MC920 – Introdução ao Processamento Digital de Imagem
Trabalho 1

Rafael Eiki Matheus Imamura - RA 176127[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (Unicamp)
Caixa Postal 6176 – CEP 13083-970 – Campinas – SP – Brasil

ra176127@students.ic.unicamp.br

1. Introdução

No processamento digital de imagem, uma operação muito comum é a aplicação de filtros. Os filtros auxiliam na visualização de características específicas nas imagens. São operações locais, que utilizam os valores de cada pixel e seus vizinhos, multiplicando-os por valores específicos e os somando. Neste trabalho, foram analisados dois tipos de filtragem: no domínio espacial e no de frequência.

Para cada um dos casos, foram usadas máscaras e analisados seu significado e efeitos nas imagens de testes. As máscaras foram fornecidas no enunciado do trabalho. Para este trabalho, foram usadas 6 imagens PNG em escala de cinza, com intensidade nos valores de 0 a 255.

2. Execução

A entrega deste trabalho inclui o presente PDF e um diretório de nome "projeto1". Nela se encontram o código, as dependências e as imagens de entrada usadas para os testes.

Para executar o código, deve-se ter instalado o OpenCV e o Numpy. Um freeze das dependências está presente no arquivo "requirements.txt". Uma alternativa rápida para fazer a instalação é o comando:

pip install -r requirements.txt

Com este comando instalado, basta chamar o arquivo de script com o parâmetro desejado:

python main.py operação arquivo_de_entrada opção arquivo_de_saida

O significado de cada parâmetro da entrada é:

1. *Operação*: define qual função será chamada para realizar a filtragem da imagem. As opções são:
    a. *h1, h2, h3* ou *h4*: realiza a operação com o filtro correspondente;

b. *h3_h4*: realiza a operação h3 e h4 na imagem de forma separada e junta elas com a operação $\sqrt{(h3)^2 + (h4)^2}$ nas imagens resultantes;

c. *g*: aplica o filtro gaussiano no domínio da frequência;

d. *all*: gera imagens com todos os filtros. São geradas versões diferentes de imagens com Gaussiano variando o desvio padrão de 5 em 5 até 50;

e. *example*: gera todas as imagens para os casos de exemplo (diretório "pictures"). Equivale a aplicar a opção *all* em todas as imagens de exemplo;

2. *Arquivo_de_entrada*: caminho para a imagem a ser filtrada. Este parâmetro não é usado para a opção *example*.
3. *Opção*: um parâmetro extra usado para definir o tipo de imagem gerado. Na operação *g*, define o desvio padrão do filtro gaussiano (grau de suavização). Nas operações *h*, define se a imagem resultante deve ser binária (preto e branco) caso seja passado "binary" como parâmetro ou na escala de cinza caso passe "-".
4. *Arquivo_de_saida*: caminho para o arquivo de saída da imagem. Este parâmetro é usado para as operações h1, h2, h3, h4, h3_h4 e g. Caso a opção seja *all*, esse parâmetro define o diretório de saída das imagens. A opção *example* não usa este parâmetro – as imagens são geradas no diretório "results".

Alguns exemplos de uso são mostrados a seguir:

1. python main.py example
2. python main.py all ../pictures/baboon.png -
3. python main.py g ../pictures/baboon.png 15 ./teste.png
4. python main.py h2 ../pictures/baboon.png binary ./teste.png

2.1. Entrada

A entrada, em todos os casos, é dada por imagens PNG em escala de cinza (de intensidade entre 0 e 255). Para este relatório, foram usadas 6 imagens, 5 das quais fornecidas no enunciado do trabalho (*http://www.ic.unicamp.br/~helio/imagens_png/*) e uma imagem que é um desenho (*poney.png*).

Todas as entradas usadas estão disponíveis dentro do zip, no diretório "pictures".

2.2. Saída

A saída do programa são imagens com as mesmas características da entrada (PNG, escala de cinza). Caso a opção "binary" seja passada para os filtros h, a imagem resultante será em preto e branco (0 ou 255). Para as opções *all* e *example* do programa, as imagens geradas tem como nome o nome original da imagem, adicionada de um sufixo indicando a operação e possíveis parâmetros.

