Aplicação de Filtragem no Dominío Espacial e de Frequência

Thalles Santos Silva*
Hélio Pedrini†

Abstract

Aplicações de filtragem em imagens vêm sendo exploradas desde os primórdios da Computação. Suas aplicações estão presentes nos mais variados setores como medicina, aplicações espaciais, tecnologia e produção de filmes. Este projeto explora a aplicação de kernels prédefinidos em imagens monocromáticas. Para isso, exploramos a aplicação de filtros em dois domínios, o domínio espacial e o domínio de frequência ou de Fourier. Tanto os filtros como as imagens exploradas neste trabalho foram predefinidos em sala de aula.

1. Configuração

Este projeto foi desenvolvido em Python 3 utlizando Jupyter Notebooks [1] para execução. As principais bibliotecas utilizadas são NumPy [2] e OpenCV [3] para processamento e matplotlib [4] para visualização dos resultados. O Notebook é dividido em subseções análogas às sessões contidas neste documento.

2. Convolução no Domínio Espacial

2.1. Filtro h1

O processo de convolução de uma função *h(x,y)* sobre uma outra função *f(x,y)*, pode ser definido tanto no domínio espacial quanto no domínio de frequência. No domínio espacial, a convolução é o processo de adicionar cada elemento da imagem aos seus vizinhos locais, ponderados pelo kernel.

Aqui, definimos a função *f(x,y)* como uma imagem e *h(x,y)* como um filtro ou kernel. Existem diferentes tipos de kernel. Em nosso primeiro exemplo, temos um kernel h1 de característica passa-alta. De fato, o kernel h1 é uma aproximação do kernel Laplaciano do Gaussiano. Esse tipo de kernel viza o realce de altas frequências na imagem.

Figure 1. Resultado da aplicação do filtro passa-alta.

Visualmente, temos a impressão de que as faixas da imagem que possuem transições mais abruptas são realçadas. Oposto a isso, regiões com intensidades mais homogêneas são suprimidas. Em outras palavras, bordas, linhas e cantos tendem a ganhar destaque. Esses efeitos ficam bem claros quando analisamos o resultado na Figura 1. Aqui, percebemos que regiões homogêneas, como as partes do fundo, apresentaram pouco ganho de realce. No entanto, as borda que separam a gaivota do fundo ganham destaque no resultado. Em geral, filtros de passa-alta destacam as transições entre regiões distintas em uma imagem.

Também percebemos que o resultado do filtro passa-alta realça o ruído na imagem. Podemos atenuar esse problema, aplicando um filtro suavizador (passa-baixa) como um passo anterior. De fato, essa é a proposta do filtro Laplaciano do Gaussiano. No entanto, a intensidade deste filtro deve ser escolhida com cuidado pois, podemos suprimir as regiões de interesse da imagem.

2.2. Filtro h2

O próximo filtro, h2, é um filtro Gaussiano com características passa-baixa. Filtros Gaussianos seguem um padrão específico. Eles possuem valores mais acentuados no pixel de referência (pixel central). Porém, a medida que nos distanciamos do pixel central, os valores ficam mais baixos.

Portanto, a convolução de um filtro Gaussiano em uma imagem é análoga a média ponderada dos pixels em uma vizinhança. A ideia central é que pixels em uma vizinhança

*Is with the Institute of Computing, University of Campinas (Unicamp). **Contact**: thalles753@gmail.com

†Is with the Institute of Computing, University of Campinas (Unicamp). **Contact**: helio@ic.unicamp.br

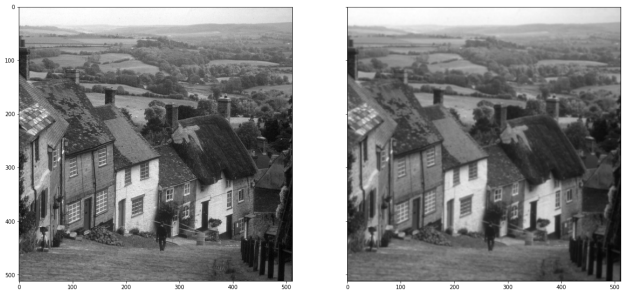

Figure 2. Resultado da aplicação do filtro passa-baixa Gaussiano.

mais próxima possuem maior influência no o resultado final do que pixels mais distantes.

Dessa forma, filtros com característica passa-baixa suprimem pixels com intensidades elevadas. Pois esses, serão suavizados pelos valores dos pixels em sua vizinhança (média ponderada). Assim, obtemos o efeito de suavização mostrado na Figure 2.

2.3. Filtros h3 e h4

Para entender a aplicação dos kernels h3 e h4, precisamos nos referir ao conceito de gradiente. Sucintamente, quando lhe damos com funções multivariáveis (como uma imagem), podemos calcular derivadas parciais com relação a cada uma das variáveis. Desta forma, o gradiente é um vetor que encapsula todas as derivadas parciais de uma função. Enquanto derivadas parciais refletem mudanças em direções específicas, o gradiente encapsula taxas de variações em todas as direções.

