PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

PPGEE

DISSERTAÇÃO DE MESTRADO

VISUALIZAÇÃO E PROCESSAMENTO DIGITAL DE IMAGENS MÉDICAS

MESTRANDO: Carlos Alberto Zaffari

Área de concentração: Tecnologia da Informação

Linha de Pesquisa: DSP e Engenharia Biomédica

Porto Alegre
2006

CARLOS ALBERTO ZAFFARI

VISUALIZAÇÃO E PROCESSAMENTO DIGITAL DE IMAGENS MÉDICAS

Dissertação apresentada como requisito para obtenção do grau de Mestre, no Programa de Pós-Graduação de Engenharia Elétrica da Pontifícia Universidade Católica do Rio Grande do Sul.

ORIENTADOR: Dario Francisco Guimarães de Azevedo, Ph.D.

CO-ORIENTADOR: Profª. Drª. Thais Russomano

Porto Alegre
2006

VISUALIZAÇÃO E PROCESSAMENTO DIGITAL DE IMAGENS MÉDICAS

Dissertação apresentada como requisito para obtenção do grau de Mestre, no Programa de Pós-Graduação de Engenharia Elétrica da Pontifícia Universidade Católica do Rio Grande do Sul.

Aprovada em _____de_____de_____

BANCA EXAMINADOREA

Profª. Drª. Thais Russomano

Prof. Dr. Paulo Roberto Girardello Franco

Profª. Drª. Ana Maria Marques

RESUMO

Zaffari, Carlos Alberto. Visualização e Processamento Digital de Imagens Médicas

Esta dissertação descreve a teoria envolvida, o projeto e os resultados obtidos com o Visualizador e Processador de Imagens Médicas (VPIM), que é voltado para a área médica e científica. Nela são cobertos o embasamento matemático, os algoritmos usados na construção das ferramentas, o ambiente usado para a implementação, a arquitetura empregada e a descrição dos principais objetos utilizados pelo VPIM. São discutidas, também, as particularidades encontradas durante a fase de implementação do projeto e o processo usado para futuras expansões (plug-in). Os resultados obtidos são apresentados mediante de uso de imagens visando um fácil entendimento dos mesmos.

Entre as funções implementadas podemos ressaltar: a rotação de imagens, o espelhamento de imagens, a negação de uma imagem, os filtros de convolução no espaço (filtro da média móvel, filtro da mediana, filtro de Sobel), os filtros em freqüência (filtros de Butterworth), a obtenção do espectro de potência da imagem, o remapeamento global ou parcial de uma imagem, a aplicação de pseudo-cor, as técnicas de combinação, o *threshold* global, as técnicas de medidas feitas sobre a imagem, as estatísticas obtidas, o histograma, as linhas de perfil, o uso de séries e estudos para apresentação de imagens e a apresentação como filme das imagens de uma série.

ABSTRACT

Zaffari, Carlos Alberto. *Visualizador e Processador de Imagens Médicas*

This text describes the theory, the project and results obtained by the *Visualizador e Processador de Imagens Médicas* (VPIM - in English: Visualization and Processing of the Medical Image). The mathematical base, the algorithms, the software and hardware environments, and the internal software architecture with the main objects of the VPIM are the main topics covered. Particularities of the software implementation and the means to extent the current set of features (through plug-ins) are also discussed. The results are shown mainly through images as a mean to ease its understanding.

Among the main features implemented there are: image rotation, mirroring, the compliment of an image, convolution filters in space-domain (moving average low-pass, median filter, and Sobel), frequency filters (Butterworth filters), the generation of a power-specter representation of an image, total and partial grayscale remapping of an image, the association of pseudo-colors to the scales of gray in an image, image combination techniques, global threshold system, measuring tools for images, histogram, profile lines, and movie-alike image presentation using DICOM Series and Studies in order to sequence images.

Dedico esta dissertação a minha esposa que muito me apoiou durante o mestrado.

AGRADECIMENTOS

Agradeço particularmente ao meu filho Paulo Zaffari pela sua ajuda nas rotinas voltadas para o Windows, e ao meu colega Sérgio Helegada pela sua colaboração na solução de problemas encontrada na implementação do VPIM.

Quero agradecer também a todas as pessoas que com sua opinião ou incentivo me ajudaram a concluir esta dissertação.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Na Medicina moderna existe uma tendência crescente de exames clínicos não-invasivos. Entre eles, pesados investimentos são realizados na área de imageamento. Entre as várias modalidades de imageamento médico encontram-se: ressonância magnética, tomografia computarizada por raios X, ultra-sonografia, endoscopia, tomografia, raios X, angiografia, *Single Photon Emission Computed Tomography* (SPECT) [7], *Positron Emission Tomography* (PET) [7], etc.

Devido ao alto custo dos equipamentos e ao alto índice de especialização no conhecimento para interpretar e gerar os diagnósticos em tais modalidades, estes equipamentos se encontram localizados em grandes centros clínicos, geralmente dispostos em regiões geográficas específicas (como capitais e grandes cidades). Assim, regiões rurais, cidades do interior e mesmo as regiões periféricas das grandes cidades, têm o acesso dificultado a tais recursos. Além disso, os médicos especialistas são em pequeno número, obrigando-se a prestar serviços geralmente em mais de uma instituição de saúde.

Estas necessidades deram origem a uma modalidade relativamente nova da *Engenharia Biomédica*: a *Telemedicina* [3]. A *Telemedicina* surgiu como uma alternativa para minimizar a centralização dos equipamentos e permitir o acesso aos exames não-invasivos a camadas menos favorecidas, inicialmente através da redução de custos pela otimização do tempos dos especialistas. A Telemedicina tem diferentes definições encontradas em diversas fontes. Uma das definições é dada pelo Conselho Federal de Medicina, que define a Telemedicina como:

> "[...] o exercício da Medicina através da utilização de metodologias interativas de comunicação audiovisual e de dados, com o objetivo de assistência, educação e pesquisa em Saúde." [3].

Um dos ramos da Telemedicina é a Teleimagem. Através dela, imagens geradas em diferentes equipamentos (por exemplo: ressonância magnética, tomografia computarizada por raios X, ultra-sonografia, endoscopia, tomografia, raios-X, angiografia, SPECT, PET, etc.) podem ser visualizadas em local distinto dos equipamentos que as geraram. Com a redução de custos dos computadores pessoais, acesso de Internet em banda larga, os médicos

> “[...] podem ter as imagens enviadas diretamente para eles para realizar o diagnóstico, ao invés de se deslocarem para um hospital ou centro clínico.”[2].

2 OBJETIVOS

2.1 Objetivos Gerais

A dissertação tem como objetivo projetar e desenvolver uma ferramenta para ser usada na visualização de imagens médicas, denominada Visualizador e Processador de Imagens Médicas (VPIM). Esta ferramenta permitirá integrar as imagens geradas nas diversas modalidades de imageamento médico ao Sistema de Telemedicina, atualmente em desenvolvimento pelo Instituto de Pesquisa Científica e Tecnológicas na Pontifícia Universidade Católica do Rio Grande do Sul. Através do VPIM será possível disponibilizar imagens aos médicos especialistas independente do local geográfico no qual a imagem foi gerada. Desta forma, camadas não favorecidas da população poderão ter acesso a esses modernos recursos.

2.2 Objetivos Específicos

Os equipamentos atuais que geram imagens médicas usam programas que as converte para um padrão denominado DICOM (*Digital Image Communication in Medicine*)[17]. Este padrão foi criado pela *National Electrical Manufacturers Association* (NEMA). Segundo este padrão, uma imagem tem uma lista de atributos que possuem informações relacionadas com a imagem: informações do paciente (nome, sexo, número de identificação), modalidade e informações sobre a forma de obtenção da imagem (calibração, paramentos do dispositivo, dosagem de radiação, contraste do meio) e informações da imagem (resolução e janelamento).

Atualmente existem diversos programas que permitem visualizar imagens médicas a partir do padrão DICOM, tais como:

- eFilm (usado no Hospital São Lucas da PUCRS - HSL) [20].
- ezDICOM (usado no Hospital São Lucas da PUCRS - HSL) [12].

O objetivo específico desta dissertação foi projetar e desenvolver funções hoje encontradas nos programas disponíveis para uso médico (programas gratuitos), adicionando funções para pesquisa de imagem na área médica (segmentação de imagens, realce de imagens, análise espectral da imagem) desenvolvendo e

implementado os algoritmos usados nestas funções. Também faz parte do objetivo específico desenvolver uma arquitetura que permita a instalação de *plug-ins* visando possibilitar ao VPIM servir de plataforma para implementação de futuras funcionalidades

O VPIM terá, no futuro, a possibilidade de se conectar com um dispositivo externo, a ser desenvolvido pelo IPCT, que permitirá calibrar o monitor de vídeo. **Com este dispositivo e uma função de calibração, a ser desenvolvida no futuro, o VPIM terá a possibilidade de ser usado em diagnósticos.**

Para o desenvolvimento do VPIM será necessário o desenvolvimento de algoritmos e programas que permitam a leitura e a escrita de arquivos no formato DICOM, a visualização destes arquivos e o processamento digital de imagens (melhoramentos, medições, e análises).

3 REVISÃO BIBLIOGRÁFICA

3.1 Exemplos de Visualizadores de Imagens Médicas

Existem vários visualizadores de imagens médicas disponíveis gratuitamente. Os dois visualizadores que foram escolhidos dois para verificação de suas funcionalidade são utilizados no Hospital São Lucas da PUCR (HSL). Por eles já serem usados no HSL, eles possibilitam uma comparação entre os seu resultados e os resultados do VPIM pelos técnicos do hospital. Estes visualizadores são:

- eFilm (usado no Hospital São Lucas da PUCRS - HSL) [20].
- ezDICOM (usado no Hospital São Lucas da PUCRS - HSL) [12].

As funcionalidades existentes no ezDICOM estão presentes no visualizador eFilm. Este também apresenta recursos adicionais não presentes no ezDICOM.

As funcionalidades dos visualizadores estão relacionadas abaixo:

- Leitura, escrita e visualização de arquivos DICOM.
- Ajuste de contraste e luminosidade da imagem
- Visualização das imagens de uma série ou de um conjunto de séries.
- Magnificação de imagem.
- Deslocamento da imagem.
- Visualização de todas as imagens de uma série como se fosse um filme.
- Calibração da imagem para medidas
- Medida de distância entre dois pontos
- Medida de área de uma elipse.
- Espelhamento horizontal e vertical.
- Rotação em 90º (para esquerda e direita).
- Inversão de imagem (negativo).
- Subtração de imagem (angiografia)
- Comparação de FOV (Field of view)
- Dados do estudo
- Possibilidade de inclusão de filtros digitais através de DLL.

As funcionalidades gráficas de (1) magnificação de imagem, (2) deslocamento da imagem, (3) espelhamento horizontal e vertical, (4) rotação em 90º (para esquerda e direita) e (5) inversão da imagem são normalmente encontradas nas bibliotecas gráficas, tais como a OpenGL[1].

O padrão DICOM [17] fornece o suporte para realizar as funcionalidades usadas em medidas (calibração, medida de distância e medida de área de elipse). As funcionalidades específicas do padrão DICOM, tais como: leitura e escrita de arquivos no padrão DICOM, e dados de estudo se encontram implementados na biblioteca DICOM Tool kit DCMTK 3.5.3 [4]. Uma introdução sobre o padrão DICOM encontra-se na Seção 3.7.

3.2 Geometria de Imageamento[6]

Como imagens médicas são projeções de um sistema tridimensional vamos desenvolver algumas transformações básicas num sistema 3-D com coordenadas X, Y e Z. Estas transformações serão: a translação de uma imagem, a mudança de escala, a rotação de um ponto em torno de um dos eixos e a rotação de uma imagem em relação a um ponto arbitrário.

3.2.1 Translação

Para transladar um pixel de uma coordenada (X,Y,Z) para uma nova coordenada final (Xf, Yf, Zf) usando se o deslocamento (Xd, Yd, Zd) usa-se a seguinte expressão:

$$\begin{aligned} Xf &= X + Xd \\ Yf &= Y + Yd \\ Zf &= Z + Zd \end{aligned} \qquad \text{Equação 1}$$

Na forma matricial temos:

$$\begin{bmatrix} Xf \\ Yf \\ Zf \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & Xd \\ 0 & 1 & 0 & Yd \\ 0 & 0 & 1 & Zd \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad \text{Equação 2}$$

A quarta linha, apresentada na Equação 2, foi acrescentada para facilitar as operações com matrizes, transformando a matriz de transformação em uma matriz quadrada.

Usando a notação de matrizes temos:

$$pf = A.pi \qquad \text{Equação 3}$$

Onde A é a matriz transformação 4x4, *pf* é um vetor coluna com as coordenadas após a transformação e *pi* é o vetor coluna com as coordenadas iniciais.

$$pf = \begin{bmatrix} Xf \\ Yf \\ Zf \\ 1 \end{bmatrix} \qquad \text{Equação 4}$$

$$pi = \begin{bmatrix} Xi \\ Yi \\ Zi \\ 1 \end{bmatrix} \qquad \text{Equação 5}$$

Com esta notação a matriz transformação para a translação é:

$$A = T = \begin{bmatrix} 1 & 0 & 0 & Xi \\ 0 & 1 & 0 & Yi \\ 0 & 0 & 1 & Zi \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Equação 6}$$

O processo de translação fica então descrito pela seguinte equação:

$$pf = T.pi \qquad \text{Equação 7}$$

3.2.2 Mudança de escala

A mudança de escala pelos fatores Sx, Sy e Sz ao longo dos eixos X, Y e Z é dado pela matriz de transformação:

$$S = \begin{bmatrix} Sx & 0 & 0 & Xi \\ 0 & Sy & 0 & Yi \\ 0 & 0 & Sz & Zi \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Equação 8}$$

Sendo, então a mudança de escala definida conforme a Equação 9:

$$pf = S.pi \qquad \text{Equação 9}$$

3.2.3 Rotação de um ponto em torno de um dos eixos

Considerando θ o ângulo de rotação em torno do eixo Z, α o ângulo de rotação em torno do eixo Y e β o ângulo de rotação em torno do eixo dos X temos:

a. Para rotação em torno do eixo Z:

$$R\theta = \begin{bmatrix} \cos\theta & sen\theta & 0 & 0 \\ -sen\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Equação 10}$$

Sendo a rotação em torno do eixo Z dada pela Equação 11:

$$pf = R\theta.pi \qquad \text{Equação 11}$$

b. Para rotação em torno do eixo X:

$$R\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & sen\alpha & 0 \\ 0 & -sen\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Equação 12}$$

Sendo a rotação em torno do eixo X dada pela Equação 13:

$$pf = R\alpha.pi \qquad \text{Equação 13}$$

c. Para rotação em torno do eixo Y:

$$R\beta = \begin{bmatrix} \cos\beta & 0 & -sen\beta & 0 \\ 0 & 1 & 0 & 0 \\ sen\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Equação 14}$$

Sendo a rotação em torno do eixo X dada pela Equação 15:

$$pf = R\beta.pi \qquad \text{Equação 15}$$

3.2.4 Rotação de um ponto em torno de outro ponto arbitrário:

Para efetuar a rotação de um ponto em torno de outro ponto arbitrário são necessárias três operações:

1. Translado do ponto arbitrário para origem.

2. Rotação propriamente dita.
3. Translado do ponto para sua posição original.

3.3 Redimensionamento de imagens

Existem diversas técnicas usadas para o aumento e a diminuição das dimensões de uma imagem. A replicação é a técnica mais simples. Atualmente os dispositivos de processamento de imagem (*video adapters*) têm embutidos os algoritmos usados para o redimensionamento de imagem. O DirectX (Versão 9C update: Agosto 2005) verifica se o dispositivo de processamento de imagem é capaz de realizar o redimensionamento usando interpolação anisotrópica. Caso não seja possível usar a interpolação anisotrópica o redimensionamento será realizado usando a interpolação linear em duas dimensões.

3.3.1 Replicação usada na ampliação de imagem

A técnica de replicação [11] é a técnica mais simples usada em ampliação de imagem. Nesta técnica, cada pixel da imagem origem é replicado na imagem destino de acordo com a Equação 16.

$$p_d(i_d, j_d) = p_o(i_o, j_o) \begin{cases} i_o = \left\lceil i_d \cdot \dfrac{L_o}{L_d} \right\rceil & i_d = 1,2...,L_d \\ j_o = \left\lceil j_d \cdot \dfrac{C_o}{C_d} \right\rceil & j_d = 1,2...,C_d \end{cases} \qquad \text{Equação 16}$$

Onde:

(i_o,j_o): coordenadas de um pixel determinado na imagem origem.

(i_d,j_d): coordenadas de um pixel determinado na imagem destino.

$p_o(i_o,j_o)$: nível de cinza da imagem origem em (i_o,j_o).

$p_d(i_d,j_d)$: nível de cinza na imagem destino em (i_d,j_d).

L_o x C_o: número de pixels da imagem origem, sendo L_o o número de linhas e C_o o número de colunas.

L_d x C_d: número de pixels da imagem destino, sendo L_d o número de linhas e C_d o número de colunas.

3.3.2 Replicação usada na redução de imagem

Nesta técnica os pixels da imagem origem são replicados na imagem destino de acordo com a Equação 17.

$$p_d(i_d, j_d) = p_o(i_o, j_o) \begin{cases} i_o = 1 + (i_d - 1).\left(\frac{L_o}{L_d}\right) & i_d = 1,2...,L_d \\ j_o = 1 + (j_d - 1).\left(\frac{C_o}{C_d}\right) & j_d = 1,2...,C_d \end{cases} \quad \text{Equação 17}$$

Onde:

(i_o,j_o): coordenadas de um pixel determinado na imagem origem.

(i_d,j_d): coordenadas de um pixel determinado na imagem destino.

$p_o(i_o,j_o)$: nível de cinza da imagem origem em (i_o,j_o).

$p_d(i_d,j_d)$: nível de cinza na imagem destino em (i_d,j_d).

L_o x C_o: número de pixels da imagem origem, sendo L_o o número de linhas e C_o o número de colunas.

L_d x C_d: número de pixels da imagem destino, sendo L_d o número de linhas e C_d o número de colunas.

3.3.3 Interpolação Linear em 2D usada na ampliação da imagem

A interpolação linear em duas dimensões usada na ampliação da imagem [11] tem este nome porque o seu algoritmo aplica duas interpolações lineares para obter imagem ampliada. A interpolação linear em duas dimensões pode ser descrita pelo algoritmo a seguir.

Considerando uma imagem origem formada por uma matriz de pixels com L_o linhas e C_o colunas, para uma ampliação com um fator de a ($a>1$, inteiro) teremos uma matriz de pixels com $L_d=aL_o$ linhas e $C_d=aC_o$ colunas. Para se obter a matriz destino seguem-se os seguinte passos:

1. Acrescenta-se (a-1) colunas após cada coluna da matriz original.
2. Os níveis de cinza das colunas inseridas são obtidos fazendo-se a interpolação linear com os dois pixels da linha da matriz original entre os quais eles foram inseridos. Neste ponto temos uma matriz intermediária com C_d colunas e L_o linhas

3. Acrescenta-se *(a-1)* linhas após cada linha da matriz intermediária.
4. Os níveis de cinza das linhas inseridas são obtidos fazendo-se a interpolação linear com os dois pixels da coluna da matriz intermediária entre os quais eles foram inseridos. O resultado é a matriz destino com *Ld* linhas e *Cd* colunas

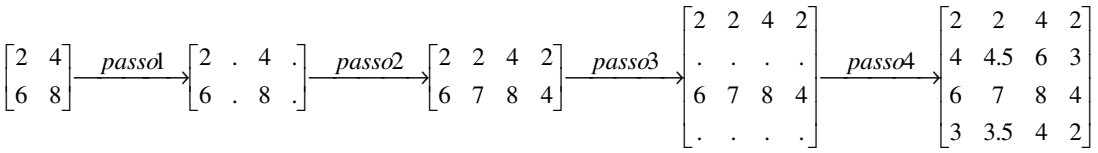


Figura 1: Interpolação linear usada para ampliação de imagens

A Figura 1 mostra um exemplo redimensionamento usando uma matriz 2x2 com uma ampliação *a=2*.

3.3.4 Média em 2D usada para a Redução da Imagem

A média linear em duas dimensões [11] usada para a redução da imagem faz uso de outras duas médias para obter a redução da imagem. A média em duas dimensões usada para a redução da imagem pode ser descrita pelo algoritmo a seguir:

Considerando uma imagem origem formada por uma matriz de pixels com L_o linhas e C_o colunas. Para uma redução de um fator r ($r>1$, inteiro) teremos uma matriz de pixels com $L_d=L_o/r$ linhas e $C_d=C_o/r$ colunas. Para se obter a matriz destino seguem-se os seguintes passos:

1. Obtêm-se as linhas da matriz intermediária, onde cada pixel é igual à soma de *r* elementos sucessivos, da linha correspondente, divididos por *r*, obtendo-se para cada pixel o valor médio dos pixels correspondentes. Neste ponto temos a matriz intermediária com *Lo* linhas e *Cd* colunas.
2. Obtêm-se as colunas da matriz intermediária, onde cada pixel é igual à soma de *r* elementos sucessivos, da coluna correspondente, divididos por *r*, obtendo-se o valor médio dos pixels correspondentes. O resultado é a matriz destino com *Ld* linhas e *Cd* colunas

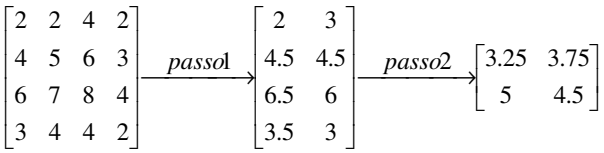


Figura 2: Interpolação de imagens usada na redução de uma imagem

A Figura 2 mostra um exemplo de redimensionamento usando uma matriz 2x2 com uma redução *r=2*.

3.3.5 Interpolação Anisotrópica usada na ampliação da imagem

A interpolação anisotrópica [22] leva em consideração a direção das bordas na vizinhança do pixel interpolado. Um método de executar a interpolação anisotrópica é mostrado a seguir.

Considerando uma determinada imagem na qual se queira interpolar o pixel u_o dos seus quatro vizinhos (u_1,u_2,u_3,u_4), neste caso u_o será dado por:

$$u_0 = \frac{1}{4}(u_1 + u_2 + u_3 + u_4) \qquad \text{Equação 18}$$

Esta interpolação não terá um bom desempenho nas proximidades das bordas, como é mostrado a seguir.

Vamos considerar que temos uma borda na direção u_1-u_3. Neste caso há uma grande variação de níveis de cinza ao longo da linha de u_2 até u_0 e de u_0 até u_4. Esta variação introduzirá um erro de interpolação considerável se for usada a Equação 18. Por outro lado o nível de cinza de u_0 é próximo da media entre u_1 e u_3, assim a melhor interpolação é:

$$u_0 = \frac{1}{2}(u_1 + u_3) \qquad \text{Equação 19}$$

Similarmente se houvesse uma borda entre os pixels u_2-u_4 a melhor interpolação seria:

$$u_0 = \frac{1}{2}(u_2 + u_4) \qquad \text{Equação 20}$$

Quando a borda estiver entre os pixels u_1-u_3 e u_2-u_4, como mostrado na Figura 3, se obtém uma melhor estimação de u_0 atribuindo se pesos para contribuição de cada pixel, com estes pesos inversamente proporcional ao ângulo formado entre u_1-u_3 ou u_2-u_4 em relação à borda, assim, quanto menor o ângulo maior o peso [22].

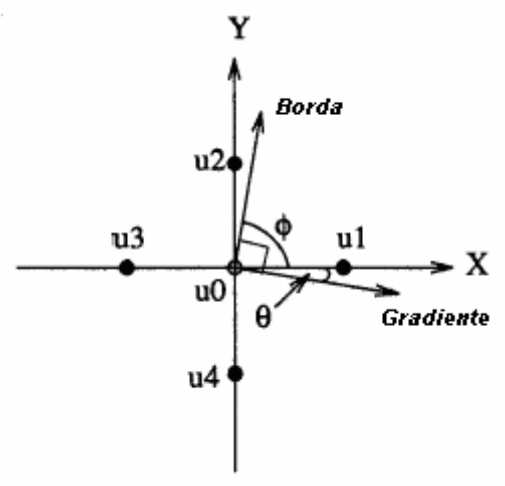


Figura 3: Bordas de direção arbitrária

O cálculo de u_0 será dado pela Equação 21:

$$u_0 = \frac{1}{2}(u_1 + u_3)\cos^2\phi + \frac{1}{2}(u_2 + u_4)\sin^2\phi \qquad \text{Equação 21}$$

O ângulo da borda poderia ser obtido, através da Equação 22:

$$\phi = \frac{\pi}{2} - \theta = \frac{\pi}{2} - \arctan\left(\frac{u_2 - u_4}{u_1 - u_3}\right) \qquad \text{Equação 22}$$

O que nos leva a Equação 23 para o cálculo de u_o:

$$u_0 = \left(\frac{0.5}{1 + \left(\frac{u_2 - u_4}{u_1 - u_3}\right)^2}(u_1 + u_3) + \frac{0.5}{1 + \left(\frac{u_1 - u_3}{u_2 - u_4}\right)^2}(u_2 + u_4) \right) \qquad \text{Equação 23}$$

3.4 Técnicas de Realce da Imagem no Domínio da Freqüência

O realce da imagem no domínio freqüência é executado através da obtenção da transformada de Fourier da imagem a ser realçada, multiplicando-se o resultado por uma função de transferência do filtro desejado. Como pelo teorema da convolução, a multiplicação da imagem no domínio freqüência corresponde a uma convolução executada no domínio espaço, o resultado da transformada inversa desta multiplicação é a imagem realçada [6].

3.4.1 Transformada de Fourier contínua

Para uma função *f(x)* contínua, a transformada de Fourier de *f(x)*, representada por $\Im\{f(x)\}$ e é definida por:

$$F(u) = \Im\{f(x)\} = \int_{-\infty}^{+\infty} f(x) e^{-j2\pi ux} dx \qquad \text{Equação 24}$$

onde $j = \sqrt{-1}$.

Tendo-se uma *F(u)* pode se obter a *f(x)* usando-se a transformada inversa de Fourier definida por:

$$f(x) = \Im^{-1}\{F(u)\} = \int_{-\infty}^{+\infty} F(u) e^{j2\pi ux} du \qquad \text{Equação 25}$$

A Equação 24 e a Equação 25 são chamadas de par de transformadas de Fourier. A relação entre os domínios existe se *f(x)* for contínua e integrável, e se *F(u)* for integrável [6].

A extensão da transformada de Fourier para uma função *f*(x,y) de duas variáveis e sua inversa é dada pela Equação 26 e pela Equação 27, respectivamente [6].

$$F(u,v) = \Im\{f(x,y)\} = \int_{-\infty-\infty}^{+\infty+\infty} f(x,y) e^{-j2\pi(ux+vy)} dxdy \qquad \text{Equação 26}$$

$$f(x,y) = \Im^{-1}\{F(u,v)\} = \int_{-\infty-\infty}^{+\infty+\infty} F(u) e^{j2\pi(ux+vy)} dudv \qquad \text{Equação 27}$$

3.4.2 Transformada de Fourier Discreta (DFT)

Considerando uma função *f(x)*, contínua, discretizada em N amostras separadas de Δx unidades, a sua discretização é representada pela seqüência:

$$\{f(x_0), f(x_0+\Delta x), f(x_0+2\Delta x), ..., f(x_0+(N-1)\Delta x)\}$$

Por conveniência chamaremos de x, uma variável discreta, que assume os valores *x=0, 1, 2, ..N-1*. A seqüência anterior pode ser descrita com uma *f(x)*:

$$f(x) = f(x_0 + x\Delta x) \qquad \text{Equação 28}$$

Assim, a seqüência *{f(0), f(1), f(2), ...f(N-1)}* denota qualquer amostragem de N valores uniformemente espaçados de uma função contínua correspondente. Considerando esta notação, o par de transformadas discretas de Fourier é dado por:

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\, e^{\frac{-j2\pi ux}{N}} \qquad \text{Equação 29}$$

$$f(x) = \sum_{u=0}^{N-1} F(u)\, e^{\frac{j2\pi ux}{N}} \qquad \text{Equação 30}$$

Para *u=0,1,2,...,N-1* e para *x=0,1,2,...,N-1*.

Os valores de *u=0,1,2,,...N-1* na transformada discreta de Fourier (Equação 29) correspondem as amostras de uma transformada contínua nos valores 0, $\Delta u, 2\Delta u, ..., (N-1)\Delta u$. Assim, *F(u*$\Delta$*u)* é representada por *F(u)*. Esta notação é similar à que foi usada para representação discreta de *f(x)* exceto que as amostras de *F(u)* iniciam-se na origem do eixo de freqüências (u_o*=0*). Os termos Δu e Δx tem a seguinte relação:

$$\Delta u = \frac{1}{N\Delta x} \qquad \text{Equação 31}$$

Para duas variáveis o par de transformadas discretas de Fourier pode ser representada da seguinte forma:

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi\left[\left(\frac{ux}{M}+\frac{vy}{N}\right)\right]} \qquad \text{Equação 32}$$

para *u, v=0,1,2,...N-1*, e

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\, e^{j2\pi\left[\left(\frac{ux}{M}+\frac{vy}{N}\right)\right]} \qquad \text{Equação 33}$$

para *x,y=0,1,2,...N-1*.

A amostragem de uma função contínua é feita em uma grade bidimensional com divisões de largura Δx e Δy nos eixos x e y respectivamente. Como no caso unidimensional a função discreta *f(x,y)* representa as amostras da função, mostrada na Equação 34, para *x=0,1,2...,M-1 e v=0,1,2...,N-1*.

$$f(x) = f(x_0 + x\Delta x, y_0 + y\Delta x) \qquad \text{Equação 34}$$

Da mesma forma, *F(u)* que fica:

$$F(u,v) = F(u\Delta u, +v\Delta v) \qquad \text{Equação 35}$$

para *u, v = 0, 1,2..., N-1*.

Os incrementos nos domínios do espaço e freqüência são relacionados por:

$$\Delta u = \frac{1}{M\Delta x} \qquad \text{Equação 36}$$

$$\Delta v = \frac{1}{N\Delta y} \qquad \text{Equação 37}$$

Para imagens amostradas numa matriz quadrada (M=N) temos,

$$F(u,v) = \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi\left(\frac{uv+xy}{N}\right)} \qquad \text{Equação 38}$$

para *u,v = 0,1, 2..., N-1*, e

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F(u,v)\, e^{j2\pi\left(\frac{uv+xy}{N}\right)} \qquad \text{Equação 39}$$

para *x,y = 0, 1, 2, ..., N-1.*

3.4.3 Propriedades da Transformada Bidimensional de Fourier

Nos itens subseqüentes serão apresentadas algumas propriedades da transformada bidimensional de Fourier. Estas propriedades podem ajudar na interpretação do espectro de potência da transformada de Fourier podem ser úteis no cálculo da transformada da própria transformada como é o caso é o caso da propriedade de separabilidade que pode ser usada para calcular a transformada bidimensional como duas transformadas unidimensionais.

