Antonio Escaño Scuri

Filtros Interativos para Imagens Digitais no Domínio da Freqüência

Dissertação de Mestrado

Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 14 de Setembro de 1994.

Antonio Escaño Scuri

Filtros Interativos para Imagens Digitais no Domínio da Freqüência

Dissertação apresentada ao Departamento de Informática da PUC-Rio como parte dos requisitos para obtenção do título de Mestre em Informática: Computação Gráfica.

Orientador: Bruno Feijó - INF / PUC-Rio
Co-orientador: Sidnei Paciornik - D.C.M.M. / PUC-Rio

Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 14 de Setembro de 1994.

a meu pai.

Meus agradecimentos,

- a Sidnei Paciornik pela amizade, paciência e ensinamentos.
- a Marcelo Gattass pelos inestimáveis conselhos e “provocações”.
- A Bruno Feijó pela cooperação com o D.C.M.M..
- a todo o Grupo Integrado de Materiais (GIM).
- a todos os colegas do ICAD e do TECGRAF que ajudaram direta ou indiretamente.
- a Ulrich Dahmen e Roar Kilaas, pela oportunidade e recepção acolhedora no Lawrence Berkeley Laboratory, como também a Laurent Normand pela grande idéia da visita.
- a Noemi Rodriguez e Roberto Ierusalimschy pela atenção.
- Jonas de Miranda pelos ensinamentos.
- a Álvaro de Miranda Filho e Ivan Melo de Carvalho pelas oportunidades indiretas de desenvolvimento do meu trabalho.
- ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pela ajuda financeira recebida durante o curso.
- a meus pais pela paciência.
- aos amigos Maurício, Carla, George, Elisa e Carim, por algo muito precioso nesse nosso pequeno mundo.
- a Laura pela maravilhosa companheira que é.

RESUMO

A grande difusão de ambientes interativos, assim como o desenvolvimento de poderosas interfaces gráficas, proporcionam a utilização de novas soluções para antigos problemas. Na área de Processamento de Imagens Digitais, isto é de grande utilidade, como na técnica de filtragem espacial no domínio da freqüência. Com esta técnica, a Transformada Rápida de Fourier (FFT) de uma imagem é obtida e então editada para manter ou rejeitar regiões específicas, em seguida a Transformada Rápida Inversa de Fourier (IFFT) da FFT editada fornece a imagem filtrada resultante. Este trabalho estuda a utilização de elementos de interface para criar máscaras gráficas que funcionem como filtros, e propõe a combinação destes filtros de forma a permitir extrema flexibilidade na técnica de filtragem. A criação dos filtros e a implementação dos algoritmos de FFT foram feitas em um protótipo que utiliza um novo sistema de processamento de imagens, **SPID**, que está sendo estruturado como uma plataforma de desenvolvimento para Processamento de Imagens no ambiente gráfico **Microsoft® Windows™**.

ABSTRACT

The widespread use of interactive environments as well as the development of powerful Graphical User Interfaces have made it possible to reach new solutions to old problems. In the field of Digital Image Processing this can be very useful, as for the technique of spatial filtering in the frequency domain. With this technique the Fast Fourier Transform (FFT) of an image is obtained and then edited to preserve or reject specific regions, and then the Inverse Fast Fourier Transform (IFFT) of the edited FFT provides the resulting filtered image. The present work studies the use of interface elements to create graphical masks that function as filters, and proposes the combination of these filters so as to provide great flexibility to the filtering technique. The creation of the filters and the implementation of the FFT algorithms were done in a prototipe of a new image processing system, **SPID**, which is under development as a platform for Image Processing applications under the **Microsoft® Windows™** graphical enviroment.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

1. INTRODUÇÃO

A área de processamento de imagens (PI) é um excelente exemplo de interdisciplinaridade. As diversas técnicas originalmente desenvolvidas para tratamento de sinais unidimensionais foram, em primeiro lugar, adaptadas para tratamento de imagens obtidas de satélites e de naves espaciais. Posteriormente, com o rápido avanço das opções de *hardware* e *software*, estas mesmas técnicas passaram a ser aplicadas em inúmeros domínios tais como medicina, ciência dos materiais, microscopia, artes, etc...

A característica interdisciplinar fica ainda mais realçada no desenvolvimento de aplicativos específicos. Neste caso os recursos da Engenharia de Computação se aliam a problemas e especificações de outras áreas para gerar soluções difíceis de serem alcançadas de forma independente.

