior=false; goto sai1; }
            sai1:

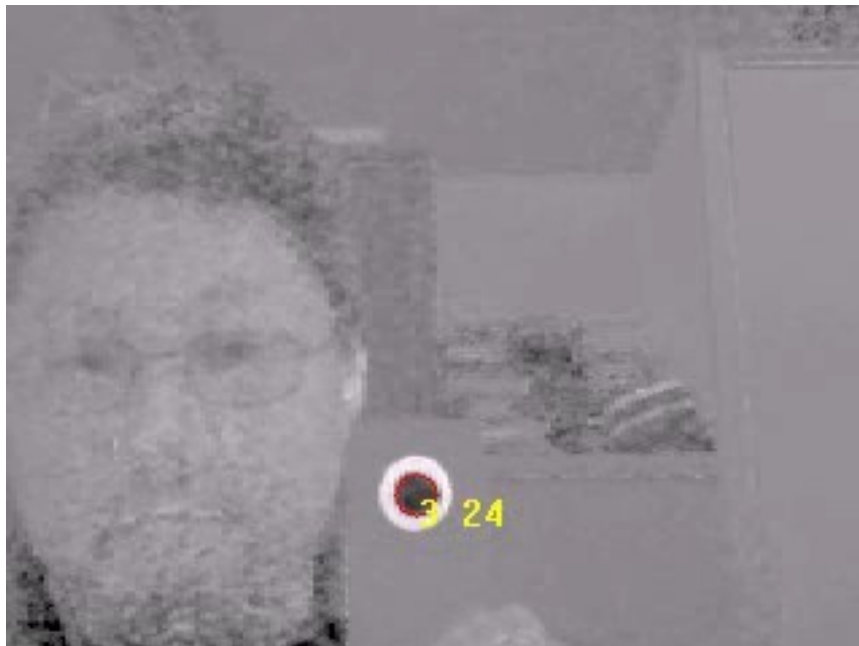
            if (omaior==true && valor>limiar) {
                sai(l, c)=COR(0, 0, 255);
                circulo(sai, l, c, arredonda(scale*2), COR(0, 0, 255));
            } else {
                bool omenor=true;
                for (int s2=-1; s2<=1; s2++)
                    for (int l2=-1; l2<=1; l2++)
                        for (int c2=-1; c2<=1; c2++)
                            if (valor>dog(s+s2, l+l2, c+c2)) { omenor=false; goto sai2; }
                sai2:
                if (omenor==true && valor<-limiar) {
                    sai(l, c)=COR(0, 255, 0);
                    circulo(sai, l, c, arredonda(scale*2), COR(0, 255, 0));
                }
            }
        }
}
imp(sai, argv[2]);
}

```

Detcirc5.cpp

Detecta círculos escuros usando diferença de gaussianas.

Veja a saída em www.lps.usp.br/~hae/apostila/detcirc5.avi



Convolução com gaussiana pode ser simulada pela “troca de calor” com os pixels vizinhos.

Difusão isotrópica linear: Dada uma imagem f , o seu espaço de escala F_t é:

$$\begin{cases} \frac{\partial F_t(x, y)}{\partial t} = \nabla \cdot [\nabla F_t(x, y)] \\ F_0(x, y) = f(x, y) \end{cases},$$

onde $\|\nabla F_t(x, y)\|$ é o magnitude do gradiente da F_t . Esta equação pode ser discretizada como:

$$I(s, t+1) = I(s, t) + \frac{\lambda}{|\eta_s|} \sum_{p \in \eta_s} \nabla I_{s,p}(t)$$

- $I(s, t)$ é a imagem discretizado espacial e temporalmente;
- s denota a posição de pixel numa grade discreta 2-D;
- t agora denota o passo de tempo discreto (número de iterações, $t \geq 0$);
- a constante $\lambda \in \mathbb{R}^+$ determina a velocidade de difusão (normalmente $\lambda = 1$);
- η_s representa o conjunto de vizinhos espaciais do voxel s . Para imagens 2-D, normalmente quatro pixels vizinhos são considerados: norte, sul, leste e oeste. Para imagens 3-D, seis voxels são normalmente considerados (os quatro voxels já mencionados mais os voxels “em cima” e “embaixo”);
- $\nabla I_{s,p}(t)$ é a magnitude do gradiente da imagem I no ponto s na direção (s, p) na iteração t : $\nabla I_{s,p}(t) = I(p, t) - I(s, t)$, $p \in \eta_s$.

Esta equação equivale a convolução com gaussiana. Simula a transferência de calor.

