



Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Departamento de Engenharia Elétrica

Trabalho de Conclusão de Curso

Aplicação de Métricas de Desempenho
a Filtros Digitais Bi-Dimensionais

Saulo Oliveira Dornellas Luiz
sauloodl@terra.com.br

Orientador:
Angelo Perkusich
perkusich@dee.ufcg.edu.br

Campina Grande, Fevereiro de 2006



Biblioteca Setorial do CDSA. Fevereiro de 2021.


Sumé - PB




Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Departamento de Engenharia Elétrica

Trabalho de Conclusão de Curso

Aplicação de Métricas de Desempenho
a Filtros Digitais Bi-Dimensionais


Saulo Oliveira Dornellas Luiz
(Aluno)


Angelo Perkusich
(Orientador)

Conteúdo

1. Introdução.....	4
2. Objetivos	5
2.1. Objetivos Gerais.....	5
2.2. Objetivos Específicos.....	5
3. Princípios de Imagens digitais.....	5
3.1. Convolução	6
3.1.1. Notação para filtros baseados na convolução	6
3.2. A transformada de Fourier	7
3.3. Ruído	7
3.4. Filtros digitais bidimensionais	8
3.4.1. Filtro de Wiener	9
3.4.2. O filtro da média	12
3.4.3. O filtro da mediana.....	13
4. Métricas de qualidade de imagem.....	13
4.1. Erro Médio Quadrático	14
4.2. Relação Sinal-Ruído.....	15
4.3. Relação Sinal-Ruído de pico.....	15
4.4. Implementação computacional das métricas de qualidade de imagem.....	15
4.4.1. Implementação em Matlab	15
4.4.2. Implementação em C.....	17
4.4.3. Aplicação das implementações em Matlab e C para comparação entre imagem ruidosa e imagem original	25
4.4.4. Análise dos resultados.....	35
5. Avaliações Subjetivas	36
5.1. Métodos de Estímulo Único.....	37
5.2. Métodos de Estímulo-Comparação	37
5.3. Realização de uma pesquisa de opinião	38
5.3.1. Página HTML para a pesquisa de opinião	39
5.3.2. Resultados da pesquisa de opinião	42
6. Conclusões	43
7. Referências Bibliográficas	45

1. Introdução

Em geral, o processo de aquisição de um sinal é também acompanhado de ruído, o que distorce a informação original. Em especial, sensores de imagem utilizados em câmeras digitais adicionam um ruído característico à imagem. A fim de reduzir o efeito do ruído, pode-se recorrer à aplicação de filtros digitais bidimensionais.

A fim de realizarmos o projeto e implementação de algoritmos para a plataforma de Processadores Digitais de Sinais baseados em aritmética de ponto fixo, devemos realizar os seguintes passos:

- 1) Formulação matemática dos algoritmos a serem implementados; Implementação e simulação dos algoritmos em linguagens de alto nível, como o MatLab e C/C++ usando aritmética de ponto flutuante;
- 2) Implementação e simulação dos algoritmos em linguagens de alto nível, como o MatLab [3] e C/C++ usando aritmética de ponto fixo;
- 3) Porte do código gerado em C/C++ usando aritmética de ponto fixo para a plataforma final (DSP);
- 4) Utilização de métricas de desempenho.

Esse projeto está dividido em três etapas, a citar:

- 1) Projeto e implementação de filtros bi-dimensionais em MatLab [3] e C para a aritmética de ponto flutuante;
- 2) Projeto e implementação de filtros bi-dimensionais em MatLab e C para aritmética de ponto fixo;
- 3) Aplicação de métricas de desempenho para os filtros digitais;

Na presente atividade, trataremos da terceira etapa do projeto. Dado um tipo de filtragem, é essencial a avaliação do filtro em questão através de métricas de desempenho, o que pode ser realizado quantitativamente através de métricas de qualidade de imagem (MSE, SNR, PSNR) ou qualitativamente através de avaliações subjetivas.

2. Objetivos

2.1. Objetivos Gerais

Conceituar o FILTRO DE WIENER e a sua versão para filtragem de imagens. Aplicar MÉTRICAS DE DESEMPENHO a imagens filtradas através do Filtro de Wiener, associando a uma AVALIAÇÃO SUBJETIVA sobre a qualidade das imagens.

2.2. Objetivos Específicos

Objetivamos descrever matematicamente o Filtro de Wiener e sua versão para remoção de ruído adaptativa bi-dimensional; aplicar Métricas de Desempenho (erro médio quadrático (MSE), relação sinal ruído (SNR), e a relação sinal ruído de pico, (PSNR)) a imagens filtradas através do Filtro de Wiener; e realizar uma Avaliação Subjetiva, com a finalidade de obter a opinião de diferentes pessoas sobre a qualidade obtida, já que cada indivíduo percebe a qualidade diferentemente. As implementações serão realizadas em ambiente MatLab utilizando aritmética de ponto flutuante.

3. Princípios de Imagens digitais

Nessa seção são apresentadas definições centrais para o processamento de imagens, e ferramentas, como a convolução, a transformada de Fourier e filtros digitais bidimensionais.

Uma imagem analógica $a(x,y)$ no espaço contínuo bidimensional, ao passar por um processo de amostragem resulta numa imagem digital $a(n_1, n_2)$ no espaço discreto bidimensional (figura 1). O processo de associação de um número inteiro a cada $a(n_1, n_2)$ é chamado de quantização [16].

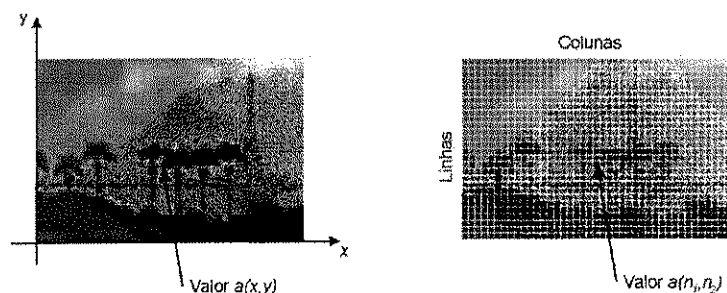


Figura 1: Digitalização de uma imagem

3.1. Convolução

A operação de convolução entre dois sinais bidimensionais a e b , resultando num sinal c bidimensional, é representada por [16]:

$$c = a * b \quad (1)$$

E o cálculo da convolução é realizado segundo a equação (2):

$$c(n_1, n_2) = a(n_1, n_2) * b(n_1, n_2) = \sum_{j=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} a(j, k) b(n_1 - j, n_2 - k) \quad (2)$$

Sejam a , b , c e d sinais bidimensionais. A operação da convolução possui as propriedades:

- Comutativa:

$$c = a * b = b * a \quad (3)$$

- Associativa:

$$c = a * (b * c) = (a * b) * c = a * b * c \quad (4)$$

- Distributiva:

$$c = a * (b + d) = a * b + a * d \quad (5)$$

3.1.1. Notação para filtros baseados na convolução

No texto desse trabalho, a descrição de filtros baseados na convolução seguirá a seguinte notação de [16]: dado um filtro $h(n_1, n_2)$, de dimensões $J \times K$, sendo J e K números ímpares, considera-se a coordenada $(n_1 = 0, n_2 = 0)$ como o centro da matriz h do filtro, como mostrado na equação abaixo.

$$h = \begin{bmatrix} h\left[-\left(\frac{J-1}{2}\right), -\left(\frac{K-1}{2}\right)\right] & \dots & \dots & h\left[0, -\left(\frac{K-1}{2}\right)\right] & \dots & \dots & h\left[\left(\frac{J-1}{2}\right), -\left(\frac{K-1}{2}\right)\right] \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \dots & h[-1, -1] & h[0, -1] & h[1, -1] & \dots & \vdots \\ h\left[-\left(\frac{J-1}{2}\right), 0\right] & \dots & h[-1, 0] & h[0, 0] & h[1, 0] & \dots & h\left[\left(\frac{J-1}{2}\right), 0\right] \\ \vdots & \dots & h[-1, 1] & h[0, 1] & h[1, 1] & \dots & \vdots \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ h\left[-\left(\frac{J-1}{2}\right), \left(\frac{K-1}{2}\right)\right] & \dots & \dots & h\left[0, \left(\frac{K-1}{2}\right)\right] & \dots & \dots & h\left[\left(\frac{J-1}{2}\right), \left(\frac{K-1}{2}\right)\right] \end{bmatrix} \quad (6)$$

O cálculo da convolução do sinal bidimensional $a(n_1, n_2)$, de dimensões $M \times N$, sendo M e N números ímpares, com um filtro $h(n_1, n_2)$ para valores de n_1 e n_2 que estejam nos limites de $a(n_1, n_2)$ evidencia a necessidade de valores de $a(n_1, n_2)$ que estão além dos limites desse sinal bidimensional, como é exemplo:

$$c(M_0, N_0) = \sum_{j=-J_0}^{+J_0} \sum_{k=-K_0}^{+K_0} h(j, k) a(M_0 - j, N_0 - k) \quad (7)$$

$$\text{onde } J_0 = \frac{J-1}{2}, \quad K_0 = \frac{K-1}{2}, \quad M_0 = \frac{M-1}{2}, \quad \text{e } N_0 = \frac{N-1}{2}$$

As alternativas comuns para esse problema são [16]: (a) estender a imagem com valores constantes (possivelmente 0), (b) estender a imagem periodicamente, (c) estender a imagem a espelhando nos seus limites (d) estender os valores nos seus limites infinitamente.

3.2. A transformada de Fourier

A transformada de Fourier resulta numa representação do sinal como uma soma de exponenciais complexas com pesos específicos [16]. Dado um sinal bidimensional a , sua transformada de Fourier é $A = F\{a\}$, onde ocorre a passagem do domínio espacial discreto para o domínio contínuo da frequência. A transformada inversa é expressa como $a = F^{-1}\{A\}$. A transformada de Fourier é inversível, assim $a = F^{-1}\{F\{a\}\}$ e $A = F\{F^{-1}\{A\}\}$.

As fórmulas para transformação entre os domínios espacial e da frequência são as seguintes:

$$A(\Omega, \Psi) = \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} a(n_1, n_2) e^{-j(\Omega n_1 + \Psi n_2)} \quad (8)$$

$$a(n_1, n_2) = \frac{1}{4\pi^2} \int_{-\pi}^{+\pi} \int_{-\pi}^{+\pi} A(\Omega, \Psi) e^{j(\Omega n_1 + \Psi n_2)} d\Omega d\Psi \quad (9)$$

Sejam os sinais bidimensionais a , b e c e suas respectivas transformadas de Fourier A , B e C . As duas equações abaixo mostram que a convolução no domínio espacial é equivalente à multiplicação no domínio da frequência e vice-versa.

$$c = a * b \xrightarrow{F} C = A \cdot B \quad (10)$$

$$c = a \cdot b \xrightarrow{F} C = \frac{1}{4\pi^2} A * B \quad (11)$$

3.3. Ruído

Os sinais obtidos a partir dos sensores de imagem podem ser corrompidos por vários tipos de ruído [16]. O ruído refere-se aqui a variações estocáticas, em

contraposição a distorções determinísticas, como é exemplo a falta de foco. São descritos a seguir alguns tipos de ruído:

- **Ruído de fóton:** Sendo a luz a natureza do sinal físico observado, então a natureza quântica da luz se torna importante [16]. Apenas um fóton com $\lambda=500$ nm carrega uma energia de $3,97 \cdot 10^{-19}$ Joules e sensores CCD modernos são suficientemente sensíveis para contar fótons individuais. O ruído de fóton deriva da natureza fundamentalmente estatística da produção de fótons. Não se pode afirmar que o mesmo número de fótons será contado num mesmo pixel em dois intervalos de tempo consecutivos mas independentes e de comprimento T.
- **Ruído eletrônico on-chip:** origina-se no processo de leitura do sinal pelo sensor; no caso de um chip CCD, pelo transistor FET.
- **Ruído KTC:** associado ao capacitor de gate de um FET. Pode ser não negligenciável.
- **Ruído de amplificação:** seu modelo padrão é aditivo, Gaussiano e independente do sinal. Pode ser negligenciado em eletrônicos modernos bem projetados. Um exemplo desse tipo de ruído é o caso de câmeras coloridas em que o canal azul requer maior amplificação que os canais verde e vermelho, causando maior ruído no canal azul.
- **Ruído de quantização:** é inerente ao processo de quantização de amplitude presente no conversor analógico-digital, ADC. Esse ruído é aditivo e independente do sinal quando o número de níveis é maior ou igual a 16. Geralmente o ruído de quantização pode ser ignorado já que a SNR total de um sistema completo é normalmente dominada pela menor SNR, que no caso de câmeras CCD é a causada pelo ruído de fóton.