3.4.3.1 Separabilidade

A transformada bidimensional de Fourier e sua inversa podem ser expressas em formas separáveis:

$$F(u,v) = \frac{1}{N^2}\sum_{x=0}^{N-1} e^{\frac{-j2\pi ux}{N}} \quad \sum_{y=0}^{N-1} f(x,y)\, e^{\frac{-u2\pi vy}{N}} \qquad \text{Equação 40}$$

para *u, v = 0, 1, ..., N-1.*

$$f(x,y) = \sum_{x=0}^{N-1} e^{\frac{j2\pi ux}{N}} \quad \sum_{y=0}^{N-1} f(x,y)\, e^{\frac{j2\pi vy}{N}} \qquad \text{Equação 41}$$

para *x, y = 0, 1, ..., N-1*.

Esta propriedade pode ser usada na obtenção da transformada bidimensional de Fourier ou da sua inversa através da aplicação de dois passos sucessivos da transformada unidimensional de Fourier ou da sua inversa [6].

Este processo torna-se mais evidente se a Equação 40 for expressa na forma mostrada na Equação 42,

$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1} F(x,v)\, e^{\frac{-j2\pi ux}{N}} \qquad \text{Equação 42}$$

onde *F(x,v)* é dada pela Equação 43.

$$F(x,v) = \frac{1}{N}\sum_{y=0}^{N-1} f(x,y)\, e^{\frac{-j2\pi vy}{N}} \qquad \text{Equação 43}$$

Para cada valor de *x,* a Equação 43 é uma transformada unidimensional com freqüência *v=0, 1, ..., N-1*.

A Figura 4 mostra resumidamente a computação de uma transformada bidimensional através de duas aplicações de uma transformada bidimensional [6].

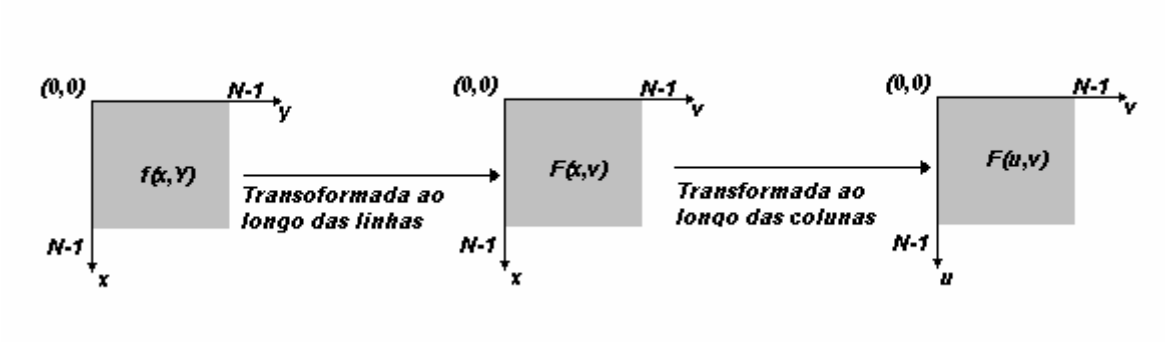


Figura 4: Transformada bidimensional de Fourier usando a propriedade da separabilidade

3.4.3.2 Translação

As propriedades de translação do par de transformadas de Fourier são:

$$f(x,y)\, e^{\frac{j2\pi(u_0x + v_0y)}{N}} \Leftrightarrow F(u-u_0, v-v_0) \qquad \text{Equação 44}$$

$$f(x-x_0, y-y_0) \Leftrightarrow F(u,v)\, e^{\frac{j2\pi(u\,x_0 + v\,y_0)}{N}} \qquad \text{Equação 45}$$

As flechas duplas indicam a correspondência entre uma função e sua transformada de Fourier (e vice-versa).

A Equação 44 mostra que a transformada de Fourier, resultante do produto da $f(x,y)$ pelo termo exponencial indicado, resulta no deslocamento da origem do plano das freqüências para o ponto (u_0,v_0). Similarmente, na Equação 45, o produto de $F(u,v)$ pelo termo exponencial indicado, mostra que a transformada inversa será deslocada para (x_0,y_0) [6].

3.4.3.3 Periodicidade e Simetria Conjugada

A transformada de Fourier e sua inversa apresentam uma periodicidade de período N. Assim temos:

$$F(u,v) = F(u+N,v) = F(u,v+N) = F(u+N,v+N) \qquad \text{Equação 46}$$

Se $f(x,y)$ for real, que é o caso das imagens em níveis de cinza, a transformada de Fourier também apresenta simetria conjugada, mostrada na Equação 47.

$$F(u,v) = F^*(-u,-v) \qquad \text{Equação 47}$$

Onde $F^*(u,v)$ representa o conjugado complexo de $F(u,v)$. Da Equação 47 se tira a relação entre os valores absolutos dados pela Equação 48 [6].

$$\left|F(u,v)\right| = \left|F(-u,-v)\right| \qquad \text{Equação 48}$$

A exibição da magnitude da transformada de Fourier é muitas vezes interessante. Para se entender melhor a implicação do significado da Equação 46 e da Equação 48, pode-se examinar a exibição da magnitude da transformada no caso de uma variável,

$$F(u) = F(u+N) \qquad \text{Equação 49}$$

e,

$$\left|F(u)\right| = \left|F(-u)\right| \qquad \text{Equação 50}$$

A propriedade da periodicidade indica que $F(u)$ tem um período N. A propriedade da simetria mostra que a magnitude da transformada está concentrada na origem. Isto pode ser observado na Figura 5. Nesta figura, pode-se observar que a magnitude dos valores da transformada de $(N/2)+1$ até $N-1$ consiste na repetição dos valores do meio período a esquerda da origem. Como a transformada de Fourier foi formulada com valores de u entre $[0, N-1]$, o resultado desta formulação implica em um período com dois semi-períodos antepostos um para o outro [6].

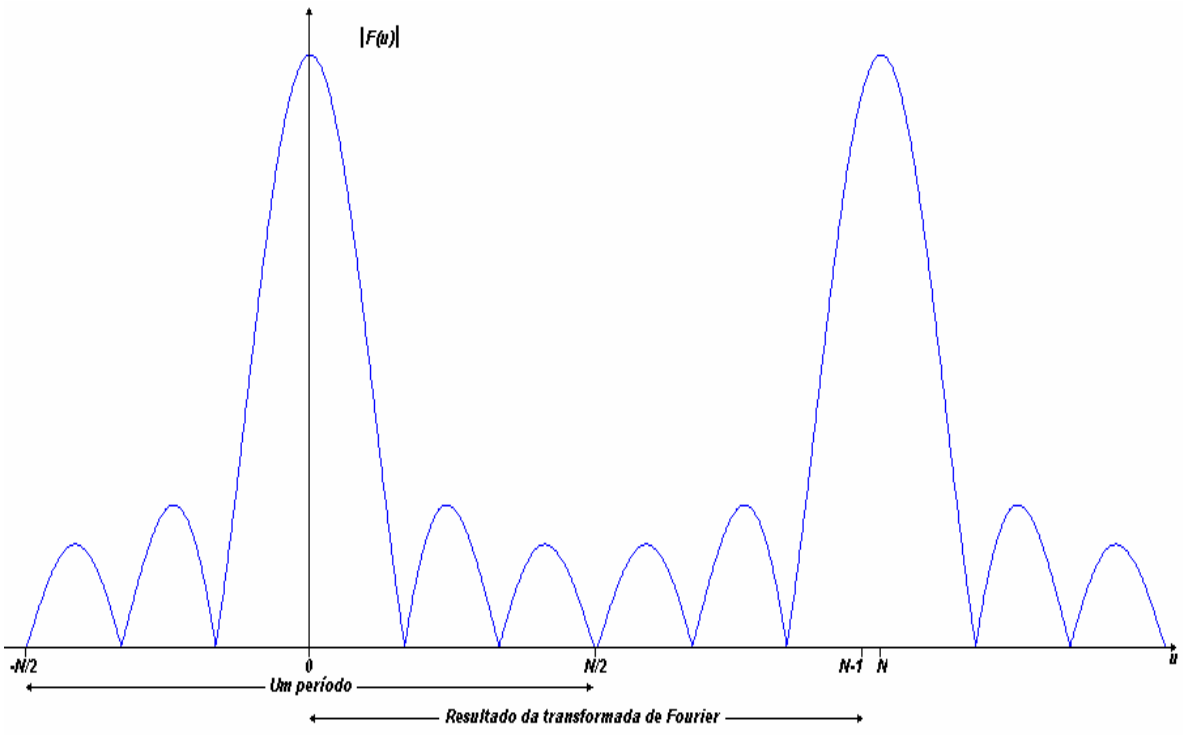


Figura 5: Espectro de Fourier como resultado do cálculo da transformada de Fourier

Para exibir um período inteiro, é suficiente mover a origem da transformada para o ponto $u=N/2$ como mostrado na Figura 6. Este deslocamento corresponde a multiplicarmos $f(x)$ por $(-1)^x$[6]

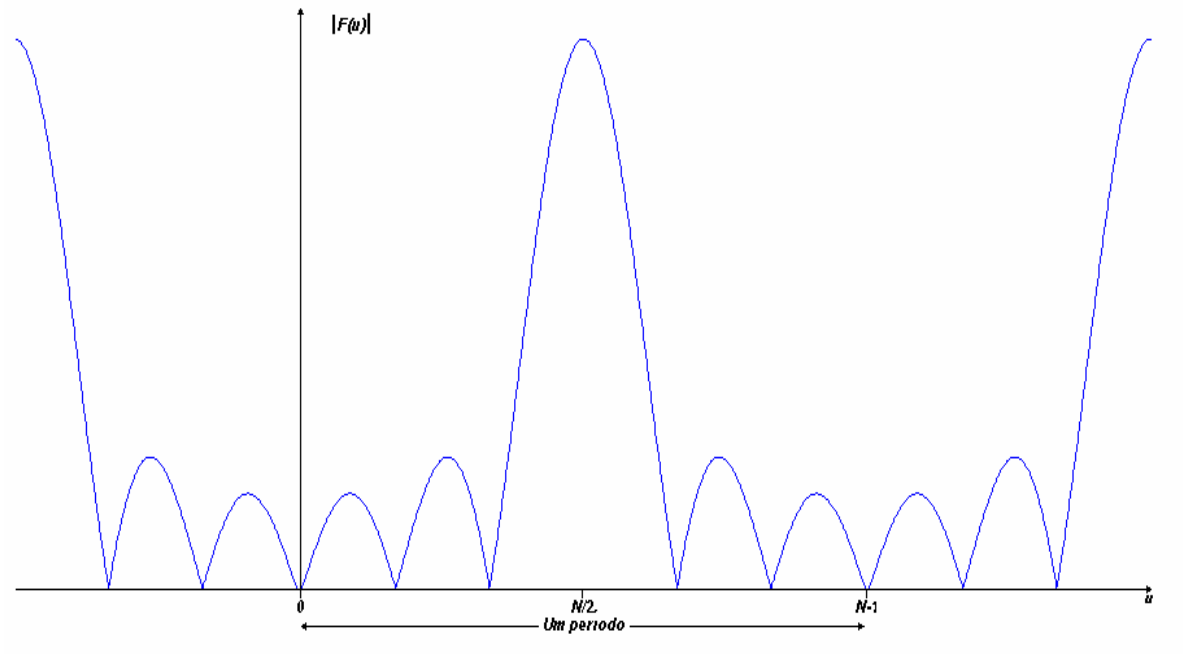


Figura 6: Espectro da transformada de Fourier deslocado

Este mesmo raciocínio pode ser aplicado para a magnitude da transformada de Fourier bidimensional, sendo a interpretação dos resultados mais difícil de interpretar se a origem da transformada não for deslocada para o ponto de freqüência (*N*/2, *N*/2) [6].

3.4.3.4 Rotação

Introduzindo coordenadas polares temos $x = r\cos\theta$, $y = r\,\mathrm{sen}\,\theta$, $u = \omega\cos\phi$ e $v = \omega\,\mathrm{sen}\,\phi$, assim *f(x,y)* e *F(u,v)* tornam-se *f(r,θ)* e *F(w, φ)* respectivamente. Considerando-se uma rotação de um ângulo θ_o teremos:

$$f(r,\theta+\theta_0) \Leftrightarrow F(\omega,\phi+\theta_0) \qquad \text{Equação 51}$$

Da Equação 51, tem-se que uma rotação de *f(x,y)* de um ângulo θ_0 resulta em uma rotação de *F(u,v)* deste mesmo ângulo [6].

3.4.3.5 Distributividade e Mudança de escala

Da definição do par de transformadas contínua ou discreta temos

$$\Im\{f_1(x,y)+f_2(x,y)\} = \Im\{f_1(x,y)\}+\Im\{f_2(x,y)\} \qquad \text{Equação 52}$$

e, em geral,

$$\Im\{f_1(x,y)\times f_2(x,y)\} = \Im\{f_1(x,y)\}\times\Im\{f_2(x,y)\} \qquad \text{Equação 53}$$

Assim, a transformada de Fourier e sua inversa são distributivas quanto à adição mas não quanto à multiplicação [6].

Para dois escalares a e b,

$$af(x,y) \Leftrightarrow aF(u,v) \qquad \text{Equação 54}$$

e

$$f(ax,by) \Leftrightarrow \frac{1}{|ab|}F(\frac{u}{a},\frac{v}{b}) \qquad \text{Equação 55}$$

3.4.3.6 Valor Médio

Uma definição muito utilizada do valor médio de uma função discreta bidimensional é definida pela Equação 56 [6].

$$\overline{f}(x,y) = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \qquad \text{Equação 56}$$

Substituindo-se *u*=*v*=0 na Equação 38 resulta,

$$F(0,0) = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \qquad \text{Equação 57}$$

Portanto $\overline{f}(x,y)$ é relacionada à transformada de Fourier de *f(x,y)* por

$$\overline{f}(x,y) = F(0,0) \qquad \text{Equação 58}$$

3.4.4 Transformada Rápida de Fourier (FFT)

O tempo de execução de um algoritmo em um computador é proporcional ao número de multiplicações e adições necessárias para sua execução. No caso da Equação 29, DFT, para cada um dos *N* valores de u, são requeridos *N* multiplicações complexas de *f(x)* por $e^{\frac{-j2\pi ux}{N}}$ e *N-1* adições dos resultados, podendo-se dizer que implementação da DFT é proporcional N^2. Os termos $e^{\frac{-j2\pi ux}{N}}$ podem se computados uma vez e armazenados numa tabela para todos os outros cálculos seguintes, por esta razão ele não é usualmente considerado para determinar o tempo de computação necessário para a implementação da DFT [6].

O algoritmo denominado *algoritmo da transformada rápida de Fourier* (FFT) executa um procedimento de decomposição da Equação 29 que torna o número de multiplicações e adições proporcional a $N\log_2 N$. A redução de N^2 para $N\log_2 N$ representa uma redução considerável no tempo de computação necessário para se obter transformada discreta de Fourier. A vantagem computacional é dada pela Equação 59.

$$VC = \frac{N}{\log_2 N} \qquad \text{Equação 59}$$

Assim, para um vetor com 8192 pontos, por exemplo, a vantagem computacional pelo uso da FFT em relação ao cálculo da DFT normal seria igual 630,15. Se o cálculo da transformada para este vetor usando a FFT utilizasse 5s, o cálculo da DFT levaria 5s x VC, aproximadamente 50 minutos, para ser executado na mesma máquina [6].

Como a relação entre o tempo despendido para o cálculo da DFT e o cálculo da FFT é logarítmica, a diferença no tempo de execução se tornará mais perceptível quando for efetuado com vetores maiores.

3.4.5 O Algoritmo da FFT

O algoritmo da FFT, descrito abaixo, faz uso da decimação do tempo. Por conveniência vamos rearranjar a Equação 29, conforme mostrado na Equação 60:

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) W_N^{ux} \qquad \text{Equação 60}$$

onde

$$W_N = e^{\frac{-j2\pi}{N}} \qquad \text{Equação 61}$$

Assumindo *N* como

$$N = 2^n \qquad \text{Equação 62}$$

com *n* um número inteiro e positivo. Portanto, *N* pode ser dado por:

$$N = 2M \qquad \text{Equação 63}$$

em que *M* também é um número inteiro e positivo [6], temos

$$F(u) = \frac{1}{2M} \sum_{x=0}^{2M-1} f(x) W_{2M}^{ux} \qquad \text{Equação 64}$$

A Equação 64 pode ser separada em dois termos:

$$F(u) = \frac{1}{2}\left(\frac{1}{M} \sum_{x=0}^{M-1} f(2x) W_{2M}^{u(2x)} + \sum_{x=0}^{M-1} f(2x+1) W_{2M}^{u(2x+1)} \right) \qquad \text{Equação 65}$$

Da Equação 61 tem-se que $W_{2M}^{2ux} = W_M^{ux}$. Assim podemos expressar a Equação 65 na forma:

$$F(u) = \frac{1}{2}\left(\frac{1}{M} \sum_{x=0}^{M-1} f(2x) W_M^{ux} + \sum_{x=0}^{M-1} f(2x+1) W_M^{ux} W_{2M}^{u} \right) \qquad \text{Equação 66}$$

Definindo-se:

$$F_{par}(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(2x) W_M^{ux} \qquad \text{Equação 67}$$

para *u= 0,1,2,...,M-1*, e

$$F_{impar}(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(2x+1) W_M^{ux} \qquad \text{Equação 68}$$

para *u=0,1,2...,M-1* a Equação 66 torna-se

$$F(u) = \frac{1}{2}\left(F_{par}(u) + F_{impar}(u) W_{2M}^{u} \right) \qquad \text{Equação 69}$$

Da Equação 61 tira-se:

$$W_M^{u+M} = W_M^{u} \qquad \text{Equação 70}$$

e que:

$$W_{2M}^{u+M} = -W_M^{u} \qquad \text{Equação 71}$$

Aplicando-se a Equação 70 e a Equação 71 na Equação 67 e na Equação 69 temos:

$$F(u+M) = \frac{1}{2}\left(F_{par}(u) - F_{impar}(u) W_{2M}^{u} \right) \qquad \text{Equação 72}$$

Analisado a Equação 69 e a Equação 72 nota-se que podemos computar a transformada discreta de Fourier em duas partes:

1. A primeira parte da *F(u)* para *u=0,1,2..,M-1* usando a Equação69.
2. A segunda parte da *F(u)* para *u=M,M+1, M+2,...2M-1* usando os mesmos valores de $F_{par}(u)$ e $F_{impar}(u)\, w_{2M}^{u}$ calculados anteriormente e agora substituídos na Equação 72 [6].

A Equação 67 e a Equação 68 de *Fpar* e *Fimpar* respectivamente, correspondem a uma decimação de *F(u)* em duas seqüências de N/2 pontos. Como *F(u)* é periódica em *u* com um período N, *Fpar* e *Fimpar* em um período N/2 [14].

Reescrevendo a Equação 67 e a Equação 68 temos:

$$F_2(u) = f_{par}(u) = \frac{1}{\left(\frac{N}{2}\right)} \sum_{x=0}^{\left(\frac{N}{2}\right)-1} f(2x) W_{\frac{N}{2}}^{ux} \qquad \text{Equação 73}$$

$$F_3(u) = F_{impar}(u) = \frac{1}{\left(\frac{N}{2}\right)} \sum_{x=0}^{\left(\frac{N}{2}\right)-1} f(2x+1) W_{\frac{N}{2}}^{ux} \qquad \text{Equação 74}$$

$$F2(u) = \frac{1}{\left(\frac{N}{4}\right)} \sum_{x=0}^{\left(\frac{N}{4}\right)-1} f(4x) W_{\frac{N}{2}}^{2ux} + \frac{1}{\left(\frac{N}{4}\right)} \sum_{x=0}^{\left(\frac{N}{4}\right)-1} f(4x+2) W_{\frac{N}{4}}^{u(2x+1)} \quad \text{Equação 75}$$

Ou seja,

$$F2(u) = \frac{1}{\left(\frac{N}{4}\right)} \sum_{x=0}^{\left(\frac{N}{4}\right)-1} f(4x) W_{\frac{N}{2}}^{ux} + W_{N/2}^{u} \frac{1}{\left(\frac{N}{4}\right)} \sum_{x=0}^{\left(\frac{N}{4}\right)-1} f(4x+2) W_{\frac{N}{4}}^{ux} \quad \text{Equação 76}$$

Que fica:

$$F2(u) = F4(u) + W_{\frac{N}{2}}^{u} F5(u) \quad \text{Equação 77}$$

Este mesmo raciocínio vale para *F3(u)*:

$$F3(u) = \frac{1}{\left(\frac{N}{4}\right)} \sum_{x=0}^{\left(\frac{N}{4}\right)-1} f(4x+1) W_{\frac{N}{4}}^{ux} + W_{\frac{N}{2}}^{u} \frac{1}{\left(\frac{N}{4}\right)} \sum_{x=0}^{\left(\frac{N}{4}\right)-1} f(4x+3) W_{\frac{N}{4}}^{ux} \quad \text{Equação 78}$$

Que fica:

$$F3(u) = F6(u) + W_{\frac{N}{2}}^{u} F7(u) \quad \text{Equação 79}$$

Sendo *F4(u)* até *F7(u)* periódicas com um período N/4.

O processo de decimação será continuado até que termine num termo único. No penúltimo passo (2^{N-2}), quatro subseqüências são obtidas tendo a forma:

$$F_x(u) = F_y(u) + W_2^u F_z(u) \quad \text{Equação 80}$$

Estas subseqüências podem ser decimadas para o último passo (2^{N-1}) com duas subseqüências, sendo o primeiro par mostrado na Equação 81:

$$\begin{aligned} F_x(u) &= f(0) + W_2^u f(N/2) \\ F_y(u) &= f(1) + W_2^u f((N/2)+1) \end{aligned} \quad \text{Equação 81}$$

No último passo, o processo termina em termos como f(0), f(N/2), f(1), e f(N/2-1). Assim, a transformada discreta de Fourier de 2^N pontos usando-se:

$$f(0), f(N/2), f(N/4), f(3N/4), ..., f((N/2)-1), f(N-1) \quad \text{Equação 82}$$

Isto corresponde ao rearranjo da entrada em uma seqüência diferente conhecida como reversão de bits [14].

3.4.6 A FFT Inversa

O algoritmo usado para o cálculo da transformada direta também pode ser usado para o cálculo da transformada inversa com pequenas alterações na entrada. Para mostrar isto, vamos tomar a Equação 29 e a Equação 30, repetidas abaixo:

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\, e^{\frac{-j2\pi ux}{N}} \qquad \text{Equação 83}$$

$$f(x) = \sum_{u=0}^{N-1} F(u)\, e^{\frac{j2\pi ux}{N}} \qquad \text{Equação 84}$$

Usando o conjugado da Equação e dividindo ambos os lados por N, temos:

$$\frac{1}{N} f^{*}(x) = \frac{1}{N}\sum_{u=0}^{N-1} F^{*}(u)\, e^{\frac{-j2\pi ux}{N}} \qquad \text{Equação 85}$$

Comparado a Equação 85 com a Equação 83, temos que o lado direito da igualdade da Equação 85 está na forma de uma transformada de Fourier. Assim se usarmos a $F^{*}(u)$ como entrada de um algoritmo para computar a transformada direta o resultado do cálculo será igual a $f^{*}(x)/N$; tomando-se o conjugado do valor calculado e multiplicando-se por N obteremos a transformada inversa $f(x)$ [6].

3.4.7 Filtros No Domínio da Freqüência

Os filtros no domínio da freqüência baseiam-se no teorema da convolução. Assim, se uma imagem $g(x,y)$ formada pela convolução de uma imagem $f(x,y)$ e um operador linear invariante $h(x,y)$:

$$g(x,y) = h(x.y) * f(x,y) \qquad \text{Equação 86}$$

Do teorema da convolução (Seção 3.4.8), tem-se a seguinte relação no domínio da freqüência:

$$G(u,v) = H(u,v)F(u.v) \qquad \text{Equação 87}$$

onde G, H, e F são transformadas de Fourier de g, h e f. Usando a terminologia da teoria de sistemas lineares $H(u,v)$ é denominada função de transferência do processo [6].

Os filtros no domínio da freqüência, que são usados para realce da imagem, são expressos na forma da Equação 87. Assim numa aplicação típica de realce de imagem se tem a imagem original *f*(x,y), se obtém a *F*(u,v), e se escolhe adequadamente uma função *H*(u,v) de forma a para se obter a *g*(x,y) que possui algumas características de *f*(x,y) realçadas. Sendo *g(x,y)* dada por:

$$g(x,y) = \Im^{-1}\left|H(u,v)F(u,v)\right| \qquad \text{Equação 88}$$

3.4.8 Convolução

A convolução [6] de duas função *f(x)* e *g(x)* é definida na Equação 89,

$$f(x) * g(x) = \int_{-\infty}^{+\infty} f(\alpha)g(x-\alpha)dx \qquad \text{Equação 89}$$

onde α é uma variável de integração.

Um resultado importante da convolução é conhecido como Teorema da Convolução. Se *f(x)* tem com transformada de Fourier *F*(u) e g(x) tem como transformada de Fourier *G*(u), a convolução, realizada no domínio do espaço, destas funções *f*(x)**g*(x) tem como transformada de Fourier, no domínio freqüência, a função *F*(u).*G*(u). Um resultado análogo é obtido no domínio da freqüência onde *F*(u)**G*(u) tem como transformada inversa *f*(x). *g*(x). Estes resultados são enunciados como:

$$f(x) * g(x) \Leftrightarrow F(u) \times G(u) \qquad \text{Equação 90}$$

$$f(x) \times g(x) \Leftrightarrow F(u) * G(u) \qquad \text{Equação 91}$$

Estes resultados permitem efetuar a convolução de duas funções através da mudança de espaço, da operação de multiplicação e do retorno para espaço original. Em termos computacionais pode-se substituir a operação de integração pela multiplicação e pela mudança de espaço, podendo reduzir o tempo de computação requerido [6].

Para aplicações em imagens, interessa a convolução em duas dimensões (2-D) que é dada pela Equação 92:

$$f(x,y) * g(xy) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(\alpha)g(x-\alpha)d\alpha d\beta \qquad \text{Equação 92}$$

O teorema da convolução para 2-D fica:

$$f(x,y) * g(x,y) \Leftrightarrow F(u,v) \times G(u,v) \quad \text{Equação 93}$$

$$f(x,y) \times g(x,y) \Leftrightarrow F(u,v) * G(u,v) \quad \text{Equação 94}$$

O conjunto dos pixels de uma imagem constitui uma matriz discreta. Para se formular a convolução discreta 2-D representa-se f(x,y) e g(x,y) como matrizes discretas de duas demissões A x B e C x D respectivamente. Estas funções devem ser periódicas com algum período M (a direção x) e N (na direção y). Para se evitar erros de revestimento nos períodos individuais de convolução se escolhe M e N tal que:

$$M \geq A + C - 1 \quad \text{Equação 95}$$

$$N \geq B + D - 1 \quad \text{Equação 96}$$

Para formar as seqüências periódicas se estende f(x,y) e g(x,y):

$$f_e(x,y) = \begin{cases} f(x,y) & 0 \leq x \leq A-1 \quad e \quad 0 \leq y \leq B-1 \\ 0 & A \leq x \leq M-1 \quad ou \quad B \leq y \leq N-1 \end{cases} \quad \text{Equação 97}$$

$$g_e(x,y) = \begin{cases} g(x,y) & 0 \leq x \leq C-1 \quad e \quad 0 \leq y \leq D-1 \\ 0 & C \leq x \leq M-1 \quad ou \quad D \leq y \leq N-1 \end{cases} \quad \text{Equação 98}$$

A convolução discreta de $f_e(x,y)$ e $g_e(x,y)$ é definida pela relação expressa na Equação 99:

$$f_e(x,y) * g_e(x,y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f_e(m,n)\, g_e(x-m, y-n) \quad \text{Equação 99}$$

3.4.9 Filtros de Butterworth

Os filtros de Butterworth possuem uma função de transferência sem descontinuidades abruptas que estabeleça um corte claro entre as freqüências passadas e as freqüências filtradas. Esta transição suave diminui o efeito do janelamento da imagem provocado pelos filtros ideais (ver Seção 3.4.10)[6].

3.4.9.1 Filtro de Butterworth Passa-baixa

A função de transferência do filtro de Butterworth passa-baixa é dada pela Equação 100.

$$H(u,v) = \frac{1}{1+\left[\frac{\sqrt{u^2+v^2}}{f_c}\right]^{2n}} \qquad \text{Equação 100}$$

Onde f_c é a freqüência de corte na qual a atenuação do nível de cinza é de 50% que ocorre quando $f_c = \sqrt{u^2+v^2}$ e n a ordem do filtro. Normalmente se estabelece a freqüência de corte onde a atenuação é de -3dB ($1/\sqrt{2}$). Neste caso a função de transferência *H(u,v)* torna-se [6]:

$$H(u,v) = \frac{1}{1+\left[\sqrt{2}-1\right]\left[\frac{\sqrt{u^2+v^2}}{f_c}\right]^{2n}} = \frac{1}{1+0.414\left[\frac{\sqrt{u^2+v^2}}{f_c}\right]^{2n}} \qquad \text{Equação 101}$$

3.4.9.2 Filtro de Butterworth Passa-alta

A função de transferência do filtro Butterworth passa-alta é dada pela Equação 102.

$$H(u,v) = \frac{1}{1+\left[\frac{f_c}{\sqrt{u^2+v^2}}\right]^{2n}} \qquad \text{Equação 102}$$

Onde f_c é a freqüência de corte na qual a atenuação do nível de cinza é de 50% que ocorre quando $f_c = \sqrt{u^2+v^2}$ e n é a ordem do filtro. Normalmente se estabelece a freqüência de corte onde a atenuação é de -3dB ($1/\sqrt{2}$). Neste caso a função de transferência *H(u,v)* torna-se [6]:

$$H(u,v) = \frac{1}{1+\left[\sqrt{2}-1\right]\left[\frac{f_c}{\sqrt{u^2+v^2}}\right]^{2n}} = \frac{1}{1+0.414\left[\frac{f_c}{\sqrt{u^2+v^2}}\right]^{2n}} \qquad \text{Equação 103}$$

3.4.9.3 Filtro de Butterworth Passa-banda

A função de transferência do filtro de Butterworth passa-banda é dada pela Equação 104.

$$H(u,v) = \frac{1}{1+\left[\frac{(\sqrt{u^2+v^2}) + f_{cs} f_{ci}}{(u^2+v^2)(f_{cs} - f_{ci})}\right]^{2n}} \qquad \text{Equação 104}$$

Onde f_{cs} e f_{ci} são as freqüências de corte inferior e superior, respectivamente, nas quais a atenuação do nível de cinza é de 50% que ocorre quando $f_{cs} = \sqrt{u^2+v^2}$ ou $f_{ci=}\sqrt{u^2+v^2}$ e n é a ordem do filtro. Normalmente se estabelece a freqüência de corte onde a atenuação é de -3dB ($1/\sqrt{2}$). Neste caso a função de transferência *H(u,v)* torna-se:

$$H(u,v) = \frac{1}{1+\left[\sqrt{2}-1\right]\left[\frac{(\sqrt{u^2+v^2}) + f_{cs} f_{ci}}{(u^2+v^2)(f_{cs} - f_{ci})}\right]^{2n}} \qquad \text{Equação 105}$$

$$H(u,v) = \frac{1}{1+0.414\left[\frac{(\sqrt{u^2+v^2}) + f_{cs} f_{ci}}{(u^2+v^2)(f_{cs} - f_{ci})}\right]^{2n}} \qquad \text{Equação 106}$$

3.4.9.4 Filtro de Butterworth Rejeita-banda

A função de transferência do filtro de Butterworth passa-banda é dada pela Equação 107.

$$H(u,v) = \frac{1}{1+\left[\frac{(u^2+v^2)(f_{cs} - f_{ci})}{(\sqrt{u^2+v^2}) + f_{cs} f_{ci}}\right]^{2n}} \qquad \text{Equação 107}$$

Onde f_{cs} e f_{ci} são as freqüências de corte inferior e superior, respectivamente, nas quais a atenuação do nível de cinza é de 50% que ocorre quando $f_{cs} = \sqrt{u^2+v^2}$ ou $f_{ci=}\sqrt{u^2+v^2}$ e n é a ordem do filtro. Normalmente se estabelece a freqüência de corte

onde a atenuação é de -3dB ($1/\sqrt{2}$). Neste caso a função de transferência *H(u,v)* torna-se:

$$H(u,v) = \frac{1}{1 + \left[\sqrt{2} - 1\right]\left[\frac{(u^2 + v^2)(f_{cs} - f_{ci})}{(\sqrt{u^2 + v^2}) + f_{cs} f_{ci}}\right]^{2n}} \qquad \text{Equação 108}$$

$$H(u,v) = \frac{1}{1 + 0.414\left[\frac{(u^2 + v^2)(f_{cs} - f_{ci})}{(\sqrt{u^2 + v^2}) + f_{cs} f_{ci}}\right]^{2n}} \qquad \text{Equação 109}$$

3.4.10 Janelamento

Uma imagem digitalizada pode ser considerada como sendo o produto, no domínio do espaço, da imagem propriamente dita (com infinitos elementos) por uma janela também com infinitos elementos em que tem *NxM* elementos cada um com valor unitário e os demais elementos são iguais a zero [18]

A multiplicação da imagem janela no domínio do espaço, resulta na convolução da transformada da imagem pela transformada da janela. Esta convolução causa um efeito indesejado conhecido como efeito de janelamento que provoca um espalhamento do espectro da imagem [14].