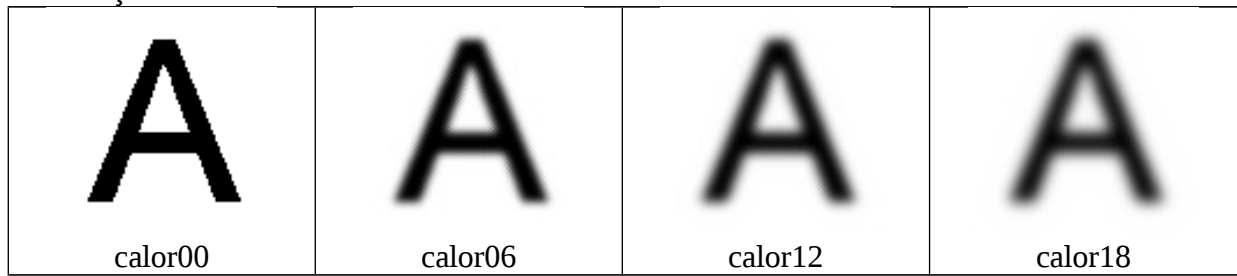
```
#include <proeikon>
//#include <imgpv>

int main()
{ IMGFLT a; le(a, "letraa.tga");
  I3DFLT I(20, a.nl(), a.nc());
  double lambda=1.0;
  I(0)=a;
  for (int i=0; i<I.ns()-1; i++) {
    for (int l=0; l<I.nl(); l++)
      for (int c=0; c<I.nc(); c++) {
        double soma = (I(i)(l-1, c, 'x')-I(i)(l, c, 'x'))+
          (I(i)(l+1, c, 'x')-I(i)(l, c, 'x'))+
          (I(i)(l, c-1, 'x')-I(i)(l, c, 'x'))+
          (I(i)(l, c+1, 'x')-I(i)(l, c, 'x'));
        I(i+1)(l, c, 'x')=I(i)(l, c, 'x')+(lambda/4.0)*soma;
      }
  }

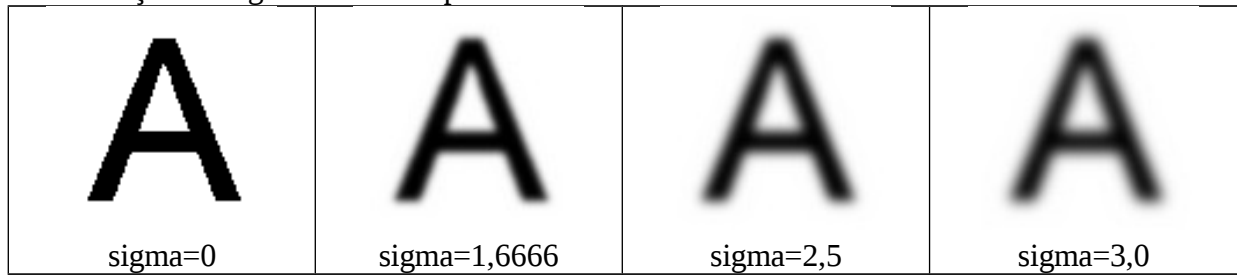
  for (int i=0; i<I.ns(); i++) {
    //pvMostra(IMGGRY(I(i)));
    char st[256];
    sprintf(st, "calor%02d.tga", i);
    imp(IMGGRY(I(i)), st);
  }

  I3DGRY g(I.ns(), I.nl(), I.nc());
  for (int i=0; i<I.ns(); i++) g(i)=I(i);
  imp(g, "calor.avi");
}
```

Simulação de transferência de calor:



Convolução com gaussiana mais parecida:



Difusão Anisotrópica

O espaço de escala linear possui muitas propriedades matemáticas atraentes. Porém, nas escalas grossas, a imagem torna-se borrada e as arestas deslocam-se espacialmente. Para manter as arestas nítidas, ao mesmo tempo em que se filtram os ruídos e os detalhes pouco importantes, Perona e Malik definiram o espaço de escala não-linear anisotrópica [Perona and Malik, 1990] modificando a equação diferencial parcial (3.1):

$$\begin{cases} \frac{\partial F_t(x, y)}{\partial t} = \nabla \cdot \left[g(\|\nabla F_t(x, y)\|) \nabla F_t(x, y) \right], \\ F_0(x, y) = f(x, y) \end{cases}$$

onde $\|\nabla F_t(x, y)\|$ é o magnitude do gradiente da F_t , e g é uma função “parada-na-aresta” (edge stopping function).

Perona e Malik discretizaram (espaço-temporalmente) a sua equação de difusão anisotrópica acima como:

$$I(s, t+1) = I(s, t) + \frac{\lambda}{|\eta_s|} \sum_{p \in \eta_s} g(|\nabla I_{s,p}(t)|) \nabla I_{s,p}(t), \quad (3.2)$$

onde:

- $I(s, t)$ é a imagem discretizado espacial e temporalmente;
- s denota a posição de pixel numa grade discreta 2-D;
- t agora denota o passo de tempo discreto (número de iterações, $t \geq 0$);
- a constante $\lambda \in \mathbb{R}^+$ determina a velocidade de difusão (normalmente $\lambda = 1$);
- η_s representa o conjunto de vizinhos espaciais do voxel s . Para imagens 2-D, normalmente quatro pixels vizinhos são considerados: norte, sul, leste e oeste. Para imagens 3-D, seis voxels são normalmente considerados (os quatro voxels já mencionados mais os voxels “em cima” e “embaixo”);
- $\nabla I_{s,p}(t)$ é a magnitude do gradiente da imagem I no ponto s na direção (s, p) na iteração t : $\nabla I_{s,p}(t) = I(p, t) - I(s, t)$, $p \in \eta_s$.

Perona e Malik sugeriram usar uma das duas funções parada-na-aresta abaixo (que vamos denotar por g_1 e g_2):

$$g_1(x) = \frac{1}{1 + \frac{x^2}{2\sigma^2}} \quad (\text{não normalizada})$$

$$g_2(x) = \exp\left[\frac{-x^2}{2\sigma^2}\right] \text{ (não normalizada)}$$

A correta escolha da função g e da escala σ afeta substancialmente o quanto as descontinuidades serão preservadas.

Black et al. [Black et al., 1998] propuseram recentemente a difusão anisotrópica robusta (RAD). Esta técnica assume que a entrada é uma imagem constante por regiões corrompida pelo ruído gaussiano aditivo com média zero e pequeno desvio-padrão. O objetivo é estimar a imagem original a partir do dado ruidoso. Black et al. usaram a estatística robusta para resolver este problema. Eles calcularam uma imagem I que satisfaz o seguinte critério de otimização:

$$\min_I \sum_{s \in I} \sum_{p \in \eta_s} \rho_\sigma(I(p) - I(s))$$

onde $I(s)$ é o valor da imagem I no pixel s , η_s é a vizinhança espacial do pixel s , ρ é uma norma de erro robusta e σ é um parâmetro de escala. A equação acima pode ser resolvida pelo sistema (3.2), fazendo $g(x) = \rho'(x)/x$. Black et al. escolheram a função “Tukey’s biweight” como a norma de erro ρ , de acordo com a teoria da estatística robusta. A correspondente função parada-na-aresta, que denotaremos como g_3 , é:

$$\begin{cases} \left[1 - \frac{x^2}{\sigma^2}\right]^2, & |x| \leq \sigma \\ 0, & \text{caso contrário (normalizada = não-normalizada)} \end{cases}$$

$$g_3(x) = \begin{matrix} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{matrix}$$

Para ter uma noção intuitiva da RAD, considere uma imagem constante por regiões, corrompida pelo ruído. A RAD executa a média da vizinhança intra-região, e evita calcular a média inter-região. Assim, este processo atenua os ruídos ao mesmo tempo em que preserva as arestas entre as diferentes regiões nítidas.