Por exemplo, as imagens com filtros h1, terão o nome "imagem_h1.png". As imagens geradas com o filtro Gaussiano com um sigma de 10, terão o nome "imagem_g_10.png".

3. Solução

A solução criada aplica os filtros em cada imagem usando o Numpy e o OpenCV. Para os filtros h1 a h4, a matriz é usada para filtrar a imagem, aplicando-se sobre cada pixel. Para o filtro que combina o h3 e o h4, foi aplicado individualmente os filtros na imagem, então calculado o quadrado de cada matriz, somadas e extraído a raiz. Após a aplicação dos filtros, em todos os casos, foi normalizado a imagem.

Em todos os casos, a imagem é aberta e transformada em um vetor de float32 usando o **np.float32**. Isso foi feito para que as operações não tivessem problemas de arredondamento, especialmente na Gaussiana e na combinação de h3 com h4.

Os filtros foram criados passando uma matriz como parâmetro para um **np.array** (SciPy.org, 2019) e usando o **cv2.filter2d** para fazer a filtragem efetivamente (OpenCV, 2015).

Para o filtro Gaussiano em uma imagem representada pelo seu espectro de Fourier, primeiro a imagem foi transformada por uso de FFT (*Fast Fourier Transform*), usando o **np.fft.fft2 (**Mordvintsev, 2013). Depois, foi dado um shift na faixa-zero, transladando-a para o centro da imagem (**np.fft.fftshift**). Foi então gerado um filtro Gaussiano que multiplicou a matriz resultante, destransladado o espectro (**np.fft.ifftshift**) e transformado de volta para a imagem (**np.fft.ifft2**). Neste ponto, a matriz continha números imaginários, então antes de normalizar a imagem, foram extraídos os valores absolutos de cada pixel.

4. Resultados

A seguir, é possível notar a diferença entra algumas imagens após serem aplicados os filtros. O teste executado foi com a operação “example” do programa. As análises a seguir são da imagem “baboon.png”.

Imagem original Filtro h1 Filtro h2

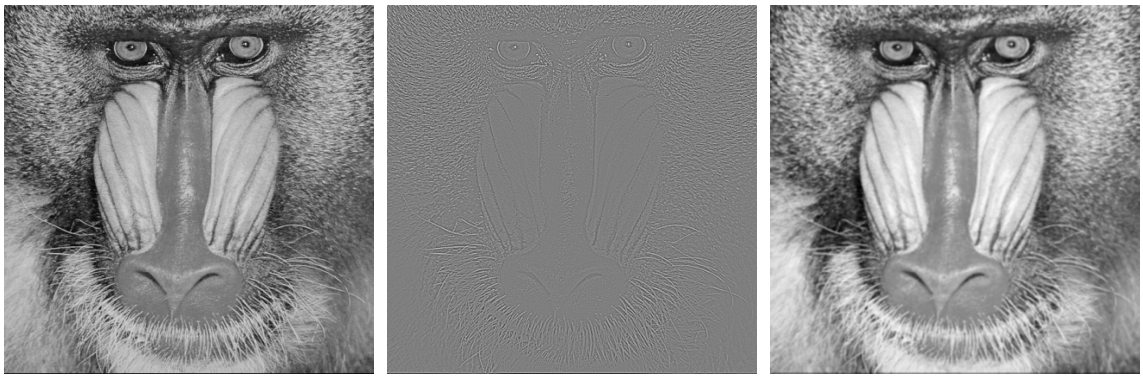


Figura 1. Comparação da imagem original “baboon.png” com os filtros h1 e h2.

A Figura 1 mostra a comparação da imagem original com o filtro h1 e o h2. O filtro h1 é um filtro passa-alta e o filtro h2 é um filtro passa-baixa, o que pode ser notado tanto pelas suas matrizes quanto pelo resultado na imagem. O h1 realça o contorno, enquanto o h2 suaviza a imagem.

