UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ELETRÔNICA E DE COMPUTAÇÃO

FILTRO SIGMA COM DETEÇÃO DE BORDAS PARA CANCELAMENTO DE RUIDO GAUSSIANO EM IMAGENS PRETO E BRANCO

Autora: Maria Moura Malburg

Orientador: Gelson Vieira Mendonça

Examinador: Mariane Rembold Petraglia

Examinador: Mauros Campello Queiroz

DEL

Novembro de 2005

Dedicatória

Aos meus pais, que tanto me ajudaram e me apoiaram ao longo dos anos de faculdade, que me mostraram o quanto era importante persistir neste curso, apesar de todas as dificuldades, que me deram absolutamente tudo o que eu precisei, para que conseguisse concluir a faculdade da melhor forma possível. Sem eles, certamente não teria conseguido.

Agradecimentos

Agradeço principalmente aos meus pais, por tudo o que me proporcionaram ao longo da faculdade, mas também aos meus irmãos, avós, tios e primos por terem me escutado e me consolado nos meus momentos de fraqueza, assim como aos amigos que fiz na faculdade, que tanto me ajudaram a enfrentar as inúmeras dificuldades acadêmicas do curso de engenharia eletrônica. Agradeço ainda aos meus outros amigos, pela simples amizade e companhia, pela paciência, e pelas praias e programações perdidas, quando tive que me dedicar à faculdade.

Resumo

O objetivo deste trabalho consiste no desenvolvimento de um método para remoção de ruído gaussiano e restauração de imagens preto e branco que apresente um desempenho superior aos métodos já conhecidos, principalmente no que se refere à conservação e restauração das bordas da imagem.

Para isso, foi desenvolvida no Matlab uma arquitetura de filtro de imagens em preto e branco que associa diversas técnicas de remoção de ruído e de restauração de imagens já conhecidas. Este filtro é baseado no filtro sigma de Lee, o qual faz a restauração de cada pixel da imagem a partir de uma análise estatística dos pixels que estão ao seu redor, e da seleção somente daqueles pixels que aparentemente pertencem à mesma classe que o pixel que está sendo restaurado, para serem considerados no cálculo da sua estimativa.

O filtro sigma é ainda associado a uma rotina de detecção de bordas da imagem, que se propõe a identificar aqueles pixels que compõem as bordas da imagem. Desta forma, somente os pixels que não pertencem às bordas da imagem são restaurados pelo algoritmo do filtro sigma, enquanto os pixels pertencentes às bordas são restaurados pela aplicação de máscaras de filtragem específicas para cada tipo de borda e destinadas a proporcionar uma melhor conservação das bordas da imagem.

Palavras chave

Imagem; filtro sigma; bordas; máscara de filtragem; ruído gaussiano.

Índice

Capítulo 1
Introdução

Existem diversas técnicas de processamentos de imagens digitais que se destinam basicamente a manipular imagens digitais por meio de operações matemáticas a fim de provocar algum efeito desejado nesta imagem. Um tipo de técnica de processamento de imagens consiste na compressão de imagens digitais, para reduzir a quantidade de memória necessária para seu armazenamento, ao mesmo tempo mantendo uma boa qualidade destas imagens.

Outra técnica de processamento de imagens bastante comum consiste na remoção de algum tipo de artefato indesejado da imagem, tal como ruído. Quando uma imagem é corrompida por ruído, diversas das suas informações são perdidas, fazendo com que ela fique muito diferente da imagem original. As técnicas de remoção de ruído se propõem a remover o efeito provocado pelo ruído sobre a imagem, de modo a restaurá-la, melhorando sua qualidade, para que ela fique similar à imagem original. Estas técnicas de processamento de imagem podem ser aplicadas, por exemplo, a imagens obtidas em exames médicos que capturam imagens no interior do corpo humano e, portanto, podem estar sujeitas a ruído e condições desfavoráveis à sua captação. Estas imagens podem ter sua qualidade melhorada através de algumas destas técnicas de processamento, a fim de aumentar a eficiência de diagnósticos com base nas mesmas.

Outro campo de aplicação para técnicas de remoção de ruído e restauração de imagem está direcionado à transmissão de imagens por canais de transmissão muito sujeitos a ruído, por exemplo, transmissão de imagens de televisão pelo ar. Neste caso, a imagem recebida pelo receptor pode ser fortemente corrompida por ruído e apresentar qualidade muito baixa. Por isso, é necessário que esta imagem seja processada, para que sua qualidade seja melhorada, antes de ser exibida no receptor.

Os métodos de processamento para remoção de ruído e restauração da imagem já conhecidos mais comuns são o filtro de média e o filtro de mediana, que buscam basicamente restaurar o valor de cada pixel da imagem com base nos valores dos seus pixels

imediatamente adjacentes. Estes filtros conseguem conservar somente algumas características da imagem, porém de um modo geral, produzem uma imagem restaurada com baixa nitidez.

Um outro método de remoção de ruído e restauração de imagens se baseia na análise estatística adaptativa dos pixels mais próximos de um dado pixel da imagem e utilizam para a restauração deste somente aqueles pixels que são mais similares ao pixel a ser restaurado [1]. Este método apresenta um desempenho superior ao dos filtros médio e mediano, pois produz uma imagem mais nítida, com uma maior quantidade de detalhes regenerados e, portanto, mais próxima da imagem original. Uma arquitetura de filtro já conhecida que emprega este tipo de processamento de imagem é o filtro sigma desenvolvido por Lee [2].

Uma vez que os métodos de remoção de ruído e restauração de imagens que se baseiam na análise estatística apresentam um bom desempenho, o objetivo deste trabalho consiste em desenvolver um algoritmo para restauração de imagem digital preto e branco corrompida por ruído gaussiano que seja baseado na arquitetura de filtro sigma, porém que produza uma imagem restaurada com maior nitidez e melhor qualidade do que aquela produzida pelo filtro sigma puro. Para isso, será usado um algoritmo de um filtro sigma, associado a um algoritmo de detecção de bordas e de restauração das regiões de borda. O algoritmo e os testes foram desenvolvidos na plataforma MATLAB.

Neste relatório, inicialmente serão descritas as arquiteturas de filtro para restauração de imagem já conhecidas da literatura, inclusive do próprio filtro sigma comum. Em seguida, serão apresentados os conceitos básicos de detecção e conservação de bordas de imagens em preto e branco que foram pesquisados e selecionados para serem implementados em combinação com o filtro sigma comum. A arquitetura do filtro desenvolvido neste trabalho será, depois, explicada detalhadamente. Em seguida, serão apresentados diversos resultados de testes do filtro desenvolvido, os quais serão comparados aos resultados do filtro sigma comum. Por fim, serão expostas as conclusões deste trabalho com base na análise dos resultados apresentados, em relação ao desempenho do filtro desenvolvido, bem como as vantagens e desvantagens que esta arquitetura apresenta em relação aos filtros já conhecidos da literatura. Os códigos fonte das principais funções desenvolvidas neste projeto serão anexados nos apêndices deste relatório.

Capítulo 2

Métodos de restauração de imagens

2.1) Conceitos fundamentais

2.1.1) A imagem digital:

Uma imagem digital em preto em branco é representada por meio de uma matriz bidimensional em que cada elemento possui um valor correspondente a um pixel da imagem. Um pixel é o menor elemento de representação da imagem, de modo que cada pixel corresponde a um ponto da imagem. A posição de cada pixel na imagem corresponde à posição do elemento que o representa dentro da matriz. Cada pixel apresenta um valor de intensidade que representa um tom correspondente a um nível de cinza. A quantidade de níveis de cinza que uma imagem pode apresentar depende do formato de representação usado para a imagem e, principalmente, da quantidade de bits usados para a representação de cada pixel. A seguir, na matriz (2.1) é mostrado um exemplo de um trecho de uma matriz de representação da imagem, bem como a correspondente à matriz na figura (2.1).

73	56	45	36	34	30	69	123	127	132	122	106	110	89	78	82	61
62	62	76	76	55	63	90	94	104	105	116	100	76	44	39	32	35
70	98	102	91	52	45	63	64	59	100	90	53	50	49	31	30	79
75	84	57	51	31	38	41	39	66	84	74	80	57	46	63	82	56
41	33	24	28	26	24	27	46	74	75	110	65	78	68	101	55	87
44	42	25	25	30	26	40	80	70	108	63	58	30	54	61	75	85
44	28	19	18	15	28	72	94	84	78	47	15	20	36	80	49	50
25	25	15	26	13	55	102	89	86	47	16	9	20	87	66	40	69
26	20	14	18	18	83	83	76	46	19	21	21	83	88	38	55	60
30	9	14	19	55	104	42	48	22	13	28	83	99	65	60	78	15
13	8	11	37	69	68	21	25	24	21	66	76	48	89	90	34	11
6	8	22	58	55	48	15	18	18	32	75	38	63	74	45	9	58
10	11	36	68	45	21	13	20	35	37	41	22	41	76	16	14	123
9	14	39	69	44	13	22	27	20	26	19	16	59	69	7	28	122
15	21	38	42	42	19	20	32	19	22	17	13	66	55	12	69	104
21	65	17	22	21	25	29	53	58	58	58	33	84	52	37	79	77
36	47	13	19	20	19	28	22	30	33	24	20	61	37	40	61	61
31	18	18	16	15	16	19	23	19	16	12	11	41	38	66	55	43
33	10	16	22	17	20	19	27	16	17	13	13	50	77	72	54	53

Fig. (2.1) Imagem gerada

Matriz 2.1: Extrato da matriz de imagem de Lena.gif

Em uma imagem que usa um número **n** de bits para representar o valor de intensidade de cada pixel, a quantidade de níveis de cinza possíveis para esta imagem é igual a $\mathbf{2^n}$, que corresponde à quantidade de valores diferentes que podem ser representados por **n** bits. Ou seja, se cada pixel da imagem é representado por 4 bits, então esta imagem poderá apresentar 16 níveis de cinza diferentes.

A matriz de intensidade da imagem pode também apresentar diferentes dimensões, o que significa que a imagem pode ser representada por diversas quantidades de pixels diferentes. Quanto maior for a quantidade de pixels, melhor é a definição da imagem, pois a quantidade de pixels usada para representar cada detalhe da imagem é maior, ou, em outras palavras, cada pixel representa uma região menor da imagem. Assim, uma imagem representada por uma matriz maior possuirá mais detalhes do que a mesma imagem representada por uma quantidade de pixels menor.

No caso particular deste trabalho, as imagens utilizadas para testes são representadas por matrizes de dimensões 256x256 pixels, o que significa que elas são representadas por 65536 pixels. Estas imagens serão processadas no formato uint8 do Matlab, que utiliza 8 bits para representar cada pixel da imagem. Desta forma, as imagens apresentarão 2^8=256 níveis de cinza, de modo que cada pixel será representado na matriz de intensidade por um valor inteiro entre 0 e 255.

2.1.2) O ruído gaussiano e a distribuição gaussiana

O ruído, de um modo geral, pode ser definido como qualquer perturbação indesejada que realiza alguma interferência em um sinal desejado. Normalmente, o ruído é gerado por acoplamentos eletrostáticos ou eletromagnéticos entre o sistema que sofre a sua interferência e alguma fonte externa, tal como transmissores de rádio, linhas de transmissão ou algum outro equipamento elétrico localizado, em geral, nas suas imediações.

Existem diversos tipos de ruído com características diferentes, porém todos eles apresentam a propriedade comum de serem um sinal aleatório. Uma vez que o ruído compreende componentes em diversas freqüências, com valores de amplitude e fase também

aleatórias, não é possível realizar qualquer previsão quanto aos seus valores de freqüência, fase ou amplitude em um dado instante futuro. No entanto, após ser analisado ao longo de um intervalo de tempo consideravelmente longo, é possível calcular seu valor médio e seu valor *rms* ao longo deste intervalo, bem como avaliar sua aleatoriedade.

O ruído gaussiano apresenta uma distribuição normal ou gaussiana dos seus valores de amplitude ao longo do tempo [3]. Sua distribuição está centrada sobre o valor médio μ das amplitudes medidas ao longo de um intervalo de tempo e seu desvio padrão σ corresponde à raiz quadrada da variância destes valores de amplitudes. A distribuição gaussiana é representada pela equação a seguir [4].

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[\frac{-(x-\mu)^2}{2\sigma^2}\right] \qquad (2.1.1)$$

Pela análise desta distribuição, podemos prever a aleatoriedade do ruído. Em geral, pode-se dizer por aproximação que o ruído elétrico que apresenta uma distribuição gaussiana está contido em uma faixa de valores de amplitude igual à média μ mais ou menos 3 vezes o desvio padrão $(\mu \pm 3\sigma)$, o que significa dizer que 99,7% deste ruído está contido na faixa $(\mu \pm 3\sigma)$. No entanto, uma boa aproximação para a avaliação de resultados em distribuição gaussiana é afirmar que praticamente todos eles estão compreendidos em uma faixa de $(\mu \pm 2\sigma)$, uma vez que esta faixa corresponde a cerca de 95% dos resultados da distribuição gaussiana [1] [5] [6].

Como a finalidade deste trabalho consiste na remoção de um ruído aditivo de uma imagem em preto e branco por meio de uma filtragem, para fins de teste do desempenho do filtro desenvolvido, será adicionado à imagem um ruído gaussiano.

2.1.3) Os efeitos da adição de ruído sobre a imagem

Quando o ruído gaussiano é adicionado à imagem, esta adição de ruído provoca a modificação dos valores de intensidade dos pixels da imagem, ou seja, os valores dos pixels são modificados em relação ao seu valor original, de acordo com a equação (2.1.2), sendo que

y(i,j) representa os valores de intensidade de cada pixel da imagem ruidosa, x(i,j) corresponde aos valores de intensidade de cada pixel da imagem original e n(i,j) representa o ruído que foi adicionado à imagem original [7].

$$y(i,j) = x(i,j) + n(i,j) \qquad (2.1.2)$$

A seguir, serão apresentadas algumas imagens testadas ao longo deste trabalho antes e após serem corrompidas por ruído, a fim de ilustrar os efeitos provocados pela inserção de ruído com variância = 0.05 na imagem.

Figura (2.2)

Figura (2.3)

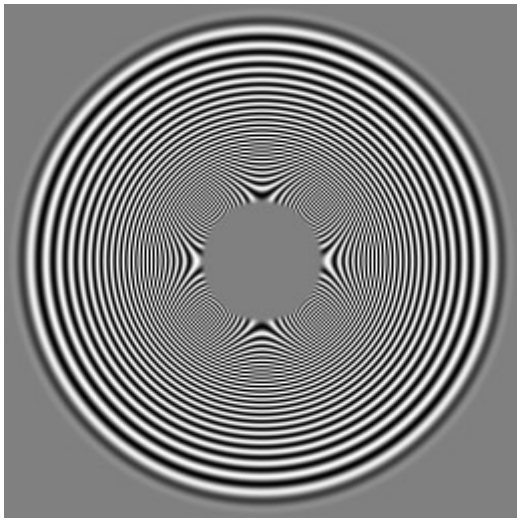


Figura (2.4)

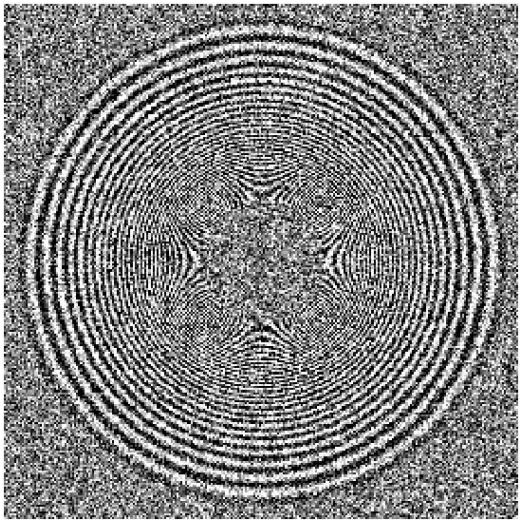


Figura (2.5)

(2.2) Imagem Lena.gif original

(2.3) Imagem Lena.gif com ruído gaussiano

(2.4) Imagem Cible.gif

(2.5) Imagem Cible.gif com ruído gaussiano

Como pode ser observado, a adição de ruído sobre a imagem faz com que diversos detalhes da imagem sejam praticamente perdidos, por exemplo, no rosto e no cabelo da menina, no caso da imagem (2.3), quando comparada à imagem (2.2). Além disso, as partes mais lisas e homogêneas da imagem, por exemplo, o plano de fundo da imagem (2.3), e as partes laterais e a parte bem central da imagem (2.5), perdem a sua homogeneidade e ficam com um aspecto "salpicado" ou manchado. A imagem perde, então, sua nitidez tanto nas regiões de bordas, que se tornam borradas, como nas regiões planas. Nota-se ainda que o ruído provoca uma distorção dos tons e do brilho da imagem de um modo geral.

2.2) Descrição dos métodos para remoção de ruído e restauração de imagem

Existem diversas arquiteturas de filtro normalmente empregadas para a remoção de ruído aditivo de imagens em preto e branco que apresentam um bom desempenho na remoção do ruído e na restauração da imagem. Nesta seção, serão descritos estes métodos e, além disso, serão também apresentados resultados obtidos por estes filtros e comentários com relação aos seus problemas de desempenho que motivaram o desenvolvimento do filtro apresentado neste relatório.

2.2.1) Conceitos básicos

2.2.1.1) A janela de filtragem

Como já foi explicado anteriormente, as imagens digitais preto e branco são representadas por meio de matrizes bidimensionais, onde cada elemento apresenta um valor representando o nível de cinza de um pixel da imagem correspondente à sua mesma posição espacial dentro desta matriz. Além disso, os pixels próximos entre si apresentam maior probabilidade de apresentarem semelhança entre seus valores de nível de cinza, e de fazerem parte de um mesmo elemento da imagem, do que pixels que estão distantes entre si. Ou seja, a

correlação entre pixels localizados próximos é mais alta do que a correlação entre pixels distantes entre si na imagem.

Devido a este formato de representação e ao fato de pixels próximos apresentarem maior correlação, é comum o uso de janelas de filtragem para o processamento e a filtragem espacial de imagens digitais. Neste caso, quando cada pixel da imagem que é processado durante a filtragem da imagem, são considerados no cálculo do seu valor estimado também os pixels localizados dentro de uma distância predeterminada deste pixel. Por exemplo, se usarmos uma janela de filtragem 5x5 para a filtragem de uma imagem com dimensões 12x12, então, para o processamento de cada pixel, será usada uma matriz de dimensões 5x5, cujo pixel central, ou seja, o elemento (3,3) da janela é o pixel que está sendo processado. Os outros elementos da janela são os pixels localizados ao redor deste pixel central na matriz de intensidade da imagem. Na matriz (2.2) a seguir, será mostrado um exemplo de uma janela de filtragem usada para processamento do pixel (4,5) na filtragem da imagem Lena.gif.

133	133	130	132	134	131	131	128	129	135	126	128
133	133	130	132	134	131	131	128	129	135	126	128
135	130	130	130	133	129	127	127	128	130	124	123
130	130	130	129	**127**	126	127	126	125	128	125	126
127	127	127	127	128	127	125	127	127	127	129	125
128	128	127	127	128	127	126	125	126	126	125	124
126	126	127	123	127	126	127	126	124	126	126	125
125	127	128	125	128	125	128	130	130	130	125	127
126	127	126	125	127	126	129	129	129	127	123	126
127	126	122	127	128	126	127	128	130	127	127	128

Matriz (2.2) Matriz de intensidade de Lena.gif com janela de filtragem indicada

O cálculo do valor estimado pelo filtro deste pixel será efetuado considerando também os demais pixels dentro da janela de filtragem indicada, uma vez que estes pixels que são localizados próximos ao pixel central apresentam, mais provavelmente, valores de intensidade próximos ao valor do pixel que está sendo processado. A janela de filtragem vai "deslizando" sobre a imagem durante o processo de filtragem, até que todos os pixels da imagem tenham sido processados como pixel central da janela.

2.2.1.2) As máscaras de filtragem

O cálculo do valor estimado de cada pixel da matriz de imagem em filtros espaciais do tipo abordado neste trabalho é realizado considerando os valores de intensidade dos pixels localizados ao seu redor, ou seja, dentro da janela de filtragem utilizada. No entanto, a influência ou o peso que cada um dos pixels dentro da janela de filtragem irá exercer sobre o cálculo do valor do pixel que está sendo processado pode variar.

Uma máscara de filtragem consiste em uma matriz usada em filtros espaciais para o cálculo do valor estimado de cada pixel. A máscara apresenta, normalmente, a mesma dimensão que a janela de filtragem, sendo que cada um dos seus elementos ou a combinação destes determina o peso que cada um dos pixels da janela de filtragem exercerá no cálculo do valor do pixel central da janela.

As máscaras podem ser usadas de diferentes maneiras na filtragem de uma imagem. Elas podem ser convoluídas com a janela de filtragem selecionada. Pode-se ainda fazer uma multiplicação matricial entre a janela de filtragem e a máscara. Ou ainda, pode-se simplesmente multiplicar cada elemento da máscara pelo elemento na posição na janela de filtragem. Depois, o valor estimado do pixel central é calculado pela média aritmética ponderada dos resultados destas multiplicações.

2.2.2) Métodos existentes

2.2.2.1) O filtro de média

O filtro de média é um dos filtros espaciais com arquitetura mais simples para restauração de imagens corrompidas por ruído. Ele estima o valor de intensidade de cada pixel da imagem através da média aritmética dos valores dos pixels selecionados para a estimação do pixel central. Por exemplo, pode-se considerar todos os pixels dentro de uma janela de filtragem, ou então apenas os 4 pixels que são vizinhos diretos do pixel central, ou qualquer outra combinação de pixels desejada. Neste caso feita uma média aritmética simples dos pixels dos pixels selecionados, de modo que cada um deles proporciona um mesmo peso sobre o valor estimado do pixel que está sendo restaurado. Portanto, o filtro de média é um filtro completamente linear.

Quando usado sozinho, ou seja, sem estar combinado a outra técnica de filtragem espacial, este filtro consegue preservar alguns detalhes e algumas bordas da imagem, porém apresenta o inconveniente de borrar bastante grande parte das bordas da imagem [1], o que resulta em uma perda de nitidez da imagem restaurada em relação à imagem original. Isso se deve ao fato desta arquitetura de filtro de média não permitir uma filtragem diferenciada nas regiões de borda da imagem e das regiões planas. Este resultado pode ser observado na figura (2.6) que é o resultado de 3 iterações do filtro médio da imagem Lena.gif em que foram considerados somente os valores dos 4 pixels adjacentes a cada pixel para o cálculo do seu valor estimado.

Figura (2.6) *Lena.gif* após 3 iterações do filtro médio

2.2.2.2) O filtro de mediana

O filtro de mediana também é um filtro espacial de arquitetura bastante simples para restauração de imagens corrompidas por ruído. Ele estima o valor de intensidade de cada pixel da imagem pelo cálculo da mediana dos valores selecionados para serem considerados no cálculo do pixel em questão.

Para o cálculo da mediana, todos os pixels selecionados são ordenados e numerados em ordem crescente. Caso o número de pixels seja ímpar, então o valor da mediana será igual ao valor do pixel central dentre os pixels selecionados, quando estão ordenados em ordem crescente. Por exemplo, se forem selecionados 11 pixels, o valor da mediana será igual ao valor do 6º pixel em ordem crescente. O exemplo abaixo ilustra esta operação:

Pixels selecionados: [13 8 11 37 69 68 21 25 24 21 66]
Ordenação dos pixels em ordem crescente: [8 11 13 21 21 24 25 37 66 68 69]
Valor da mediana: 24

Caso a seqüência de valores selecionados para o cálculo da mediana tenha uma quantidade par de elementos, então o valor da mediana pode ser calculado como a média aritmética entre os dois valores centrais da seqüência. Por exemplo, se considerarmos a mesma seqüência anterior, porém com 10 elementos, então o valor da mediana será calculado pela média aritmética dos valores do 5º e do 6º elemento da seqüência, da seguinte maneira.

Pixels selecionados em ordem crescente: [8 11 13 21 21 24 25 37 66 68]
Pixels centrais: 21 e 24
Valor da mediana: 22,5

Este processo é completamente não linear, por isso pode produzir resultados completamente aleatórios. Normalmente, este filtro também gera uma imagem com as bordas menos borradas do que o filtro de média, mas também diminuindo a nitidez da imagem restaurada em relação à imagem original. Também neste tipo de filtro, não é possível filtrar de forma diferenciada os pixels que pertencem às regiões de borda da imagem e aqueles que pertencem às regiões planas.

A imagem (2.7) é um exemplo da aplicação do filtro de mediana á imagem *Lena.gif* com ruído gaussiana com variância igual a 0.1.

Figura (2.7) - *Lena.gif* filtrada pelo filtro de mediana

Tanto o filtro de média quanto o filtro de mediana realizam uma filtragem passa-baixa da imagem, retirando o ruído de freqüência alta.