O campo de Ciência dos Materiais e Microscopia (ótica e eletrônica) é um caso protótipo deste tipo de interação. O presente trabalho se propõe a utilizar recursos modernos de computação, tais como orientação por objetos e conceitos de portabilidade, assim como técnicas de tratamento de sinais para tratar problemas da área de materiais, mais especificamente em microscopia eletrônica de transmissão (MET). (Embora sua aplicação imediata seja em MET, o enfoque de desenvolvimento será genérico possibilitando outras áreas tomarem partido do mesmo.)

Neste tipo de aplicação imagens de alta resolução são obtidas de diferentes tipos de amostras visando elucidar sua estrutura atômica. Este tipo de estudo é vital para a previsão das propriedades estruturais de novos materiais. Estas imagens em geral se compõem de diferentes estruturas geométricas superpostas, correspondendo às diferentes estruturas cristalinas ou não cristalinas que formam a amostra em estudo. A distinção destas diversas estruturas é vital para a compreensão das propriedades do material e técnicas de PI são extensivamente utilizadas para isso.

Uma das técnicas mais poderosas para análise de estrutura atômicas é a filtragem espacial da Transformada de Fourier (FT) das imagens. Neste caso, a imagem original é transformada através de um algoritmo de Transformada Rápida de Fourier (FFT), máscaras são aplicadas sobre a imagem da transformada, selecionando regiões que serão mantidas ou eliminadas e, finalmente, a imagem filtrada é obtida através da FFT inversa (IFFT).

A utilização da transformada é justificada pela periodicidade bem determinada destas estruturas atômicas. No domínio da freqüência esta periodicidade aparece isolada, podendo-se realizar uma distinção das estruturas a partir da região de freqüência de cada uma.

As máscaras aplicadas sobre a imagem da transformada devem portanto refletir características da imagem no domínio da freqüência, para que então se possa isolá-las.

O cerne deste trabalho é a especificação destas máscaras. Foram identificadas diversas características de imagens no domínio da freqüência que definiram o formato de várias máscaras.

A maneira mais fácil de aplicá-las sobre a imagem é fazê-lo diretamente sobre as imagens. Para tanto necessitamos de um ambiente gráfico interativo para o nosso protótipo. Escolheu-se o **Microsoft® Windows™** pela sua grande aceitação no mercado e pela disponibilidade de ferramentas para desenvolvimento neste ambiente.

Como suporte para processamento de imagens utilizou-se o sistema **SPID**, que mesmo em desenvolvimento, permitiu a inclusão das máscaras no seu escopo.

Convém lembrar que algumas máscaras são baseadas em máscaras semelhantes implementadas no **Digital Micrograph**, um programa para **Macintosh** dedicado à aplicações em microscopia eletrônica.

A utilização de Processamento de Imagens digitais na PUC-Rio vem ocorrendo há alguns anos. Embora adotada por diversos departamentos, tomou um caráter mais genérico no Departamento de Ciência dos Materiais e Metalurgia, mais especificamente no Grupo Integrado de Materiais (GIM). O percurso de PI pelo GIM é descrito no capítulo **1.** Erro! Auto-referência de indicador não válida..

A transformada de Fourier, a transformada rápida e a metodologia de filtragem no domínio da freqüência são descritas no capítulo **2**. Histórico. Na seção **7 Os Filtros** são deduzidas as máscara a partir dos fenômenos respectivos.

No capítulo **3.** Apresentação Teórica é apresentado o **SPID** como uma plataforma para desenvolvimento de aplicativos para processamento de imagens, como também são descritos detalhes sobre o funcionamento das máscaras.

É muito difícil prever que tipo de máscara será usado para uma determinada imagem. A melhor maneira é calcular a FFT e tentar identificar na imagem alguma característica conhecida à qual se adeque um certo tipo de máscara. Assim, o capítulo **8.** Implementação mostra todo o

processo de filtragem para algumas imagens bastante didáticas, com o objetivo de demonstrar o potencial de várias máscaras.

Questões de performance e qualidade dos resultados são discutidos no capítulo **16.** Exemplos. Neste capítulo também são feitas considerações a respeito de melhorias imediatas para o SPID e do seu futuro como base para desenvolvimento em PI.

2. HISTÓRICO

Em meio a um projeto na área de Litografia por Feixe de Elétrons e a partir da aquisição de um digitalizador de sinal de vídeo em **1988**, iniciou-se paralelamente um trabalho na área de processamento de imagens que se chamaria **Imago**, [GALUCIO 90].

O *hardware* e o sistema operacional de desenvolvimento trouxeram diversas limitações ao **Imago**. Embora possuísse uma simulação de memória virtual, o programa era limitado aos 640Kb fornecidos aos programas pelo sistema operacional, no caso o MS-DOS®.

