Universidade Estadual de Campinas

Instituto de Computação

Introdução ao Processamento Digital de Imagem (MC920 / MO443)

Trabalho 1

Leodécio Braz - 230219

April 17, 2019

1 Especificação do Problema

O objetivo deste trabalho é implementar alguns filtros de imagens no domínio espacial e de frequências. A filtragem aplicada a uma imagem digital é uma operação local que altera os valores de intensidade dos pixels da imagem levando-se em conta tanto o valor do pixel em questão quanto valores de pixels vizinhos.

No processo de filtragem, utiliza-se uma operação de convolução de uma máscara pela imagem. Este processo equivale a percorrer toda a imagem alterando seus valores conforme os pesos da máscara e as intensidades da imagem.

2 Script

O script foi desenvolvido em Python na versão 3.0 e foi utilizado o framework Jupyter para documentação do mesmo. As bibliotecas utilizadas foram **OpenCV** [5], **Numpy**[3], **matplotlib**[2], **scipy**[4], **skimage**[1] e a **time**.

2.1 Entrada

As imagens de entrada foram disponibilizadas pelo professor no endereço que se encontra em sua página[1]

As imagens estão localizadas no diretório **./images/**, e já se encontram declaradas no código.

2.2 Saída

Após a execução, o script irá gerar tanto resultados intermediários visíveis na tela, quanto as imagens resultante dos processos que serão salvas no diretório **./output/**.

[1] $http://www.ic.unicamp.br/\ helio/imagens_png/$

As imagens de saída estão no formato PNG (*Portable Network Graphics*) e seus nomes seguem o padrão como na Tabela 1.

Tabela 1: Nome dos arquivos de saída

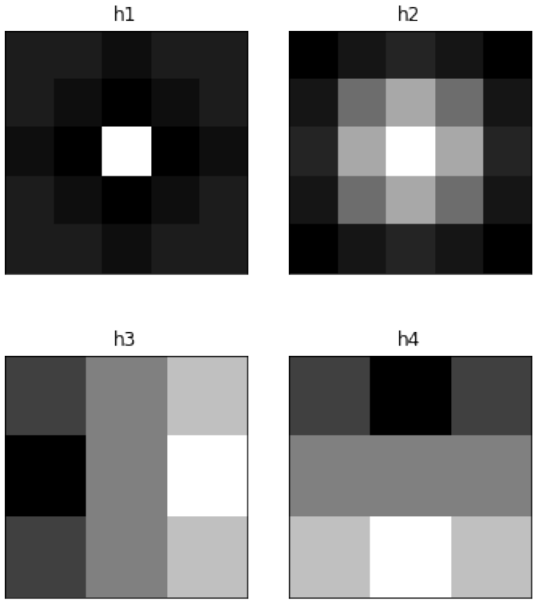
Nome do arquivo da imagem	**Descrição**
[imagem_entrada]_h1	Imagem de entrada após a aplicação do filtro h1 (problema 1.1)
[imagem_entrada]_h2	Imagem de entrada após a aplicação do filtro h2 (problema 1.1)
[imagem_entrada]_grau_suavização_[sigma]	Imagem de entrada após a aplicação de filtro Gaussiano com determinado valor de sigma (problema 1.2)

2.3 Leitura das imagens e Filtros

As imagens de entrada foram lidas de forma monocromática através da função **imread** da biblioteca do OpenCV, que gera um *Array* do Numpy [3] de dimensões MxN que corresponde ao tamanho da imagem.

Os filtros, também no formato de *Array* do Numpy, foram declarados manualmente no código. A figura 1 ilustra uma representação visual dos mesmos.

Figura 1: Representação visual dos filtros h1 a h4



3 Resolução

3.1 Filtragem no Domínio Espacial

3.1.1 Convolução no plano discreto

A grosso modo, a operação de convolução de um filtro e uma imagem no domínio espacial se dá através de operações de soma e multiplicação entre o filtro e a imagem.

Na convolução, aplica-se uma rotação de 180° no filtro, e sua matriz se torna uma espécie de 'janela deslizante' que 'desliza' ao longo da matriz da imagem, multiplicando filtro e imagem elemento por elemento e ao final somando estes resultados.

Como o modelo de coordenadas é diferente, os filtros não são centrados na posição (0,0) mas sim na posição que representa exatamente o seu centro. Assim, se um filtro possui dimensão de MxN, ele estará centrado na posição ($\lfloor \frac{M}{2} \rfloor$, $\lfloor \frac{N}{2} \rfloor$), por exemplo, um filtro de tamanho 3x3 estará centrado na posição (1,1).

