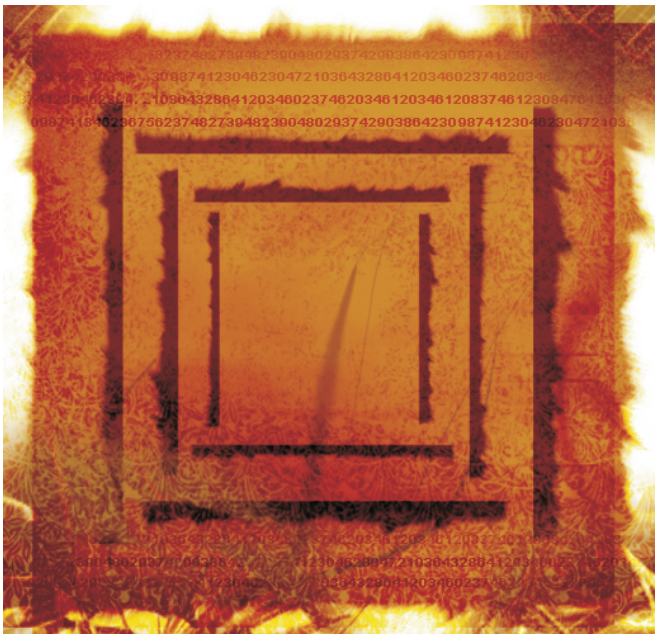
Comunicado Técnico 96

Dezembro, 2008
Campinas, SP



ISSN 1677-9274

Filtro de Canny para detecção de bordas: implementação Java

José Iguelmar Miranda[1]
João Camargo Neto[2]

O objetivo deste comunicado é apresentar a implementação Java™ do filtro para detecção de bordas de Canny (Canny, 1986). A implementação é uma necessidade de uso em projetos desenvolvidos na Embrapa Informática Agropecuária na área de processamento de imagens aplicado à agropecuária.

O processo de detecção de bordas dos objetos de uma imagem é essencial para a análise de imagens, porque podem ser detectadas precisamente e os objetos podem ser localizados e suas propriedades básicas, como área e perímetro a serem medidos. Trata-se de um dos processos mais comuns nas técnicas de análise de imagens digitais, contando com uma grande variedade de algoritmos. Uma introdução sobre o significado de bordas em imagens pode ser consultado em Miranda & Camargo Neto (2006a). Também em Miranda & Camargo Neto (2006a) encontra-se a implementação Java do filtro de difusão linear complexa, derivado do trabalho de Gilboa et al. (2004), que desenvolveram um filtro de difusão linear complexa com base na ideia de que em várias áreas da física e da engenharia podia-se estender a análise do domínio real para o domínio complexo.

Miranda & Camargo Neto (2006b) também implementaram, em Java, um filtro de difusão complexa não linear, ou anisotrópico, tomando por base as propriedades do filtro anterior, de difusão linear complexa (Gilboa et al., 2004). O objetivo desse filtro é atenuar ruídos e realçar bordas da imagem. Os autores denominaram esse filtro de "preservação de rampa" para atenuação de ruídos, com base no fato de que funções de rampa podem ser usadas como modelo da estrutura básica das bordas das imagens. E em Miranda & Camargo Neto (2007), é apresentado mais um filtro com base em equações diferencias parciais ou modelos de difusão anisotrópica (não linear). O formalismo desse problema é a ideia da filtragem, ou transformação, do espaço-escala. A ideia essencial para essa abordagem tem como base envolver a imagem original numa família de imagens derivadas, obtidas pela convolução da imagem original com um filtro Gaussiano, com variância ("tempo") *t* (Perona & Malik, 1990). Tecnicamente, a *detecção de bordas* visa localizar os *pixels* de borda enquanto o *realce de bordas* incrementa o contraste entre as bordas e o fundo, tornando as bordas mais visíveis (Parker, 1997). Na prática, ambos os termos são usados com a mesma finalidade.

Com esta publicação, dá-se prosseguimento na implementação de algoritmos úteis para a detecção de

[1] *Ph.D. em Geoprocessamento, Analista da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP. (e-mail: miranda@cnptia.embrapa.br)*
[2] *Ph.D. em Processamento de imagens, Analista da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP. (e-mail: camargo@cnptia.embrapa.br)*

Embrapa

bordas, atividade importante também no campo de visão por computador. O objetivo maior destas implementações é a constituição de uma biblioteca de processamento de imagens em Java, como software livre, sob a GNU (Licença Pública Geral) conforme publicada pela *Free Software Foundation*. Todas as implementações se encontram disponíveis no diretório da rede Agrolivre:
(<http://repositorio.agrolivre.gov.br/projects/pid/>).