A imagem fornecida pelo digitalizador de sinal de vídeo possuia apenas 64 tons de cinza. Isto direcionou a implementação do sistema para apenas este tipo de imagem.

A visualização das imagens também era limitada a placas de vídeo de baixa resolução ou a duas marcas de placa de vídeo de alta resolução.

Estas limitações se tornaram mais evidentes com a necessidade de manipular imagens maiores em novas placas de vídeo disponíveis no mercado.

Em janeiro de **1991**, iniciou-se um projeto de um sistema de processamento de imagens, onde suas rotinas básicas tivessem características que facilitassem principalmente sua **portabilidade** e **expansão**. Para isso, optou-se pela utilização de técnicas de **orientação a objetos**, associadas a uma linguagem portável e voltada para software básico. A escolha de **C++** como linguagem de desenvolvimento foi considerada como a melhor opção naquele momento.

No final deste mesmo ano surge o primeiro protótipo destas idéias: o programa **PINTA** [SCURI 91]. Este programa tem como função a leitura de arquivo e visualização de imagens digitais monocromáticas em um monitor, acessado através de uma placa gráfica **Super VGA**, com resolução até 800 x 600 e 64 tons de cinza. Nesta etapa houve um grande aprendizado e foi dado o pontapé inicial para atingir as melhorias desejadas.

Também no final deste mesmo ano começava a despontar no mercado um ambiente gráfico lançado pela **Microsoft Corporation** em maio de **1990**, o **Windows 3.0**. Até então ferramentas para desenvolvimento neste ambiente eram escassas. Em **1992**, tudo acontece. Surge a nova versão do ambiente, o **Windows 3.1**, e chegam à nossas mãos ferramentas poderosas de desenvolvimento para este ambiente. Isso possibilitaria uma melhor utilização do hardware que já estava à disposição, sem custos adicionais.

Como as funções desenvolvidas para manipulação de imagens são portáveis, nada mais natural do que propor uma versão do programa para o ambiente **Microsoft Windows 3.1**. Surge o programa **PIXIE** no final de **1992** [ALEGRETTE 92]. Neste programa foram incluídas rotinas para leitura de imagens gravadas em arquivo no padrão **BMP** (do próprio **Windows**) e no padrão **TIFF** (um dos mais bem aceitos padrões de arquivo da área de processamento de imagens).

No ano de **1993**, tudo foi reestruturado e muito se evoluiu. O programa **PIXIE** amadureceu e ganhou um novo nome: **SPID** (Sistema de Processamento de Imagens Digitais) [SCURI 93].

No ambiente **DOS**, o programa **PINTA** ganhou uma versão em **modo protegido**, chamado **MOSTRA**. Este programa não tem a limitação da área de memória do **DOS**, chamada de barreira dos 640Kb, utilizando, portanto, toda memória disponível no micro. Um outro recurso acrescentado foi o acesso a qualquer tela **Super VGA** através do padrão **VESA** (Video Electronics Standard Association), o que permitiu o programa atingir resoluções maiores, com uma velocidade considerada bastante alta.

Em **1994**, o **SPID** ganha novos recursos gráficos com a chegada de uma nova ferramenta de desenvolvimento. E sofre uma migração para uma plataforma de 32 bits portável para **Windows 3.1 + Win32s**, **Windows NT** e futuramente o **Windows 4.0** (nome código Chicago).

Esta migração proporcionou um enorme ganho de velocidade e a eliminação de alguns dos problemas do Windows 3.1. A versão para **Windows NT** dá muita flexibilidade ao **SPID**, principalmente porque a Microsoft promete versões deste ambiente para outras máquinas além do tradicional PC (**Intel x86**), tais como **MIPS R4000** e **DEC Alpha**.

O **SPID** encontra-se em fase de desenvolvimento, onde a base do projeto está toda pronta e estão sendo acrescentadas funções de processamento de imagens.

Todo esse crescimento veio de experiências com diversos sistemas de processamento de imagens. Principalmente, com o **Aldus PhotoStyler**, o **Adobe PhotoShop** e o **Corel PhotoPaint**, no ambiente **Microsoft Windows** e com o **Digital Micrograph**, no ambiente **Macintosh**. Os programas para **Windows** mencionados são voltados para a área de *desktop publishing*, embora ricos neste sentido, não possuem suporte para imagens científicas, nem funções de reconhecimento de padrões. Já o **Digital Micrograph** é justamente voltado para a área científica, mais especificamente a microscopia eletrônica.