Para entender melhor o espalhamento do espectro vamos usar um sinal unidimensional *y(x)* com duração infinita. Utilizando uma porção relativamente curta do sinal, teremos,

$$f(x) = \begin{cases} y(x) & 0 \le x \le N-1 \\ 0 & 0 < x > N-1 \end{cases} \qquad \text{Equação 110}$$

Podemos descreve a operação de obtenção de f(x) de y(x) como sendo a multiplicação de *y(x)* por uma janela retangular $j_r(x)$, conforme mostrado em Equação 111 [18].

$$f(x) = y(x) \times j_r(x) \quad onde \quad j_r(x) = \begin{cases} 1 & 0 \le x \le N-1 \\ 0 & 0 < x > N-1 \end{cases} \qquad \text{Equação 111}$$

Segundo o teorema da convolução, a transformada de Fourier de *f(x)* é:

$$\Im(f(x) = \Im(y(x) \times j_r(x)) = \Im(y(x)) * \Im(j_r(x)) \qquad \text{Equação 112}$$

A análise da $\Im(j_r(x))$ nos leva as seguintes conclusões:

1. A transformada de Fourier da janela retangular é dada por,

$$J_r(u) = \frac{sen(\pi u N)}{sen(\pi u)} \qquad \text{Equação 113}$$

Sendo sua representação gráfica é mostrada na Figura 7.

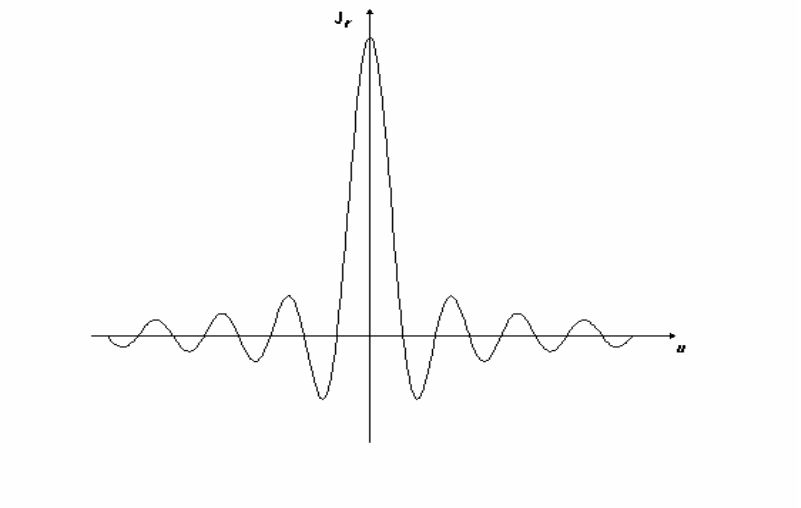


Figura 7: Transformada de Fourier de uma janela retangular

2. O espectro de potência é mostrada na Figura 8.

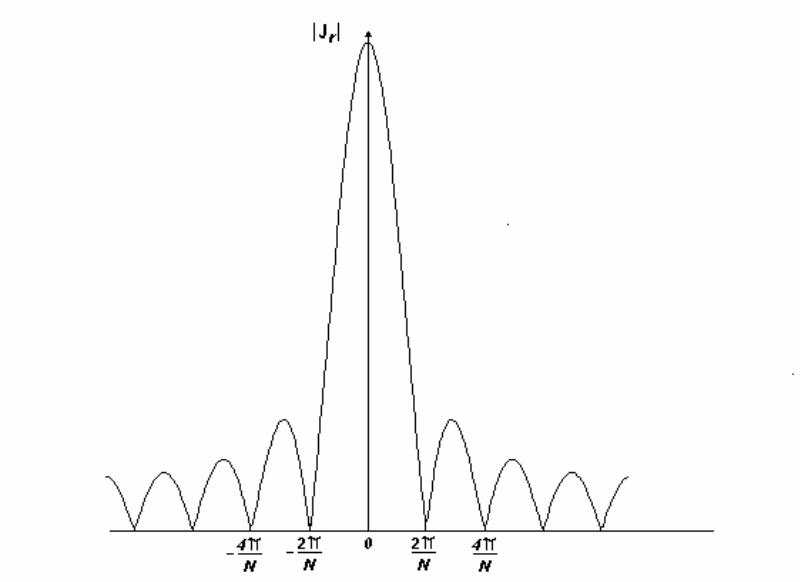


Figura 8: Espectro de potência da janela retangular

Onde, a largura do lobo principal é de 4π/N e a atenuação do lobo principal para o primeiro lobo lateral é de aproximadamente 13.5 dB [18], [14].

Para reduzir o espalhamento de espectro provocado pela convolução do sinal com a janela usam-se janelas que variem suavemente nas bordas.

3.4.10.1 Janela Blackman

A janela blackman é dada pela função exposta pela Equação 114 e a sua resposta em amplitude é mostrada na Figura 9 [9].

$$J_B = 0.42 - 0.5\cos\left(\frac{2\pi x}{N-1}\right) + 0.08\cos\left(\frac{4\pi x}{N-1}\right) \qquad \text{Equação 114}$$

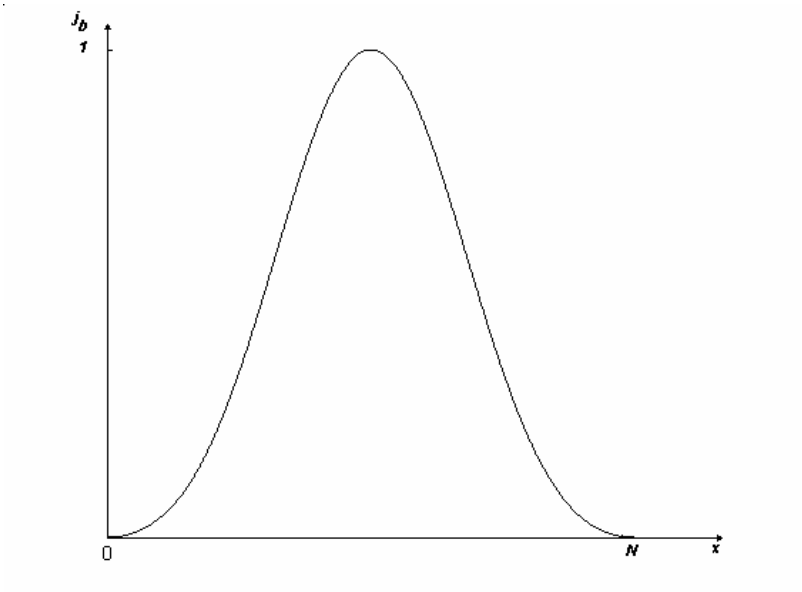


Figura 9: Resposta em amplitude da janela blackman

O espectro de potência da janela blackman, mostrado na Figura 10, apresenta uma atenuação do lobo principal para o primeiro lobo lateral de aproximadamente 58 dB [14].

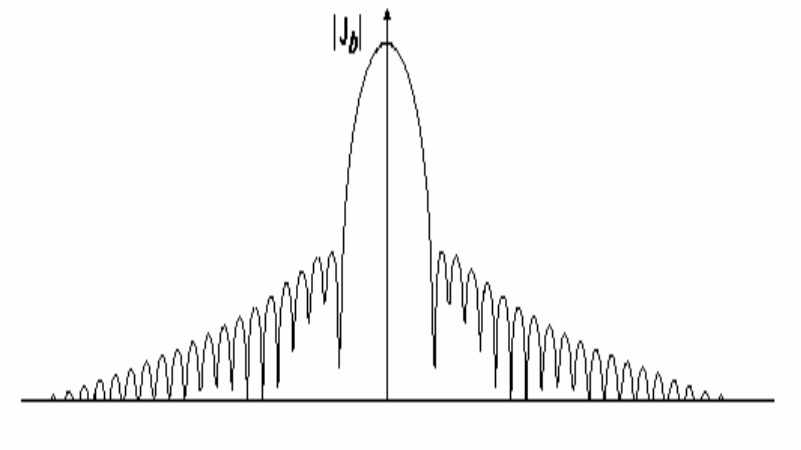


Figura 10: Espectro de potência da janela blackman

3.4.10.2 Janela cos^4

A janela cos^4 é dada pela função exposta na Equação 115 e a resposta em amplitude esta mostrada na Figura 11 [14].

$$J_c = \cos^4\left(\frac{\pi\left(x-\frac{N}{2}\right)}{N}\right) \qquad \text{Equação 115}$$

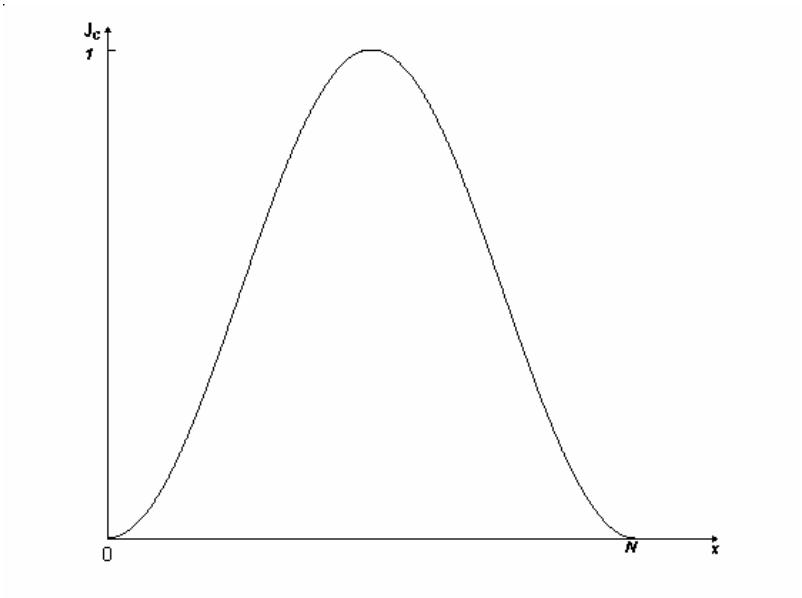


Figura 11: resposta em amplitude da janela cos^4

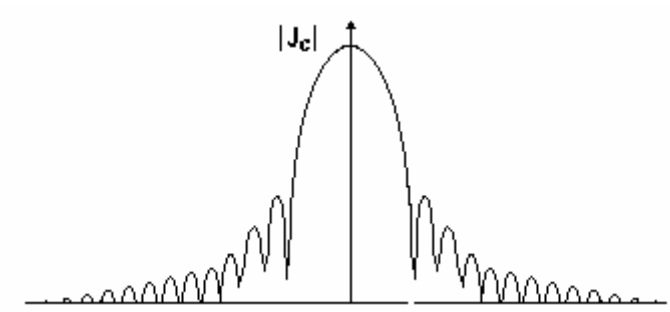


Figura 12: Espectro de potência da janela cos^4

O espectro de potência da janela cos^4, mostrado na Figura 12 apresenta uma atenuação do lobo principal para o primeiro lobo lateral de aproximadamente 47 dB [14].

3.5 Técnicas de Realce da Imagem no Domínio do Espaço

A expressão *domínio do espaço* refere-se ao conjunto de pixels que constituem uma imagem. As operações feitas neste domínio atuam diretamente sobre os pixels. Funções de processamento imagem no domínio espacial podem ser expressas como:

$$g(x, y) = T[f(x, y)] \qquad \text{Equação 116}$$

onde:

f(x,y): imagem original.

g(x,y): imagem processada.

T: operador sobre *f* definido sobre alguma vizinhança de (x,y)

3.5.1 Filtros Espaciais ou Filtros de Convolução no Domínio do Espaço

Os filtros espaciais operam transformações na imagem pixel a pixel. Estas transformações podem depender do nível de cinza do pixel que está sendo transformado, bem como dos valores de nível de cinza dos pixels vizinhos a ele. Eles podem ser ainda filtros lineares ou não lineares.

Nos filtros espaciais de convolução (lineares), para se efetuar a filtragem, faz-se a convolução de uma máscara sobre a imagem que se deseja filtrar. Esta máscara é uma matriz de pesos pelos quais são multiplicados o nível de cinza do próprio pixel e de seus vizinhos. A soma destas multiplicações resulta no nível de cinza do pixel da imagem filtrada. Para se garantir que as características estatísticas da imagem sejam mantidas (por exemplo: a média dos níveis de cinza) é feita a normalização da máscara. A normalização consiste em dividir cada peso da máscara pelo número total de pesos. Assim o fator de normalização k é o inverso do número de pesos da matriz da máscara.

Os filtros espaciais de convolução (não lineares) operam na vizinhança de um determinado pixel. Sua operação baseia-se nos valores de níveis de cinza da vizinhança considera.

3.5.1.1 Filtro Passa-Baixa ou Filtro da Média

O filtro da média é normalmente usado para eliminar ruídos de alta freqüência e para redução de níveis de cinza onde existam transições abrutas de níveis de cinza. Ele é um filtro linear. Sua máscara 3x3 é mostrada na Figura 13.

$$\frac{1}{9}\times\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Figura 13: Máscara de um filtro passa-baixa 3x3

Ao ser aplicado este de filtro nota-se borramento da imagem especialmente nas bordas, ressaltando as baixas freqüências. Este tipo de filtro é usado quando se deseja suavizar a imagem[6].

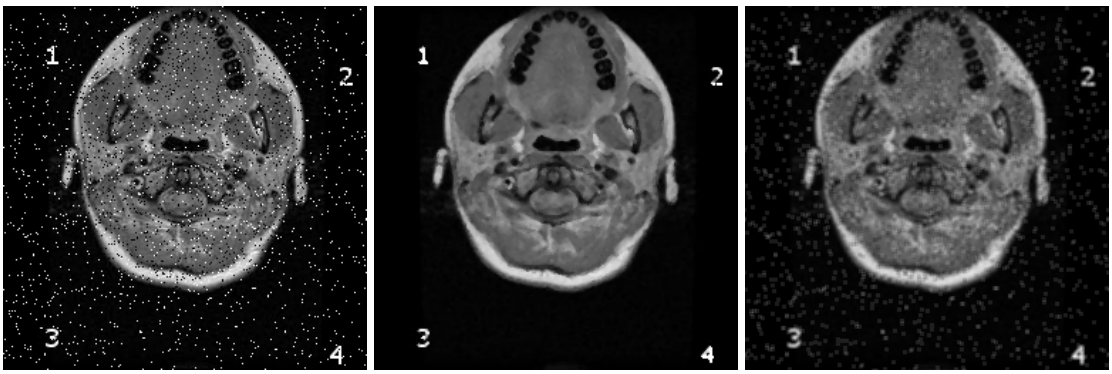
3.5.1.2 Filtro da Mediana

O filtro da mediana é um filtro não linear que é usado para reduzir o ruído impulsivo de alta freqüência nas imagens (*spike*). Neste tipo filtro considera-se os níveis de cinza de uma determinada área da imagem e obtém-se o valor médio desta área. Considerando-se a que a área da imagem para cálculo da mediana é uma matriz de 3x3, cujo pixel central na imagem origem é o que se deseja calcular o correspondente na imagem destino, teremos:

$$\begin{bmatrix}Po_{11} & Po_{12} & Po_{13}\\ Po_{21} & Po_{22} & Po_{23}\\ Po_{31} & Po_{32} & Po_{33}\end{bmatrix}$$

Figura 14: Matriz 3x3 da área de uma imagem

Onde Po_{ij} ($i=1,2,3$ e $j=1,2,3$) representa o valor do nível de cinza de cada ponto sobreposto pela máscara na imagem original. Para se determinar o nível de cinza Pd_{22} na imagem destino, correspondente à posição do pixel Po_{22} na imagem origem, organiza-se os nove valores de nível de cinza, abrangidos pela máscara, em um vetor de nove elementos Vn ($n=1,2,..9$), organizados em ordem crescente de seus valores de nível cinza. O quinto valor (valor mediano) corresponderá ao valor do nível de cinza de Pd_{22}.



a **b** **c**

Figura 15: Comparação entre o filtro da media e o filtro da mediana

Na Figura 15.a temos uma imagem com ruído impulsivo (*spike*). Na Figura 15.b temos o resultado da aplicação do filtro da mediana e na Figura 15.c temos o resultado do filtro da média móvel. Como se pode observar, para este tipo de ruído, o filtro da mediana apresenta resultados melhores.

3.5.1.3 Filtros de Aguçamento (Sharpening)

Os filtros de aguçamento são empregados para ressaltar detalhes que tenham sido borrados devido a erros ou a particularidades do método usado na aquisição da imagem.

3.5.1.3.1 Filtro Passa-Alta básico

O filtro passa-alta básico é um filtro linear. Usado para acentuar detalhes finos da imagem (alta freqüência). A resposta ao impulso deste tipo de filtro indica que o filtro deve ter coeficientes positivos próximos ao centro e negativos na periferia. Para uma máscara 3x3, escolhe-se um valor positivo no centro, com coeficientes negativos nos outros elementos da matriz. A Figura 16 mostra uma máscara de filtro espacial que atende a estas condições. Nele a soma dos seus coeficientes é zero. Para áreas com nível de cinza constante ou com pequenas variações sua saída é zero ou muito próxima disto. Ele também reduz o valor médio dos níveis de cinza pela eliminação do termo de freqüência zero [6].

$$\frac{1}{1}\times\begin{bmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{bmatrix}$$

Figura 16: Máscara de um filtro passa-alta básico

3.5.1.3.2 Filtros de Alto Reforço

Quando se deseja detalhes finos da imagem (alta freqüência), sem a perda dos componentes de baixa freqüência, usam-se filtros de alto reforço. Para este tipo de filtro usa-se a máscara mostrada na Figura 17.

$$\frac{1}{9}\times\begin{bmatrix}-1 & -1 & -1\\ -1 & w & -1\\ -1 & -1 & -1\end{bmatrix}$$

Figura 17: Máscara para o filtro de alto reforço

Onde:

$w=9A-1$ com $A\geq 1$

A: fator de amplificação dos níveis de cinza

Para $A=1$ temos o filtro passa-alta básico. Com $A>1$ temos a filtragem de alto reforço. Os melhores resultados para este tipo de filtro são obtidos com $A=1.1$. Com $A>1.2$ a ampliação dos ruídos se torna mais acentuada[6].

3.5.1.4 Filtro por derivadas

O filtro da média tende a borrar os detalhes da imagem. O efeito da convolução do filtro da média sobre a imagem é análogo à integração da imagem. Portanto, a diferenciação causa o efeito inverso, ressaltando as altas freqüências. Um dos métodos usados para diferenciação da imagem é o gradiente[6].

Considerando a função $f(x,y)$ o gradiente de f para as coordenas (x,y) é definido por:

$$\nabla f = \begin{bmatrix}\frac{\partial f}{\partial x}\\ \frac{\partial f}{\partial y}\end{bmatrix} \qquad \text{Equação 117}$$

Sendo a magnitude desse vetor dada por:

$$\nabla f = mag(\nabla f) = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2} \quad \text{Equação 118}$$

Para uma determinada área da imagem temos na região 3x3 pixels, onde *z* representa os níveis de cinza da imagem (Figura 18):

$$\begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix}$$

Figura 18: Representação de uma área com 3x3 pixels da imagem

Uma das maneiras de obter a magnitude do gradiente consiste em usar a diferença (z5-z8) para o eixo dos x e (z5-z6) para o eixo dos y, conforme expresso Equação 119.

$$\nabla f \approx \sqrt{(z_5 - z_8)^2 + (z_5 - z_6)^2} \quad \text{Equação 119}$$

Uma aproximação da Equação 119 pode ser feita com o uso de valores absolutos como mostrada na Equação 120:

$$\nabla f \approx |z_5 - z_8| + |z_5 - z_6| \quad \text{Equação 120}$$

Outra aproximação da Equação 119 pode ser obtida usando-se as diferenças cruzadas:

$$\nabla f \approx \sqrt{(z_5 - z_9)^2 + (z_6 - z_8)^2} \quad \text{Equação 121}$$

Ou através da diferença cruzada usando-se valores absolutos:

$$\nabla f \approx |z_5 - z_8| + |z_5 - z_6| \quad \text{Equação 122}$$

A Equação 119, Equação 120, Equação 121 e a Equação 122 podem ser implementadas através da soma de duas máscaras 2x2. As máscaras 2x2 usadas para implementar a Equação 120 são chamadas operadores cruzados de Roberts. Eles estão mostrados na Figura 19.

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Figura 19: Operadores cruzados de Roberts

Por facilidade de implementação são usadas matrizes 3x3. Uma aproximação para a Equação 118 no ponto z5, usando uma vizinhança de 3x3 é apresentada na Equação 123:

$$\nabla f \approx \left|(z7+z8+z9)-(z1+z2+z3)\right|+\left|(z3+z6+z9)-(z1+z4+z7)\right| \qquad \text{Equação 123}$$

As máscaras usadas para implementar a Equação 123, são chamados Operadores de Prewitt, eles são mostradas na Figura 20.

$$\begin{bmatrix}-1 & -1 & -1\\ 0 & 0 & 0\\ 1 & 1 & 1\end{bmatrix}\begin{bmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ 1 & 0-1 & 1\end{bmatrix}$$

Figura 20: Operadores de Prewitt

3.5.2 Processamento de Imagens em pseudo-cores

As técnicas que usam pseudo-cores em imagens monocromáticas fazem a atribuição de cores baseado nas várias propriedades dos seus conteúdos de níveis de cinza [6].

A técnica de fatiamento por intensidade atribui duas cores a imagem monocromática baseada no nível de cinza. Assim, algumas imagens com *K* possíveis níveis de cinza e que se queira atribuir n cores com $n \leq K$, a cor atribuída a um determinado pixel com um nível de cinza p será dada por:

$$C_m \quad se \quad \left\lfloor m \times \frac{K}{n} \right\rfloor \leq p < \left\lfloor (m+1) \times \frac{K}{n} \right\rfloor \quad m = 0,1,...n-1 \qquad \text{Equação 124}$$

3.5.3 Remapeamento Linear Global e Local

Remapeamento linear é uma técnica usada normalmente para realçar imagens que contêm uma pequena faixa de níveis de cinza. O remapeamento linear global consiste dos seguintes passos:

1. Encontram-se os pixels com o menor valor e o maior valor nível de cinza da imagem
2. Os valores são normalizados no intervalo [0,1].
3. Multiplica-se cada pixel normalizado pela maior nível de cinza desejado obtendo-se a imagem remapeada [10].

Bons resultados também são obtidos fazendo-se remapeamento linear local. O processo está descrito abaixo:

1. Encontram-se os pixels com o menor valor e o maior valor de nível de cinza da imagem de uma área escolhida da imagem.

2. Os valores que ficarem abaixo e acima do menor valor escolhido são igualados ao menor e ao maior valor, respectivamente, da área escolhida.
3. Os valores são normalizados no intervalo [0,1].
4. Multiplica-se cada pixel normalizado pela maior nível de cinza desejado obtendo-se a imagem remapeada [10].

3.5.4 Transformação para obtenção do negativo de uma imagem

O negativo de uma imagem é obtido através da transformação *s=Tn(r)* dada pela Equação 125:

$$s = Tn(r) = (N-1) - r \qquad \text{Equação 125}$$

Onde:

N: número de níveis de cinza.

r: nível de cinza da imagem original

s: nível de cinza resultante

Tn(r): transformação par obtenção do negativo da imagem.

O objetivo da operação é inverter a ordem do preto para o branco, de tal forma que a intensidade da imagem resultante da operação de negação diminua à medida que a intensidade da imagem que sofre a operação aumente[6].

3.5.5 Equalização de Histograma

Considerando uma imagem em níveis de cinza a função discreta de densidade de probabilidade dos níveis de cinza é:

$$p_r(r_k) = \frac{NC_k}{N} \quad 0 \le r_k \le 1 \quad e \quad k = 0,1...,NC-1 \qquad \text{Equação 126}$$

Onde:

NC: número de níveis de cinza.

r_k: k-éssimo nível cinza, sendo $r_k=0$ para o preto e $r_k=1$ para o branco

$p_r(r_k)$: probabilidade do *k*-ésimo nível de cinza.

NC_k:é o número de vezes que o nível de cinza k aparece na imagem.

N: é o número total de pixels da imagem

O gráfico de $p_r(r_k)$ *x* r_k é denominado histograma, e a transformação para obter um histograma uniforme é conhecida como equalização de histograma ou linearização de histograma [6].

A forma discreta da transformação usada para obter a equalização de histograma é dada por:

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{NC_j}{N} = \sum_{j}^{k} p_r(r_j) \quad 0 \le r_k \le 1 \quad e \quad k = 0,1,...N-1 \qquad \text{Equação 127}$$

3.5.6 Controle de Brilho e Saturação

Para o controle de brilho e saturação usa-se o modelo HSV (*Hue, Saturation, and Value*). As cores no modelo HSV estão normalizadas em relação ao vermelho, verde e azul do modelo RGB (*Red, Green, and Blue*) [6]. Assim temos:

$$r = \frac{R}{R+G+B} \qquad \text{Equação 128}$$

$$g = \frac{G}{R+G+B} \qquad \text{Equação 129}$$

$$b = \frac{B}{R+G+B} \qquad \text{Equação 130}$$

$$r+g+b = 1 \qquad \text{Equação 131}$$

$$V = \frac{1}{3}(R+G+B) \qquad \text{Equação 132}$$

Onde:

R,G e B: são, respectivamente vermelho, verde e azul no modelo RGB.

r,g e b: são respectivamente vermelho, verde e azul no modelo HSV.

V: valor ou brilho.

Para calcular o matiz (*hue*) usa-se a Equação 133:

$$H = \cos^{-1}\left(\frac{\frac{1}{2}[(R-G)+(R-B)]}{\left[(R-G)^2 + (R-B) + (G-B)^{\frac{1}{2}} \right]} \right) \qquad \text{Equação 133}$$

Onde:

H: é o matiz

R,G e B: são, respectivamente vermelho, verde e azul no modelo RGB.

O cálculo da saturação é dado pela Equação134:

$$S = 1 - \min(R, G, B) \qquad \text{Equação 134}$$

Onde:

S: saturação

R,G e B: são, respectivamente vermelho, verde e azul no modelo RGB.

O calculo de V (*value*), brilho foi apresentado na Equação132.

Para o controle do brilho altera-se convenientemente o valor de V. Para o controle da saturação altera-se convenientemente o valor de S. Uma vez obtidos os novos valores de S ou V, calcula-se as cores normalizadas r, g e b [6]:

Para $0^O < H \leq 120^O$:

$$b = \frac{1}{3}(1 - S) \qquad \text{Equação 135}$$

$$r = \frac{1}{2}\left[1 + \frac{S\cos H}{\cos(60^O - H)}\right] \qquad \text{Equação 136}$$

$$g = 1 - (r + b) \qquad \text{Equação 137}$$

Para $12\,0^O < H \leq 240^O$:

Fazendo:

$$H = H - 120^O \qquad \text{Equação 138}$$

Temos,

$$r = \frac{1}{3}(1 - S) \qquad \text{Equação 139}$$

$$g = \frac{1}{3}\left[1 + \frac{S\cos H}{\cos(60^O - H)}\right] \qquad \text{Equação 140}$$

$$b = 1 - (r + g) \qquad \text{Equação 141}$$

Para $240^O < H \leq 360^O$:

Fazendo:

$$H = H - 240^O \qquad \text{Equação 142}$$

$$g = \frac{1}{3}(1 - S) \qquad \text{Equação 143}$$

$$b = \frac{1}{3}\left[1 + \frac{S\cos H}{\cos(60^O - H)}\right] \qquad \text{Equação 144}$$

$$r = 1 - (b + g) \qquad \text{Equação 145}$$

Uma vez obtido os valores de r, g e b retorna-se ao modelo R, G e B fazendo [6]:

$$R = 3Vr \qquad \text{Equação 146}$$

$$G = 3Vg \qquad \text{Equação 147}$$

$$B = 3Vb \qquad \text{Equação 148}$$

3.5.7 Conversão de imagens coloridas para uma imagem em niveis de cinza

Para converter uma imagem colorida do modelo RGB para uma imagem em níveis de cinza usa-se a Equação 149:

$$Y = 0.299R + 0.587G + 0.114B \qquad \text{Equação 149}$$

Onde:

Y: brilho

R: nível da cor vermelha

G: nível da cor verde

B: nível da cor azul

Assim, para se converter uma imagem colorida para uma imagem em níveis de cinza, para cada pixel se calcula o seu brilho (Y), usando a Equação 149, e em seguida se faz R=G=B=Y, obtendo-se assim a conversão para uma imagem em níveis de cinza [8].

3.5.8 Correção Gama

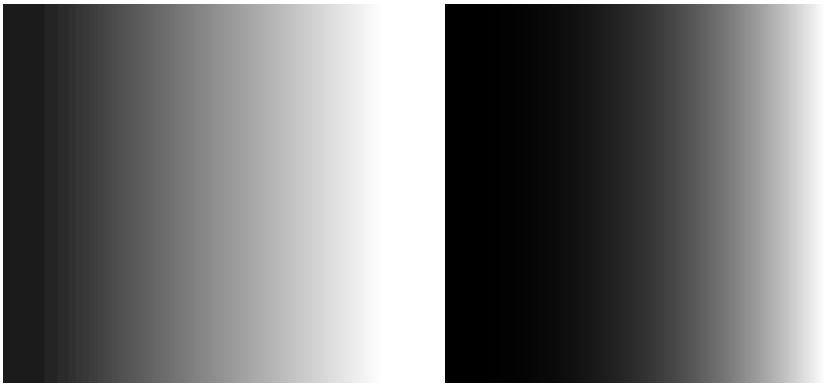
A correção gama é usada para corrigir distorções provocadas pela não linearidade da luminância mostrada na superfície de um tubo de raios catódicos em relação à tensão aplicada em seus terminais [19].