2.2.2.3) O filtro sigma comum

O filtro sigma também é um filtro espacial para restauração de imagens corrompidas por ruído, que foi desenvolvido para com a finalidade principal de proporcionar um desempenho superior ao dos filtros médios no que se refere à conservação das bordas da imagem. O fundamento principal deste filtro consiste na estimação do valor de intensidade de cada pixel da imagem com base na média e na variância dos valores dos pixels localizados dentro de uma janela de filtragem de dimensões 2m+1 x 2n+1 centrada no pixel a ser estimado [6] [1]. Uma vez obtidos estes valores, a estimativa do pixel é feita somente com base nos pixels da janela que pertencem à mesma classe que o pixel central [5]. Por pixels que pertencem a uma mesma classe, deve ser entendido aqueles pixels que apresentam valores de intensidade próximos e provavelmente pertencem a um mesmo elemento da imagem. O funcionamento detalhado deste filtro será explicado a seguir.

A imagem a ser filtrada, após sofrer interferência de um ruído, pode ser representada pela equação (2.1.2) sendo que y(i,j) representa os valores de intensidade de cada pixel da imagem, após a ação do ruído, x(i,j) corresponde aos valores de intensidade de cada pixel da imagem original e n(i,j) representa o ruído que foi adicionado à imagem original [7].

O filtro sigma utiliza uma janela de filtragem de dimensões 2m+1 x 2n+1 centrada em um pixel y(i,j), e esta janela vai deslizando por todos os pixels da imagem, até que toda a imagem tenha sido filtrada. Em geral, os valores utilizados para m e n são m=n=2. A média μ e a variância σ^2 dos valores de intensidade dos pixels contidos em cada janela são calculados da seguinte forma:

$$\mu(i,j) = \frac{1}{(2m+1)\times(2n+1)} \sum_{k=i-n}^{i+n} \sum_{l=j-m}^{j+m} y(k,l) \qquad (2.2)$$

$$\sigma^2(i,j) = \frac{1}{[(2m+1)\times(2n+1)]-1} \sum_{k=i-n}^{i+n} \sum_{l=j-m}^{j+m} [y(k,l)-\mu)]^2 \qquad (2.3)$$

sendo que μ é a média dos valores de intensidade dos pixels dentro da janela de filtragem e σ^2 é a variância dos mesmos. Nestas equações, os índices *k* e *l* são usados para indexar os elementos dentro da janela de filtragem e i e j são os índices da matriz de intensidade. Depois que estes valores são obtidos, o novo valor estimado u(i,j) do píxel y(i,j) que está sendo filtrado é calculado pela média aritmética somente dos pixels que apresentam um valor de intensidade dentro de uma determinada faixa ao redor do valor da média. As equações usadas nesta operação são as seguintes:

$$u(i,j) = \frac{\sum_{k=i-n}^{n+i} \sum_{l=j-m}^{m+j} w_{k,l} \times y(k,l)}{\sum_{k=i-n}^{n+i} \sum_{l=j-m}^{m+j} w_{k,l}} \qquad (2.4)$$

$$w_{k,l}: \begin{cases} 0 & \text{se } |y(k,l)-\mu| > \\ 1 & \text{se } |y(k,l)-\mu| \leq \end{cases} \qquad (2.5)$$

onde u é o novo valor estimado para o pixel e $w_{k,l}$ é matriz de fatores multiplicadores de y(k,l), que permite que somente aqueles pixels cuja intensidade encontra-se dentro da faixa de valores adequada ao redor da média da janela sejam considerados no cálculo da estimativa do pixel da imagem restaurada. A matriz que é formada pela variável $w_{k,l}$ funciona como uma espécie de máscara, pois somente os elementos nas posições correspondentes aos pixels que serão considerados no cálculo da estimativa de u são iguais a 1. Então quando os elementos da matriz são multiplicados pelos elementos nas suas posições correspondentes da janela de filtragem, todos elementos da matriz resultante localizados nas posições dos pixels que não serão considerados na estimativa são zerados, de modo que sobram somente os elementos correspondentes aos pixels que serão consideados no cálculo da estimativa.

Normalmente, esta faixa de valores de intensidade está relacionada à variância da janela de filtragem. No entanto, algumas considerações devem ser feitas. Como nas regiões de borda da imagem, os valores de intensidade dos pixels apresentam grande variação, logo, a variância destas regiões será bem mais alta do que a variância das regiões planas da imagem, onde os valores dos pixels são mais próximos entre si [6]. Desta forma, é necessário que os valores dos limites desta faixa sejam diferentes para cada janela, e também suficientemente grandes para que se possa fazer a filtragem.

Portanto, para se estabelecer os limites desta faixa de valores, determinou-se que, como valor inicial, seria adequado fazer $\delta = 2\sigma$, uma vez que é sabido que, em uma distribuição normal, cerca de 95%, dos pontos estão compreendidos entre a média mais ou menos duas vezes o desvio padrão $(\mu \pm 2\sigma)$. A aplicação de um valor muito alto pode resultar na não conservação de algumas bordas menos significativas da imagem, as quais, por sua vez, apresentam menor variância. Como conseqüência, a imagem pode ficar borrada e perder nitidez [6]. Em vista disso, o valor de Δ , que representa o limite da faixa de valores no algoritmo, pode ser modificado para cada iteração de filtragem, de acordo com a segunda equação do sistema a seguir, a fim de preservar estas bordas [6]:

$$\begin{cases} \Delta = \delta \\ \Delta = \delta \times \max\left[0.2, \dfrac{1}{1+\dfrac{\sigma}{\delta}}\right] \end{cases} \qquad (2.6)$$

No entanto, a filtragem sigma só é realizada, quando o número M de pixels contidos na janela de filtragem cujo valor de intensidade está dentro da faixa desejada for superior a um valor K. Normalmente, K =min[n+1, m+1] [6] [8]. Caso contrário, ao invés do valor do pixel ser estimado com base na média aritmética dos pixels com valor dentro da faixa desejada, ele é estimado pela média aritmética dos seus 4 pixels adjacentes, de forma similar ao filtro de média conhecido. Este cálculo da estimativa do pixel é representado pelas equações 2.4 e 2.5. Alternativamente, o valor do pixel também pode ser estimado pelo valor da mediana dos seus 4 pixels adjacentes.

2.2.3) Os problemas de desempenho das arquiteturas apresentadas

O filtro de média, assim como o filtro de mediana, são as arquiteturas de filtro para remoção de ruído de imagens em preto e branco sem dúvida de mais simples implementação. No entanto, estes filtros realizam um procedimento de filtragem igual para todos os pixels da imagem, independente destes pertencerem a uma borda da figura, ou a uma região de textura, ou a uma região plana. Cada pixel adjacente ao pixel que está sendo restaurado acaba exercendo igual influência na estimativa do valor deste pixel, sejam eles pertencentes a um mesmo objeto da imagem (mesma classe), sejam eles parte de objetos completamente distintos e com valores de intensidade completamente diferentes.

Contudo, um processamento adaptativo de remoção de ruído da imagem que processa cada pixel da imagem em trechos de regiões suaves e ao mesmo conserva e acentua as suas bordas [9] provê como resultado uma imagem de melhor qualidade, onde as bordas são mais acentuadas, o que torna a imagem mais nítida, e ao mesmo tempo, as regiões planas tem grande parte de seu ruído removido.

O filtro sigma se propõe a realizar este tipo de filtragem adaptativa de cada pixel, em função da avaliação dos pixels que estão à sua volta. Este procedimento de filtragem sofreu diversas adaptações ao longo das últimas décadas, em relação à arquitetura proposta inicialmente por Lee em [1]. A mais importante destas alterações foi aquela realizada por C. H. Kuo, A. H. Tewfic em [6], em que foi criado um algoritmo para o cálculo do valor de limite de Δ que deveria ser aplicado ao filtro sigma, na etapa em que ocorre a separação dos pixels que pertencem à mesma classe que o pixel central e, portanto, serão considerados no cálculo da sua estimativa, daqueles pixels que não pertencem à mesma classe e, em vista disso, serão descartados no cálculo da estimativa do pixel central. Os efeitos deste resultado sobre o desempenho do filtro sigma serão ilustrados no próximo capítulo deste trabalho, ocasião em que serão comparados os resultados das simulações de algumas arquiteturas de filtro.

No entanto, tanto o filtro sigma proposto por Lee, assim como as alterações propostas em relação ao filtro sigma original se propõem unicamente a separar os pixels que serão considerados na estimativa do pixel central daqueles que não serão considerados naquela estimativa. O efeito final desta filtragem acaba por resultar em uma conservação de bordas da imagem. Todavia, nenhum destes filtros se propõe a identificar quais são os pixels da imagem que fazem parte de alguma borda da mesma, a fim de prover um processamento específico e diferenciado para estes pixels, com base nas características e na direção desta borda.

O objetivo deste trabalho é de tentar prover um mecanismo de detecção de bordas a ser usado em conjunto com o filtro sigma, a fim de possibilitar que as bordas da imagem sofram um processamento diferenciado e específico em função do tipo de borda à qual elas pertencem. Desta forma, o desempenho do filtro poderia ser otimizado, uma vez que este mecanismo aplicado em conjunto com o filtro sigma pretende melhorar ainda mais a conservação das bordas da imagem.

A fim de prover meios para desenvolver este mecanismo e implementá-lo juntamente com a arquitetura do filtro sigma, foram pesquisadas diversas técnicas de tratamento e processamento de imagem, com o intuito de encontrar alguma forma de processamento que pudesse atingir o resultado desejado. Dentre as técnicas pesquisadas, aquelas que se mostraram relevantes e foram aplicadas e testadas em conjunto com o filtro sigma serão explicadas a seguir.

2.3) Conceitos teóricos aplicados ao desenvolvimento deste projeto

Nesta seção, serão descritos alguns conceitos teóricos que foram estudados ao longo do desenvolvimento deste projeto e que serão aplicados ao filtro desenvolvido. Serão também descritos alguns outros conceitos importantes à compreensão deste trabalho.

2.3.1) Filtragem por meio de *kernels*

Um *kernel* consiste em uma máscara que é normalmente utilizada para ser multiplicada ou convoluída pela matriz de intensidade de uma imagem e produzir o efeito de um filtro. Os *kernel*s também são denominados, em alguns casos, de operadores. A matriz do *kernel* pode ser construída em uma forma específica tal, para que se tenha um determinado efeito desejado após ser aplicada na restauração da imagem. Normalmente, a matriz utilizada como *kernel* apresenta dimensões iguais às da janela de filtragem, sendo portanto bem menores do que a matriz da imagem.

Quando uma imagem é filtrada pela multiplicação ou convolução da sua matriz de intensidade de pixels com um determinado *kernel*, é gerada uma nova matriz de intensidade de pixels, em que o novo valor de intensidade de cada pixel é proporcional a uma média ponderada dos pixels que estão ao seu redor. O *kernel* utilizado é que irá definir esta ponderação entre os pixels. Por exemplo, se um *kernel* for aplicado à imagem pela simples multiplicação de cada um dos seus elementos pelo elemento em posição correspondente dentro da janela de filtragem, então o valor de cada elemento do *kernel* estabelecerá qual o peso que o pixel correspondente na janela exercerá no cálculo do valor de intensidade do pixel central.

Com base nas informações apresentadas acima, que esclarecem a influência de cada parâmetro do *kernel* no resultado da filtragem, é possível elaborar *kernels* que sejam capazes de exercer efeitos obtidos com outros filtros utilizados na filtragem de sinais contínuos, ou quaisquer outros tipos que realizem um efeito desejado sobre a imagem. Alguns *kernels,* ou operadores, já são bastante conhecidos no campo do processamento de imagens, os quais serão mencionados ao longo deste trabalho, pois realizam funções bastante úteis para a

manipulação de imagens, tais como o cálculo de gradientes de intensidade da imagem, ou a filtragem com uma função gaussiana bi-dimensional ou suas derivadas.

2.3.2) Detecção de borda

As regiões de borda da imagem são caracterizadas por apresentarem uma grande variação de intensidade dos pixels, onde ocorre a separação entre os dois objetos da imagem que constituem a borda. Estas regiões podem ser também designadas como regiões de altas freqüências da imagem, devido à grande variação de intensidade entre os pixels.

As técnicas de detecção de borda são utilizadas para se identificar as regiões ou os pixels da imagem correspondentes às suas bordas. Elas são empregadas em combinação com alguns algoritmos de filtragem digital de imagem. Através desta combinação de algoritmos, a imagem pode ser filtrada de maneira mais eficiente, uma vez que as regiões de borda podem ser tratadas de maneira diferente das regiões planas. Desta forma, a imagem resultante poderá apresentar maior nitidez, uma vez que suas bordas são conservadas, e suas regiões planas ficam mais isentas de ruídos.

Para o caso particular do filtro sigma, esta melhora do resultado da filtragem ocorre uma vez que o parâmetro fundamentalmente usado para a estimativa dos pixels é o valor da variância dos pixels dentro da janela de filtragem, como já explicado anteriormente. Em uma janela de filtragem disposta sobre uma região de borda, evidentemente a variância dos pixels será mais alta do que a variância dos pixels em uma janela sobre uma região plana, uma vez que uma borda na imagem necessariamente implica em uma variação brusca dos níveis de intensidade dos pixels entre os dois elementos da imagem que compõem esta borda. Já em uma região plana e sem texturas, pixels adjacentes apresentam um valor de intensidade praticamente igual e, portanto, um baixo valor de variância.

O valor de Δ empregado no filtro sigma determina os limites da faixa de valores de intensidade dos pixels dentro da janela de filtragem, delimitando assim os pixels que serão considerados no cálculo do valor estimado do pixel que está sendo filtrado. Quanto maior o valor de Δ, maior será a quantidade de pixels dentro da janela que serão considerados na estimativa do pixel filtrado, bem como maior será a diferença entre os seus valores de intensidade, ou seja, serão considerados então pixels cujos valores são bem distantes do valor

original do pixel filtrado. A tendência é que ocorra uma homogeneização dos valores de intensidade dos pixels dentro de uma mesma região, uma vez que muitos pixels contidos na janela são usados no cálculo da estimativa, o que aumenta a influência da média geral da intensidade dos pixels dentro daquela região sobre o valor estimado do pixel filtrado. Portanto, o uso de um alto valor de Δ é vantajoso e desejável para a remoção de ruído das regiões planas da imagem.

Nas regiões de borda, por outro lado, este efeito não é desejável, pois reduz a diferença entre as intensidades dos dois elementos de imagem que compõem a borda, o que resulta na perda de nitidez da imagem. Em vista disso, para se obter uma melhor conservação de bordas, é necessário que um baixo valor de Δ seja empregado na filtragem. Desta forma, somente os pixels com valor de intensidade mais próximos da média são considerados na estimativa do pixel filtrado.

2.3.3) Gradientes

Uma das maneiras de se determinar as bordas de uma imagem é por meio do cálculo dos gradientes de intensidade da imagem. O gradiente é calculado pela derivada da intensidade da imagem. Na prática, como a função que representa a intensidade da imagem consiste em uma função unidimensional discreta, a primeira derivada desta função pode ser calculada pela diferença absoluta entre os valores de intensidade de pixels adjacentes ou dentro de uma mesma janela de filtragem. Por exemplo, no caso de um filtro que utilize uma determinada janela de filtragem de dimensões (2m+1) x (2n+1) para se estimar o valor do pixel central (m,n), pode-se calcular os gradientes do pixel central em relação a todos os outros pixels contidos na janela correspondentes. Estes gradientes consistirão na diferença entre o valor de intensidade do pixel (m,n) e o valor de intensidade de cada um dos outros pixels dentro da janela. Portanto, se uma janela de filtragem de dimensões, por exemplo, 3 x 3 pixels está sendo usada, a qual contém 9 pixels, então serão calculados 8 gradientes de intensidade dentro desta janela.

Para se calcular o valor absoluto do gradiente dentro de uma janela, normalmente calcula-se separadamente o gradiente G_x na direção x (horizontal) e o gradiente G_y na direção y (vertical). A magnitude do gradiente é então calculada por:

$$|G| = \sqrt{G_x^2 + G_y^2} \qquad (2.7)$$

Um alto valor de gradiente indica uma maior variação de intensidade entre pixels, o que pode caracterizar uma região de borda. Por outro lado, uma região da imagem que apresenta pouca variação de intensidade é uma região com pequena quantidade de detalhes. É possível, portanto, estabelecer-se uma relação entre os valores dos gradientes calculados dentro de uma determinada janela de filtragem e as características e a quantidade de detalhes da imagem na sua região correspondente.

Com base neste aspecto, podemos classificar os pixels da imagem, determinando se pertencem a uma região de borda, ou de textura, ou a uma região plana. Para este fim, são estabelecidas faixas de valores de gradiente indicando os valores do gradiente que correspondem a pixels de borda, os valores que correspondem a pixels de regiões de textura e os que correspondem a regiões planas.

2.3.4) Método de detecção de bordas de *Canny*

O método de detecção de bordas de *Canny* é uma das técnicas de detecção de bordas em imagens *grayscale*. Seu algoritmo, que se propõe a ser um algoritmo ótimo de detecção de bordas, apresenta diversas etapas de processamento que serão descritas a seguir [10].

2.3.4.1) Suavização da imagem

A primeira delas consiste na aplicação de um filtro gaussiano à imagem, a fim de remover os componentes de mais alta freqüência e assim suavizar ("smoothing") a imagem e remover uma parcela do ruído inserido na imagem [11]. Assim, quanto mais a imagem for suavizada, aquelas bordas menos importantes e significativas da imagem tendem a ser removidas e permanecem principalmente as bordas mais fortes e importantes. Esta filtragem consiste na convolução da imagem com um operador gaussiano unidimensional [12] representado pela função

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \qquad (2.8)$$

em que σ corresponde ao desvio padrão (sendo σ^2 a variância) da distribuição gaussiana e corresponde à sua média, que neste caso está sendo considerada como sendo nula.

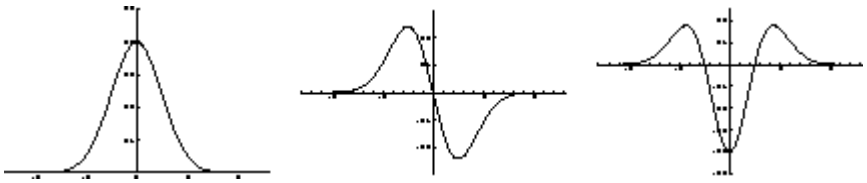
A primeira derivada da função gaussiana unidimensional é dada por:

$$G'(x) = \frac{-x}{\sigma^3\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \qquad (2.9)$$

A segunda derivada da função gaussiana unidimensional é dada por:

$$G''(x) = -\frac{1}{\sqrt{2\pi}\,\sigma^3} e^{-\frac{x^2}{2\sigma^2}} \left[1 - \frac{x^2}{\sigma^2}\right] \qquad (2.10)$$

Os gráficos da figura (2.8) seguir ilustram o comportamento das equações 3.2 a 3.4 acima [13]:



G(x) G'(x) G''(x)

Figura (2.8)

A função gaussiana independente e com o mesmo σ^2 também pode ser representada na forma bi-dimensional (direções x e y). A melhor forma de representarmos esta função na forma bi-dimensional é por meio de coordenadas polares **[**14**]**, de modo que $r = \sqrt{x^2 + y^2}$, onde r representa a distância do ponto de coordenada (x, y) em relação à origem.

A função gaussiana bi-dimensional em coordenadas polares é dada por:

$$G(r) = \frac{1}{2\pi\sigma^2} e^{\frac{-r^2}{2\sigma^2}} \qquad (2.11)$$

A representação gráfica desta função é ilustrada no gráfico da figura (2.9) a seguir, em função das coordenadas (x,y).

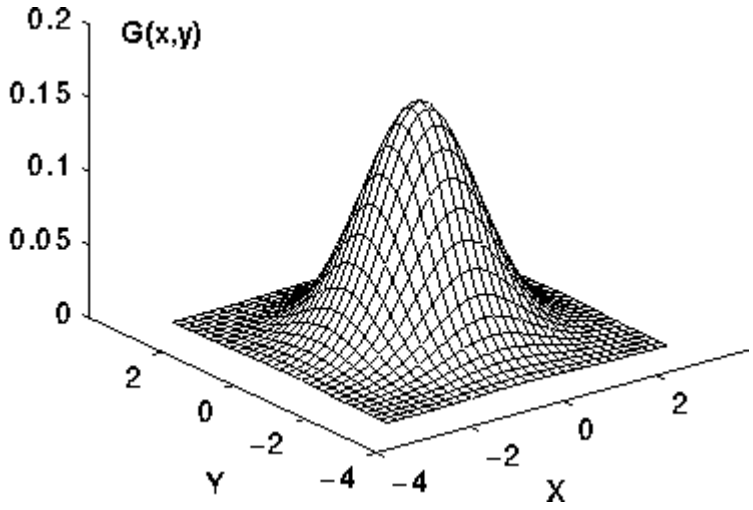


Figura (2.9)

A primeira derivada da função gaussiana bi-dimensional é dada por:

$$G'(r) = \frac{-r}{2\pi\sigma^4} e^{\frac{-r^2}{2\sigma^2}} \qquad (2.12)$$

A segunda derivada da função gaussiana bi-dimensional é dada por:

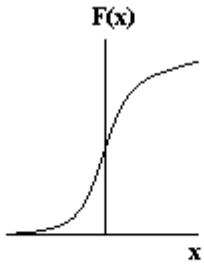
$$G''(r) = -\frac{1}{2\pi\sigma^4} e^{\frac{-r^2}{2\sigma^2}} \left[1 - \frac{r^2}{\sigma^2}\right] \qquad (2.13)$$

Nas regiões de borda da imagem, ocorre uma grande variação de intensidade entre pixels adjacentes que fazem parte de objetos diferentes que compõem esta borda. Se traçarmos um gráfico de intensidade (função intensidade I(x)) dos pixels próximos à região de borda em uma das direções, por exemplo, na direção x, notaremos que este gráfico terá um formato que se aproxima de um degrau, sendo que o local onde se dá a grande variação de intensidade corresponde exatamente à posição da borda.

Se convoluirmos a função intensidade da imagem unidimensional f(I), a qual tem o formato aproximado de um degrau, com a função gaussiana unidimensional

$$F(x) = G(x) * I(x) \qquad (2.14)$$

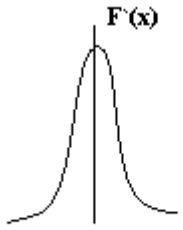
obteremos como resultado uma função F(x) cujo gráfico mostrado na figura (2.10) se assemelha a um degrau suavizado, conforme ilustrado a seguir:



Função Intensidade suavizada

Figura (2.10)

A primeira derivada desta função apresentará um ponto de máximo exatamente no local onde a função F(x) cruza o eixo vertical e que corresponde exatamente ao valor de x onde está localizada a borda da imagem. A derivada segunda de F(x), por outro lado, terá valor F''(x)=0 na posição da borda da imagem. Os gráficos das figuras (2.11) e (2.12) abaixo ilustram o comportamento de F'(x) e F''(x).



Primeira derivada de F(x)

Figura (2.11)

Segunda derivada de F(x)

Figura (2.12)

Os gráficos acima ilustram as derivadas primeira e segunda de F(x), ou seja, após a convolução de I(x) e G(x). No entanto, resultados similares serão obtidos se convoluirmos I(x) diretamente com a primeira derivada ou segunda derivada de G(x).

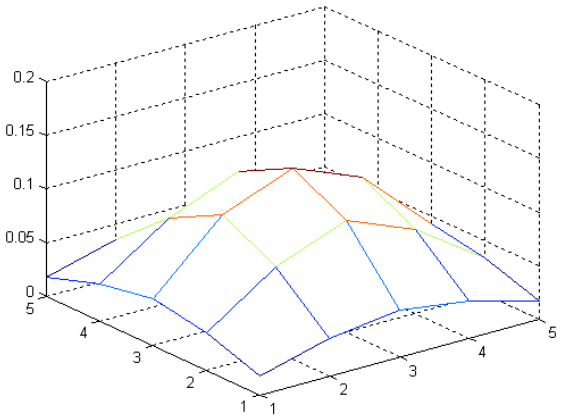
O método de detecção de bordas de Canny utiliza os resultados expostos anteriormente para identificar onde estão localizadas as bordas de uma imagem. A primeira etapa deste

método consiste na suavização da imagem. Uma vez que a imagem estática é representada por uma matriz bi-dimensional de pixels, idealmente ela deveria ser convoluida com a função gaussiana bi-dimensional. No entanto, dependendo das dimensões da matriz de pixels da imagem, este cálculo poderia ficar muito complexo. Analisando-se a equação 3.5 que representa a função gaussiana bi-dimensional em função de r, se substituirmos a variável r pela expressão correspondente $\sqrt{x^2 + y^2}$, a fim de representar esta função em função das variáveis x e y, nota-se que a equação da função gaussiana bi-dimensional em função de x e y pode ser separada em uma componente em função de x e em uma componente em função de y, já que estas variáveis são independentes. Em vista disso, a fim de simplificar os cálculos e reduzir a complexidade computacional do algoritmo de detecção de bordas, em algumas das suas aplicações, podem ser utilizadas duas convoluções unidimensionais da função intensidade com a função gaussiana unidimensional, sendo uma delas na direção x (horizontal) e outra na direção y (vertical).

