

Análise e Aplicação do Filtro de Kuan em Imagens com Ruído Speckle

I. Introdução

O filtro de Kuan [1] está entre as diversas técnicas propostas na literatura para redução de ruído em imagens. O objetivo deste trabalho é implementar o algoritmo proposto no artigo original, desenvolver uma metodologia de medição da eficácia do filtro, e apresentar os resultados obtidos em imagens produzidas artificialmente.

II. Filtro de Kuan

II.a Forma Geral

Kuan é baseado em um modelo estatístico de imagem em que a média e variância são não estacionárias. O modelo de ruído multiplicativo é da forma $g = f + f(n - 1)$, que na realidade é derivação da forma aditiva $g = f + n$, onde g é o sinal da imagem observada, f é o sinal original e n é o ruído. A partir do modelo estatístico e do ruído multiplicativo, os autores desenvolveram a estimação linear do erro quadrático médio (EQM) mínimo, expressa da seguinte forma geral:

$$\hat{f}(i, j) = \bar{f}(i, j) + \frac{v_f(i, j)}{v_f(i, j) + \sigma_n^2(i, j)}(g(i, j) - \bar{g}(i, j)) \quad (1)$$

O artigo mostra que localmente $\bar{f} = \bar{g}$. As estatísticas locais $\bar{g}(i, j)$, $v_g(i, j)$ e $v_f(i, j)$ de uma janela de tamanho $(2r + 1)(2s + 1)$ são dadas por:

$$\bar{g}(i, j) = \frac{1}{(2r + 1)(2s + 1)} \sum_{p=i-r}^{i+r} \sum_{q=j-s}^{j+s} g(p, q) \quad (2)$$

$$v_g(i, j) = \frac{1}{(2r + 1)(2s + 1)} \sum_{p=i-r}^{i+r} \sum_{q=j-s}^{j+s} (g(p, q) - \bar{g}(i, j))^2 \quad (3)$$

$$v_f(i, j) = v_g(i, j) - \sigma_n^2(i, j) \quad (4)$$

II.b Filtro de Kuan para Ruídos Multiplicativos

A equação geral para o filtro foi mostrada em 1, porém para ruídos multiplicativos a estimação \hat{f} é representada da seguinte forma:

$$\hat{f}(i, j) = \bar{f}(i, j) + \frac{v_f(i, j)}{v_f(i, j) + \frac{\sigma_u^2}{E(u)^2}[(\bar{f}(i, j))^2 + v_f(i, j)]}(g(i, j) - \bar{f}(i, j)) \quad (5)$$

Aqui, u é o ruído multiplicativo cuja média e variância são estacionárias. Com v_f dada por:

$$v_f(i, j) = \frac{\sigma_g^2(i, j) - \frac{\sigma_u^2}{E(u)^2}E(g(i, j))^2}{1 + \frac{\sigma_u^2}{E(u)^2}} \quad (6)$$

Podemos também reescrever a equação 5 da seguinte forma:

$$\hat{f}(i, j) = \bar{f}(i, j) + w(i, j)(g(i, j) - \bar{f}(i, j))$$

$$\Rightarrow \hat{f}(i, j) = w(i, j)g(i, j) + \bar{f}(1 - w(i, j)) \quad (7)$$

Sendo $w(i, j)$ definida por:

$$w(i, j) = \frac{v_f(i, j)}{v_f(i, j) + C_u[(\bar{f}(i, j))^2 + v_f(i, j)]} \quad (8)$$

C_u é o coeficiente de variação do ruído, que aqui assumimos ser estacionário, e portanto pode ser facilmente estimado com o cálculo da seguinte equação em uma região homogênea da imagem:

$$C_u^2 = \left(\frac{\sigma_u}{E(u)}\right)^2 \quad (9)$$

Sendo assim, podemos reescrever 6 substituindo $\left(\frac{\sigma_u}{E(u)}\right)^2$ por C_u^2 :

$$\begin{aligned} v_f(i, j) &= \frac{\sigma_g^2(i, j) - C_u^2 E(g(i, j))^2}{1 + C_u^2} \\ &= \frac{\sigma_g^2(i, j) - C_u^2 (\bar{g}(i, j))^2}{1 + C_u^2} \end{aligned} \quad (10)$$