A luminância (I) mostrada na superfície de um tubo de raios catódicos é função da tensão aplicada em seus terminais (Vs). Esta função não é linear e é dada pela seguinte função:

$$I \sim V_S^{\gamma} \qquad \text{Equação 150}$$

Onde γ (gama) varia entre 1,5 e 3,0. O valor mais comum para um tubo de raio catódico é 2,5.

Para uma imagem com a tensão de branco 0,0V e a de preto 1,0V, todas as demais variações de cinza que ficam entre o intervalo 0,0V e 1,0V seriam alteradas quando visualizadas num tubo de raios catódicos. A Figura 21.a mostra a imagem que se esperaria obter quando se aplica um sinal de 0V a 1V num tubo de raios catódicos, variando linearmente. A imagem da Figura 21.b é a imagem resultante após a da aplicação de 0V a 1V num tubo de raios catódicos. Observa-se, na que imagem resultante só é igual à imagem esperada quando a tensão aplicada está em 0V. Os outros pontos apresentam uma luminância menor do que a imagem esperada.



a **b**

Figura 21: Imagem esperada x imagem obtida sem correção gama

Para compensar este efeito pode-se aplicar a função inversa sobre Vs, fazendo-se assim que a resposta de luminância apresentada pelo tubo de raios catódicos seja linear ao sinal de tensão aplicada. Assim:

$$V_C \sim V_S^{(1/\gamma)} \qquad \text{Equação 151}$$

Onde Vc é a tensão corrigida e Vs a tensão que seria originalmente aplicada. Para um tubo de raios catódicos com γ=2,5, o fator de correção gama (1/γ) seria 0,4.

A seguir é apresentado o algoritmo para correção gama para imagem preto e branco:

1. Encontram-se os pixels com o menor valor e o maior valor de níveis de cinza da imagem.
2. Os valores de níveis de cinza de cada pixel são normalizados no intervalo [0,1].
3. Cada pixel tem seu valor corrigido pelo fator de correção gama.
4. Obtêm-se os novos valores de níveis de cinza para cada pixel, multiplicando-se os valores corrigidos pela soma do maior valor do nível de cinza pelo menor valor do nível de cinza.

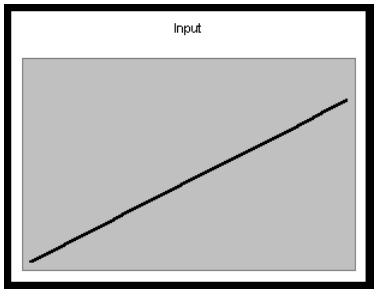
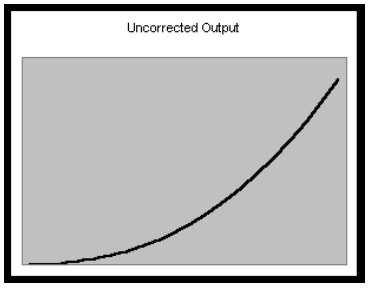
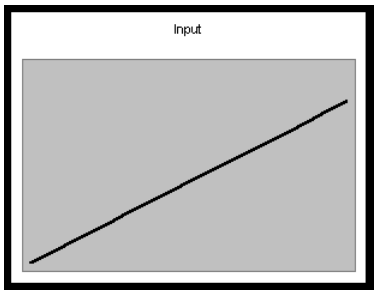
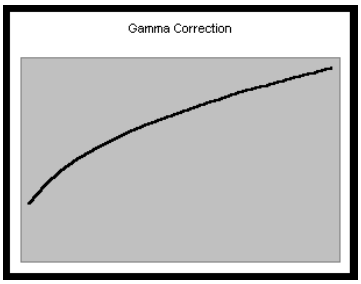
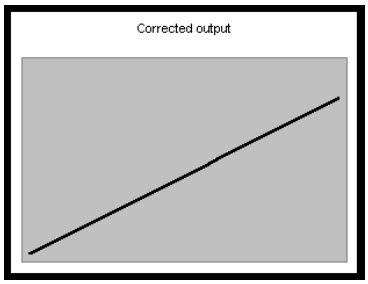
Sinal de luminância desejado em função de Vs	**Correção de gama**	**Sinal de luminância no tubo de raios catódicos**
	**Nenhuma**	
1. Luminância esperada em função da tensão aplicada (VS)	2. Nenhuma correção de gama é usada.	3. Curva de luminância apresentada pelo tubo de raios catódicos em função da tensão Vs.
		
1. Luminância esperada em função da tensão aplicada (VS)	2. Tensão aplicada nos terminais do tubo de raios catódicos com correção gama: Vc	3. Curva de luminância apresentada pelo tubo de raios catódicos em função da tensão Vs.

Figura 22: Influência da correção gama[1]

A Figura 22 mostram a relação entre a aplicação ou não da correção gama na tensão aplicada nos terminais do tubo de raios catódico e a curva de luminância resultante.

Atualmente existem monitores de hardware que fazem a correção gama automaticamente. A correção gama também se encontra disponível em sistemas operacionais como, por exemplo, o Windows XP.

[1] A Figura 21 foi obtida de: http://www.herbario.com.br/fotografia_digital/cap08.htm Acessado em: 21/07/2005

3.6 Segmentação de Imagem

A segmentação da imagem divide a imagem em suas partes constituintes. Os algoritmos de segmentação para uma imagem monocromática são baseados na descontinuidade ou na similaridade dos valores de níveis de cinza[6].

Quando se está trabalhando com a descontinuidade dos valores de níveis de cinza procura-se particionar a imagem baseado em mudanças bruscas nos níveis de cinza. Neste caso se procura detectar pontos isolados, linhas e bordas na imagem. [6].

O estudo da similaridade dos valores de níveis de cinza baseia-se em limiarização, crescimento de regiões, o particionamento e a fusão de regiões [6].

3.6.1 Detecção de Descontinuidades

Serão abordadas três técnicas básicas de detecção de descontinuidade em imagem digital a detecção de pontos, linhas e bordas. A maneira mais usual de se obter uma descontinuidade na imagem é através da varredura da imagem por uma máscara ou *kernel* de convolução. Considerando uma máscara genérica 3x3, como a mostrada na Figura 23 [6].

$$\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix}$$

Figura 23: Máscara genérica

Calcula-se a soma dos produtos dos coeficientes pelo nível cinza contido na região da imagem englobada pela máscara. A resposta da máscara em qualquer ponto da imagem é:

$$R = w_1 z_1 + w_2 z_2 + \ldots w_9 z_9 = \sum_{i=1}^{9} w_i z_i \qquad \text{Equação 152}$$

Onde z_i é o nível de cinza associado com o coeficiente w_i da máscara. A resposta da máscara é definida em relação a sua posição central. Quando a máscara é posicionada em um pixel da borda da imagem a resposta é computada utilizando-se a vizinhança parcial adequada [6].

3.6.1.1 Detecção de pontos

A detecção de um ponto isolado em uma imagem é obtida através do uso da máscara mostrada na Figura 24 [6].

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Figura 24: Máscara usada para detecção de pontos isolados

Um ponto na posição da máscara é encontrado se:

$$|R| \rangle T \qquad \text{Equação 153}$$

onde T é um limiar positivo e R é dado pela Equação 152. Basicamente o que se faz é medir a diferença ponderada entre um ponto e seus vizinhos. Considera-se que o nível de cinza de um ponto isolado é diferenciado em relação ao nível de cinza de seus vizinhos [6].

A máscara mostrada na Figura 24 é a mesma máscara usada na filtragem de alta-freqüência (Figura 16). Contudo, a ênfase aqui é a detecção de pontos, assim sendo. Apenas as diferenças acima do limiar T serão consideradas pontos isolados na imagem estudada.

3.6.1.2 Detecção de Linhas

A detecção de linhas requer um nível de complexidade um pouco maior levando em conta quatro máscaras mostradas na Figura 25 [6].

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

Figura 25 Máscara de detecção de linha horizontal, $+45^0$, vertical e -45^0

Considerando a máscara de detecção de linha horizontal, mostrada na Figura 25, se ela fosse movida por toda a imagem ela deveria responder fortemente as linhas (de largura de um pixel) orientadas horizontalmente. Se usarmos uma imagem de fundo fixo com uma linha horizontal (com nível de cinza mais alto) será sobre os pontos desta linha que a máscara terá resposta máxima. Similarmente a segunda máscara terá uma resposta máxima para uma linha com uma inclinação de $+45^0$. A terceira máscara terá a

resposta máxima para uma linha vertical e quarta máscara para uma linha com uma inclinação de -45^0.

3.6.1.3 Detecção de bordas

A detecção de bordas é a abordagem mais comum para a detecção de descontinuidades significativas dos níveis de cinza de uma imagem, uma vez que pontos e linhas não são ocorrências freqüentes na maioria das aplicações práticas [6].

Nesse estudo vamos considerar uma borda como o limite entre duas regiões com níveis de cinza relativamente distintos entre si. Vamos assumir que as regiões consideradas são suficientemente homogêneas, de forma que a transição entre as duas regiões pode ser determinada com base na descontinuidade dos níveis de cinza.

A idéia básica utilizada na maioria das técnicas de detecção de bordas consiste na computação de um operador diferencial. A Figura 26 mostra uma imagem (a) com uma faixa clara sobre um fundo escuro. O perfil de nível de cinza ao longo de uma linha de varredura horizontal da imagem a primeira derivada e a segunda derivada desta linha de perfil. Pelo perfil observa-se que a borda de transição (do escuro para o claro) é modelada como uma mudança suave de níveis de cinza, em vez de uma mudança abrupta. Este modelo reflete o fato que as bordas em imagens digitais são, geralmente levemente borradas devido à amostragem [6].

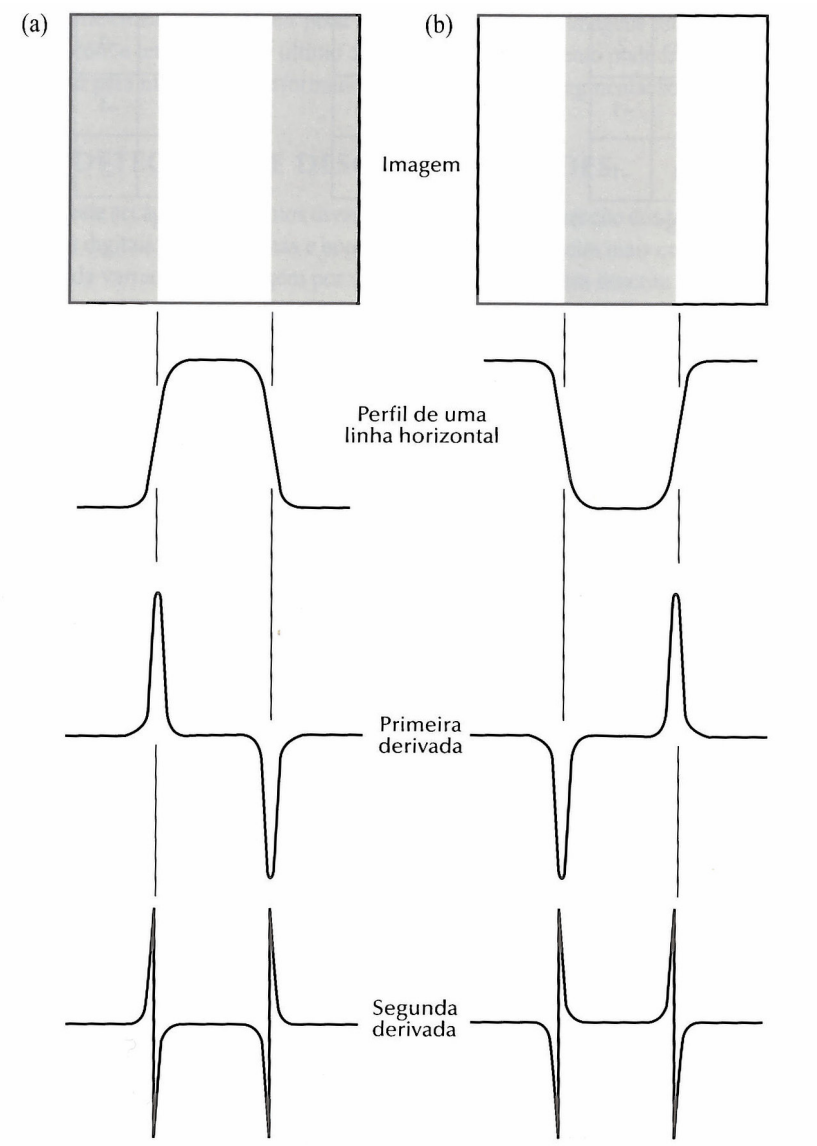


Figura 26: Detecção de bordas por operadores de derivação[2]

A primeira derivada do perfil de nível de cinza do perfil da imagem (a) é positiva na primeira borda (esquerda) e negativa na segunda borda (direita). Nula nas áreas de nível de cinza constante. A segunda derivada é positiva na transição entre a borda escura e a borda clara e negativa na transição entre a borda clara e a borda escura e nula nas áreas de nível de cinza constante. Portanto, podemos usar a magnitude primeira derivada para detectar a presença de uma borda na imagem e o sinal da segunda derivada pode ser usado para determinar se um determinado pixel localiza-se no lado claro ou no lado escuro da borda. Embora este exemplo tenha se limitado a um perfil horizontal unidimensional, uma argumentação similar se aplica a uma borda de qualquer orientação na imagem. Basta definir um perfil perpendicular a direção da borda em qualquer ponto desejado e interpretar o resultado[6].

[2] [Gonzalez, 2000], página 298

A primeira derivada em qualquer ponto da imagem é obtida usando a magnitude do gradiente naquele ponto. A segunda derivada é obtida utilizando-se o laplaciano.

3.6.1.4 Operadores Gradiente

Na Seção 3.5.1.4 introduzimos o conceito da utilização de gradiente para diferenciação de imagens. Da Equação 117 observa-se que o gradiente de uma imagem *f*(x,y) é dado pelo vetor:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \qquad \text{Equação 154}$$

Da análise vetorial tem-se que o vetor gradiente aponta na direção da mudança mais rápida de *f* na posição (x,y). Na detecção de bordas a magnitude deste vetor é uma quantidade importante, geralmente chamada simplesmente gradiente e denotada por ∇f onde[6]:

$$\nabla f = mag(\nabla f) = \sqrt{G_x^2 + G_y^2} \qquad \text{Equação 155}$$

Este valor é equivalente ao maior aumento de *f*(x,y), por unidade de distância, na direção ∇f .

Uma aproximação empregada para ∇f estima o gradiente com valores absolutos:

$$\nabla f = |G_x| + |G_y| \qquad \text{Equação 156}$$

Este resultada requer um menor tempo de processamento que o da equação anterior.

A direção do vetor do gradiente é um outro dado importante. Considerando $\alpha(x.y)$ o ângulo da direção do vetor ∇f na posição (x,y), tem-se que

$$\alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \qquad \text{Equação 157}$$

Onde o ângulo é medido em relação ao eixo x.

Considerando-se as Equação 154 e a Equação 155, nota-se que o cálculo do gradiente de uma imagem é resultado das derivadas parciais $\frac{\partial f}{\partial x}$ e $\frac{\partial f}{\partial y}$ na posição de cada pixel. Como apresentado em 3.5.1.4 o cálculo das derivadas pode ser feito de muitas formas

diferentes. Considerando uma região genérica da imagem, de tamanho 3x3, como mostrada na Figura 27 [6].

$$\begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix}$$

Figura 27: Região genérica de uma imagem de tamanho 3x3

Uma das aproximações para o cálculo do gradiente é mostrada na Equação 123, e considerando também a Equação 156 temos:

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \qquad \text{Equação 158}$$

$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \qquad \text{Equação 159}$$

As equações Equação 158 e Equação 159 podem ser implementadas através de duas máscaras conhecidas como operadores de Sobel mostrados na Figura 28

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Figura 28: Operadores de Sobel

Os operadores de Sobel têm como característica além de ressaltar as bordas suaviza a imagem. Esta característica é normalmente desejável visto que a derivação aumenta os ruídos presente na imagem.

3.6.1.5 Laplaciano

O laplaciano de uma função bidimensional é uma derivada de segunda ordem expressa pela Equação 160 [6]:

$$\nabla^2 = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad \text{Equação 160}$$

A Equação 160 pode ser aproximada de diferentes maneiras de uma forma similar do que foi feito com os gradientes.

Considerando novamente uma região 3x3, conforme mostrado na Figura 27 a forma mais freqüente encontrada na prática para o cálculo do laplaciano é dada pela Equação 161:

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8) \qquad \text{Equação 161}$$

E sua máscara é a mostrada pela Figura 29:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Figura 29: Máscara usada para implementação do laplaciano

Para o cálculo do laplaciano é necessário que o coeficiente associado com o pixel central seja positivo e que os outros associados aos pixels externos sejam negativos. Uma vez que o laplaciano é uma derivada, a soma de seus coeficientes tem que ser nula.

Embora o laplaciano responda a transição na intensidade ele raramente é usado na detecção de bordas. Isto se deve as seguintes razões:

1. É bastante sensível ao ruído por ser uma derivada de segunda ordem.
2. Produz bordas duplas.
3. Não é capaz detectar a direção da borda.

Por essas razões o laplaciano usualmente cumpre um papel secundário com um detector de bordas sendo usada para determinar se um pixel está no lado claro ou escuro da borda [6].

3.6.2 *Threshold* Global

O *threshold global* é uma operação que envolve teste de uma função T

$$T=T[x,y,p(x,y),f(x,y)] \qquad \text{Equação 162}$$

Onde:

f(x,y): nível de cinza do pixel *x,y.*

p(x,y): propriedade local deste pixel

Quando T depende apenas de *f(x,y)* o *threshold* (limiar) e dito global. Uma função de limiarização global ,*g(x,y)*, pode ser definida como [6]:

$$g(x,y) = \begin{cases} 255 & se \quad f(x,y) \geq T \\ 0 & se \quad f(x,y) < T \end{cases} \qquad \text{Equação 163}$$

3.7 O Padrão DICOM

3.7.1 Introdução

Com a introdução da tomografia computadorizada seguida de outras modalidades de diagnósticos por imagem, nos anos de 70, e com o incremento do uso de computadores em aplicações clínicas, o *American College of Radiology* (ACR) e a *National Electrical Manufactures Association* (NEMA) identificaram a necessidade de um método padrão para transferir imagens e informações entre equipamentos fabricados por diversos fabricantes. Até então, cada fabricante de equipamento desenvolvia um formato próprio de imagens digitais. [17]

O ACR e a NEMA criaram um comitê em 1983 com o objetivo de estabelecer um padrão para:

- Promover a comunicação de informação de imagens digitais independente do fabricante dos equipamentos
- Facilitar o desenvolvimento e o armazenamento de imagens e a comunicação entre sistemas (*Picture archiving and communications Systems*: PACS) [17]

A versão 1.0 do padrão ACR-NEMA foi publicado em 1985 (ACR-NEMA *Standards Publication* Nº 300-1985). O padrão sofreu duas revisões: a primeira datada de outubro de 1986 e segunda de janeiro de 1988.

A norma ACR-NEMA Publication Nº 300-1988, publicada em 1988, foi designada versão 2.0. Ela inclui a versão 1.0, as revisões publicadas e revisões adicionais. Ela também inclui um novo material que provê o suporte para comandos dos equipamentos de visualização, introduzindo um novo esquema de hierarquia para identificar uma imagem e adicionar dados específicos para a descrição de uma imagem.
Estas publicações do padrão especificam a interface de hardware, o conjunto mínimo de comandos de software, e um conjunto consistente de formatos de dados.
O padrão que é atualmente designado *Digital Imaging and Communication in Medicine* DICOM, incorpora a maioria dos melhoramentos das versões anteriores do padrão ACR-NEMA:

a. Esse padrão é aplicável a um ambiente de rede. O padrão ACR-NEMA versão 2.0 era aplicado somente em ambientes ponto a ponto; para operações em um ambiente de rede era necessário usar um *Network Interface Unit*

(NIU). O DICOM suporta operações em ambiente de rede usando o protocolo padrão TCP/IP.

b. Esse padrão é aplicável em um ambiente off-line. O padrão ACR-NEMA versão 2.0 não especificava um formato de arquivo ou escolha de meio físico ou um sistema lógico de arquivos. O padrão DICOM suporta operações off-line usando as mídias padrões industriais, como CD-ROM e sistemas lógicos de arquivos como o ISO 9660 e o sistema de arquivos usados em PC (FAT16).
c. O padrão DICOM especifica como os dispositivos conformes com o padrão reagem a comandos e dados que estejam sendo trocados. O padrão ACR-NEMA estava confinado a transferência de dados, mas o padrão DICOM especifica, através do conceito de classes de serviço, a semântica de comandos associados a dados.
d. O padrão DICOM especifica o nível de conformidade. O padrão ACR-NEMA especificava somente o nível mínimo de conformidade. O padrão DICOM explicitamente descreve como implementar uma estrutura (*Conformance Statement*) para selecionar opções específicas.

A descrição do padrão DICOM é composta de 18 partes:

- Parte 1 (PS 3.1): Introdução e visão geral
- Parte 2 (PS 3.2): Conformidade
- Parte 3 (PS 3.3): Informações sobre a definição de objetos
- Parte 4 (PS 3.4): Serviços e especificações de classes
- Parte 5 (PS 3.5): Estrutura de Dados e codificação
- Parte 6 (PS 3.6): Dicionário de dados
- Parte 7 (PS 3.7): Troca de mensagens
- Parte 8 (PS 3.8): Comunicação em rede e suporte para troca de mensagens
- Parte 9 (PS 3.9): Retirada
- Parte 10 (PS 3.10): Armazenamento em formato de arquivos para troca de dados
- Parte 11 (PS 3.11): Mídia de armazenamento e perfis de aplicação
- Parte 12 (PS 3.12): Formatos de mídia e mídia física para troca de dados

- Parte 13 (PS 3.13): Retirada
- Parte 14 (PS 3.14): Escala de cinza padrão e funções de visualização
- Parte 15 (PS 3.15): Perfis de segurança
- Parte 16 (PS 3.16): Conteúdo do mapeamento de recursos
- Parte 17 (PS 3.17): Informações explicativas
- Parte 18 (PS 3.18): Acesso à WEB para objetos DICOM persistentes (WADO).

Um arquivo DICOM contém um cabeçalho onde estão as informações do paciente (nome, tipo do equipamento, dimensões da imagem, e outros), assim como todos os dados da imagem (que pode conter informações em três dimensões). A imagem DICOM pode ser comprimida (encapsulada) para redução de tamanho. Os dados podem ser comprimidos usando variantes do formato JPEG com ou sem perdas de informação. Também pode ser usada a codificação e compressão sem perdas *Run-Length Encoding* (idêntico à compressão de bits encontrada em alguns formatos de imagem TIFF)[17].

3.7.2 O cabeçalho de um arquivo DICOM

O tamanho do cabeçalho pode variar dependendo das informações armazenadas. A Tabela 1, está definida no padrão DICOM e mostra as informações que podem estar presentes no cabeçalho de um arquivo.

Nome do Atributo	Tag	Tipo	Descrição
Preâmbulo do arquivo	(Sem tag)	1	128 bytes. Normalmente posto em 00H
Prefixo DICOM	(Sem tag)	1	4 bytes contendo o string "DICM". Este prefixo é usado para reconhecer que este é um arquivo DICOM
Tamanho de Grupo	(0002, 0000)	1	Número de bytes seguindo este elemento do meta arquivo (fim do campo de valores) até e incluindo o último elemento de informação do meta arquivo do Grupo 2 .
Informação da versão do meta arquivo	(0002, 0001)	1	Campo de 2 bytes onde cada bit identifica a versão do cabeçalho do meta arquivo
UID da Classe SOP da mídia de armazenamento.	(0002, 0002)	1	Identifica unicamente a Classe SOP associada com conjunto de dados. O UID da classe SOP permitido para mídia de armazenamento são especificados na PS3.11 do padrão DICOM – Mídia de armazenamento e perfis de aplicação
Instância de SOP da UID da mídia de armazenamento	(0002,0003)	1	Identifica unicamente a instância de SOP associada com o conjunto de dados colocado no arquivo seguindo as informações do meta arquivo.
UID da sintaxe de transferência	(0002, 0010)	1	Identifica unicamente a sintaxe de transferência usada para codificar o conjunto de dados. Esta sintaxe de transferência não se aplica as informações do meta arquivo.
Implementação da classe UID	(0002, 0012)	1	Identifica unicamente a implementação que escreveu o arquivo e seu contexto. Ele prove uma identificação não ambígua do tipo de implementação que escreveu o arquivo pela última vez no evento de problemas de troca. Segue a mesma política definida pela PS 3.7 do padrão DICOM (*association negotiation*)
Implementação do nome da versão	(0003, 0013)	3	Identifica unicamente a versão de implementação de uma classe UID usando 16 caracteres do repertório identificado na seção 8.5. Segue a mesma política definida pela PS 3.7 do padrão DICOM *(association negotiation*)
Fonte do título da entidade de aplicação	(0002,16)	3	O título da entidade de aplicação DICOM que escreveu o conteúdo do arquivo ou que fez as últimas alterações. Se usado, permite o rastreamento da fonte de erros no evento de problemas troca. A política associada ao título da entidade de aplicação segue aquelas definidas em OS 3.8 do padrão DICOM .

Nome do Atributo	Tag	Tipo	Descrição
UID privado do criador das Informações.	(0002, 0100)	3	Contém o UID privado do criador das informações.
Informações privadas	(0002, 0102)	1C	Contém as informações privadas postas nas informações do meta arquivo. O criador pode ser identificado em (0002,0100). É requerida se o UID privado do criador de informações (0002,0100) está presente.

Tabela 1: Cabeçalho usado pelo padrão DICOM [17]

Abreviações utilizadas:

- SOP : Service-object pair
- UID: Unique Identifier

O campo "Tipo" indica o requisito de um objeto:

(1): requerido

(2): deve estar presente mas pode ter o tamanho zero se desconhecido.

(3): opcional

(1C) similar ao (1) mas é incluído com se satisfizer uma condição.

(2C) similar ao (2) mas só pode ser incluído se satisfizer uma condição.

Maiores detalhes sobre o padrão DICOM podem ser encontrados em no link http://medical.nema.org/dicom/2004.html, que contém as 18 partes do padrão DICOM.

4 DESCRIÇÃO DO PROJETO

4.1 As Tecnologias Empregadas

O VPIM e seus módulos de expansão, também chamados de *plug-ins*, foram criados utilizando a linguagem C++ seguindo o modelo de programação orientado a objetos. A plataforma mínima de execução necessária para o programa é Windows 98SE utilizando DirectX 9c, atualização de agosto de 2005.

O DirectX9c é um conjunto de bibliotecas servem como interface de alto nível para funções de hardware e que é capaz de emular estas funções quando o hardware não é capaz de executar uma função especificada.

O ambiente de desenvolvimento recomendado para sua compilação é o Microsoft Visual C++ .NET 2003 com DirectX9c SDK instalado.

Na implementação da transformada rápida de Fourier foi utilizada a ARIS FFT [5].

O suporte a arquivos no padrão DICOM foi implementado utilizando o DicomToolkit [4]. A biblioteca libtiff [13] foi utilizada para o suporte a arquivos tiff e a biblioteca Zlib [23] é utilizada internamente pelo DicomToolkit sendo indiretamente utilizada pelo VPIM portanto.

Para o mecanismo de adição dos módulos de expansão o VPIM utiliza *Dynamic Link Libraries*, ou bibliotecas de vínculo dinâmico. O mecanismo de DLL é um recurso oferecido pelo sistema operacional para o qual o VPIM foi projetado e consistem em programas independentes cujos endereços de execução podem ser mapeados sobre os endereços de execução de outro programa de forma a permitir sua operação conjunta.

A comunicação entre os módulos é realizada através de interfaces abstratas que permitem a abstração da implementação de um módulo nos módulos que as utilizam. Essas interfaces abstratas são um recurso da linguagem C++ essencial para o funcionamento dos *plug-ins* que, criados como DLLs e portanto executáveis independentes, não podem ter compartilhamento de implementação.

Uma interface abstrata consiste em uma descrição de uma estrutura de dados contendo apenas cabeçalhos de funções. Essa estrutura, por sua vez, é transformada pelo compilador em uma tabela de ponteiros para funções que, ao ter um objeto instanciado, preenche a tabela com os ponteiros das implementações reais. Esse mecanismo garante

que todos os métodos apontados pela tabela serão válidos através de tratamento do compilador da linguagem C++, que não permite a instanciação de objetos que não definam todos os métodos abstratos de suas classes pai.

Muitas das implementações utilizando a FFT foram primeiramente implementadas usando o MATLAB 6.0. Estas implementações formaram a base para as atuais implementações em C++. O MATLAB também foi utilizado na validação das funções implementadas [21].

4.2 A Arquitetura

O VPIM é organizado em três principais módulos: o módulo núcleo, também chamado de *kernel*; o módulo das janelas, para interface com o usuário; e o conjunto de *plug-ins*. Esta divisão foi feita com o intuito de simplificar ao máximo a implementação e incorporação de novas funcionalidades que operem sobre os dados das imagens através da distribuição de responsabilidades entre os módulos.

As responsabilidades do *kernel* são: iniciar todos os subsistemas, executar o laço principal da aplicação com o sistema de coleção e despacho de eventos, carregar arquivos de imagem, finalizar a execução do programa e servir como ponto central de onde qualquer dado da aplicação é acessível direta ou indiretamente.

A responsabilidade do módulo das janelas é unicamente implementar a interface com o usuário. Essa implementação consiste no controle de exibição de seus dados individuais e em funções de suporte convenientes para outros módulos.

A responsabilidade do módulo composto pelo conjunto de *plug-ins* é a de expandir a aplicação em qualquer funcionalidade necessária. Os *plug-ins* desenvolvidos até o momento, entretanto, se restringem unicamente à aplicação de filtros sobre as imagens e eventual controle sobre interfaces com o usuário de que necessitem.

4.3 As Estruturas de Dados

As principais estruturas de dados do VPIM são suas interfaces abstratas e suas representações das imagens. Existem interfaces abstratas para o *kernel*, para cada uma das janelas do módulo de janelas, e para os subsistemas de dados que controlam a organização das imagens em estudos, séries e imagens dentro de uma série. As representações de imagens, por sua vez, são divididas em dois principais grupos:

representações de imagem no domínio espaço e representações de imagem no domínio freqüência.

4.4 As Imagens no Domínio Espaço

As imagens representadas no domínio do espaço utilizam uma representação por *scanlines* RGBA, ou seja, consistem de uma seqüência de bytes de tamanho igual a quatro vezes o produto da altura pela largura em *pixels* da imagem. O motivo para este valor está no fato de cada *pixel* ser representado por 4 *bytes*: 1 *byte* representando a componente vermelha da cor, 1 *byte* representando a componente verde da cor, 1 *byte* representando a componente azul da cor e 1 *byte* representando a opacidade do ponto.

Internamente a seqüência de *bytes* é organizada em um conjunto de linhas com um número de *pixels* igual ao número de colunas. Para cada *pixel* dentro de cada coluna o primeiro *byte* representa a componente azul de cor, o segundo a componente verde, o terceiro a componente vermelha e o quarto a opacidade.