O filtro de Canny

De uma maneira geral, é requerido de todo algoritmo de detecção de bordas uma baixa taxa de erro na sua identificação. Igualmente, espera-se que os *pixels* das bordas sejam localizados com precisão, minimizando a distância entre os *pixels* encontrados pelo detetor e a borda verdadeira. Um terceiro critério na detecção deveria evitar a possibilidade da geração de múltiplas bordas para expressar uma borda simples. Canny (1986) considerou essas três questões no seu trabalho, que poderiam ser assim anunciadas:

1. Boa detecção: o detetor de bordas deveria ter baixa probabilidade de falhar na detecção das bordas e baixa probabilidade de marcar pixels fora das bordas.
2. Localização: a distância entre os pixels da borda encontrados pelo detetor de bordas e a borda verdadeira deveria ser a menor possível.
3. Resposta simples: minimizar o número de bordas. O detector de bordas não poderia identificar múltiplas bordas aonde existe somente uma.

Segundo o autor, o problema seria transformar esses critérios em formalismo matemático. Ele iniciou seu trabalho analisando o comportamento dos critérios acima em uma dimensão. Para o primeiro critério, considerou a razão entre sinal e ruído. Para o sinal, considerar *f(x)* a resposta do filtro a uma borda *G(x)*. A resposta do filtro a essa borda, com centro em H_G, é dada por uma integral de convolução (Canny, 1986):

$$H_G = \int_{-w}^{w} G(-x)\, f(x)\, dx \qquad (1)$$

Sendo que o filtro tem uma resposta de pulso limitada por *[-w, w]* e zero fora desse intervalo. Para o ruído, *n(x)*, considerar a raiz quadrada de sua resposta (Canny, 1986):

$$H_n = n_0 \left[\int_{-w}^{w} f^2(x)\, dx \right]^{1/2} \qquad (2)$$

Sendo n_0^2 o a amplitude quadrática média do ruído por unidade de comprimento. Portanto, o primeiro critério pode ser formalizado como a razão entre essas duas respostas (Canny, 1986):

$$SNR = \frac{\left| \int_{-W}^{W} G(-x)\, f(x)\, dx \right|}{n_0 \sqrt{\int_{-w}^{w} f^2(x)\, dx}} \qquad (3)$$

Para o segundo critério, Canny considerou a necessidade de uma medida que crescesse à medida que a localização melhorasse e, para isso, usou a inversa da distância da raiz da média quadrática. A marcação de bordas acontece em máximos locais da resposta do operador, *f(x)*, ou seja, onde sua primeira derivada seja nula. Os detalhes para mostrar como o autor chegou à medida da localização podem ser obtidos em Canny (1986):

$$SNR = \frac{\left| \int_{-W}^{W} G(-x)\, f(x)\, dx \right|}{n_0 \sqrt{\int_{-w}^{w} f^2(x)\, dx}} \qquad (4)$$

Canny demonstrou que usando apenas os dois primeiros critérios, o detector ótimo para bordas com ruídos (Fig. 1a) produz também um resultado truncado, semelhante a usar um filtro de diferença de caixas (Fig. 1c). Esse tipo de filtro, por ter uma largura de banda grande, tende a produzir muitos máximos locais, os quais deveriam ser considerados errados, de acordo com o primeiro critério. A distância média entre os máximos adjacentes e a resposta do ruído de *f(x)*, denominada X_{max}, restringe a escolha de *f(x)* de acordo com um critério simples, o terceiro:

$$X_{max}\,[f(x)] = kW \qquad (5)$$

Onde *k* é uma fração de um coeficiente definida e *W*, a largura do operador. Devido à complexidade da formulação, nenhuma solução analítica foi encontrada. Uma variante foi desenvolvida permitindo a determinação de bordas. Para valores pequenos de X_{max}, o operador de Canny se aproxima da ação do filtro caixa de diferença (Fig. 1c). Para grandes valores de X_{max}, ele se aproxima da ação de um filtro do tipo primeira derivada da Gaussiana (Fig. 1b), também conhecido por DoG ("*derivative of Gaussian*").

