Universidade Tecnológica Federal do Paraná - UTFPR
Campus Curitiba
Departamento Acadêmico de Eletrônica - DAELN
Curso de Engenharia Eletrônica
Prof. Gustavo B. Borba e Prof. Humberto R. Gamba

Notas de aula e prática MATLAB #05

Arquivos necessários [disponíveis no zip ou MATLAB built-in]

1. Lenna256g.png [adaptada de https://en.wikipedia.org/wiki/Lenna]
2. boat.512.tiff [disponível em http://sipi.usc.edu/database/]
3. b1s.100.bmp, b5s.100.bmp e b9s.100.bmp [Imagens geradas em BrainWeb: Simulated Brain Database, https://brainweb.bic.mni.mcgill.ca/brainweb]
4. carro1mm.jpg
5. cameraman.tif [MATLAB built-in]

5a) Separabilidade dos filtros lineares [[BB], Tópico 6.3.2]

Os filtros lineares recebem este nome porque podem ser implementados a partir da operação da convolução (a convolução é linear, só pra reforçar). O filtro Gaussino e o box filter (média), por exemplo, são filtros lineares. O filtro da mediana, por exemplo, é não-linear. Relembrando algumas propriedades da convolução, no contexto de imagens:

- Linearidade: mutiplicar a imagem por um escalar constante s e depois aplicar a convolução, dá o mesmo resultado que aplicar a convolução e depois multiplicar por s. Ainda, somar duas imagens e depois aplicar a convolução, dá o mesmo resultado que aplicar a convolução em cada imagem individualmente e somá-las.

$$(s \cdot I) * H = I * (s \cdot H) = s \cdot (I * H)$$

$$(I_1 + I_2) * H = (I_1 * H) + (I_2 * H)$$

- Comutatividade: convoluir o filtro com a imagem ou a imagem com o filtro (!) dá o mesmo resultado.

$$I * H = H * I$$

- Associatividade: sucessivos filtros lineares podem ser aplicados em qualquer ordem.

$$A * (B * C) = (A * B) * C$$

Agora, vamos considerar que podemos expressar o kernel de convolução desejado através da convolução de diferentes kernels. Um kernel que pode ser expresso dessa forma é chamado de ***separável*** *(separable)*:

$$H = H_1 * H_2 * \cdots * H_n$$

Então, a convolução de uma imagem com esse kernel é

$$I * H = I * (H_1 * H_2 * \cdots * H_n)$$

Mas como a convolução é associativa, podemos escrever:

$$I * H = (\cdots((I * H_1) * H_2) * \cdots H_n)$$

Para ilustrar a coisa toda e compreender a implicação prática disso, vamos usar como exemplo um box filter 3x3. O kernel desse filtro é separável e pode ser escrito como mostrado a seguir (se quiser testar, faça `Hx = [1/3 1/3 1/3]; Hy = [1/3; 1/3; 1/3]; Hxy = conv2(Hx,Hy);`).

$$\underbrace{\begin{bmatrix}\frac{1}{3} & \frac{1}{3} & \frac{1}{3}\end{bmatrix}}_{H_x} * \underbrace{\begin{bmatrix}\frac{1}{3}\\ \frac{1}{3}\\ \frac{1}{3}\end{bmatrix}}_{H_y} = \underbrace{\begin{bmatrix}\frac{1}{9} & \frac{1}{9} & \frac{1}{9}\\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9}\\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9}\end{bmatrix}}_{H_{xy}}$$

Isso significa que, para obter a imagem filtrada F, ao invés de convoluir Hxy com a imagem de entrada I, podemos fazer:

$$F = (I * H_x) * H_y$$

Isto é, convoluir separadamente: convoluir primeiramente I com Hx e, o resultado disso, convoluir com Hy.