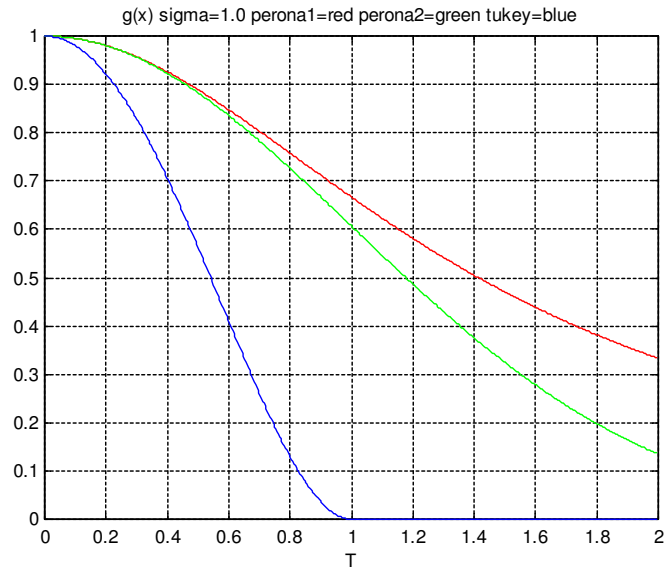
A figura 3.3a mostra as três funções parada-na-aresta. Repare que as três estão em escalas diferentes, de forma que é necessário normalizá-las para poder compará-las. Para isso, considere a função $\psi(x) = xg(x) = \rho'(x)$. Esta função é denominada função de influência na estatística robusta e indica o quanto o erro cometido por uma medida particular (e quantificado pela norma de erro ρ) influencia na solução. A figura 3.3b mostra as 3 funções de influência correspondentes às 3 funções parada-na-aresta. Para normalizar as 3 funções parada-na-aresta, os pontos de máximo das 3 funções de influência foram calculados, e as funções ψ_1 e ψ_2 foram ajustadas de forma que os seus pontos de máximo coincidam com o ponto de máximo da ψ_3

$(x = \sqrt{0,2})$. Fazendo isso, obtivemos as funções g_1 e g_2 normalizadas abaixo. A função g_3 não foi alterada.

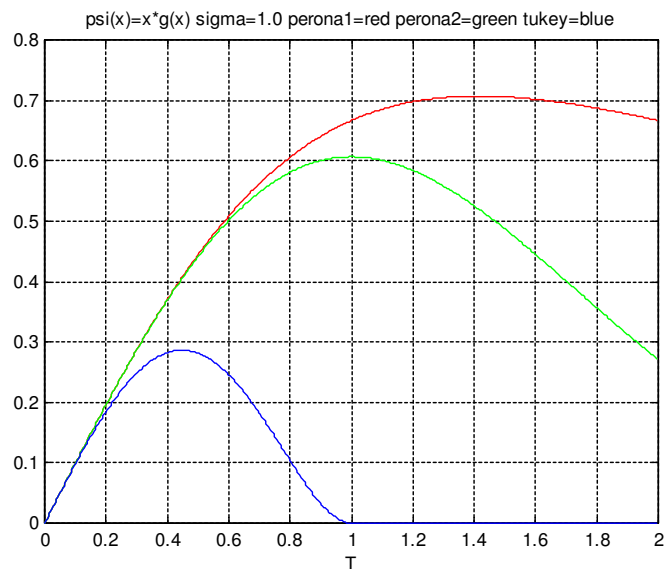
$$g_1(x) = \frac{1}{1 + \frac{5x^2}{\sigma^2}} \quad (\text{normalizada})$$

$$g_2(x) = \exp\left[\frac{-5x^2}{2\sigma^2}\right] \quad (\text{normalizada})$$

As figuras 3.3c e 3.3d mostram as funções parada-na-aresta e de influência normalizadas.

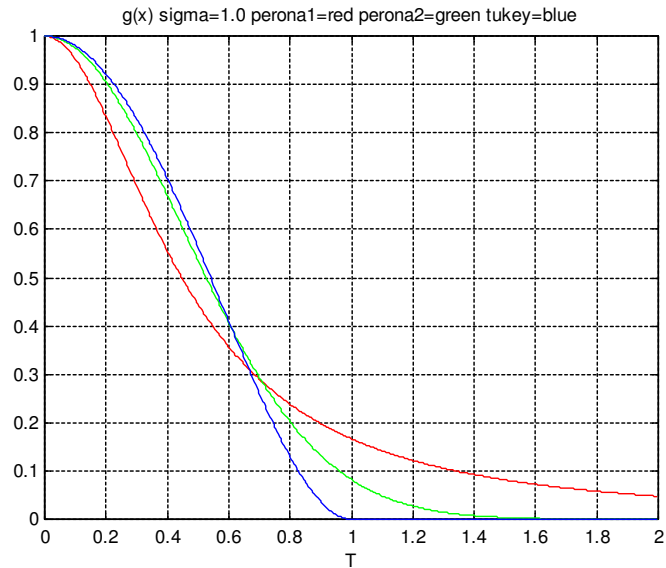


(3.3a) Funções parada-na-aresta não-normalizadas com $\sigma=1$: g_1 (Perona-Malik 1, em vermelho), g_2 (Perona-Malik 2, em verde) e g_3 (Tukey's biweight, em azul).