No caso bidimensional, a função Gaussiana é dada por Parker (1997):

$$G(x, y) = \sigma^2 \exp - \left(\frac{x^2 + y^2}{2\sigma^2} \right) \qquad (6)$$

Portanto, a aproximação do filtro ótimo de Canny para a detecção de bordas de imagens digitais é a primeira derivada da função Gaussiana, conseguido pela convolução da imagem de entrada com *G'*. A imagem de saída desse processo terá suas bordas realçadas,

mesmo na presença de ruídos, incorporado no modelo de bordas da imagem.

Para otimizar o processo computacionalmente, a convolução bidimensional, expressa em (6), pode ser dividida em duas convoluções com Gaussianas unidimensional, sendo a diferenciação feita a posteriori. O leitor interessado em maiores detalhes pode consultar o extenso artigo de Canny (1986).

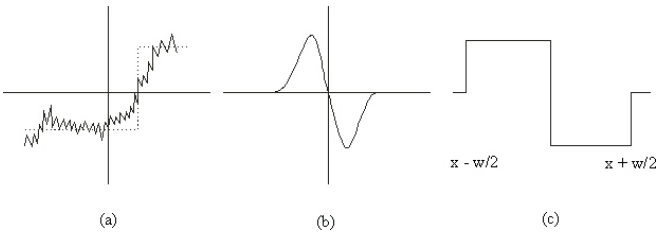


Fig. 1. Sinal com ruído, filtro da primeira derivada Gaussiana e filtro de caixa de diferença.

A Fig. 2 mostra o efeito, aproximado, dos filtros primeira derivada Gaussiana, Fig. 1b, e caixa de diferença, Fig. 1c, no sinal mostrado na Fig. 1a. Percebe-se que o sinal filtrado com o filtro diferença de caixa ainda apresenta problemas de máximos local.

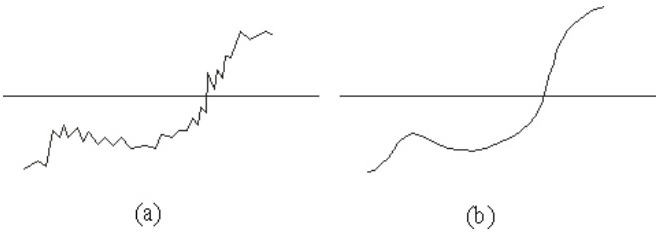


Fig. 2. Efeito dos filtros primeira derivada Gaussiana e caixa de diferença no sinal mostrado na Fig. 1a.

Aspectos da implementação

Para implementar o filtro de Canny, desenvolvemos uma aplicação Java pura, contendo apenas um construtor e uma chamada através do `main()`. Evitamos o uso de ambiente de desenvolvimento integrado, como NetBeans, ou uma interface mais rebuscada. O objetivo maior é disponibilizar o filtro de maneira limpa, para que possa ser inserido em outras aplicações. O programa deve ser compilado e executado em uma janela DOS, ou um TERM do Linux. Para compilar o programa:

```
C:[diretório de instalação]>javac -deprecation FiltroCanny.java
```

E para executá-lo:

```
C:[diretório de instalação]>java FiltroCanny <imagem> <dp> <inf> <sup>
```

Onde os parâmetros são:

`<imagem>` – a imagem a ser filtrada;
`<dp>` – o desvio padrão;
`<inf>` – o valor do limiar inferior;
`<sup>` – o valor do limiar superior.

Os valores de desvio padrão e limiares devem ser testados pelo usuário, até que se encontre uma imagem de saída com bom resultado. Sugerimos a construção de uma interface gráfica que permita mudar esses valores com um JSlider, o que facilitaria analisar o comportamento das bordas com as mudanças. O construtor do filtro de Canny é definido como:

```
public FiltroCanny(String aFile, double s,
int inf, int sup) {
 ...
}
```

onde :

1. `aFile`: nome do arquivo com a imagem digital a ser filtrada;
2. `s`: desvio padrão;
3. `inf`: no programa, a identificação dos *pixels* que pertencem à borda é feito com um processo de limiarização. Começando nos *pixels* com valores maiores que o limiar superior (*sup*), traça uma sequência conectada de pixels que possuem valores maiores que o limiar inferior (`inf`);
4. `sup`: comentário anterior.