Um *byte* utiliza 8 *bits*[3] de memória e pode representar 256 valores diferentes, de 0 (mais escuro) a 255 (mais claro), existindo, portanto essa variação de intensidades para cada canal de cor. Combinando todos os canais de cores são representáveis 2^{24} cores distintas, ou seja, 16777216 cores sendo que elas podem ter 256 níveis de transparência. Nesse espaço de cores é possível representar 256 níveis de cinza, pois uma cor no padrão RGBA é um nível de cinza se e somente se todos seus componentes de cor são iguais. A transparência foi apenas deixada para uso futuro.

4.5 As Imagens no Domínio Freqüência

As imagens médicas, por serem em níveis de cinza, têm seus componentes R, G e B iguais. As imagens representadas do domínio da freqüência são criadas a partir da componente verde (G) das imagens representadas no domínio do espaço transformada pela FFT.

Se a imagem no domínio do espaço não tiver dimensões em potência de 2, a imagem criada no domínio freqüência será criada com lados equivalentes à menor

[3] Um bit é a menor unidade de informação que um computador pode armazenar podendo ter apenas dois valores possíveis 0 ou 1.

potência de 2 maior que cada lado da imagem que tem potência de 2. O valor 0 será assumido para os novos pontos criados.

As imagens representadas do domínio da freqüência são representadas por *scanlines* de dois planos, real e imaginário de ponto flutuante de precisão dupla; ou seja, consistem de uma seqüência de dados do tipo *double* do C++ (ponto flutuante de 64 bits no padrão IEEE) de tamanho igual a duas vezes o produto da altura pela largura em *pixels* da imagem transformada para potência de 2.

Internamente a seqüência de *doubles* é organizada em dois conjuntos de linhas com um número de elementos igual ao número de colunas em cada conjunto. O primeiro conjunto de linhas representa a parte real da imagem e o segundo a parte imaginária.

4.6 A Interface Abstrata do Módulo *Kernel*

A interface avançada do módulo *Kernel* possui unicamente dois métodos: o método GetWindow e o método GetStudyManager. O método GetWindow recebe como parâmetro uma string padrão C[4] e retorna a interface abstrata CBaseWindowAbstractInterface. O método GetStudyManager não recebe parâmetros e retorna a interface abstrata CStudyManagerAdvancedInterface. A interface CBaseWindowAbstractInterface e os possíveis valores para o parâmetro do método GetWindow serão explicados no Seção 4.7 ,dedicada ao módulo de janelas.

A interface CStudyManagerAdvancedInterface tem a responsabilidade de disponibilizar as operações que podem ser realizadas sobre um estudo e gerenciar os estudos carregados na memória. Objetivando cumprir sua função ela disponibiliza as seguintes operações: a de criação de um novo estudo baseado em um identificador único de estudo, a de obtenção das interfaces abstratas dos estudos através do índice interno do estudo ou de seu identificador único, a de ordenação de estudos, a de liberação da memória de um estudo, a de cadastro de estudos com processamento pendente, a de fornecimento de uma fatia de tempo para atualização dos estudos com processamento pendente e o método reservado para uso interno ao VPIM que retorna a implementação concreta do gerenciador de estudos. Segue o cabeçalho C++ desta interface abstrata:

```
class CStudyManagerAdvancedInterface
```

[4] Uma string C é uma seqüência de bytes cujo delimitador é um byte com o valor 0. Ou seja, é a presença de um byte de valor 0 que determina o tamanho total da string.

```
{
 public:
   virtual CStudyAdvancedInterface*      CreateStudy(const char* szUID)=0;
   virtual const size_t                  GetNumberOfStudies()=0;
   virtual CStudyAdvancedInterface*      GetStudy(const size_t nStudyIndex) =0;
   virtual CStudyAdvancedInterface*      GetStudy(const char* szUID) =0;
   virtual CStudyAdvancedInterface*      GetNextStudy(const char* szCurrentStudyUID) =0;
   virtual CStudyAdvancedInterface*      GetNextStudy(CStudyAdvancedInterface* poCurrentStudy)
=0;
   virtual CStudyAdvancedInterface*      GetPreviousStudy(const char* szCurrentStudyUID) =0;
   virtual CStudyAdvancedInterface*      GetPreviousStudy(CStudyAdvancedInterface*
poCurrentStudy) =0;
   virtual const size_t                  UnloadStudy(const char* szCurrentStudyUID) =0;
   virtual void                          Run()=0;
   virtual void                          RegisterStudyRun(CStudyAdvancedInterface*  poStudy) =0;
   virtual void                          Finish()=0;
   virtual CStudyManager*                GetImplementation()=0;
};
```

Os métodos cujo tipo de retorno é “CStudyAdvancedInterface*” disponibilizam a interface abstrata responsável pela manipulação e acesso dos dados de um estudo. Essa interface permite: a criação de séries; a navegação pelas séries já na memória do estudo em questão através do identificador único da série desejada, ou iterando sobre os índices internos, ou mesmo requerendo o próximo estudo, ou estudo anterior de forma transitiva; descarregar da memória uma série; descobrir o identificador único do estudo em questão; realizar o controle de referências do estudo; fornecer uma fatia de tempo para atualização de seus dados internos e dos dados internos de suas séries que precisam de atualização; retornar o gerenciador de estudos. Essa interface também possui um método reservado para uso interno ao VPIM que retorna a implementação concreta do gerenciador de estudos.

Segue o cabeçalho C++ desta interface abstrata:

```
class CStudyAdvancedInterface
{
 public:
   virtual ~CStudyAdvancedInterface(){};
   virtual CSeriesAdvancedInterface*     CreateSeries(const char* szSeriesUID)=0;
   virtual const size_t                  GetNumberOfSeries()=0;
   virtual CSeriesAdvancedInterface*     GetSeries(const size_t nSeriesIndex)=0;
   virtual CSeriesAdvancedInterface*     GetSeries(const char*  szSeriesUID)=0;
   virtual CSeriesAdvancedInterface*     GetNextSeries(const char* szCurrentSeriesUID)=0;
```

```
   virtual CSeriesAdvancedInterface*       GetNextSeries(CSeriesAdvancedInterface*
poCurrentSeries)=0;
   virtual CSeriesAdvancedInterface*       GetPreviousSeries(const char* szCurrentSeriesUID)=0;
   virtual CSeriesAdvancedInterface*       GetPreviousSeries(CSeriesAdvancedInterface*
poCurrentSeries)=0;
   virtual const size_t                UnloadSeries(const char* nSeriesUID)=0;
   virtual const char*                 GetStudyUID()=0;
   virtual const size_t                GetNumberOfReferences()=0;
   virtual const size_t                AddReference()=0;
   virtual const size_t                RemoveReference()=0;
   virtual void                        Run()=0;
   virtual CStudy*             GetImplementation()=0;
   virtual CStudyManagerAdvancedInterface* GetStudyManager()=0;
};
```

A interface abstrata CSeriesAdvancedInterface funciona de forma análoga a interface CStudyAdvancedInterface. As principais diferenças entre suas funcionalidades estão, principalmente, nos contextos em que trabalham. Enquanto a CStudyAdvancedInterface é responsável pelo nível de estudo, a CSeriesAdvancedInterface é responsável pelo nível de série.

Sendo responsável pelo gerenciamento dos dados internos de uma série, esta interface, tem a responsabilidade de criar novas imagens, gerenciar as imagens e seu ordenamento. É também sua responsabilidade conhecer seu estudo pai. A fim de cumprir com suas responsabilidades métodos específicos foram criados.

Entre os métodos criados para esta interface que merecem particular atenção está o CreateImage. Este método, além de simplesmente criar uma imagem, ele também geram uma string no padrão C que é única para cada imagem carregada em cada sessão de execução do programa[5].

A necessidade da geração da string identificadora está no fato de que o padrão DICOM não possui um ordenador implícito de imagens e de que o programa abre imagens em padrões que também não possuem uma ordenação implícita. A string identificadora, portanto, gera uma ordenação única das imagens e, quando utilizado um arquivo no padrão DICOM, tenta ordenar a imagem primeiramente segundo os seguintes critérios: número da imagem, número da instância, posição da fatia e horário de criação. A formação completa do identificador único será explicada posteriormente ainda nessa Seção.

Segue o cabeçalho da interface CSeriesAdvancedInterface:

```
class CSeriesAdvancedInterface
{
 public:
   virtual ~CSeriesAdvancedInterface(){};
```

[5] Uma sessão de execução do programa é o intervalo em que o programa começa a ser executado até o momento em que ele é encerrado ou que o computador é desligado ou reiniciado.

```
   virtual CImageAdvancedInterface* CreateImage(const char* szFilename,const char* szCreationTime,
const unsigned int nInstanceNumber,const float fSlicePosition,const unsigned int nImageNumber=0)=0;
   virtual const char*                    GetSeriesUID()=0;
   virtual CStudyAdvancedInterface*       GetParentStudy()=0;
   virtual const size_t                    GetNumberOfImages()=0;
   virtual CImageAdvancedInterface*       GetImage(const size_t nImageContainerIndex)=0;
   virtual CImageAdvancedInterface*       GetImage(const char* szImageKey)=0;
   virtual const bool GetImageIndex(CImageAdvancedInterface* poCurrentImage, size_t& nIndex)=0;
   virtual const bool                  GetImageIndex(const char* szImageKey,size_t& nIndex)=0;
   virtual CImageAdvancedInterface*       GetNextImage(const char* szImageKey)=0;
   virtual CImageAdvancedInterface*       GetNextImage(CImageAdvancedInterface*
poCurrentImage)=0;
   virtual CImageAdvancedInterface*       GetPreviousImage(const char* szCurrentImageUID)=0;
   virtual CImageAdvancedInterface*       GetPreviousImage(CImageAdvancedInterface*
poCurrentImage)=0;
   virtual const size_t       UnloadImage(CImage*   poImage)=0;
   virtual const size_t       UnloadImage(const char* szImageKey)=0;
   virtual const size_t  GetNumberOfReferences()=0;
   virtual const size_t  AddReference()=0;
   virtual const size_t  RemoveReference()=0;
   virtual void               Run()=0;
   virtual CSeries*           GetImplementation()=0;
};
```

A interface abstrata CImageAdvancedInterface é significativamente diferente das anteriormente apresentadas nesta Seção. Enquanto nas anteriores a relação dos dados era de para cada elemento existiam N[6] outros sub-elementos do mesmo tipo, nesta interface, todos os seus sub-elementos estão presentes uma única vez internamente. A interface em questão, em função desta diferença, tem um funcionamento muito similar ao de um nodo folha em uma estrutura de dados contendo, portanto, o conjunto realmente relevante de informações para o usuário e para o programa que a utiliza.

Além da representação da imagem original e da imagem de trabalho do programa, essa interface encapsula os seguintes dados, quando presentes: a string identificadora da imagem, o espaçamento horizontal e vertical entre os pixels da imagem, o nome do arquivo a partir do qual a imagem foi criada, o horário de criação da imagem, o número da instância, a posição da fatia, o número da imagem, a identificação do paciente, o nome do paciente, a modalidade, a data do estudo, a hora do estudo, o número de acesso, o sexo do paciente, a data de nascimento do paciente, o nome da instituição onde a imagem foi criada, o status da imagem, o usuário do dispositivo de captura, o fator de zoom da imagem original, a descrição do estudo, o nome do equipamento de captura, o nome do tecnólogo responsável, o histórico do paciente, o nome de

[6] N representa um número arbitrário qualquer inteiro, positivo ou igual à zero.

nascimento da mãe do paciente, a idade do paciente, o nome do contraste, o tipo de exame e o radiologista responsável.

A seguir o cabeçalho da interface CImageAdvancedInterface.

```
class CImageAdvancedInterface
{
 public:
   virtual ~CImageAdvancedInterface(){};
   virtual const bool         SetOriginalTexture(IDirect3DTexture9* piOriginalTexture)=0;
   virtual void               SetWorkTexture(IDirect3DTexture9* piWorkTexture)=0;
   virtual IDirect3DTexture9*  GetOriginalTexture()=0;
   virtual IDirect3DTexture9*  GetWorkTexture()=0;
   virtual const bool         ResetWorkTexture()=0;
   virtual const char*              GetImageKey()=0;
   virtual CSeriesAdvancedInterface*    GetParentSeries()=0;
   virtual const size_t  GetNumberOfReferences()=0;
   virtual const size_t  AddReference()=0;
   virtual const size_t  RemoveReference()=0;
   virtual void               SetHorizontalPixelSpacing(const double dHorizontalPixelSpacing)=0;
   virtual void               SetVerticalPixelSpacing(const double dVerticalPixelSpacing)=0;
   virtual const double       GetHorizontalPixelSpacing()=0;
   virtual const double       GetVerticalPixelSpacing()=0;
   virtual const bool         HasValidPixelSpacingData()=0;
   virtual const char*        GetFilename()=0;
   virtual const char*        GetCreationTime()=0;
   virtual const unsigned int  GetInstanceNumber()=0;
   virtual const float        GetSlicePosition()=0;
   virtual const unsigned int  GetImageNumber()=0;
   virtual CImage*            GetImplementation()=0;
   virtual void SetPatientId(const char* szPatientId)=0;
   virtual void SetPatientName(const char* szPatientName)=0;
   virtual void SetModality(const char* szModality)=0;
   virtual void SetStudyDate(const char* szStudyDate)=0;
   virtual void SetStudyTime(const char* szStudyTime)=0;
   virtual void SetAccessNumber(const char* szAccessNumber)=0;
   virtual void SetPatientSex(const char* szPatientSex)=0;
   virtual void SetPatientBirthDate(const char* szPatientBirthDate)=0;
   virtual void SetStudyId(const char* szStudyId)=0;
   virtual void SetReferingPhisiciansName(const char* szReferingPhisiciansName)=0;
   virtual void SetInstitutionName(const char* szInstitutionName)=0;
   virtual void SetStatus(const char* szStatus)=0;
   virtual void SetUser(const char* szUser)=0;
   virtual void SetZoomFactor(const char* szZoomFactor)=0;
```

```
    virtual void SetStudyDescription(const char* szStudyDescription)=0;
    virtual void SetMachine(const char* szMachine)=0;
    virtual void SetTechnologistName(const char* szTechnologistName)=0;
    virtual void SetPatientHistory(const char* szPatientHistory)=0;
    virtual void SetPatientMotherBirthName(const char* szPatientMotherBirthName)=0;
    virtual void SetPatientAge(const char* szPatientAge)=0;
    virtual void SetContrast(const char* szContrast)=0;
    virtual void SetPerforming(const char* szPerforming)=0;
    virtual void SetRadiologist(const char* szRadiologist)=0;
    virtual void SetPixelSpacing(const char* szPixelSpacing)=0;
    virtual const char* GetPatientId()=0;
    virtual const char* GetPatientName()=0;
    virtual const char* GetModality()=0;
    virtual const char* GetStudyDate()=0;
    virtual const char* GetStudyTime()=0;
    virtual const char* GetAccessNumber()=0;
    virtual const char* GetPatientSex()=0;
    virtual const char* GetPatientBirthDate()=0;
    virtual const char* GetStudyId()=0;
    virtual const char* GetReferingPhisiciansName()=0;
    virtual const char* GetInstitutionName()=0;
    virtual const char* GetStatus()=0;
    virtual const char* GetUser()=0;
    virtual const char* GetZoomFactor()=0;
    virtual const char* GetStudyDescription()=0;
    virtual const char* GetMachine()=0;
    virtual const char* GetTechnologistName()=0;
    virtual const char* GetPatientHistory()=0;
    virtual const char* GetPatientMotherBirthName()=0;
    virtual const char* GetPatientAge()=0;
    virtual const char* GetContrast()=0;
    virtual const char* GetPerforming()=0;
    virtual const char* GetRadiologist()=0;
};
```

A geração da string C identificadora única de imagem para os objetos concretos que implementam a interface CImageAdvancedInterface é realizada através da seguinte especificação de formato, de acordo com o padrão de formatação da linguagem C para o comando printf: "%010d%010d%+#010.10E%s%s". Os parâmetros que seguem essa string de

formatação são um unsigned int [7]contendo o número da imagem, um unsigned int contendo o número da instância, um float [8]contendo a posição da fatia da imagem, uma string C contendo o horário de criação, um string C contendo o nome do arquivo. O número da imagem é um índice monotonicamente crescente interno ao VPIM e que garante a unicidade para até 2^{32} imagens carregadas simultaneamente que estejam empatadas em todos os outros critérios de ordenação.

4.7 O Módulo de Janelas

O módulo de janelas consiste em um conjunto de classes C++ que implementam o controle de exibição, das funcionalidades e dos dados internos das janelas do programa. Existem, para cada uma das janelas, duas interfaces: a CBaseWindowAbstractInterface e a sua respectiva interface avançada. Enquanto a CBaseWindowAbstractInterface é uma interface comum a todas as janelas, as interfaces avançadas apresentam variações para cada janela de acordo com seus dados e funcionalidades específicas implementadas.

Não existe relação hierárquica entre a interface avançada de uma janela e sua CBaseWindowAbstractInterface. A implementação da classe da janela, entretanto, herda e implementa indiretamente ambas as classes.

Como mencionado na Seção 4.6, a partir do módulo *Kernel* do programa, através do método GetWindow desse módulo, obtém-se acesso a um objeto que implementa a interface CBaseWindowAbstractInterface. A forma de determinar exatamente qual objeto, e portanto qual janela esse objeto controla, é utilizando o parâmetro "szWindowName" do método GetWindow. Esse parâmetro é o nome do objeto da janela sobre o qual se deseja operar. Cada classe de janela possui um nome e ele é único em todo o programa. O registro do nome é feito no construtor do objeto da janela e o registro de nomes é realizado antes da execução da função principal do programa (WinMain).

Segue a interface CBaseWindowAbstractInterface:

```
class CBaseWindowAbstractInterface
{
 public:
   CBaseWindowAbstractInterface(){};
   virtual ~CBaseWindowAbstractInterface(){};
```

[7] O tipo de dados unsigned int é um tipo de dados para números inteiros positivos e o 0 padrão da linguagem C. Ele, nesta implementação, tem 32 bits de largura de dados podendo, portanto, representar até 2^{32} números.
[8] O tipo de dados float é um tipo de dados para a representação de números reais da linguagem C. Ele é implementado utilizando o padrão de 32bits de ponto flutuante do IEEE.

```
   virtual const bool  IsInit()=0;
   virtual const char* GetName()=0;
   virtual const bool  Run()=0;
   virtual void*       GetAdvancedInterface()=0;
};
```

O método GetAdvancedInterface é o mais importante desta interface. Através da sua utilização é possível ter acesso as interfaces avançadas de cada janela. Note-se, entretanto, que este método tem como tipo de retorno "void*". O compilador, para este tipo de dado, não faz qualquer verificação em tempo de compilação para que outro tipo de dado ele seja convertido e, portanto, requer que quem utilizar esta interface conheça a que nome que interface avançada está associada para que seja feita a correta conversão. As interfaces avançadas e seus respectivos nomes serão listados posteriormente neste Seção.

O principal ponto de acesso do módulo de janelas é a classe que representa a janela principal do programa. O nome deste módulo é "MainWindow" e sua interface é a CMainWindowAdvancedInterface expressa a seguir:

```
class CMainWindowAdvancedInterface
{
 public:
   typedef  void(*TDCallback)(void *);
   virtual bool*                    GetInformationDisplayStatus()=0;
   virtual bool*                    GetInformationDisplay()=0;
   virtual bool*                    GetInformationDisplayChangedStatus()=0;
   virtual const bool   SetInformationTagDisplayMode(EImageInformationTags eTag,const bool
boShow)=0;
   virtual const bool   GetInformationTagDisplayMode(EImageInformationTags eTag, bool&
boStatus)=0;
   virtual const bool                ToggleInformationTagDisplayMode(EImageInformationTags
eTag)=0;
   virtual CImageWindowAdvancedInterface*  GetSelectedImageWindow()=0;
   virtual const bool  SetSelectedImageWindow(CImageWindowAdvancedInterface*
poImageWindow)=0;
   virtual void        NotifyImageWindowClick(CImageWindowAdvancedInterface*
poImageWindow)=0;
   virtual const bool  RegisterPluginCommand(const char*
szCommandName,CMainWindowAdvancedInterface::TDCallback  pfnCallback,void*
poOwner=NULL)=0;
   virtual const bool RegisterPluginCommand(const char* szCommandName,const char**
szMenuPath,const size_t nLevels,CMainWindowAdvancedInterface::TDCallback  pfnCallback,void*
poOwner=NULL)=0;
```

```
  virtual void                KillWindow(CImageWindowAdvancedInterface* poWindow)=0;
  virtual HWND                GetWindowHandle()=0;
  virtual void RedistribuiSeries(const unsigned int nLinhas, const unsigned int nColunas,const unsigned
int nLinhasImagens, const unsigned int nColunasImagens)=0;
};
```

O acesso às janelas foi projetado para acontecer sobre as janelas de imagem. Ele pode ser obtido através do método GetSelectedImageWindow, que retorna uma interface para objeto CImageWindowAdvancedInterface ou NULL, valor 0, caso não haja nenhuma janela de imagem selecionada. É necessário, sempre que este método for chamado, verificar se seu valor de retorno é NULL para evitar falhas catastróficas do sistema.

A CMainWindowAdvancedInterface possui, também, outros métodos de significativa importância para uso em conjunto com os *plug-ins*. Estes métodos são as duas sobrecargas do RegisterPluginCommand. Ambos registram criam comandos de menu para os *plug-ins* cujo nome será a string C passada no parâmetro szCommandName e em ambos, quando o comando do *plug-in* for ativado a função pfnCallback será invocada com o parâmetro poOwner sendo passado. A diferença entre ambos é que a sobrecarga que aceita menos parâmetros insere, por facilidade para o usuário, comandos somente no sub-menu filtros enquanto a versão de mais parâmetros permite colocar o comando em qualquer sub-menu ou criar novos sub-menus.

Para criar um novo sub-menu utilizando a versão com mais parâmetros do método RegisterPluginCommand é necessário passar para ele um vetor[9] de *string*s C e no parâmetro szMenuPath. No parâmetro nLevels o número de elementos do vetor deve ser passado.

A interface CImageWindowAdvancedInterface é a interface através da qual é possível obter o acesso a imagens e a dados de exibição da imagem. Ela é apresentada a seguir:

```
class CImageWindowAdvancedInterface
{
 public:
  virtual const bool    SetInformationTagDisplayMode(EImageInformationTags eTag,const bool
boShow)=0;
  virtual const bool    GetInformationTagDisplayMode(EImageInformationTags eTag, bool&
boStatus)=0;
  virtual const bool     ToggleInformationTagDisplayMode(EImageInformationTags eTag)=0;
```

[9] O significado de vetor nesse ponto do texto diz respeito ao jargão de linguagens de programação referindo-se a um conjunto de posições de memória contígua que contém uma seqüência de elementos de um mesmo tipo de dados.

```
   virtual const bool    SetCurrentImage(CImageAdvancedInterface*  poImage)=0;
   virtual const bool    UpdateImage()=0;
   virtual void GetImageRect(unsigned int& nLeft,unsigned int& nTop,unsigned int& nRight,unsigned
int& nBottom)=0;
   virtual void                    UpdateWindowDependencies()=0;
   virtual const HWND              GetWindowHandle()=0;
   virtual IDirect3DTexture9*      GetCurrentImage()=0;
   virtual CImageAdvancedInterface*    GetImage()=0;
   virtual double*                 GetImageComplexRepresentation()=0;
   virtual const bool              ComputeFastFourrierTransform(const bool boReverse=false)=0;
   virtual void GetComplexImageDimensions(unsigned int& nComplexWidth,unsigned int&
nComplexHeight)=0;
   virtual const bool              UpdateImageFromComplexRepresentationUsingOnlyRealPart()=0;
   virtual void                    SetMouseSelectionMode(const bool boSelectionMode)=0;
   virtual const bool              GetMouseSelectionMode()=0;
   virtual void                    ToggleMouseSelectionMode()=0;
   virtual void                    SetMouseSelectionRectangle(int nLeft,int nTop,int nRight,int
nBottom)=0;
   virtual const bool              CreateTempTexture()=0;
   virtual const bool              CommitTempTexture()=0;
   virtual const bool              DestroyTempTexture()=0;
   virtual const bool              SetBrightValue(const double dValue)=0;
   virtual const bool              SetSaturationValue(const double dValue)=0;
   virtual const bool              Run()=0;
   virtual CImageWindow*           GetImplementation()=0;
   virtual const unsigned int      GetNumberOfPaletteClasses()=0;
   virtual const unsigned int      GetPaletteClassDescriptors(PALETTEENTRY* astColor,unsigned
char* anLowerLimit,unsigned char* anHigherLimit)=0;
};
```

Existem duas formas através das quais o VPIM foi planejado para fornecer acesso dos dados de uma imagem para um *plug-in*: através da representação no domínio da freqüência da imagem e através da representação no domínio espaço. A representação no domínio do espaço pode ser obtida através do método GetCurrentImage, ele retorna a interface do DirectX através da qual é possível manipular a imagem em exibição na janela.

A representação no domínio da freqüência é obtida através do método GetImageComplexRepresentation. Este método, entretanto, só retornará um valor válido (diferente de NULL) caso antes o método ComputeFastFourrierTransform com o parâmetro boReverse com o valor *false* seja chamado.

É possível que as dimensões da imagem sejam diferentes de sua representação complexa. Em função disso, é sempre necessário chamar o método GetComplexImageDimensions para obter as dimensões da representação complexa.
Uma vez que as operações sobre a representação da imagem em domínio freqüência estejam completas, é necessário invocar o método ComputeFastFourrierTransform mais uma vez passando como parâmetro o valor *true*.

A representação complexa da imagem obtida através do método GetImageComplexRepresentation é baseada em scanlines de elementos do tipo double do C++, planar. Ou seja, primeiramente todo o plano real da imagem é representado com as dimensões obtidas em GetComplexImageDimensions e nas posições de memória adjacentes todo o plano imaginário da imagem.

4.8 Os *Plug-ins*

Os *plug-ins*, como mencionado anteriormente, são DLLs que expandem as funcionalidades do VPIM. Para que sejam utilizados pelo programa é necessário que eles exportem uma função de nome "CreatePlugin". O protótipo para está na interface CPluginAbstractInterface da qual, idealmente, o objeto principal do *plug-in* deve ser derivado. A seguir a interface CPluginAbstractInterface:

```
class CPluginAbstractInterface
{
public:
  typedef CPluginAbstractInterface* (*TDEntrypoint)(CKernelAdvancedInterface*);
  static VPIMAPI CPluginAbstractInterface* CreatePlugin(CKernelAdvancedInterface* poKernel);
};
```

A definição da macro VPIMAPI pode ser encontrada no arquivo "defines.h" do VPIM, estando também transcrita abaixo:

```
#ifdef _USRDLL
#define VPIMAPI __declspec(dllexport)
#else
#define VPIMAPI __declspec(dllimport)
#endif
```

A macro VPIMAPI permite que o VPIM compreenda que deve importar da DLL do *plug-in* a função "CreatePlugin" enquanto cada *plug-in* deve exportar essa função. É essencial, entretanto, que cada *plug-in* implemente a função CPluginAbstractInterface::

CreatePlugin retornando um novo objeto de sua classe concreta em sua implementação. O VPIM espera por esse comportamento.

Para garantir o correto comportamento da exportação dos símbolos da DLL é necessário adicionar um arquivo de definições ao projeto do *plug-in* com o seguinte conteúdo:

```
EXPORTS
    CreatePlugin   @1
```

Uma vez que estas condições sejam satisfeitas, e que o *plug-in* esteja no subdiretório Plugins do VPIM, ele será carregado quando o programa iniciar. Recomenda-se que no construtor do objeto concreto do *plug-in* seja registrado um comando de *plug-in* no menu da aplicação passando-se como parâmetro o valor *this* para o parâmetro poOwner do método RegisterCommand da janela principal do programa. Isso permitirá que quando a função registrada for chamada, seja possível acessar o objeto do *plug-in*.

4.9 Casos Especiais de Implementação

A implementação do VPIM foi normalmente a aplicação direta da teoria esplanada na revisão bibliográfica (Seção 3). Alguns casos especiais foram encontrados e suas diferenças são mostrados a seguir.

4.9.1 Implementação da Magnificação, Deslocamento e Rotação da Imagem

Para operações de magnificação da imagem, deslocamento e rotação é criada uma matriz de transformação que é passada para a placa de vídeo usando o método SetTransform da interface ID3DXSprite do DirectX9c. A matriz de transformação é normalmente aplicada na imagem pelo hardware da placa de vídeo. Caso o hardware não seja capaz de aplicar a matriz de transformação requerida *o hardware emulation layer* do DirectX9c aplica esta transformação [16].

4.9.2 FFT Bidimensional

A implementação da FFT foi feita usando-se a propriedade da separablidade da FFT (Seção 3.4.3.1). Assim, para cada cálculo de FFT de uma imagem, foi efetuado primeiro o cálculo da FFT unidimensional em uma das dimensões e sobre este resultado calculado a FFT sobre a outra dimensão.

4.9.3 Espectro de Potência da Imagem

Para facilitar a visualização do espectro de potência se fez o deslocamento da origem para o ponto de freqüência (N/2,N/2) usando-se a propriedade da periodicidade da transformada bidirecional de Fourier [6](Seção 3.4.3.3).

Como a variação da escala dinâmica do espectro de potência é, em geral muito, maior do que aquela possível de se reproduzir com fidelidade por um monitor de vídeo. Assim foi necessário a aplicação da função mostrada abaixo no espectro de potência [6]:

$$D(u,v) = c \times \log(1 + |F(u,v)|) \qquad \text{Equação 164}$$

Onde:

F(u,v): FFT da imagem original no ponto

c: Nível de cinza

Além da aplicação desta função também o VPIM gera o espectro de potência com remapeamento global (Seção 3.5.3) que também permite uma melhoria na imagem do espectro de potência.

4.9.4 Aplicação do Filtro de Butterworth

A função de transferência do filtro de Butterworth foi pensada para ser aplicada imaginando que a origem do domínio de freqüência da imagem estivesse localizada no ponto (N/2,N/2). Mas o resultado da transformada de Fourier tem sua origem no ponto (0,0). Conforme mostrado na Figura 30.a.