Procuramos também informações em sistemas mais antigos, aos quais não temos acesso, mas algumas referências, ver [SCURI 92].

Alguns programas mais específicos na área científica também foram consultados, tais como **CRISP** e **Global Lab Image**, ambos para **Windows** e o **Image** para **Macintosh**.

3. APRESENTAÇÃO TEÓRICA

4. Transformada de Fourier

Joseph Fourier deixou para a matemática um rico legado: a transformada que leva o seu nome é utilizada em diversas áreas da ciência e tecnologia hoje em dia. Sua aplicação em PI tem relevada importância, desde a simples filtragem até a utilização na compressão de arquivos de imagem.

Esta transformada é uma das maneiras de realizar uma mudança de domínio espacial em espectral (onde um sinal fica caracterizado por suas componentes de freqüência).

Para imagens digitais (uma função bidimensional discreta), utiliza-se a Transformada de Fourier Discreta (DFT - Discrete Fourier Transform), que é definida como:

$$F(u,v) = \frac{1}{\sqrt{M} \cdot \sqrt{N}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \cdot e^{\frac{-2pi}{N}(xu+yv)} \qquad \text{Equação 3-1}$$

Por sua vez a transformada inversa:

$$f(x,y) = \frac{1}{\sqrt{M} \cdot \sqrt{N}} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) \cdot e^{\frac{+2pi}{N}(xu+yv)} \qquad \text{Equação 3-2}$$

Estas duas equações representam um par de transformadas. Pode-se obter outros pares trocando o sinal das exponenciais ou então realizando a multiplicação pela constante apenas na transformada inversa. Inclusive esta última sugestão é recomendada para a implementação do algoritmo pois reduz o número de operações a serem realizadas.

O termo exponencial nas equações é visivelmente separável. Fazendo com que o somatório em x possa conter todos os termos em *x*, assim a equação 3-1 fica:

$$F(u,v) = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} F_x(u,y) \cdot e^{\frac{-2pi}{N}(yv)} \qquad \text{Equação 3-3}$$

onde,

$$F_x(u,y) = \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} f(x,y) \cdot e^{\frac{-2pi}{N}(xu)} \qquad \text{Equação 3-4}$$

ou seja, chegamos a duas trasformadas unidimensionais, uma para as colunas e outra para as linhas (a inversa tem demonstração equivalente). Esta propriedade permite com que a DFT seja calculada em duas etapas: uma para as colunas e outra para as linhas obtidas com a etapa anterior.

Os somatórios nas equações vistas são aproximações de integrais, que por sua vez fornecem uma medida da "densidade" da freqüência *u* ou *v* ao longo de todo o domínio de f, porém não nos dá informação sobre a localização dessa freqüência no domínio da função.

Existem outras transformadas que nos levam a outros modelos espectrais, tais como a Transformada Cosseno e a Transformada de Wavelets.

A imagem complexa resultante da DFT possue a freqüência (0, 0) no *pixel* de coordenada (0, 0). Neste ponto a imagem atinge o seu maior valor, pois nele é computada justamente a menor freqüência presente na imagem e pode-se deduzir que este é o valor médio da imagem. Sabendo-se que a DFT é periódica com período N, este ponto irá se repetir em (N, N). Estamos observando o início e o fim de um período. Contudo para fins de análise é mais conveniente que observemos de -N/2 até +N/2, colocando a freqüência zero no centro da imagem.

A centralização da imagem da transformada é feita de maneira muito simples, já que:

$$f(x,y)\cdot e^{\frac{2pi}{N}(u_0x+v_0y)} \Leftrightarrow F(u-u_0,v-v_0) \qquad \text{Equação 3-5}$$

e quando $u_0 = v_0 = N/2$, ten-se $e^{pi(x+y)} = (-1)^{x+y}$, e

$$f(x,y)(-1)^{x+y} \Leftrightarrow F(u-\frac{N}{2},v-\frac{N}{2}) \qquad \text{Equação 3-6}$$

Portanto, durante o cálculo da primeira etapa podemos realizar esta multiplicação e automaticamente obter a imagem da transformada já centrada. O problema com esta multiplicação está no fato de que quando calcularmos a inversa obteremos a imagem original com as multiplicações efetuadas. Assim, é necessário que estas multiplicações sejam desfeitas no algoritmo da transformada inversa.