O construtor tem quatro tarefas: fazer a leitura da imagem a ser processada; chamar o método canny, que implementa o filtro de Canny; definir os pixels da borda, usando os valores passados como parâmetros, `inf` e `sup`, e mostrar a imagem resultante. O método canny implementa os principais passos do algoritmo, assim descritos (Parker, 1997):

1. Criação de uma máscara Gaussiana, de uma dimensão, para fazer a convolução da imagem original. O parâmetro desvio padrão, `s`, da função Gaussiana é usado neste passo.

   ```
   double funcGauss[]
   ```

2. Criação de uma máscara, de uma dimensão, para a primeira derivada da função Gaussiana nas direções `x` e `y`. O valor do desvio padrão é usado também aqui.

   ```
   double derivadaGauss[]
   ```

3. Fazer a convolução da imagem original com a Gaussiana, ao longo das linhas para produzir a imagem componente em `x`, `componenteX`, e ao longo das colunas, para produzir a imagem componente em `y`, `componenteY`.

   ```
   convolveImagemXY(imagem,funcGauss,
   ```

```
width, componenteX, componenteY);
```

4. Fazer a convolução a imagem componente em `x`, `componenteX`, para produzir a imagem derivada na direção `x`, `derivadaX`, e da imagem componente em `y`, `componenteY`, para produzir a imagem derivada na direção `y`, `derivadaY`.

```
derivadaX =
convolveDerivadaXY(componenteX,   nr,
nc, derivadaGauss,   width, 1);
derivadaY =
convolveDerivadaXY(componenteY,   nr,
nc, derivadaGauss,  width, 0);
```

5. Combinar as duas imagens derivadas, com o cálculo da magnitude ou norma do gradiente, para obter a imagem resultado:

```
z = norma(derivadaX[j][i],
derivadaY[j][i]);
```

6. Suprimir os falsos máximos, com o objetivo de zerar os pixels que não sejam máximos local:

```
removeFalsoMax(derivadaX,
 derivadaY, nr, nc, imagemMag,
   imagemOrig);
```

Estudo de caso

Na sequência de figuras a seguir, mostramos o resultado da aplicação do filtro de Canny para detecção de bordas. Em todas as imagens, os valores dos parâmetros de limiarização foram `inf` = 10 e `sup` = 20. Variamos apenas os valores do desvio padrão, que mostram comportamentos diferentes do filtro. A primeira figura mostra a detecção de bordas em objetos simples. A Fig. 3a, mostra a imagem original. Em (b), passamos o filtro com `s` = 1,5 e em (c), com `s` = 2,6. O primeiro valor de desvio padrão permite que o filtro detecte as bordas dos objetos, mas mostra um efeito colateral, com a passagem de muito ruído proveniente da cor de fundo da imagem. Com um valor maior do desvio padrão, esse ruído é filtrado e podemos obter um resultado melhor do processo de detecção das bordas.

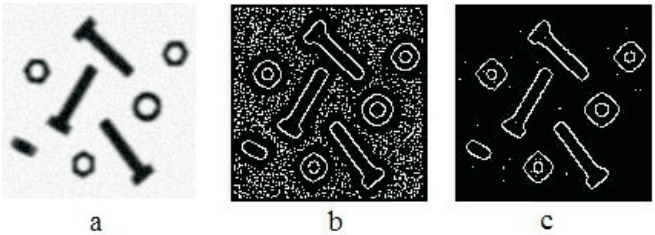


Fig. 3. (a) imagem original, (b) `s` = 1,5 e (c) `s` = 2,6.

A Fig. 4 mostra uma área de agricultura em Guaíra, SP, com pivôs central (áreas circulares), algumas áreas de solo nu (área branca), cana-de-açúcar e matas ciliares ao longo de córregos. A primeira imagem foi filtrada com `s` = 1,5. À semelhança do que ocorreu com a figura anterior, esse valor de desvio padrão permite a passagem de muito ruído, o que na realidade, são bordas de pequenos objetos da imagem. O resultado, visualmente, não é bom. Um incremento no valor do desvio padrão permite filtrar as bordas de pequenos objetos, "limpando" mais a imagem final, dando um efeito visual melhor. À medida que incrementados o valor do desvio padrão, a imagem aparece mais limpa, ao custo do algoritmo detectar menos bordas.

Fig. 4. (a) imagem original, (b) `s` = 1,5 e (c) `s` = 2,2.