Na prática: a convolução de Hxy com a imagem de entrada requer 9 (3x3=9) multiplicações para cada pixel. Já a convolução com Hx e depois com Hy requer 6 (3+3=6) multiplicações para cada pixel. Então, no final das contas, a separabilidade permite a implementação mais eficiente de filtros lineares. Quanto maiores forem as dimensões do kernel, maior é este ganho.

"In general, the number of operations for a 2D filter grows quadratically with the filter size (side length) but only linearly if the filter is x/y-separable. Clearly, separability is an eminent bonus for the implementation of large linear filters" [[BB] Tópico 6.3.2, página 102].

IMPORTANTE

- O filtro Gaussiano é separável. Em [[BB] Tópico 6.3.2] há uma análise detalhada.
- Para implementar um box filter também há as *integral images* (introduzidas por [VJ]). Em [KD] há uma boa revisão sobre as *integral images*. Na Wikipedia, está em [IIw].

5.1) Filtro separável

Elabore um script para comparar o desempenho do box filter 5x5 com a sua versão separável. Para capturar o tempo de execução no MATLAB, utilize as funções `tic` e `toc`. Não utilize `imfilter`, nem `conv`, nem `conv2` nem... faça tudo na unha, com `for` dentro de `for` à vontade. Assim, é possível garantir que você estará comparando laranjas com laranjas. **Resposta disponível em box_filter_separavel.m.**

Esse exercício é interessante também para você reforçar o procedimento de uma convolução/correlação 2D e visualizar como é implementada em um nível mais baixo. Para te ajudar, aí vai um código em C, mas fique à vontade pra implementar como achar melhor (desde que a comparação dos desempenhos ainda seja válida. Você pode reaproveitar código do exercício que pedia para implementar o filtro da mediana na unha, de aulas anteriores). Só por curiosidade, em [JT] há n+1 maneiras de se implementar convolução/correlação 2D.

[MW], Tópico 8.2, página 99:
"A C program segment to compute a simple discrete convolution is given next. It convolves a 3x3 mask with an MxM image to give a MxM convolved result. This convolution ignores a single pixel edge around the image.

```
for (i=1; i<M-2; ++i)
    for(j=1; j<M-2; ++j)
        Result[i][j] = Image[i-1][j-1] * mask[0][0] +
                       Image[i-1][j]   * mask[0][1] +
                       Image[i-1][j+1] * mask[0][2] +
                       Image[i][j-1]   * mask[1][0] +
                       Image[i][j]     * mask[1][1] +
                       Image[i][j+1]   * mask[1][2] +
                       Image[i+1][j-1] * mask[2][0] +
                       Image[i+1][j]   * mask[2][1] +
                       Image[i+1][j+1] * mask[2][2];   "
```

5b) Edge-preserving filters

Conforme vimos na aula anterior e observamos na prática, através das atividades, ao suavizar o ruído de uma imagem acaba-se também suavizando as bordas, gerando o efeito indesejado de blurring e perda dos detalhes finos da imagem. Em imagens médicas, por exemplo, esta deterioração das bordas pode ser até mais prejudicial que em imagens convencionais, já que detalhes importantes das estruturas anatômicas ou de lesões podem ser corrompidos.

O objetivo de um *edge-preserving filter* (também chamado de **filtro adaptativo**) é remover o máximo possível do ruído da imagem, comprometendo o mínimo possível a integridade das bordas.

5.2) MMSE filter (edge-preserving filter)

Um *edge-preserving filter* não-linear simples para a suavização do ruído Gaussiano é o MMSE (Minimum Mean Square Error) [[SV] Equação 4; [MW] Tópico 11.4 Equação 26; [GD] Tópico 8.3.1 Equação 8.14; [GW] Tópico 5.3.3 Equação 5.3-12 (o que aqui chamamos de MMSE não recebe nenhum nome em GW)]. Cada pixel de saída do MMSE é obtido aplicando-se a equação a seguir para cada pixel correspondente da imagem de entrada. O MMSE também opera sobre uma janela quadrada, que define a vizinhança de pixels da imagem de entrada a serem utilizados no cálculo da saída.