A mudança da origem se torna ainda mais interessante quando constatamos que a maior parte de nossas imagens a serem processadas pela DFT são imagens reais. A DFT neste caso exibe uma característica peculiar de simetria, onde:

$$F(u,v) = F^*(-u,-v) \qquad \text{Equação 3-7}$$

Quando visualizamos o módulo da transformada temos:

$$|F(u,v)|=|F(-u,-v)| \qquad \text{Equação 3-8}$$

ou seja, observamos uma imagem simétrica em relação à origem. As imagens complexas com esta característica são chamadas de hermitianas. Esta propriedade nos ajuda a reduzir o tempo de cálculo de transformada (basta calcular a metade), e define uma das características dos filtros deste trabalho, descritos adiante.

A DFT de uma imagem é essencialmente uma representação em série de Fourier de uma função bidimensional. Sendo assim, para ser válida esta função precisa ser periódica. Essa periodicidade é obtida considerando que a imagem se repete em um plano infinito como na Figura 3-1:

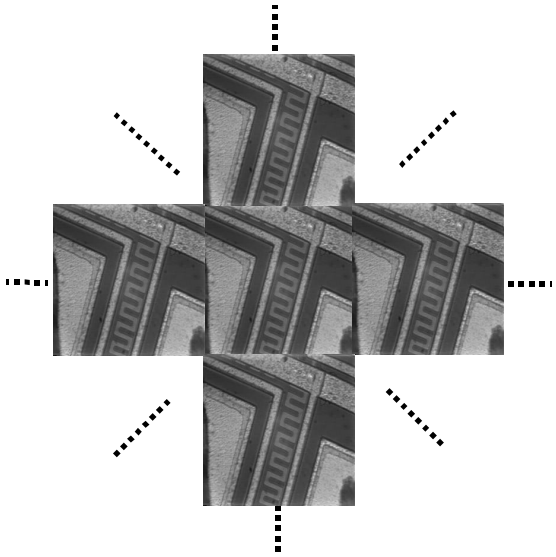

Figura 3-1 Imagem Replicada em um Plano Infinito.

Esta consideração para a validade da operação justifica o aparecimento de uma cruz central passando pela origem da imagem da transformada. São falsas freqüências espaciais ao longo dos eixos, necessárias para a representação da imagem dentro de uma área limitada. Estas freqüências não são prejudiciais à reconstrução, pelo contrário são necessárias para reconstruir as bordas da imagem. Naturalmente isso só acontece para imagens que possuam bordas laterais com intensidades diferentes, provocando uma descontinuidade após a replicação.

Caso se deseje eliminar a cruz central para uma melhor visualização das freqüências presentes na imagem, uma solução é aplicar uma máscara que suavize a proximidade da borda até que o valor do pixel na mesma seja zero, eliminando a descontinuidade.

A principal propriedade da FT é dada pelo Teorema da Convolução, que diz que a convolução entre duas funções no domínio real tem como transformada a multiplicação das transformadas das duas funções no domínio da freqüência, ou seja:

$$f(x) \otimes g(x) \Leftrightarrow F(u) \cdot G(u) \qquad \text{Equação 3-9}$$

Quando consideramos *g*(*x*) como uma função que modifica *f*(*x*), a operação de convolução no domínio real pode ser compreendida como uma filtragem. Uma operação de multiplicação é muito mais simples do que uma operação de convolução. Mas, para realizarmos a mesma filtragem no domínio da freqüência temos que calcular a FT das duas funções, multiplicá-las e calcular a IFT da imagem resultante, o que aparentemente não nos dá muita vantagem. Uma comparação mais interessante é feita na seção **6 Filtragem no Domínio da Freqüência**.

5. Transformada Rápida de Fourier

A primeira divulgação de forma ampla do algoritmo de transformada rápida de Fourier (FFT - Fast Fourier Transform) ocorreu através do trabalho de J. W. Cooley e J. W. Tukey em 1960, [BRIGHAM 74]. Mas, pelo menos 20 anos antes, Danielson e Lanczos formularam uma das derivações mais claras do algoritmo, [PRESS 89].

Eles mostraram que a DFT pode ser reescrita como a soma de duas outras DFTs, cada uma de tamanho N/2. Uma delas é formada pelos ponto pares e a outra pelos pontos ímpares do conjuto inicial. A prova é simples:

$$\begin{aligned} F_k &= \sum_{j=0}^{N-1} e^{\frac{2pijk}{N}} f_j \\ &= \sum_{j=0}^{N/2-1} e^{\frac{2pik(2j)}{N}} f_{2j+} + \sum_{j=0}^{N/2-1} e^{\frac{2pik(2j+1)}{N}} f_{2j+1} \\ &= \sum_{j=0}^{N/2-1} e^{\frac{2pikj}{N/2}} f_{2j+} + W^k \sum_{j=0}^{N/2-1} e^{\frac{2pikj}{N/2}} f_{2j+1} \\ &= F_k^e + W^k F_k^o \end{aligned} \qquad \text{Equação 3-10}$$