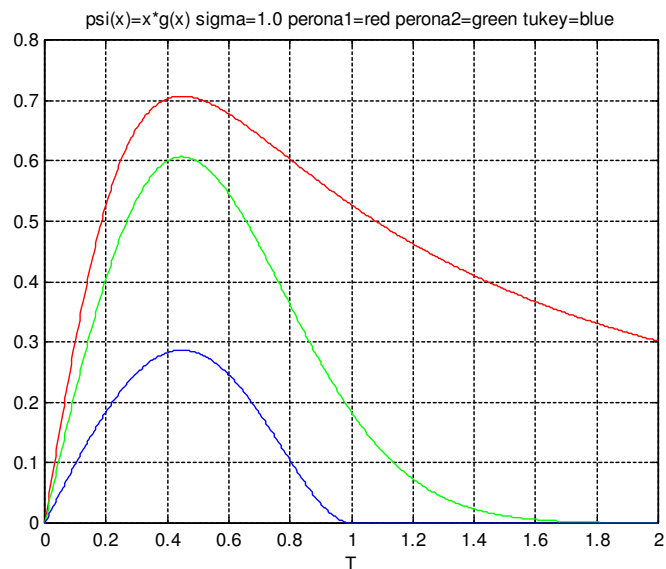


(3.3b) Funções de influência não-normalizadas: ψ_1 (Perona-Malik 1, em vermelho), ψ_2 (Perona-Malik 2, em verde) e ψ_3 (Tukey's biweight, em azul).

Fig. 3.3: Continua na próxima página.



(3.3c) Funções parada-na-aresta normalizadas com $\sigma=1$: g_1 (Perona-Malik 1, em vermelho), g_2 (Perona-Malik 2, em verde) e g_3 (Tukey's biweight, em azul).



(3.3d) Funções de influência normalizadas: ψ_1 (Perona-Malik 1, em vermelho), ψ_2 (Perona-Malik 2, em verde) e ψ_3 (Tukey's biweight, em azul).

Fig. 3.3: Funções parada-na-aresta e de influência, antes e depois da normalização.

Mudando um pouco o programa de transferência de calor, obtemos a difusão anisotrópica:

```
#include <proeikon>
#include <imgpv>

double g3(double x, double sigma)
{ if (abs(x)<=sigma)
  return elev2(1-(x*x)/(sigma*sigma));
  else
  return 0.0;
}

int main()
{ IMGFLT a; le(a,"letrarui.tga");
  I3DFLT I(20,a.nl(),a.nc());
  double lambda=1.0;
  double sigma=0.3;
  I(0)=a;
  for (int i=0; i<I.ns()-1; i++) {
    for (int l=0; l<I.nl(); l++)
      for (int c=0; c<I.nc(); c++) {
        double soma =0.0;
        double grad=(I(i)(l-1,c,'x')-I(i)(l,c,'x'));
        soma += g3(grad,sigma)*grad;
        grad=(I(i)(l+1,c,'x')-I(i)(l,c,'x'));
        soma += g3(grad,sigma)*grad;
        grad=(I(i)(l,c-1,'x')-I(i)(l,c,'x'));
        soma += g3(grad,sigma)*grad;
        grad=(I(i)(l,c+1,'x')-I(i)(l,c,'x'));
        soma += g3(grad,sigma)*grad;