Substituindo 10 em 8 e assumindo $\bar{f} = \bar{g}$ teremos:

$$\begin{aligned} w(i, j) &= \frac{v_f(i, j)}{v_f(i, j) + C_u^2[(\bar{f}(i, j))^2 + v_f(i, j)]} = \frac{v_f(i, j)}{v_f(i, j) + C_u^2(\bar{g}(i, j))^2 + C_u^2 v_f(i, j)} \\ &= \left[\frac{\sigma_g^2(i, j) - C_u^2 (\bar{g}(i, j))^2}{1 + C_u^2} \right] \cdot \left[\frac{1}{\frac{\sigma_g^2(i, j) - C_u^2 (\bar{g}(i, j))^2}{1 + C_u^2} (1 + C_u^2) + C_u^2 (\bar{g}(i, j))^2} \right] \\ &= \left[\frac{\sigma_g^2(i, j) - C_u^2 (\bar{g}(i, j))^2}{1 + C_u^2} \right] \cdot \left[\frac{1}{\sigma_g^2(i, j) - C_u^2 (\bar{g}(i, j))^2 + C_u^2 (\bar{g}(i, j))^2} \right] \\ &= \frac{\sigma_g^2(i, j) - C_u^2 (\bar{g}(i, j))^2}{(1 + C_u^2) \sigma_g^2(i, j)} = \frac{\sigma_g^2(i, j)}{(1 + C_u^2) \sigma_g^2(i, j)} - \frac{C_u^2 (\bar{g}(i, j))^2}{(1 + C_u^2) \sigma_g^2(i, j)} \\ &= \frac{1}{1 + C_u^2} - \frac{C_u^2 \left(\frac{\bar{g}(i, j)}{\sigma_g}\right)^2}{1 + C_u^2} \\ &= \frac{1 - C_u^2 \left(\frac{\bar{g}(i, j)}{\sigma_g}\right)^2}{1 + C_u^2} \end{aligned} \quad (11)$$

Fazendo $C_i^2(i, j) = \frac{\sigma_x(i, j)}{\bar{g}(i, j)}$ e substituindo em 11, temos:

$$w(i, j) = \frac{1 - \frac{C_u^2}{C_i^2}}{1 + C_u^2} \quad (12)$$

III. Implementação e Testes

III.a Implementação

Usamos Python e algumas bibliotecas científicas como IPython [4], SciPy [5] e NumPy [6] em todos os algoritmos e testes.

Para o filtro de Kuan, foram implementadas as equações 7 e 13. O algoritmo para cálculo de $w(i, j)$ é apresentado na listagem abaixo:

Listagem 1: método *weight* de kuan.py

```

1 def weight(window, cu):
2     ci = variation(window)
3     ci_2 = ci * ci
4     cu_2 = cu * cu
5
6     if cu_2 > ci_2:
7         w = 0.0
8     else:
9         w = (1.0 - (cu_2 / ci_2)) / (1.0 + cu_2)
10
11     return w

```

E a listagem a seguir mostra o algoritmo do filtro de Kuan:

Listagem 2: método *filter* de kuan.py

```

1 def filter(img, window_size, cu):
2     img = np.float64(img)
3     img_filtered = np.zeros_like(img)
4
5     N, M = img.shape
6     win_offset = window_size / 2
7
8     for i in xrange(0, N):
9         xleft = i - win_offset
10        xright = i + win_offset
11        # (...)
12
13        for j in xrange(0, M):
14            yup = j - win_offset
15            ydown = j + win_offset
16            # (...)
17
18            window = img[xleft:xright, yup:ydown]
19            w = weight(window, cu)
20            img_filtered[i, j] = round((img[i, j] * w) + (window.mean() * (1.0 - w)))
21
22    return img_filtered

```

Algumas partes do código foram omitidas para manter a clareza. O código pode ser consultado na íntegra em <http://github.com/igortopcin/ptc5892>.

III.b Medição da Eficácia

Como Kuan é um filtro adaptativo, espera-se que ele se comporte como um filtro de média nas regiões homogêneas da imagem, e um filtro identidade nas bordas. Por esse motivo foram usados dois critérios para a medição da eficácia do filtro implementado. Em cada teste, quantificamos a preservação da média e a redução do desvio padrão em regiões homogêneas. Além desse critério, também quantificamos a preservação das bordas da imagem usando Pratt's *Figure of Merit* (*FOM*) [2], também usado por Yu *et al.* para medir e comparar o desempenho do filtro SRAD PDE [3].