onde W é definido como:

$$W^k \equiv e^{\frac{2pi}{N}} \qquad \text{Equação 3-11}$$

e F_k^e e F_k^o correspondem a k-ésima componente da DFT de comprimento N/2 formada com os componentes pares e ímpares, respectivamente, do conjunto inicial. A grande idéia por tráz desta propriedade é que pode ser usada recursivamente até o ponto em que temos uma transformada de apenas 2 termos, que possue cálculo trivial.

Assim, durante o cálculo da FFT teremos uma progressiva mudança de dimensão da transformada: 2, 4, 8, ..., N. Obviamente N aqui é uma potência de 2, sendo esta a maior restrição deste algoritmo. Como isto também é válido para M, vamos impor que M=N, pois simplificará o tratamento algoritmico. O fato de M e N serem potências de 2 enfatiza a diferença entre eles e na prática acabariamos trabalhando quase sempre com imagens quadradas.

O principal problema aqui é saber como combinar as transformadas de ordem 2, depois as de ordem 4, etc...

A situação mais simples que poderíamos ter é combinar pares adjacentes de elementos, depois combinar elementos resultantes em pares também adjacentes e assim por diante. Para tanto, temos que reordenar os elementos dessa maneira.

Aplicando sucessivas divisões entre pares e ímpares até obtermos uma transformada de 1 termo, chegamos ao seguinte:

$$F_k^{eoeoeoeoeoeoeoeo\ldots oee} = f_n \qquad \text{Equação 3-12}$$

onde f_n não depende de *k* (obs: e - *even*, o - *odd*), nos resta agora descobrir quem é *n* no conjunto original. Se substituirmos os *e*'s e os *o*'s por 0's e 1's, respectivamente, obtemos um número binário. O valor de *n* é exatamente este número com a ordem dos bit invertida. Isto é verdade porque as subdivisões sucessivas em pares e ímpares são testes sucessivos de bits de baixa ordem de *n*.

Para exemplificar isso tomemos uma transformada de 8 pontos {*f*(0), *f*(1), *f*(2), *f*(3), *f*(4), *f*(5), *f*(6), *f*(7)}. A primeira divisão entre pares e ímpares nos fornece dois conjuntos {*f*(0), *f*(2), *f*(4), *f*(6)} e {*f*(1), *f*(3), *f*(5), *f*(7)}. Na segunda divisão temos quatro conjuntos {*f*(0), *f*(4)}, {*f*(2), *f*(6)}, {*f*(1), *f*(5)} e {*f*(3), *f*(7)}. Escrevendo esta ordem na tabela podemos comparar os índices e a ordem dos bits dos mesmos.

Conjunto Original	Índice	Conjunto Reordenado	Índice
f(0)	000	*f*(0)	000
f(1)	001	*f*(4)	100
f(2)	010	*f*(2)	010
f(3)	011	*f*(6)	110
f(4)	100	*f*(1)	001
f(5)	101	*f*(5)	101
f(6)	110	*f*(3)	011
f(7)	111	*f*(7)	111

Tabela 3-1 Reordenação Usando a Inversão de Bits.

Portanto, o algoritmo da FFT é dividido em duas partes. Na primeira os dados são reordenados (*Bit Reversal Section*) e em seguida para cada potência de 2 computam-se os valores das subtransformadas.

Este algoritmo não é nem um pouco trivial. Sua vantagem reside justamente no tempo de processamento muitas vezes menor que o método de cálculo direto. Isto pode ser melhor evidenciado através do conceito de Complexidade Algoritmica.

Tomando em consideração o número de multiplicações complexas, a equação 3-1 possui NxN multiplicações, que são realizadas para todos os elementos da imagem, ou seja, NxN pontos. Portanto, a complexidade da DFT é obviamente $O(N^4)$. No caso da FFT, continuamos possuindo NxN pontos. Mas, a subdivisão sucessiva possue complexidade O(logN), resultando para a FFT $O(N^2 x \log N)$, um resultado que valida todo o esforço de se implementar o algoritmo. No fundo estamos aplicando aqui o conceito de dividir para conquistar em um conjunto de N pontos (colunas), o que dá O(NxlogN), repetido para as N linhas, totalizando $O(N^2 x \log N)$.