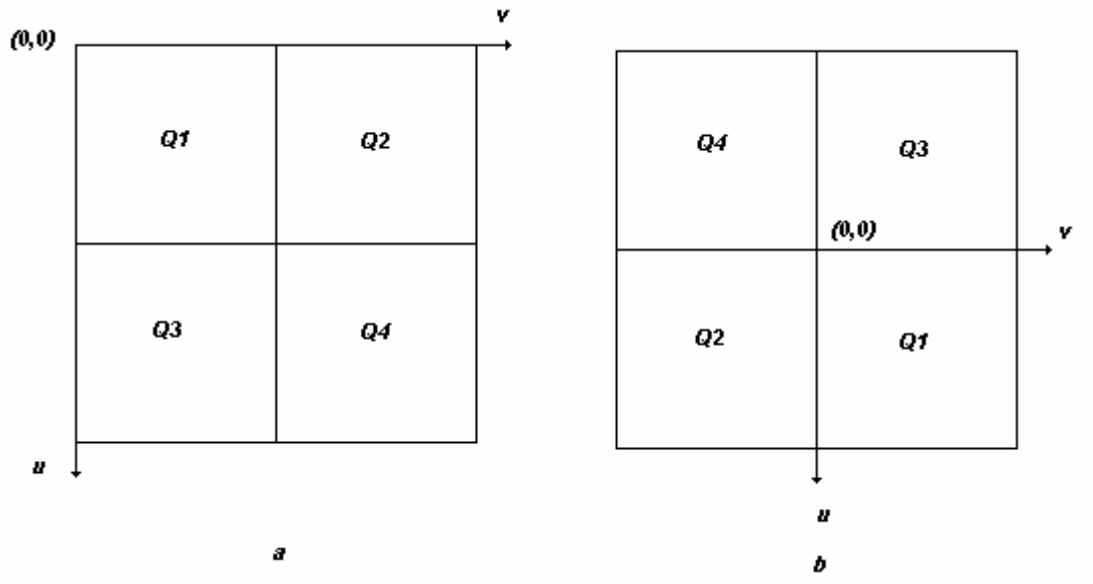


Figura 30: Coordenadas do espectro de potência

Uma solução para aplicar a função de transferência do filtro de Butterworth é, obter o cálculo da FFT, mover os quadrantes para deslocar a origem para centro (conforme o mostrado na Figura 30.b), aplicar a função de transferência, mover o resultado novamente para a posição original e obter a inversa da FFT.

Este método possui duas movimentações de memória desnecessárias. O que foi feito no VPIM foi calcular a posição para o qual o ponto iria se o centro fosse deslocado conforme mostra a Figura 30.b e aplicar a função de transferência. Com isto se evitou duas movimentações de memória otimizando o tempo de execução do filtro.

4.9.5 O Cálculo da Distância e Área

Para a realização do cálculo da distância entre dois pontos de uma imagem, o que acontece de forma análoga para o cálculo de áreas, é utilizado a distância euclidiana entre os pixels da imagem. Os pixels da imagem são calculados aplicando a matriz inversa de transformação sobre os pontos em que o mouse foi clicado.

Quando é utilizado um arquivo do formato DICOM, caso esteja presente e seja válido o campo DCM_PixelSpacing para as distâncias horizontal e vertical, a distância será calculada considerando-se os parâmetros do arquivo. Esta distância será dada em centímetros ou pixels se o campo DCM_PixelSpacing for respectivamente válido ou inválido. A distância vertical será multiplicada pelo valor vertical e o horizontal pelo seu respectivo pelo valor vertical. No caso de distâncias que tenham componentes

verticais e horizontais a distância será o resultado da raiz quadrada da soma dos quadrados da distância vertical e da distância horizontal.

4.9.6 Diferenciando Séries e Estudos

Os arquivos DICOM são organizados em uma estrutura hierárquica de informações. Parte da hierarquia diz respeito a organização entre estudos, séries e imagens.

Um estudo é o agrupamento mais abrangente de dados de imagens. Ele pode conter múltiplas séries e pode ser identificado no arquivo pelo campo DCM_StudyInstanceUID. As séries dentro de um estudo podem conter múltiplas imagens e podem ser identificadas pelo campo DCM_SeriesInstanceUID. Estudos e séries podem estar dispersos em múltiplos arquivos, mas os campos mencionados neste parágrafo possibilitam reagrupa-los logicamente independentemente disso.

4.10 Resultados Obtidos

Aqui serão mostrados os resultados obtidos com as operações de processamento de imagem. A maioria dos resultados aqui descritos foram obtidos tendo como imagem original a imagem do arquivo brain1234.png, mostrada na Figura 31. A imagem deste arquivo foi criada usando a imagem contida no arquivo brain_001dcm [15], que é uma secção transversal do crânio gerada por um equipamento de ressonância magnética. A esta imagem foram acrescentados os números 1, 2, 3 e 4, para facilitar a observação de alguns resultados. A imagem foi gravada num arquivo com o formato, *Portable Network Graphics* (png).

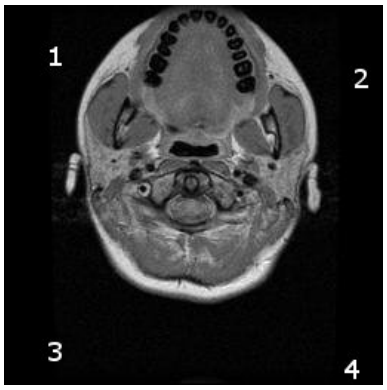


Figura 31: Imagem original do arquivo brain1234.jpg

As imagens da Figura 42.a, Figura 43.a, Figura 44.a, e da Figura 68 foram gerados no MATLAB. A aplicação do ruído impulsivo sobre a imagem da Figura 31, que é mostrado na Figura 50, foi efetuado no MATLAB. As imagens da Figura 67.a e da Figura 73.a foram obtida com uma câmara Kodak DX4330 (3.1 Mpixels).

4.10.1 Espelhamento

Na operação de espelhamento temos duas opções: o espelhamento horizontal e o espelhamento vertical. Na Figura 32 temos o resultado do espelhamento horizontal feito pelo VPIM. Na Figura 33 temos o resultado do espelhamento horizontal.

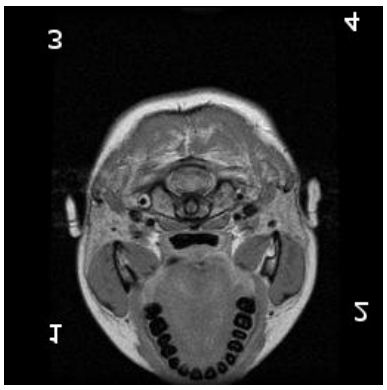


Figura 32: Imagem do arquivo brain1234 espelhada horizontalmente

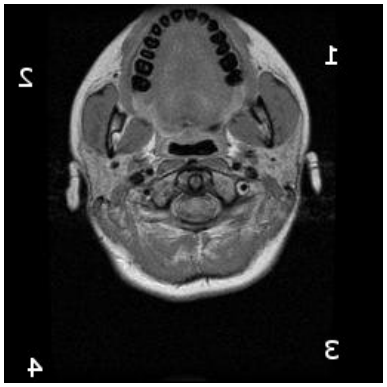


Figura 33: Imagem do arquivo brain1234 espelhada verticalmente

Em ambos os casos, fora o efeito do espelhamento não existe alteração na imagem. No espelhamento vertical e horizontal não há alteração nas estatísticas da imagem, nem no histograma das mesmas e nem nos valores dos pixels.

4.10.2 Rotação

Para a rotação também temos duas opções: horária e anti-horária. A rotação horária corresponde a uma rotação de +90º e a rotação anti-horária a uma rotação de -90º. A aplicação da rotação de horária pode ser observada na Figura 34 e a rotação anti-horária na Figura 35.

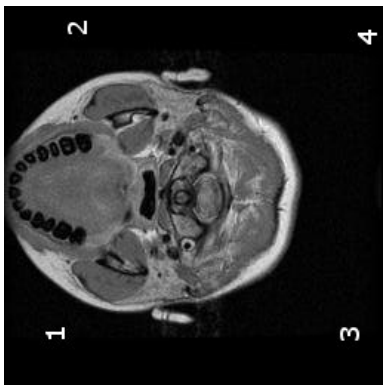


Figura 34: Imagem do arquivo brain1234 com uma rotação horária

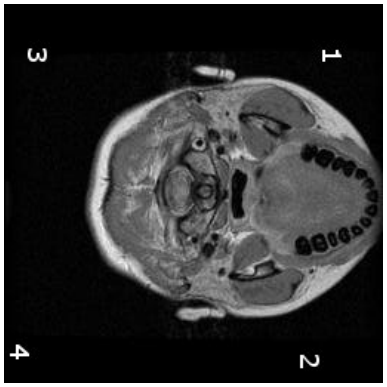


Figura 35: Imagem do arquivo brain1234 com uma rotação anti-horária

Por se tratar de ângulos de rotação de +90º e -90º além do efeito da rotação não há diferença nos níveis de cinza entre a imagem original e as imagens rotadas. Neste caso não existe erro de arredondamento. Na rotação horária e anti-horária não alteração nas estatísticas da imagem e nem no histograma da mesmas.

Além da rotação de +90º e -90º é possível rotar a imagem em ângulo predeterminado. O resultado da rotação em +22º é mostrado na Figura 36.

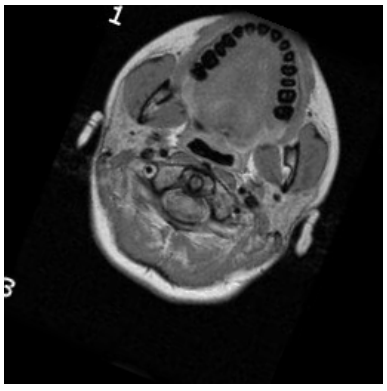


Figura 36: Imagem do arquivo brain1234 rotada em +22^{o}

4.10.3 Negativo da Imagem

O resultado da aplicação da função de transformação para a obtenção do negativo da imagem da Figura 31 é mostrado na Figura 37. Esta transformação normalmente altera as estátiscas e o histograma da imagem.

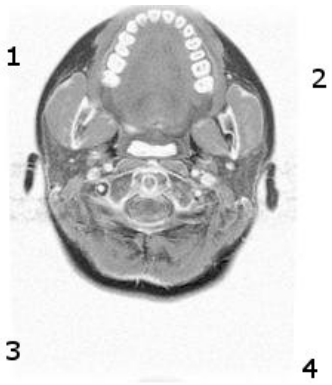


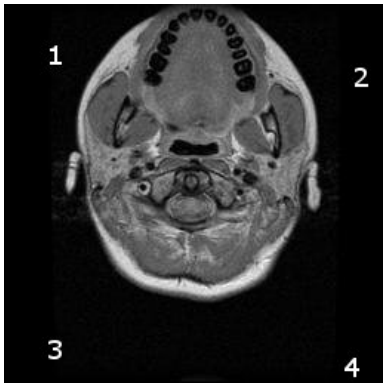
Figura 37: Negativo da imagem do arquivo brain1234

4.10.4 Magnificação e redução da imagem

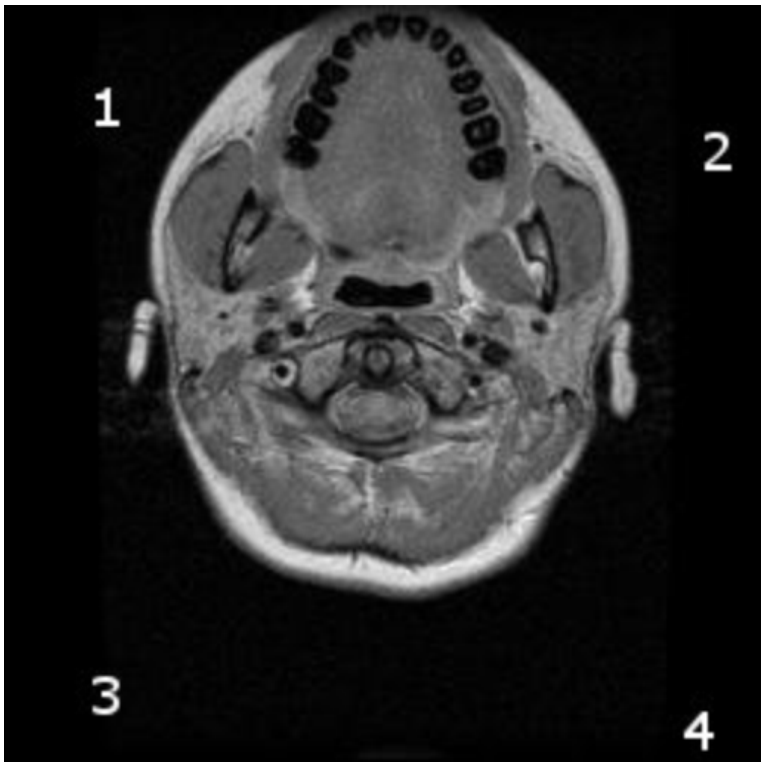
A magnificação da imagem, no VPIM, pode ser feita usando a interpolação linear ou a interpolação anisotrópica.

A imagem da Figura 38.a foi ampliada em 2x usando-se a interpolação anisotrópica. A interpolação anisotrópica é usada na ampliação sempre que sua execução for feita pelo *hardware* da placa de vídeo. O resultado da aplicação desta ampliação está mostrado na Figura 38.b.

Para fins de comparação, foram mantidas as proporções entre as imagens dos arquivos e as imagens mostradas a seguir.

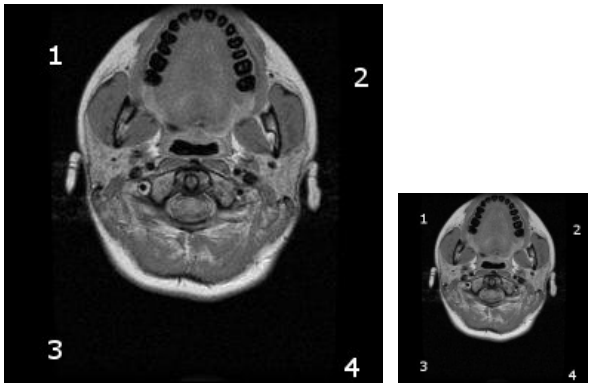


a



b

Figura 38: Ampliação de 2x usando interpolação anisotrópica



a **b**

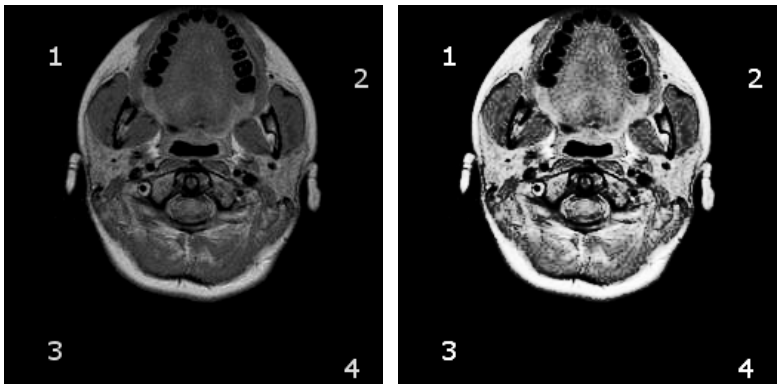
Figura 39: Redução de 2x usando média em 2D

No caso de redução da imagem foi usada a média em 2D, também executada pelo *hardware* da placa de vídeo. A Figura 39.b é o resultado de uma redução de 2x da Figura 39.b usando a média em 2D.

Nos dois casos, embora as imagens resultantes não sejam exatamente iguais as imagem originais, são visualmente muito próximos da mesma.

4.10.5 Remapeamento Linear

O remapeamento linear é uma técnica muito efetiva de realce da imagem, conforme pode ser visto a seguir.



a **b**

Figura 40: Exemplo de imagem remapeada globalmente

A Figura 40.a foi obtida através da subtração por 32 do nível de cinza de todos os pixels da imagem da Figura 31. Sobre a imagem da Figura 40.a foi efetuado o remapeamento global que é mostrado na Figura 40.b.

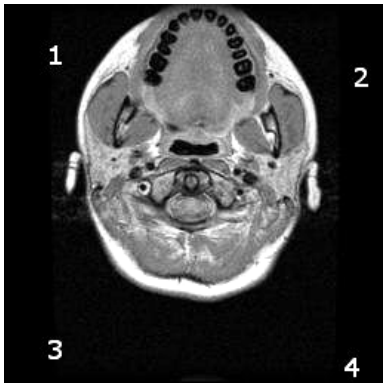
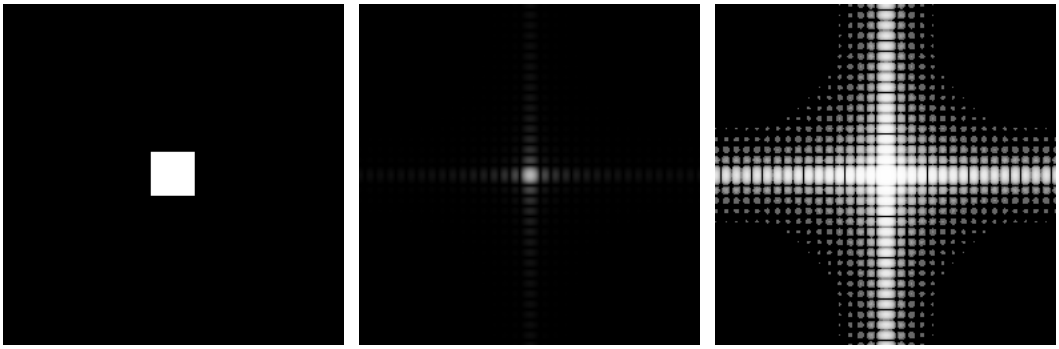


Figura 41: Imagem remapeada localmente

Um resultado melhor pode ser obtido com escolha adequada da área para remapeamento local. A Figura 41 é o resultado do remapeamento da imagem da Figura 31 usando os dados obtidos entre as linhas 60 e 100 e as colunas 100 e 130.

4.10.6 Espectro de potência da imagem

O VPIM calcula e mostra o espectro de potência de uma imagem. O VPIM tem duas funções para visualização do espectro de potência. Uma opção é visualizar o espectro de potência em escala logarítmica. A outra opção de visualização, disponível no VPIM, gerar o espectro de potência aplica a escala logarítmica e executa o remapeamento global. (ver Secção 3.5.3)

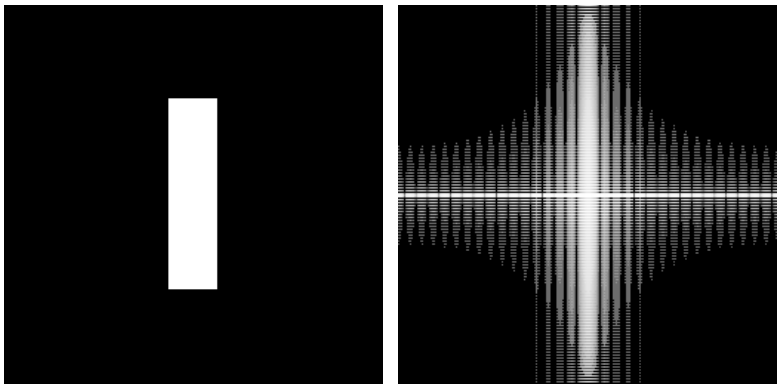


a **b** **c**

Figura 42: Espectro de potência de um quadrado

Na Figura 42.a temos a imagem original que é um quadrado branco sobre um fundo preto. O seu espectro de potência, em escala logarítmica, esta mostrado na Figura 42.b. Este resultado é mais apropriado para comparação de arquivos que para a visualização. Na Figura 42.c temos o resultado mais apropriado para a visualização, que além de aplicar a escala logarítmica sobre o espectro de potência, também é feito o remapeamento global.

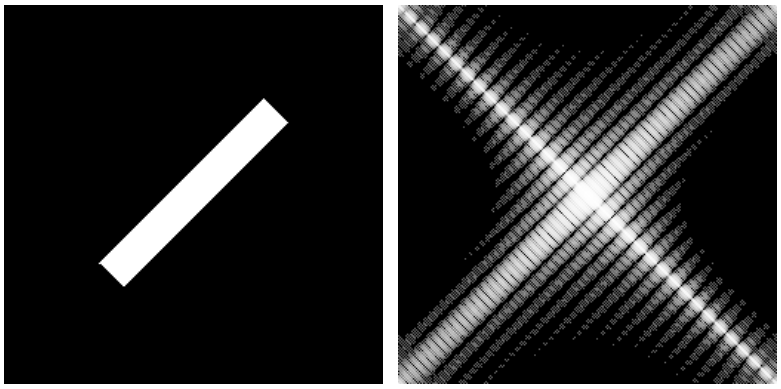
Para ilustrar melhor a geração do espectro de potência na Figura 43.a temos a imagem de um retângulo e de seu espectro (Figura 43.b).



a **b**

Figura 43: Imagem de um retângulo e de seu espectro

Na Figura 44.a temos o mesmo retângulo da Figura 43.a com uma inclinação de 45º. Note que o espectro de potência, mostrado na Figura 44.b tem uma inclinação de 45º em relação o espectro da Figura 43.b. Este resultado está de acordo com a propriedade de rotação da transformada de Fourier apresentado no Seção 3.4.3.4.



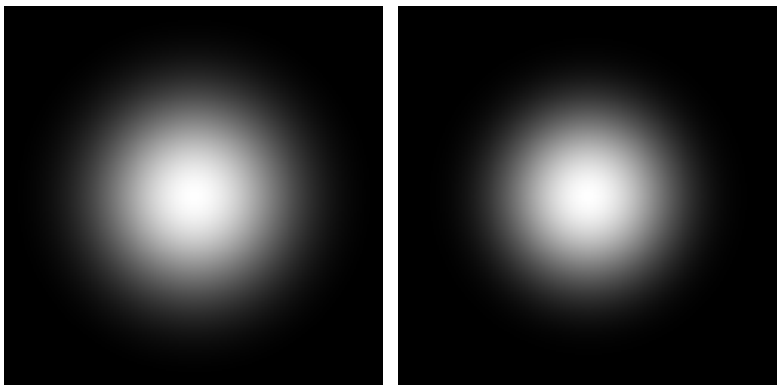
a **b**

Figura 44: Imagem de um retângulo inclinado e de seu espectro

4.10.7 Janelamento

Como apresentado na Seção 3.4.10 usa-se o janelamento para amenizar o espalhamento do espectro provocado pela multiplicação da imagem real (infinita) por uma janela quadrada. As duas máscaras de janelamento implementadas para reduzir o espalhamento do espectro da imagem foram a janela $\cos^4$ e a janela Blackman.

O resultado da aplicação da janela $\cos^4$ e da janela Blackman sobre uma imagem quadrada de 256x256 pixel toda branca pode ser visto na Figura 45. A Figura 45.a mostra a aplicação da técnica de janelamento blackman e Figura 45.b mostra a aplicação da técnica de janelamento $\cos^4$.



a **b**

Figura 45: Aplicação das janelas blackman e cos4

4.10.8 Filtros

No VPIM foram implementados os filtros de convolução no domínio espaço e convolução no domínio freqüência. São mostrados os resultados obtidos para os diferentes filtros implementados.

4.10.8.1 Filtros de Butterworth

Foram implementados os filtros de Butterworth passa-baixa, passa-alta, passa-faixa e elimina-faixa. Os resultados da aplicação destes filtros, com ordem igual a 2, podem ser vistos nas figuras à seguir.

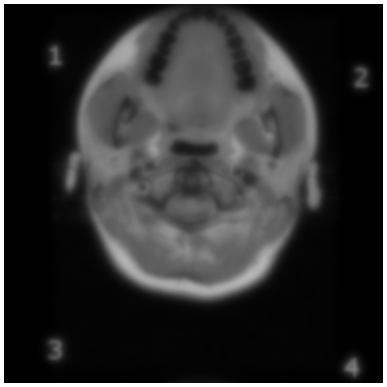


Figura 46: Aplicação do filtro de Butterworth passa-baixa fc=10

A Figura 46 foi obtida com a aplicação do filtro de Butterworth passa-baixa com freqüência de corte igual a 10. Neste caso nota-se o leve borramento da imagem provocado pela retirada das freqüências espaciais mais altas.

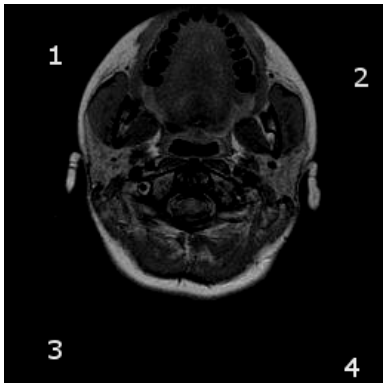


Figura 47: Aplicação do filtro de Butterworth passa-alta fc=2

Na Figura 47 é mostrado o resultado da aplicação do filtro de Butterworth passa-alta com freqüência de corte igual a 5. Como grande parte da informação está nas freqüências inferiores houve um escurecimento de grande parte da imagem. Nota-se que os números 1,2 3 e 4 da imagem, que são formadas por elementos de alta freqüência não sofreram muitas alterações com a aplicação do filtro.

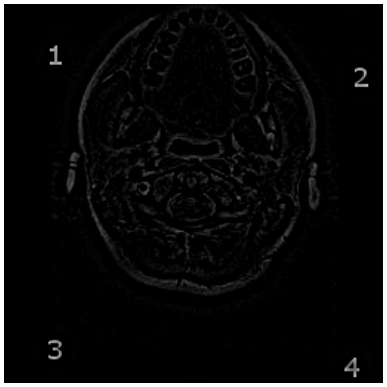


Figura 48 Aplicação do filtro de Butterworth passa-faixa com fci=50 e fcs=55

Na Figura 48 se vê o resultado da aplicação do filtro passa-faixa de Butterworth, com freqüência de corte inferior (fci) igual a 50 e freqüência de corte superior (fcs) igual a 55. Nota-se que os números são visíveis. Eles têm uma parte razoável do seu espectro nesta faixa de freqüência. O centro da imagem tem também componentes nesta faixa de freqüências embora em pequena proporção, devido a isto o centro da imagem resultante não é formado só de níveis de cinza igual a zero.

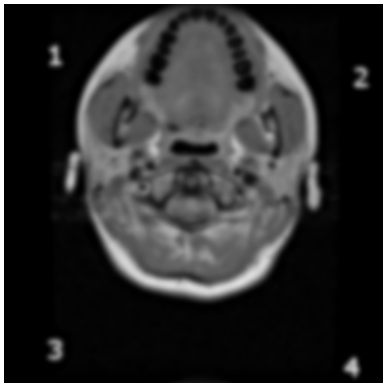


Figura 49: Aplicação do filtro de Butterworth passa faixa com fci=50 e fcs=55

Após a aplicação do filtro elimina faixa de Butterworth com freqüência de corte inferior (fci) igual a 50 e freqüência de corte superior igual a 55 nota-se pouca mudança na parte central da imagem porque praticamente não tem estas freqüências no seu espectro. Já os números sofreram um borramento devido à retirada destas freqüências.

4.10.8.2 Filtro da Mediana

Com visto na Seção 3.5.1.2 o filtro da mediana é um filtro não linear que é usado para reduzir o ruído impulsivo (*spike*) que possa estar presente nas imagens. Para verificação da implementação foi adicionado ruído impulsivo na imagem do arquivo brain1234.pgn. A imagem resultante é mostrada na Figura 50.

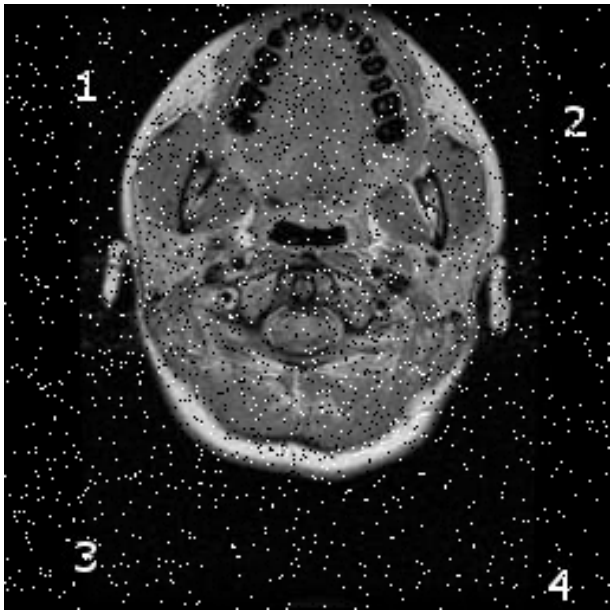


Figura 50: Imagem do arquivo brain1234 com aplicação de ruído de alta freqüência

Como pode ser visto na Figura 51 a aplicação do filtro da mediana praticamente restaurou a imagem original eliminando o ruído impulsivo e preservando a agudeza das bordas.

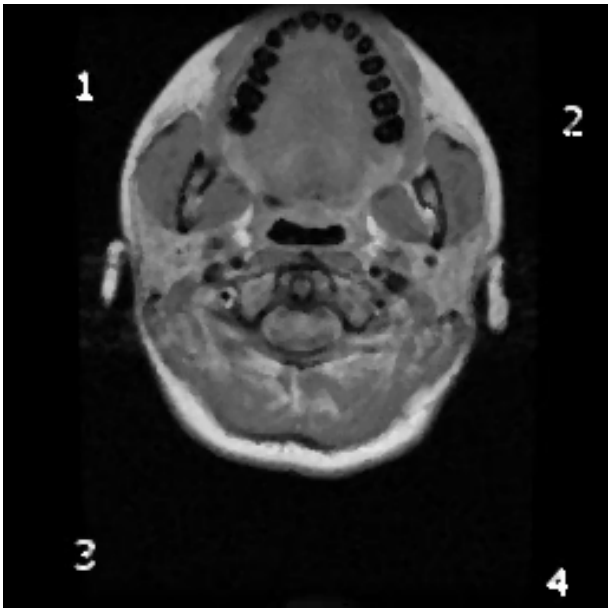


Figura 51: Resultado da aplicação do filtro da mediana

4.10.8.3 Filtro da média móvel

O filtro da média móvel é um filtro passa-baixa. Ele pode ser usado para filtrar ruídos ou atenuar altas freqüências.

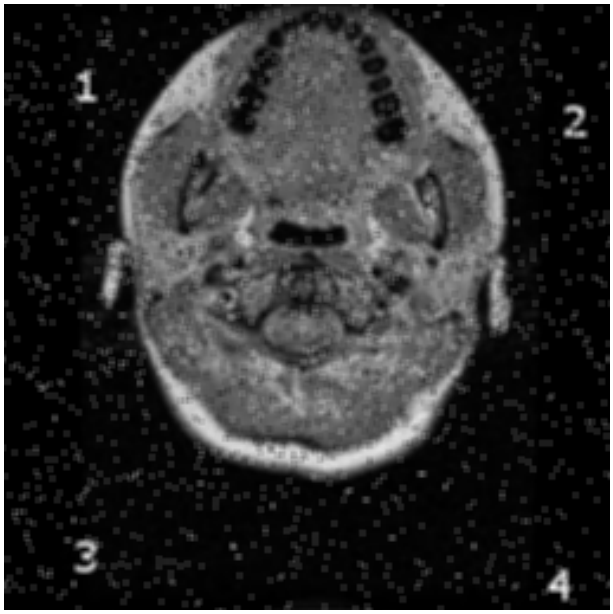


Figura 52: Aplicação do filtro da média móvel

Na Figura 52 temos o resultado da aplicação do filtro da média móvel sobre a imagem da Figura 50. Embora o ruído tenha sido atenuado os resultados para este tipo de ruído não são tão bons como o filtro da mediana.

4.10.8.4 Filtro passa-alta e filtro de alto reforço

O filtro passa-alta aplicado sobre uma imagem ressalta as mudanças bruscas atenuando as demais, como se pode ver Figura 53, que é do resultado da aplicação do filtro passa-alta na imagem do arquivo brain1234.pgn.