A equação que define o *FOM* é dada por:

$$FOM = \frac{1}{\max\{\hat{N}, N_{ideal}\}} \sum_{i=1}^{\hat{N}} \frac{1}{1 + d_i^2 \alpha} \quad (13)$$

Onde \hat{N} é o número de *pixels* de borda detectados, N_{ideal} é o número de *pixels* de borda do nosso *padrão ouro*, d_i é a distância euclidiana entre o i -ésimo *pixel* de borda detectado e o mais próximo *pixel* de borda do *padrão ouro*.

Assim como Yu, usamos o detector de bordas Canny [7] em nossa implementação do *FOM*, mostrado na listagem 3. Para evitar que Canny suavize demasiadamente a imagem, usamos desvio padrão $\sigma = 0,1$.

Na prática, o valor do *FOM* pode variar de 0 a 1, sendo 1 o valor para o *padrão ouro*.

Listagem 3: fom.py

```

1 def fom(img, img_gold_std, alpha = DEFAULT_ALPHA):
2     edges_img = canny(img, 0.1)
3     edges_gold = canny(img_gold_std, 0.1)
4
5     dist = distance_transform_edt(np.invert(edges_gold))
6
7     fom = 1.0 / np.maximum(
8         np.count_nonzero(edges_img),
9         np.count_nonzero(edges_gold))
10
11     N, M = img.shape
12
13     for i in xrange(0, N):
14         for j in xrange(0, M):
15             if edges_img[i, j]:
16                 fom += 1.0 / ( 1.0 + dist[i, j] * dist[i, j] * alpha)
17
18     fom /= np.maximum(
19         np.count_nonzero(edges_img),
20         np.count_nonzero(edges_gold))
21
22     return fom

```

III.c Imagens de Teste

Aplicou-se um ruído *speckle* às imagens 1a e 3a, e um ruído Gaussiano à imagem 2a. As imagens com esses ruídos são 1b, 2b e 3b, que foram em seguida submetidas ao filtro de Kuan. Os testes do filtro resultaram nas imagens 1c, 2c e 3c.

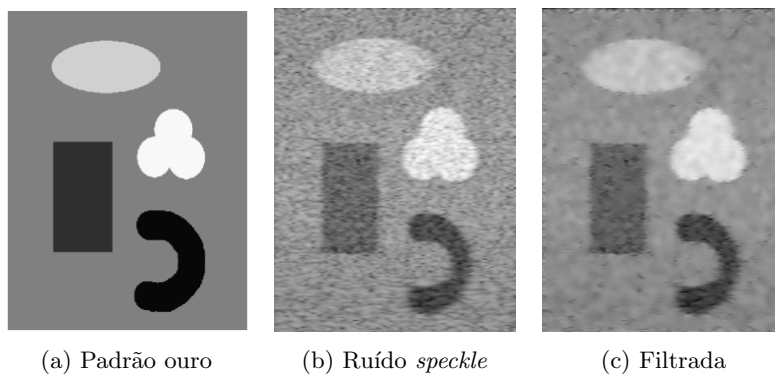


Figura 1: Imagem de teste 1, com figuras geométricas e ruído do tipo *speckle*

IV. Resultados e Discussão

IV.a Média e Desvio Padrão

Escolhemos arbitrariamente duas áreas homogêneas de cada uma das 3 imagens de teste. Para cada área, comparamos sua média e desvio padrão na imagem original e filtrada. Os resultados são mostrados na tabela 1. Repare que em todas as áreas, exceto na área II da imagem 2c, a média local sofreu leves alterações, enquanto que o desvio padrão diminuiu acentuadamente. Já na área II da imagem 2c, a média e desvio padrão permaneceram quase inalterados. Isso também pode ser percebido visualmente nas imagens 2b e 2c. A área II é o cisto mais escuro, no topo da imagem. Veja que o ruído nessa área da imagem permaneceu quase o mesmo após a filtragem. Isso se deve à escolha do C_u usado no filtro de Kuan, que difere muito do C_i da área II.