        I(i+1)(l,c,'x')=I(i)(l,c,'x')+(lambda/4.0)*soma;
      }
    //pvMostra(IMGGRY(I(i+1)));
    //char st[256];
    //sprintf(st,"calor%02d.tga",i);
    //imp(IMGGRY(I(i+1)),st);
  }
  I3DGRY g(I.ns(),I.nl(),I.nc());
  for (int i=0; i<I.ns(); i++) g(i)=I(i);
  imp(g,"difanis.avi");
}
```

Filtragem de sinais unidimensionais pela difusão anisotrópica

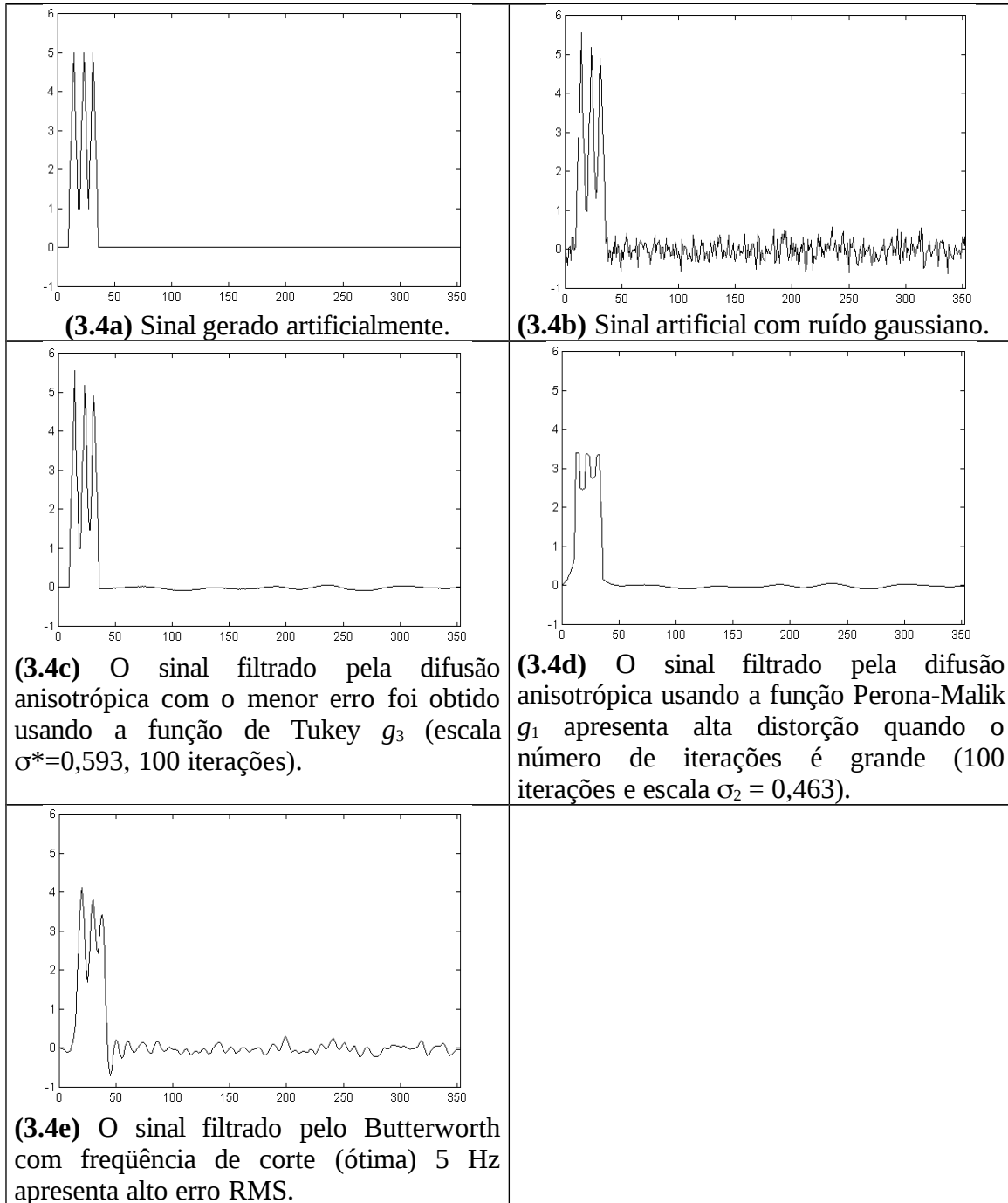


Fig. 3.4: Filtragem de um sinal sintetizado pela difusão anisotrópica.

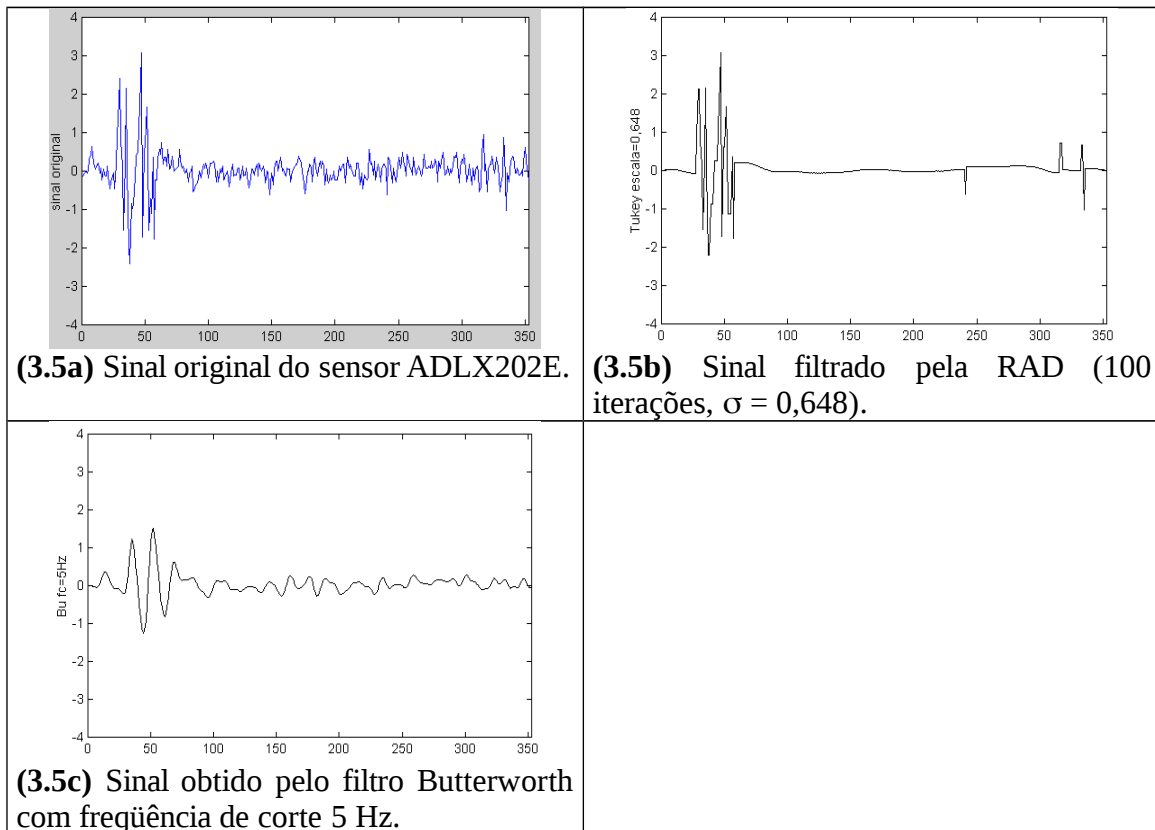


Fig. 3.5: Filtragem do sinal do sensor de aceleração ADLX202E pela difusão anisotrópica e pelo filtro Butterworth.

Detecção de arestas pela difusão anisotrópica

A figura 3.6 mostra a detecção de arestas usando várias funções parada-na-aresta e diferentes escalas σ . O número de iterações foi mantido fixo em $t_{max} = 50$. Uma comparação visual entre as figuras 3.2 e 3.6 permite constatar que a difusão anisotrópica preserva muito melhor a nitidez e a localização das bordas do que a difusão isotrópica.

A figura 3.7 permite constatar a superioridade da função parada-na-aresta de Tukey sobre aquelas de Perona-Malik. Compare as imagens da figura 3.7 (500 iterações) com as imagens da última linha da figura 3.6 (50 iterações). Todas essas imagens foram obtidas usando a escala $\sigma=0,08$. Quando o número de iterações é grande a função g_1 , e em menor grau a função g_2 , borra as arestas. Enquanto isso a RAD (g_3) mantém as arestas perfeitamente nítidas.