Por este motivo foi necessário adicionar bordas (*padding*) ao redor da imagem, para que, ao 'deslizar' o filtro na imagem as informações da imagem ficassem alinhadas com as do filtro.

Para o processo de convolução, realizamos uma sequencia de etapas para chegar ao objetivo final. Implementamos uma função onde a mesma recebe dois parâmetros, uma imagem e um filtro. Realizamos então uma rotação de 180° no filtro e em seguida adicionamos as bordas na imagem, o tamanho das bordas são definido por $\lfloor \frac{x}{2} \rfloor$ e $\lfloor \frac{y}{2} \rfloor$, onde x e y são as dimensões do filtro.

Após estas etapas, percorremos a altura e largura da imagem 'deslizando' o filtro sobre a mesma e realizando as multiplicações elemento por elemento. Ao final, a matriz da nova imagem naquela posição, recebe a soma dessas multiplicações e a 'janela' do filtro desliza novamente. Esse processo é feito até que o filtro percorra toda a imagem.

3.1.2 Resultados da aplicação dos filtros na imagem

Após a aplicação dos filtros na imagem, é notório os efeitos causados e algumas considerações podem ser feitas:

1. **Filtro h1**

   Filtro Passa-Alta, ele atenua o valor central ao percorrer o filtro na imagem.

2. **Filtro h2**

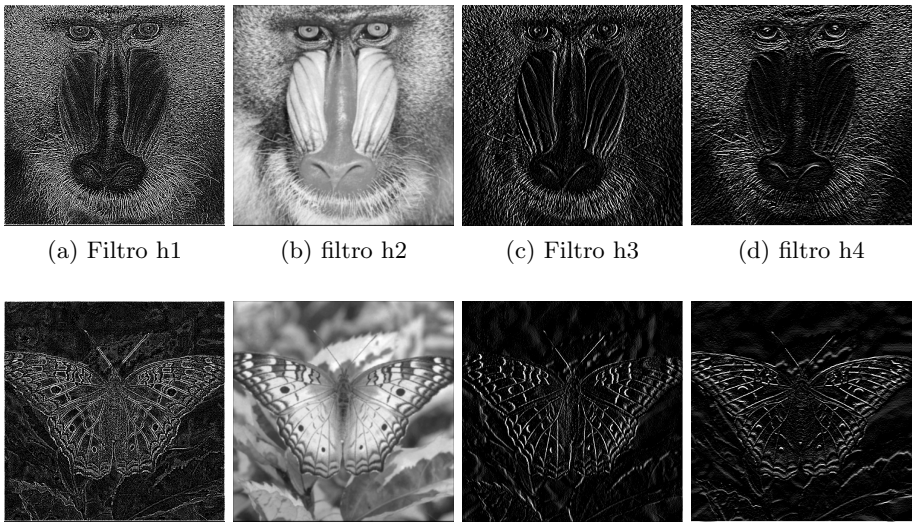
   Filtro Passa-Baixa, um filtro da média. Ele produz um borramento leve na imagem.

3. **Filtros h3 e h4**

Conhecidos como filtro de Sobel, eles atenuam as bordas da vertical e da horizontal da imagem.

A Figura 2 ilustra os efeitos da aplicação destes filtros nas imagens.

Figura 2: Aplicação dos filtros na imagem



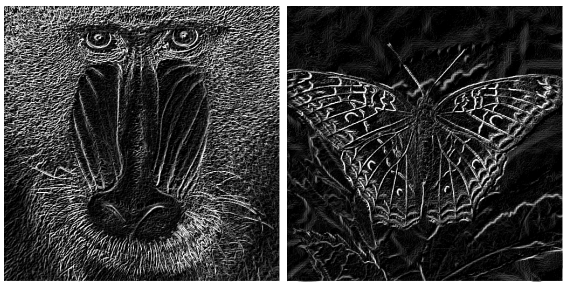
(a) Filtro h1 (b) filtro h2 (c) Filtro h3 (d) filtro h4

(e) Filtro h1 (f) Filtro h2 (g) Filtro h3 (h) Filtro h4

Também foram analisadas as imagens resultantes através da combinação dos filtros $h3$ e $h4$ por meio da fórmula: $\sqrt{(h3)^2 + (h4)^2}$. Como a fórmula já sugere, é realizado um *merge* entre as imagens que foram aplicadas a h3 e a h4 atenuando as bordas da imagem em ambas a direções. O resultado dessa operação para as mesmas imagens acima podem ser visualizados na Figura 3.