$$\hat{f}(i,j) = \left(1 - \frac{\sigma_n^2}{\sigma_l^2}\right) r(i,j) + \frac{\sigma_n^2}{\sigma_l^2} m_l$$

Onde $r(i,j)$ é a imagem a ser filtrada (com ruído), σ_l^2 é a variância dos pixels pertencentes à janela e m_l a média dos pixels pertencentes à janela. σ_n^2 é a variância do ruído, que será uma constante. A variância do ruído (σ_n^2) deve ser estimada antes e utilizada como um parâmetro do filtro, da seguinte forma: obter a variância de pixels pertencentes a uma região escura (sem sinal, portanto apenas ruído) e homogênea (sem bordas) da imagem.

As imagens a seguir mostram a imagem cameraman original, com ruído, e processada com o MMSE.

Interpretando a equação, explique o princípio de funcionamento do MMSE. Dica: suponha as situações nas quais (i) a variância da janela é igual à variância estimada do ruído e (ii) a variância da janela é alta em relação à variância estimada do ruído. Lembre que: a principal finalidade do MMSE é suavizar o ruído e preservar as bordas e que a variância da janela é alta quando a ela está posicionada sobre uma borda. **Resposta disponível nos slides.**

5.3) MMSE filter (edge-preserving filter)

Crie um script que implementa o MMSE filter. Lembando: os parâmetros de entrada do filtro são a matriz imagem a ser filtrada, as dimensões da janela, sempre quadrada, e a variância estimada do ruído (uma constante). Utilize a função `imcrop` para permitir a interação do usuário para selecionar a região da imagem que será utilizada para o cálculo da variância do ruído. Mostrar os resultados (imagem de entrada e imagem de saída). Compare com a saída do filtro da média (box filter) de janela de mesmas dimensões para verificar a afirmação de que este filtro é melhor em preservar as bordas e se ele suaviza o ruído tanto quanto o filtro da média. **Resposta disponível em mmse_filter.m.**

5.4) Bilateral filter (edge-preserving filter)

O ***bilateral filter*** é um edge-preserving filter bastante utilizado. O kernel w de um filtro bilateral é composto pela multiplicação ponto-a-ponto de dois kernels, denominados *spatial kernel* s (também chamado de *domain kernel* d) e *range kernel* r. s é uma função Gaussiana 2D convencional com σ_s. r é uma função (Gaussiana com σ_r) que, ao invés de considerar a distância do centro para as bordas do kernel, considera a similaridade (subtração) entre o nível de cinza do pixel que está sendo filtrado (pixel central) e os demais pixels pertencentes ao kernel. "Pixels which are similar to the current center pixel contribute strongly, while highly different pixels add little to the result […] the bilateral filter is a filter with a convolution kernel that is adaptively controlled by the local image content" [BB3] Capítulo 17 - Edge-Preserving Smoothing Filters, página 497.

As figuras a seguir ilustram o conceito.

[PKTD] Figura 2.2:

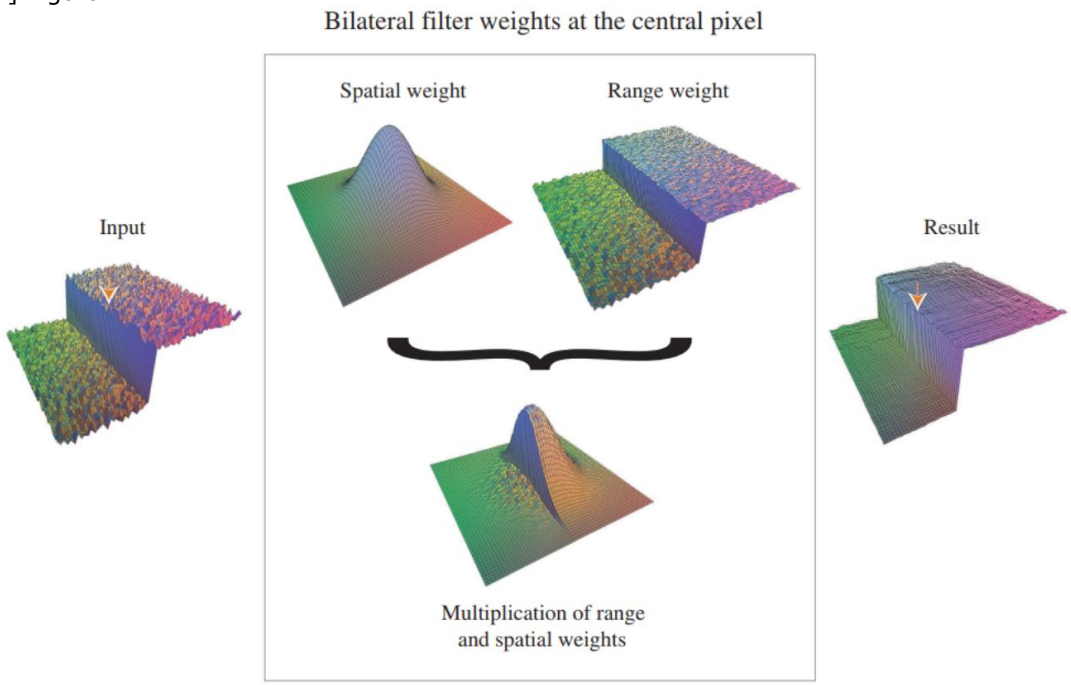

Fig. 2.2 The bilateral filter smooths an input image while preserving its edges. Each pixel is replaced by a weighted average of its neighbors. Each neighbor is weighted by a spatial component that penalizes distant pixels and range component that penalizes pixels with a different intensity. The combination of both components ensures that only nearby similar pixels contribute to the final result. The weights shown apply to the central pixel (under the arrow). The figure is reproduced from [21].

[SB] Figura 5.17, Tópico 5.5.3:

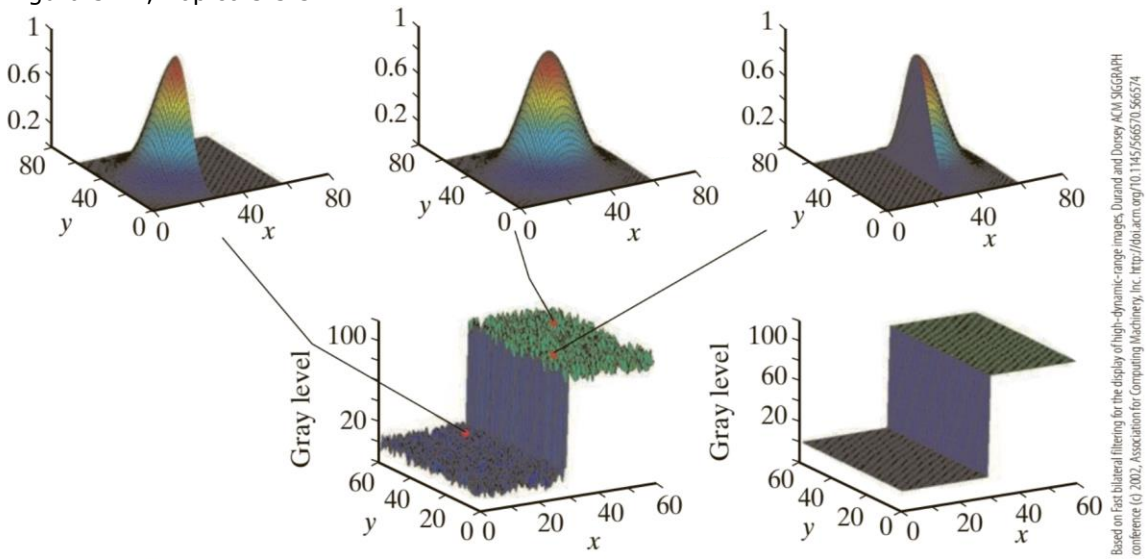

Figure 5.17 Bilateral filtering of a noisy step edge preserves the crisp edge as it smooths out the noise on either side of the edge. The top row shows the kernel at three locations: Far from the edge, the kernel approximates a Gaussian, whereas near the edge it approximates half a Gaussian.