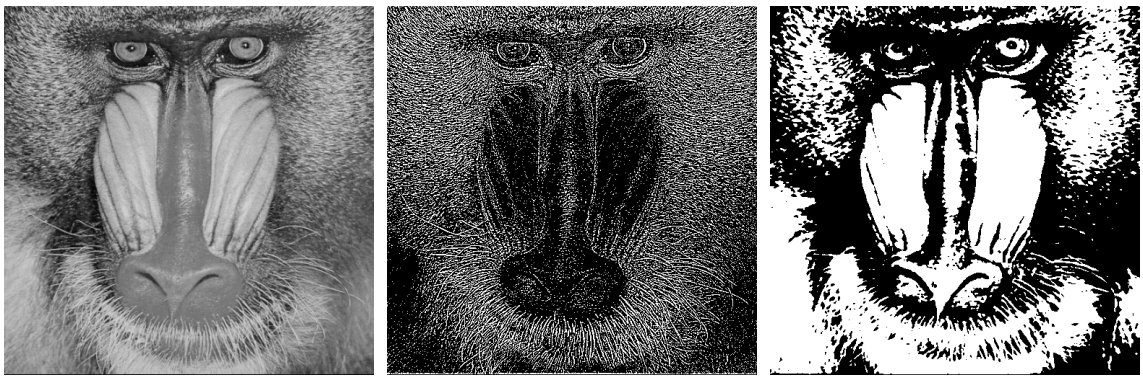
Filtro h3 Filtro h4 Filtro h3 e h4 combinados

Figura 2. Comparação da imagem "baboon.png" com os filtros h3 e h4 individualmente e combinados.

Os filtros h3 e h4, individuais e combinados, realçam o contorno da imagem (Figura 2). O h3 realça os traços verticais, o h4 os traços horizontais, e a combinação mostra o contorno geral. Todos eles são filtros passa-alta.

Após comparar essas imagens, foi possível perceber como as imagens com os filtros h1, h3, h4 (individuais e combinados) podem ser difíceis de se visualizar por mostrarem a diferença de contorno. Assim, foi criada a opção de deixar a imagem em preto e branco. O *threshhold* usado é o valor 128, um valor no meio entre 0 e 255. Assim, as imagens transformadas em binárias são mostradas na Figura 3.

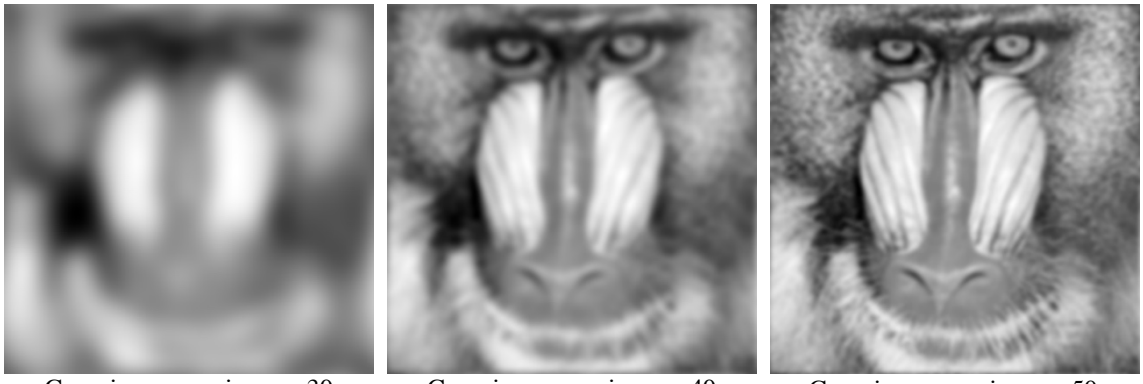
Imagem original Filtro h1 (binária) Filtro h2 (binária)



Filtro h3 (binária) Filtro h4 (binária) Filtro h3 e h4 (binária)

Figura 3. Comparação da imagem "baboon.png" com os filtros h em binário.

Gaussiano com sigma = 5 Gaussiano com sigma = 15 Gaussiano com sigma = 25



Gaussiano com sigma = 30 Gaussiano com sigma = 40 Gaussiano com sigma = 50

Figura 4. Comparação dos resultados do filtro Gaussiano com diferentes graus de suavização na imagem "baboon.png".

O filtro Gaussiano é um filtro passa-baixa, suavizando a imagem (Figura 4). Conforme o desvio padrão (sigma) aumenta, a imagem vai ficando mais nítida. Para valores menores do desvio, especialmente até 25, a diferença visual é bem grande. O grau de suavização é afetado diretamente pelo desvio padrão, que define o tamanho do filtro.