Figura 3: Aplicação fórmula $\sqrt{(h3)^2 + (h4)^2}$

(a) Baboon (b) Butterfly



3.2 Filtragem no domínio de frequência

O processamento de imagens no domínio de frequência é comumente realizado através de três passo:

1. É preciso primeiramente alterar o domínio da imagem, isto é a imagem é transformada do plano espacial para o plano de frequência (ou, do plano discreto para o plano continuo) utilizando a transformada de Fourier.

2. Realiza as operações no plano de frequência.

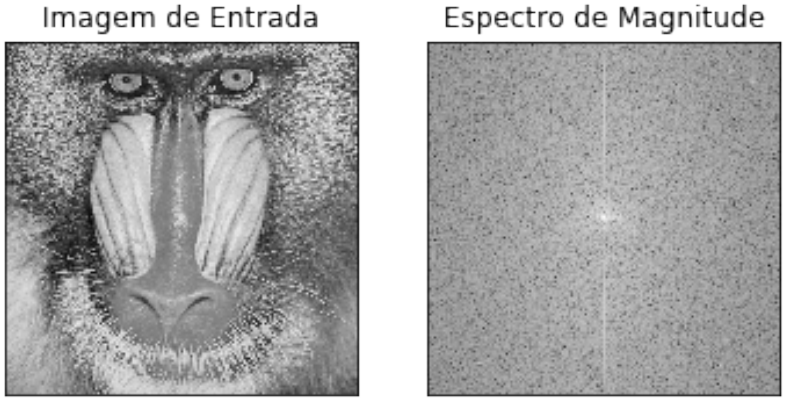
3. Realiza novamente uma transformação na imagem, dessa vez do domínio de frequência para o domínio espacial utilizando a transformada inversa de Fourier

Para realizar a transformada de Fourier da imagem e gerar seu espectro de Fourier, utilizamos as bibliotecas OpenCV e Numpy.

Através do**OpenCV.dft** geramos o espectro de Fourier, em seguida por meio do **Numpy.fft.fftshift** transladamos o espectro para que a sua componente de frequência esteja no centro do espectro.

Após essas operação podemos calcular o espectro de magnitude da imagem, como mostra a Figura 4.

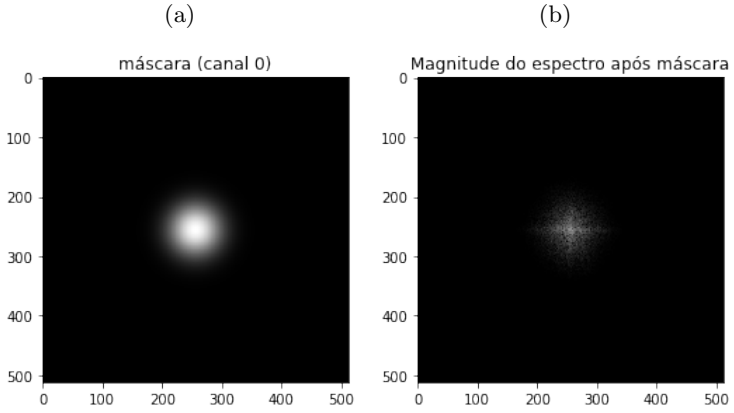
Figura 4: Imagem de entrada e seu espectro de magnitude



Em seguida, geramos um filtro gaussiano com as dimensões da imagem original e com um grau de suavização σ que recebemos como parâmetro. A partir do filtro, geramos uma máscara com 2 canais de profundidade e replicamos o filtro em ambos canais. A máscara possui dois canais pois ela será aplicada no espectro de Fourier que foi gerado. A Figura 5a ilustra uma representação visual de um filtro gaussiano gerado com $\sigma = 30$, no canal 0 da máscara.

Aplicamos então esta máscara no espectro, multiplicando $máscara * espectro$. Podemos visualizar assim (Figura 5b) a magnitude do espectro após essa operação.

Figura 5: Máscara e espectro de magnitude



Após essas operações no domínio de frequência, 'convertemos' novamente a imagem, agora do domínio de frequência para o domínio espacial através da transformada inversa de Fourier. Para isso utilizamos a função **OpenCv.idft** para realizar essa transformação. Esta função recebe como parâmetro um espectro no domínio de frequência e retorna a imagem equivalente no domínio espacial.

3.2.1 Resultados da aplicação dos filtros na imagem

Testamos os resultados gerados por estas operações para filtros com diferentes valores de de grau de suavização σ.

Podemos notar que quanto maior o grau de suavização, mais nítida fica a imagem, e quanto menor este grau, maior o borramento que produzido na mesma.

A Figura 6 mostra as saídas para estes diferentes valores de σ.