Caso a opção seja por realizar a convolução da função bidimensional com a imagem, uma alternativa para esta operação seria pelo uso de um *kernel* que representa aproximadamente o formato da função gaussiana bi-dimensional como ilustrada no gráfico da figura (2.9). Os valores empregados do *kernel* dependem entre outros do formato do filtro gaussiano que será aplicado, ou seja, dos valores de média e desvio selecionados para a gaussiana. Um exemplo de um *kernel* de dimensões 5x5 normalmente utilizado para este fim, representando uma gaussiana com média 0 e $\sigma = 1.4$ apresenta os valores da matriz G a seguir [15] e é espacialmente representado pela matriz (2.3) abaixo.

$$G = \frac{1}{115}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \qquad (2.15)$$

Matriz (2.3) Representação espacial de G



Este *kernel* é então deslizado ao long
de filtragem, e é multiplicado pelos pixels sobre os quais ele se sobrepoe. O *kernel*, no entanto, pode assumir dimensões diferentes, podendo também se adequar à janela de filtragem que está sendo usada. Normalmente, quanto maiores forem as dimensões deste *kernel*, em

contrapartida, menor será a sensibilidade do filtro gaussiano a ruídos [16], podendo também aumentar o erro de localização da borda.

Ao final desta etapa, são obtidos dois valores resultantes da filtragem gaussiana para cada pixel, sendo um deles correspondente à convolução na direção x (direção horizontal da imagem) e outro deles correspondente à convolução na direção y (direção vertical da imagem).

2.3.4.2) Diferenciação da imagem

Em seguida à etapa de suavização da imagem, pela aplicação do filtro gaussiano nas direções x e y, é realizada a diferenciação do resultado obtido, para que assim sejam calculados os gradientes da imagem. Pelos valores dos gradientes é possível que os pixels da imagem sejam identificados quanto ao tipo de região da imagem da qual fazem parte [17].

O gradiente da imagem pode ser calculado da seguinte forma, sendo x e y as coordenadas de cada pixel:

$$\nabla F = \left[\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y} \right] \qquad (2.16)$$

A magnitude e a direção do gradiente são respectivamente calculadas através das equações (2.17) e (2.18) abaixo:

$$|\nabla F| = \sqrt{\left(\frac{\partial F}{\partial x} \right)^2 + \left(\frac{\partial F}{\partial y} \right)^2} \qquad (2.17)$$

$$\theta = \tan^{-1}\left(\frac{\partial F}{\partial y} \Big/ \frac{\partial F}{\partial x} \right) \qquad (2.18)$$

A diferenciação da imagem pode ser realizada pela aplicação de operadores derivativos, por exemplo. Em geral, a imagem suavizada é diferenciada nas duas direções x e

y através de um operador derivativo para cada direção. Operadores derivativos adequados a esta aplicação podem ser, por exemplo, operadores que representem a primeira derivada da função gaussiana bi-dimensional. Como esta função é separável em uma componente na direção x e outra na direção y, a diferenciação também pode ser feita separadamente para cada direção.

Uma outra alternativa é usar operadores que calculem a diferença entre o valor da intensidade de um pixel e o valor da intensidade do seu pixel adjacente em duas direções ortogonais, como x e y, ou qualquer outro par de direções ortogonais. Um exemplo simples de operador que realiza esta operação é o operador *"Robert's Cross"* representado pelas matrizes a seguir, que calcula o gradiente fazendo a diferença entre adjacentes nas direções diagonais.:

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \text{e} \quad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad (2.19)$$

Também podem ser usados operadores de dimensões 3 x 3 para calcular o gradiente de cada pixel. Os operadores de *Prewitt* calculam o gradiente fazendo a diferenciação nas direções x e y através dos seguintes operadores:

$$\nabla_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{e} \quad \nabla_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \qquad (2.20)$$

O operador de *Sobel* 3 x 3 também calcula o gradiente pela diferenciação nas direções x e y, no entanto, ele atribui um peso maior aos pixels localizados no centro, e portanto adjacentes nas direções x e y ao pixel central da janela de filtragem, de modo a aumentar a influência destes pixels sobre o valor calculado para o gradiente. As matrizes a seguir representam o operador de Sobel [15].

$$\nabla_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & -1 \end{bmatrix} \quad \text{e} \quad \nabla_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

(2.21)

Para obtermos o valor do gradiente da imagem I, basta convoluirmos estes operadores com a janela de filtragem da imagem. Em seguida, somamos os valores obtidos por cada uma das convoluções.

2.3.4.3) Eliminação de não-máximos

Os pontos de máximo gerados pelo cálculo dos gradientes correspondem aos pixels que constituem as bordas da imagem. Alternativamente, os pixels de borda podem também ser identificados pelos pontos em que a derivada dos gradientes é igual a zero.

A finalidade da etapa de eliminação de não-máximos é eliminar pontos de não-máximos nas proximidades de e na direção perpendicular aos pontos de máximo, de modo que permaneçam somente os pontos de máximo locais de cada região, que constituem a bordas da imagem [17]. Portanto, os pontos de máximo ao longo da borda da imagem que não são máximos locais são eliminados. Ao final, é gerada uma matriz em que todos os elementos correspondentes à posição de pixels que não são de borda são nulos.

Para que esta eliminação seja feita, cada pixel é considerado como sendo o pixel central dentro de uma janela de dimensões 3 x 3. Dada a direção do gradiente relativo àquele ponto, é feita a interpolação dos pixels ao redor do pixel central da janela, através da qual é calculado o gradiente nas duas direções ortogonais à direção do gradiente do pixel central. Caso a magnitude do gradiente do pixel central for menor do que a magnitude dos gradientes em uma das direções ortogonais, então este pixel é considerado como um não-máximo e, portanto, suprimido.

2.3.4.4) Estabelecimento de valores de limiar

Esta etapa é utilizada para eliminar alguns pixels que foram considerados como borda nas etapas anteriores, e que na realidade não correspondem a bordas, mas, em geral, a ruído.

Geralmente, o método de *Canny* utiliza o método de histerese para realizar o "thresholding" da imagem, em que são estabelecidos dois valores de limiar, um superior (t_1) e um inferior (t_2) **[**18**]**. Inicialmente, os pixels cuja intensidade é maior do que t_1 são automaticamente estabelecidos como pontos de borda forte e os pixels cuja intensidade é

menor do que t_2 são imediatamente considerados como não sendo bordas. Os pixels com intensidade entre t_1 e t_2 precisam ser avaliados em relação aos pixels de borda forte, para serem confirmados como pixels de borda ou não.

Quando um pixel com intensidade maior do que t_1 é identificado, ele é considerado como o início (ou a "semente") de uma borda forte [19]. A partir daí, é feita a avaliação dos pixels dentro de uma janela 3x3 centrada ao redor desta semente, ou seja, todos os pixels ao redor da semente são analisados em conexão com esta semente. Os pixels cuja intensidade estiver entre t_1 e t_2 são considerados como borda. Os que forem inferiores são descartados. Este procedimento é realizado sucessivamente, até que seja localizado o final da borda. Isto acontece quando todos os 8 pixels ao redor do pixel central da janela possuem intensidade menor do que o limiar inferior t_2.

Esta etapa de estabelecimento de limiares dos valores de intensidade dos pixels para que sejam detectados como sendo bordas é muito importante para que as linhas de contorno dos objetos da imagem sejam continuas e livres de interrupções. Isso é possível, uma vez que, para que ocorra uma falha em uma linha de uma borda, é necessário que ocorra uma variação muito grande entre as intensidades dos seus pixels, pois é preciso que esta linha seja constituída de pixels com intensidade maior do que t_1 e ao mesmo tempo pixels com intensidade menor do que t_2.

Este método, por utilizar os valores de limiar superior e inferior para detecção de bordas, apresenta menor probabilidade de ser interpretado como ruído [15], ou seja, de identificar ruído como sendo borda, do que os outros métodos de detecção de bordas, pois as bordas fracas somente são identificadas quando estão relacionadas a alguma borda forte. Além disso, este método tem melhor desempenho na identificação de bordas fracas de um modo geral.

2.3.4.5) Ajuste de intensidade para valorização de bordas

Como já explicado anteriormente, o efeito de detecção de bordas alcançado pelas técnicas de processamento de sinais mencionadas no item anterior é obtido pela detecção da variação brusca de intensidade entre pixels adjacentes ou bem próximos. O cálculo do gradiente indica justamente qual é a variação de intensidade em cada direção do pixel central

e, a partir desse dado, é possível detectar em que direção se estende a borda do objeto da imagem.

Uma das alternativas que aparentemente poderiam ajudar no aumento da quantidade de bordas da imagem detectadas pelos mecanismos de detecção de bordas seria através da valorização dos contrastes da imagem, aumentando assim o contraste entre as bordas da imagem. O fato de se aumentar o contraste entre os valores de intensidade dos pixels adjacentes poderia permitir que fossem detectadas determinadas bordas mais fracas da imagem que antes não haviam sido detectadas pelas técnicas de detecção de bordas, uma vez que a diferença entre as intensidades dos pixels constituindo estas bordas não era grande o suficiente para que pudessem ser identificados como uma borda.

Uma das formas de se aumentar o contraste entre estas bordas é através da redistribuição dos valores de intensidade dos pixels da imagem dentro da faixa total de valores de intensidade permitida pela definição (escala) usada na representação da imagem [18]. Por exemplo, em uma imagem representada no formato uint8 do Matlab, o valor de intensidade de cada pixel é representado através de 8 bits, o que possibilita que a intensidade dos pixels seja representada através de 2^8=256 valores diferentes. No entanto, algumas vezes os valores de intensidade não ocupam toda a faixa dinâmica de 256 valores que esta representação permite.

O Matlab possui uma função chamada *imadjust* (J=imadjust(I)) capaz de realizar esta tarefa de redistribuição dos valores de intensidade da imagem, de modo que seu histograma ocupe toda a faixa dinâmica de valores possíveis de intensidade usados na representação dessa imagem. Este procedimento é chamado de equalização de histograma [18]. Esta função faz com que 1% dos pixels da imagem fiquem com seus valores saturados nos valores de intensidade mais alto e mais baixo da representação da imagem empregada. Este processamento aumenta consideravelmente o contraste da imagem resultante. Também é possível fazer o ajuste de intensidade da imagem, sem utilizar toda a faixa dinâmica de valores que a representação da imagem permite, através da sintaxe J=imadjust(I, [*min_in,max_in*], [*min_out,max_out*]). Neste caso, os valores da imagem de entrada [*min_in,max_in*] são respectivamente mapeados para [*min_out,max_out*]. Os valores menores que *min_in* e maiores que *max_in* são fixados respectivamente em *min_out* e *max_out*. Esta função do *Matlab* foi também utilizada nos testes de desempenho do novo filtro desenvolvido,

em conjunto com o filtro sigma para remoção de ruído de imagens em preto e branco já conhecido.

Capítulo 3

Método proposto

Nesta seção, a arquitetura do filtro desenvolvido neste projeto será descrita detalhadamente. Serão também apresentadas outras considerações relacionadas ao projeto, quanto às tecnologias selecionadas para o desenvolvimento do mesmo.

As funções que constituem o filtro desenvolvido foram desenvolvidas na plataforma Matlab versão 7.0. Esta plataforma foi escolhida devido à sua praticidade e facilidade de utilização.

3.1) Considerações com relação às imagens

Neste trabalho será feita a análise dos resultados obtidos pela aplicação do filtro desenvolvido sobre diversas imagens em preto e branco, apresentando características diferentes. As imagens em preto e branco são representadas por pixels com valores de intensidade dentro de uma faixa determinada, que depende do tipo de representação utilizado, que definem uma escala de níveis de cinza.

Quando uma imagem em preto e branco é processada digitalmente, ela deve estar em um formato tal que os valores de luminância ou intensidade de cada pixel sejam representados numericamente. No caso deste projeto, as imagens utilizadas estão originalmente no formato .gif ou .jpg e serão lidas pela função imread do Matlab. Esta função converte automaticamente a imagem .gif ou .jpg para o formato uint8 do Matlab. Isto significa que a imagem será transformada em uma matriz de pixels, em que cada pixel será representado por um valor inteiro entre 0 e 255 correspondente ao seu valor de intensidade, utilizando 8 bits para tal representação.

Durante os testes de desempenho deste filtro serão utilizadas algumas imagens que apresentam características diversas, por exemplo, em relação à quantidade de altas e baixas freqüências, bordas mais fortes, bordas mais fracas e texturas, e presença de detalhes. Esta variedade de imagens testadas permitirá que o desempenho do filtro seja verificado na

restauração de cada um destes tipos de características das imagens. As imagens testadas são *Lena.gif*, *Baboon.jpg*, *Cameraman.gif* e *Cible.gif*.

3.2) A arquitetura do filtro desenvolvido

Nesta seção serão descritas e explicadas as funções desenvolvidas no Matlab para a implementação do filtro para a restauração de imagens deste projeto, bem como a arquitetura final do filtro com base nestas funções.

3.2.1) A função "*tratamatriz*"

A função "tratamatriz" foi criada para adaptar as dimensões da imagem às etapas de filtragem.

Como já explicado anteriormente, o filtro gaussiano, que serve de base para o filtro desenvolvido neste projeto, utiliza janelas de filtragem de dimensões (2m +1) x (2n +1) para o cálculo da estimação do valor de cada pixel. Estas janelas são centradas sobre o pixel cujo valor será estimado. Em vista disso, para que os pixels localizados nas bordas da matriz de imagem pudessem ser filtrados, tivemos que aumentar as dimensões da imagem, de modo a permitir que estes pixels possam ficar centrados na janela de filtragem.

Para que esta função fosse construída, foi necessário determinar variáveis, na função principal de filtragem "***sigmaedge***", que definissem as dimensões das janelas de filtragem, as quais foram denominadas de m e n. Estas variáveis são, naturalmente, passadas para a função ***tratamatriz***.

A função ***tratamatriz*** é responsável por acrescentar ao redor da imagem o número de linhas e colunas de pixels necessário para que todos os pixels iniciais da imagem sejam filtrados. Os pixels acrescentados não serão filtrados, ou seja, eles nunca serão o pixel central da janela de filtragem. Como a janela de filtragem tem dimensões (2m +1) x (2n +1), então para que estes pixels nunca sejam filtrados e ao mesmo tempo todos os pixels iniciais da imagem possam ser filtrados, devem ser acrescentadas exatamente m linhas acima e abaixo da matriz de pixels da imagem e n colunas à esquerda e à direita da matriz de pixels da imagem.

Primeiramente, foi gerada uma matriz com as novas dimensões da matriz de imagem a ser processada, ou seja, com as dimensões originais da imagem e acrescidas de 2m linhas e 2n colunas. A imagem original ficou localizada no centro da matriz e todos os pixels adicionados foram estabelecidos com o valor 255, correspondendo à cor branca. Este processo foi realizado apenas para facilitar as manipulações a seguir.

Foram implementadas algumas modalidades diferentes desta função. Em uma primeira opção, a imagem permanece com todos os pixels adicionais iguais a 255. Outra possibilidade consiste em estabelecer todos os pixels adicionais com valor 0, ou seja, representando a cor preta. E em uma última alternativa, ao invés de valores específicos serem atribuídos aos pixels, as colunas ou linhas da imagem foram rebatidas para as linhas e colunas adicionais. Então, por exemplo, as linhas 1 a n da imagem foi repetidas acima da matriz de pixels original da imagem, porém de forma invertida. Ou seja, a linha 1 foi repetida logo acima dela mesma, a linha 2 foi repetida acima da repetição da linha 1, a linha 3 foi repetida acima da repetição da linha 2 e assim sucessivamente. O mesmo procedimento foi utilizado nas outras 3 bordas da matriz da imagem.

A forma como será feita a ampliação das dimensões da matriz (repetição de linhas e colunas, ou atribuição do valor 0 ou 255 a todos os elementos) é um dos parâmetros passados a função, denominado **tipo**.

3.2.2) A função *"médiadesvio"*

Como já explicado anteriormente, o filtro sigma, que é usado como base para o filtro desenvolvido neste projeto, utiliza os parâmetros de média (μ) e variância (σ^2) dos pixels dentro de uma janela de filtragem para calcular a estimativa do valor do pixel central da janela. Este procedimento se repete para cada pixel da imagem original.

A função ***"médiadesvio"*** tem por finalidade calcular a média e a variância da intensidade dentro de uma dada janela de pixels. Esta função será bastante utilizada no filtro deste projeto.

A janela que será considerada, bem como suas dimensões são passadas diretamente para a função. Como resultado, são passados para a função ***"sigmaedge"*** os valores de média

e desvio, bem como o valor de (Δ) que será usado na função ***"compintens"*** que será explicada a seguir.

3.2.3) A função *"compintens"*

O filtro sigma comum analisa os valores de intensidade dos pixels dentro da janela de filtragem com relação ao valor da média (μ) das intensidades dos pixels da janela. Somente aqueles pixels cuja intensidade y(i,j) está dentro de uma faixa de valores determinada por $\mu + \Delta \leq y(i,j) \leq \mu - \Delta$ são considerados no cálculo da estimativa do pixel central da janela. Como já mencionado, o valor de (Δ) pode estar, por exemplo, atrelado ao valor da variância (σ) dentro da janela.

A função ***"compintens"*** é responsável por realizar a etapa em que todos os pixels contidos dentro da janela de filtragem são comparados com a faixa de valores $\mu + \Delta \leq y(x,y) \leq \mu - \Delta$. Os parâmetros (μ) e (σ), bem como a janela de filtragem e suas dimensões são passados para a função. Como resultado da comparação das intensidades, a função devolve a quantidade **(M)** de pixels cujo valor está dentro da faixa de valores determinada e a quantidade de píxels **(count)** não contida nesta faixa. A função gera ainda uma matriz **w** com as mesmas dimensões da janela de filtragem, em que os elementos nas posições dos pixels que estão dentro da faixa de valores determinada recebem o valor 1 e os elementos nas posições dos pixels cujas intensidades estão fora da faixa de valores determinada recebem o valor 0. Todos estes resultados são fundamentais para que a função ***"sigmaedge"*** faça o cálculo da estimativa do valor do pixel central da janela.

3.2.4) A função *sigmaedge*

A função *sigmaedge* é a função principal do filtro desenvolvido. O funcionamento de cada etapa desta função será explicado a seguir, com base também nas informações anteriores relativas às outras funções desenvolvidas e que são executadas em conjunto com a função *sigmaedge*.

3.2.4.1) Preparo da imagem

Inicialmente, a imagem é preparada para ser submetida ao processamento do filtro. As etapas de preparo da imagem são responsáveis por colocar a imagem nas condições

necessárias para que ela possa ser filtrada. Estas etapas são implementadas na função principal do filtro denominada de ***"sigmaedge"***.

Primeiramente, a imagem **(A)** a ser processada é lida pela função "*imread*", através da qual ela será transformada em uma matriz em que cada pixel assumirá um valor de intensidade entre 0 e 255.

Em seguida, um ruído é adicionado à imagem **(A)**, através da função "*imnoise*". Foi utilizado o ruído gaussiano branco, pois ele possibilita que sua média e variância sejam ajustadas e, desta forma, é possível controlar a relação sinal ruído da imagem. A função padrão do Matlab ajusta automaticamente os valores destes parâmetros como média = 0 e variância = 0.01. Neste trabalho, os valores destes parâmetros foram variados, a fim de testar a eficiência do filtro para diversos valores de relação sinal ruído. A influência destes parâmetros será mais detalhadamente discutida na seção "Parâmetros importantes do filtro" **(**3.3**)**.

Antes que a imagem comece a ser processada para que o ruído gaussiano seja removido, ela precisa sofrer um tratamento, para que suas dimensões se adaptem às funções de processamento às quais a imagem será submetida. Para isso, aplicamos a função "***tratamatriz"*** à matriz da imagem **(A)**. É gerada a matriz **(J)** que será efetivamente filtrada, com as dimensões aumentadas de 2m linhas e 2n colunas, atendendo assim aos requisitos de tamanho necessários para que seja filtrada.

3.2.4.2) Iterações de filtragem da imagem

Uma vez que a matriz se encontra nas dimensões adequadas, ela pode ser submetida às etapas de filtragem. Agora o algoritmo entra em um *loop*, onde todas as operações a seguir são realizadas para todos os pixels da imagem.

3.2.4.2.1) Detecção de bordas

A matriz J gerada pela função *tratamatriz* será então submetida às técnicas de detecção de bordas. A função *edge* é usada para identificar os pixels da matriz que fazem parte de bordas da imagem. Esta função do Matlab permite que diversos métodos de detecção sejam

usados. O método de detecção é um dos parâmetros que deve ser passado para a função. O método de detecção de *Canny* foi escolhido para ser utilizado no projeto. Este método já foi explicado em detalhes no capítulo 2 deste relatório.

A função *edge* recebe ainda, como parâmetro de entrada, a matriz de imagem **J**, que será analisada quanto à presença de bordas, e devolve uma matriz **BW**, com dimensões idênticas às da matriz que foi analisada, em que os elementos correspondentes às posições de pixels de borda possuem o valor 1 e os demais elementos possuem valor 0.

Quando somente a matriz de entrada (**J)** e o método de detecção de bordas (***Canny***) são passados como parâmetros de entrada para a função *edge*, então a própria função estipula um valor de sigma usado no filtro gaussiano do método de *Canny,* como explicado no ítem 2.3.4.1. O valor de sigma padrão fixo da função *edge* é igual a 1. Já os valores de limiar máximo e mínimo explicados no item 2.3.4.4, também usados no método de *Canny*, são calculados automaticamente pela função *edge* para cada pixel da matriz a ser processada. Na realidade, o valor de limiar mínimo é calculado em função das características da imagem e o valor de limiar máximo é calculado proporcionalmente ao valor de limiar mínimo encontrado.

Alternativamente, os parâmetros sigma e valores de limiar máximo e mínimo do método de detecção de bordas de *Canny* também podem ser passados como parâmetros de entrada da função *edge*. A sintaxe desta função do Matlab permite que estes três parâmetros sejam passados conjuntamente, ou que somente os valores de limiar sejam passados e, neste caso, o valor de sigma é atribuído automaticamente como igual a 1.

3.2.4.2.2) Definição dos parâmetros da janela de filtragem

Após detectadas as bordas da imagem, na matriz **J**, esta será efetivamente submetida ao *loop* de filtragem e restauração dos pixels. A primeira etapa consiste na definição da janela de filtragem. Uma matriz denominada **janela** armazena os pixels correspondentes à janela de filtragem. Esta matriz tem dimensões **(2m+1) x (2n+1)**, sendo que normalmente os valores atribuídos a **m** e **n** são em torno de 2 ou 3. Esta matriz **janela** é submetida à função ***mediadesvio***, para que sejam calculados os valores de média e variância desta janela, bem como o valor de delta (Δ) que será usado pelo filtro sigma. Os valores de média, variância e

delta calculados para cada janela são respectivamente armazenados nas matrizes **matmedia**, **matdesvio** e **matdelta**.

3.2.4.2.3) Filtragem dos pixels que não constituem bordas

Se o elemento da matriz BW correspondente ao pixel da matriz J que está sendo processado através da janela de filtragem for igual a 0, isto significa que este pixel não é um pixel de borda. Neste caso, a janela de filtragem (matriz **janela**) é submetida a um filtro sigma comum. A matriz **janela** é aplicada à função ***compintens***, para se identificar os pixels dentro da janela que serão considerados para a estimativa do valor do pixel central. O parâmetro correspondente à média da janela, que é necessário à função ***compintens*** já havia sido calculado pela função ***mediadesvio*** no início do algoritmo e armazenado na matriz **matmedia**. O parâmetro delta (Δ) também necessário à função ***compintens*** pode ser recalculado em função de algum outro parâmetro desejado, ou então pode ser aproveitado o valor de delta armazenado na matriz **matdelta** e que havia também sido calculado pela função ***mediadesvio***.

Como já explicado, a função ***compintens*** devolve a quantidade de pixels **(M),** cujo valor de intensidade está dentro da faixa de valores determinada pela equação abaixo, onde μ é a média dos valores de intensidade dos pixels dentro da janela de filtragem.

$$\mu + \Delta \leq y(i,j) \leq \mu - \Delta \qquad (3.1)$$

Os valores de delta são calculados através da fórmula (2.6) apresentada no item (2.2.2.3) deste relatório.

Pelos princípios do filtro sigma, caso o valor de **M** for superior a um valor de limiar **K** estipulado, então a estimativa do pixel central da janela é feita com base na média dos valores dos pixels que se encontram dentro da faixa de valores determinada pela equação acima. Esta operação é realizada com o auxílio da matriz **w**, gerada pela função ***compintens***, que indica as posições dos pixels dentro da janela de filtragem que devem ser usados para a estimativa do pixel central. A matriz **w** apresenta dimensões iguais às da matriz **janela** e funciona como uma espécie de uma máscara para os pixels selecionados para o cálculo da estimativa do pixel central da janela de filtragem.

Como já mencionado antes, normalmente, **K** é estipulado como **min[n+1, m+1]**, sendo que **(2m+1)** e **(2n+1)** são as dimensões da janela de filtragem.