3.4. Filtros digitais bidimensionais

Os equipamentos utilizados na aquisição e transmissão de imagens podem adicionar a elas ruído. Para combater este tipo de ruído são normalmente empregados filtros digitais bidimensionais. Tal técnica se baseia na idéia de que, se a imagem do ruído ou a sua transformada de Fourier podem ser determinados, então subtraindo-se a transformada do ruído da transformada da imagem resulta na transformada da imagem sem ruído desejada, à qual pode ser aplicada a transformada inversa.

Filtros lineares podem ser tratados através da análise de Fourier, enquanto filtros não-lineares não podem ser tratados por essa análise[16]. Nesta seção, serão

$$\begin{aligned}
E\{f(n_1, n_2)g^*(m_1, m_2)\} &= E\{(e(n_1, n_2) + p(n_1, n_2))g^*(m_1, m_2)\} \\
&= E\{p(n_1, n_2)g^*(m_1, m_2)\} = E\{g(n_1, n_2) * h(n_1, n_2)g^*(m_1, m_2)\} \\
&= \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} h(k_1, k_2) E\{g(n_1 - k_1, n_2 - k_2)g^*(m_1, m_2)\}
\end{aligned} \tag{16}$$

Reescrevendo a equação acima:

$$\begin{aligned}
R_{fg}(n_1 - m_1, n_2 - m_2) &= \\
&= \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} h(k_1, k_2) R_g(n_1 - k_1 - m_1, n_2 - k_2 - m_2)
\end{aligned} \tag{18}$$

Assim:

$$R_{fg}(n_1, n_2) = h(n_1, n_2) * R_g(n_1, n_2) \tag{19}$$

$$H(\omega_1, \omega_2) = \frac{P_{fg}(\omega_1, \omega_2)}{P_g(\omega_1, \omega_2)} \tag{20}$$

O filtro definido pela equação acima é denominado filtro de Wiener não causal [12].

Supondo que $f(n_1, n_2)$ é descorrelacionado com $v(n_1, n_2)$,

$$E\{f(n_1, n_2)v^*(m_1, m_2)\} = E\{f(n_1, n_2)\}E\{v^*(m_1, m_2)\} \tag{21}$$

Devido ao fato de $f(n_1, n_2)$ e $v(n_1, n_2)$ serem processos de média zero, tem-se

$$\begin{aligned}
R_{fg}(n_1, n_2) &= R_f(n_1, n_2) \\
R_g(n_1, n_2) &= R_f(n_1, n_2) + R_v(n_1, n_2)
\end{aligned} \tag{22}$$

Assim como

$$\begin{aligned}
P_{fg}(\omega_1, \omega_2) &= P_f(\omega_1, \omega_2) \\
P_g(\omega_1, \omega_2) &= P_f(\omega_1, \omega_2) + P_v(\omega_1, \omega_2)
\end{aligned} \tag{23}$$

Então

$$H(\omega_1, \omega_2) = \frac{P_f(\omega_1, \omega_2)}{P_f(\omega_1, \omega_2) + P_v(\omega_1, \omega_2)} \tag{24}$$

Adicionando a condição de que $f(n_1, n_2)$ e $v(n_1, n_2)$ são funções amostra de um campo aleatório Gaussiano, a equação (14) será um problema de estimação do tipo mínimo erro médio quadrático e o filtro de Wiener da equação 24 é o estimador ótimo para erro médio quadrático mínimo.

As DEP's $P_f(\omega_1, \omega_2)$ e $P_v(\omega_1, \omega_2)$ são reais e não negativas, o que implica que $H(\omega_1, \omega_2)$ também é real e não negativa. Logo o filtro de Wiener modifica a DEP do sinal em amplitude mas não em fase. Também percebe-se a partir da equação (24) que se $P_v(\omega_1, \omega_2) \rightarrow 0$, $H(\omega_1, \omega_2) \rightarrow 1$, mostrando que o filtro tende a preservar as componentes em frequência com alta SNR. Já se $P_v(\omega_1, \omega_2) \rightarrow \infty$, $H(\omega_1, \omega_2) \rightarrow 0$, mostrando que o filtro tende a atenuar as componentes em frequência com baixa SNR [12].

3.4.1.2. Filtro de Wiener Adaptativo: o método de Lee

O filtro de Wiener exposto acima considera que a imagem é homogênea, surgindo então um filtro invariante no espaço. Contudo uma imagem é um campo aleatório não homogêneo, fazendo com que um modelo homogêneo não seja o mais adequado. O filtro de Wiener discutido na seção 3.4.1.1 requer a estimativa da média m_f do sinal, a média m_v do ruído, as DEP's $P_f(\omega_1, \omega_2)$ e $P_v(\omega_1, \omega_2)$ do sinal e do ruído respectivamente. O filtro de Wiener adaptativo ao invés de assumir esses parâmetros fixos para toda a imagem, os estima localmente. A discussão que se segue descreve o método de Lee para projeto do filtro de Wiener adaptativo [12].

Considera-se que o ruído aditivo branco $v(n_1, n_2)$ tem média nula e variância σ_v^2 . Logo sua DEP $P_v(\omega_1, \omega_2) = \sigma_v^2$. Considera-se então uma pequena região local em que o sinal $f(n_1, n_2)$ é tido como homogêneo, e modelado como

$$f(n_1, n_2) = m_f + \sigma_f w(n_1, n_2) \quad (25)$$

em que m_f e σ_f são a média local e o desvio padrão de $f(n_1, n_2)$ e $w(n_1, n_2)$ é um ruído branco com média zero e variância unitária. Assim no interior da região local, o filtro de Wiener $H(\omega_1, \omega_2)$ e $h(n_1, n_2)$ é expresso por:

$$H(\omega_1, \omega_2) = \frac{P_f(\omega_1, \omega_2)}{P_f(\omega_1, \omega_2) + P_v(\omega_1, \omega_2)} = \frac{\sigma_f^2}{\sigma_f^2 + \sigma_v^2} \quad (26)$$

$$h(n_1, n_2) = \frac{\sigma_f^2}{\sigma_f^2 + \sigma_v^2} \delta(n_1, n_2) \quad (27)$$

Assim a imagem filtrada $p(n_1, n_2)$ no interior da região local será

$$\begin{aligned}
p(n_1, n_2) &= m_f + (g(n_1, n_2) - m_f) * \frac{\sigma_f^2}{\sigma_f^2 + \sigma_v^2} \delta(n_1, n_2) \\
&= m_f + \frac{\sigma_f^2}{\sigma_f^2 + \sigma_v^2} (g(n_1, n_2) - m_f)
\end{aligned} \tag{28}$$

Assumindo que m_f e σ_f são calculados para cada pixel:

$$p(n_1, n_2) = m_f(n_1, n_2) + \frac{\sigma_f^2(n_1, n_2)}{\sigma_f^2(n_1, n_2) + \sigma_v^2(n_1, n_2)} (g(n_1, n_2) - m_f(n_1, n_2)) \tag{29}$$

No caso onde se possui apenas uma imagem com ruído, só é possível estimar $\sigma_g^2(n_1, n_2)$. Como $\sigma_g^2(n_1, n_2) = \sigma_f^2(n_1, n_2) + \sigma_v^2(n_1, n_2)$, a equação 29 se torna

$$p(n_1, n_2) = m_f(n_1, n_2) + \frac{\sigma_g^2(n_1, n_2) - \sigma_v^2(n_1, n_2)}{\sigma_g^2(n_1, n_2)} (g(n_1, n_2) - m_f(n_1, n_2)) \tag{30}$$

O desempenho do filtro de Wiener adaptativo é melhor que sua versão não adaptativa [12].

3.4.2. O filtro da média

O filtro da média associa a cada pixel da imagem a média entre esse pixel e seus vizinhos. Em geral as imagens filtradas pelo filtro da média sofrem o efeito “blur”, ou seja, se tornam mais turvas, perdendo detalhes de contornos e das fronteiras em suas regiões [13].

Um exemplo de filtro da média para o caso de uma janela de 5 x 5 é [16]:

$$h = \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{31}$$

Um filtro da média de dimensões $J \times K$, sendo J e K números ímpares, é

normalizado de tal forma que $\sum_{j=-\frac{J-1}{2}}^{\frac{J-1}{2}} \sum_{k=-\frac{K-1}{2}}^{\frac{K-1}{2}} h(j, k) = 1$.

3.4.3. O filtro da mediana

O filtro da mediana associa a cada pixel a mediana da região local em que esse pixel se encontra. A mediana é obtida ordenando-se os valores das magnitudes dos pixels em ordem crescente e então tomando o valor no centro; se dois valores estiverem no centro, toma-se a média entre os dois. O filtro da mediana apresenta bom desempenho na remoção de ruído do tipo salt and pepper, pois nos filtros em geral os pixels desse tipo de ruído são considerados nos cálculos, enquanto no filtro da mediana são tomados apenas um ou dois valores de “pixels saudáveis”. Assim como o filtro da média, o filtro da mediana também reduz a qualidade da imagem [13].

4. Métricas de qualidade de imagem

Nessa seção introduziremos três métricas de qualidade de imagem a filtros digitais bidimensionais: o erro médio quadrático, MSE (do inglês *mean squared error*); a relação sinal-ruído, SNR (do inglês *signal to noise ratio*); e a relação sinal-ruído de pico, PSNR (do inglês *peak signal to noise ratio*).

A restauração de imagens representa a reconstrução ou estimativa de uma imagem não corrompida a partir de sua versão distorcida. Há aplicações em diversas áreas, tais como medicina, exploração espacial e comércio [6].

O processo de filtragem de uma imagem tem por objetivo reconstruir uma imagem corrompida a partir de sua versão com ruído. No contexto de restauração de imagens, a expressão “qualidade de uma imagem” normalmente denota a sua fidelidade com relação à sua versão original sem ruído. Assim a aplicação de uma métrica de qualidade de imagem a um filtro digital bidimensional significa medir o aumento na qualidade da imagem devido à filtragem [6].

A meta da pesquisa em avaliação objetiva da qualidade de imagens é desenvolver métricas que possam prever automaticamente a qualidade da imagem. As métricas de desempenho são de grande importância numa ampla gama de aplicações, tais como aquisição de imagens, compressão, comunicação, restauração, análise, reprodução, impressão e marcas d’água [9].

Uma métrica de qualidade de imagem é geralmente utilizada nos seguintes casos [11]:

- Monitoramento da qualidade de imagem para sistemas de controle de qualidade de imagem. Como exemplo disso, num sistema de aquisição de

vídeo e imagem, uma métrica de qualidade pode ser utilizada para monitorar e realizar ajustes para se obter a melhor qualidade de imagem e vídeo.

- Estabelecer um nível de referência para sistemas e algoritmos de processamento de imagem. Para exemplificar isso é possível citar o caso em que se deseja selecionar um entre vários sistemas de processamento de imagem para uma tarefa específica, situação em que uma métrica de qualidade de imagem pode oferecer indicações sobre qual deles oferece imagens de melhor qualidade.
- A inclusão de métricas de qualidade de imagem em sistemas de processamento de imagem para otimizar os algoritmos e parâmetros de configuração.

As métricas de qualidade de imagem podem ser classificadas quanto à disponibilidade de uma imagem original a ser comparada com a imagem ruidosa. Assim as avaliações de qualidade podem ser dos tipos [14]:

- *Referência completa*, em que uma imagem de referência completa está disponível;
- *Sem referência*, em que não há nenhuma imagem de referência;
- *Referência reduzida*, onde uma imagem de referência é parcialmente disponibilizada, sob a forma de um conjunto de características que podem auxiliar a avaliação da imagem distorcida.