"The bilateral filter is controlled by two parameters: σ_s and σ_r. Figure bellow illustrates their effect. As the range parameter σ_r increases, the bilateral filter gradually approximates Gaussian convolution more closely because the range Gaussian $G\sigma_r$ widens and flattens, i.e., is nearly constant over the intensity interval of the image. Increasing the spatial parameter σ_s smooths larger features. In practice, in the context of denoising, Liu et al. [41] show that adapting the range parameter σ_r to estimates of the local noise level yields more satisfying results. The authors recommend a linear dependence: $\sigma_r = 1.95\ \sigma_n$, where σ_n is the local noise level estimate" [PKTD].

[PKTD] Figura 2.3:

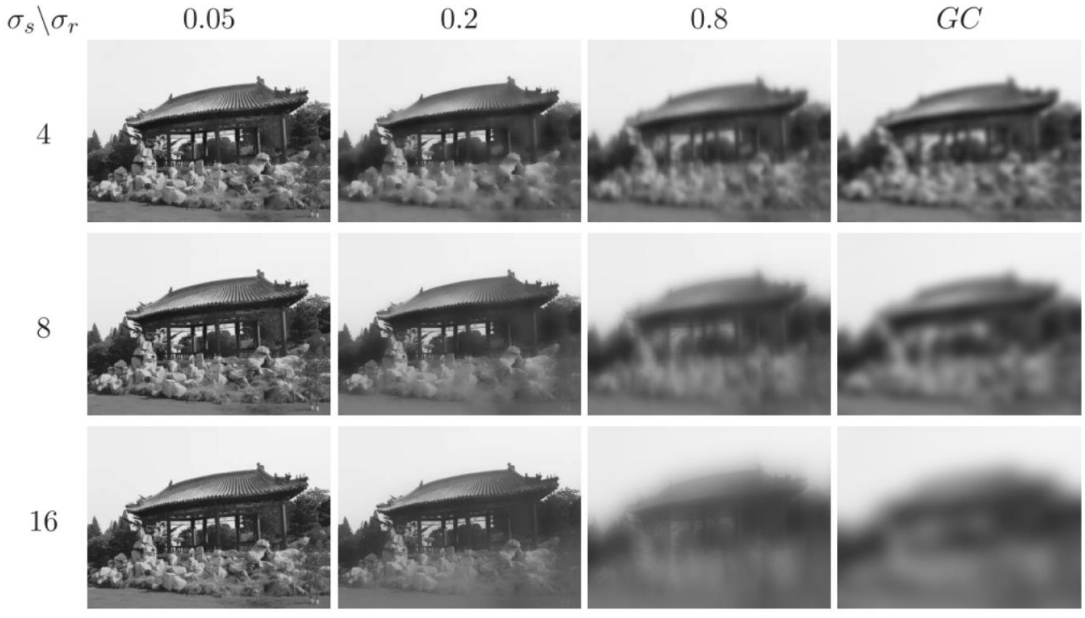

Fig. 2.3 The bilateral filter's range and spatial parameters provide more versatile control than Gaussian convolution. As soon as either of the bilateral filter weights reaches values near zero, no smoothing occurs. As a consequence, increasing the spatial sigma will not blur an edge as long as the range sigma is smaller than the edge amplitude. For example, note the rooftop contours are sharp for small and moderate range settings σ_r, and that sharpness is independent of the spatial setting σ_s. The original image intensity values span $[0, 1]$.

