PIM

Prof. Gilmário

Trabalho Prático – aplicação do operador gradiente na detecção de bordas
Salvo recomendação alguma explícita, não utilize operadores já prontos do

Implemente as convoluções necessárias.

Veja os detalhes da entrega no link de upload no Moodle

Entrega acompanhada de relatório

I) Objetivos

Estudo da aplicação de filtro passa baixa e o filtro diferencial (Sobel, Prewitt) na composição do operador gradiente na detecção de pixels de borda de objetos. Utilização de pacotes Python skimage.SSIM e skimage.filters.

II) Descrição da arquitetura do operador desejado:

Uma imagem em tons de cinza é uma função bidimensional f(x,y) que pode ser interpretada como uma superfície contendo aclives e declives correspondentes às transições, mais ou menos acentuadas, de intensidades de brilho. Essas transições podem ser úteis, por exemplo, como fonte de informação para configurar um descritor da imagem ou para uma operação de realce de bordas de elementos da imagem.

A taxa de variação pela transição de regiões de brilho na imagem pode ser detectada, em cada coordenada de pixel, pela magnitude do vetor $|\nabla(f)|$ obtido a partir da aplicação do operador gradiente $\nabla(f)$. Como o gradiente é um vetor, a sua determinação depende dos componentes horizontal e vertical (Gx,Gy) que o compõem em cada coordenada da imagem,

$$G_x = \frac{\partial f}{\partial x}$$

$$G_y = \frac{\partial f}{\partial y}$$

$$\nabla(f) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

O cálculo dos componentes equivale à aplicação de derivadas direcionais no domínio discreto, as quais podem ser realizadas por filtros derivativos adequados tais como Prewitt ou Sobel.

Além da aplicação de filtros derivativos existem algumas operações adicionais que podem ser necessárias. Filtros derivativos são muito sensíveis a ruídos na imagem. Se imagem de entrada estiver “ruidosa”, será necessário aplicar um filtro adequado para atenuação desse ruído.

As etapas básicas da aplicação do operador gradiente constam abaixo, na Figura 1.

OBSERVAÇÃO: caso a imagem de teste esteja em RGB, antes de qualquer operação, a converta para tons de cinza.