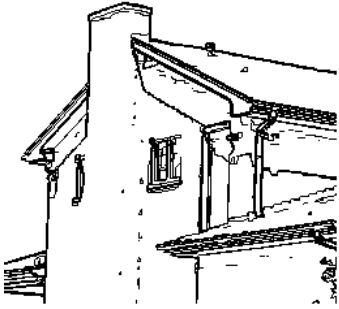
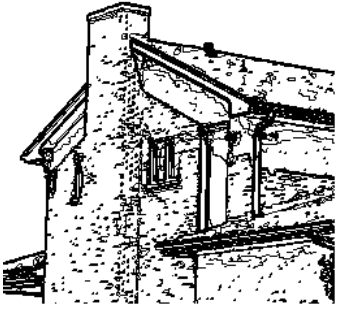




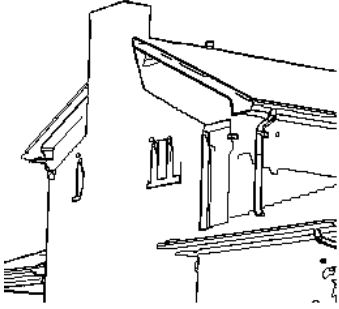
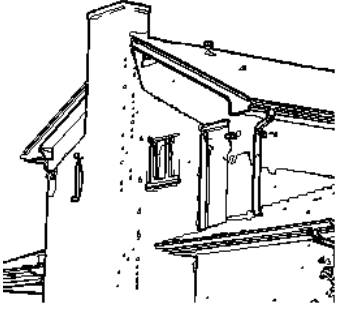
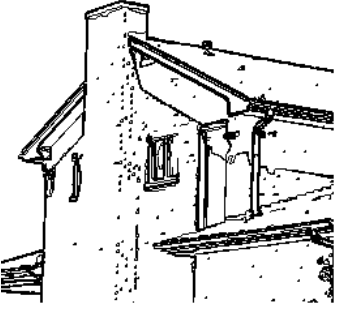
Perona-Malik 1 (g_1)	Perona-Malik 2 (g_2)	RAD (g_3)
		
 <p data-bbox="309 869 510 902">(3.6a) $\sigma = 0,02$</p>	 <p data-bbox="697 869 898 902">(3.6b) $\sigma = 0,02$</p>	 <p data-bbox="1085 869 1286 902">(3.6c) $\sigma = 0,02$</p>
		
 <p data-bbox="309 1590 510 1624">(3.6d) $\sigma = 0,04$</p>	 <p data-bbox="697 1590 898 1624">(3.6e) $\sigma = 0,04$</p>	 <p data-bbox="1085 1590 1286 1624">(3.6f) $\sigma = 0,04$</p>

Fig. 3.6: Continua na próxima página.

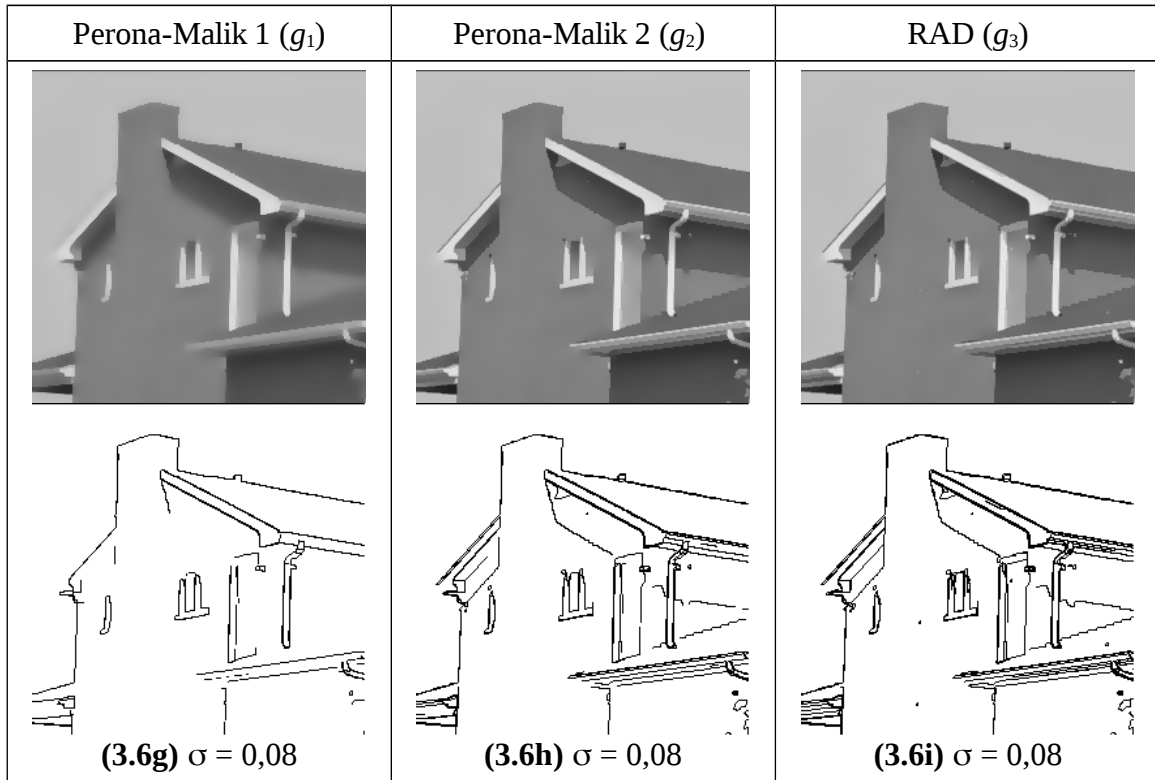


Fig. 3.6: Detecção de arestas usando a difusão anisotrópica com diferentes funções parada-na-aresta e várias escalas σ . O número de iterações foi mantido fixo em $t_{max} = 50$.