No contexto de imagens, a derivada é definida como a subtração dos pixels vizinhos. Esse é o conceito dos filtros passa-alta de Sobel h3 e h4. Quando aplicados h3 e h4 separadamente em uma imagem, nós obtemos a imagem do gradiente. A Figura 3 (a) e (b), mostram as derivadas parciais na direção vertical e horizontal respectivamente.

Percebemos que os filtros de Sobel conseguem achar picos nas derivadas (nas duas direções vertical e horizontal). Como as bordas representam as porções da imagem com maior mudança de intensidade (maiores derivadas), os filtros de Sobel enfatizam as bordas de uma imagem.

Finalmente, se calcularmos a magnitude do gradiente, teremos uma imagem de bordas - Figura 3 (c).

3. Notas sobre Implementação

Dada uma imagem com dimensões $M \times M$, e um filtro com dimensões $N \times N$, uma implementação clássica de convolução necessita de M^2N^2 multiplicações e adições. Note que esse é o custo total de convolução do filtro sobre toda a imagem.

No entanto, para **filtros linearmente separáveis**, (como o filtro Gaussiano e os filtros de Sobel), podemos aproveitar a propriedade de associatividade da convolução e realizar a mesma operação com um ganho significativo de performance.

Filtros linearmente separáveis são filtros que podem ser representados a partir da convolução de um vetor de coluna com o vetor de linha. Dessa forma, ao invés de fazer uma convolução entre uma imagem *f* com um filtro *h*, podemos representar a mesma operação com duas operações de convolução.

Primeiro, realizamos uma convolução simples verticalmente. Em seguida, aplicamos uma convolução horizontal sobre o resultado anterior. Vide Figura 4. Dessa forma, mesmo utilizado 2 operações de convolução, ao invés de apenas uma, obtemos uma redução no número de operações para $2NM$. Que é algumas ordens de magnitude menor que N^2M^2.

Para a aplicação do filtro Gaussiano e de Sobel implementamos a função *fast_conv2d()* que implementa convolução com kernels linearmente separaveis. A função utiliza a rotina *convolve(image, kernel)* da biblioteca NumPy.

Observe que a função *convolve(image, kernel)*, implementa uma operação discreta de convolução. Dessa maneira, antes de 'deslizar' o kernel sobre a imagem, **este é rotacionado em um ângulo de** $180^{\mathbf{o}}$. Para kernels simétricos (como o Gaussiano), tal rotação equivale a uma função identidade. No entanto, para kernels não simétricos, como Sobel as duas operações são distintas.

É importante salientar que nossa implementação de *fast_conv2d()* resolve bordas com padding de zeros (0s). Assim, ao convoluirmos uma imagem com um kernel utilizando *fast_conv2d()*, obtemos uma outra image com as mesmas dimenssões da imagem de entrada. Na literatura, essa processo é referido como "SAME" padding. Como resultado, convoluções que utilizam "SAME" padding apresentam uma borda escurecida ao redor da imagem.

Para aplicação do kernel h1 (passa-alta), utilizamos a função *filter2D()* do OpenCV. Essa função implementa a operação correlação (não convolução). Porém, como temos um filtro h1 simétrico, as duas operações produzem o mesmo resultado.

4. Convolução no Domínio de Frequência

Uma importante propriedade quando trabalhamos com imagens no domínio de frequência, é que, convolução no domínio espacial é equivalente a multiplicação no domínio de frequência. Desta forma, dada uma imagem *f* e um kernel gaussiano *h* no domínio espacial, podemos:

- Computar a transformada de Fourier para a imagem *f*.
- Computar a transformada de Fourier para o filtro Gaussiano **h**.

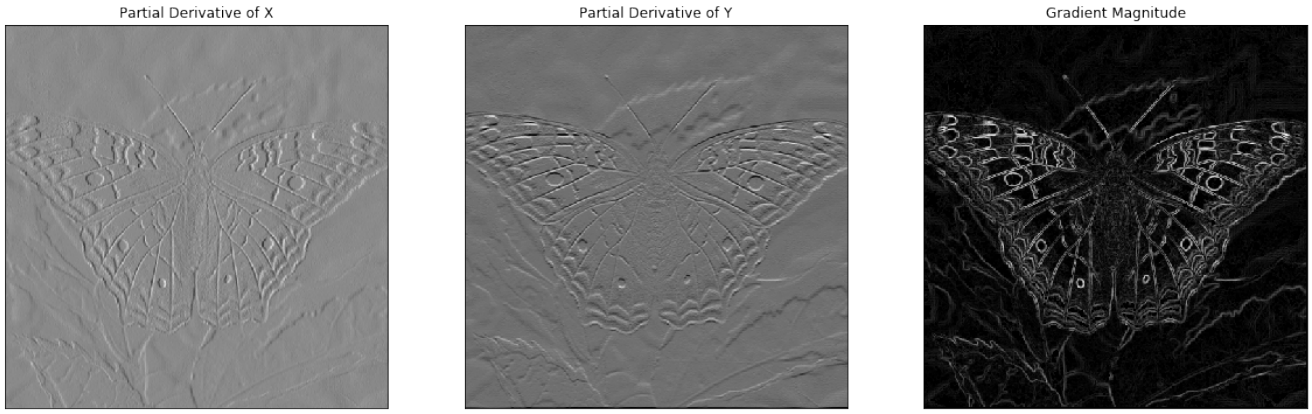

Figure 3. Derivadas parciais nas direções x, y, e magnitude do gradiente calculados com os filtros de Sobel.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I_{x,y} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * ([-1 \quad 0 \quad 1] * I_{x,y})$$

Figure 4. Propriedade de separabilidade linear do filtro de Sobel.