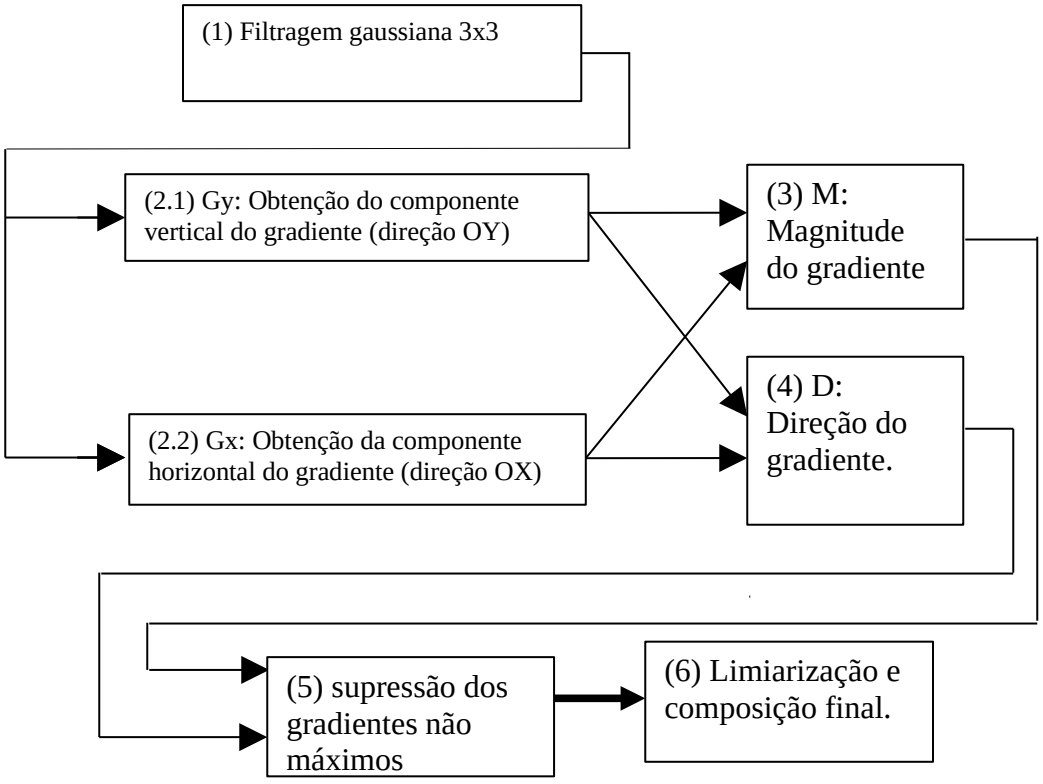

Figura 1: Fluxograma do cálculo do gradiente de uma imagem em tons de cinza.

1) Pré-filtragem

As etapas 2.1 e 2.2 são realizadas por filtros (*h*) baseados em derivadas, os quais são sensíveis às componentes de alta frequência. Por conta disso, a aplicação

destes filtros é precedida por uma atenuação dos ruídos (leia a seção 2.9 do Pedrini [2]) na imagem a ser tratada.

2) Aplicação do filtro derivativo
2.1) Determinação das componentes horizontais (Gx) utilizando máscara adequada (Sobel, Prewitt) para a direção OX (taxa de variação entre colunas da imagem);
2.2) Determinação das componentes verticais (Gy) utilizando máscara adequada (Sobel, Prewitt) para a direção OY (taxa de variação entre linhas da imagem);

Atenção: você não pode misturar operadores, por exemplo: aplicar Gx por Sobel e Gy por Prewitt.

3) Determinação da magnitude do gradiente (M)
As matrizes/imagens Gx e Gy são utilizadas para a construção da matriz/imagem M, a qual representa as magnitudes dos pixels de *f*.

$$M(i,j) = \sqrt{(G_y(i,j))^2 + (G_x(i,j))^2}$$

4) Determinação da direção do gradiente (D)
As matrizes/imagens Gx e Gy são utilizadas para a construção da matriz/imagem D, a qual representa as direções dos gradientes cujas magnitudes foram calculadas em M(i,j):

$$D(i,j) = \arctan \frac{G_y(i,j)}{G_x(i,j) + \varepsilon}$$

Computacionalmente, o denominador precisa sem somado a um epsilon igual a um valor próximo de zero, por exemplo, $\varepsilon = 10^{-8}$).

Sugere-se a utilização da função *math.atan2* do Python (AQUI).

5) Criando uma matriz M_g de gradientes selecionados suprimindo valores fracos demais:
Na Figura 2, suponha que A, B, C, D, E, F e G são magnitudes de gradientes, sendo que E(i,j) corresponde à magnitude na coordenada do pixel (i,j) e as demais magnitudes sejam associadas aos 8 vizinhos do E(i,j).
Você deseja avaliar se o vetor gradiente relacionado a E(i,j) em relação ao par de vizinhos que estejam mais alinhados com sua direção Θ(i,j). Por meio das direções Θ calculadas, você determina o par desejado. Caso E(i,j) seja menor do pelo menos um dos seus vizinhos então $M_g(i,j)=0$, caso contrário $M_g(i,j)=E(i,j)$