Caso o valor de **M** retornado pela função ***compintens*** seja menor do que **K**, então o valor estimado do pixel central é calculado pelo valor da média aritmética ou da mediana dos valores dos seus 4 pixels imediatamente adjacentes (superior, inferior, à esquerda e à direita).

Esta etapa do filtro sigma pode também ser modelada pelas equações (2.4) e (2.5) apresentadas no item (2.2.2.3) deste relatório. A equação (2.5) resume o funcionamento da função ***compintens***, pois verifica se a intensidade de cada pixel está contida na faixa de valores $\mu \pm \Delta$. A variável $\delta_{k,l}$ representa basicamente a matriz **w**, uma vez que recebe o valor 1, caso o valor de **janela(k,l)** esteja dentro de $\mu \pm \Delta$, e 0, caso y(k,l) esteja fora de $\mu \pm \Delta$, sendo que **janela(k,l)** corresponde a cada um dos pixels da janela de filtragem.

A equação (2.4) representa o cálculo da estimativa do pixel central, para o caso em que **M > K**. A variável **u** representa a estimativa do valor do pixel que é calculada pela multiplicação de cada elemento da matriz $w_{k,l}$ da equação (2.4) pelo elemento na mesma posição da matriz **janela.** Os produtos destas multiplicações são armazenados em uma matriz denominada de **pix1**. Para o cálculo da estimativa de **u,** os elementos da matriz **pix1** são somados e armazenados na variável **pix3**. O resultado desta soma, ou seja, o valor final de **pix3**, é dividido pela quantidade de pixels dentro da janela usados na estimativa de **u.** Esta quantidade pode ser calculada pelo somatório dos elementos não nulos da matriz **w**. Como a matriz w só tem elementos iguais a 0 ou 1, então para calcular a quantidade de elementos em **w**, basta fazer a soma de todos os elementos contidos em **w,** a qual é armazenada na variável **den2**. O resultado da divisão de **pix3** por **den2** é armazenado na variável **pixel**, que corresponde ao valor estimado do pixel que está sendo filtrado naquela iteração.

Todas estas etapas do filtro sigma são realizadas somente se o pixel em questão, que está no centro da janela de filtragem, for identificado como não sendo um pixel de borda da imagem, ou seja, se o elemento de **BW** correspondente a este pixel for igual a 0.

3.2.4.2.4) Filtragem dos pixels que constituem bordas

Se a matriz **BW** identificar o pixel como sendo um pixel de borda, o calculo do valor estimado deste pixel será diferente daquele descrito no item 3.24.2.3. Neste caso, é utilizado um procedimento que visa a determinar aproximadamente a direção da borda da qual este pixel faz parte. Em função da direção desta borda, será aplicado um *kernel*, ou máscara, à janela de filtragem, o qual busca valorizar aqueles pixels na direção da borda, para o cálculo da estimativa do valor do pixel em questão.

Para a determinação da direção aproximada da borda da qual o pixel faz parte, é feita a soma dos elementos da matriz **BW** em diversas direções. Como a matriz **BW** é composta somente por elementos iguais a 0 representando os pixels que não são de borda, e iguais a 1, representando os pixels de borda, então naturalmente a soma dos elementos na direção mais próxima à direção da borda deve apresentar um valor mais alto do que a soma dos elementos de BW nas outras direções testadas.

As direções consideradas para a estimativa da direção da borda são as direções horizontal, vertical, diagonal esquerda e diagonal direita. A direção horizontal é constituída pelos elementos presentes na mesma linha que o elemento correspondente ao pixel cujo valor está sendo estimado, e a soma dos seus elementos é armazenada na variável **somah**. A direção vertical é constituída pelos elementos presentes na mesma coluna que o elemento correspondente a este mesmo pixel, e a soma dos seus elementos é armazenada na variável **somav**. A direção diagonal esquerda é constituída pelos elementos presentes na diagonal da matriz **BW** que parte do seu canto superior esquerdo e passa pelo elemento central, correspondente ao pixel em questão. A soma dos seus elementos é armazenada na variável **somade**. A direção diagonal direita é constituída pelos elementos presentes na diagonal da matriz **BW** que parte do seu canto superior direito e passa pelo elemento central, correspondente ao pixel em questão. A soma dos seus elementos é armazenada na variável **somadd**.

Em seguida, os valores das variáveis **somah**, **somav**, **somade** e **somadd** são comparados, para se determinar qual deles é o maior. Um *kernel* correspondente à direção cuja soma dos elementos apresenta maior valor é, então, aplicado à janela de filtragem original **J(k,l),** para o cálculo da estimativa de **u**.

Como estes *kernel*s são aplicados à matriz **J**, eles devem apresentar as mesmas dimensões desta matriz. Este filtro foi implementado utilizando janelas de filtragem (matriz **J(k,l)**) com dimensões 5x5, que são as dimensões mais comumente utilizadas em implementações de filtros sigma, de acordo com a maior parte dos artigos que tratam do assunto. Os *kernel*s aplicados à janela foram então construídos com estas mesmas dimensões. No entanto, em caso de utilização de janelas de filtragem com dimensões maiores, basta inserir linhas e colunas adicionais com elementos iguais a 0 ao redor do *kernel* original, até que ele apresente as mesmas dimensões da janela.

Os seguintes *kernel*s foram utilizados na estimativa de **u** [20]:

$$d1 = \begin{bmatrix} 0 & 1 & 4 & 1 & 0 \\ 0 & 4 & 16 & 4 & 0 \\ 0 & 8 & 32 & 8 & 0 \\ 0 & 4 & 16 & 4 & 0 \\ 0 & 1 & 4 & 1 & 0 \end{bmatrix} \qquad (3.2)$$

O *kernel* **d1** mostrado na equação (3.2) é utilizado, quando se estima que a direção da borda seja mais próxima da direção vertical. Em vista disso, os elementos da coluna central apresentam um maior peso e, portanto, maior influência no cálculo da estimativa de **u**, e a medida que os elementos vão se afastando desta coluna central, seu peso vai gradativamente diminuindo. Os elementos mais próximos do pixel central também apresentam um maior peso do que os elementos mais distantes do mesmo.

$$d2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 8 & 4 & 1 \\ 4 & 16 & 32 & 16 & 4 \\ 1 & 4 & 8 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{(3.3)}$$

O *kernel* **d2** mostrado na equação (3.3) é utilizado, quando se estima que a direção da borda seja mais próxima da direção horizontal. Logo, os elementos da linha central apresentam um maior peso e, portanto, maior influência no cálculo da estimativa de **u.** À medida que os elementos vão se afastando desta linha central, seu peso vai gradativamente

diminuindo. Assim como no caso anterior, os elementos mais próximos do pixel central apresentam um maior peso do que os elementos mais distantes do mesmo.

$$d3 = \begin{bmatrix} 4 & 4 & 1 & 0 & 0 \\ 4 & 16 & 8 & 2 & 0 \\ 1 & 8 & 32 & 8 & 1 \\ 0 & 2 & 8 & 16 & 4 \\ 0 & 0 & 1 & 4 & 4 \end{bmatrix} \qquad (3.4)$$

O *kernel* **d3** mostrado na equação (3.4) é utilizado, quando se estima que a direção da borda seja mais próxima da direção da diagonal esquerda, que parte do canto superior esquerdo da matriz, passa pelo elemento central e termina no canto inferior direito da mesma. Portanto, os elementos desta diagonal esquerda apresentam um peso maior e, assim, maior influência no cálculo da estimativa de **u.** A medida que os elementos vão se afastando desta diagonal esquerda, seu peso vai gradativamente diminuindo. Assim como nos outros casos, os elementos mais próximos do pixel central apresentam um peso maior do que os elementos mais distantes do mesmo.

$$d4 = \begin{bmatrix} 0 & 0 & 1 & 4 & 4 \\ 0 & 2 & 8 & 16 & 4 \\ 1 & 8 & 32 & 8 & 1 \\ 4 & 16 & 8 & 2 & 0 \\ 4 & 4 & 1 & 0 & 0 \end{bmatrix} \qquad (3.5)$$

O *kernel* **d4** mostrado na equação (3.5) é utilizado, quando se estima que a direção da borda seja mais próxima da direção da diagonal direita, que parte do canto superior direito da matriz, passa pelo elemento central e termina no canto inferior esquerdo da mesma. Então, os elementos desta diagonal direita apresentam um peso maior e, portanto, maior influência no cálculo da estimativa de **u.** A medida que os elementos vão se afastando da diagonal direita, seu peso vai gradativamente diminuindo. Como nos casos anteriores, mais uma vez, os elementos mais próximos do pixel central apresentam um peso maior do que os elementos mais distantes do mesmo.

É possível que duas ou mais direções apresentem um mesmo valor correspondente à soma dos seus elementos, e este valor é superior ao valor da soma dos elementos nas outras

direções. Neste caso em que há um empate entre duas direções com maior valor, a fim de não priorizar nenhuma direção em detrimento de outra, mas valorizar somente aqueles pixels mais próximos do pixel central da janela, é aplicado o *kernel* **d5** da equação (3.6).

$$d5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (3.6)$$

Uma vez que o *kernel* adequado foi selecionado, ele é passado para a variável **w** e então cada elemento deste *kernel* é multiplicado ao elemento de mesma posição da variável **janela**, que corresponde à janela de filtragem. Os resultados das multiplicações são armazenados em uma nova matriz denominada **pix1**.

Por fim, o valor final do pixel é calculado de maneira similar ao caso da filtragem para os pixels que não constituem borda. Os elementos da matrix **pix1** são somados e o resultado da soma é dividido pela soma dos valores dos elementos de **w**. A diferença neste caso é que a matriz **w** usada nesta etapa da filtragem possui elementos de diferentes valores, que correspondem aos pesos atribuídos aos pixels usados na estimativa do pixel central da janela. Então, ao invés de ser dividida pela quantidade de pixels considerados na estimativa, a soma dos elementos de **pix1** será dividida pela soma dos pesos atribuídos a cada pixel da janela de filtragem pela matriz **w**.

Todo o procedimento de filtragem acima pode ser realizado repetidas vezes para cada imagem, para que seja obtida uma imagem de melhor qualidade. Normalmente, pode-se alterar o valor de algum dos parâmetros de filtragem mencionados ao longo da descrição do funcionamento deste filtro, para cada iteração **p** do mesmo. Particularmente, é comum variar o valor de Δ , uma vez que a quantidade de ruído vai sendo gradativamente reduzida, então o valor de Δ pode ser modificado para se adaptar a esta circunstância.

Os *kernels* apresentados até agora nesta seção apresentam dimensões 5x5. Estas dimensões são particularmente convenientes, uma vez que os filtros sigma, em geral, utilizam janelas de filtragem também de dimensões 5x5, de modo que o fato de tanto a janela quanto

os *kernels* apresentarem o mesmo tamanho facilita a implementação deste filtro. No entanto, em alguns casos, o filtro desenvolvido neste trabalho será simulado com janelas de filtragem de dimensões 3x3. Nestes casos, serão usados *kernels* de filtragem com dimensões também 3x3. Estes *kernels* são diretamente derivados dos *kernels* das equações (3.2) a (3.6), e são obtidos retirando-se as linhas e colunas das bordas mais externas dos *kernels* originais. Assim, os *kernel*s de dimensões 3x3 usados neste projeto são:

$$d1 = \begin{bmatrix} 4 & 16 & 4 \\ 8 & 32 & 8 \\ 4 & 16 & 4 \end{bmatrix} \quad (3.7) \qquad d2 = \begin{bmatrix} 4 & 8 & 4 \\ 16 & 32 & 16 \\ 4 & 8 & 4 \end{bmatrix} \quad (3.8)$$

$$d3 = \begin{bmatrix} 16 & 8 & 2 \\ 8 & 32 & 8 \\ 2 & 8 & 16 \end{bmatrix} \quad (3.9) \qquad d4 = \begin{bmatrix} 2 & 8 & 16 \\ 8 & 32 & 8 \\ 16 & 8 & 2 \end{bmatrix} \quad (3.10) \qquad d5 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (3.11)$$

As etapas de filtragem do algoritmo descrito acima podem ser realizadas tantas vezes quanto se desejar, através de diversas iterações do filtro, até que uma imagem de boa qualidade seja obtida. Durante a seção 4 deste relatório, será demonstrado que existe uma faixa de valores ótima para o número de iterações do filtro. Este número de iterações é definido no início da simulação do algoritmo.

3.3) Parâmetros importantes do filtro

Os parâmetros do filtro *sigmaedge* que serão brevemente descritos a seguir serão variados durante as simulações deste filtro, a fim de avaliar como eles afetam o desempenho do filtro em diversos tipos de imagens e com diferentes valores de relação sinal ruído da imagem.

3.3.1) Média e variância do ruído gaussiano

A média e a variância do ruído gaussiano apresentam influência direta sobre a relação sinal ruído da imagem, após a adição deste ruído. A relação sinal ruído (RSR) da imagem é calculada pela razão entre a variância da imagem e a variância do ruído.

$$RSR = \frac{\sigma^{2}_{imagem}}{\sigma^{2}_{ruído}} \qquad (3.12)$$

Normalmente, esta relação é calculada em decibéis (dB), pela equação (3.13)

$$RSR(dB) = 10\log_{10}\left(\frac{\sigma^{2}_{imagem}}{\sigma^{2}_{ruído}}\right) \qquad (3.13)$$

Para o cálculo desta relação, foi desenvolvida a função "***RSR"*** que calcula a RSR, com base na função ***mediadesvio*** mencionada anteriormente. A função ***RSR*** calcula a média e o desvio padrão da imagem pela função ***mediadesvio*** e, em seguida, calcula a RSR em dB pela equação **(**3.13**)**, utilizando como valor para a variância do ruído o valor que tiver sido designado para este na função principal ***sigmaedge***.

Como pode ser observado das equações acima, quanto maior for o valor da variância do ruído, maior será seu impacto sobre a imagem e maior será a perturbação causada por ele. Um baixo valor de RSR significa maior quantidade de ruído em relação à quantidade de sinal. Logo, quanto mais baixo for o valor de RSR maior será o impacto do ruído sobre a imagem.

3.3.2) Dimensões da janela de filtragem (variáveis m e n)

As variáveis m e n definem as dimensões da janela de filtragem que deslizará sobre a imagem, para a remoção do ruído e o cálculo da estimativa do valor de cada pixel. Quanto maior forem os valores de m e n, obviamente maior será a janela. O aumento da janela implica no aumento da quantidade de pixels que serão considerados na estimativa de cada pixel.

Quanto maior a distância entre dois pixels, menor será a correlação existente entre eles, pois aumentam as chances destes dois pixels fazerem parte de elementos distintos dentro

da imagem. Portanto, na maioria dos casos, não é vantajoso utilizar janelas de filtragem de dimensões muito grandes, ou seja, maiores do que 7x7.

Por outro lado, dentro de uma janela de filtragem muito pequena, a quantidade de pixels ao redor do pixel central considerados no cálculo da sua estimativa torna-se também muito reduzida. Com isso, um pixel que tenha sido fortemente afetado por ruído pode acabar causando grande influência sobre a estimativa do pixel central, vindo a prejudicar os resultados do filtro. Além disso, como o filtro sigma baseia-se sobre uma análise estatística dos valores de intensidade dos pixels dentro da janela de filtragem e considerando-se que análises estatísticas não devem ser aplicadas sobre uma quantidade muito reduzida de dados, logo a janela de filtragem não pode apresentar dimensões muito reduzidas, ou seja, menores do que 3x3.

Com base nas informações acima, e também na análise feita em [6], [7], conclui-se que o uso de $1<m=n<3$ é razoável para os testes deste projeto. Uma vez que a real dimensão destas janelas é de $(2m+1)x(2n+1)$, o uso de $m=n=3$ implica em janelas de filtragem contento 49 pixels, e o uso de $m=n=1$ implica em janelas de filtragem contento 9 pixels.

3.3.3) Valor dos pixels adicionados à janela na função *tratamatriz*

Aparentemente e intuitivamente, o tipo de pixels acrescentado ao redor da imagem para o ajuste das suas dimensões em função das dimensões da janela de filtragem exerce pouca influência no resultado final da imagem filtrada.

Este resultado se deve ao fato das dimensões das janelas de filtragem serem bastante pequenas em relação à dimensão da imagem inteira. Normalmente são acrescentadas 2 ou 3 linhas e colunas de pixels de cada lado da imagem. O efeito destas linhas e colunas é muito pouco visível, principalmente quando analisado dentro do conjunto todo da imagem. Portanto, a aplicação de pixels com valor 0, ou valor 255, ou a repetição das linhas e colunas da imagem não resultam em influências significativas, sem melhorias perceptíveis no desempenho geral do filtro.

3.3.4) O valor de Δ no filtro sigma

Se o valor de Δ for diretamente proporcional ao desvio padrão da imagem, de acordo com a primeira equação do sistema (2.6) do item (2.2.2.3), então a faixa de valores de intensidade delimitada por Δ será maior justamente nas regiões de borda, onde a variância dos pixels é maior, e menor nas regiões planas. Este efeito é exatamente o inverso do que desejamos obter, uma vez que um melhor resultado é alcançado com altos valores de Δ nas regiões planas e baixos valores de Δ nas regiões de borda. Desta forma, a segunda equação do sistema (2.6) do item (2.2.2.3) mostra-se mais adequada para o filtro sigma, uma vez que estabelece uma relação de proporcionalidade inversa entre Δ e σ .

No filtro sigma convencional, o parâmetro delta exerce enorme influência sobre o resultado final. Isto ocorre, porque nas regiões de bordas, o modelo y(i,j) = x(i,j) + n(i,j) para estimar o valor do pixel não é mais válido [6], uma vez que a variância entre os pixels se deve à real diferença de níveis e cinza entre os pixels que compõem à borda.

Então, nas regiões de borda, é esperado que a variância calculada seja bem maior do que nas regiões planas da imagem. Neste caso, se o delta do filtro sigma for escolhido como $\Delta = 2\sigma$, então a faixa de valores de intensidade dos pixels considerados na estimativa do pixel central da janela fica muito grande, e consequentemente os pixels que não fazem parte da mesma classe que o pixel central acabam sendo também considerados na estimativa deste, o que piora o resultado da filtragem. Como resultado, o filtro torna-se basicamente um filtro passa-baixas e as informações de borda acabam se perdendo na imagem final.

A segunda equação usada como alternativa para estimar o valor de delta se propõe justamente a conservar as bordas menos significativas da imagem, usando um delta menor do que a primeira equação e, assim, reduzindo a influência de pixels cuja intensidade é muito deferente da intensidade do pixel central da janela.

3.3.5) O uso de média ou mediana no filtro sigma

O filtro *sigmaedge* será testado utilizando o cálculo da média e a mediana dos pixels vizinhos ao pixel central da janela de filtragem, para a estimativa do mesmo, quando o número M de pixels cuja intensidade está fora da faixa de $\mu \pm \Delta$ for menor do que K.

Como já explicado nos itens (2.2.2.1) e (2.2.2.2) deste relatório, os filtros usando média e mediana podem produzir uma imagem com bordas borradas, reduzindo a nitidez da imagem. Nos testes realizados, serão analisadas as influências do uso da média e da mediana na restauração dos pixels em conjunto com a arquitetura de filtro desenvolvida, para avaliar qual dentre elas produz um melhor resultado, quando aplicada ao filtro *sigmaedge*.

3.3.6) *Kernels* de conservação de borda

Os de conservação de bordas são usados na restauração de pixels identificados como bordas da imagem. Eles têm como finalidade aumentar a influência sobre o pixel de borda restaurado somente daqueles pixels que mais provavelmente pertencem à mesma borda da imagem da qual o pixel que está sendo restaurado faz parte.

Não serão testados *kernels* de conservação diferentes daqueles já descritos durante a explicação do filtro na seção anterior. A única alteração dos *kernels* durante os testes será nos momentos em que as dimensões da janela de filtragem forem modificadas de 5x5 para 3x3.

3.3.7) Número de iterações do filtro

Durante os testes dos filtros sigma e *sigmaegde*, o número de iterações do *loop* de filtragem será variado, com a finalidade de se identificar a quantidade de iterações do filtro que produz um melhor resultado.

Quando é realizada uma quantidade de iterações inferior ao número ótimo, não ocorre a remoção máxima possível de ruído da imagem, principalmente do ruído de fundo. Por ruído de fundo, deve-se entender as partes da imagem em que há presença de pixels que ainda não foram suavizados pelo filtro, e portanto apresentam valores de intensidade muito diferentes e destoantes da média geral da intensidade dos demais pixels naquela região. Por outro lado, quando são feitas mais iterações do que o número ótimo, a imagem começa a perder nitidez, as bordas são menos conservadas, e a qualidade da imagem também piora.

Portanto, a avaliação dos resultados de diversas quantidades de iterações dos filtros busca identificar o número de iterações em que a quantidade de ruído removido for máxima

enquanto as bordas da imagem permanecem conservadas satisfatoriamente, sem causar uma perda muito substancial de nitidez da imagem. Este resultado também deve variar em função da quantidade de bordas fortes ou fracas da imagem, bem como da RSR aplicada.

3.3.8) Sigma e valores de limiar inferior e superior da função *edge* de *Canny*

Estes parâmetros referem-se aos valores de limiar ou *"thresholds"* do método de detecção de bordas de *Canny* [18] explicado no item 2.3.4 deste relatório, e do valor de sigma do filtro gaussiano aplicado à imagem pela função *edge* para a detecção de bordas.

O valor de sigma determina a quantidade de pixels sobre os quais uma borda será testada durante o procedimento de detecção de bordas. Para bordas muito agudas e evidentes, não é necessário espalhar o teste por um número grande de pixels, logo, o valor de sigma não precisa ser muito alto e, de acordo com a literatura, é normalmente utilizado sendo igual a 1. Para bordas menos evidentes em que há variações suaves de intensidade, o método de detecção de bordas de *canny* alcança melhores resultados quando o teste de bordas é espalhado sobre uma maior quantidade de pixels, para que se possa detectar a presença de uma borda. Nestes casos, de acordo com a literatura, é recomendável usar valores em torno de 4 ou 5. Portanto, nos testes do filtro desenvolvido, os valores de sigma serão variados aproximadamente dentro desta faixa de 1 a 5 [17].

É importante notar que a etapa de suavização da imagem pode atenuar algumas bordas fracas, geralmente correspondentes a pequenos detalhes da imagem, e prejudicar a restauração das mesmas na reconstituição da imagem.

Os valores de limiar máximos e mínimos definem a faixa de valores de gradientes dos pixels que serão considerados como borda pelo método de detecção de *Canny*. Quanto maior for o valor de limiar superior, maior deverá ser o valor do gradiente, para que um pixel seja imediatamente identificado como borda. E quanto menor for o valor de limiar inferior, menor deverá ser o valor do gradiente, para que o pixel seja automaticamente identificado como não sendo pixel de borda. Pela sintaxe da função *edge* do Matlab, estes valores podem variar entre 0 e 1, sendo que o limiar superior sempre deve ser maior do que o limiar inferior, obviamente.

3.3.9) Ajuste de intensidade

O ajuste de intensidade da imagem, aumentando-se o contraste entre as intensidades dos pixel da figura, é feito a fim de se obter uma valorização das bordas da imagem, e aumentar a quantidade de bordas identificadas pela função *edge*. É necessário que este ajuste seja feito de maneira cuidadosa, para que os tons da imagem final não sejam muito distorcidos em relação aos tons da imagem inicial.

Os filtros passa-baixas para a restauração de imagens, como é o caso do filtro desenvolvido neste projeto, têm a tendência a diminuir consideravelmente o contraste da imagem. Nota-se pela observação do resultado do filtro sigma comum, principalmente após um número grande de iterações, que os tons de cinza dos diversos elementos da imagem tendem a ficar mais próximos entre si, e as bordas tornam-se menos nítidas.

A fim de guardar a proporcionalidade entre os tons da imagem final e da imagem inicial no ajuste de intensidade, é vantajoso que este seja realizado em função da perda de contraste que efetivamente ocorre durante a filtragem, nas primeiras iterações do filtro, para compensar a perda de contraste das iterações posteriores. Para estimar esta perda de contraste, foram calculados os valores máximos e mínimos de intensidade de cada imagem antes da filtragem e isenta de ruído, e também os valores máximos e mínimos da imagem após a inserção do ruído e algumas iterações de filtragem. Estes valores são passados para a função *imadjust* respectivamente como e os valores máximos e mínimos para quais serão mapeados os valores de entrada e como os valores máximos e mínimos da imagem de entrada.

É esperado que o uso desta técnica de valorização de bordas proporcione um aumento da quantidade de bordas detectadas pelos métodos de detecção de bordas do filtro sigma e também melhore o contraste da imagem resultante da filtragem.

Por outro lado, como esta função aumenta não só o contraste entre as bordas, como também o contraste dos pixels atingidos por ruído, é também possível que o uso desta função acabe aumentando a influência do ruído sobre a imagem.

Capítulo 4

Simulações do filtro e resultados obtidos

Nesta etapa, serão apresentados os testes das imagens mencionadas no item 3.1 deste relatório resultantes de diversas configurações dos filtros sigma e *sigmaedge*, em que os parâmetros dos filtros listados no item 3.3 serão variados. Serão aplicados diferentes valores de relação sinal-ruído às imagens testadas, a fim de se determinar as influências de cada um destes parâmetros sobre diversos tipos de imagens em circunstâncias diferentes. Os resultados obtidos do filtro *sigmaedge* serão comparados entre si e principalmente em relação aos resultados do filtro sigma.