A seguir, são mostrados alguns outros exemplos e discutidos suas diferenças – imagens "city.png" (Figuras 5, 6 e 7) e "poney.png" (Figuras 8, 9 e 10).

Imagem original Filtro h1 (binária) Filtro h2

Figura 5. Comparação da imagem "city.png" com os filtros h1 e h2.

Filtro h3 (binária) Filtro h4 (binária) Filtro h3 e h4 (binária)

Figura 6. Comparação da imagem “city.png” com os filtros h3 e h4, individual e combinado.

Gaussiano com sigma = 5 Gaussiano com sigma = 20 Gaussiano com sigma = 35

Figura 7. Comparação da imagem “city.png” com o filtro Gaussiano com diferentes desvios padrão.

No caso da imagem “city.png”, o filtro h1 ficou com bastante ruído, assim como no “baboon.png” e o filtro h2 borrou levemente os contornos (Figura 5). Os filtros h3 e h4 exibem os contornos de h1 com menos ruído, mas perde um pouco de informação (Figura 6). O filtro gaussiano com sigma = 5 deixa a imagem totalmente irreconhecível devido ao nível de detalhe (Figura 7). Com o desvio padrão valendo 35, ainda se têm uma imagem mais borrada que h2.

Imagem original Filtro h1 (binária) Filtro h2

Figura 8. Comparação da imagem “poney.png” com os filtros h1 e h2.

Filtro h3 (binária) Filtro h4 (binária) Filtro h3 e h4 (binária)

Figura 9. Comparação da imagem "poney.png" com os filtros h3 e h4, individual e combinado.

Gaussiano com sigma = 5 Gaussiano com sigma = 25 Gaussiano com sigma = 45

Figura 10. Comparação da imagem "poney.png" com o filtro Gaussiano com diferentes desvios padrão.

A imagem "poney.png" foi testada por ser um desenho linear simples, com menos detalhes. Com o filtro h1, aqui, ele pega os detalhes da personagem muito bem, sem ruídos. O filtro h2 nivela um pouco os tons de cinza da imagem e as bordas ainda continuam nítidas (Figura 8). Os filtros h3 e h4 não conseguem pegar bem o contorno da figura, perdendo detalhes como a boca e o nariz da personagem (Figura 9). Os filtros Gaussianos borram a imagem, mas a falta de detalhes faz com que a diferença não seja tão grande como nos outros casos (Figura 10).

5. Conclusões

Neste trabalho, foram implementados diferentes filtros que manipulam imagens do formato PNG em escala de cinza. A observação dos resultados dos filtros nas imagens mostra as diferentes propriedades que eles apresentam em diferentes níveis de detalhes. Para o caso em que o contorno era importante, transformar a imagem em binária ajudou a deixar os resultados mais visíveis.

O filtro h1, enquanto realça contornos, para figuras com muitos detalhes traz muitos ruídos. Para uma figura simples, ele conseguiu detectar bem esses contornos. O filtro h2 teve uma diferença visual relativamente pequena, borrando bordas em imagens com detalhes e uniformizando a cor no caso da figura mais simples. O filtro h3 realçou

traços verticais de borda, enquanto o h4 pegou os horizontais. A combinação dos dois juntava esses traços, mas curvas que tinham propriedades tanto verticais quanto horizontais foram perdidas. O h3 e h4 juntos se mostraram o melhor filtro de realce de contorno para imagens com mais detalhes. Os filtros Gaussianos, como filtros passa-baixa, borraram a imagem, tendo resultados mais suaves com valores maiores de desvio padrão.

O programa não trata imagens coloridas e foram usadas poucas imagens preto e branco. Com uma quantidade maior de entradas, pode ser que determinadas observações ficassem mais claras. O projeto, porém, cumpre o escopo proposto na especificação do problema.

Referências

Mordvintsev, A. (2013) "Fourier Transform". OpenCV-Python Tutorials. Disponível em: <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_transforms/py_fourier_transform/py_fourier_transform.html>. Consulta em abr. 2019.

OpenCV. (2015) "OpenCV Documentation". Disponível em: <https://docs.opencv.org/>. Acesso em: abr. 2019.

SciPy.org. (2019) "NumPy Reference". Disponível em: < https://docs.scipy.org/doc/numpy/reference/>. Acesso em: abr. 2019.