Atividade 1 - Filtragem

Daniel Tatsch
Mestrado em Computação Aplicada - Visão Computacional
Univali - Universidade do Vale do Itajaí
Setembro de 2020

1 Introdução

Este trabalho tem como objetivo demonstrar o processo de filtragem de imagens utilizando implementações disponíveis na biblioteca OpenCV da linguagem de programação Python. Foram analisados os filtros *Non-Local Means* (NLM), filtro bilateral (BF) e filtro de mediana.

Métricas como a relação sinal ruído de pico (PSNR, *Peak Signal Noise to Ratio*), erro médio quadrático (MSE, *Mean Squared Error*) e a própria percepção visual humana foram analisadas para validar o filtro que melhor removesse o ruído e preservasse os contornos da imagem.

O arquivo com o código desenvolvido pode ser acessado através da plataforma Github.

2 Implementação

Foi selecionada uma imagem colorida em formato *JPG*, com resolução de 241 x 280 pixels. A partir dela, foi extraída a sua representação em tons de cinza e aplicado ruído do tipo Gaussiano (AWGN) com média igual a 0 e desvio padrão unitário. A Figura 1 apresenta a imagem original utilizada no experimento, sua versão em tons de cinza e a soma desta com o ruído AWGN.

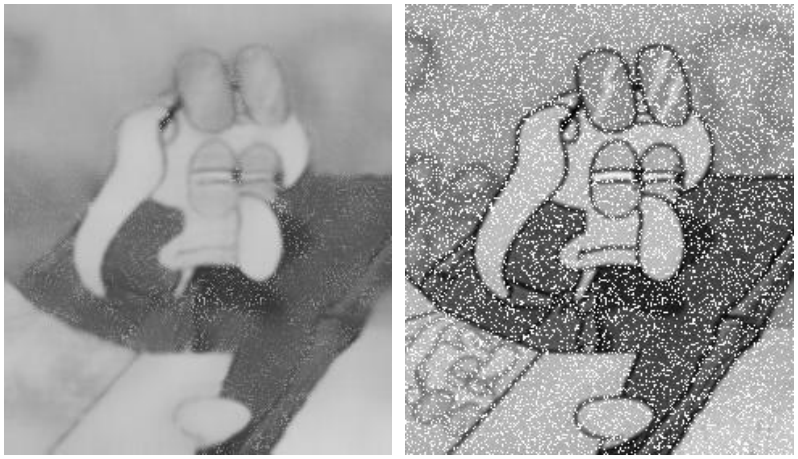
(a) (b) (c)

Figura 1 - (a) Imagem original (b) Tons de cinza (c) Imagem corrompida com AWGN.

2.1 Filtro NLM

Para utilizar o filtro NLM disponível na biblioteca OpenCV do Python foi utilizada a função *cv2.fastNlMeansDenoising*. Ela recebe como parâmetro um fator h de intensidade do filtro que é aplicado sobre a imagem corrompida. Um h com valor muito grande pode remover completamente o ruído aditivo, porém tende a suavizar ou remover detalhes como bordas e contornos. Ao selecionar um valor muito pequeno do fator h, as bordas são preservadas, porém o ruído não é completamente eliminado.

Um laço de execução foi realizado a fim de se verificar qual o valor do fator h resultava no maior PSNR e no menor erro quadrático médio. Verificou-se que executando 60 vezes o filtro, o maior PSNR encontrado foi de aproximadamente $18,6dB$, com $h = 52$. No entanto, o menor valor de MSE, $17,31$, foi obitdo com o $h = 9$, apesar de remover uma quantidade significativamente menor que no valor encontrado no PSNR. A Figura 2 apresenta as duas imagens resultantes da filtragem com os dois valores de h relatados.



(a) (b)

Figura 2 - Resposta do filtro NLM. (a) h = 52 (b) h = 9.