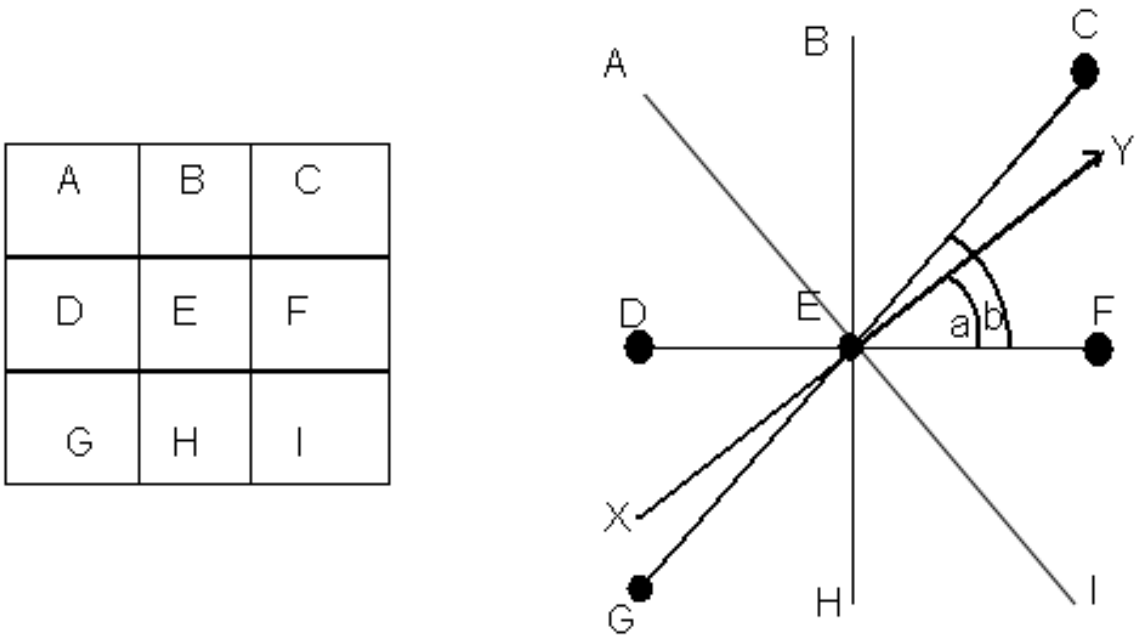

Figura 2: A direção 'a' é mais próxima de 45º do que 90º ou 0º, portanto, E será comparado a G e a C.

6) Limiarização e histerese:

Calcule a intensidade mediana de M_g e defina T_L e T_h como uma porcentagem dessa mediana. Justifique sua escolha!
M_{gH} e M_{gL} são inicialmente zeradas.

$M_{gH}(i,j)= M_g(i,j) \geq T_H$
$M_{gL}(i,j)= M_g(i,j) \geq T_L$

Calcule $M_{gL}(i,j)=|M_{gL}(i,j)-M_{gH}(i,j)|$

$M_{gL}(i,j)$ contém as bordas mais fracas e $M_{gH}(i,j)$ as mais fortes.

A partir dessas matrizes se executa uma operação chamada histerese para conectar possíveis bordas fracas que são fragmentos de bordas fortes. Leia a descrição na seção “detector de borda de Canny” no Gonzalez [1] (“The Canny Edge Detector” na 5ª edição em inglês):

III) Tarefa

A) implementação do operador gradiente (não utilize biblioteca para as convoluções dos filtros, ou seja, implemente as convoluções que aplicam os filtros):

Utilize as imagens moedas.png, Lua1_gray.jpg, chessboard_inv.png e img2.jpg disponíveis no site da disciplina no Moodle (você pode utilizar outras adicionalmente, mas não em substituição a essas imagens citadas).

Devem ser testadas duas máscaras Sobel e Prewitt. Implemente o operador gradiente de acordo com o que foi especificado anteriormente (Seção II).

A implementação completa deve ser conforme descrito na seção anterior, as escolhas precisam ser justificadas.

Compare os resultados das soluções pelas duas máscara (Prewitt e Sobel), analise quais resultados apresentaram bordas mais bem definidas e coerentes com as respectivas imagens originais. Analise o impacto da variação do valor dos limiares T_H e T_L.

$$Operadores\ de\ Prewitt: \quad Pw_x = \begin{pmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{pmatrix} Pw_y = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{pmatrix}$$

$$Operadores\ de\ Sobel: \quad Sb_x = \begin{pmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{pmatrix} Sb_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{pmatrix}$$

B) Comparativo com OPENCV

Aplique Python skimage.filters (AQUI) sobre as mesmas imagens e utilize SSIM (AQUI) para comprar os resultados obtidos pelo seu método com os resultados via OpenCV.

Compare os resultados.

Bibliografia

[1] Gonzalez, R. e Woods, R. "Processamento digital de Imagens", 3a ed. Ed. Pearson, 2010.

[2] Pedrini, Hélio. Livros Análise de Imagens Digitais - Princípios, Algoritmos e Aplicações. Editora Thomson Learning,2007.