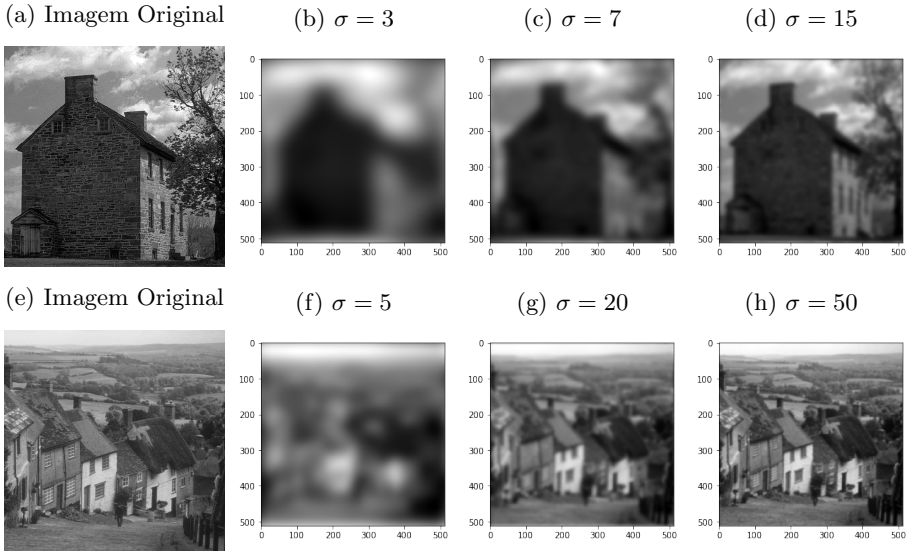
4 Resultados

Na Figura 7 podemos visualizar o efeito da aplicação de um filtro gaussiano em uma mesma imagem com as diferentes abordagens citadas. Definimos um grau de suavização 40 para o filtro gaussiano, e executamos a convolução no domínio espacial e de frequência. Podemos notar que os resultados são bem semelhantes.

Além dos resultados das imagens geradas que foram mostrados nas seções anteriores, analisamos o tempo de execução para as diferentes abordagens propostas.

Submetemos ambos os algorítimos sobre filtros gaussianos com mesmos valores de σ e para as mesmas imagens, com imagens de diferentes tamanhos.

Figura 6: Aplicação dos filtros na imagem



Definimos um valor de $\sigma = 30$ para gerar os filtros e analisamos os tempos para imagens de dimensões $512x512$, $1000x1000$, $1500x1500$ e $2000x2000$. Os resultados dos tempos podem ser vistos na Tabela 2.

Tabela 2: Tempo em segundos do processo de convolução nos diferentes domínios

	Tamanho das imagens			
	512x512	1000x1000	1500x1500	2000x2000
Convolução no domínio espacial	2.16s	7.70s	18.15s	32.26s
Convolução no domínio de frequência	0.05s	0.08s	0.19s	0.25s

References

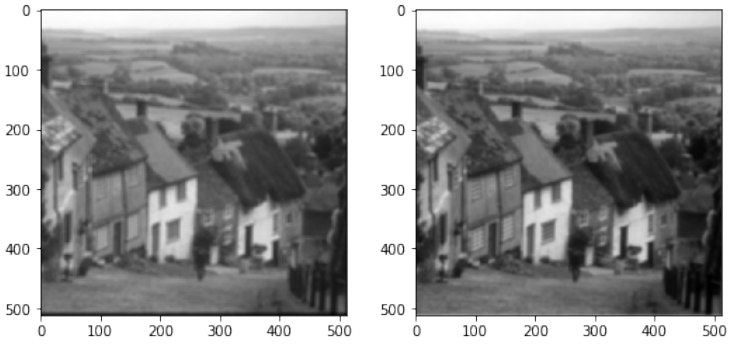
[1] Image processing scikit. https://scikit-image.org/docs/stable/overview.html. Accesso em: 16/04/2019.

[2] Matplotlib user guide. https://matplotlib.org/contents.html. Accesso em: 14/04/2019.

[3] Numpy user guide. https://docs.scipy.org/doc/numpy/user/. Accesso em: 14/04/2019.

Figura 7: Resultado da convolução nos diferentes domínios

(a) Domínio espacial (b) Domínio de frequência



[4] Scipy tutorial. https://docs.scipy.org/doc/scipy/reference/tutorial/index.html. Accesso em: 16/04/2019.

[5] Welcome to opencv documentation! https://docs.opencv.org/2.4/index.html. Accesso em: 14/04/2019.