Este trabalho utiliza avaliações de qualidade do tipo referência completa, mais especificamente a MSE, a SNR e a PSNR que possuem significados físicos.

A avaliação objetiva da qualidade de imagens tem por finalidade prever automaticamente a qualidade percebida de imagens [11].

4.1. Erro Médio Quadrático

A função erro [4] é definida segundo a equação (32):

$$e(x, y) = (f(x, y) - A(x, y)) \quad (32)$$

onde $f(x, y)$ é o nível do sinal vermelho, verde ou azul individual da imagem original e $A(x, y)$ é o correspondente nível do sinal vermelho, verde ou azul individual da imagem filtrada. Neste trabalho serão analisadas imagens no formato BMP que estão

representadas no esquema RGB (vermelho, verde, azul), e a aplicação da função erro ocorrerá para cada cor individualmente.

A partir da função erro, obtemos o erro médio quadrático [10] segundo a equação (33):

$$MSE = \frac{1}{NM} \sum_{x=1}^N \sum_{y=1}^M (e(x, y))^2 \quad (33)$$

onde N é o número de pixels na direção horizontal, e M é o número de pixels na direção vertical.

4.2. Relação Sinal-Ruído

A relação sinal-ruído [4] é definida pela equação (34):

$$SNR_{rms} = \frac{\sum_{x=1}^N \sum_{y=1}^M (A(x, y))^2}{\sum_{x=1}^N \sum_{y=1}^M (e(x, y))^2} \quad (34)$$

onde $A(x,y)$ é o nível do sinal vermelho, verde ou azul individual da imagem filtrada e $e(x,y)$ é a função erro definida na equação (32).

4.3. Relação Sinal-Ruído de pico

A relação sinal-ruído de pico [4] é definida pela equação (35):

$$PSNR = 20 \log_{10} \frac{2^p - 1}{\sqrt{MSE}} \quad (35)$$

Onde p é o número de bits por pixel usado para definição do nível da componente de cor (vermelho, verde ou azul) considerada e MSE é definido na equação (33). A PSNR é expressa em dB .

4.4. Implementação computacional das métricas de qualidade de imagem

4.4.1. Implementação em Matlab

As métricas de qualidade de imagem definidas nas seções 4.1, 4.2 e 4.3 foram implementadas em Matlab através do programa abaixo, que tem como entrada uma imagem de referência (primeiro parâmetro) e uma imagem filtrada (segundo parâmetro) e como saída uma imagem erro imagemErro.bmp cujas componentes vermelha, verde e azul são obtidas aplicando-se a função erro às correspondentes componentes das

imagens de entrada. O programa calcula o erro médio quadrático, MSE, a relação sinal-ruído, *SNR*, e a relação sinal-ruído de pico, *PSNR*, para cada componente de cor.

Programa em Matlab:

```
*****
% A função AvaliacaoObjetiva calcula as métricas de qualidade
% da imagem MSE, SNR RMS e PSNR.

% Parâmetros:
% imagemOriginal: uma imagem de referência (sem ruído).
% imagemFiltrada: imagem resultante da filtragem de uma versão ruidosa da
% imagem de referência.
% Essas imagens devem estar no workspace do Matlab em que se está
% trabalhando.
% Exemplo de uso: AvaliacaoObjetiva('saladaCCD.bmp', 'wiener.bmp')
function AvaliacaoObjetiva(imagemOriginal, imagemFiltrada)

clc
close all

% Abrindo os arquivos que contém as imagens original e filtrada
I1 = imread(imagemOriginal);
I2 = imread(imagemFiltrada);

RGB1 = 255*im2double(I1);
RGB2 = 255*im2double(I2);

% Cálculo dos erros ponto-a-ponto
e = RGB1-RGB2;

% Cálculo do MSE para a figura filtrada
N = size(e(:,1),2);
M = size(e(:,1),1);
for i = 1:3
    MSE(i) = 1/(N*M)*sum(sum(e(:,i).^2));
end
disp(['MSE na Componente Vermelha: ' num2str(MSE(1))]);
disp(['MSE na Componente Verde: ' num2str(MSE(2))]);
disp(['MSE na Componente Azul: ' num2str(MSE(3))]);

% Cálculo da relação sinal ruído SNR para a figura filtrada
SNRrmsVermelha = (sum(sum(RGB2(:,1).^2)))/(sum(sum(e(:,1).^2)));
SNRrmsVerde = (sum(sum(RGB2(:,2).^2)))/(sum(sum(e(:,2).^2)));
SNRrmsAzul = (sum(sum(RGB2(:,3).^2)))/(sum(sum(e(:,3).^2)));
disp(['SNR RMS na Componente Vermelha: ' num2str(SNRrmsVermelha)]);
disp(['SNR RMS na Componente Verde: ' num2str(SNRrmsVerde)]);
disp(['SNR RMS na Componente Azul: ' num2str(SNRrmsAzul)]);
```



```

%Cálculo da relação sinal ruído de pico PSNR para a figura filtrada
p=8;
PSNRrms = 20*log10((2^p-1)./sqrt(MSE));
disp(['PSNR na Componente Vermelha: ' num2str(PSNRrms(1))]);
disp(['PSNR na Componente Verde: ' num2str(PSNRrms(2))]);
disp(['PSNR na Componente Azul: ' num2str(PSNRrms(3))]);

RGB1=1/255*RGB1;
RGB2=1/255*RGB2;
e=1/255*(255-5*abs(e));
figure(1)
imshow(RGB1)
title('Imagem original')

figure(2)
imshow(RGB2)
title('Imagem filtrada')

figure(3)
imshow(e)
title('Imagem erro')
*****

```

4.4.2. Implementação em C

As métricas de qualidade de imagem definidas nas seções 4.1, 4.2 e 4.3 foram implementadas em C através do programa abaixo. O programa tem como entrada uma imagem de referência (primeiro parâmetro) e uma imagem filtrada (segundo parâmetro) e como saída uma imagem erro imagemErro.bmp cujas componentes vermelha, verde e azul são obtidas aplicando-se a função erro às correspondentes componentes das imagens de entrada. O programa calcula o erro médio quadrático, MSE, a relação sinal-ruído, SNR, e a relação sinal-ruído de pico, PSNR, para cada componente.

O programa em C deve ser chamado da seguinte forma:

AvaliacaoObjetiva imagemOriginal.bmp imagemFiltrada.bmp

Ele gera um arquivo imagemErro.bmp, que é a imagem erro resultante da comparação entre as imagens original e filtrada. O programa também escreve na saída padrão o resultado das métricas de qualidade de imagem, como ilustra a figura 7 para o exemplo do filtro de Wiener.

Programa em C:

```

*****
#include <stdio.h>
#include <stdlib.h>

```

```

#include <time.h>
#include <math.h>

/*-----PROTOTYPES-----*/
long getImagemInfo(FILE*, long, int);
void copyImagemInfo(FILE* inputFile, FILE* outputFile);
void imagemErro(unsigned char *, unsigned char *, unsigned char *, int, int );
void funcaoErro(unsigned char *, unsigned char *, int *, int, int );
double mse(int *, int, int );
double snrRMS(unsigned char *, int *, int, int );
double psnr(int, double);

void main(int argc, char* argv[])
{

    FILE *entradaBMP1, *entradaBMP2, *saidaBMP;

    int linhas, colunas, l, c;
    int i=0;
    int power=1;
    int janela=0;
    long nColors;
    unsigned char    someChar;
    unsigned char*   pChar;
    long fileSize; /* BMP file size */
    double initClock, time;

    initClock=clock();

    /* Inicializa ponteiro */
    someChar = '0';
    pChar = &someChar;

    if(argc < 3)
    {
        printf("\nUso: %s imagemOriginal.bmp imagemFiltrada.bmp\n", argv[0]);
        exit(0);
    }

    printf("Lendo arquivo %s\n", argv[1]);

    entradaBMP1 = fopen(argv[1], "rb");//Abrindo arquivo de entrada 1 (imagem
ruidosa)
    entradaBMP2 = fopen(argv[2], "rb");//Abrindo arquivo de entrada 2 (imagem
filtrada)
    saidaBMP = fopen("imagemErro.bmp", "wb");//Abrindo arquivo de saída
(imagem erro)

    /*-----OBTENDO DADOS DO ARQUIVO DE ENTRADA 1-----*/

```

```

colunas = (int)getImageInfo(entradaBMP1, 18, 4);
linhas = (int)getImageInfo(entradaBMP1, 22, 4);
nColors = getImageInfo(entradaBMP1, 46, 8);
fileSize = getImageInfo(entradaBMP1, 2, 4);

//Tratamento para o caso de o cabeçalho do arquivo de entrada 1 estar
//corrompido
if(fileSize!=(linhas*colunas*3+54)){
    printf("\nArquivo de entrada errado... Pssivelmente seu cabeçalho está
corrompido\n");
    exit(0);
}

/*-----IMPRIME OS DADOS DO ARQUIVO DE ENTRADA 1 NA TELA-----*/
printf("largura: %d\n", colunas);
printf("Altura: %d\n", linhas);
printf("Tamanho do arquivo: %ld\n", fileSize);
printf("#Cores: %ld\n", nColors);
printf("Tamanho da janela: %d\n", janela);

copyImageInfo(entradaBMP1, saidaBMP);

/*----PARA BMP 24-BIT, NÃO HÁ TABELA DE CORES----*/
//Posicionando os arquivos BMP no setor de dados
fseek(entradaBMP1, 54, SEEK_SET);
fseek(entradaBMP2, 54, SEEK_SET);
fseek(saidaBMP, 54, SEEK_SET);

//Alocação dinâmica para as componente vermelha, verde e azul.

{//Escopo para execução das métricas de desempenho

unsigned char *vermelho1 =(unsigned char *) malloc((sizeof(unsigned
char)*linhas*colunas));
unsigned char *verde1 = (unsigned char *)malloc((sizeof(unsigned
char)*linhas*colunas));
unsigned char *azul1 = (unsigned char *)malloc((sizeof(unsigned char) * linhas
* colunas));

unsigned char *vermelho2 =(unsigned char *) malloc((sizeof(unsigned
char)*linhas*colunas));
unsigned char *verde2 = (unsigned char *)malloc((sizeof(unsigned
char)*linhas*colunas));
unsigned char *azul2 = (unsigned char *)malloc((sizeof(unsigned char) * linhas
* colunas));

unsigned char *vermelhoSaida =(unsigned char *) malloc((sizeof(unsigned
char)*linhas*colunas));

```

```

unsigned char *verdeSaida = (unsigned char *)malloc((sizeof(unsigned
char)*linhas*colunas));
unsigned char *azulSaida = (unsigned char *)malloc((sizeof(unsigned char) *
linhas * colunas));

int *vermelhoErro =(int *) malloc((sizeof(int)*linhas*colunas));
int *verdeErro = (int *)malloc((sizeof(int)*linhas*colunas));
int *azulErro = (int *)malloc((sizeof(int) * linhas * colunas));

double mseVermelho, mseVerde, mseAzul;

/*---LENDO AS COMPONENTES DE COR DO ARQUIVO DE ENTRADA 1---*/
for(l=0; l<=linhas - 1; l++){
    for(c=0; c<=colunas - 1; c++){

        /*LENDO O PRIMEIRO BYTE PARA OBTER A COMPONENTE AZUL*/
        fread(pChar, sizeof(char), 1, entradaBMP1);
        *(azul1+l*colunas + c) = *pChar;

        /*-----LENDO O PROXIMO BYTE PARA OBTER A COMPONENTE
VERDE-----*/
        fread(pChar, sizeof(char), 1, entradaBMP1);
        *(verde1+l*colunas + c) = *pChar;

        /*-----LENDO O PROXIMO BYTE PARA OBTER A COMPONENTE
VERMELHA-----*/
        fread(pChar, sizeof(char), 1, entradaBMP1);
        *(vermelho1+l*colunas + c) = *pChar;
    }
}

/*-----LENDO AS COMPONENTES DE COR DO ARQUIVO DE ENTRADA
2-----*/
for(l=0; l<=linhas - 1; l++){
    for(c=0; c<=colunas - 1; c++){

        /*-----LENDO O PRIMEIRO BYTE PARA OBTER A COMPONENTE
AZUL-----*/
        fread(pChar, sizeof(char), 1, entradaBMP2);
        *(azul2+l*colunas + c) = *pChar;

        /*-----LENDO O PROXIMO BYTE PARA OBTER A COMPONENTE
VERDE-----*/
        fread(pChar, sizeof(char), 1, entradaBMP2);
        *(verde2+l*colunas + c) = *pChar;

        /*-----LENDO O PROXIMO BYTE PARA OBTER A COMPONENTE
VERMELHA-----*/
        fread(pChar, sizeof(char), 1, entradaBMP2);
        *(vermelho2+l*colunas + c) = *pChar;
    }
}