Em algumas situações, a variação de alguns parâmetros gera efeitos muito sutis nos resultados de filtragem das imagens. Estes efeitos são acentuados, ou pelo menos ficam mais visíveis, em função de algumas características particulares de cada imagem, associadas à quantidade de ruído aplicado. Portanto, não serão apresentados neste relatório os resultados das variações de cada um destes parâmetros para cada uma das imagens testadas. Foram selecionados para cada imagem somente os resultados de variações de parâmetros que geravam efeitos mais evidentes e não tão facilmente identificáveis, quando tais parâmetros são aplicados a outras imagens.

Alguns parâmetros foram mantidos constantes em todas as configurações testadas para o filtro. Estas variáveis são:

- o valor de limiar K do *loop* que faz a filtragem sigma foi sempre calculado como sendo o valor mínimo entre (2m+1) e (2n+1). Portanto, o valor de K será modificado somente nos casos em que as dimensões da janela de filtragem forem modificadas. Este é o valor usual para todas as variações de filtros sigma que foram desenvolvidos até o momento.
- A média do ruído gaussiano foi mantida igual a 0.

4.1) Primeiro conjunto de testes – Imagem *Lena.gif*

Primeiramente, serão apresentados os testes da imagem *Lena.gif* com definição de 256x256 pixels com algumas variações de parâmetros do filtro e com valores de relação sinal-ruído variados.

Serão apresentados os resultados usando-se diferentes números de iteração, valores de delta, ajuste de intensidade e alternando media e mediana. Estes resultados serão comparados ao resultado obtido pela filtragem com o filtro sigma comum.

Figura (4.1.1) - Imagem *Lena.gif* original

A Figura (4.1.1) é a imagem *Lena.gif* original

4.1.1) O primeiro conjunto de testes foi realizado usando a variância do ruído gaussiano igual a 0.01, o que implica em uma relação sinal ruído RSR da imagem igual a 46,345 dB. Os demais parâmetros do filtro foram ajustados nos seguintes valores:

Dimensões da janela de filtragem: 5x5 (m=n=2)

Delta do *loop* sigma: $\Delta = \delta \times \max\left[0.2, \frac{1}{1+\frac{\sigma}{\delta}}\right]$ **(**4.1**)**

A equação 4.1 corresponde à segunda equação do sistema de equações (2.6) do item (2.2.2.3) do relatório.

O sigma do filtro usado na função *edge* para detecção de borda foi atribuído como sigma=1. Este valor foi calculado através de testes em que a função de detecção de borda *edge* do Matlab foi simplesmente aplicada à imagem ruidosa com variância 0.01, e dentre os valores de sigma testados, foi escolhido aquele em que a relação entre bordas detectadas corretamente e bordas detectadas incorretamente (quando o ruído é identificado como parte de uma borda) mostrou o melhor resultado. Durante estes testes com esta relação sinal ruído aplicada à imagem, observa-se que valores de sigma superiores a 1 (até cerca de 3.5) também produzem resultados de filtragem igualmente bons. Isto ocorre porque com o valor de RSR testado, a quantidade de ruído na imagem ainda não é tão alta, o que torna a detecção de bordas mais fácil.

Os valores de limiar superior e inferior do filtro da função *edge* mostraram-se mais sensíveis à detecção de ruído como borda do que o valor de sigma. Quando a distância entre os limites inferior e superior é aumentada, a quantidade de bordas erradamente detectadas aumenta muito, principalmente nas regiões planas da imagem. Uma pequena variação nos valores de limiar provoca uma variação substancial na quantidade de bordas detectadas pela função. Para estipular-se um valor adequado para os limiares superior e inferior, foram primeiramente calculados os seus valores médios bem como sua variância ao longo de toda a imagem, na forma de implementação da função *edge* em que estes limites são calculados automaticamente pela função, para cada pixel da imagem. Com base nestes valores, a função *edge* foi aplicada à imagem ruidosa e os valores de limiar superior e inferior foram variados, até que a melhor relação entre bordas detectadas corretamente e bordas detectadas incorretamente fosse encontrada. Estes testes foram realizados juntamente com os testes para a determinação do sigma, de modo que a melhor combinação entre os valores de limiar e o valor de sigma fosse encontrada. Os valores de limiar máximo e mínimo ótimos encontrados para estas condições foram min=0.05 e máx=0.3.

Os testes foram realizados com um número amplo de iterações, variando de 1 até 8 neste caso particular. Os resultados obtidos foram analisados, a fim de encontrar o número ótimo de iterações que produz o melhor resultado. A seguir, serão mostrados alguns resultados a fim de ilustrar o efeito obtido pelo filtro *sigmaedge*, à medida que o número de iterações vai sendo aumentado. Será também feita uma comparação do melhor resultado gerado pelo filtro *sigmaedge* com o resultado obtido pelo filtro sigma comum usando os mesmos valores para os parâmetros que são comuns aos dois filtros.

Figura (4.1.2)

Figura (4.1.3)

Figura (4.1.4)

Figura (4.1.5)

Figura (4.1.6)

Figura (4.1.7)

(4.1.2) Imagem com ruído (RSR = 46,345 dB)

(4.1.3) Imagem resultante de 3 iterações de *sigmaedge*

(4.1.4) Imagem resultante de 5 iterações de *sigmaedge*

(4.1.5) Imagem resultante de 7 iterações de *sigmaedge*

(4.1.6) Imagem resultante de 7 iterações de sigma

(4.1.7) Imagem resultante de 8 iterações de *sigmaedge*

Tabela (4.1.1) – RSR dos resultados obtidos

Figura	Relação Sinal Ruído (em dB)
4.1.3	**46,104**

4.1.4	**46,082**
4.1.5	**46,072**
4.1.6	**46,091**
4.1.7	**46,070**

Nota-se que nos resultados da 3ª e da 5ª iterações do filtro *sigmaedge*, ainda há presença de ruído de fundo sobre a imagem. Somente após 7 iterações, o filtro *sigmaedge* conseguiu remover todo o ruído de fundo. Por outro lado, nota-se que houve perda de nitidez das bordas da imagem no resultado da 7ª iteração, em comparação com os resultados da 3ª e da 5ª iteração. Mesmo assim, após 7 iterações o filtro sigma*edge* foi capaz de conservar as bordas e os tons de todos os elementos da imagem bastante bem, ao mesmo tempo em que removeu todo o ruído de fundo. O resultado da 8ª iteração deste filtro já apresenta piora de nitidez bem como perda de contraste dos tons da imagem, sendo que não há mais remoção de ruído de fundo. Portanto, nestas condições de testes, o uso de 7 iterações parece produzir o melhor resultado da imagem, considerando-se conjuntamente a quantidade de ruído removido e a conservação de bordas alcançada.

Comparando-se o resultado do filtro *sigmaedge* com o resultado do filtro sigma também após 7 iterações, nota-se que o filtro sigma não foi capaz de remover todo o ruído de fundo da imagem após 7 iterações. Tanto as regiões planas da imagem, quanto as regiões de borda apresentaram um melhor resultado quando restauradas pelo filtro *sigmaedge* do que pelo filtro sigma.

O número de iterações ótimo é igual a 7. Neste resultado, podemos notar que há considerável conservação de bordas não tão fortes da imagem, como por exemplo os traços do rosto e do cabelo da Lena, além das suas bordas fortes, como o chapéu e os elementos do cenário à esquerda e à direita da personagem. Ao mesmo tempo, ao compararmos este resultado com os resultados da 1ª e da 4ª iteração do filtro, podemos notar claramente que quase todo o ruído de fundo da imagem foi também removido, ao mesmo tempo em que estas bordas foram conservadas. O resultado de 9 iterações parece ser consideravelmente menos nítido do que o resultado de 7 iterações. As bordas não tão fortes da imagem, como os traços do rosto e do cabelo da Lena, começam a ser bastante atingidas e borradas, mas a um ponto que torna a imagem um pouco distorcida. Além disso, nota-se uma redução do contraste entre os pontos mais claros e os pontos mais escuros da imagem, o que também contribui para a perda de qualidade da mesma.

Quando comparado ao resultado do filtro sigma comum, nota-se que o filtro desenvolvido neste trabalho apresenta resultado consideravelmente superior para este valor de relação sinal ruído.

Nas primeiras iterações, cada vez mais o ruído de fundo vai sendo eliminado, enquanto as bordas principais e secundárias da imagem vão sendo conservadas. Neste caso, mesmo as regiões de textura da imagem são conservadas, como, por exemplo, no chapéu da Lena. Nota-se que a partir de um determinado número de iterações, a imagem começa a perder nitidez e suas bordas tornam-se mais borradas. Em nenhum momento, filtro é capaz de eliminar completamente o ruído de fundo, enquanto as bordas são completamente conservadas. Quanto mais o ruído de fundo é eliminado, mais as bordas tendem a perder a nitidez.

4.1.2) O segundo conjunto de testes foi realizado com a imagem com maior quantidade de ruído, pois a variância do ruído gaussiano foi aumentada para 0.05, implicando em uma RSR de 38,248 dB.

Os testes foram realizados com o filtro nas mesmas configurações do primeiro conjunto de testes, ou seja, janela de filtragem com dimensões 5x5, sigma da função *edge* igual a 1 e valores de limiar superior e inferior iguais a [0.05;0.3]. A escolha dos mesmos parâmetros da função *edge* foi feita com base em diversos testes similares àqueles realizados para o primeiro conjunto de testes, a fim de determinar a combinação ótima destes valores para a imagem com esta relação sinal ruído.

Nesta etapa, serão apresentados os resultados de simulações usando ajuste de intensidade e também em que o valor de delta do *loop* de filtragem sigma foi aumentado, a fim de se avaliar a influência destes parâmetros na restauração da imagem. Novamente, será realizada uma comparação com resultados correspondentes produzidos pelo filtro sigma comum. Os resultados apresentados resultam de 4 iterações dos filtros sigma e *sigmaedge*, pois nestas condições as diferenças entre os resultados gerados pelas duas arquiteturas e os efeitos provocados pelos parâmetros do filtro variados mostraram-se mais evidentes.

Figura (4.1.8)

Figura (4.1.9)

Figura (4.1.10)

Figura (4.1.11)

Figura (4.1.12)

Figura (4.1.13)

(4.1.8) Imagem com ruído – RSR = 38,248

(4.1.9) Imagem resultante de 4 iterações de sigma

(4.1.10) Imagem resultante de 4 iterações *sigmaedge* com ajuste de intensidade

(4.1.11) Imagem resultante de 4 iterações de *sigmaedge*

(4.1.12) Imagem resultante de 4 iterações de *sigmaedge* com $\Delta = 2\sigma$

(4.1.13) Imagem resultante de 7 iterações de *sigmaedge*

Tabela (4.1.2) – RSR dos resultados obtidos

Figura	Relação Sinal Ruído (dB)
4.1.9	**37,939**
4.1.10	**38,031**
4.1.11	**37,929**
4.1.12	**37,945**
4.1.13	**37,912**

Comparando-se a imagem resultante do filtro *sigmaedge* com a imagem gerada pelo filtro sigma após 4 iterações, pode-se perceber que o filtro *sigmaedge* conseguiu remover uma maior quantidade de ruído de fundo do que o filtro sigma, tornando as bordas da imagem mais nítidas. O uso do ajuste de intensidade, neste caso, proporcionou uma pequena melhora de nitidez da imagem em algumas das suas bordas, principalmente no lado esquerdo do rosto da Lena que ficou bastante borrado nos resultados das outras duas configurações de filtro que não usaram o ajuste de intensidade. No entanto, o ajuste de intensidade resulta em uma piora de qualidade das regiões planas da imagem, que tornam-se mais manchadas e com uma

coloração (tons de cinza) bem menos homogênea do que a coloração das regiões planas das imagens filtradas sem ajuste de intensidade.

Analisando-se o resultado decorrente da variação do parâmetro delta, percebe-se que o aumento do valor de delta do filtro *sigmaedge*, que determina a faixa de valores de pixels ao redor do valor do pixel central que serão considerados na estimativa do valor deste pixel, provoca uma mudança considerável no resultado da imagem. Nota-se que as bordas fortes da imagem são muito bem conservadas por esta arquitetura e que o ruído das partes planas da imagem é removido com bastante eficiência. Por outro lado, as bordas mais fracas da imagem praticamente não são conservadas, fazendo com que grande quantidade de detalhes da imagem sejam perdidos. No entanto, estas partes planas ficam com um aspecto bastante diferente do resultado obtido pelos outros filtros. Elas parecem ficar divididas em "pequenos pedaços" cuja tonalidade é bem homogênea, no entanto há uma diferença bastante visível entre as tonalidades de "pedaços" adjacentes. As regiões muito escuras da imagem, em que há grande quantidade de ruído, não foram restauradas por esta configuração do filtro.

Com relação à quantidade ótima de iterações do filtro, novamente após 7 iterações, praticamente todo o ruído de fundo da imagem foi removido, assim como foi o caso da imagem com RSR=46,325. Porém, após este número de iterações, alguns detalhes da imagem foram perdidos e as bordas mais fracas, como, por exemplo, os olhos e a boca da Lena, não ficaram muito nítidos. O resultado obtido após 7 iterações da imagem com RSR=46,325 foi superior com relação à nitidez da imagem. Contudo, embora um número menor de iterações do filtro produza uma imagem com algumas bordas um pouco mais nítidas, esta imagem terá maior quantidade de ruído de fundo, o que também prejudica a sua qualidade.

As figuras (4.1.14) e (4.1.15) mostram os resultados das simulações do filtro *sigmaedge* com as mesmas configurações dos testes anteriores, janela de dimensões 5x5, sem ajuste de intensidade e com delta do *loop* do filtro sigma variando de acordo com a equação (4.1).

A primeira imagem corresponde à simulação usando o valor da mediana dos quatro pixels adjacentes ao pixel central da janela de filtragem para a restauração dos pixels, quando a quantidade de pixels dentro da janela cuja intensidade está dentro da faixa determinada é muito pequena (M<K). A segunda imagem corresponde à mesma simulação usando a

estimativa do pixel central da janela é feita pelo cálculo do valor da média dos quatro pixels adjacentes ao pixel central, quando M<K.

Figura (4.1.14)

Figura (4.1.15)

(4.1.14) Imagem resultante de 3 iterações de *sigmaedge* com mediana

(4.1.15) Imagem resultante de 3 iterações de *sigmaedge* com média

Estes resultados mostram que o uso da mediana na restauração da imagem provocou uma piora grande na conservação tanto das bordas fortes quanto das bordas fracas de toda a imagem. Todas as bordas ficam muito borradas, o que produz o aspecto de que a figura está desfocada. Além disso, houve grande perda de contraste entre os diversos níveis de cinza da figura. Por outro lado, as partes planas da imagem, onde há pouca variação de intensidade entre os pixels, foram melhor restauradas através do uso da mediana do que pela média dos quatro pixels adjacentes ao pixel central. Após apenas 3 iterações do filtro usando a mediana, a remoção de ruído de fundo da imagem foi muito superior à remoção de ruído de fundo do mesmo filtro usando o cálculo da média dos quatro pixels adjacentes ao filtro central da janela de filtragem, para a restauração do mesmo.

Portanto, embora remova menos o ruído de fundo, a configuração usando média aritmética mostra claramente um melhor desempenho do que a configuração usando mediana.

Fazendo uma breve comparação do desempenho do filtro *sigmaedge* para a RSR=46,345, usada no 1º conjunto de testes, e para a RSR=38,248, usada neste conjunto de testes, podemos

notar que os detalhes de textura da imagem, como por exemplo as texturas do chapéu da Lena, quase não são mais conservados após apenas 3 iterações do filtro para uma RSR=38,248. Estes detalhes de textura ainda foram conservados pelo filtro após 3 iterações, quando a imagem tinha RSR=46,345, como pode ser observado na imagem correspondente a 3 iterações do filtro apresentada no 1º conjunto de testes. Portanto, o desempenho do filtro *sigmaedge* para conservação de texturas da imagem sofre uma queda visível, em função da queda de 8 dB da RSR.

4.2) Segundo conjunto de testes – Imagem *Baboon.jpg*

Os resultados a seguir se referem aos testes realizados com a imagem *Baboon.jpg* com definição de 256x256 pixels. Nestes testes, foram variados os parâmetros números de iteração, dimensões da janela de filtragem e dimensões dos *kernel*s de conservação de bordas. Estes resultados serão comparados ao resultado obtido pela filtragem com o filtro sigma comum.

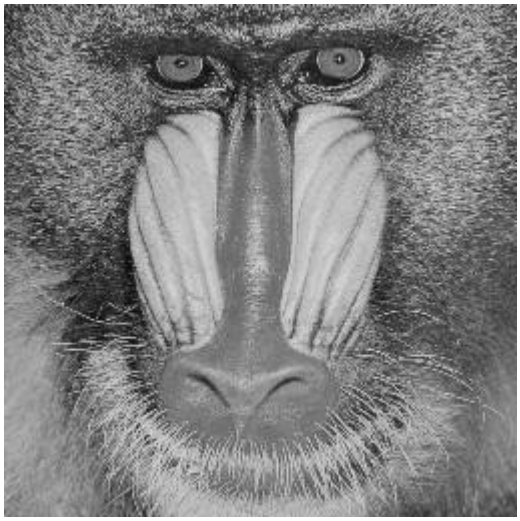


Figura (4.2.1) - Imagem *Baboon.jpg* original

4.2.1) O primeiro conjunto de testes para a imagem *Baboon.jpg* foi realizado usando a variância do ruído gaussiano igual a 0.05, o que implica em uma relação sinal ruído RSR da imagem igual a 38,563 dB. O parâmetro Δ do *loop* do filtro sigma foi calculado de acordo com a equação (4.1).

Os parâmetros da função *edge* foram também selecionados com base em testes similares àqueles descritos na análise de resultados da imagem *Lena.gif*. Os valores selecionados foram sigma=1 e limiares inferior e superior iguais a 0.05 e 0.3, pois esta combinação de valores de sigma e de limiares inferior e superior apresentou melhor resultado na detecção de bordas corretas da imagem e não identificação de ruído como sendo bordas.

Os testes foram novamente realizados com até 8 iterações de filtragem. Os resultados a serem analisados a seguir procuram mostrar a influência da quantidade de iterações do filtro e da variação das dimensões da janela de filtragem sobre a imagem produzida. Os resultados serão também comparados aos resultados do filtro sigma comum, também utilizado uma janela de filtragem com dimensões diferentes.

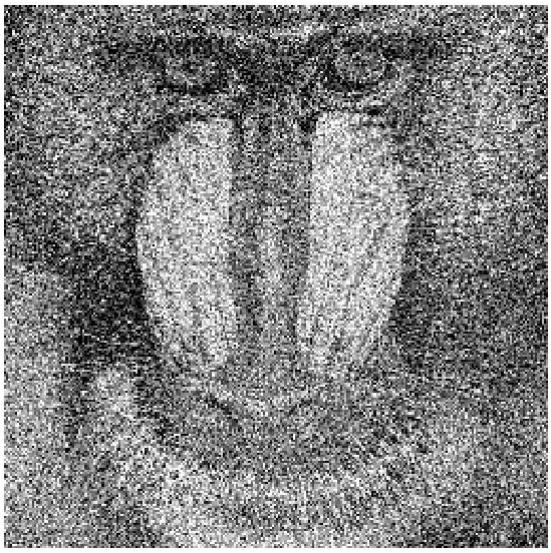


Figura (4.2.2)

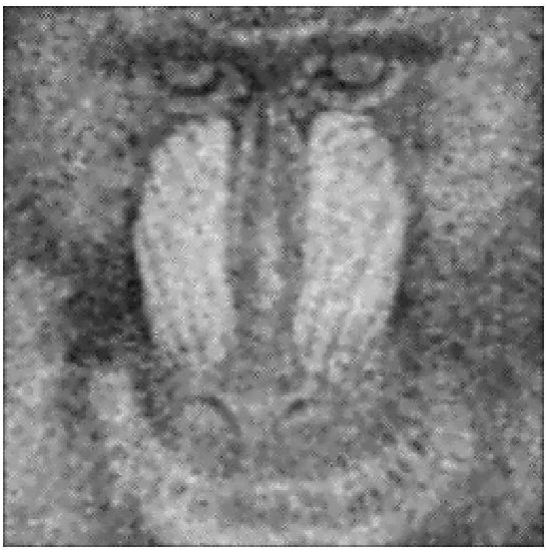


Figura (4.2.3)

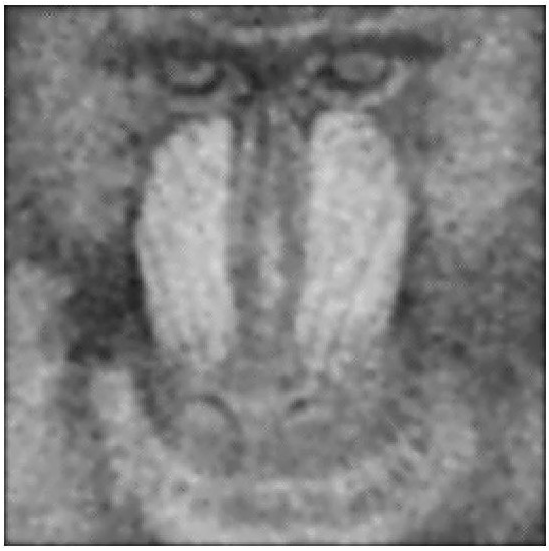


Figura (4.2.4)

Figura (4.2.5)

(4.2.2) Imagem com ruído – RSR = 38, 563 dB

(4.2.3) Imagem resultante de 2 iterações de *sigmaedge* com janela 5x5

(4.2.4) Imagem resultante de 3 iterações de *sigmaedge* com janela 5x5

(4.2.5) Imagem resultante de 3 iterações de *sigmaedge* com janela 3x3

Tabela (4.2.1) – RSR dos resultados obtidos

Figura	Relação Sinal Ruído (em dB)
4.2.3	**37,604**
4.2.4	**37,454**
4.2.5	**37,955**

O aumento de 2 para 3 iterações do filtro *sigmaedge* resulta na remoção de boa parte do ruído de fundo da imagem. Após apenas 3 iterações, este filtro é capaz de remover uma grande quantidade deste ruído de fundo. Por outro lado, o aumento de 3 para 5 iterações produz uma imagem praticamente livre do ruído de fundo, porém em troca, a imagem perde nitidez, havendo ainda redução do seu nível de contraste.

Comparando-se estes resultados ao resultado do filtro *sigmaedge* usando janela de filtragem de dimensões 3x3, percebe-se que janelas de filtragem com dimensões maiores, tipo 5x5, realizam a remoção do ruído de fundo da imagem muito mais rapidamente, ou seja, em um número muito menor de iterações do que as janelas de filtragem menores. A imagem filtrada pela janela de dimensões 3x3 após 3 iterações possui mais ruído de fundo do que a imagem filtrada pela janela de dimensões 5x5 após apenas 2 iterações. Além disso, o uso das

janelas de filtragem maiores faz com que as regiões planas da imagem sejam melhor restauradas, ou seja, fiquem com uma tonalidade mais homogênea, do que pelo uso de janelas de dimensões menores. Por outro lado, a janela de filtragem 3x3 permite maior conservação de pequenos detalhes da imagem e afeta menos o contraste dos tons da imagem. Este resultado é visível principalmente nas regiões de tons mais claros da imagem, como na parte branca do focinho do macaco, ou ao redor da sua boca. Na imagem filtrada com janela de dimensões 5x5, estas regiões apresentam uma coloração mais escura do que as regiões correspondentes da imagem filtrada pela janela de dimensões 3x3.

Figura (4.2.6)

Figura (4.2.7)

Figura (4.2.8)

Figura (4.2.9)

(4.2.6) Imagem resultante de 5 iterações de *sigmaedge* com janela 5x5

(4.2.7) Imagem resultante de 5 iterações de sigma com janela 5x5

(4.2.8) Imagem resultante de 7 iterações de *sigmaedge* com janela 5x5

(4.2.9) Imagem resultante de 7 iterações de sigma com janela 5x5

Tabela (4.2.2) – RSR dos resultados obtidos

Figura	Relação Sinal Ruído (dB)
4.2.6	**37,388**
4.2.7	**37,474**
4.2.8	**37,186**
4.2.9	**37,430**

Quando a variância é maior ou igual a 0.05, nota-se que quase todos os detalhes nas áreas laterais da imagem são removidos ao longo da filtragem, principalmente à medida que o número de iterações aumenta. A técnica de detecção de bordas é capaz de detectar uma quantidade pequena de bordas, da imagem, principalmente as bordas mais fortes, localizadas no centro da imagem do *baboon*. Após algumas iterações, notamos que a imagem perde bastante sua nitidez. Contudo, embora a técnica de detecção de bordas detecte poucas bordas, estas são justamente aquelas mais importantes para a percepção da imagem. Nota-se então que há uma melhora considerável no desempenho do algoritmo *sigmaedge* em relação ao filtro sigma comum, principalmente nas áreas próximas a estas bordas, por exemplo, em volta dos olhos, próximo do nariz e nas bordas da parte branca do focinho, onde nota-se que uma quantidade bem maior de ruído foi removido da imagem, o que deixou-a mais nítida.