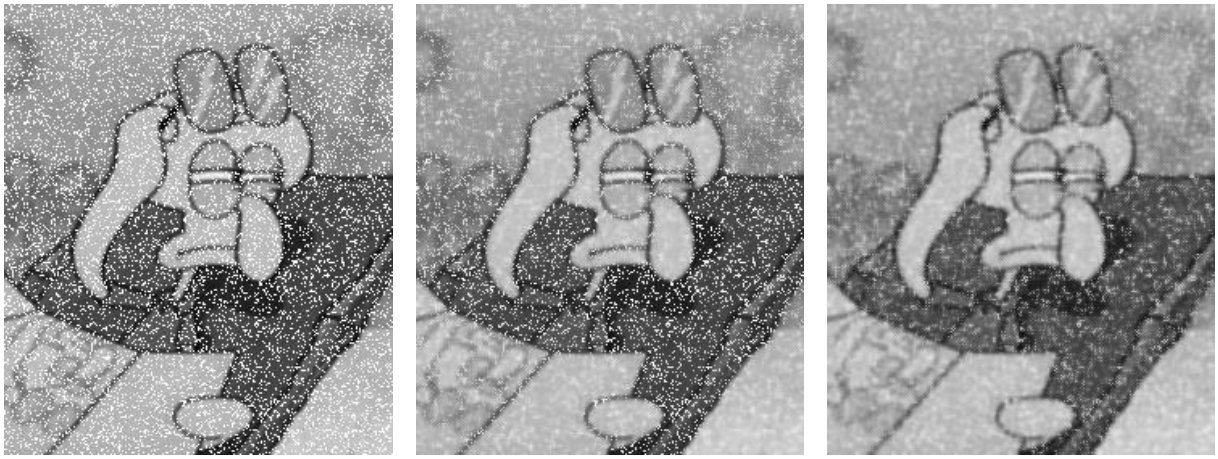
2.2 Filtro bilateral

A biblioteca OpenCV possui uma função específica para execução do filtro bilateral, a *cv2.bilateralFilter*. Assim como no caso do filtro NLM, é necessário passar mais parâmetros além da imagem em tons de cinza. Neste caso, é necessário informar o diâmetro d (em quantidade de pixels) da vizinhança que será utilizada para calcular a média ponderada sobre o pixel de interesse. Essa ponderação também pode ser controlada através dos parâmetros $SigmaColor$ e $SigmaSpace$. O primeiro, quando muito alto, faz com que cores mais distantes dentro da vizinhança sejam misturadas, resultando em áreas com cores mais homogêneas. O $SigmaSpace$ por sua vez, incide sobre a influença que os pixels mais distantes terão sobre eles mesmos, desde que possuam cores próximas.

Nas referências da biblioteca, é indicado que os dois parâmetros de $Sigma$ tenham o mesmo valor. Se estes tiverem valores muito pequenos (< 10), o filtro não terá o efeito desejado. No entanto, com valores muito grandes (>150), a imagem filtrada tende a ficar com efeito "cartoonizado". Em relação ao diâmetro d, é dito que para valores superiores a 5, o filtro tende a ser mais lento, não indicado para aplicações em tempo real. Além disso, valores muito grandes acabam ofuscando a imagem, suavizando ou eliminando os contornos.

A Figura 3 mostra os resultados obtidos com o filtro bilateral, utilizando o diâmetro da vizinhança $d = 5$ e variando os parâmetros de $Sigma$ entre os valores pequenos,

intermediários e grandes especificados na biblioteca. A Tabela 1 apresenta a relação do PSNR e MSE de acordo com a variação dos parâmetros $Sigma$.



(a) (b) (c)

Figura 3 - Resposta do filtro bilateral (a) Sigma = 10 (b) Sigma = 75 (c) Sigma = 150.

Tabela 1 - SNR e MSE do filtro bilateral

Sigma	PSNR (dB)	MSE
10	13,48	21,04
75	15,94	57,61
150	19,9	89,81

2.3 Filtro de mediana

O filtro de mediana atua sobre o cálculo da mediana dos pixels presentes no *kernel*, substituindo o pixel central pelo valor computado. A função *cv2.medianBlur* pode ser utilizada para obter uma imagem filtrada, de acordo com o tamanho do *kernel* passado como parâmetro (deve ser passado um valor ímpar). A resposta do filtro com a variação do tamanho do *kernel* utilizado (3, 5 e 7) pode ser observada na Figura 4.

(a) (b) (c)

Figura 4 - Resposta do filtro de mediana (a) *kernel* = 3 (b) *kernel* = 5 (c) *kernel* = 7.

Pode-se perceber que com um *kernel* de tamanho 3 (a), o filtro de mediana consegue manter muito bem os contornos da imagem, porém fragmentos do ruído ainda foram mantidos. Com filtragem utilizando um *kernel* de tamanho 5 (b), praticamente

todo ruído foi removido, suavizando minimamente os contornos. A partir do $kernel = 7$, praticamente não há ganho no que diz respeito à remoção do ruído e além disso, os contornos das imagens são cada vez mais ofuscados.

Essas informações também podem ser validadas através da Tabela 2, onde estão dispostos os valores de PSNR e MSE de cada imagem filtrada. Nota-se uma melhoria no PSNR com a utilização de um *kernel* de tamanho 5, em contraste com a piora desta métrica no caso último caso.

Tabela 2 - SNR e MSE do filtro bilateral

kernel	**PSNR (dB)**	**MSE**
3	24,98	18,59
5	26,25	28,33
7	23,53	33,21

3 Conclusão

Este trabalho teve como objetivo analisar diferentes processos de filtragem a fim de comparar os resultados obtidos de acordo com a variação de seus parâmetros. Para tal, além da análise visual humana, foram utilizadas métricas objetivas como PSNR e MSE.

Assim como aprendido em aula, notou-se que pelo ruído possuir uma variação grande em relação aos pixels das imagens, a sua filtragem implica diretamente em outros componentes de frequência elevada, como bordas, texturas e contornos. Esse comportamento pôde ser observado na utilização do filtro NLM com fator $h = 52$, no qual foi obtido um valor alto de PSNR, porém praticamente todos os contornos da imagem ficaram comprometidos.

O filtro que obteve tanto objetivamente quanto subjetivamente os melhores resultados foi o filtro de mediana com $kernel = 5$. Observou-se um leve efeito de borrão, porém sem grandes perdas de generalidade. Além de ter resultado em um PSNR menor que no caso anterior, a implementação com *kernel* de tamanho 7 teve o pior erro médio quadrático dos filtros de mediana implementados, além de resultar em uma suavização excessiva dos contornos da imagem filtrada.

4 Referências

GOYAL, B. et al. Image denoising review: From classical to state-of-the-art approaches.Inf. Fusion, v. 55, p. 220–244, 2020.

OPENCV. Python Open Source Computer Vision. [S.l.]. Disponível em: <https://docs.opencv.org/master/index.html>. Acesso em: 13 set. 2020.