A Fig. 5 mostra um conjunto de glóbulos. O primeiro filtro usou também um valor de desvio padrão de 1,5, apresentando resultados semelhantes aos anteriores. O segundo filtro usou um valor de 2,6, apresentando uma imagem mais "limpa". Nesse caso, à semelhança do que aconteceu com a Fig. 3, onde temos apenas objetos, a sua discriminação é bem melhor. Na Fig. 4, uma imagem com muitos objetos e nuanças, o processo de filtragem fica comprometido no delineamento dos objetos da cena.

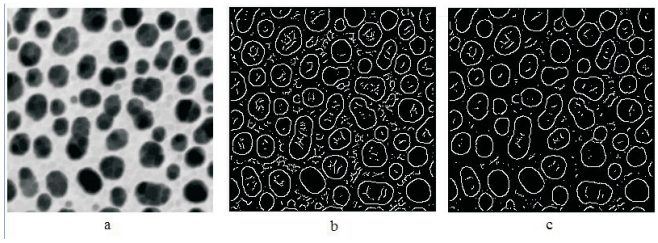


Fig. 5. (a) imagem original, (b) **s** = 1,5 e (c) **s** = 2,6.

Conclusões

- A utilização do filtro de Canny se mostrou eficiente para a detecção de bordas;
- Sua utilização em cenas com objetos distintos é eficiente, mas em cenas com objetos pequenos em grande quantidade há um comprometimento de sua eficiência, mas isso não chega a comprometer sua eficiência. Nesses casos, um filtro prévio de filtragem de altas frequências ajudaria no delineamento de objetos maiores;
- A implementação Java da versão do filtro se encontra disponível e operacional;
- O fonte, considerado software livre, está sob a GNU (Licença Pública Geral) conforme publicada pela *Free Software Foundation*, e disponível no diretório da

rede Agrolivre:
(<http://repositorio.agrolivre.gov.br/projects/pid/>).

Referências Bibliográficas

CANNY, J. A computational approach to edge detection. *IEEE Transactions on Patterns Analysis and Machine Intelligence*, v. 8, n. 6, p. 679-698, 1986.

GILBOA, G.; SOCHE, N.; ZEEVI, Y. Y. Image enhancement and denoising by complex diffusion processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 26, n. 8, p. 1020-1036, 2004.

MIRANDA, J. I.; CAMARGO NETO, J. Detecção de bordas com o modelo de difusão anisotrópica. In: SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO (SBSR), 13, 2007, Florianópolis. *Anais...* São José dos Campos: INPE, 2007. p. 5957-5964. CD-ROM. Disponível em: <http://www.dsr.inpe.br/sbsr2007>. Acesso em: 15 set. 2008.

MIRANDA, J. I.; CAMARGO NETO, J. *Filtro de difusão linear complexa para detecção de bordas: implementação Java*. Campinas: Embrapa Informática Agropecuária, 2006a. 5 p. (Embrapa Informática Agropecuária. Comunicado Técnico, 75). Disponível em:
<http://www.cnptia.embrapa.br/modules/tinycontent3/content/2006/ct75.pdf>. Acesso em: 15 set. 2008.

MIRANDA, J. I.; CAMARGO NETO, J. *Filtro de difusão complexa não linear para atenuação de ruídos: implementação Java*. Campinas: Embrapa Informática Agropecuária, 2006b. 4 p. (Embrapa Informática Agropecuária. Comunicado Técnico, 72). Disponível em:
<http://www.cnptia.embrapa.br/modules/tinycontent3/content/2006/ct72.pdf>. Acesso em: 15 set. 2008.

PARKER, J. R. *Algorithms for image processing and computer vision*. New York, NY: John Wiley, 1997. 417 p.

PERONA, P.; MALIK, J. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Patterns Analysis and Machine Intelligence*, v. 12, n. 7, p. 629-639, 1990.

Comunicado Técnico, 96

Ministério da Agricultura, Pecuária e Abastecimento

BRASIL UM PAÍS DE TODOS GOVERNO FEDERAL

Embrapa Informática Agropecuária
Área de Comunicação e Negócios (ACN)
Endereço: Caixa Postal 6041 - Barão Geraldo
13083-886 - Campinas, SP
Fone: (19) 3211-5743
Fax: (19) 3211-5754
URL: http://www.cnptia.embrapa.br
e-mail: sac@cnptia.embrapa.com.br

1ª edição on-line - 2008