Embora tenhamos passado por todas as etapas do algoritmo, na prática não é utilizada a recursão e a implementação se torna complexa. Dentre os algoritmos pesquisados existem diversas pequenas diferenças, a literatura apenas explica o funcionamento de cada um e não como se chegou ao mesmo. Portanto, acaba-se por copiar algum algoritmo implementado, entendendo-se como funciona mas não como deduzi-lo.

O algoritmo de FFT é feito *in-place*, ou seja, sobre o próprio conjunto de dados complexos, no caso de imagens reais preencheríamos o conjunto complexo com a parte imaginária nula e

realizaríamos o cálculo. Mas, podemos ainda utilizar um truque para o cálculo das colunas, utilizando a parte imaginária como se fosse uma divisão entre pares e ímpares, ou seja, coloca-se os termos pares do conjunto real na parte real do conjunto complexo e os termos ímpares na parte imaginária. Estamos então utilizando uma transformada complexa de N pontos para calcular duas também de N pontos. Após o cálculo temos que separar o resultado utilizando o fato de que a transformada de um conjunto real é complexa hermitiana.

Existem ainda diversas variações do algoritmo de FFT. Quando se realiza os cálculos primeiro e depois a reordenação o algoritmo tem o codnome de Sande-Tukey, onde antes era Cooley-Tukey. Também pode-se acelerar o processo de cálculo forçando N a ser uma potência de 4 ou 8, neste caso são chamados de FFT de ordem 4, 8 (quando a ordem não é especificada, assume-se ordem 2). Existem ainda algoritmos para $N=n_1 \times n_2 \times n_3$ ... onde os n's são números primos. Nestes dois últimos casos é preciso saber balancear a complexidade de implementação do algoritmo com o ganho de performance obtido.

A DFT também pode ser obtida a partir de uma outra transformada, a de Hartley [O'NEILL 88]. Utilizando um algoritmo semelhante ao da FFT a FHT (Fast Hartley Transform) é mais rápida e consome menos recursos que a FFT, pois transforma dados reais em dados também reais. A DFT é obtida usando as equações a seguir:

$$F(u) = (H(u) + H(N-u)) + (H(u) - H(N-u))i \qquad \text{Equação 3-13}$$

O módulo da DFT pode ser obtido usando:

$$|F(u)| = \frac{(H(u)^2 + H(N-u)^2)}{2} \qquad \text{Equação 3-14}$$

Infelizmente a FHT não foi aproveitada neste trabalho. Mas convém ser mencionado para futuros melhoramentos.

Vimos nesta seção as diversas facetas do algoritmo de FFT e seu uso em imagens digitais. Agora que já conseguimos chegar até a imagem complexa no domínio da freqüência, podemos introduzir as técnicas de filtragem.

6. Filtragem no Domínio da Freqüência

Como explicado anteriormente, a técnica da filtragem no domínio da freqüência é muito simples, pois equivale a uma multiplicação. Na Figura 3-2 podemos ver uma seqüência inteira de filtragem, tanto no domínio real, como no domínio da freqüência. Na primeira parte vemos uma senóide modelada com um ruído também senoidal. Como a freqüência do ruído é maior que a da senóide, este se manifesta como os dois impulsos mais externos no domínio da freqüência. Para removê-los basta aplicar um passa baixa no domínio da freqüência, que de volta ao domínio real obteremos a senóide sem o ruído.