Com base no exemplo em https://www.mathworks.com/help/images/ref/imbilatfilt.html, crie um script para testar o bilateral filter (função `imbilatfilt`). **Resposta disponível em bilateral_imbilatfilt.m**

5.5) Bilateral filter (edge-preserving filter)

Crie um ou mais scripts para implementar o bilateral filter e que permita a visualização do *domain kernel* e do *range kernel* para pixels especificados pelo usuário. **Resposta disponível em bilateral_filter_test_point.m e bilateral_filter_image.m**

5c) Image quality metrics (IQM)

Quando comparamos imagens visualmente e dizemos, por exemplo, "o ruído na imagem A foi mais suavizado que na imagem B" ou "a imagem C está menos borrada e portanto melhor que a imagem D", estamos fazendo uma avaliação *qualitativa*. Estas avaliações são importantes, mas pouco relevantes cientificamente, pois não podem ser reproduzidas por outros pesquisadores para a comparação de diferentes algoritmos. É possível coletar a opinião de uma quantidade significativa de pessoas a respeito da qualidade da imagem, com base em um sistema de pontuação pré-definido. Este é um método subjetivo de avaliação de qualidade de imagem, denominado *mean opinion score* (MOS). Um MOS é uma métrica válida, pois é *quantitativa*, mas difícil de ser obtida, já que envolve voluntários, isto é, observadores humanos. Assim, são necessárias métricas *quantitativas objetivas* para a medida do desempenho dos algoritmos de filtragem. As métricas utilizadas para isto são denominadas de *métricas objetivas para a medida de qualidade de imagem* [WBL]. Atualmente, não existe uma métrica objetiva universalmente aceita para a medida de qualidade de imagem. Uma das métricas objetivas do tipo *full-reference* (definição a seguir) mais adotadas é o índice *structural similarity (SSIM)* [WBo], disponível no MATLAB a partir da versão 2014a, função **`ssim`** da IPT (http://www.mathworks.com/help/images/ref/ssim.html). Também há outras implementações por aí, como a dos próprios autores, por exemplo: https://ece.uwaterloo.ca/~z70wang/research/ssim/#usage.

As métricas objetivas podem ser classificadas ainda em função da necessidade ou não da imagem original (sem ruído, não degradada) para o seu cálculo. Se uma métrica requer a imagem original ela é do tipo *full-reference*. Se não é necessária a imagem original, a métrica é do tipo *no-reference*, também chamada de *blind-metric*.

Uma métrica objetiva bastante utilizada é a *relação sinal-ruído* (SNR, signal-to-noise ratio). Genericamente, SNR = sinal/ruído. Quanto maior for a SNR, melhor, já que maior é o sinal em relação ao ruído presente.

- [WB] Tópico 2.6 Equação 2.5:

$$SNR_L = \frac{m_s}{\sigma_n}$$

A SNR_L é uma métrica do tipo *no-reference*, utiliza apenas a imagem que se deseja avaliar. m_s é a média do sinal, estimada considerando-se os pixels de uma região clara (sinal) e homogênea (sem bordas) da imagem. σ_n é o desvio padrão do ruído, estimado considerando-se os pixels de uma região escura (sem sinal, portanto, apenas ruído) e homogênea da imagem. Quanto maior o valor de SNR_L, melhor.

- [ASP] Equação 10:

$$PSNR = 20log\frac{L}{\sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{(\hat{f}(i,j)-f(i,j))^2}{MN}}}[dB]$$

A *PSNR* (o *P* significa *Peak*) é uma métrica do tipo *full-reference*. L é a faixa disponível dos pixels (255 para imagens de 8 bits por pixel), M e N são o número linhas e colunas da imagem, $f\hat{}(i,j)$ os pixels da imagem que ser quer avaliar e $f(i,j)$ os pixels da imagem original (sem ruído). Quanto maior o valor de PSNR, melhor. Disponível no MATLAB, função **`psnr`** (https://www.mathworks.com/help/images/ref/psnr.html).