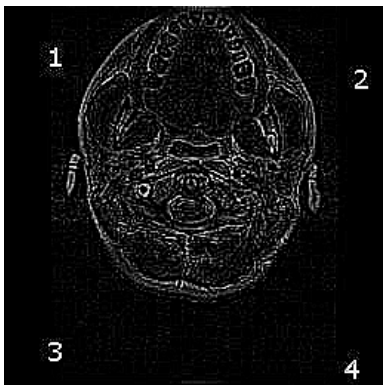


Figura 53: Aplicação do filtro passa-alta

O filtro de alto reforço também ressalta as transições bruscas de níveis de cinza, como ao resultado obtido pelo passa-alta é adicionado uma porção da imagem original, o filtro de alto reforço consegue ressaltar estas transições retendo boa parte da informação da imagem original. A aplicação do filtro de alto reforço sobre a imagem do arquivo brain1234.png pode ser visto na ver na Figura 54. Convém notar que na Figura 53 foi aplicado o remapeamento global.

Figura 54: Aplicação do filtro de alto reforço

4.10.8.5 Laplaciano e Filtro de Sobel

O laplaciano e o filtro de Sobel são usados para detecção de bordas. Os resultados da aplicação destes dois filtros podem ser vistos na Figura 55 e na Figura 56 respectivamente.

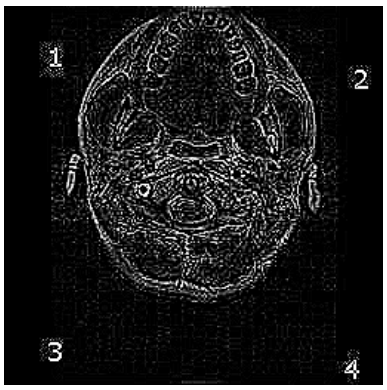


Figura 55: Aplicação do laplaciano

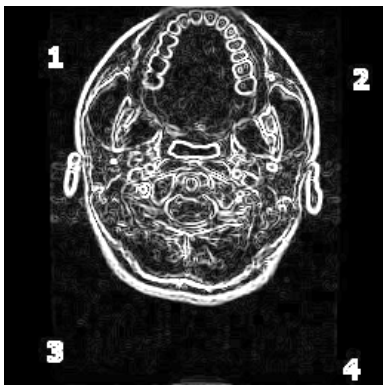


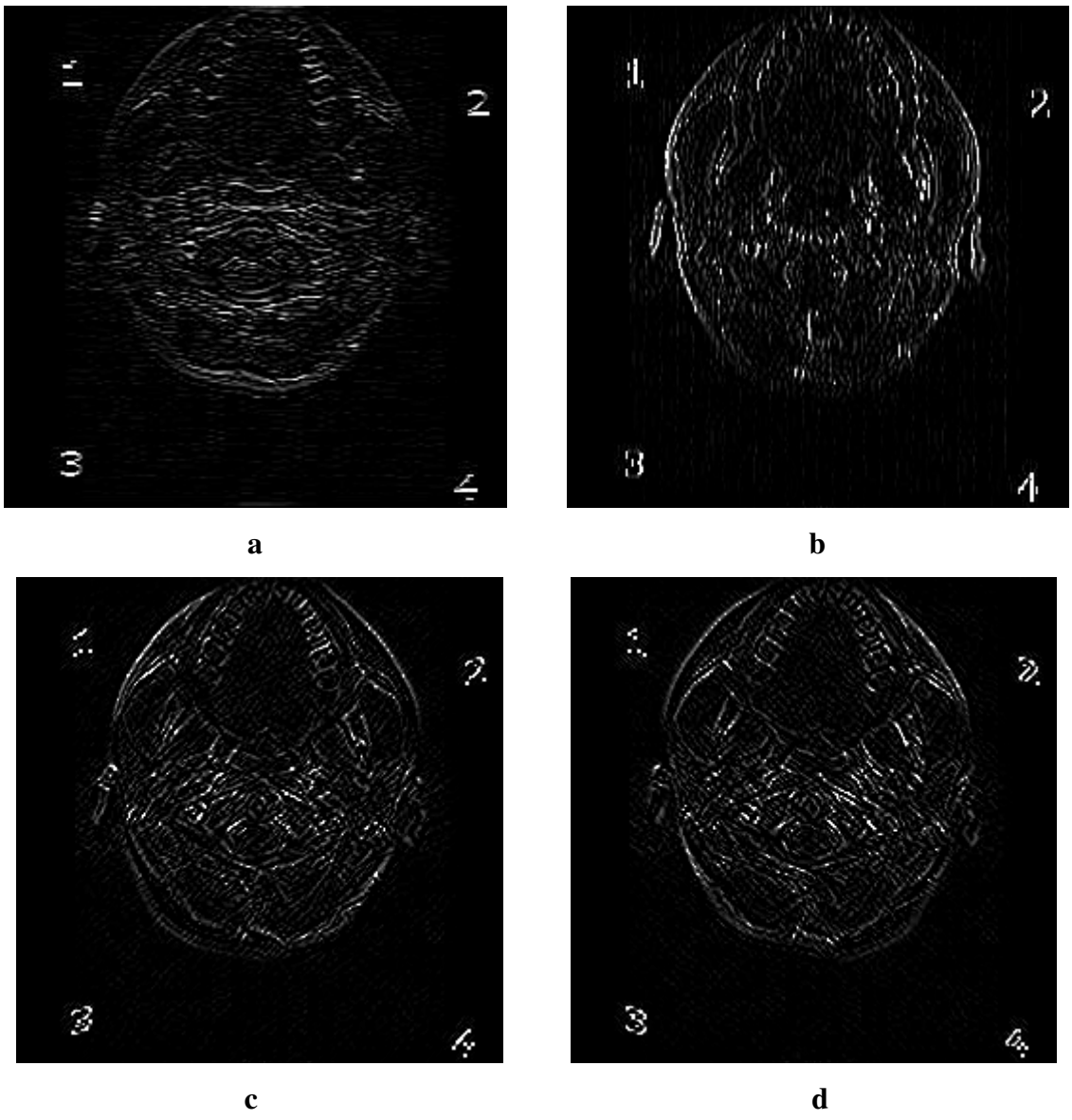
Figura 56: Aplicação do filtro de Sobel

Como pode ser visto destas duas imagens o filtro de Sobel apresenta um melhor resultado na caracterização das bordas.

4.10.8.6 Detecção de linhas

A Figura 57 mostra o resultado da aplicação dos filtros de detecção de na imagem do arquivo brain1234.png.

A Figura 57.a mostra a aplicação da detecção de linhas horizontais, a Figura 57.b mostra a aplicação do filtro da detecção de linhas verticais, a Figura 57.c mostra a aplicação do filtro de detecção de linhas com inclinação de +45º e a Figura 57.d mostra a aplicação do filtro de detecção de linhas com inclinação de -45º. Além do filtro foi efetuado um branqueamento da imagem para melhor visualização. Os pixels, com nível de cinza maior que 125, tiveram o seu valor alterado para 255 (branco).



a **b**

c **d**

Figura 57: Aplicação dos filtros de detecção de linhas

A detecção das linhas horizontais e verticais pode ser bem notada tanto no centro como nos números nas bordas da Figura 57.a e Figura 57. Já a detecção das linhas de +45º e -45º aparecem melhor nos números nas bordas.

4.10.9 Resolução

No VPIM existem dois comandos para alteração da resolução que são: decimação e a extensão.

A decimação diminui a resolução da imagem eliminando amostras. A Figura 58 mostra a imagem resultante da aplicação de decimação com redução da freqüência de amostragem em duas vezes.

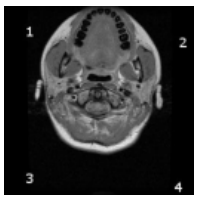


Figura 58: Resultado da aplicação da decimação

Esta técnica é semelhante às técnicas inicialmente usadas para fazer redução da imagem (Seção 3.3.2). Apresenta resultados próximos ao da redução de escala pela média mostrados na Figura 39.

A técnica de extensão aumenta a resolução em freqüência através do acréscimo de zeros a partir do fim da imagem.

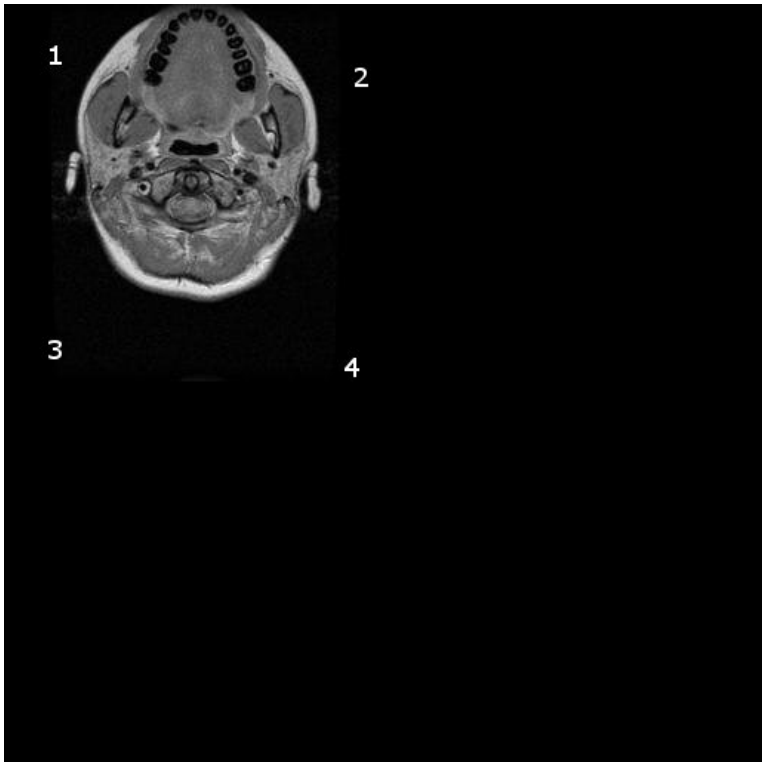


Figura 59: Aplicação da técnica de extensão

Na Figura 59 pode ser vista a aplicação da técnica de extensão, sobre a imagem da Figura 39, cuja resolução em freqüências foi estendida em 2x com a adição de zeros.

4.10.10 Recorte

No recorte se forma uma nova imagem formada por uma região de interesse (ROI) da imagem original. O VPIM permite a escolha seleção da ROI de duas formas:

1. pelo teclado através da digitação posição horizontal e vertical iniciais e finais,
2. pelo mouse construindo um retângulo sobre a área desejada.

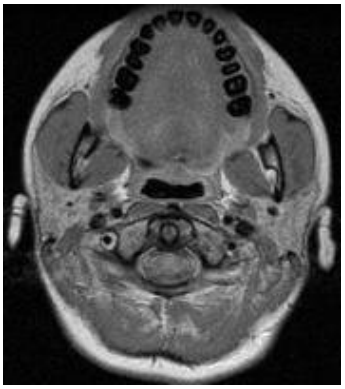


Figura 60: Resultado da aplicação de um recorte

A Figura 60 foi feita com um recorte da imagem da Figura 31, usando o teclado, e considerando como sendo a ROI as linhas de 0 a 200 e as colunas de 39 a 217. Com isto foram retirados os números e boa parte do fundo que compunha a imagem.

4.10.11 Combinação

Nas técnicas de combinação usam-se duas imagens, I_1 e I_2 , para formar a imagem resultante I_R. As técnicas de combinação implementadas no VPIM foram: pixel alternado, subtração e combinação binária.

4.10.11.1 Técnica de Pixel Alternado

Na combinação através de pixel alternado obtém-se uma nova imagem formada por pixels alternadamente de I_1 e I_2 para se obter a imagem resultante I_R

A imagem mostrada na Figura 61 é um resultado da técnica de combinação binária usado pixels alternados de duas imagens. Para conseguir este resultado primeiro foram combinadas as imagens da Figura 57.a com a da Figura 57.b. O seu resultado foi combinado com resultado da combinação entre as Figura 57.b e Figura 57.c. Para que a imagem ficasse mais visível na impressão foi feito um branqueamento dos pontos em níveis de cinza acima de 70.

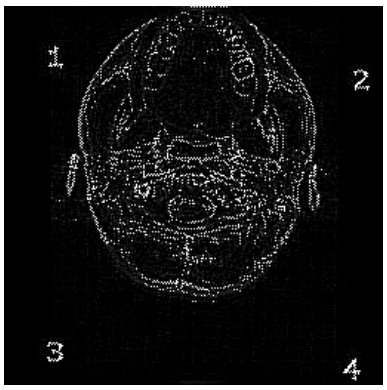
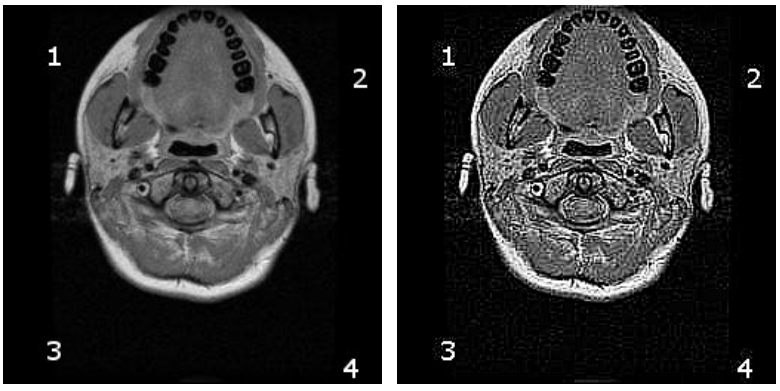


Figura 61: Exemplo de combinação de imagem usando pixel alternado

4.10.11.2 Combinação por subtração de imagens

Nesta técnica a imagem resultante (I_R) é resultado da subtração dos níveis de cinza pixel a pixel das imagens I_1 e I_2. Esta técnica pode ser usada para avaliar os efeitos dos filtros sobre uma imagem, possibilitando a verificação do o que foi perdido ou introduzido na imagem. Na Figura 62.a temos a imagem original (I_1) a qual foi aplicado o filtro de alto reforço gerando a imagem que é mostrado na Figura 62.b (I_2).



a **b**

c

Figura 62: Exemplo de combinação por subtração de imagem

A diferença entre I_1 e I_2 (I_R), é mostrada na Figura 62.c. Este resultado corresponde a aplicação de um filtro pasa-alta sobre a imagem da Figura 62.a.

4.10.11.3 Combinação Binária

Nesta técnica para combinar duas imagens I_1 e I_2, se fornece um valor de nível de cinza para comparação. Caso, para um determinado pixel, o valor de nível de cinza forem maior do que o valor fornecido o pixel correspondente da imagem resultante será de I_1 senão será de I_2.

Figura 63: Exemplo de combinação binária

A Figura 63 é um exemplo de combinação binária. Nela foi usada a imagem da na Figura 54 como imagem I_1 e a Figura 31 como I_2. O valor de comparação foi feito igual a 190.

4.10.12 Outros

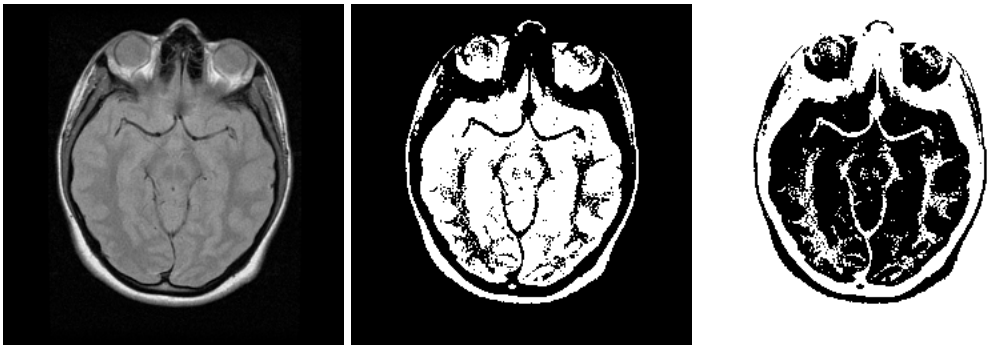
Nesta Seção foram agrupados uma série de técnicas implementadas no VPIM. Seus resultados estão mostrados a seguir.

4.10.12.1 Threshold Global

Para formar a imagem resultante usado a técnica de *threshold* Global se fornece um valor de nível de cinza para comparação. Se o valor de nível de cinza de um pixel da imagem original for maior ou igual do que o valor de comparação ele é convertido para o branco (valor do nível de cinza igual a 255), os pixels com níveis de cinza serão

mudados para preto (valor do nível de cinza igual a 0). Pode-se também inverter o resultado fazendo que os pixels que tenha o valor de nível de cinza maior ou igual ao valor de comparação sejam convertidos para o preto e os pixels que tenham seu valor de nível de cinza menor que o valor de comparação sejam mudados para o branco.

Esta imagem foi obtida usando o a imagem contida no arquivo brain_010dcm. A imagem original é a da Figura 64.a [15].



a **b** **c**

Figura 64: Exemplos de aplicação de *threshold* global

Na Figura 64.b é mostrado aplicação da técnica de *threshold* global, com conversão para o branco dos pixels com valor do nível de cinza maior ou igual a 118. Na Figura 64.c manteve-se o valor de comparação, mas se fez a conversão para o preto.

4.10.12.2 Remoção de Fundo

Na técnica de remoção de fundo um determinado valor fornecido é subtraído do valor do nível de cinza de cada pixel.

Um das utilizações desta técnica é na preparação de imagens para um posterior processamento. Na Figura 65 é mostrado o resultado da aplicação do remapeamento local sobre a imagem da Figura 31. Embora não seja visível n imagem da Figura 31, ela tem um ruído de fundo. Ao se efetuar o remapeamento este ruído foi também realçado.

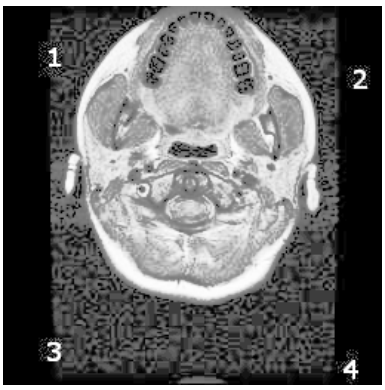
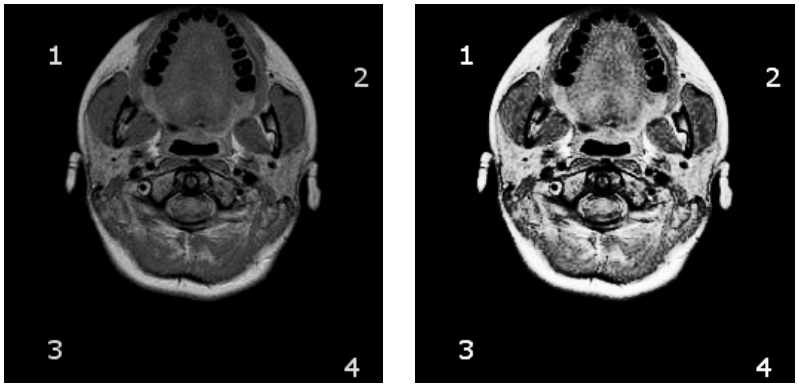


Figura 65: Remapeamento global sem remoção de fundo

A imagem da Figura 66.a foi obtida mediante a subtração por 32 do nível de cinza de cada um dos pixels da imagem da Figura 31. Sobre esta imagem foi aplicado o remapeamento global mostrado na Figura 66.b.



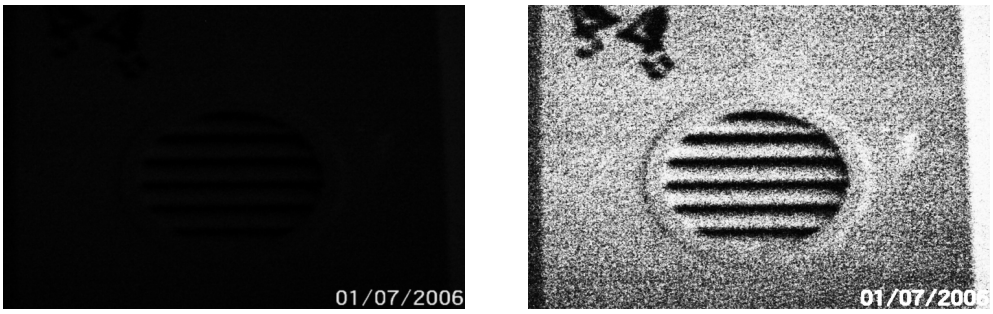
a **b**

Figura 66: Remapeamento global com remoção de fundo

O resultado da remoção de fundo com a posterior aplicação do remapeamento global melhorou os resultados obtidos conforme pode ser visto comparando as imagens da Figura 65 e da Figura 66.b.

4.10.12.3 Equalização de Histograma

O resultado da aplicação da equalização de histograma sobre a imagem da Figura 67.a. é mostrado na imagem da Figura 67.b



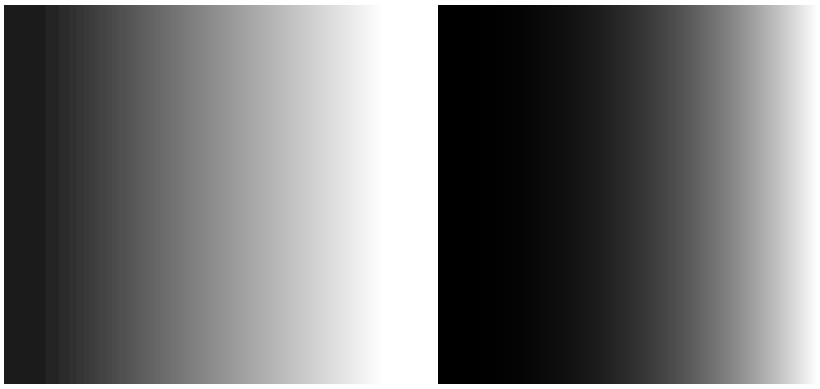
a **b**

Figura 67: Aplicação da equalização de histograma

4.10.12.4 Correção Gama

A correção gama além de poder ser usada para corrigir distorções causadas pelo tubo de raios catódicos, casa esta correção não esteja disponível monitor, também pode ser usado como uma técnica não linear de realce da imagem.

A Figura 68.a mostra uma imagem com níveis de cinza variando linearmente, em cada linha, de 0 a 255. Se esta imagem for aplicada num monitor de vídeo sem correção do gama, a imagem que será mostrada pelo monitor de vídeo será o da Figura 68.b. Esta imagem que foi gerada no MATLAB simulando uma imagem que resultaria da aplicação sobre um monitor de tubos de raios catódicos com fator gama 2.5, sem correção gama.



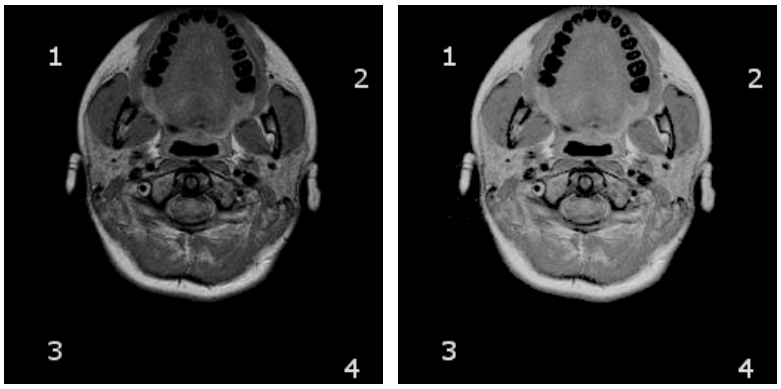
a **b**

Figura 68: Imagem simulando fator gama 2.5

A Figura 69 é o resultado da aplicação do fator de correção gama 0.4 sobre a imagem da Figura 68.b no VPIM. Nota-se que a imagem que tinha uma variação exponencial de níveis de cinza passou a ter uma variação linear de níveis de cinza.

Figura 69: Imagem gerado com fator de correção 0.4

A imagem da Figura 70.b apresenta o resultado aplicação do fator gama de 0.5 sobre a imagem Figura 70.a.

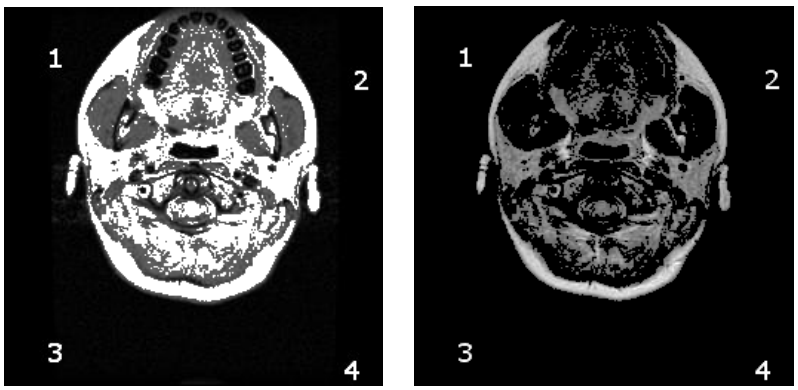


a **b**

Figura 70: Exemplo de aplicação do fator de correção 0.5

4.10.12.5 Branquemento e Preteamento da Imagem

As técnicas de branqueamento e preteamento usam um valor de referência que é comparado com o valor de nível de cinza de cada pixel da imagem. No branqueameamento, se o pixel comparado tiver o valor de cinza igual maior que o valor de comparação ele é convertido para branco (valor do nível de cinza igual a 255), os pixels que não satisfizerem está condição são mantidos inalterados. No preteamento se o valor do nível de cinza do pixel for menor ou igual que o valor de referência o pixel é convertido para o preto (valor do nível de cinza igual a 0), os pixels que não satisfizerem está condição são mantidos inalterados.



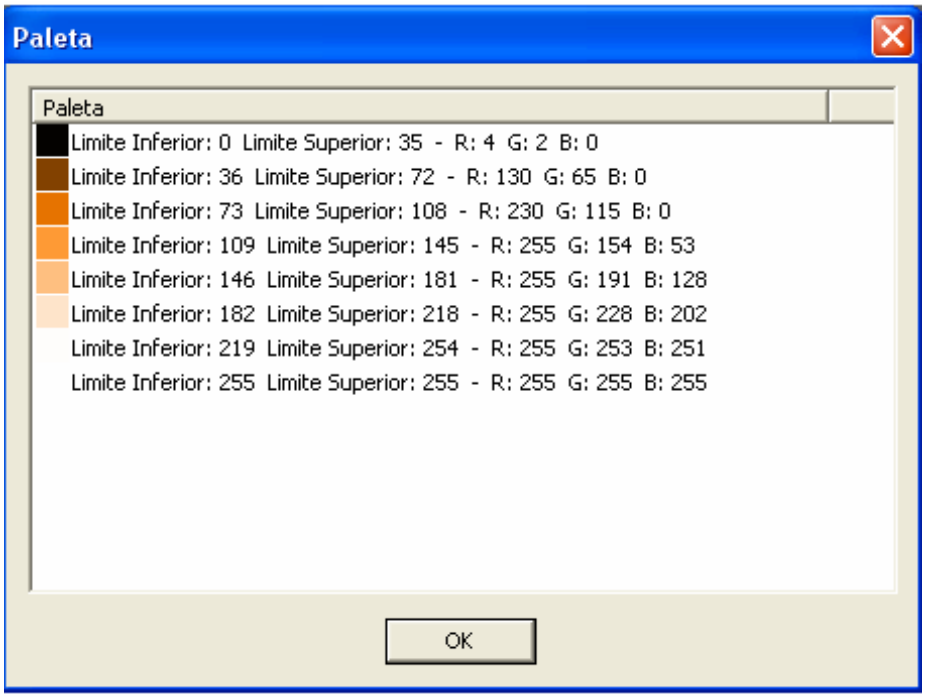
a **b**

Figura 71: Aplicação do branquamento e preteamento

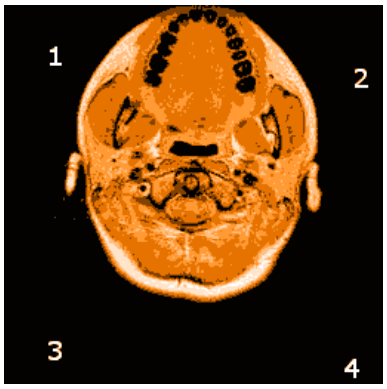
A Figura 71.a mostra o resultado da aplicação do branqueamento sobre a imagem da Figura 31 com valor de comparação igual a 100. A Figura 71.b mostra o resultado do preteamento sobre a imagem da Figura 31 com o mesmo valor de comparação.

4.10.12.6 Pseudo-cor

O VPIM possui uma ferramenta par adicionar pseudo-cores a uma imagem em níveis de cinza. Esta ferramenta permite criar uma paleta, editar a paleta, distribuir as cores manualmente ou automaticamente, gravar a paleta, recuperar uma paleta gravada, aplicar a paleta sobre a imagem e visualizar a paleta aplicada.



a



b

Figura 72: Aplicação de pseudo-cor

Na Figura 72.a pode se ver a relação das cores com os níveis de cinza da paleta de pseudo-cor, que foi aplicado sobre a imagem da Figura 31. O resultado da aplicação da paleta pode ser visto na Figura 72.b.

4.10.12.7 Conversão de uma imagem colorida para níveis de cinza

Para permitir o processamento de imagens coloridas pelas ferramentas do VPIM é possível transformar uma imagem colorida em uma imagem em níveis de cinza.

a

b

Figura 73: Transformação de uma imagem colorida em níveis de cinza

A Figura 73.b mostra a aplicação desta técnica sobre a imagem colorida da Figura 73.a.

4.10.12.8 Controle de Brilho e Saturação

O VPIM permite alterar em até 8 vezes o brilho de uma imagem em níveis de cinza e a saturação de cor de uma imagem colorida.

Figura 74: Alteração da saturação

A Figura 74 mostra o resultado da alteração da saturação da imagem da Figura 73.a. A saturação foi aumentada em 1.95 vezes.

Figura 75: Alteração do brilho

A Figura 75 mostra a alteração do brilho da imagem da Figura 31 em 1.95 vezes.

4.10.12.9 Visualização de Séries e Estudos

O padrão DICOM permite a trabalhar com estudos e séries além de imagens individuais. Uma série é um conjunto de imagens, normalmente obtidas simultaneamente. O estudo é composto de um conjunto de imagens e ou séries. Para poder comportar esta estrutura o VPIM tem ferramentas que permitem a visualização de séries/estudos. Através desta ferramenta é possível ver uma a uma as imagens de uma série manualmente ou automaticamente. Na opção automática se controla o tempo para

mudança das imagens. Outra possibilidade é ver todas as imagens numa mesma tela. Esta opção está mostrada na Figura 76.