```

```

    }
}

//Encontrando as funcoes erro das componentes Azul, Verde e Vermelho
//funcaoErro(vermelho1, vermelho2, vermelhoErro, linhas, colunas);
funcaoErro(verde1, verde2, verdeErro, linhas, colunas);
funcaoErro(azul1, azul2, azulErro, linhas, colunas);

//Encontrando as imagens erro das componentes Azul, Verde e Vermelho
imagemErro(vermelho1, vermelho2, vermelhoSaida, linhas, colunas);
imagemErro(verde1, verde2, verdeSaida, linhas, colunas);
imagemErro(azul1, azul2, azulSaida, linhas, colunas);

/*-----ESCREVENDO A IMAGEM DE SAÍDA-----*/
for(l=0; l<=linhas - 1; l++){
    for(c=0; c<=colunas - 1; c++){

        *pChar=(azulSaida+l*colunas + c);
        fwrite(pChar, sizeof(char), 1, saidaBMP);

        /*----ESCREVENDO A COPONENTE VERDE NO PROXIMO BYTE-----*/
        *pChar=(verdeSaida+l*colunas + c);
        fwrite(pChar, sizeof(char), 1, saidaBMP);

        /*----ESCREVENDO A COPONENTE VERMELHA NO PROXIMO BYTE-----*/
        *pChar=(vermelhoSaida+l*colunas + c);
        fwrite(pChar, sizeof(char), 1, saidaBMP);
    }
}

//Encontrando o MSE das componentes Azul, Verde e Vermelho
mseAzul = mse(azulErro, linhas, colunas);
mseVerde = mse(verdeErro, linhas, colunas);
mseVermelho = mse(vermelhoErro, linhas, colunas);

printf("\nMSE na Componente Vermelha: %.3f", mseVermelho);
printf("\nMSE na Componente Verde: %.3f", mseVerde);
printf("\nMSE na Componente Azul: %.3f\n", mseAzul);

//Encontrando o SNR RMS das componentes Azul, Verde e Vermelho
printf("\nSNR RMS na Componente Vermelha: %.3f", snrRMS(vermelho2,
vermelhoErro, linhas, colunas));
printf("\nSNR RMS na Componente Verde: %.3f", snrRMS(verde2, verdeErro,
linhas, colunas));
printf("\nSNR RMS na Componente Azul: %.3f\n", snrRMS(azul2, azulErro,
linhas, colunas));

//Encontrando o PSNR das componentes Azul, Verde e Vermelho
printf("\nPSNR na Componente Vermelha: %.3f dB", psnr(8, mseVermelho));

```

```

printf("\nPSNR na Componente Verde: %.3f dB", psnr(8, mseVerde));
printf("\nPSNR na Componente Azul: %.3f dB\n", psnr(8, mseAzul));

} //Fim do escopo para execução das métricas de desempenho

fclose(entradaBMP1); //Fechando arquivo de entrada 1 (imagem ruidosa)
fclose(entradaBMP2); //Fechando arquivo de entrada 2 (imagem filtrada)
fclose(saidaBMP); //Fechando arquivo de saída (imagem erro)

/*Cálculo e impressão do tempo decorrido para calculo das metricas de
desempenho*/
time=((double)clock()- (double)initClock)/(double)CLK_TCK;
printf("\nTempo decorrido para calculo das metricas de desempenho: %.3f
seconds\n", time);

}

/*-----SUBPROGRAMA PARA OBTER DADOS DA IMAGEM-----*/
long getImagemInfo(FILE* inputFile, long offset, int numberOfChars)
{
    unsigned char        *ptrC;
    long                 value = 0L;
    unsigned char        dummy;
    int                  i;
    float power=1.0;

    dummy = '0';
    ptrC = &dummy;

    fseek(inputFile, offset, SEEK_SET);

    for(i=0; i<numberOfChars; i++)
    {
        fread(ptrC, sizeof(char), 1, inputFile);
        /* calculate value based on adding bytes */

        value = (long)(value + (*ptrC)*(power));
        power=power*256;
    }
    return(value);
} /* end of getImagemInfo */

/*-----COPIA CABEÇALHO E DADOS-----*/
void copyImagemInfo(FILE* inputFile, FILE* outputFile)
{
    unsigned char        *ptrC;
    unsigned char        dummy;
    int                  i;

```

```

dummy = '0';
ptrC = &dummy;

fseek(inputFile, 0L, SEEK_SET);
fseek(outputFile, 0L, SEEK_SET);

for(i=0; i<=53; i++)
{
    fread(ptrC, sizeof(char), 1, inputFile);
    fwrite(ptrC, sizeof(char), 1, outputFile);
}
}

/*-----GERA A IMAGEM ERRO ENTRE DUAS IMAGENS-----*/
//Parâmetros:
//imagem1 é a imagem original.
//imagem2 é a imagem filtrada.
//ImagemSaida é a imagem erro.
//linhas e colunas é o números de linhas e colunas de qualquer uma dessas
//imagens.
void imagemErro(unsigned char *imagem1, unsigned char *imagem2, unsigned
char *imagemSaida, int linhas, int colunas)
{
    int n,m,tmp, erro;

    for (n=0;n<linhas;n++){
        for (m=0; m<colunas; m++){
            tmp=n*colunas+m;
            //A linha de comando abaixo seguida do if calcula o erro
            //entre dois pixels correspondentes de imagem1 e
            //imagem2 como |255-5*|imagem1[tmp]-imagem2[tmp]|
            //O fator 5 multiplicando o módulo da diferença entre os
            //pixels é um realça o erro para que ele poça ser
            //melhor visualizado. Esse produto é subtraído de 255,
            //porque se fosse renderizada a imagem de
            //5*|imagem1[tmp]-imagem2[tmp]| teríamos uma imagem
            //escura com pontos mais claros, de difícil visualização;
            //assim a imagem erro é feita igual a |255-5*|imagem1[tmp]-
            //imagem2[tmp]| para que se tenha uma imagem clara
            //com pontos em que ocorre o erro.
            erro=(int)(255-5*fabs(imagem1[tmp]-imagem2[tmp]));
            if(erro<0){
                imagemSaida[tmp]=(unsigned char)0;
            }else{
                imagemSaida[tmp]=(unsigned char)erro;
            }
        }
    }
}

```

```

}

/*-----CALCULA O ERRO ENTRE DUAS IMAGENS-----*/
//Parâmetros:
//imagem1 é a imagem original.
//imagem2 é a imagem filtrada.
//ImagemSaida é a imagem erro.
//linhas e colunas é o números de linhas e colunas de qualquer uma dessas
//imagens.
void funcaoErro(unsigned char *imagem1, unsigned char *imagem2, int
*imagemSaida, int linhas, int colunas)
{
    int n,m,tmp;

    for (n=0;n<linhas;n++){
        for (m=0; m<colunas; m++){
            tmp=n*colunas+m;
            imagemSaida[tmp]=(int)((int)imagem1[tmp]-
(int)imagem2[tmp]);
        }
    }
}

/*-----CALCULA O ERRO MÉDIO QUADRÁTICO-----*/
//Parâmetros:
//ImagemErro é a imagem erro.
//linhas e colunas é o números de linhas e colunas respectivamente da imagem
//erro.
double mse(int *imagemErro, int linhas, int colunas)
{
    int n,m,tmp;
    double mse = 0;

    for (n=0;n<linhas;n++){
        for (m=0; m<colunas; m++){
            tmp=n*colunas+m;

            mse=mse+((double)imagemErro[tmp])*((double)imagemErro[tmp]);
        }
    }
    mse = 1/((double)linhas*(double)colunas)*mse;

    return mse;
}

/*-----CALCULA A RELAÇÃO SINAL-RUÍDO RMS-----*/
//Parâmetros:

```



```

//imagem é a imagem original.
//ImagemErro é a imagem erro.
//linhas e colunas é o números de linhas e colunas respectivamente da imagem
//erro.
double snrRMS(unsigned char *imagem, int *imagemErro, int linhas, int
colunas)
{
    int n,m,tmp;
    double snrrms = 0;
    double num = 0;
    double den = 0;

    for (n=0;n<linhas;n++){
        for (m=0; m<colunas; m++){
            tmp=n*colunas+m;
            num=num+((double)imagem[tmp])*((double)imagem[tmp]);

            den=den+((double)imagemErro[tmp])*((double)imagemErro[tmp]);
        }
    }

    if(den==0){
        snrrms = 100000;
    }else{
        snrrms = num/den;
    }

    return snrrms;
}

/*-----CALCULA A RELAÇÃO SINAL-RUÍDO DE PICO-----*/
//Parâmetros:
//p é o número de bits por pixel usado para definição do nível da componente
//de cor (vermelho, verde ou azul) considerada
//mse é o erro médio quadrático, calculado através da função mse
double psnr(int p, double mse)
{
    return 20*log10((pow(2,p)-1) / sqrt(mse));
}
*****

```

4.4.3 Aplicação das implementações em Matlab e C para comparação entre imagem ruidosa e imagem original

Nesse seção utilizou-se como imagem original o arquivo saladaCCD.bmp (figura 2) obtida através de uma câmera digital Sony, modelo P93A na resolução de VGA 640X480 pixels.