Embora estas bordas das regiões mais laterais da imagem do baboon, correspondentes aos pelos do animal, sejam consideradas como regiões de bordas fracas, ainda assim, elas podem ser consideradas como regiões de altas freqüências da imagem. Isto ocorre, pois o olho humano sofre de limitações para enxergar detalhes em freqüências muito altas da imagem, de modo que a perda destes detalhes não são necessariamente um incômodo ou aparentam nitidamente ser uma perda de qualidade muito grande da imagem resultante. Em vista disso, a imagem *baboon.jpg*, após ser filtrada pelo filtro sigma*edge*, não aparenta, à primeira vista, ter sofrido de uma perda de qualidade tão grande. O mais importante na restauração desta imagem é que sejam conservadas as bordas da região central do rosto do animal, representando seus olhos e seu focinho.

Novamente, a imagem gerada por 7 iterações do filtro *sigmaedge* obteve o melhor resultado, em que todo o ruído de fundo foi removido, ao mesmo tempo em que as suas bordas fortes e também algumas bordas fracas foram conservadas, tornando a imagem bastante nítida. Comparando-se este resultado ao resultado de 7 iterações produzido pelo filtro sigma comum, nota-se o melhor desempenho do filtro sigma*edge* na remoção de ruído de fundo e principalmente na conservação das bordas mais fracas da imagem, como por exemplo nas linhas transversais da parte branca do focinho do animal. Estas linhas quase não são identificáveis ou pelo menos estão muito borradas na imagem produzida pelo filtro sigma. De um modo geral, a imagem produzida pelo filtro *sigmaedge* está mais nítida do que a imagem produzida pelo filtro sigma.

4.2.2) O segundo conjunto de testes para a imagem *Baboon.jpg* foi realizado usando a variância do ruído gaussiano igual a 0.1, o que implica em uma relação sinal ruído RSR da imagem igual a 35,251 dB. O parâmetro Δ do *loop* do filtro sigma foi calculado de acordo com a equação (4.1).

Os parâmetros da função *edge* foram também selecionados com base em testes similares àqueles descritos na análise de resultados da imagem *Lena.gif*. Os resultados encontrados para os testes anteriores, em que a quantidade de ruído era mais baixa, não foram satisfatórios para a detecção de bordas da imagem neste caso particular. Para esta RSR, os valores selecionados foram sigma=2 e limiares inferior e superior iguais a 0.15 e 0.35. Este pequeno aumento de todos os valores resultou em um melhor desempenho na detecção de bordas corretas da imagem e não identificação de ruído como sendo bordas.

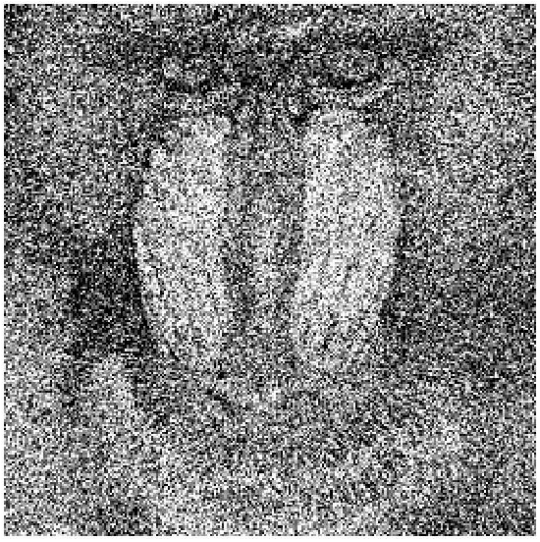


Figura (4.2.10)

Figura (4.2.11)

Figura (4.2.12)

Figura (4.2.13)

Figura (4.2.10) Imagem com ruído – RSR = 35,251 dB

Figura (4.2.11) Imagem resultante de 5 iterações de sigma com janela 5x5

Figura (4.2.12) Imagem resultante de 5 iterações de *sigmaedge* com janela 3x3

Figura (4.2.13) Imagem resultante de 5 iterações de *sigmaedge* com janela 5x5

Tabela (4.2.3) – RSR dos resultados obtidos

Figura	Relação Sinal Ruído (dB)
4.2.11	**33,837**
4.2.12	**34,361**

4.2.13	33,795

O filtro *sigmaedge* proporciona uma melhor conservação das bordas mais fortes da imagem, como, por exemplo, em torno do nariz e da parte branca do focinho do animal, além das duas narinas do animal. As linhas transversais da parte branca do focinho ainda são visíveis na imagem do filtro *sigmaedge* e quase não podem ser identificadas na figura do filtro sigma comum. Algumas bordas mais fracas são ainda visíveis na imagem do filtro *sigmaedge*. Estes resultados comprovam então o melhor desempenho do filtro *sigmaedge* na conservação tanto das bordas mais fortes quanto das bordas mais fracas da imagem. Esta melhora de desempenho neste caso é mais visível do que nos testes feitos com menor quantidade de ruído. A imagem resultante do filtro sigma apresenta ainda algumas manchas cinza, por exemplo, sobre o olho direito e sobre as bordas do nariz e da parte branca do focinho. Este tipo de efeito não é gerado pelo filtro *sigmaedge*.

Comparando-se o resultado, após 5 iterações, de uma janela de filtragem de dimensões 3x3 com o resultado de uma janela de filtragem 5x5, neste caso, observa-se que a imagem resultante da filtragem com janela de dimensões 3x3 contém muito mais ruído do que a imagem filtrada com janela de dimensões 5x5, o que piora muito a sua qualidade final. Em contrapartida, não nota-se uma conservação de bordas muito mais acentuada da imagem resultante da filtragem com a janela menor, em relação à imagem gerada pelo filtro usando a janela de filtragem maior. Portanto, neste caso, o uso de uma janela de dimensões 5x5 mostrou-se muito mais vantajoso do que o uso de uma janela de dimensões 3x3.

Embora as janelas de filtragem menores tenham em certas ocasiões um melhor desempenho na conservação de bordas menos significativas da imagem em relação às janelas maiores, este desempenho superior somente é alcançado para valores de RSR mais altos, quando a quantidade de ruído na imagem é menor. A medida que a quantidade de ruído aumenta, o desempenho do filtro usando janelas de filtragem menores é muito prejudicado, pois a conservação de bordas mais fracas torna-se cada vez mais difícil e o fato destes filtros não serem muito eficientes na remoção de ruído das áreas planas da imagem prejudica muito a qualidade final da imagem nestas circunstâncias de alto ruído. Então, para casos em que a quantidade de ruído na imagem é maior, o uso de janelas de filtragem maiores torna-se ainda mais vantajoso.

4.3) Terceiro conjunto de testes – Imagem *Cameraman.gif*

Os resultados a seguir se referem aos testes realizados com a imagem *Cameraman.gif* com definiço de 256x256 pixels. Nestas simulações foram usados diferentes números de iteração, variações dos parâmetros sigma e valores de limiar da função *edge* e ajuste de intensidade. Estes resultados serão comparados ao resultado obtido pela filtragem com o filtro sigma comum.

Figura (4.3.1) - Imagem *Cameraman.gif* original

4.3.1) O primeiro conjunto de testes para a imagem *cameraman.gif* foi realizado usando a variância do ruído gaussiano igual a 0.01, o que implica em uma relação sinal ruído RSR da imagem igual a 47,413 dB. O parâmetro Δ do *loop* do filtro sigma foi calculado de acordo com a equação (4.1).

Os parâmetros da função *edge* foram também selecionados com base em testes similares àqueles descritos na análise de resultados das imagens *Lena.gif* e *Baboon.gif*. Os valores selecionados foram sigma igual a 1 e limiares inferior e superior iguais a 0.05 e 0.3, pois esta combinação de valores de sigma e de limiares inferior e superior apresentou melhor resultado na detecção de bordas corretas da imagem e não identificação de ruído como sendo bordas.

Os testes foram novamente realizados com até 8 iterações do filtro. Os resultados a serem analisados a seguir procuram mostrar a influência da quantidade de iterações do filtro e da variação das dimensões da janela de filtragem sobre a imagem produzida. Os resultados serão também comparados aos resultados do filtro sigma comum, também utilizado uma janela de filtragem com dimensões diferentes.

Figura (4.3.2)

Figura (4.3.3)

Figura (4.3.4)

Figura (4.3.2) Imagem com ruído 47,413 dB

Figura (4.3.3) Imagem resultante de 4 iterações de *sigmaedge*

Figura (4.3.4) Imagem resultante de 4 iterações do filtro *sigma*

Tabela (4.3.1) – RSR dos resultados obtidos

Figura	Relação Sinal Ruído (dB)
4.3.3	47,350
4.3.4	47,366

As bordas da imagem *cameraman.gif*, principalmente da parte da figura em primeiro plano, são predominantemente bordas fortes. Já os elementos em segundo plano na imagem são bordas predominantemente fracas. Observando-se os resultados obtidos após 4 iterações dos filtros sigma e *sigmaedge*, notamos que as bordas mais fracas da imagem são melhor conservadas pelo filtro *sigmaedge*. Por exemplo, o prédio à direita da imagem, no plano de fundo, é mais nítido e melhor conservado por *sigmaedge*. Os detalhes da câmera também ficam mais nítidos com filtro *sigmaedge.*

Para o caso desta imagem particular, o melhor resultado de filtragem obtido pelo filtro *sigmaedge* foi após apenas 4 iterações, ao invés de 7 iterações como no caso das imagens anteriores. Após 5 iterações do filtro, não há aumento da remoção de ruído da imagem, principalmente pelo filtro *sigmaedge*, e a figura começa a ficar muito borrada por ambos os filtros.

Esta imagem apresenta bordas laterais e superior muito claras. Portanto, quando do tratamento da matriz de imagem pela função *tratamatriz* para ajustar o seu tamanho antes da filtragem, se forem adicionadas linhas e colunas com intensidade 0, correspondendo à cor preta, isto gera uma borda ao redor da imagem que destoa muito da tonalidade real das suas bordas. Neste caso, a imagem fica com qualidade melhor, quando a função *tratamatriz* repete as últimas linhas e colunas em volta da imagem.

4.3.2) O segundo conjunto de testes para a imagem *cameraman.gif* foi realizado usando a variância do ruído gaussiano igual a 0.05, o que implica em uma relação sinal ruído RSR da imagem igual a 39,923 dB. Os demais parâmetros do filtro foram mantidos iguais aos dos testes do item anterior. Os resultados a seguir mostram unicamente o efeito do uso do ajuste de intensidade no filtro *sigmaedge*. Será feita também uma comparação destes resultados com a imagem gerada pelo filtro sigma comum.

Esta imagem apresenta muitas partes totalmente pretas, na região do corpo do cameraman, portanto, o ajuste de intensidade neste caso deve ser feito com os limites máximos da imagem de entrada e da imagem de saída em 1, para que estas regiões pretas sejam conservadas, sem alterar o contraste da imagem e distorcer os tons da imagem original.

Figura (4.3.5)

Figura (4.3.6)

Figura (4.3.7)

Figura (4.3.8)

Figura (4.3.5) Imagem com ruído – RSR = 39,923 dB

Figura (4.3.6) Imagem resultante de 5 iterações de sigma

Figura (4.3.7) Imagem resultante de 5 iterações de *sigmaedge* com ajuste de intensidade

Figura (4.3.8) Imagem resultante de 5 iterações de *sigmaedge* sem ajuste de intensidade

Tabela (4.3.2) – RSR dos resultados obtidos

Figura	Relação Sinal Ruído (dB)
4.3.6	39,317
4.3.7	39,441
4.3.8	39,305

O uso do filtro *sigmaedge* novamente produziu um resultado visivelmente melhor principalmente na remoção de ruído nas regiões próximas das bordas mais fortes da imagem, em torno do corpo do cameraman e do aparelho fotográfico. Além disso, algumas bordas mais fracas do segundo plano da imagem como, por exemplo, do prédio à direita do cameraman, ficaram sensivelmente mais nítidas na filtragem com *sigmaedge* do que com o filtro sigma comum. O uso do ajuste de intensidade melhorou o desempenho do filtro *sigmaedge* com relação às bordas mais fracas no segundo plano da imagem.

O ajuste de intensidade torna algumas bordas menos significativas da imagem mais visíveis do que pela filtragem sem ajuste de intensidade, principalmente à medida que o número de iterações aumenta. Neste exemplo, após 5 iterações do filtro *sigmaedge* sem ajuste de intensidade, o prédio à direita e no fundo da imagem praticamente não é mais visível, pelo menos sua borda superior não pode mais ser identificada, ao passo que na imagem resultante do mesmo número de iterações do filtro sigma*edge* com ajuste de intensidade, este prédio ainda pode ser visto. O pé central do suporte da câmera, assim como algumas bordas da câmera, também ficam bem mais nítidos na imagem com ajuste de intensidade. Por outro lado, o gramado e o céu desta imagem ficam com aspecto mais borrado do que a imagem que não sofre ajuste de intensidade.

4.3.3) O terceiro conjunto de testes para a imagem *cameraman.gif* foi realizado usando a variância do ruído gaussiano igual a 0.1, o que implica em uma relação sinal ruído RSR da imagem igual a 35,266 dB. Os demais parâmetros do filtro foram mantidos iguais aos dos testes do item anterior, à exceção dos parâmetros da função *edge*.

Neste caso, com o aumento do ruído, um melhor resultado de detecção de bordas é alcançado quando aumentamos os valores de limiar superior e inferior da função *edge* de

[0.05 ; 0.3] para [0.1 ; 0.35], enquanto o valor de sigma foi mantido em 1. Este ajuste permite que uma menor quantidade de ruído seja detectada erradamente como borda, ao mesmo tempo em que essencialmente as mesmas bordas que são detectadas com o uso de valores de limiar mais baixo continuam a ser detectadas através dos valores de limiar mais alto. Por outro lado, um aumento unicamente no valor do sigma da função *edge* de 1 para 2, mesmo mantendo-se os valores de limiar mais baixos, diminui bastante a quantidade de ruído detectado como borda, mas também reduz muito a quantidade de bordas menos significativas detectadas pela função *edge*. Por isso que o valor deste parâmetro foi mantido igual a 1.

Os resultados das figuras (4.3.10) e (4.3.11) mostram os resultados alcançados após 4 e 5 iterações dos filtros sigma e *sigmaedge*. Nas figuras (4.3.12) e (4.3.13), serão mostradas partes mais detalhadas das imagens obtidas com as duas arquiteturas de filtro, a fim de demonstrar mais claramente o efeito da técnica de detecção e conservação de bordas quando utilizada em conjunto com um filtro sigma comum, como é o caso da arquitetura de filtro desenvolvida neste trabalho. Nestes testes, os demais parâmetros do filtro não serão variados, a fim de permitir um enfoque maior nos efeitos e vantagens que a nova arquitetura de filtro proposta neste trabalho em relação ao filtro sigma comum.

Figura (4.3.9)

Figura (4.3.10)

Figura (4.3.11)

Figura (4.3.9) Imagem com ruído 35,266 dB

Figura (4.3.10) Imagem resultante de 4 iterações de *sigmaedge*

Figura (4.3.11) Imagem resultante de 4 iterações de sigma

Tabela (4.3.3) – RSR dos resultados obtidos

Figura	Relação Sinal Ruído (dB)
4.3.10)	**34,809**
4.3.11)	**34.867**

Observando-se estas duas imagens (figuras (4.3.10) e (4.3.11)) resultantes de 4 iterações dos filtros sigma e *sigmaedge*, pode-se notar uma remoção de ruído muito maior pelo filtro *sigmaedge* do que pelo filtro sigma, principalmente nas regiões próximas às bordas. Além disso, o *sigmaedge* proporciona uma melhor conservação das bordas mais fracas.

A menor quantidade de ruído, conjuntamente com a restauração das bordas mais eficientes proporcionada pelas técnicas de detecção e conservação de bordas através da função *edge* associada aos *kernel*s de conservação de bordas, torna a imagem mais nítida. A maior nitidez pode ser percebida, por exemplo, no rosto do cameraman e na câmera.

As figuras (4.3.12) e (4.3.13) a seguir mostram a região da cabeça e da câmera do *cameraman* da figura *cameraman.gif* em detalhe ampliado, para mostrar mais claramente os resultados obtidos após 4 iterações dos filtros sigma e *sigmaedge*.

Figura (4.3.12)

Figura (4.3.13)

Figura (4.3.12) Detalhe da imagem resultante de 4 iterações *sigmaedge*
Figura (4.3.13) Detalhe da imagem resultante de 4 iterações sigma

Estas figuras (l) e (m) mais detalhadas dos resultados dos dois filtros após 4 iterações confirmam o melhor desempenho do filtro *sigmaedge* na conservação de bordas e remoção de ruído em relação ao filtro sigma. Nota-se claramente a maior nitidez das bordas da câmera, do rosto e do espaço entre o braço do personagem e a câmera. Além disso, a haste da câmera que se estende por cima da capa do cameraman também apresenta menos ruído e está melhor conservada na imagem. Este detalhe constitui uma região de bordas fracas da imagem, uma vez que a diferença de intensidade entre a capa e a haste é pequena.

Este resultado comprova que as técnicas de detecção e conservação de bordas usadas neste trabalho conseguem realmente prover uma melhora na restauração das bordas fortes e fracas de uma imagem corrompida por ruído.

Figura (4.3.14)

Figura (4.3.15)

Figura (4.3.14) Imagem resultante de 5 iterações de *sigmaedge*
Figura (4.3.15) Imagem resultante de 7 iterações de *sigmaedge*

Após 5 iterações (figura (4.3.14)) é obtido o melhor desempenho relativo entre conservação de bordas e nitidez da imagem e remoção de ruído. A grande maioria do ruído de fundo foi removido, enquanto a imagem conserva-se razoavelmente nítida. O aumento do número de iterações vai tornando a imagem cada vez menos nítida, sem trazer mais nenhuma melhora com relação à remoção de ruído, como pode ser visto do resultado de 7 iterações deste filtro (Figura (4.3.15)). Então, para esta imagem, o número ótimo de iterações foi inferior ao número ótimo de iterações para as outras imagens, mesmo quando a quantidade de ruído adicionado à imagem é aumentado e a RSR é reduzida.

O filtro *sigmaedge* mais uma vez apresentou melhores resultados nas regiões de bordas da imagem, em que uma maior quantidade de ruído foi removida do que pelo filtro sigma comum, principalmente nas regiões de bordas mais fortes. Mesmo o ruído de fundo da imagem em algumas regiões foi removido com mais eficiência pelo *simgaedge* do que pelo filtro sigma comum. Isto pode ter ocorrido nas regiões em que o ruído foi detectado como borda. Quando isto acontece nos casos em que a quantidade de ruído da imagem é maior, normalmente a remoção de ruído tem sido melhor nas áreas detectadas como bordas do que nas áreas não identificadas como bordas.

Os resultados deste conjunto de testes mostraram que a diferença de desempenho do *sigmaedge* em relação ao sigma torna-se mais evidente à medida que o ruído é aumentado.

4.3.4) O quarto conjunto de testes para a imagem *cameraman.gif* foi realizado usando a variância do ruído gaussiano igual a 0.5, o que implica em uma relação sinal ruído RSR da imagem igual a 28,411 dB. Os demais parâmetros do filtro foram mantidos iguais aos dos testes do item anterior, à exceção dos parâmetros da função *edge*. Esta quantidade de ruído já é bastante alta, de modo que as bordas menos significativas no segundo plano da imagem são muito fortemente afetadas pelo ruído.

Neste caso, com o novo aumento do ruído, foi necessário aumentar novamente os valores de limiar superior e inferior da função *edge* de [0.1 ; 0.35] para [0.25 0.4], para se obter um melhor resultado de detecção de bordas. Além disso, o valor de sigma foi aumentado de 1 para 2. Esta combinação de valores proporcionou a melhor identificação de bordas dentre os valores testados.

Abaixo, serão mostrados os resultados, nas figuras (4.3.17) a (4.3.18) de 5 iterações dos filtros sigma e sigma*edge*, as quais serão comparadas entre si, bem como a imagem gerada a partir de 7 iterações do filtro *sigmaedge*.

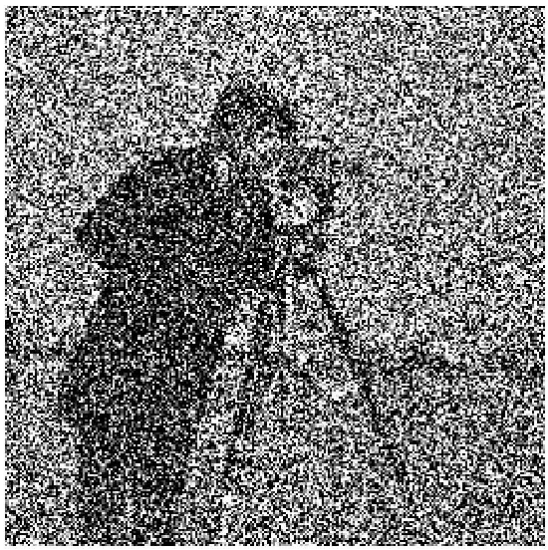


Figura (4.3.16)

Figura (4.3.17)

Figura (4.3.18)

Figura (4.3.19)

Figura (4.3.16) Imagem resultante de com ruído 28,411 dB

Figura (4.3.17) Imagem resultante de 5 iterações do filtro sigma

Figura (4.3.18) Imagem resultante de 7 iterações do *sigmaedge*

Figura (4.3.19) Imagem resultante de 5 iterações de *sigmaedge*

Tabela (4.3.4) – RSR dos resultados obtidos

Figura	Relação Sinal Ruído (dB)
4.3.17	**27.011**
4.3.18	**26,832**
4.3.19	**26,883**

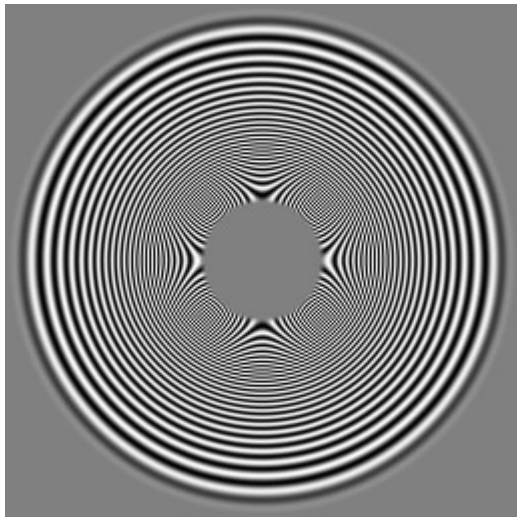
Como pode ser visto das figuras (4.3.16), (4.3.17), (4.3.18) e (4.3.19), a filtragem sigma normal deixa a imagem com muitos borrões em cinza, que não ocorrem na imagem filtrada por *sigmaedge*. Além disso, o filtro *sigmaedge* remove uma quantidade de ruído visivelmente superior tanto nas regiões planas, que ficam menos manchadas, como nas proximidades das bordas da figura. A região do rosto do cameraman, por exemplo, é consideravelmente melhor conservada pelo filtro *sigmaedge*. De um modo geral, o desempenho do filtro *sigmaedge* mostrou-se superior ao do filtro sigma comum.

Após 7 iterações do filtro *sigmaedge*, uma quantidade maior de ruído é removida da imagem, em comparação com o resultado de apenas 5 iterações deste filtro. Mas, embora quase todo o ruído tenha sido removido através das 7 iterações, a imagem perde bastante a nitidez. As bordas menos significativas da imagem são muito pouco conservadas pelo filtro para esta relação sinal ruído.

À medida que a quantidade de ruído sobre a imagem é aumentada, nota-se que o desempenho do filtro *sigmaedge* torna-se cada vez mais superior ao desempenho do filtro sigma. No caso da figura *cameraman.gif*, a diferença entre os resultados de filtragem *sigmaedge* e sigma da imagem com maior quantidade de ruído é bem maior, evidenciando mais claramente o melhor desempenho do filtro *sigmaedge*, do que a diferença entre os resultados das filtragens da imagem com baixa quantidade de ruído.

4.4) Quarto conjunto de testes – Imagem *Cible.gif*

Os resultados a seguir se referem aos testes realizados com a imagem *Cible.gif* com definição de 256x256 pixels. Nestas simulações, foram usados diferentes números de iteração, e foram variados os parâmetros sigma e valores de limiar da função *edge.* Foi também empregado o ajuste de intensidade. Estes resultados serão comparados ao resultado obtido pela filtragem com o filtro sigma comum.



(4.4.1) Imagem cible.gif original

A figura *cible.gif* apresenta muitos componentes em freqüências muito altas, principalmente na região mais central da imagem, bem como componentes em freqüências mais baixas, mais próximo às suas extremidades.

4.4.1) O primeiro conjunto de testes para a imagem *Cible.gif* foi realizado usando a variância do ruído gaussiano igual a 0.01, o que implica em uma relação sinal ruído RSR da imagem igual a 45,135 dB. O parâmetro Δ do *loop* do filtro sigma foi calculado de acordo com a equação (4.1).