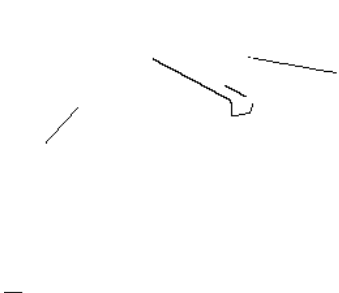
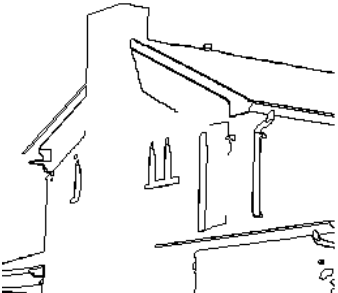
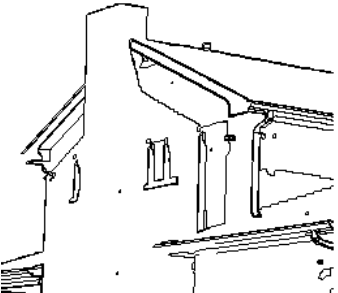
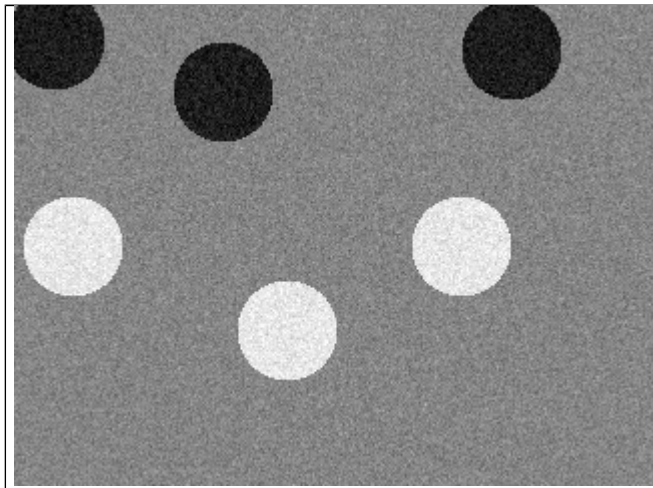
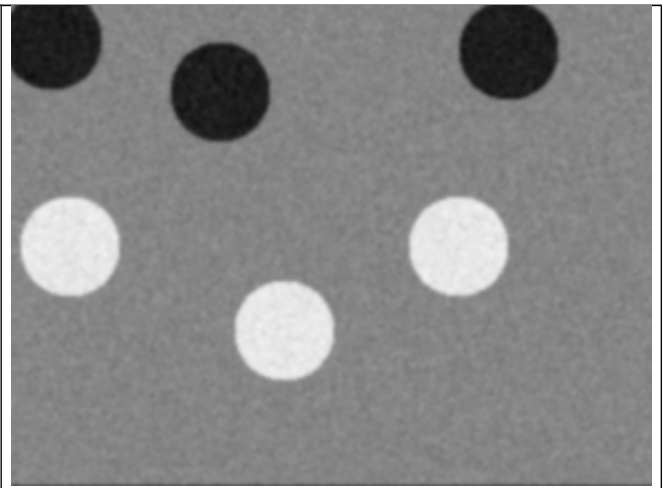
Perona-Malik 1 (g_1)	Perona-Malik 2 (g_2)	RAD (g_3)
		
		
(3.7a) $t_{max} = 500, \sigma = 0,08$	(3.7b) $t_{max} = 500, \sigma = 0,08$	(3.7c) $t_{max} = 500, \sigma = 0,08$

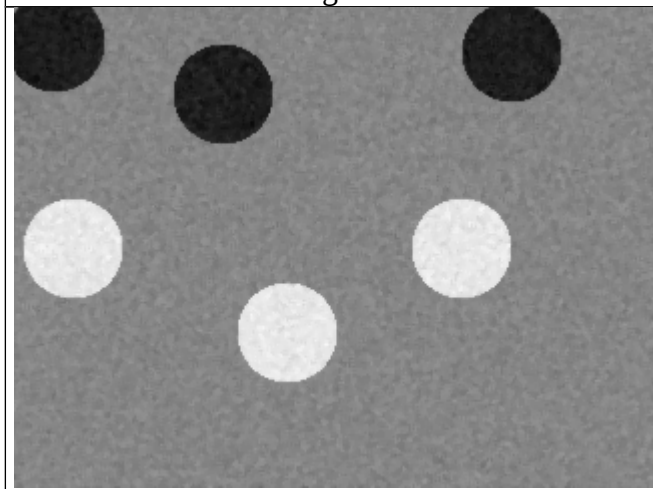
Fig. 3.7: Comportamento da difusão anisotrópica com grande número de iterações ($t_{max} = 500$). A função parada-na-aresta Perona-Malik 1 acaba borrando as arestas. A função de Tukey é a que consegue manter as arestas mais nítidas, pois está baseada na estatística robusta. Compare com a última linha da figura 3.6, onde tínhamos $t_{max} = 50$.



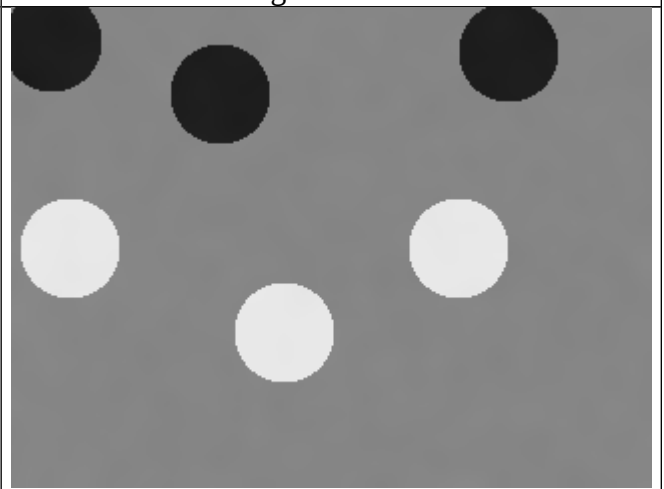
original



gaussiana



mediana



difusão anisotrópica

Colocar exemplo que sobra ruído com difusão anisotrópica

Explicar escala robusta

Explicar speckle reducing diffusion

Explicar difusão direcional

[Aurélio, 1999] Aurélio Buarque de Holanda Ferreira, *Dicionário Aurélio Eletrônico Século XXI*, 1999.