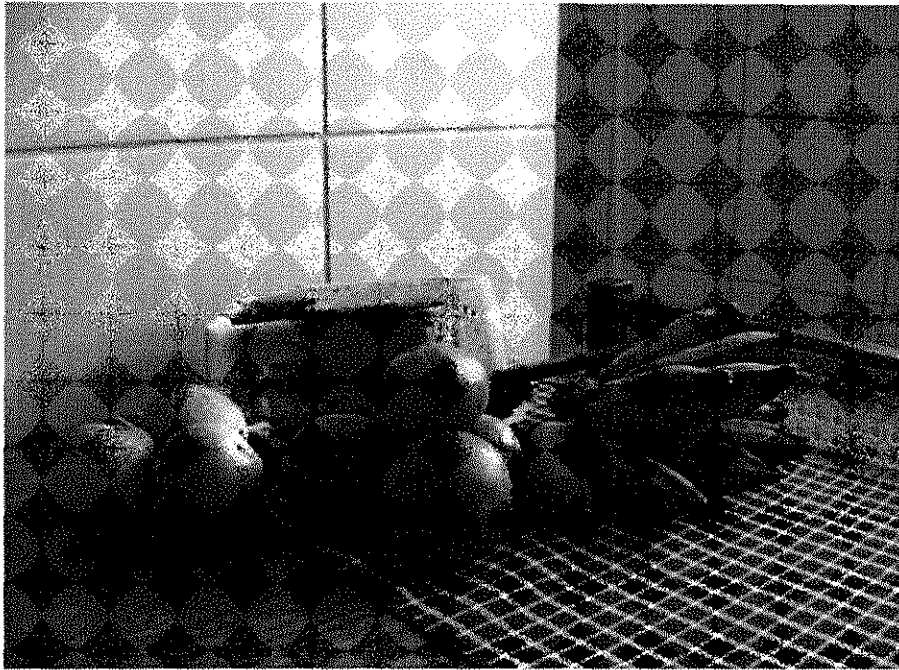


Figura 2: Imagem original

Através do programa em Matlab abaixo podem ser adicionados três tipos de ruído por vez à imagem original: gaussiano, salt and pepper, e multiplicativo. O usuário pode selecionar o tipo de ruído que deseja adicionar à imagem, assim como os parâmetros de cada tipo de ruído. A imagem resultante é salva no workspace atual do Matlab num arquivo *imagemRuidosa.bmp* para ser posteriormente filtrada através dos filtros avaliados nesse trabalho. Aplicando-se esse programa à imagem da figura 2, obtêm-se as imagens com ruído mostradas na figura 3.

```
*****
%A função adicao_de_ruido permite que o usuário escolha um dentre as
opções
%disponíveis de tipos de ruído para aplicação a uma imagem. As imagens
%original e filtrada são exibidas pelo matlab e um arquivo
imagemRuidosa.bmp
%contendo a versão da imagem original com ruído é salvo no diretório do
%workspace atual do Matlab.

%Parâmetro imagem: é um string com o nome do arquivo da imagem a que
se
%deseja adicionar ruído, por exemplo, 'saladaCCD.bmp'. Essa imagem
deve
%estar no workspace do Matlab em que se está trabalhando.
%Exemplo de uso: adicao_de_ruido('saladaCCD.bmp')
function adicao_de_ruido(imagem)
```

```

clc;%limpa a tela

%Abrindo o arquivo que contém a imagem original
RGB = imread(imagem);
I = im2double(RGB);

%Seleção do tipo de ruído
tipoDeRuido =input('\nDigite o número correspondente ao tipo de ruído que
se deseja adicionar à imagem\n1- Ruído gaussiano\n2- Ruído salt &
pepper\n3- Ruído multiplicativo\n>> ');

entradaValida = true;

%Seleção dos parâmetros de cada tipo de ruído e aplicação à imagem
%original.
switch tipoDeRuido
    case 1

        m = input('Média (Enter para valor default de 0): ');
        if isempty(m)
            m = 0;
        end

        v = input('Variância (Enter para valor default de 0.01): ');
        if isempty(v)
            v = 0.01;
        end

        J = imnoise(I,'gaussian',m,v);%Ruído gaussiano

    case 2

        d = input('Densidade d do ruído. Afeta aproximadamente
d*prod(size(imagemOriginal)) pixels\nda imagem original. (Enter para valor
default de 0.05): ');
        if isempty(d)
            d = 0.05;
        end

        J = imnoise(I,'salt & pepper',d);%Ruído salt & pepper

    case 3

        v = input('Variância (Enter para valor default de 0.04): ');
        if isempty(v)
            v = 0.04;
        end

        J = imnoise(I,'speckle',v);%Ruído multiplicativo

```

```

otherwise

    disp('Tipo de Ruído inválido');
    entradaValida = false;
end

%Exibição das imagens original e filtrada pelo Matlab
if entradaValida
    figure(1)
    imshow(I)
    title('Imagem original')

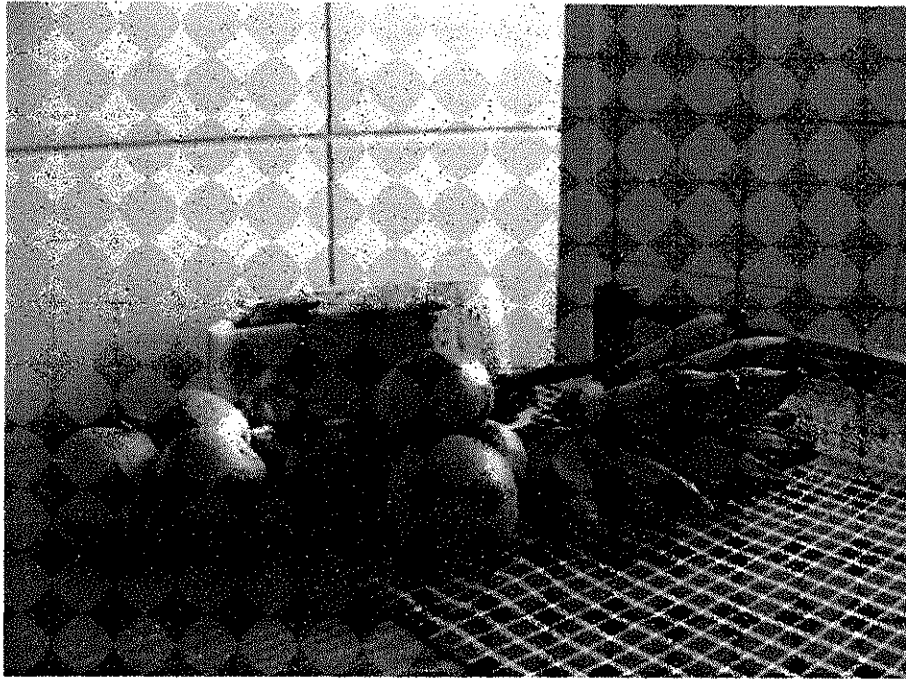
    figure(2)
    imshow(J)
    title('Imagem com ruído')

    %Salvando o arquivo com a imagem com ruído
    disp([sprintf('\n') 'A imagem com ruído foi salva no workspace atual no
arquivo imagemRuidosa.bmp.'])
    imwrite(J,'imagemRuidosa.bmp','bmp')
end

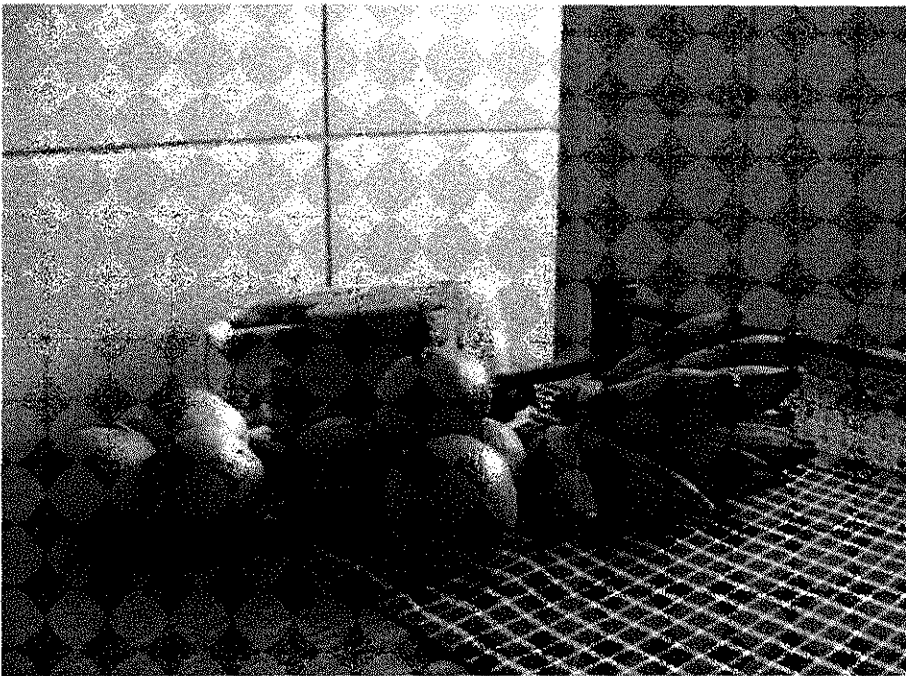
```



(a)



(b)



(c)

Figura 3: Imagens com ruído (a) Gaussiano com média 0 e variância 0,001 (b) salt and pepper com densidade de ruído 0,01 (c) multiplicativo com variância 0,01

As figuras 4, 5 e 6 contém, para cada tipo de ruído, as imagens filtradas através dos filtros da Média, Mediana e Wiener e suas respectivas imagens erro (obtidas através da comparação entre a imagem original *saladaCCD.bmp* e cada imagem filtrada) geradas através do programa em C da seção 3.4.2, e as tabelas 1, 2 e 3 mostram os

resultados numéricos das métricas de qualidade para essa implementação. Para comparar o desempenho dos filtros sob diferentes níveis de PSNR da imagem ruidosa de entrada, foi realizado um teste variando-se a variância do ruído gaussiano sob a imagem da figura 2 (o mesmo que foi adicionado à imagem da figura 3a). Foram registradas as PSNR da componente vermelha das imagens de saída de cada filtro (Média, Mediana e Wiener). Os resultados são mostrados na figura 9.

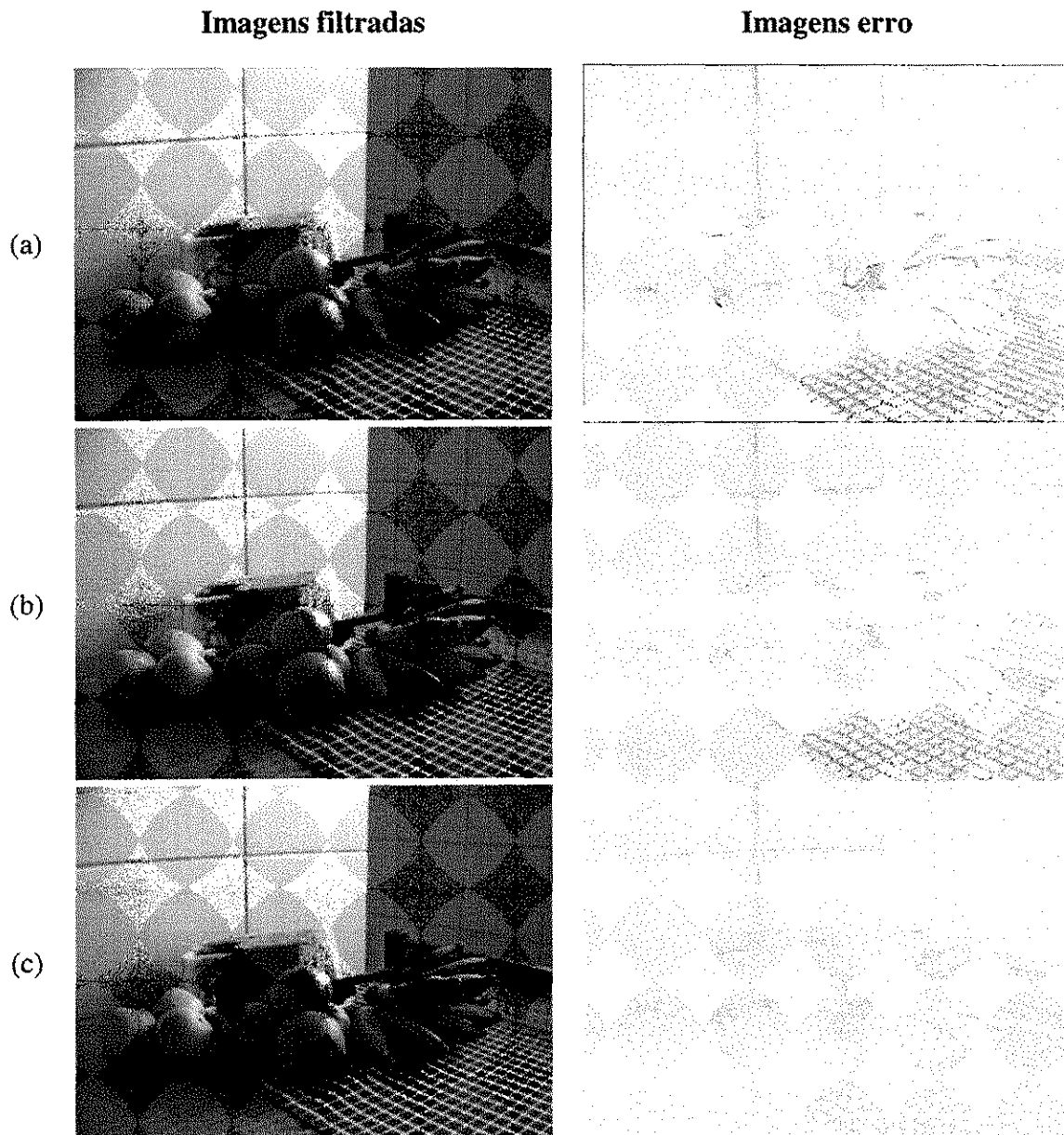


Figura 4: Imagens filtradas (resultantes da filtragem da figura 3a) e imagens erro geradas através da implementação em C (a)Média, (b)Mediana e (c)Wiener