- [GW] Tópico 5.8 Equação 5.8-4:

$$MSE = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}(f(i,j) - \hat{f}(i,j))^2}{MN}$$

A *mean squared error* (MSE) é do tipo *full-reference*. M e N são o número linhas e colunas da imagem, $f\hat{}(i,j)$ os pixels da imagem que ser quer avaliar e $f(i,j)$ os pixels da imagem original (sem ruído). Quanto menor o valor de MSE, melhor. Disponível no MATLAB, função **`immse`** (https://www.mathworks.com/help/images/ref/immse.html).

5.6) IQM SNR_L

Implemente a métrica SNR_L. Utilize a função `imcrop` para permitir a interação do usuário para selecionar as regiões da imagem correspondente ao sinal (m_s) e ao ruído (σ_n). Obtenha os valores da SNR_L para as imagens *b1s.100.bmp*, *b5s.100.bmp* e *b9s.100.bmp*. Use as mesmas regiões para as medidas nas três imagens. **Resposta disponível em snrl_iqm.m.**

Referências

[BB] Wilhelm Burger, Mark Burge, Digital image processing – an algorithmic introduction using Java, Springer, 2010.

[GW] Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins, Digital image processing, Pearson Prentice Hall, 3rd ed, 2008.

[VJ] Paul Viola, Michael Jones, Rapid object detection using a boosted cascade of simple Features, Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - CVPR 2001, vol. 1, p. I:511–518, 2001.

[KD] Konstantinos G. Derpanis, Notes in Computer vision, Integral image-based representations, Department of Computer Science and Engineering, York University, 2007. Disponível em http://www.cse.yorku.ca/~kosta/CompVis_Notes/integral_representations.pdf

[IIw] Wikipedia, Summed Area Table http://en.wikipedia.org/wiki/Summed_area_table.

[WB] Wolfgang Birkfellner, Applied medical image processing: a basic course, CRC Press, 2011.

[ASP] C. B. Ahn, Y. C. Song, D. J. Park, Adaptive Template Filtering for Signal-to-Noise Ratio Enhancement in Magnetic Resonance Imaging, IEEE Transactions on Medical Imaging, vol. 18, no. 6, p. 549-556, June 1999.

[WBL] Zhou Wang, Alan C. Bovik, Ligang Lu, Why is image quality assessment so difficult?, IEEE International conference on Acoustics, Speech & Signal Processing, May 2002. Disponível em https://ece.uwaterloo.ca/~z70wang/publications/icassp02a.html

[WBo] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, Eero P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Transactions on Image Processing, vol. 13, no. 4, p. 600-612, 2004. Disponível em https://ece.uwaterloo.ca/~z70wang/publications/ssim.pdf

[GD] Geoff Dougherty, Digital image processing for medical applications, Cambridge University Press, 2009. Imagens disponíveis em www.cambridge.org/dougherty

[SV] Sun, X.Z., Venetsanopoulos, A.N., Adaptive schemes for noise filtering and edge detection by use of local statistics, IEEE Transactions on Circuits and Systems, vol. 35, n. 1, p. 57-69, DOI 10.1109/31.1700, 1998.

[MW] Harley R. Myler, Arthur R. Weeks, Computer imaging recipes in C, Prentice-Hall, 1993.

[JT] Jan Thorbecke, 2D Convolution Algorithms, 2000. Disponível em http://janth.home.xs4all.nl/Publications/html/conv2d.html

[SB] Stan Birchfield, Image Processing and Analysis, Cengage Learning, 2018.

[PKTD] S. Paris, P. Kornprobst, J. Tumblin and F. Durand, Bilateral Filtering: Theory and Applications, Foundations and Trends in Computer Graphics and Vision, Vol. 4, No. 1 (2008) 1–73. 2009. Disponível em https://www.researchgate.net/publication/220427978_Bilateral_Filtering_Theory_and_Applications

[BB3] Wilhelm Burger and Mark J. Burge, Digital Image Processing: An Algorithmic Introduction, 3a Ed., Springer, 2022. Site do livro: https://imagingbook.com/