Os parâmetros da função *edge* foram também selecionados com base em testes similares àqueles descritos na análise de resultados das imagens *Lena.gif* e *Baboon.gif*. Os valores selecionados foram sigma=0.5 e limiares inferior e superior iguais a 0.05 e 0.3, pois esta combinação de valores de sigma e de limiares inferior e superior apresentou melhor resultado na detecção de bordas corretas da imagem e não identificação de ruído como sendo bordas. No entanto, foram testados outros valores de sigma, e dos valores de limiar, a fim de estudar o efeito dos mesmos nas imagens obtidas.

Os resultados dos testes a serem analisados a seguir procuram mostrar a influência do valor de sigma da função *edge* sobre a conservação de bordas, juntamente com uma janela de filtragem de dimensões 3x3 e com ajuste de intensidade. Os resultados serão também comparados aos resultados do filtro sigma comum, também utilizado uma janela de filtragem com dimensões 3x3.

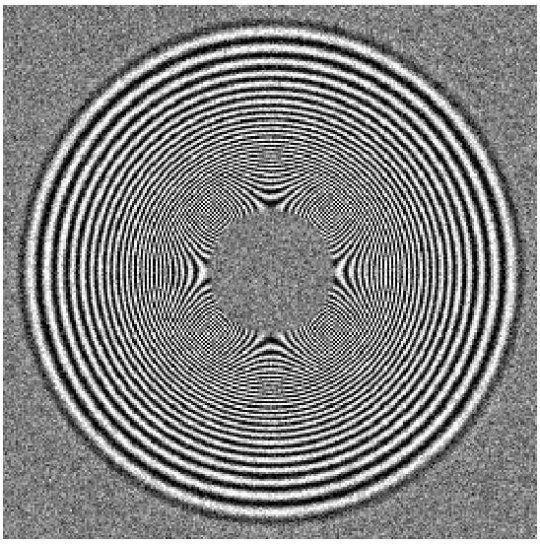


Figura (4.4.2)

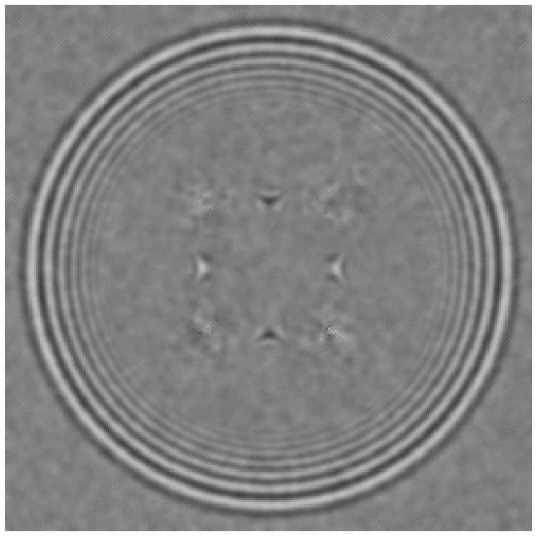


Figura (4.4.3)

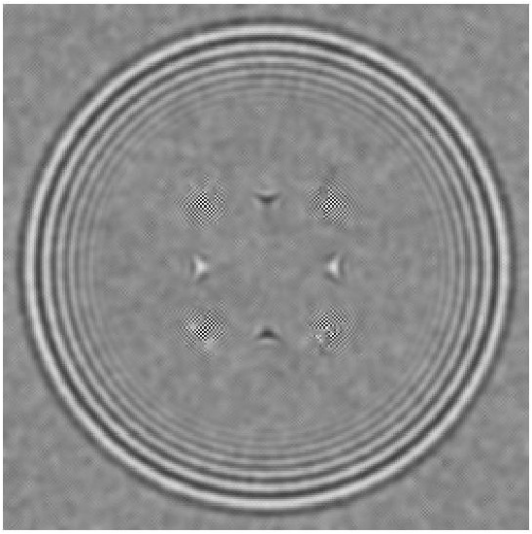


Figura (4.4.4)

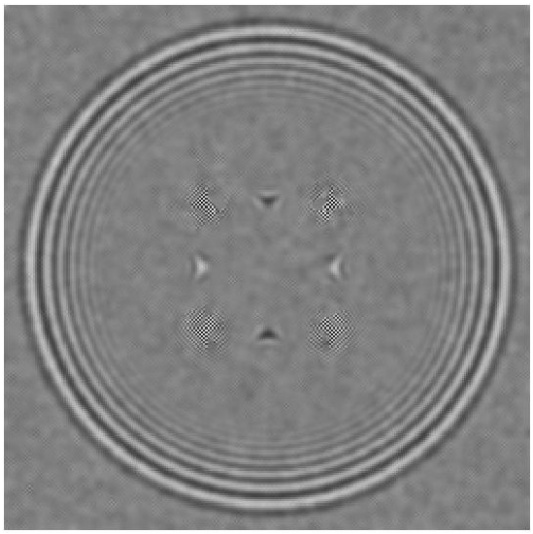


Figura (4.4.5)

Figura (4.4.2) Imagem com ruído – RSR = 45,135 dB

Figura (4.4.3) Imagem resultante de 5 iterações *sigmaedge,* janela de filtragem 3x3 e sigma=0.5 (41,923)

Figura (4.4.4) Imagem resultante de 5 iterações de *sigmaedge,* janela de filtragem 3x3, sigma=1 e ajuste de intensidade

Figura (4.4.5) Imagem resultante de 5 iterações *sigmaedge* , janela de filtragem 3x3 e sigma=1

Tabela (4.4.1) – RSR dos resultados obtidos

Figura	Relação Sinal Ruído (dB)
4.4.3	**41,303**
4.4.4	**42,235**
4.4.5	**41,428**

Pela observação das figuras (4.4.2), (4.4.3), (4.4.4) e (4.4.5), percebe-se que o valor de sigma exerce uma influência notável sobre a imagem gerada nestas configurações de teste. Devido às freqüências muito altas na região próxima ao centro da imagem, inicialmente parece necessário usar um valor de sigma da função *egde* mais baixo. A mudança do valor de sigma de 1 para 0.5 provoca um aumento grande de bordas em altas freqüências detectadas. Nota-se que a quantidade de círculos internos conservados na figura resultante do sigma mais baixo é um pouco maior do que no caso da imagem filtrada usando sigma mais alto (8

círculos pretos para sigma=0.5 contra 7, para sigma=1). Este resultado pode ser visto mais claramente pela figura versão mais detalhada da imagem obtida.

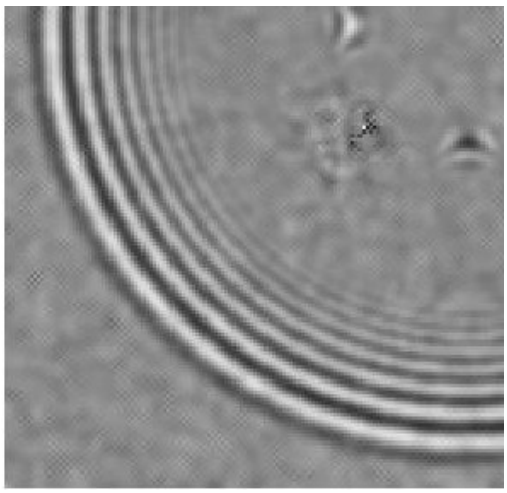


Figura (4.4.3.2)

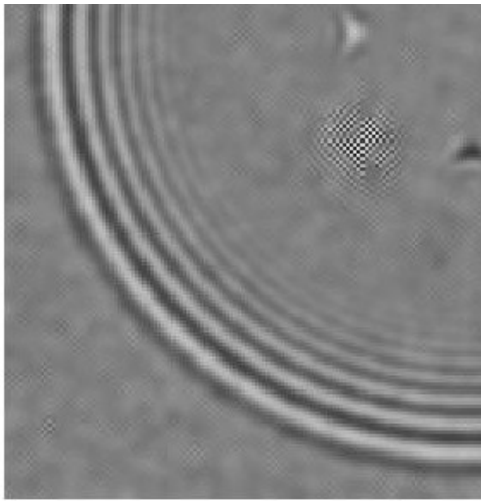


Figura (4.4.5.2)

Figura (4.4.3.2) Imagem resultante de 5 iterações *sigmaedge* com janela 3x3 e sigma=0.5

Figura (4.4.5.2) Imagem resultante de 5 iterações *sigmaedge* com janela 3x3 e sigma=1

Além disso, as bordas em altas freqüências dos detalhes nos cantos superior direito da figura foram também melhor conservados pelo filtro usando sigma=0.05.

Ao mesmo tempo em que a detecção de bordas menos significativas é aumentada para sigma=0.05, é também aumentada a quantidade de ruído detectado erradamente. Para cancelar este efeito, foi aumentado o valor do limiar inferior da função *edge* de 0.05 para 0.1, o que teve como resultado a redução considerável de ruído detectado como borda, enquanto que substancialmente todas as bordas reais detectadas com o valor de limiar inferior mais baixo continuaram a ser identificadas pela função *edge*.

Contudo, aumentando-se a quantidade de bordas em altas freqüências detectadas, o desempenho do filtro é sensivelmente pior do que quando um número menor de bordas em altas freqüências é detectado, especialmente nas regiões de altas freqüências.

Quando o valor de sigma da função *edge* é aumentado de 0.5 para 1, uma quantidade menor de bordas é detectada pela função *edge*, principalmente nas regiões de altas freqüências da imagem, na sua região central. Esta região não compreende propriamente bordas, mas sim diversos pixels variando de intensidade muito rapidamente. Então, quando esta imagem é filtrada com o filtro *sigmaedge* usando um valor de sigma mais alto, de modo a não indentificar estas regiões de altas freqüências como bordas, o resultado de conservação desta região é significativamente melhorado. Isto acontece, porque, pela arquitetura do filtro *sigmaedge*, quando pixels não são identificados como bordas, eles são processados da mesma maneira que por um filtro sigma comum. Uma vez que o filtro sigma comum apresentou um melhor resultado na filtragem desta região de altas freqüências que não são bordas do que o resultado da filtragem dos mesmos pixels quando são identificados como bordas (pelo filtro *sigmaedge*), logo, é esperado que esta combinação de valores do filtro *sigmaedge,* com a variável sigma da função *edge* igual a 1, produza melhores resultados nesta região específica. Por outro lado, há uma queda de desempenho da conservação das bordas menos significativas, ou seja, dos círculos mais centrais da figura, em relação à arquitetura de *sigmaedge* usando um valor de sigma igual a 0.5. Esta queda de desempenho é ainda mais acentuada, à medida que o número de iterações é aumentado, por exemplo, após 5 iterações. O uso de ajuste de intensidade não mostra sensível melhora com relação a estas bordas, após 5 iterações. Pode-se apenas notar ligeira melhora no desempenho das regiões de altas freqüências.

4.4.2) O segundo conjunto de testes para a imagem *Cible.gif* foi realizado usando a variância do ruído gaussiano igual a 0.05, o que implica em uma relação sinal ruído RSR da imagem igual a 45,687 dB. Os demais parâmetros do filtro foram mantidos iguais ao do primeiro conjunto de testes para esta figura, sendo que o valor de sigma da função *edge* foi mantido igual a 1, pois apresentou melhores resultados nos testes anteriores. Os testes a seguir se referem unicamente ao uso do ajuste de intensidade, para 5 iterações do filtro *sigmaedge*, e ao aumento das dimensões da janela de filtragem para 5x5.

O aumento do ruído faz com que os valores de sigma e limiar inferior não sejam adequados à detecção de bordas da imagem com ruído. Estes valores muito baixos causam a detecção de uma quantidade muito grande de ruído como se fossem regiões de bordas, por isso, estes valores foram reajustados de modo a diminuir a identificação de ruídos como sendo bordas, e ao mesmo tempo manter a identificação de bodas que haviam sido corretamente

identificadas. A combinação de valores que mais se adequou a este requisito foi sigma =1 e limiar mínimo =0.15. O limiar máximo foi mantido em 0.3.

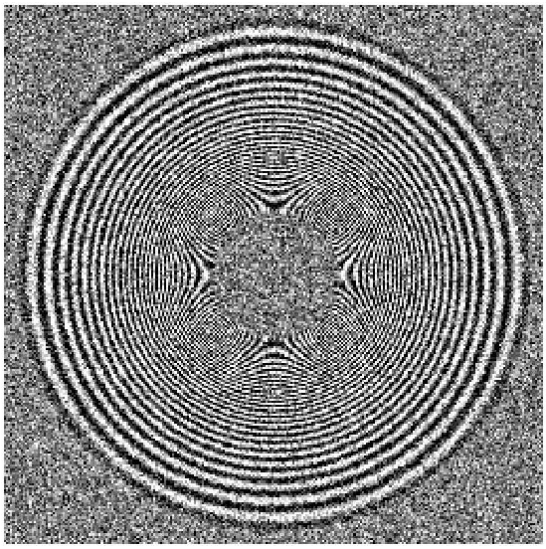


Figura (4.4.6)

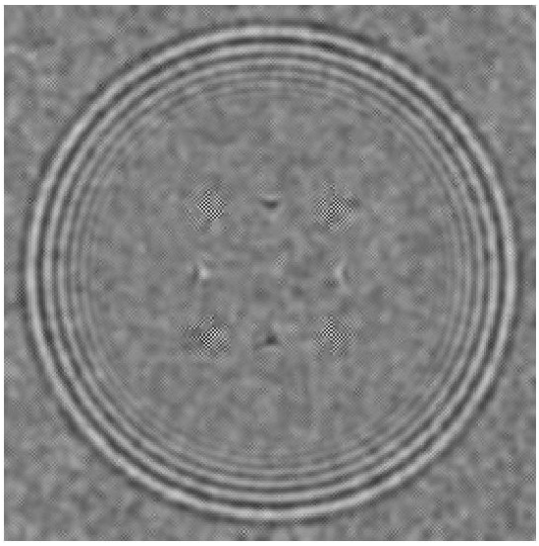


Figura (4.4.7)

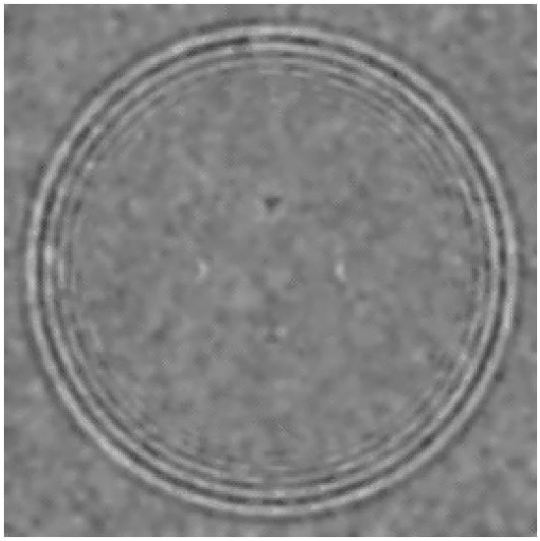


Figura (4.4.8)

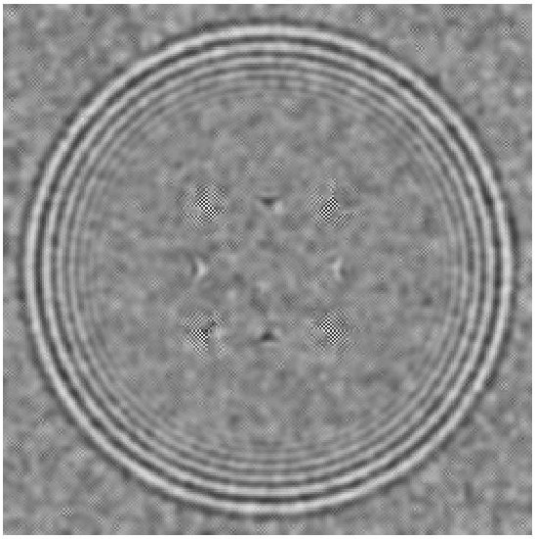


Figura (4.4.9)

Figura (4.4.6) Imagem *Cible.gif* com ruído – RSR = 45,687 dB

Figura (4.4.7) Imagem resultante de 5 iterações sigma com janela de filtragem 3x3

Figura (4.4.8) Imagem resultante de 5 iterações *sigmaedge* com janela de filtragem 5x5

Figura (4.4.9) Imagem resultante de 5 iterações *sigmaedge* com janela de filtragem 3x3 e ajuste de intensidade

Tabela (4.4.2) – RSR dos resultados obtidos

Figura	Relação Sinal Ruído (dB)
4.4.7	43,374
4.4.8	41,181
4.4.9	43,980

Após 5 iterações, o resultado do filtro *sigmaedge* com ajuste de contraste mostra-se melhor do que o resultado do mesmo número de iterações com o filtro sigma comum, pois consegue conservar um maior número de bordas menos significativas, visto que um maior número de círculos internos pode ser visto na imagem resultante do filtro *sigmaedge* do que na imagem resultante do filtro sigma. Além disso, mais uma vez o filtro *sigmaedge* tem um melhor desempenho na remoção de ruído nas regiões próximas às bordas do que o filtro sigma. Este resultado pode ser visto claramente nos círculos brancos da figura após a filtragem *sigmaedge*, que possuem menor quantidade de ruído do que as mesmas regiões da figura resultante da filtragem sigma.

O aumento das dimensões da janela de filtragem para 5x5 mostra-se desastroso neste caso específico, embora produza uma imagem com relação sinal ruído mais baixa do que a imagem produzida com a janela de dimensões 3x3. A imagem resultante ficou muito borrada e a quantidade de círculos internos de bordas menos significativas conservados foi muito reduzida em relação à filtragem com uma janela de dimensões 3x3. Mesmo o uso dos parâmetros da função *edge* que não são capazes de identificar as altas freqüências do centro da imagem como bordas não foi capaz de impedir que estes componentes fossem perdidos pela filtragem com uma janela de dimensões maiores. O valor mais baixo de RSR produzido pela janela de dimensões 5x5 se deve ao fato de uma maior quantidade de ruído de fundo da imagem ser removida do que pelo uso de janela com dimensões 3x3. Neste caso, a RSR não é um bom parâmetro para avaliar comparativamente a qualidade da imagem restaurada pelo filtro.

4.4.3) O terceiro conjunto de testes para a imagem *Cible.gif* foi realizado usando a variância do ruído gaussiano igual a 0.1, o que implica em uma relação sinal ruído RSR da imagem igual a 35,333 dB. Os demais parâmetros do filtro foram mantidos iguais ao do primeiro conjunto de testes para esta figura, com janela de filtragem de dimensões 3x3.

Com o aumento da variância do ruído, os parâmetros usados para a detecção de bordas dos casos anteriores, em que a variância do ruído era mais baixa, não apresentaram desempenho satisfatório na detecção de bordas. Apesar de bordas menos significativas continuarem a ser detectadas, a quantidade de ruído detectada erradamente como borda aumentou. Além disso, muitas bordas foram identificadas na região central da imagem, onde estão localizados os pixels de altas freqüências que não constituem bordas.

Após diversos testes, foi possível detectar que a modificação mais adequada dos parâmetros da função *edge* seria o aumento dos limiares para [0.2 0.4]. O valor de s foi mantido em 1. O aumento dos valores de limiar neste caso foi mais eficiente na seleção das bordas corretas pela função *edge*, do que o aumento do valor de s. Um pequeno aumento de 0.2 no valor de s fez com que uma quantidade grande de bordas menos significativas deixe de ser detectada. O aumento somente dos valores de limiar permite que as bordas menos significativas da imagem continuem a ser detectadas, enquanto uma boa parte do ruído de fundo e dos pixels de altas freqüências do centro da imagem que, na realidade, não constituem bordas, deixam de ser identificados como bordas pela função *edge*.

A seguir serão mostrados resultados que ilustram os efeitos da variação dos valores de limiar da função *edge* sobre a imagem gerada, comparando-se a imagem gerada após 1 iteração do filtro *sigmaedge* com limiares iguais a [0.05 0.3] e [0.2 0.4]. Serão também mostrados resultados gerados pelo aumento do número de iterações do filtro.

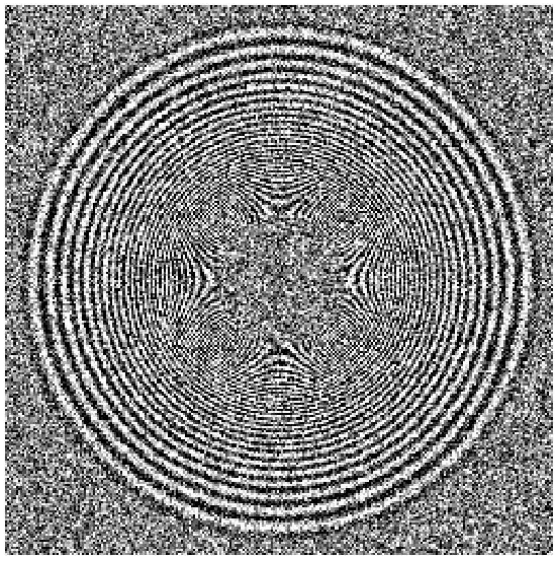


Figura (4.4.10)

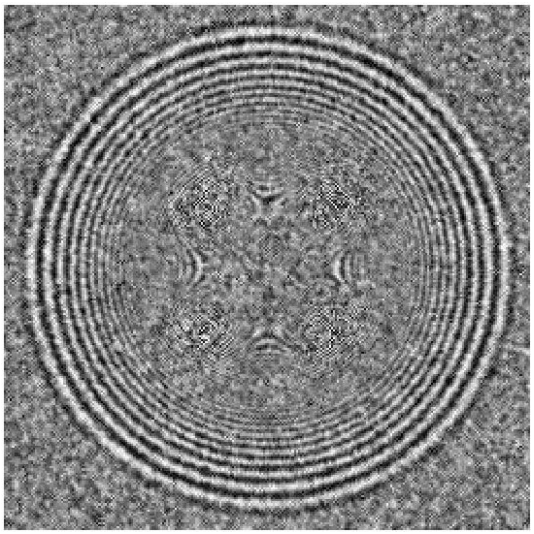


Figura (4.4.11)

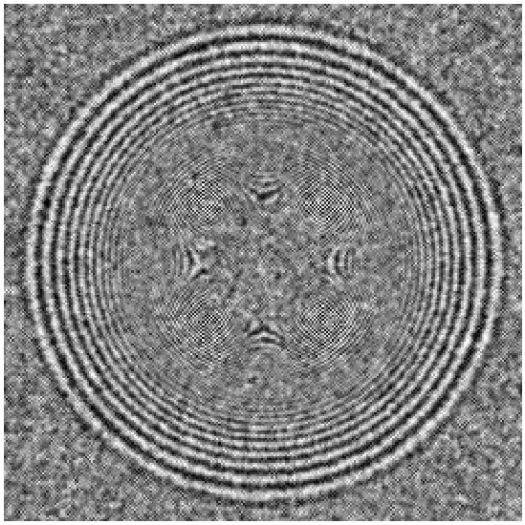


Figura (4.4.12)

Figura (4.4.10) Imagem *Cible.gif* com ruído – RSR = 35,333 dB

Figura (4.4.11) Imagem resultante de 1 iteração do *sigmaedge* com limiares [0.05 0.3]

Figura (4.4.12) Imagem resultante de 1 iteração do *sigmaedge* com limiares [0.2 0.4]

Tabela (4.4.3) – RSR dos resultados obtidos

Figura	Relação Sinal Ruído (dB)
4.4.11	**34,577**
4.4.12	**34,574**

A modificação dos valores de limiar produziu um melhor resultado com relação à conservação dos pixels de altas freqüências localizados mais próximos do centro da imagem. Além disso, a região bem central da imagem, que originalmente é lisa e em tom de cinza médio, parece ter ficado mais manchada na imagem filtrada com os valores de limiar mais baixos do que a imagem resultante do filtro usando valores de limiar mais altos. Por outro lado, após uma iteração apenas, nota-se que uma quantidade menor de bordas menos significativas é conservada pelo filtro com os valores de limiar mais altos do que pelo filtro com os valores de limiar mais baixos iguais a [0.05 e 0.3].