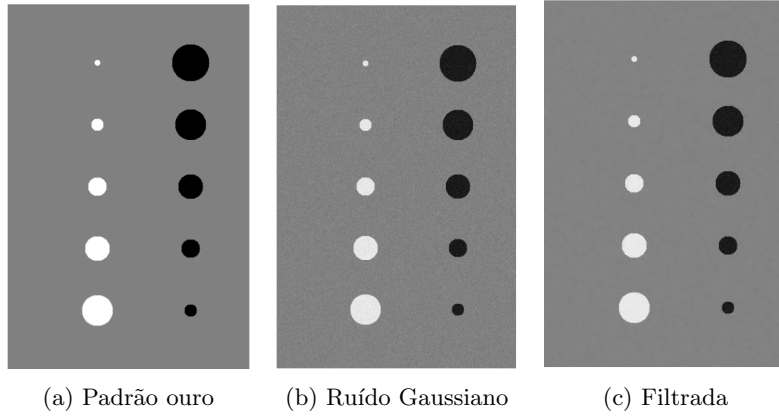


Figura 2: Imagem de teste 2, com cistos e ruído do tipo Gaussiano de 10dB

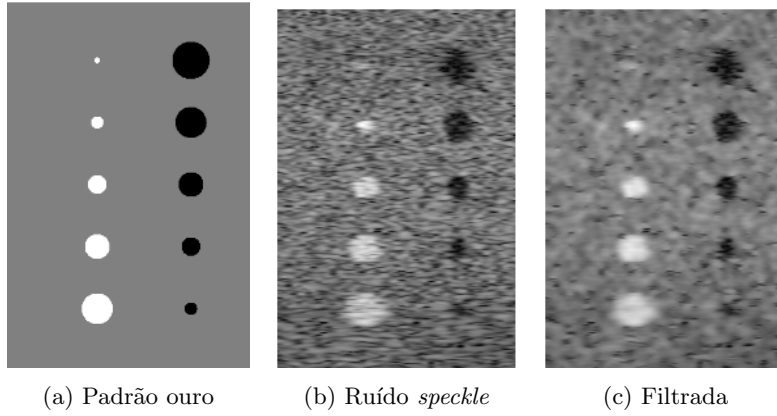


Figura 3: Imagem de teste 3, com cistos e ruído do tipo *speckle*

Imagem		C_u	Área I			Área II		
			Média	Desvio Padrão	C_i	Média	Desvio Padrão	C_i
1b	com ruído	0,090	77,10	10,23	0,133	110,78	10,21	0,092
1c	filtrada		72,07	6,63	0,092	107,87	4,40	0,041
2b	com ruído	0,045	96,99	3,80	0,039	19,79	3,87	0,195
2c	filtrada		97,70	0,92	0,009	19,73	3,79	0,192
3b	com ruído	0,200	87,92	20,17	0,229	138,11	16,24	0,118
3c	filtrada		95,32	12,84	0,135	149,30	10,61	0,071

Tabela 1: Média e desvio padrão antes e após o filtro de Kuan

IV.b Preservação das Bordas

Na seção III.b, **Medição da Eficácia**, apresentamos o *FOM* de Pratt como o método escolhido para quantificar o critério de preservação das bordas da imagem. As figuras 4 e 5 mostram como o filtro de Kuan se comporta com relação a esse critério.

Repare que as imagens 4a e 5b se aproximam muito mais do padrão ouro que 5a.

A tabela 2 apresenta os valores do *FOM* antes e após a filtragem com o algoritmo de Kuan. A imagem de teste 2, com ruído Gaussiano, teve o melhor resultado. A imagem do teste 3, com maior nível de ruído *speckle*, teve o pior resultado.

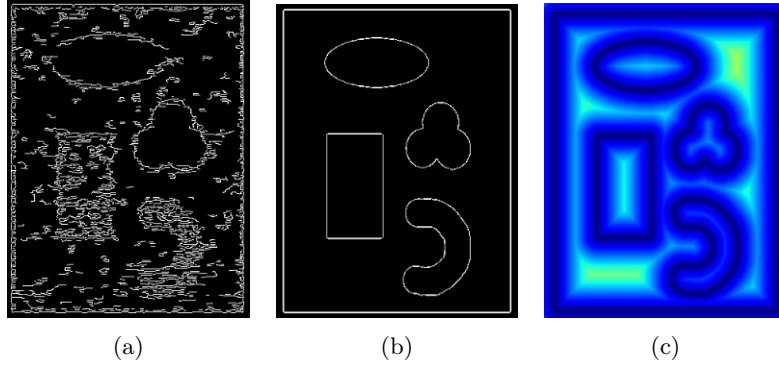


Figura 4: Teste 1; (a) e (b) são resultados do detector de bordas Canny aplicado à imagem filtrada e padrão ouro, respectivamente. (c) é a transformada da distância de (b), usada no cálculo do *FOM*