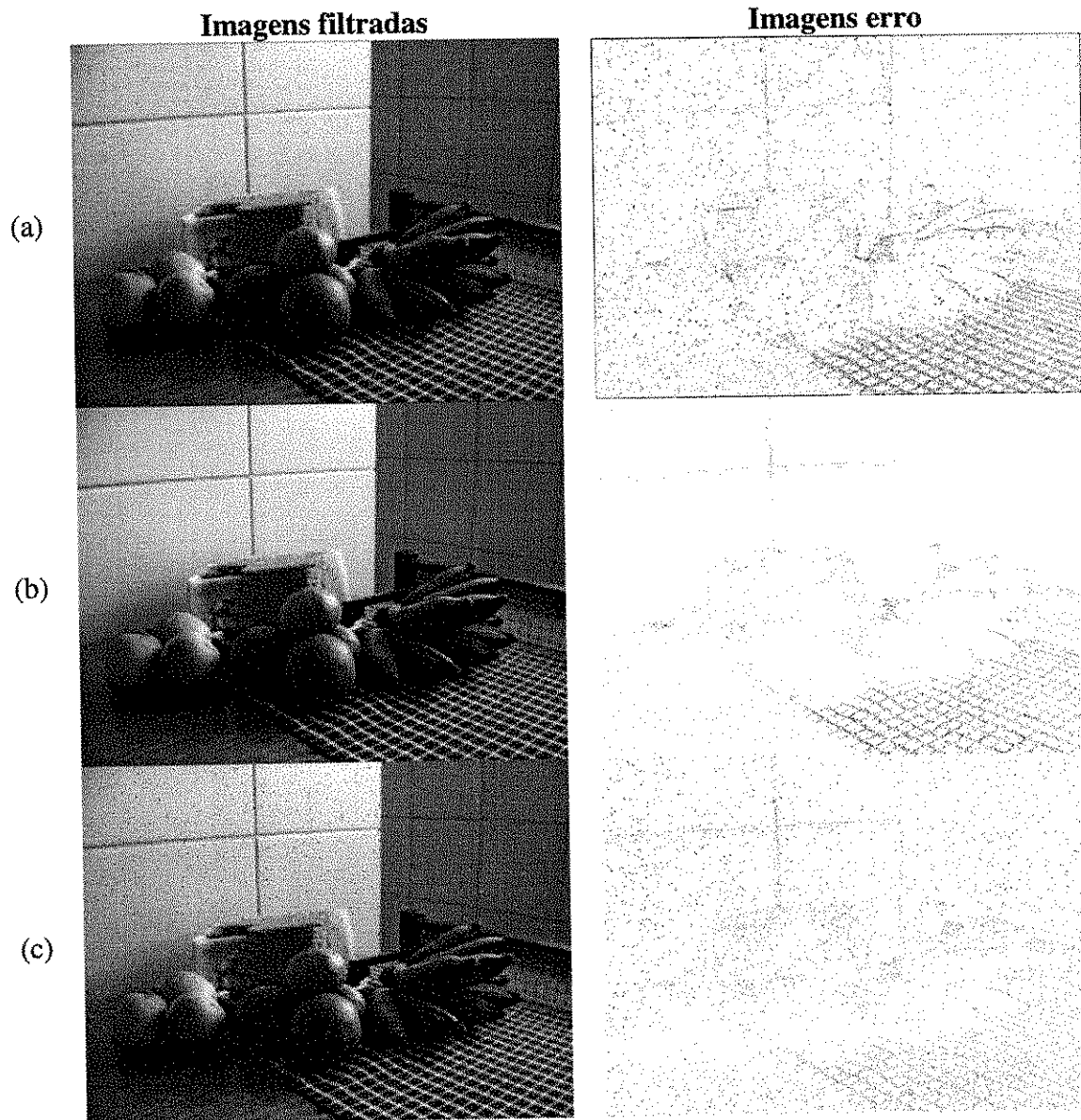


Figura 5: Imagens filtradas (resultantes da filtragem da figura 3b) e imagens erro geradas através da implementação em C (a) Média, (b) Mediana e (c) Wiener

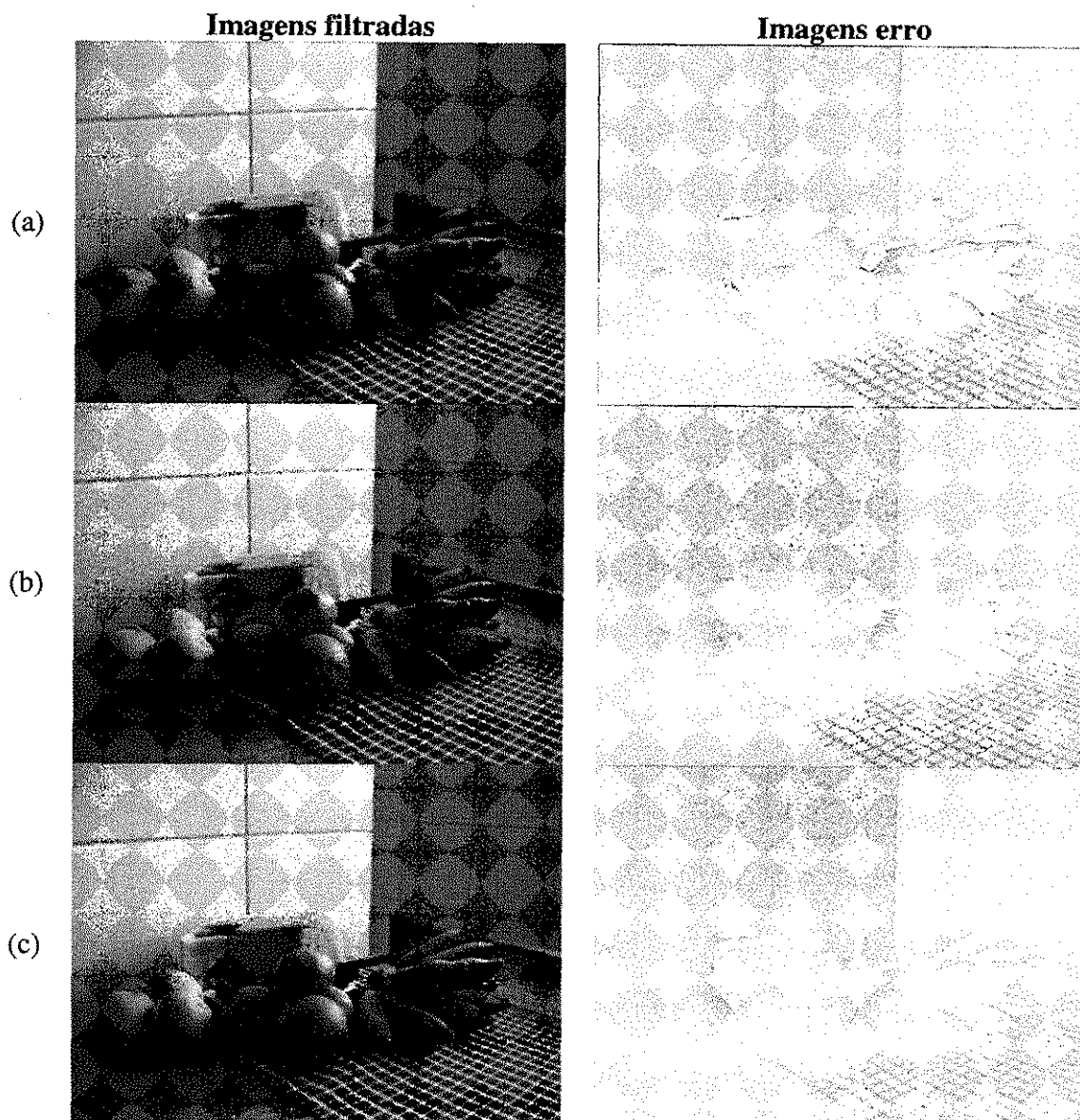


Figura 6: Imagens filtradas (resultantes da filtragem da figura 3c) e imagens erro geradas através da implementação em C (a) Média, (b) Mediana e (c) Wiener

MSE na Componente Vermelha: 16.677
MSE na Componente Verde: 17.186
MSE na Componente Azul: 16.943

SNR RMS na Componente Vermelha: 1420.577
SNR RMS na Componente Verde: 1059.420
SNR RMS na Componente Azul: 981.954

PSNR na Componente Vermelha: 35.910 dB
PSNR na Componente Verde: 35.779 dB
PSNR na Componente Azul: 35.841 dB

Tempo decorrido para calculo das metricas de desempenho: 0.291 seconds

Figura 7: Exemplo da saída padrão da implementação em C para o caso do filtro de Wiener aplicado à figura 3a.

Tabela 1: Resultados das métricas de desempenho para os resultados da filtragem da figura 3a através da implementação em C.

Métrica	Filtro		
	Média	Mediana	Wiener
MSE na Componente Vermelha	42.425	24.612	16.677
MSE na Componente Verde	46.095	28.528	17.186
MSE na Componente Azul	41.906	27.116	16.943
SNR RMS na Componente Vermelha	555.636	961.762	1420.577
SNR RMS na Componente Verde	392.247	636.950	1059.420
SNR RMS na Componente Azul	394.285	612.176	981.954
PSNR na Componente Vermelha	31.855 dB	34.219 dB	35.910 dB
PSNR na Componente Verde	31.494 dB	33.578 dB	35.779 dB
PSNR na Componente Azul	31.908 dB	33.799 dB	35.841 dB

Tabela 2: Resultados das métricas de desempenho para os resultados da filtragem da figura 3b através da implementação em C.

Métrica	Filtro		
	Média	Mediana	Wiener
MSE na Componente Vermelha	57.257	11.603	167.828
MSE na Componente Verde	62.551	14.774	171.895
MSE na Componente Azul	60.299	13.575	191.960
SNR RMS na Componente Vermelha	409.436	2035.425	141.106
SNR RMS na Componente Verde	287.809	1227.696	106.126
SNR RMS na Componente Azul	273.123	1221.006	87.059
PSNR na Componente Vermelha	30.552 dB	37.485 dB	25.882 dB
PSNR na Componente Verde	30.168 dB	36.436 dB	25.778 dB
PSNR na Componente Azul	30.328 dB	36.804 dB	25.299 dB

Tabela 3: Resultados das métricas de desempenho para os resultados da filtragem da figura 3c através da implementação em C.