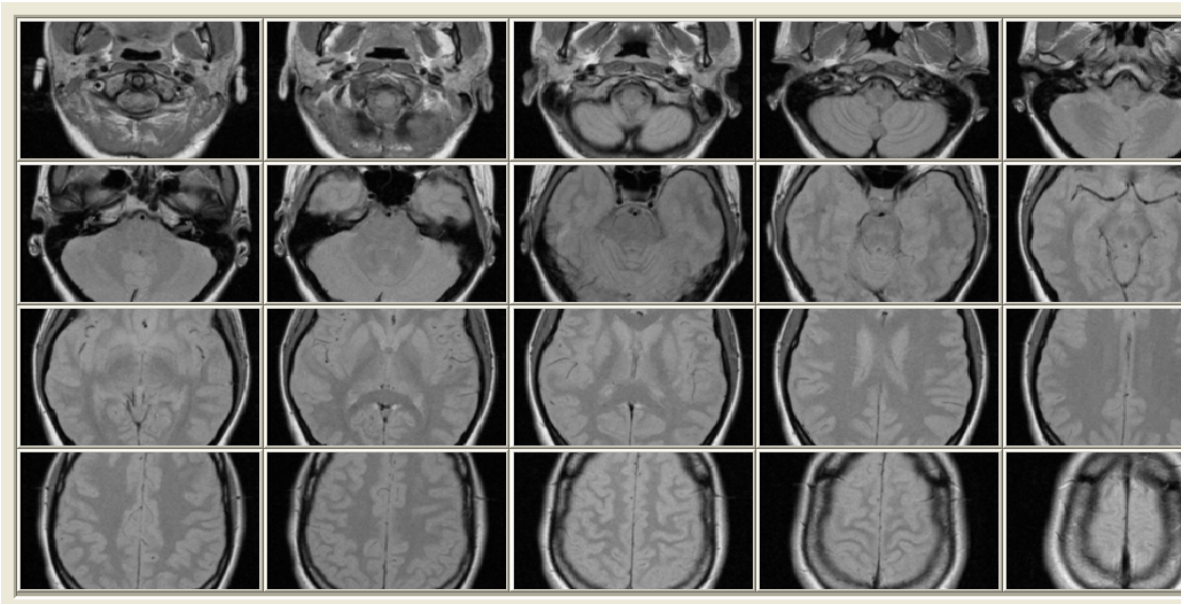
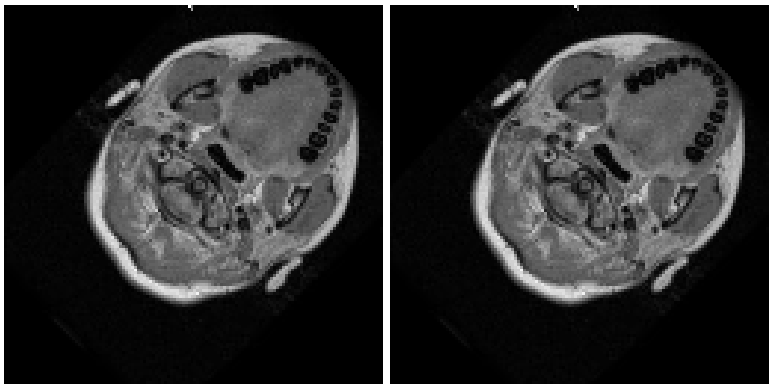


Figura 76: Exemplo de exibição de imagens de uma série

A Figura 76 é uma tela do VPIM exibindo todas as imagens dos arquivos brain_001.dcm a brain_020.dcm [15].

4.10.12.10 Equalização do Campo de Visão (FOV)

A equalização do campo de visão permite que as transformações aplicadas sobre uma imagem sejam aplicadas automaticamente sobre uma outra imagem para facilitar a comparação entre as duas imagens.



a **b**

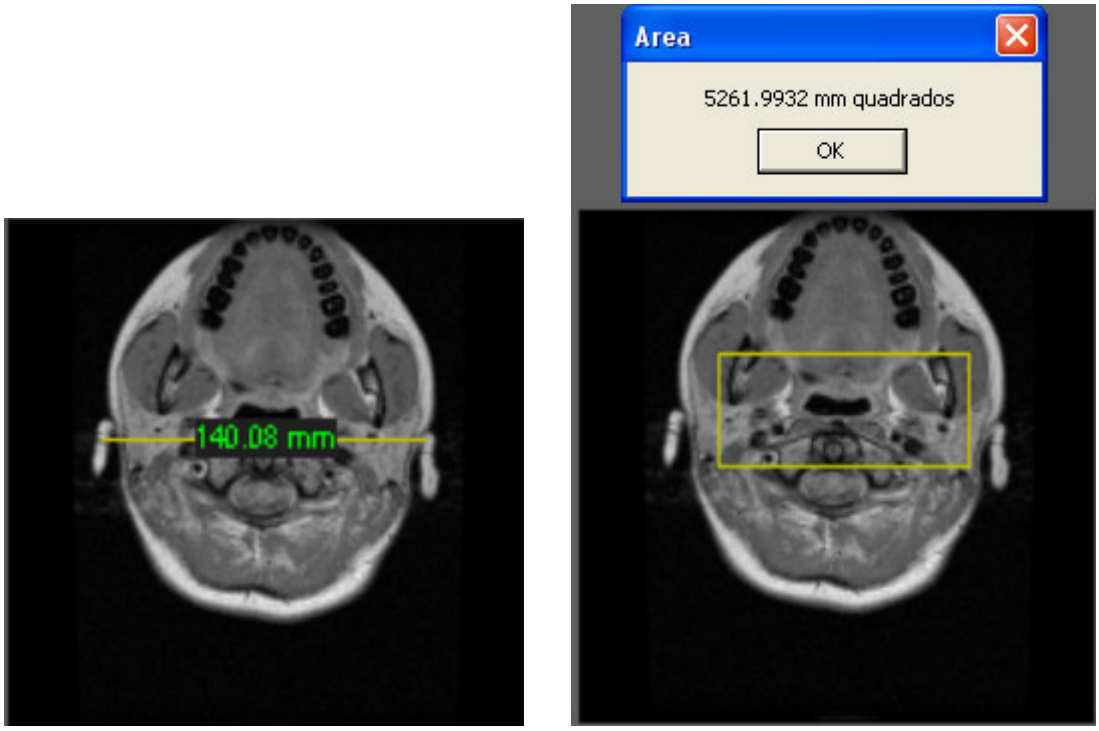
Figura 77: Aplicação da Equalização do Campo de Visão

Para obter as Figura 77.a e Figura 77.b foi inicialmente obtida a imagem do brain_001dcm [15] e foi feito uma cópia desta imagem no VPIM. Após, a imagem original foi reduzida em 2x e sofreu uma rotação de +45°. Para obter a imagem da Figura 77.b se aplicou a equalização do campo de visão sobre a imagem copiada tendo como base as transformações efetuadas na imagem original. Note que ambas as figuras ficaram idênticas.

4.10.12.11 Medidas

As medidas disponíveis no VPIM são: distância linear, área, histograma, dados estatísticos (valor máximo, valor mínimo, valor médio, desvio padrão dos valores em níveis de cinza dos pixels, dimensões da imagem em pixel), linhas de perfil e medida do valor do nível de cinza de um determinado pixel.

Na Figura 78.a e Figura 78.b são mostradas as medidas de distância linear e área. Para obter estes resultados foi usada a imagem do arquivo brain_001.dcm [15]. Por se tratar de um arquivo padrão DICOM e ter válida as informação de distância horizontal e vertical, entre cada pixel , foi possível obter a distância em mm e a área em mm^2 respectivamente.



a **b**

Figura 78: Medidas de distância e área

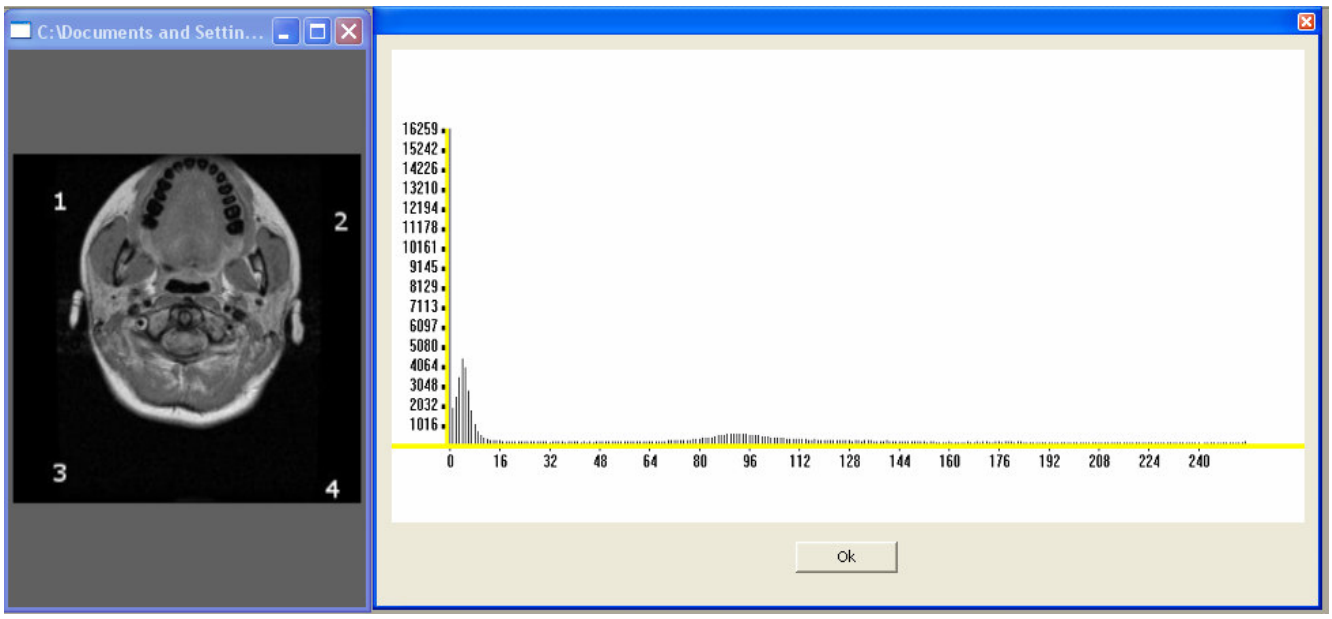


Figura 79: Histograma

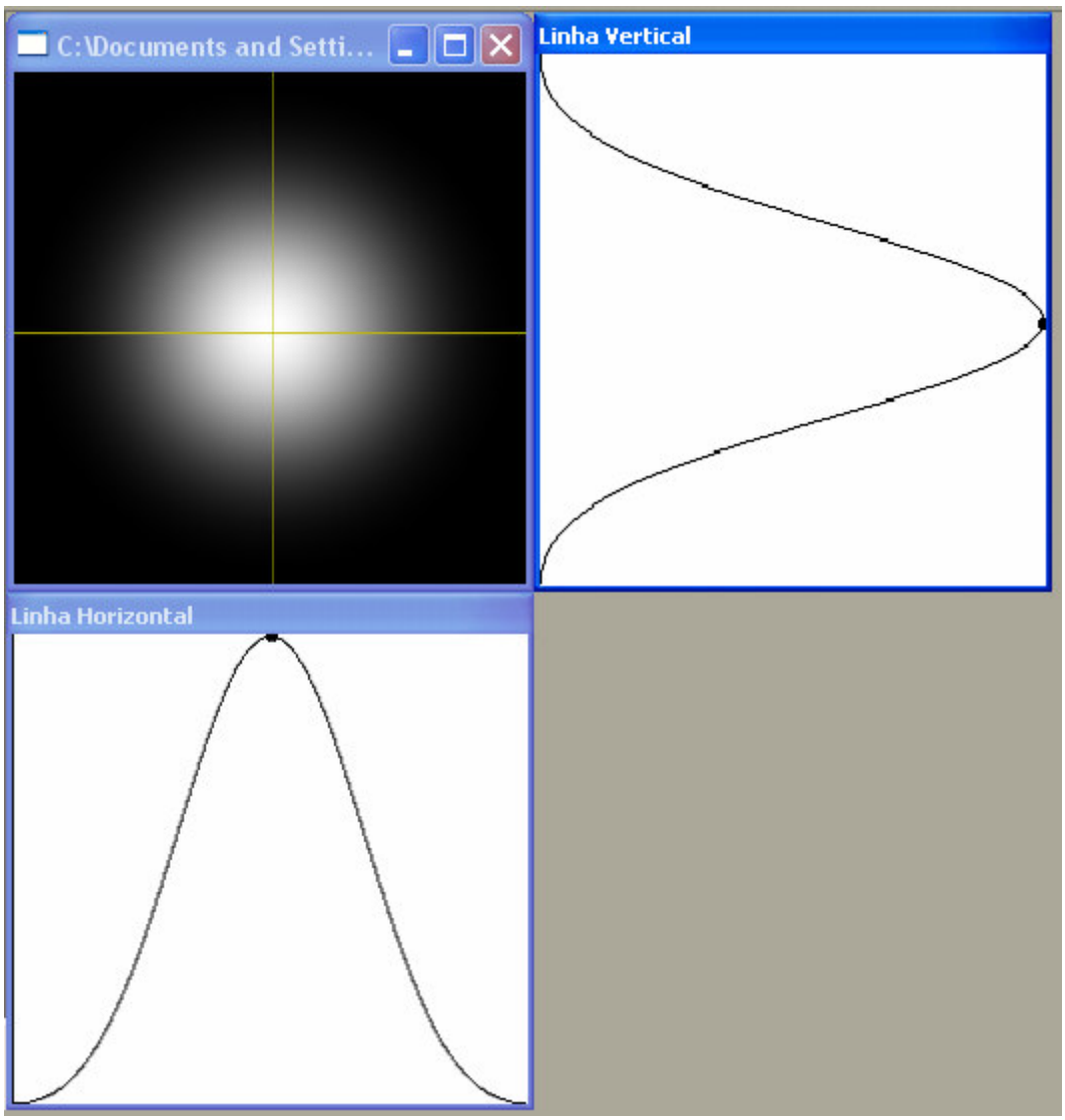


Figura 80: Linhas de perfil

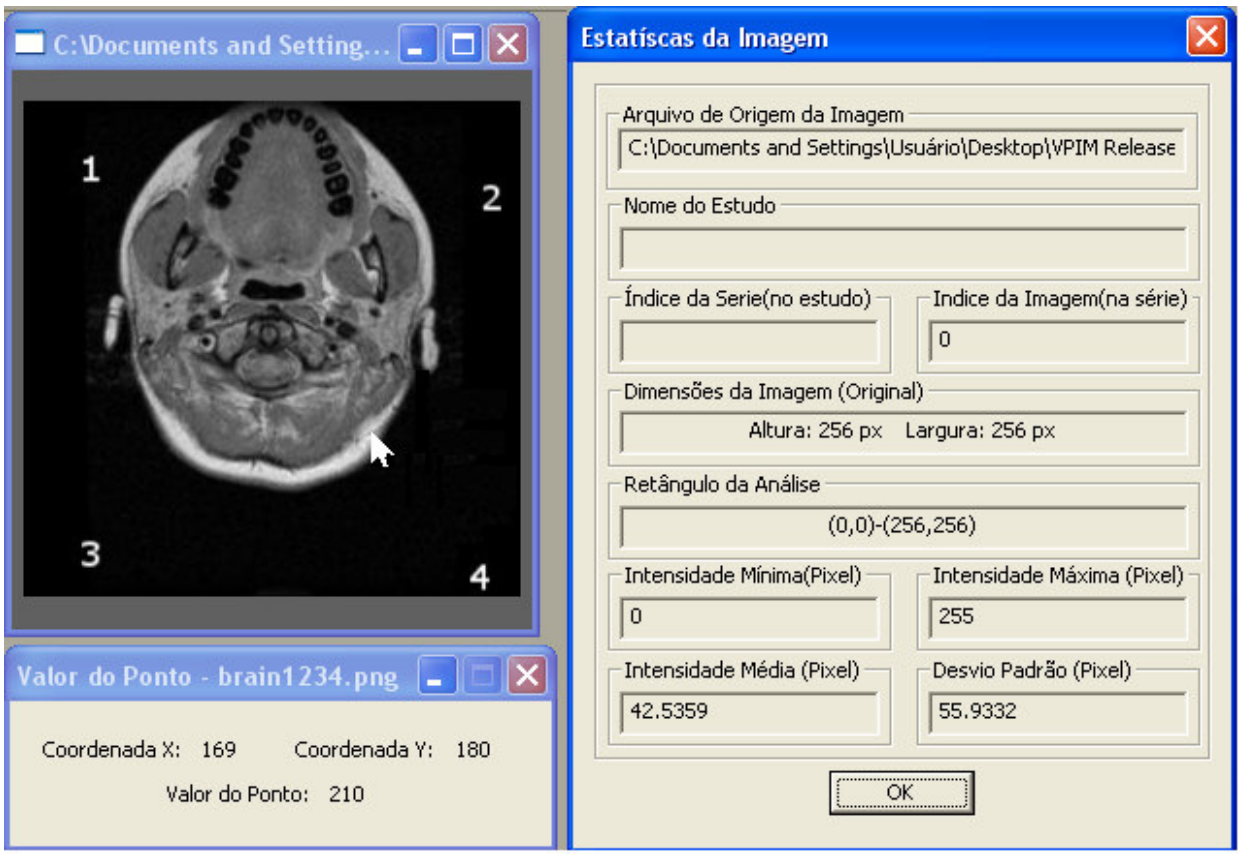


Figura 81: Valor do ponto e estatísticas da imagem

A Figura 79 mostra o histograma da imagem da Figura 31. Na Figura 80 é mostrado as linhas de perfil da imagem resultante da aplicação da máscara $\cos^4$. Na Figura 81 é mostrado o valor em nível de cinza de um ponto determinado (coordenadas 169,180) da imagem da Figura 31. Ao lado da janela do valor do ponto é mostrada a janela com os dados estatísticos da imagem da Figura 31. Estas duas janelas de medidas, mostradas na Figura 81, não aparecem juntas no VPIM.

4.10.12.12 Informações diversas

O padrão DICOM, além da imagem tem outras informações, como por exemplo: o nome do paciente, registro do paciente, histórico, equipamento utilizado, médico solicitante do exame, intervencionista e outros. As informações disponíveis no arquivo brain_0001.dcm estão mostrados na Figura 82.

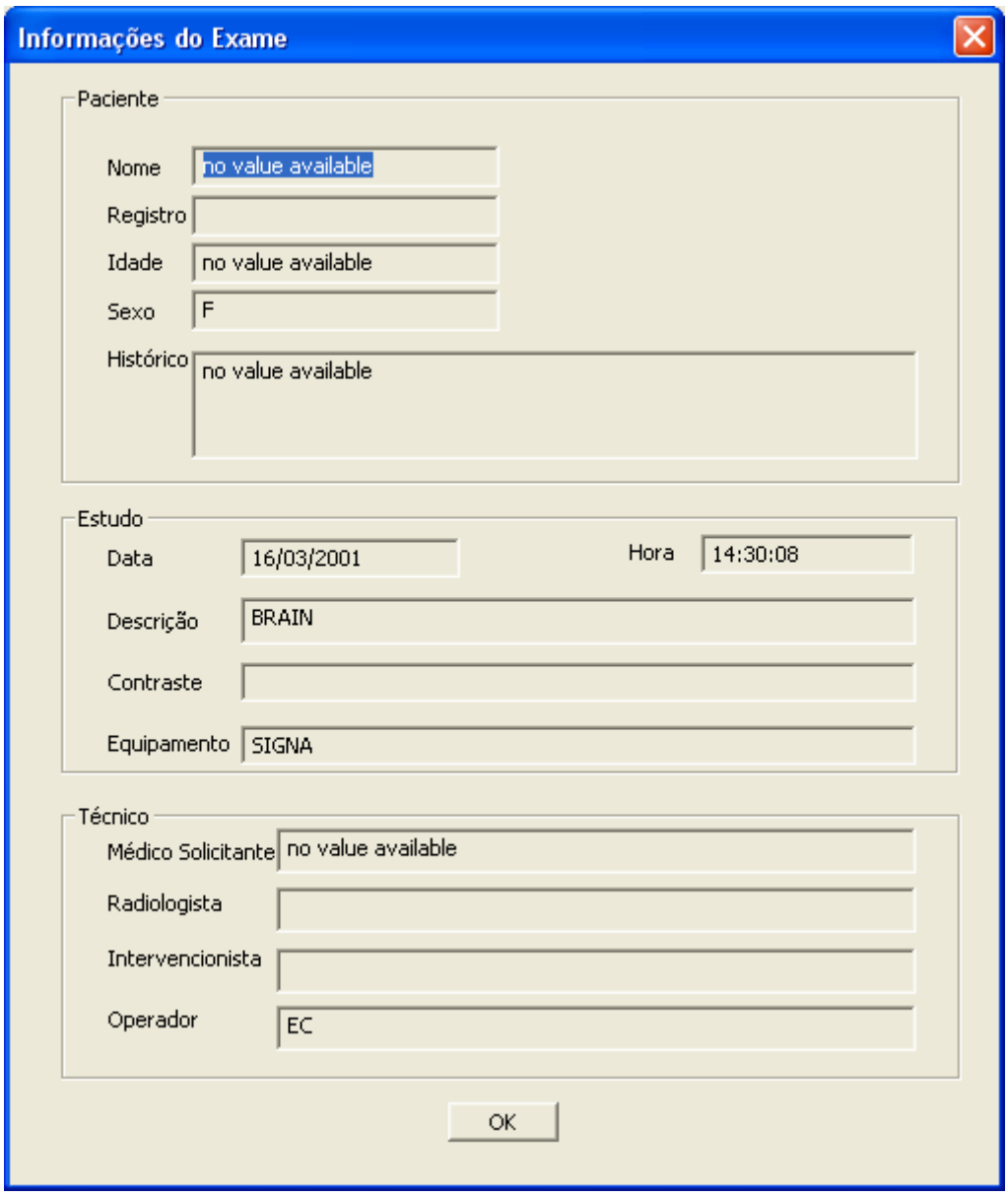


Figura 82: Exemplo de informações disponíveis no padrão DICOM

5 CONCLUSÃO

5.1 Objetivo

Os objetivos desta dissertação foram alcançados:

- Projeto e desenvolvimento de uma ferramenta para ser usada na visualização de imagens médicas, denominada Visualizador e Processador de Imagens Médicas (VPIM).
- Projeto e implementação de funções hoje encontradas nos programas disponíveis para uso médico.
- Implementação de funções para pesquisa de imagem na área médica (segmentação de imagens, realce de imagens, análise espectral da imagem): desenvolvimento e implementação dos algoritmos usados para estas funções.
- Desenvolvimento de uma arquitetura que permite a instalação de *plug-ins* visando possibilitar ao VPIM servir de plataforma para implementação de futuras funcionalidades.

A Figura 64.a [15] mostra um exemplo de visualização, gerado por um equipamento de ressonância magnética. A imagem da Figura 64.c é um exemplo de processamento de imagem do VPIM, ela foi obtida mediante a aplicação de técnica de *threshold* global, com uma conversão para o preto dos pixels com valor do nível de cinza maior ou igual a 118.

O VPIM opera com arquivos no formato DICOM, fornecendo todas as informações disponíveis nos mesmos, e podendo, inclusive, trabalhar imagens, imagens agrupadas em séries, estudos contendo imagens ou séries, fazendo dele uma ferramenta, neste aspecto, equivalente às existentes no mercado.

A Figura 34, a Figura 56 e a Figura 60 mostram o resultado da aplicação de algumas funções sobre a imagem da Figura 31 presentes nos visualizadores e também disponíveis no VPIM. Maiores detalhes sobre a obtenção destes resultados podem ser obtidos na Seção 4.10. A lista todas as funções do VPIM podem ser vista na Tabela 2.

Além das funções normalmente encontradas nos visualizadores usados para análise de imagens médicas, foram incluídas funções voltadas para pesquisa que executam o processamento da imagem, tais como: espectro de potência, filtro de

convolução no domínio freqüência (filtros Butterworth), geração de máscaras para janelamento (Blackman e $\cos^4$), linhas de perfil, e outras.

A Figura 42.c, a Figura 46, e a Figura 80 mostram algumas imagens, obtidas pelo VPIM , que são usadas pelos pesquisadores. Maiores detalhes sobre a obtenção das imagens das figuras relacionadas neste parágrafo podem ser encontradas na Seção 4.10.

A expansibilidade do VPIM foi feita através de uso de *plug-ins* (DLLs). Muitos dos filtros existentes (Sobel, média móvel, mediana e outros) foram gerados desta forma. Assim, usando esta técnica, futuras funcionalidades poderão facilmente ser adicionadas ao VPIM.

O VPIM tem espera para um dispositivo de calibração automático, que está sendo desenvolvido pelo IPCT, que permitirá futuramente que o programa seja usado para diagnóstico.

5.2 Validação e Calibração

As rotinas pertinentes foram validadas contra rotinas executando as mesmas funções implementadas no MTLAB.

As medidas foram calibradas por comparação com medidas obtidas no software de uso médico eFilm [20].

5.3 Lista das Funções do VPIM

A Tabela 2 apresenta uma lista das funções implementadas e disponíveis no VPIM.

Menu	Função	
Arquivo	Abrir	
	Salvar como	
	Fechar Estudos	
	Sair	
Visualização	Agrupamento	
	Equalização do campo de visão (FOV)	
	Listar informações	
	Escala de Cinza	
	Inserir Texto na Imagem	
	Ligar/Desligar Visualização de texto	
	Bordas das janelas	
	Imagem com tamanho normal/ou da janela	
	Palheta de cores	Adicionar
		Remover
		Editar
		Redistribuir
		Gravar
		Carregar
	Editar Palheta de cores	
	Associar palheta de cores a imagem	
Navegação	Próximo Estudo	
	Estudo Anterior	
	Próxima Série	
	Série Anterior	
	Próxima Imagem	
	Imagem Anterior	
	Controle de Exibição de imagens	Velocidade de exibição
		Tocar
		Parar
		Próxima imagem
		Imagem anterior
		Ordem reversa
		Ordem direta

Menu	Função		
Processamento	Espelhamento	Horizontal	
		Vertical	
	Rotação	De 90º	Horária
			Anti-horária
		Livre	
	Negativo		
	Magnificação & Redução		
	Transformadas	Espectro de potência com remapeamento	
		Espectro de potência sem remapeamento	
	Janelamento	Cos^4	
		Blackman	
	Filtros	Butterworth	Passa-baixa
			Passa-alta
			Passa-faixa
			Elimina-faixa
		Filtro da Mediana	
		Filtro de Alto Reforço	
		Filtro Passa-alta	
		Filtro da Média Móvel	
		Detecção de linhas	+45º
			-45º
			Horizontais
			Verticais
		Matriz de convolução 5x5	
		Sobel	
	Resolução	Reamostrar	
		Estender	
	Recorte	Pelo Mouse	
		Pelo Teclado	
	Remapeamento Linear	Local	Pelo mouse
			Pelo teclado
		Global	
	Combinação	Pixel alternado	
		Subtração de imagem	
		Combinação binária	
		Clonar imagem	
Outros	*Threshold* Global		
	Remoção de Fundo		
	Equalização de histograma		
	Correção Gama		
	Brilho & Saturação		
	Fundo	Branqueamento	
		Preteamento	
Medidas	Distância linear		
	Área		
	Histograma		
	Linhas de perfil		
	Valor do ponto		
	Estatísticas		
	Limpar Medidas		
	Queimar medidas na imagem		

Menu	Função
Ajustes	Teste de Calibração (espera)
Ajuda	Sobre
	Conteúdo (espera)

Tabela 2: Funções do VPIM

5.4 Próximos Passos

O trabalho realizado no VPIM poderia ser estendido com os seguintes passos:

1. Alterar o VPIM, que atualmente mostra e processa imagens em 256 níveis de cinza (8 bits), para que passe a mostrar e processar imagens com até 65536 níveis de cinza (16 bits).
2. Realização de um estudo para melhorar e padronizar a interface com usuário.
3. Teste médico extensivo das funções VPIM para validação médica.
4. O VPIM foi extensivamente testado pelo implementador, teste conhecido como *unit test*. Deve ainda ser aplicado um teste por testadores que não participaram da implementação para evitar os vícios resultantes da implementação.
5. Adição do *driver* para o calibrador de monitores de vídeo, já em desenvolvimento pelo IPCT. Os usos do calibrador, junto com os testes extensivos do VPIM, permitirão o seu uso para diagnósticos médicos.
6. Seria interessante que na sua próxima versão o VPIM incorporasse as seguintes funções:
    - Funções lógicas sobre a imagem
    - Reconhecimento de padrões baseado em convolução e redes neurais.
    - Funções voltadas segmentação, por exemplo, o crescimento e a erosão de regiões.
    - Funções para efetuar o registro de imagens (translação, rotação, correlação cruzada e magnificação).

6 REFERÊNCIAS BIBLIOGRÁFICAS

[1] Angel, Eduard. Interactive Computer Graphics: A top-down approach with OpenGL, second edition. Addison Wesley Longman, Inc., January 2000.

[2] Brown, Nancy. Telemedicine coming of Age. January 2005
Disponível em:
http://tie.telemed.org/articles/article.asp?path=articles&article=tmcoming_nb_tie96.xml#dperedniarefT .
Acessado em: 31/07/2005

[3] Conselho Federal de Medicina, Resolução CFM nº 1.6432/2002, Art 1º.
Obtido em: www.portalmédico.org.br/resolucoes/dfm/1992/16432002.htm
Acessado em: 13/04/2004.

[4] DICOM@OFFIS, DICOM Tool kit DCMTK 3.5.3 source code and documentation. May 27th, 2005
Obtido em: http://dicom.offis.de/dcmtk.php.en.
Acessado em: 31/07/2005.

[5] FFT & DCT Library.
Obtido em: http://www.codeproject.com/library/ArisFFTDFTLibrary.asp
Acessado em: 3/12/2006

[6] Gonzalez, Rafael C., Woods, Richard E.. Processamento de Imagens. Editora Edgard Blücher Ltda., 2000.

[7] Hajnal, Joseph V., Hill, Derek C. G.., Hawkes, David J.. Medical Image Registration. CRC Press, 2001.

[8] Hutson, Geoffrey H., Teoria da Televisão a Cores. Editora McGraw-Hill do Brasil Ltda., 1974

[9] Ingle Vinary K., Proakis John G.. Digital Signal Processing using MATLAB, Br ooks/Cole Publishing Company, 2000

[10] Jähne, Bernard. Image Processing for Scientific Applications. CRC Press LCC, 1997

[11] Jain, Anil K.. Fundamental of Digital Image Processing. Prentice-Hall, Inc., 1989

[12] Krug, Wolfgang; Rorden, Chris. Programa ezDICOM [version 1.0, rev 12, release 19]. July 27th, 2001

Obtido em:

http://www.psychology.nottingham.ac.uk/staff/cr1/ezdicom.html#users

Acessado em: 17/04/2005

[13] LibTIFF – TIFF Library and Utilities,

Obtido em: http://www.remotesensing.org/libtiff/

Acessado em: 29/12/2005

[14] Lindquist, Claude S.. Adaptative & Digital Signal Processing with Digital Filtering Applications. Steward & Sons, 1989

[15] MATLAB CENTRAL,

Obtido em:

http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=2762&ObjectType=File

Acessado em: 13/12/2005

[16] Microsof DirectX Developer center.

Obtido em: http://msdn.microsoft.com/directx/

Acessado em: 07/08/2005

[17] National Electrical Manufacturers Association (NEMA). DICOM Standard, Obtido em: http://medical.nema.org/dicom/2004.html. Acessado em: 13/07/2005.

[18] Porat, Boas. A Course in Digital Signal Processing. John Wiley & Sons, 1997

[19] Poynton, Charles. A Technical Introduction to Digital Video. John Wiley & Sons, 1996

[20] Programa eFilm Workstation 1.5.3; Version 1.5.3; Built: May 9, 2001 15:40:11.

[21] Programa MATLAB, Version 6.0.0.88 Release 12, September 22, 2000

[22] Yu-Li You; Kaveh, M. Pyramidal image compression using anisotropic and error-corrected interpolation. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-96). 1996. Volume 4, 7-10 May 1996 Page(s):1946 - 1949 vol. 4

[23] Zlib home page Obtido em: http://www.zlib.net Acessado em: 20/12/2005