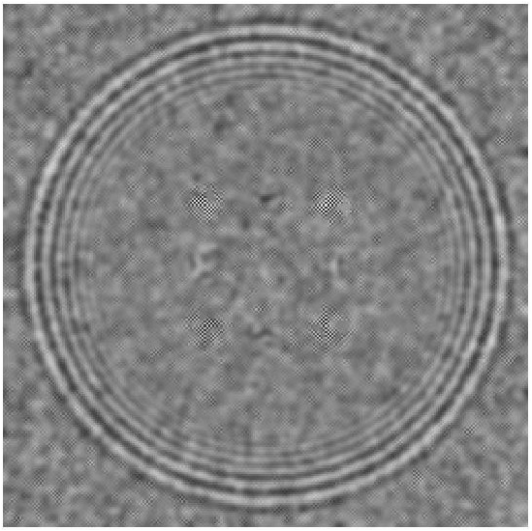


Figura (4.4.13)

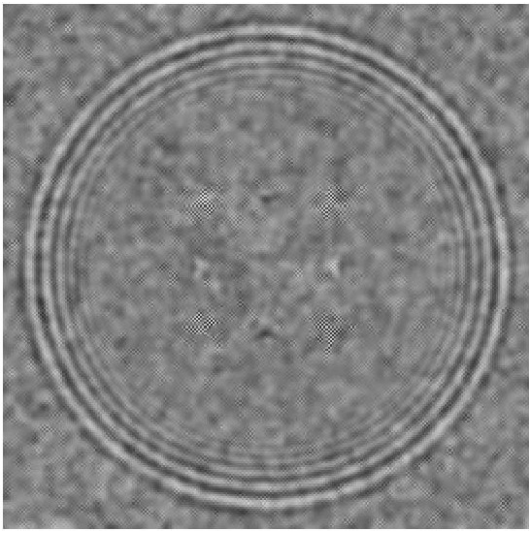


Figura (4.4.14)

Figura (4.4.13) Imagem resultante de 5 iterações do filtro *sigmaedge*

Figura (4.4.14) Imagem resultante de 5 iterações do filtro sigma

Tabela (4.4.4) – RSR dos resultados obtidos

Figura	Relação Sinal Ruído (dB)
4.4.13	32,854
4.4.14	32,581

Após 5 iterações, pode-se notar que o filtro *sigmaedge* produz um melhor resultado de conservação de bordas do que o filtro sigma comum, pois permite que uma maior quantidade de círculos internos seja visualizada na figura gerada. Isto significa que o filtro *sigmaedge* apresenta um melhor desempenho na conservação de bordas menos significativas do que o filtro sigma. Neste caso, não foi usado o ajuste de intensidade. Além disso, nota-se que as bordas mais significativas da imagem foram também melhor conservadas pelo filtro *sigmaedge*, uma vez que visivelmente os círculos brancos mais externos da figura apresentam menos ruído do que no caso da figura gerada pelo filtro sigma. Ao mesmo tempo, as regiões de altas freqüências no centro da figura foram igualmente bem conservadas por ambos os filtros *sigmaedge* e sigma. Portanto, embora o filtro sigma tenha gerado uma imagem com uma RSR mais baixa, o filtro *sigmaedge* produziu um melhor resultado com relação à conservação de bordas e restauração da imagem. A RSR mais alta da imagem gerada por *sigmaedge* se deve ao fato deste filtro ter conseguido conservar mais bordas em altas freqüências, o que provoca um aumento da variância final da imagem e, consequentemente, da sua RSR.

Capítulo 5

Conclusões

A determinação do melhor resultado do filtro pode ser difícil, uma vez que diz respeito a uma melhor relação entre a quantidade de ruído de fundo da imagem e a conservação das bordas da mesma. Nota-se que quanto mais o ruído de fundo das regiões planas da imagem é removido, aumentando-se o número de iterações do filtro, mais as bordas da imagem vão tornando-se borradas, reduzindo sua nitidez. Portanto, é difícil afirmar que um determinado número exato de iterações do filtro produz um melhor resultado, principalmente nos casos em que a RSR é mais baixa e, portanto, a quantidade de ruído na imagem é maior, uma vez que este resultado depende de uma combinação de características da imagem resultante.

Pelos resultados obtidos, podemos notar que o filtro sigma comum, quando usa janelas de filtragem de dimensões 3x3, ou seja, com m=n=1, produz resultados em que as bordas da imagem são melhor conservadas quando comparadas ao filtro que usa janelas de dimensões 5x5, com m=n=2, principalmente à medida que o número de iterações aumenta, como pode ser observado pelos testes da figura *Cible.gif* (figuras (4.4.7), (4.4.8) e (4.4.9)). Por outro lado, o desempenho do filtro com janelas menores para a retirada de ruído das regiões planas da imagem é bem mais fraco do que o desempenho do filtro usando janelas de dimensões maiores. Isto ocorre, porque quando uma janela de dimensões muito pequenas é utilizada, o pixel central da janela é restaurado somente com base na análise dos valores dos 9 pixels que estão em volta dele, que é uma quantidade muito pequena de pixels para a realização de uma análise estatística. Então, se estes vizinhos imediatos do pixel central foram fortemente atingidos por ruído, certamente a estimativa do valor deste pixel calculada também sofrerá forte influência do ruído que incidiu sobre seus vizinhos.

De fato, tal como esperado, o uso da função de ajuste de contraste valoriza as bordas da imagem e, assim, aumenta a quantidade de bordas verdadeiras detectadas pelas técnicas de detecção de bordas. No entanto, à medida que a quantidade de ruído na imagem é aumentada, nota-se que o uso do ajuste de contraste realmente provoca um aumento da influência do ruído na imagem filtrada. Em outras palavras, algumas regiões ruidosas que não ficam muito aparentes nas imagens filtradas sem o ajuste de contrastes tornam-se mais evidentes nas imagens filtradas com o ajuste de imagem. Este efeito é bastante visível no caso da imagem

Cameraman.gif (figuras (4.3.7) e (4.4.8)) ou *Lena.gif* (figuras (4.1.4) e (4.1.5)) com variância do ruído igual a 0.05, em que a imagem final fica mais manchada do que a imagem final filtrada sem ajuste de intensidade.

Por outro lado, o uso do ajuste de intensidade faz com que os tons da imagem resultante da filtragem permanecam mais próximos dos tons da imagem original, principalmente quando a quantidade de ruído na imagem é menor, pois assim a imagem fica menos manchada e a influência do ruído sobre a imagem, embora aumentada, não é capaz de piorar sensivelmente a qualidade da imagem.

No caso da imagem *Cible.gif*, que apresenta grande quantidade de componentes em altíssima freqüência, quando diminuímos a janela de filtragem de dimensões 5x5 para dimensões 3x3, o resultado é melhorado enormemente (figuras (4.4.8) e (4.4.9)). Estas melhorias de resultados se devem ao fato de que os pixels dos componentes de imagem em altíssimas freqüências variam muito rapidamente, em um intervalo menor do que a janela de filtragem de dimensões 5x5. Então, o uso de uma janela de filtragem com dimensões maiores do que o tamanho adequado em vista da rápida variação de intensidade dos pixels acaba por considerar, na estimativa do pixel central, muitos pixels de classes diferentes da classe deste pixel central.

Deste fato, pode-se concluir que, para as regiões de imagem em freqüências muito altas, não necessariamente constituindo bordas, o melhor resultado é realmente obtido pelos filtros com janela de filtragem de dimensões 3x3 (testes da imagem Cible.gif, figuras (4.4.7), (4.4.8) e (4.4.9)). Em particular neste caso, o filtro sigma com janela 3x3 tem resultado melhor do que seu correspondente filtro *sigmaedge* com janela de filtragem de iguais dimensões. Isto ocorre, pois o filtro *sigmaedge* detecta os pixels em altíssimas freqüências como sendo bordas, e então tenta restaurá-lo com base em algum dos *kernel*s, o que acaba causando uma distorção do valor estimado do pixel, formando algumas manchas nestas regiões, efeito este que não acontece quando da aplicação do filtro sigma comum. Por outro lado, nas regiões efetivamente de borda da imagem, mesmo em freqüências mais altas, ou seja, nos círculos mais internos da figura, o filtro *sigmaedge* apresenta um resultado melhor do que o filtro sigma, principalmente quando combinado com o ajuste de intensidade da imagem. Percebe-se que as bordas menos significativas destes círculos mais internos são mais conservadas e permanecem mais nítidas do que aquelas da imagem filtrada pelo filtro sigma

comum,. Nota-se que o ruído nas partes internas de todos os círculos brancos é removido em maior escala pelo filtro *sigmaedge* do que pelo filtro *sigma*. O filtro *sigmaedge* mais uma vez apresentou um melhor desempenho na remoção de ruído próximo de todas as bordas da imagem.

O filtro sigma apresenta um melhor desempenho do que o filtro *sigmaedge* em determinados aspectos, tal como a conservação de regiões de altas freqüências que não constituem propriamente bordas da imagem, como é o caso da região central da figura *cible.gif*. Em outros aspectos, como a conservação de bordas menos significativas da imagem, o filtro *sigmaedge* apresenta um melhor desempenho do que o filtro sigma, como foi demonstrado em diversos testes realizados ao longo deste trabalho. A técnica de detecção de bordas de *Canny* utilizada neste trabalho é uma função das variáveis sigma e valores de limiar máximo e mínimo, que são parâmetros passados para a função *edge*. A variação destes parâmetros permite um controle da quantidade e do tipo de bordas que serão detectadas. Em uma imagem muito ruidosa, pode-se priorizar a detecção de uma grande quantidade de bordas menos significativas e, em troca, uma maior parcela de ruído será também identificada como borda. Por outro lado, pode-se também priorizar a não identificação de ruídos como sendo bordas da imagem. Neste caso, algumas das suas bordas menos significativas também não serão identificadas, de modo que essencialmente as bordas mais fortes da imagem serão conservadas pelo filtro. Como conseqüência, as regiões onde não houver bordas detectadas serão processadas por uma filtragem sigma normal.

O fato de se utilizar, juntamente com o filtro sigma, esta técnica de detecção de bordas que pode ser ajustada em função da quantidade e qualidade de bordas que se deseja detectar permite uma maior flexibilidade de utilização do filtro de imagem proposto em comparação ao filtro sigma comum. Isto se deve ao fato de que a possibilidade de ajuste da quantidade de bordas que se deseja detectar possibilita um controle das regiões da imagem que serão filtradas pelo filtro sigma comum e das regiões que serão filtradas pelo filtro *sigmaedge* com conservação de bordas, visto que somente as regiões que possuem bordas identificadas pela função *edge* serão filtradas pelo filtro *sigmaedge* com conservação de bordas, como foi mostrado pelos testes da figura *cible.gif*.

O ajuste de intensidade realça os efeitos do ruído sobre a imagem. Então, à nas imagens em que há grande quantidade de ruído, a qualidade da imagem piora com o ajuste de intensidade, pois os efeitos provocados pelo ruído ficam ainda mais visíveis.

-

O aumento do valor de delta resulta na piora da detecção de bordas mais fracas da imagem, causando a perda de detalhes e também perda de nitidez da figura. Este resultado é esperado, uma vez que a variável delta é responsável por estabelecer a faixa de valores de intensidade ao redor do valor de intensidade médio da janela de filtragem, que será considerada para a restauração do valor do pixel central desta janela. O aumento desta faixa de valores faz com que diversos pixels cuja intensidade é muito diferente do valor de intensidade do pixel central sejam considerados no cálculo do valor estimado deste pixel. Assim, pixels que não pertencem à mesma classe que o pixel central acabam influenciando o novo valor calculado para o mesmo. As bordas mais fracas da imagem não apresentam uma grande variação de intensidade entre os pixels que a constituem. Algumas vezes, dependendo da quantidade de ruído presente na imagem e também dos parâmetros passados para a função *edge* que realiza a detecção de bordas, estas bordas mais fracas não são nem mesmo identificadas pela função *edge*, de modo que não são processadas pelo filtro *sigmaedge* pelo *loop* de restauração de bordas, mas sim pelo *loop* que faz a filtragem sigma comum. Quando o valor de delta usado no *loop* que faz a filtragem sigma é pequeno, somente são considerados na estimativa do pixel central da janela aqueles pixels cuja intensidade é próxima do valor médio de intensidade da janela, e número de pixels dentro desta faixa de intensidade for muito pequeno, são considerados somente os 4 pixels adjacentes ao pixel central. Desta forma, é possível realizar uma melhor conservação das bordas mais fracas, uma vez que este procedimento permite a seleção dos pixels cujo valor é realmente próximo ao do pixel central da janela de filtragem.

O fato de se diminuir a RSR, aumentando a quantidade de ruído em uma mesma imagem, em geral, não faz com que a quantidade de iterações necessária para a remoção total do ruído de fundo seja aumentada. Em geral, são necessárias entre 5 e 7 iterações, dependendo da imagem, para que esta tarefa seja cumprida. No entanto, nem sempre a qualidade da imagem produzida é boa, quando são realizados os mesmos números de iterações, para diferentes valores de RSR, pois a medida que a RSR diminui, a qualidade da imagem gerada tende a piorar. Além disso, em alguns casos, como por exemplo, para a imagem *Cible.gif*, nem sempre a RSR é o melhor parâmetro para se avaliar comparativamente

o desempenho dos filtros testados neste trabalho, como pode ser visto nas figuras (4.4.11), (4.4.12), (4.4.13) e (4.4.14). Muitas vezes, o fato de uma imagem apresentar uma RSR mais baixa que a outra, significa que ela perdeu também mais informações de altas freqüências importantes, e não necessariamente que teve uma maior quantidade de ruído removido, o que acaba prejudicando a qualidade final da imagem.

A aplicação da mediana mostrou-se mais vantajosa para a restauração das regiões planas da imagem, onde há pouca quantidade de detalhes, ou seja, nas regiões de baixas freqüências. Nestes casos, a remoção de ruído através da mediana foi maior do que quando a estimativa do pixel central é feita através da média dos seus quatro pixels imediatamente adjacentes. Contudo, nota-se que a imagem sofre grande perda de nitidez pelo uso da mediana, como pode ser visto dos testes com a imagem *Lena.gif*. Então, de um modo geral, o uso do cálculo da média dos 4 pixels adjacentes ao pixel central da janela para sua restauração mostra-se mais vantajoso.

É importante notar que nunca duas simulações do programa produzirão resultados exatamente idênticos. Isso ocorre uma vez que o ruído gaussiano é aleatório e, portanto, nunca incide exatamente sobre os mesmos pixels da imagem com a mesma intensidade. Portanto, em alguns casos, não podemos avaliar a qualidade do resultado com base em detalhes muito pequenos da imagem resultante, por exemplo, unicamente com base na boca ou no olho da Lena, pois estes resultados são relativamente aleatórios e dependerão daquela simulação específica do ruído gaussiano sobre a imagem.

Uma sugestão para um trabalho futuro seria o desenvolvimento de um método de remoção de ruído e restauração de imagem que pudesse também ser aplicado a imagens coloridas.

Capítulo 6

Bibliografia

[1] T. Seemann, P. Tischer, *Structure Preserving Noise Filtering of Images using Explicit Local Segmentation*

[2] J. S. Lee, *Digital Image Smoothing And The Sigma Filter,* Comput. Graphics & Image Processing 1983, Vol. 24, pp. 255-26

[3] Knoll, *Radiation, Detection and Measurement*, Wiley Text Books – 3rd Edition, 1989

[4] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering* – Addison-Wesley Publishing Company – 2nd edition, 1994

[5] C. H. Kuo, A. H. Tewfic, *Active Contour Based Rock Sole Recognition*, ICIP00 – Vol II, pp:724-727.

[6] C. H. Kuo, A. H. Tewfic, *Multiscale Sigma Filter And Active Contour For Image Segmentation,* IEEE 1999 – **ICIP99, pp:353-357.**

[7] N. C. Kim, S. H. Jung, *Adaptive Image Restoration Using Local Statistics And Directional Gradient Information,* ELECTRONICS LETTERS, 1987, Vol. 23, pp. 610-611.

[8] S. H. Jung, N. C. Kim, *Adaptive Image Restoration Of Sigma Filter Using local Statistics And Human Visual Characteristics*, ELECTRONICS LETTERS, 18th February 1988 Vol. 24 No. 4.

[9] A. E. Lawabni, A. H. Tewfik, *Image Transmission Over Error-Prone Channels: Sigma Filtering And Multiple Description Objectives,* 2001 IEEE - Electonics Letter 24, pp.201-202.

[10] http://www.icaen.uiowa.edu/~dip/LECTURE/PreProcessing3.html (Acessado em 18.10.2005)

[11]http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MARBLE/low/*edge*s/canny.htm (Acessado em 18.10.2005)

[12] G. M. do Vale, E. A. S. Galvanin, A. P. D. Poz, *O detector de canny-edp: uma combinação entre as teorias de canny e de difusão anisotrópica não linear* - Revista Brasileira de Cartografia Nº 56/02, 2004.

[13] http://www.inf.ufsc.br/~visao/2000/Bordas/ (Acessado em 18.10.2005)

[14]http://me.queensu.ca/people/sellens/research/sprayFlow/mcleod/research/imagepro.htm (Acessado em 18.10.2005)

[15] http://www.pages.drexel.edu/~weg22/can_tut.html (Acessado em 18.10.2005)

[16] http://www.cee.hw.ac.uk/hipr/html/canny.html (Acessado em 18.10.2005)

[17]http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT6/node2.html (Acessado em 18.10.2005)

[18] http://matlabserver.cs.rug.nl (Acessado em 18.10.2005)

[19]http://www.pages.drexel.edu/~pyo22/students/designTeams/kite2001WorkFolder/canny*Edge*Detector.pdf (Acessado em 18.10.2005)

[20] J. R. Price**,** *Method for removing artifacts in an electronic image decoded from a block-transform coded representation of an image* – Patente Americana US 6,427,031

Apêndice 1

Programa *Sigmaedge.m*

```
I= imread('lena.gif','gif');
A = imnoise(I,'gaussian',0,0.01);
% Para cada pixel, calcular o filtro para uma janela (2m+1)x(2n+1)
[maxline,maxcol]=size(A); %numero total de linhas e colunas
m1=(maxline/2)-1;
n1=(maxcol/2)-1;
global m
m=2; % linha da janela de filtragem
global n
n=2; % coluna da janela de filtragem
global K  % valor limite de pixels dentro da faixa
K=min(2*n+1,2*m+1);
% kernels para filtrar pixels de borda
d1=[0 1 4 1 0;0 4 16 4 0;0 8 32 8 0;0 4 16 4 0;0 1 4 1 0];
d2=[0 0 0 0 0;1 4 8 4 1;4 16 32 16 4;1 4 8 4 1;0 0 0 0 0];
d3=[4 4 1 0 0;4 16 8 2 0;1 8 32 8 1;0 2 8 16 4;0 0 1 4 4];
d4=[0 0 1 4 4;0 2 8 16 4;1 8 32 8 1;4 16 8 2 0;4 4 1 0 0];
d5=[0 0 0 0 0;0 0 1 0 0;0 1 2 1 0;0 0 1 0 0;0 0 0 0 0];
% Para definir qual tipo de função tratamatriz será usado
tipo1=2;
tipo2=1;
s=1;
lim=5;

% Tratamento da matriz
for p=1:lim
 [J]=tratamatriz(A,maxline,maxcol,m,n,tipo1);
thres=[0.05 0.3];
   BW=edge(A,'canny',thres,s);
for i=(1+m):(maxline+m)
   for j=(1+n):(maxcol+n)

      % Rotina que define a janela para calcular média e desvio

      for i2=1:(2*m+1)
         for j2=1:(2*n+1)
```

```
            janela(i2,j2)=J((i2+i-m-1),(j2+j-n-1));
        end
    end

    % calculo da media e do desvio
    [mediajan,deltajan,desvio] = mediadesvio(janela,i,j,m,n);
    delta1=deltajan;

    matdesvio(i-m,j-n)=desvio;
    matmedia(i-m,j-n)=mediajan;
    matdelta(i-m,j-n)=deltajan;
    delta1=matdelta(i-m,j-n);
    desvio1=desvio;

if BW(i,j)==0
 % Neste caso, o pixel não pertence a borda, então é feita a filtragem sigma

    [M,count,w1]=compintens(janela,mediajan,delta1,m,n);
    countmat(i,j)=count;
    w=w1;

    % se o numero de pixels estiver fora da faixa, recalcular a média
    if M < K
      pixel=(janela(m+1,n)+janela(m,n+1)+janela(m+1,n+2)+janela(m+2,n+1))/4;
    else

   % rotina para calcular o novo valor do pixel

   % Adaptação que só filtra se o valor do pixel for fora de média +-desvio
       if ((mediajan+delta1)>J(i,j)>(mediajan-delta1))
            pixel=J(i,j);
       else
   % se estiver fora da faixa, calcula normalmente em função de w.

      for i3=1:(2*m+1)
          for j3=1:(2*n+1)
              %pix1(i3,j3)=w(i3,j3)*J((i3+i-m-1),(j3+j-n-1));
              pix1(i3,j3)=w(i3,j3)*janela(i3,j3);

          end
```

```
  end
pix2=sum(pix1,1);
pix3=sum(pix2,2);
den1=sum(w,1);
den2=sum(den1,2);
pixel=pix3/den2;
   end
end
else
 %Neste caso, o pixel pertence a uma borda então é feita a detecção da direção e a aplicação do kernel
% adequado usando kernel para borda

            for j1=1:(2*n+1)
              somah=BW(m,j1);
            end
            for i1=1:(2*m+1)
              somav=BW(i1,n);
            end
            for i1=1:(2*m+1)
                 somade=BW(i1,i1);
                 somadd=BW(i1,2*n+2-i1);
            end

            % calcula a direção onde a soma é maior e escolhe o kernel correspondente

            if somah>=somav
               if somah>=somade
                  if somah>=somadd
                     w=d2;
                  else
                     w=d4;
                  end
               else
                  if somade>=somadd
                    w=d3;
                  else
                    w=d4;
                  end
               end
            elseif (somav>somade)&(somav>somadd)
```

```
                w=d1;
            elseif somade>=somadd
                w=d3;
            elseif (somadd>=somade)
                w=d4;
            else
                w=d5;
            end

            for i3=1:(2*m+1)
              for j3=1:(2*n+1)
                 pix1(i3,j3)=w(i3,j3)*J((i3+i-m-1),(j3+j-n-1));
              end
            end

        % restauraçao da imagem
        pix2=sum(pix1,1);
        pix3=sum(pix2,2);
        den1=sum(w,1);
        den2=sum(den1,2);
        pixel=pix3/den2;

        if den2==0
           den(i,j)=0;
        end
end

        strfinal=[strfinal;pixel]; %pixel eh o novo valor calculado
    end
end
     Ifinal=reshape(strfinal,maxline,maxcol);
% Rotina de ajuste de intensidade
%    if (p==1)
%       A=imadjust(uint8(Ifinal'),[0.07 0.84],[0 0.88]);
 %   else
        A=uint8(Ifinal');
  %  end
end

% fazer uma string com todos os novos valores de pixel e depois reshape
```

```
Ifinal2=uint8(Ifinal');
figure(3);
imshow(Ifinal2);
```

Programa *mediadesvio.m*

```
function [media,delta2,desvio] = mediadesvio(janela,i,j,m,n)
soma1 = sum(janela,1);
soma2 = sum(soma1,2);
K = min(m+1,n+1); %limite de pixels dentro do padrao de intensidade

media = soma2/[(2*n+1)*(2*m+1)];
for i2=1:(2*m+1)
   for j2=1:(2*m+1)
      valorpixel=(janela(i2,j2)-media);
      valor = valorpixel.^2;
      jandelta(i2,j2) = valor;
   end
end

somadelta1 = sum(jandelta,1);
somadelta2 = sum(somadelta1,2);
desvio = sqrt(somadelta2/[(2*n+1)*(2*m+1)]);

%delta1 = 2*desvio;
delta2 = desvio*max(0.2,1/(1+(desvio/2*desvio)));
end
```

Programa compintens.m

```
function [M,count,w]=compintens(image,media,delta1,m,n)
% [m,n]=size(J);
M=0;
count=0;
for i1=1:(2*m+1)
   for j1=1:(2*n+1)
     if ((image(i1,j1)-media)>delta1)|((image(i1,j1)-media)<-delta1);
          w(i1,j1)=0;
          count=count+1;
     else
        w(i1,j1)=1;
        M=M+1;
     end
   end
end
```

Programa *tratamatriz.m*

```
function [L]=tratamatriz(A,maxline,maxcol,m,n,tipo)

for i4=1:(maxline+2*m)
   for j4=1:(maxcol+2*n)
      J(i4,j4)=255;
   end
end

if tipo==1
%Botando intensidade 0 nas linhas em volta
   for i4=1:(maxline+2*m)
      for j4=1:(maxcol+2*n)
         J(i4,j4)=0;
      end
   end
   B(1:maxline,1:maxcol)=A(1:maxline,1:maxcol);
   J(m+1:(maxline+m),n+1:(maxcol+n))=B;
   L=J;
else
% Repetindo linhas e colunas
B(1:maxline,1:maxcol)=A(1:maxline,1:maxcol);
J(m+1:(maxline+m),n+1:(maxcol+n))=B;
%J(m+1:(maxline+m),n+1:(maxcol+n))=A(1:maxline,1:maxcol);
%J(m+1:(maxline+m),n+1:(maxcol+n))=A;
L=J;
for i5=1:m
   for j5=1:(maxcol+2*n)
      L(i5,j5)=J(2*m+1-i5,j5);
   end
end
% Para as últimas m linhas
for i5=(maxline+m+1):(maxline+2*m)
   for j5=1:(maxcol+2*n)
      L(i5,j5)=J(2*maxline+2*m+1-i5,j5);
   end
end

% Para as primeiras n colunas:
```

```
for j5=1:n
   for i5=1:(maxline+2*m)
      L(i5,j5)=J(i5,2*n+1-j5);
   end
end

% Para as últimas n colunas:
for j5=(maxcol+n+1):(maxcol+2*n)
   for i5=1:(maxline+2*m)
      L(i5,j5)=J(i5,2*maxcol+2*n+1-j5);
   end
end
end
```