Processamento e Análise de Imagens Digitais

Exercício 02 - *Kernels*

Artur Rodrigues Rocha Neto - 431951
Mestrando em Engenharia de Teleinformática
artur.rodrigues26@gmail.com

André Washington Morais de Freitas - 407343
Doutorando em Engenharia de Teleinformática
andre.was12@gmail.com

12 de Março de 2019

1 Introdução

Uma imagem digital nada mais é que uma matriz cujos elementos guardam valores de intensidade, armazenadas em um (tom de cinza) ou mais (cor) canais. Existem diversas operações, também chamadas de transformações, em análise de imagens digitais. Uma dessas operações mais comuns é a aplicação de *kernels*. Um *kernel*, de forma simplificada, é uma pequena matriz responsável por varrer uma imagem *pixel* a *pixel* e dessa extrair informações, atenuar características, remové-las, etc. Por isso, a aplicação de *kernel* também é conhecida como filtragem espacial, pois age no domínio do espaço da imagem. A varredura de uma imagem pode ser efetuada através da convolução ou da correlação [1].

Dada uma imagem de entrada $f(x, y)$ e um *kernel* K, podemos obter a imagem de saída $g(x, y)$ através da operação de convolução $\star$: varredura de $f(x, y)$ ao redor das vizinhanças de cada *pixel* (i, j) (Equação 1):

$$g(x,y) = \sum_i \sum_j \begin{bmatrix} \omega_0 & \omega_1 & \omega_2 \\ \omega_3 & \omega_4 & \omega_5 \\ \omega_6 & \omega_7 & \omega_8 \end{bmatrix} \star \begin{bmatrix} f(i-1,j-1) & f(i-1,j) & f(i-1,j+1) \\ f(i,j-1) & f(i,j) & f(i,j+1) \\ f(i+1,j-1) & f(i+1,j) & f(i+1,j+1) \end{bmatrix} \tag{1}$$

2 Metodologia

São explorados quatro *kernels* nesse trabalho: laplaciano, gaussiano, mediana e moda. Para cada um, pede-se:

1. Demonstrar matemáticamente sua construção; e
2. Implementar em linguagem de programação escolhida e aplicá-los em imagens de *benchmark*.

A linguagem usada para a codificação dos *kernels* foi a Python. Fez-se uso também de funções gerais de análise de imagens digitais da biblioteca OpenCV [2]. O código-fonte criado está disponível para *download*[1].

3 Resultados

A forma geral de um *kernel* linear K é dada a partir de uma função de *kernel* $w(x, y)$:

$$K = \begin{bmatrix} \omega_0 & \omega_1 & \omega_2 \\ \omega_3 & \omega_4 & \omega_5 \\ \omega_6 & \omega_7 & \omega_8 \end{bmatrix} = \begin{bmatrix} f(x-1, y-1) & f(x-1, y) & f(x-1, y+1) \\ f(x, y-1) & f(x, y) & f(x, y+1) \\ f(x+1, y-1) & f(x+1, y) & f(x+1, y+1) \end{bmatrix} \quad (2)$$

Basta substituir os coeficientes correpondentes de $f(x, y)$ na matriz para a obtenção do filtro lienar. Fazemos essa distinção pois existem alguns filtros (como o da mediana e da moda) que são aplicados de formas distintas, não necessariamente a partir de uma convolução.

Os filtros foram aplicados em duas imagens de entrada para cada amostra de *benchmark*: a imagem original e uma adicionada de ruído gaussiano ($\sigma = 20$).

3.1 Laplaciano

A derivada de segunda ordem aponta para mudanças duplas na continuidade de uma função. De maneira intuitiva, podemos pensar como uma mudança dupla na composição de uma imagem digital o salto entre três *pixels* distintos indo de um certo valor de intensidade, passando por um distinto e então voltando para um valor próximo do inicial. Esse tipo de manifestação define uma borda.

Muitos filtros são responsáveis por detectar bordas em imagens. Esse processo também é chamado de segmentação. Um dos filtros mais comuns para essa tarefa é o laplaciano. Dada uma imagem $f(x, y)$, podemos demonstrar o laplaciano como:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (3)$$

Como a derivada em qualquer direção é uma operação linear, o Laplaciano é um filtro linear. Fazendo na direção x e y separadamente, temos:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y) \quad (4)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y) \quad (5)$$

Substituindo 4 e 5 em 3, temos:

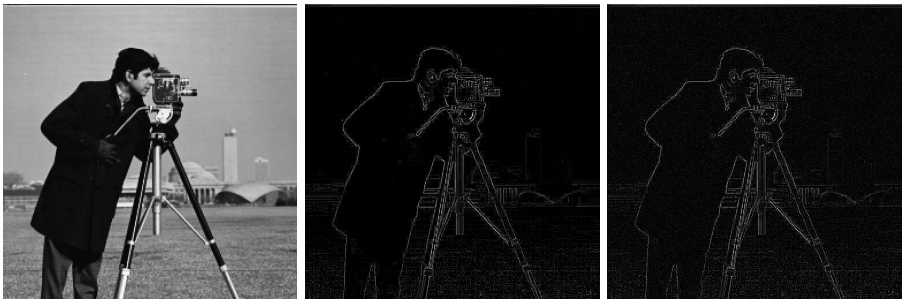
$$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y) \quad (6)$$