- Multiplicar os resultados obtidos acima $f \times h$ (Simples multiplicação).
- Aplicar a transformada inversa de Fourier para o dominio espacial.

A Figure 5 ilustra esse conceito.

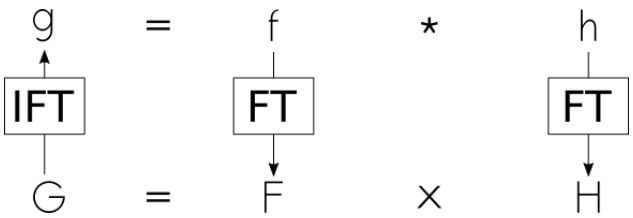

Figure 5. Relação entre a operação de convolução no domínio espacial e de frequência.

Curiosamente, quando aplicamos a Transformada de Fourier (TF) em um kernel Gaussiano, obtemos como resultado outro kernel Gaussiano. No entanto, eles possuem uma caracteristica inversa. Em outras palavras, dado um kernel gaussiano largo (grande sigma) sua TF é uma gaussiana com com um sigma pequeno (gaussiana fina). Analogamente, dado um kernel gaussiano com um sigma pequeno, sua TF resulta em uma gaussiana com um sigma largo. A figura 6 mostra essa relação inversa da gaussiana nos domínios espaciais e de frequência.

A Figura 7 apresenta os resultados da aplicação do filtro gaussiano em uma imagem no domínio de frequência. A

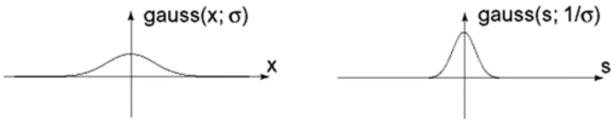

Figure 6. Resultado da Transformada de Fourier em um kernel Gaussiano.

primeira linha mostra como isso é feito no domínio espacial (via convolução), enquanto a segunda linha da figura mostra a mesma operação no domínio de frequência (simples multiplicação).

A Figura 7 mostra alguns pontos interessantes. Perceba como uma gaussiana com um sigma pequeno, equivale a outra gaussiana com sigma bem maior no domínio de frequência.

Para criar o kernel gaussiano foi utilizado a função *getGaussianKernel()* do OpenCV. Exploramos os efeitos de vários valores de sigma, por motivos de espaço, apresentamos na Figura 7 um kernel Gaussiano com sigma 3.

Ao centralizar o espectro da Transformada de Fourier, temos baixa frequência no centro e alta frequência nos arredores. Ao multiplicarmos as duas TF, percebemos na Figura 7(f) como o kernel gaussiano mantém as baixas frequências e atenua as altas frequências. Como resultado, obtemos uma imagem suavizada Figura 7(c). Optamos por utilizar as rotinas *fft.fft2()*, *fft.fftshift()* e *fft.ifft2()* da biblioteca NumPy para as aplicações descritas acima.

5. Conclusions and Future Work

Os resultados do trabalho foram satisfatórios ao que se refere a aplicação de kernels de diversas características nas imagens definidas pelo professor. Investigamos os efeitos de vários limiares (como o valor de sigma para construção das Gaussianas). Exploramos operações de convolução

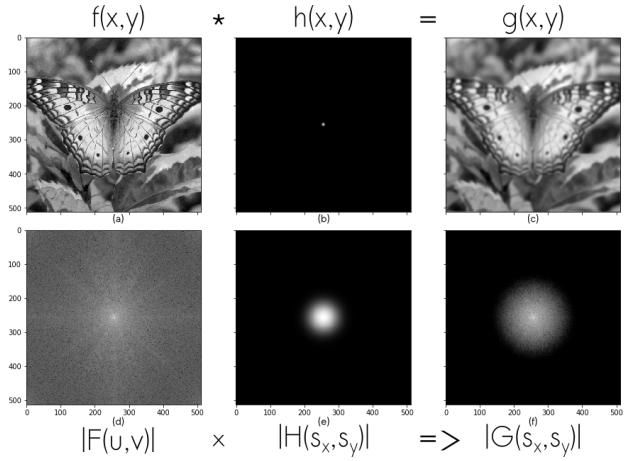

Figure 7. Convolução no domínio de frequência com kernel Gaussiano.

tanto no domínio espacial quanto no de frequência, e enfatizamos as ligações e particularidades entre as duas representações.

References

[1] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016. 1

[2] Oliphant Travis E. A guide to numpy, 2006. [Online; accessed 15/04/2018]. 1

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 1

[4] John D. Hunter. Matplotlib: A 2d graphics environment, 2007. [Online; accessed 15/04/2018]. 1