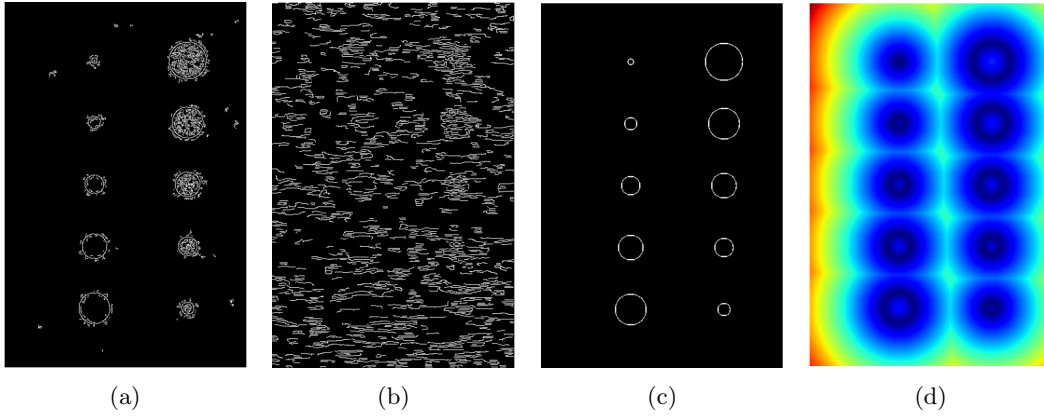


Figura 5: Testes 2 e 3; (a) e (b) são resultados do detector de bordas Canny aplicado às imagens filtradas **2c** e **3c**, respectivamente. (c) é Canny aplicado ao padrão ouro de ambos os testes. (d) é a transformada da distância de (c), usada no cálculo do *FOM*

Imagem	<i>FOM</i> com ruído	<i>FOM</i> filtrada
Teste 1	0.1766	0.3481
Teste 2	0.0430	0.4453
Teste 3	0.0462	0.0735

Tabela 2: Valores do *FOM* antes e após a filtragem

V. Conclusões

Neste trabalho conseguimos mostrar a viabilidade da implementação do filtro de Kuan para imagens com ruído multiplicativo. Também mostramos dois critérios de quantificação da eficácia do algoritmo, que permitem avaliar seu desempenho em regiões homogêneas e nas bordas da imagem. Tais critérios também foram implementados e publicados para livre uso em fins científicos ou comerciais.

Após a discussão sobre a implementação do filtro e os critérios de avaliação de desempenho, conduzimos 3 experimentos em imagens produzidas artificialmente, e apresentamos os resultados qualitativos e quantitativos de cada um deles. Algumas conclusões que merecem destaque são:

- A escolha de um bom valor de C_u é fundamental.
- Não há forma automática de se escolher C_u .
- Conhecer o tipo de ruído é essencial para obter resultados ótimos.

Referências

- [1] D. T. Kuan, A. A. Sawchuk, T. C. Strand, P. Chavel, *Adaptive Noise Smoothing Filter for Images with Signal-Dependent Noise*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 7, No. 2, (1985)
- [2] W. K. Pratt, *Digital Image Processing*, Nova Iorque, Wiley, (1977)
- [3] Y. Yu and S. T. Acton, *Speckle Reducing Anisotropic Diffusion*, IEEE Transactions on Image Processing, Vol. 11, No. 11, (2002)
- [4] F. Pérez, B. E. Granger, *IPython: A System for Interactive Scientific Computing*, Computing in Science and Engineering, Vol. 9, No. 3, (2007)
- [5] E. Jones, T. Oliphant, P. Peterson *et al.*, *SciPy: Open Source Scientific Tools for Python*, (2001)
- [6] T. Oliphant *et al.*, *Python for Scientific Computing*, Computing in Science and Engineering, Vol. 9, No. 3, (2007)
- [7] J. Canny, *A computational approach to edge detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, (1986)