Métrica	Filtro		
	Média	Mediana	Wiener
MSE na Componente Vermelha	61.778	76.203	52.085
MSE na Componente Verde	59.210	66.605	44.792
MSE na Componente Azul	53.562	61.557	46.097
SNR RMS na Componente Vermelha	380.886	310.231	454.412
SNR RMS na Componente Verde	305.014	272.207	406.206
SNR RMS na Componente Azul	307.981	268.716	360.652
PSNR na Componente Vermelha	30.222 dB	29.311 dB	30.964 dB
PSNR na Componente Verde	30.407 dB	29.896 dB	31.619 dB
PSNR na Componente Azul	30.842 dB	30.238 dB	31.494 dB

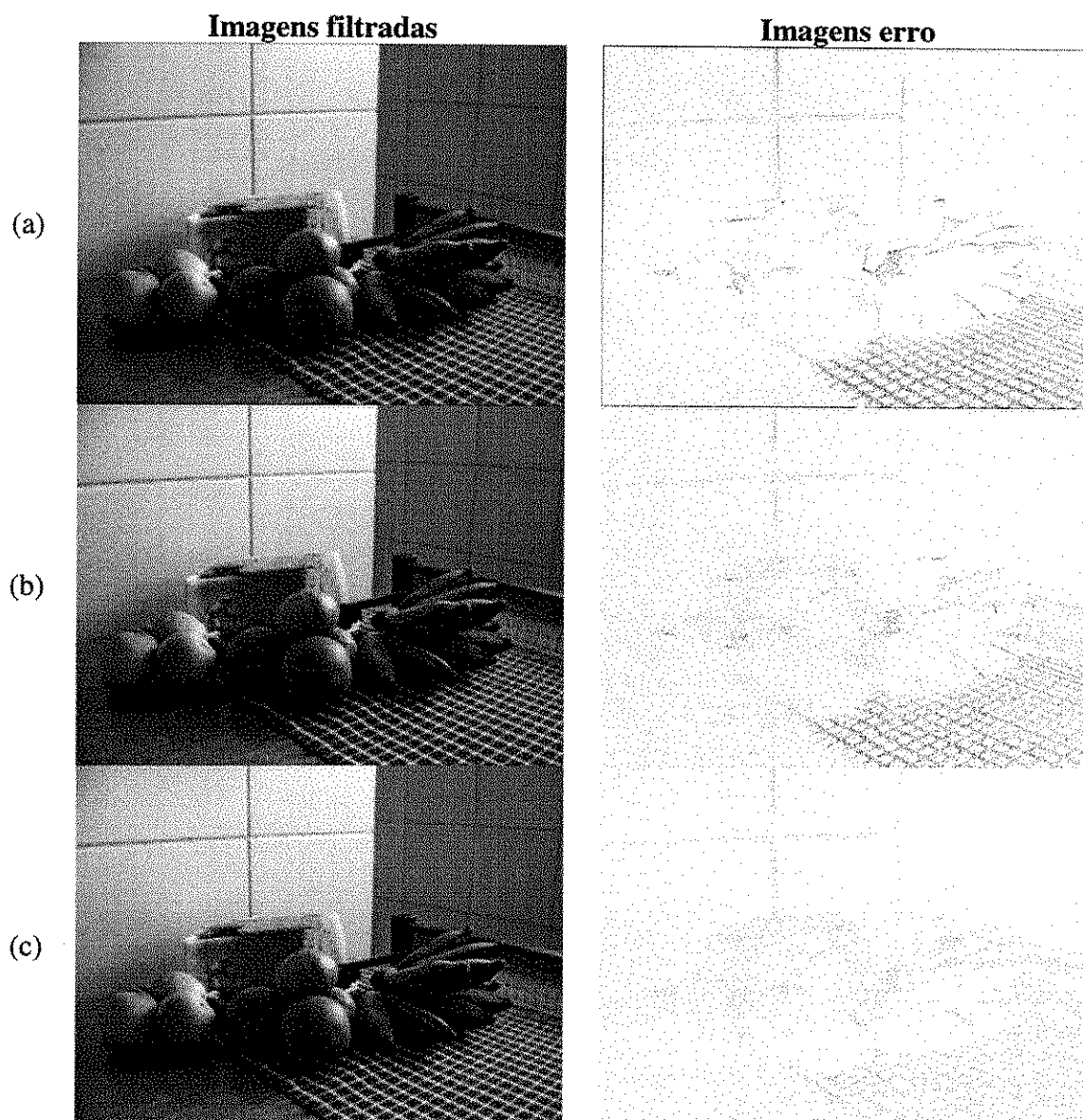


Figura 8: Imagens filtradas (resultantes da filtragem da figura 3a) e imagens erro geradas através da implementação em Matlab (a) Média, (b) Mediana e (c) Wiener

Tabela 4: Resultados das métricas de desempenho para os resultados da filtragem da figura 3a através da implementação em Matlab.

Métrica	Filtro		
	Média	Mediana	Wiener
MSE na Componente Vermelha	42.4249	24.612	16.6772
MSE na Componente Verde	46.0948	28.5285	17.186
MSE na Componente Azul	41.9059	27.1161	16.9425
SNR RMS na Componente Vermelha	555.636	961.762	1420.577
SNR RMS na Componente Verde	392.247	636.950	1059.420
SNR RMS na Componente Azul	394.285	612.176	981.954
PSNR na Componente Vermelha	31.8546 dB	34.2193 dB	35.9096 dB
PSNR na Componente Verde	31.4943 dB	33.578 dB	35.779 dB
PSNR na Componente Azul	31.9081 dB	33.7985 dB	35.841 dB

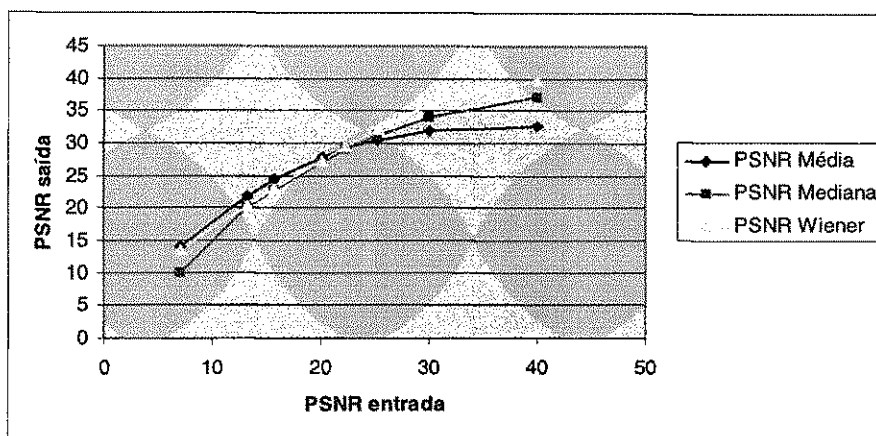


Figura 9: PSNR da componente vermelha de imagens filtradas através dos filtros da Média, Mediana e Wiener em função da PSNR da componente vermelha da imagem ruidosa de entrada.

4.4.4. Análise dos resultados

Verifica-se que os resultados das métricas de qualidade de imagem correspondentes em Matlab e C foram muito próximos, assim como as imagens erro geradas por cada implementação, o que valida os resultados, como esperado.

Comparando as imagens erro dos três filtros, verifica-se que o filtro de Wiener apresenta menor erro nas bordas dos objetos que os outros dois filtros. Os três filtros removeram satisfatoriamente o ruído presente nas texturas dos objetos.

O MSE máximo que poderia ocorrer na comparação entre duas imagens BMP ocorreria no caso extremo de os seus correspondentes pixels tivessem valor 0 numa imagem e 255 na outra, por exemplo uma imagem branca e outra preta; então o valor do MSE para cada componente seria $1/(N \times M) \times N \times M \cdot 255^2 = 65025$ para uma imagem de dimensões $N \times M$. Nota-se que os valores de MSE obtidos quando comparada a imagem da figura 2 e as imagens filtradas são cerca de um milésimo desse valor máximo, mostrando que os filtros preservaram satisfatoriamente as características da imagem original e da filtrada.

Na filtragem de imagens com ruído gaussiano, os valores do erro médio quadrático, para cada componente de cor e para cada filtro, evidenciam que o filtro de Wiener causa erros de menor magnitude que o da Mediana e que este causa menos erros que o da Média. Isto era esperado teoricamente, pois o filtro de Wiener é ótimo no

sentido de minimizar o erro médio quadrático. Quanto à relação sinal-ruído, *SNR*, e a relação sinal-ruído de pico, *PSNR*, nota-se que o filtro de Wiener é o que apresenta maior relação sinal-ruído, o que mostra sua robustez quanto à preservação do sinal face à interferência do ruído.

Na filtragem de imagens com ruído salt and pepper, os valores do erro médio quadrático, para cada componente de cor e para cada filtro, evidenciam que o filtro da Mediana causa erros de menor magnitude que o da Média e que este causa menos erros que o de Wiener. Isto era esperado teoricamente, pois o filtro da mediana apresenta bom desempenho na remoção de ruído do tipo salt and pepper, e nos filtros da Média e de Wiener os pixels desse tipo de ruído são considerados nos cálculos, enquanto no filtro da mediana são tomados apenas um ou dois valores de “pixels saudáveis”. Quanto à relação sinal-ruído, *SNR*, e a relação sinal-ruído de pico, *PSNR*, nota-se que o filtro da Mediana é o que apresenta maior relação sinal-ruído, o que mostra sua robustez quanto à preservação do sinal face à interferência do ruído salt and pepper.

Na filtragem de imagens com ruído multiplicativo, os valores do erro médio quadrático, para cada componente de cor e para cada filtro, evidenciam que o filtro de Wiener causa erros de menor magnitude que o da Média e que este causa menos erros que o da Mediana. Quanto à relação sinal-ruído, *SNR*, e a relação sinal-ruído de pico, *PSNR*, nota-se que o filtro de Wiener é o que apresenta maior relação sinal-ruído, o que mostra sua robustez quanto à preservação do sinal face à interferência do ruído multiplicativo.

Em relação à avaliação da *PSNR* da componente vermelha das imagens filtradas em função da *PSNR* da componente vermelha da imagem ruidosa de entrada, cujos resultados são mostrados na figura 9, verifica-se que abaixo de um certo nível na entrada, aproximadamente 23dB no caso do experimento, o filtro da Média apresentou resultados melhores que os filtros de Wiener e o da Mediana, e acima desse mesmo nível o filtro de Wiener apresentou melhores resultados que os demais.

5. Avaliações Subjetivas

Possivelmente a melhor forma de se avaliar a qualidade de uma imagem é simplesmente olhar para ela, devido ao fato de os olhos humanos serem o receptor da informação na maioria dos ambientes de processamento de imagem. A avaliação subjetiva MOS (do inglês *mean opinion score*) têm sido amplamente utilizada por muitos anos. Ela é contudo inconveniente, lenta e cara para o uso prático.

Nessa seção introduziremos dois métodos de avaliação subjetiva da qualidade de imagens: o Método do Estímulo Único e o Método de Estímulo-Comparação, descritos na 6ª seção da Recomendação ITU-R BT.500-11 (*Methodology for the subjective assessment of the quality of television pictures*). Mostraremos como foi realizada uma pesquisa de opinião utilizando esses dois métodos através da Internet.

5.1. Métodos de Estímulo Único

Os Métodos de Estímulo Único (Single-stimulus (SS) methods) são descritos na seção 6.1 da ITU-R BT.500-11 (pág. 18). Nesses métodos, é apresentada uma única imagem ou uma seqüência de imagens, e o observador atribui um índice para a apresentação inteira.

Como opção para a escala de avaliação, utilizou-se o Método de Estímulo Único do tipo Julgamento de Categoria por Adjetivo (Adjectival categorical judgement method), descrito na seção 6.1.4.1 da ITU-R BT.500-11. O observador associa um adjetivo dentre um conjunto de categorias à imagem ou seqüência de imagens. A tabela 5 apresenta a escala utilizada nesse trabalho.

Tabela 5: Escala de qualidade ITU-R

5	Excelente
4	Bom
3	Regular
2	Ruim
1	Péssima

5.2. Métodos de Estímulo-Comparação

Os Métodos de Estímulo-Comparação (Stimulus-comparison methods) são descritos na seção 6.2 da ITU-R BT.500-11 (pág. 20). Nesses métodos, são apresentadas duas imagens ou seqüências de imagens, e o observador oferece uma nota da relação entre as duas apresentações.

Como opção para a escala de avaliação, utilizou-se o Método de Estímulo-Comparação do tipo Julgamento de Categoria por Adjetivo (Adjectival categorical judgement method), descrito na seção 6.2.4.1 da ITU-R BT.500-11. Os observadores associam a relação entre membros de um par a um adjetivo de um conjunto de categorias, que podem fazer menção à existência de diferenças perceptíveis (p.ex. INDIFERENTE, DIFERENTE), a existência e direção de diferenças perceptíveis (p.ex.

MENOS, INDIFERENTE, MAIS), ou julgamentos de extensão e direção. A tabela 6 apresenta a escala utilizada nesse trabalho.

Tabela 6: Escala de comparação

+3	Bem melhor
+2	Melhor
+1	Levemente melhor
0	Indiferente
-1	Levemente pior
-2	Pior
-3	Muito pior

5.3. Realização de uma pesquisa de opinião

A Recomendação ITU-R BT.500-11 expõe em sua 2ª seção condições gerais para avaliações subjetivas. Com o objetivo de realizar um pesquisa de opinião com baixo custo operacional e com resultados satisfatórios para a aplicação em estudo (Filtros Digitais Bi-Dimensionais) utilizou-se a Recomendação ITU-R BT.500-11 apenas como referência, não impondo à pesquisa de opinião realizada as mesmas condições de teste prescritas pela recomendação citada, mas aproveitando dela os elementos de interesse para realizar essa pesquisa.