[1] `Https://github.com/keizerzilla/pdi-pos/blob/master/T02.py`

Por fim, basta substituir os coeficientes de 6 na matriz descrita em 2 para chegarmos à matriz de convolução:

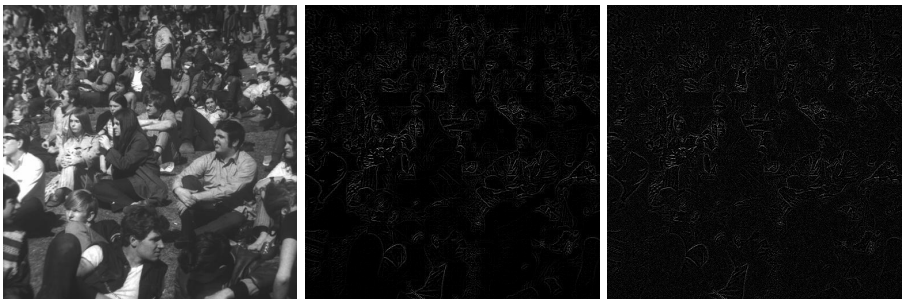
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{7}$$

As Figuras 1, 2 e 3 trazem o resultado da aplicação do filtro laplaciano em imagens de *benchmark*.



(a) Original (b) Sem ruído (c) Com ruído

Figura 1: Aplicação do filtro laplaciano em amostras com e sem ruído na imagem `cameraman.jpg`



(a) Original (b) Sem ruído (c) Com ruído

Figura 2: Aplicação do filtro laplaciano em amostras com e sem ruído na imagem `crowd2.jpg`

(a) Original (b) Sem ruído (c) Com ruído

Figura 3: Aplicação do filtro laplaciano em amostras com e sem ruído na imagem `lenna.jpg`

3.2 Gaussiano

Outra aplicação comum em processamento de imagens digitais é a suavização. Ela é muito usada para a remoção de ruídos de diversas naturezas. Um filtro simples de suavização é o filtro da média: o *pixel* gerado para a imagem de saída terá a intensidade média da vizinhança (i, j) na imagem de entrada. Outra forma de suavizar uma imagem é aplicar uma distribuição de intensidade nas vizinhanças de *pixel*, tornando a imagem de saída uma coleção de intensidades dessa distribuição.

A partir da fórmula da distribuição normal (Equação 8), podemos gerar filtros de suavização dos mais diversos, apenas trocando o parâmetro σ (desvio padrão).

$$h(x, y) = \frac{1}{2\pi\sigma^2} e^{(-\frac{x^2+y^2}{2\sigma^2})} \tag{8}$$

A Matriz em 9 é a implementação do *kernel* gaussiano usado nesse trabalho.

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{9}$$

As Figuras 4, 5 e 6 trazem o resultado da aplicação do filtro gaussiano em imagens de *benchmark*.

(a) Original (b) Sem ruído (c) Com ruído

Figura 4: Aplicação do filtro gaussiano em amostras com e sem ruído na imagem `cameraman.jpg`

(a) Original (b) Sem ruído (c) Com ruído

Figura 5: Aplicação do filtro gaussiano em amostras com e sem ruído na imagem `crowd2.jpg`

(a) Original (b) Sem ruído (c) Com ruído

Figura 6: Aplicação do filtro gaussiano em amostras com e sem ruído na imagem `lenna.jpg`

3.3 Mediana

O filtro da mediana funciona de forma diferente daqueles vistos até agora. Como o nome sugere, a medida estatística de mediana é usada para a escolha do *pixel*

na imagem de saída. Dada a vizinhação do *pixel* (i, j) na imagem de entrada, a intensidade do correspondente na saída será a mediana dos valores de intensidade da vizinhança. O filtro da mediana é não-linear usado para suavização.

As Figuras 7, 8 e 9 trazem o resultado da aplicação do filtro da mediana em imagens de *benchmark*.

(a) Original (b) Sem ruído (c) Com ruído

Figura 7: Aplicação do filtro da mediana em amostras com e sem ruído na imagem `cameraman.jpg`

(a) Original (b) Sem ruído (c) Com ruído

Figura 8: Aplicação do filtro da mediana em amostras com e sem ruído na imagem `crowd2.jpg`

(a) Original (b) Sem ruído (c) Com ruído

Figura 9: Aplicação do filtro da mediana em amostras com e sem ruído na imagem `lenna.jpg`

3.4 Moda

O filtro da moda tem demonstração semelhante ao filtro da mediana. Como o nome sugere, a medida estatística de moda é usada para a escolha do *pixel* na imagem de saída. Dada a vizinhação do *pixel* (i, j) na imagem de entrada, a intensidade do correspondente na saída será o valor mais frequente encontrado na vizinhança. O filtro da moda é não-linear usado para agussamento.

As Figuras 10, 11 e 12 trazem o resultado da aplicação do filtro da moda em imagens de *benchmark*.

(a) Original (b) Sem ruído (c) Com ruído

Figura 10: Aplicação do filtro da moda em amostras com e sem ruído na imagem `cameraman.jpg`

(a) Original (b) Sem ruído (c) Com ruído

Figura 11: Aplicação do filtro da moda em amostras com e sem ruído na imagem `crowd2.jpg`

(a) Original (b) Sem ruído (c) Com ruído

Figura 12: Aplicação do filtro da moda em amostras com e sem ruído na imagem `lenna.jpg`

Referências

[1] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.