[Black et al., 1998] M. J. Black, G. Sapiro, D. H. Marimont, and D. Heeger, “Robust Anisotropic Diffusion,” *IEEE T. Image Processing*, vol. 7, no. 3, pp. 421-432, March 1998.

[Jackway and Deriche, 1996] P.T. Jackway and M. Deriche, “Scale-Space Properties of the Multiscale Morphological Dilation-Erosion,” *IEEE T. Pattern Analysis and Machine Intell.*, vol. 18, no. 1, pp. 38-51, 1996.

[Lindeberg, 1994] T. Lindeberg, *Scale-Space Theory in Computer Vision*, Kluwer, 1994.

[Perona and Malik, 1987] P. Perona and J. Malik, “Scale Space and Edge Detection Using Anisotropic Diffusion,” in Proc. *IEEE Comp. Soc. Workshop Computer Vision*, pp. 16-27, 1987.

[Perona and Malik, 1990] P. Perona and J. Malik, “Scale-Space and Edge Detection Using Anisotropic Diffusion,” *IEEE. Trans. Patt. Anal. and Machine Intell.*, vol. 12, no. 7, pp 629-639, 1990.

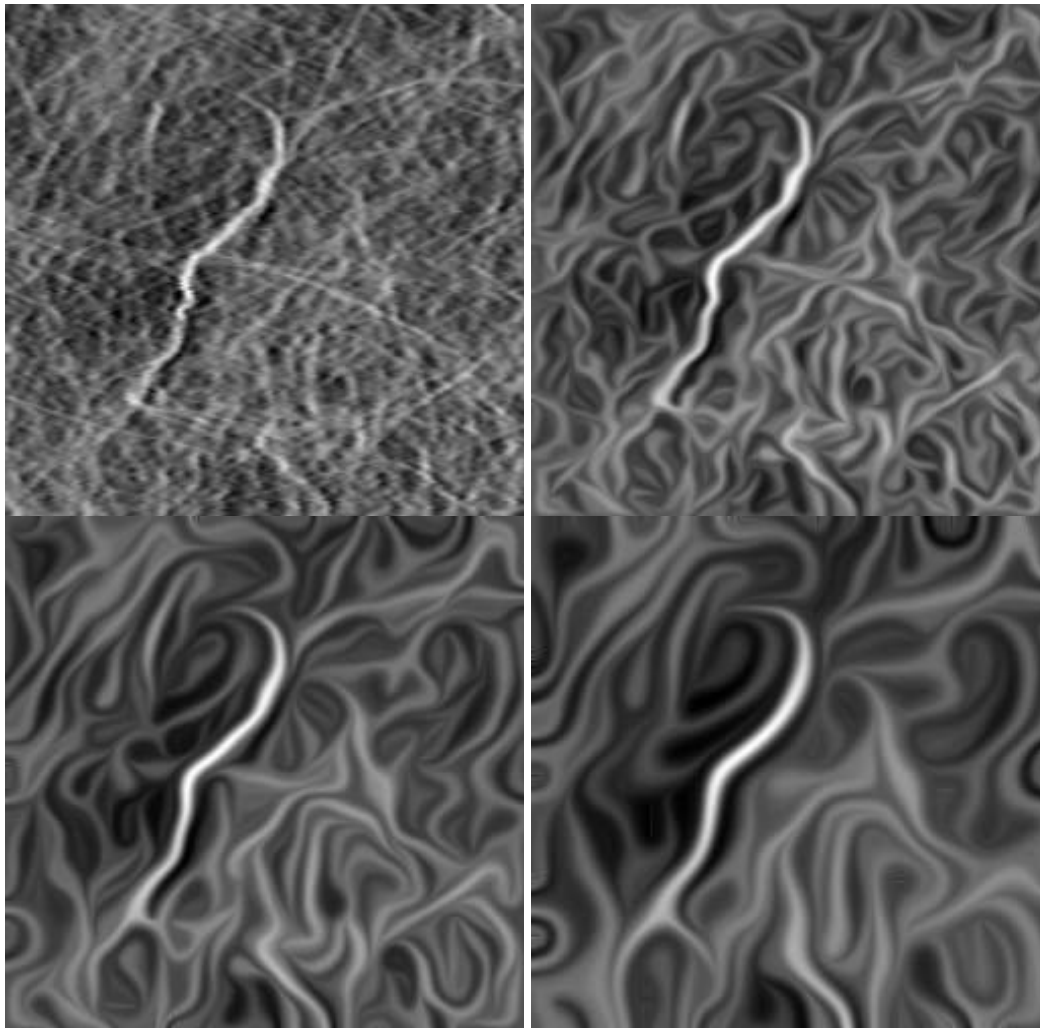
[Velho et al., 2000] L. Velho, R. Teira and J. Gones, *Introdução aos Espaços de Escala*, 12ª Escola de Computação, 2000.

[Witkin, 1983] A. P. Witkin, “Scale-Space Filtering,” *Proc. 8th Int. Joint Conf. Art. Intelligence*, vol. 2, pp. 1019-1022, 1983.

[Weickert, 1999] Joachim Weickert, “Coherence Enhancing Diffusion Filtering,” *International Journal of Computer Vision*, Vol. 31, No 2/3, pp. 111-127, April 1999.



Figuras retiradas de [Weickert, 1999]



Scale space behaviour of coherence enhancing diffusion [Weickert, 1999]