A proposta de realizar a pesquisa de opinião através da Internet deve-se à vantagem da não obrigatoriedade de realização das entrevistas num local, horário e grupo de entrevistados específico, o que demandaria a alocação de recursos físicos (sala de entrevista e computadores) e pessoal (entrevistadores e voluntários).

A pesquisa foi realizada por meio de uma homepage que permitiu a coleta dos dados inseridos pelos entrevistados. Para isso foi necessário o uso do conceito de Formulário na construção das páginas html dedicadas à entrevista.

O formulário consiste de uma apresentação do teste a que o entrevistado será submetido. O entrevistado insere os dados através de botões rádio e caixas de texto e realiza o envio dos dados inseridos para o e-mail pessoal do entrevistador clicando no botão de submissão.

Um modelo para a construção do formulário foi obtido no site <http://www.response-o-matic.com/template.htm>. Este site oferece além do serviço de criação de um modelo para o formulário, a inserção no mesmo de comandos em html que permitem o envio das respostas de um dado entrevistado via o site www.response-o-matic.com para o e-mail fornecido pelo entrevistador.

5.3.1 Página HTML para a pesquisa de opinião

A página web abaixo foi utilizada para colher a opinião de usuários da Internet sobre a qualidade na filtragem das imagens expostas na página.

No método de estímulo-comparação foi utilizada uma imagem adquirida através de uma webcam Genius, modelo VideoCaM NB na resolução de 640X480 pixels e suas versões filtradas através dos filtros considerados nesse trabalho.

No método de estímulo-único, foram utilizadas versões filtradas de uma imagem adquirida com uma câmera digital Olympus (sensor de imagem CCD) referência u-miniD, Stylus V à qual foi adicionado ruído salt and pepper.



**Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Departamento de Engenharia Elétrica**

Pesquisa de opinião para a Disciplina Projeto em Engenharia Elétrica

Título: Aplicação de Métricas de Desempenho a Filtros Digitais Bi-Dimensionais

Aluno: Saulo Oliveira Dornellas Luiz

Orientador: Angelo Perkusich

Método de Estímulo-comparação

Você participará de uma Avaliação Subjetiva utilizando o Método de Estímulo-comparação para avaliar a qualidade de imagens filtradas. Nos pares de imagens abaixo, a do lado esquerdo é a original e a do lado direito é a filtrada. Para cada par, indique segundo um dos sete níveis da escala como ficou a imagem após a filtragem.

Filtro da Média

Imagem ruidosa

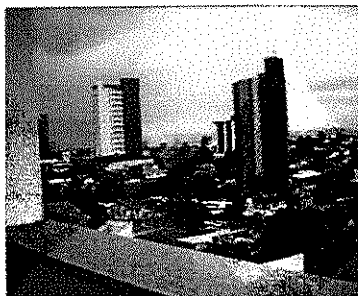


Imagem filtrada



Em sua opinião, a imagem ficou...

- Bem melhor
- Melhor
- Levemente melhor
- Indiferente
- Levemente pior
- Pior
- Muito Pior

Filtro da mediana

Imagem ruidosa



Imagem filtrada



Em sua opinião, a imagem ficou...

- Bem melhor
- Melhor
- Levemente melhor
- Indiferente
- Levemente pior
- Pior
- Muito Pior

Filtro de Wiener

Imagem ruidosa

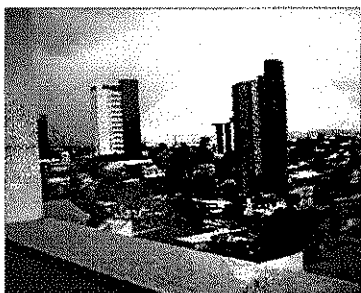


Imagem filtrada



Em sua opinião, a imagem ficou...

- Bem melhor
- Melhor
- Levemente melhor
- Indiferente
- Levemente pior
- Pior
- Muito Pior

Método de Estímulo-Único

Você participará de uma Avaliação Subjetiva utilizando o Método de Estímulo-Único para avaliar a qualidade de imagens filtradas através dos filtros da Média, da Mediana e de Wiener. Para cada uma das imagens, classifique a qualidade da filtragem segundo um dos cinco níveis da escala ao lado da imagem.

Filtro da Média



Em sua opinião, a qualidade da imagem é...

- Excelente
- Bom
- Regular
- Ruim
- Péssima

Filtro da Mediana



Em sua opinião, a qualidade da imagem é...

- Excelente
- Bom
- Regular
- Ruim
- Péssima

Filtro de Wiener



Em sua opinião, a qualidade da imagem é...

- Excelente
- Bom
- Regular
- Ruim
- Péssima

Para finalizar a pesquisa de opinião:

Enviar dados

5.3.2 Resultados da pesquisa de opinião

A recomendação ITU-R BT.500-11, em sua 2ª seção determina a participação de no mínimo 15 pessoas em qualquer das pesquisas de opinião descritas por essa recomendação. A pesquisa de opinião realizada nesse trabalho contou com a participação de 40 pessoas.

Os resultados do método de estímulo-comparação são apresentados na tabela 7. Cada voto de um participante é convertido de adjetivo para peso numérico segundo a tabela 6, e é tomada a média de todos os votos. A última linha da tabela 7 contém as médias obtidas para cada filtro e evidencia-se que o filtro da média obteve peso entre levemente pior e pior, o filtro da mediana obteve média entre indiferente e levemente pior, e o filtro de wiener, com melhor resultado, obteve média entre indiferente e levemente melhor. Assim a imagem filtrada pelo filtro de wiener “pareceu melhor aos olhos do participante” em média.

Tabela 7: Resultados da pesquisa de opinião para o método de estímulo-comparação

Filtro da Média	Filtro da mediana	Filtro de Wiener
-1.1	-0.65	0.375

A tabela 8 contém os resultados para método de estímulo-único convertidos de adjetivos para pesos numéricos segundo a tabela 5. As imagens utilizadas nesse método foram resultados da filtragem de uma imagem com ruído do tipo salt and pepper. É sabido que o filtro da mediana é altamente eficiente na remoção deste tipo de ruído. A última linha contém as médias obtidas para cada filtro e evidencia-se que o filtro da média obteve peso entre ruim e regular, o filtro da mediana, com melhor resultado, obteve média entre bom e excelente, e o filtro de wiener obteve média entre ruim e regular. Assim, devido ao fato de se tratar da filtragem de uma imagem com ruído salt and pepper, a imagem filtrada pelo filtro da mediana “pareceu melhor aos olhos do participante”.

Tabela 8: Resultados da pesquisa de opinião para o método de estímulo-único

Filtro da Média	Filtro da mediana	Filtro de Wiener
2.85	4.3	2.45

6. Conclusões

Foi analisado neste trabalho o processo de restauração de imagens exemplificado por filtros digitais bidimensionais e foram aplicadas métricas de desempenho aos resultados desses filtros sob a forma de métricas de qualidade de imagem e avaliações subjetivas. Deduziu-se o filtro de Wiener adaptativo e foram apresentados os da média e da mediana, destacando-se em cada um suas vantagens e desvantagens com base em seus aspectos construtivos.

A aplicação de métricas de qualidade de imagem aos resultados dos filtros mostrou que o filtro de Wiener ofereceu menor erro médio quadrático, o que está de acordo com o fato de este ser projetado para ser ótimo para minimização do erro médio quadrático.

A realização de uma avaliação subjetiva, apesar de inconveniente e lenta, foi necessária, já que as métricas objetivas geralmente não são bem correlacionadas com os resultados de avaliações subjetivas. Esta avaliação verificou que, para o observador humano, uma imagem com ruído salt and pepper filtrada através do filtro da mediana apresenta maior qualidade que imagens filtradas com os demais filtros deste trabalho, e que uma imagem captada através de uma webcam teve, dentre suas versões filtradas, o melhor resultado oferecido pelo filtro de Wiener.

A tabela 9 mostra os resultados previamente mostrados e permite uma comparação entre os resultados das métricas de qualidade de imagem e as avaliações subjetivas.

Tabela 9: Tabela resumo dos resultados das métricas de desempenho

		Filtro da Média	Filtro da Mediana	Filtro de Wiener
Métricas de qualidade de imagem	Gaussiano Branco	O menos indicado; maiores MSE, e menores SNR RMS e PNSR.	Resultados intermediários entre os filtros da Média e de Wiener.	O mais indicado; menores MSE, e maiores SNR RMS e PNSR.
	Salt and Pepper	Resultados intermediários entre os filtros da Mediana e de Wiener.	O mais indicado; menores MSE, e maiores SNR RMS e PNSR.	O menos indicado; maiores MSE, e menores SNR RMS e PNSR.
Avaliações subjetivas	Gaussiano Branco (Método de Estímulo-Comparação)	O menos indicado; entre pior e levemente pior.	Resultado intermediário entre os filtros da Média e de Wiener; entre levemente pior e indiferente.	O mais indicado; entre indiferente e levemente melhor.
	Salt and Pepper (método de estímulo-único)	Resultado intermediário entre os filtros da Média e de Wiener; entre ruim e regular.	O mais indicado; entre bom e excelente.	O menos indicado; entre ruim e regular.

7. Referências Bibliográficas

- [1] VASEGHI, S. V. **Advanced Digital Signal Processing and Noise Reduction**. 2. ed. John Wiley & Sons Ltd, 2000.
- [2] ÅSTRÖM, K. J., WITTENMARK, B. **Computer-Controlled Systems: theory and design**. 3. ed. New Jersey: Prentice Hall, 1997.
- [3] VINIOTIS, Y. **Probability and Random Processes for Electrical Engineers**. 1. ed. Singapore: McGraw-Hill, 1998.
- [4] CONCI, A., AQUINO, F. R. **Fractal coding based on image local fractal dimension**. *Computational & Applied Mathematics*, Petrópolis, RJ, v.24, n.1, p. 83-98, 2005.
- [5] Recomendação ITU-R BT.500-11, “**Methodology for the subjective assessment of the quality of television pictures.**” International Telecommunication Union, Geneva, Switzerland, 2002.
- [6] CHOY, S. S. O., CHAN, Y., SIU, W. **An improved quantitative measure of image restoration quality**. *Acoustics, Speech, and Signal Processing*, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on Volume 3, 7-10 May 1996 Page(s):1613 - 1616 vol. 3.
- [7] RUSS, J. C. **The Image Processing Handbook**. 3. ed. Singapore: CRC Press, 1998.
- [8] ESKICIOGLU, A. M., FISHER, P. S. **Image quality measures and their performance**. *IEEE Trans. Communications*, v. 43, p. 2959–2965, dec. 1995.
- [9] WANG, Z., SIMONCELLI, E. P., BOVIK, A. C. **Multi-scale structural similarity for image quality assessment**. *Proceedings of the 37th IEEE Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, nov. 2003.
- [10] KUMAR, S. **An Introduction to Image Compression**, <http://www.debugmode.com/imagecmp> (outubro de 2005).
- [11] WANG, Z., BOVIK, A. C., LU, L. **Why is image quality assessment so difficult?** *Acoustics, Speech, and Signal Processing*, 2002. Proceedings. (ICASSP '02). IEEE International Conference on Volume 4, 13-17 May 2002 Page(s):IV-3313 - IV-3316 vol.4
- [12] ZOU, J. **A study of Wiener filter and its extensions with experiments**. ECSE 6630 Digital Image and Video Processing final project report.
- [13] VANDEVENNE, L. **Image Filtering**. <http://www.student.kuleuven.ac.be/%7Em0216922/CG/index.html> (outubro de 2005).
- [14] WANG, Z., BOVIK, A. C., SHEIKH, H. R., SIMONCELLI, E. P. **Image Quality Assessment: From Error Visibility to Structural Similarity**. *IEEE Transactions on Image Processing*, v. 13, no. 4, p. 600-612, Apr. 2004.
- [15] KUO, S. M., LEE, B. H. **Real-Time Digital Signal Processing: Implementations, Applications and Experiments with the TMS320C55X**. John Wiley & Sons, LTD., 2001.

[16] **An interactive image processing course.** <http://www.ph.tn.tudelft.nl/Courses/FIP/> (outubro de 2005).