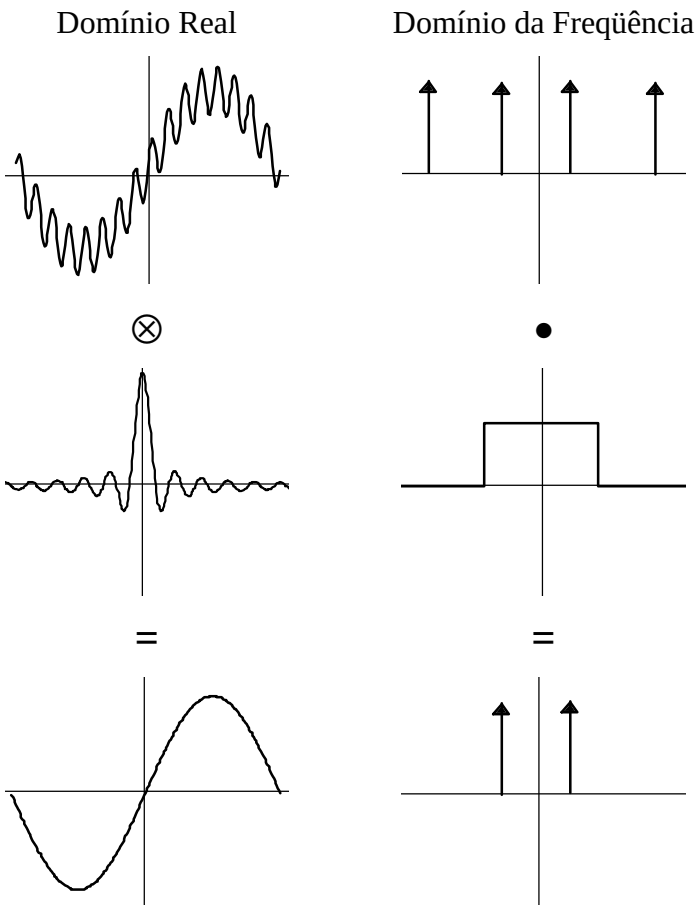

Figura 3-2 Comparação entre Filtragem no Domínio Real e no da Freqüência

Mas, surge um outro problema: estamos trabalhando com uma função discretizada e com a DFT e não a FT. Como vimos a DFT é uma aproximação para a FT. Ao aplicarmos algum tipo de filtragem com um filtro semelhante ao da Figura 3-2 (ideal), a transformada inversa fornecerá uma imagem com um fenômeno chamado *ringing* (anelamento). Visualmente este efeito é identificado

por “anéis” (também chamados de artefatos) que surgem nas bordas dos objetos da imagem. Este fenômeno acontece devido a descontinuidade imposta pelo filtro ideal.

Para evitar a introdução de artefatos após a filtragem os filtros devem ter sua borda suavizada por uma função com derivada contínua e faixa de decaimento controlada. É muito comum o uso de uma gaussiana ou de um cosseno para tal. O controle da faixa de decaimento permite o balanceamento entre a existência de artefatos e o quanto o filtro permitirá passar outras freqüências que não aquelas sob o mesmo.

É preciso então saber balancear até que ponto a filtragem no domínio real é vantajosa em relação ao domínio das freqüências. A convolução no domínio real é definida como:

$$f(x,y) = \sum_{j=0}^{K}\sum_{i=0}^{K} k(i,j)\, f\left(x + i - \frac{K}{2}, y + j - \frac{K}{2}\right) \qquad \text{Equação 3-15}$$

onde *k* é uma matriz quadrada de K^2 elementos.

Usando novamente o conceito de Complexidade Algoritmica, vemos que a Equação 3-15 possue complexidade $O(K^2N^2)$. A filtragem no domínio da freqüência exige um FFT, uma multiplicação pelo filtro e uma IFFT. Assim, a complexidade resultante é $O(N^2 x \log N) + O(N^2) + O(N^2 x \log N) = O(N^2 x \log N)$, ou seja comparável a convolução no domínio real. É claro que o conceito de complexidade esconde diversas constantes que se tornam relevantes na prática. Mas, de qualquer maneira isso mostra que o tamanho da matriz de convolução K não pode ser muito grande.

Na prática K assume valores 3, 5, 7 e 9, e a filtragem no domínio da freqüência permite uma seleção mais precisa de que elementos se deseja filtrar. Pois, a filtragem espacial não deixa de ser uma média ponderada da vizinhança de um determinado *pixel*, em contraste com uma visão global fornecida pela FT.

7. Os Filtros

Como dito anteriormente, vários dos fenômenos ocorridos no domínio da freqüência são simples de serem modelados.

A filtragem mais básica de todas e também a mais conhecida é o filtro passa faixa. Quando este está próximo à origem é chamado de passa baixa. Quando está afastado é chamado de passa alta. No domínio bidimensional este filtro é representado por dois anéis concêntricos e centrados na origem, Figura 3-3.

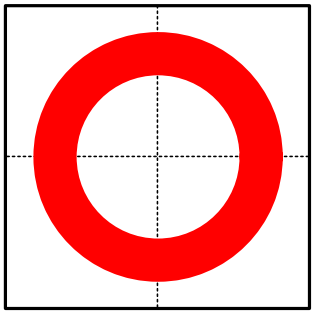

Figura 3-3 Setor Circular Centrado na Origem.

Dois casos particulares do filtro anterior acontecem quando queremos isolar apenas freqüências horizontais ou apenas verticais, como visto na Figura 3-4 e Figura 3-5, respectivamente.

Repare que para isolar freqüências horizontais necessitamos de um retângulo vertical, e vice-versa. Também pode-se ver dois retângulos simétricos em relação à origem, isto é necessário pois estamos supondo que a imagem complexa no domínio da freqüência proveio de uma imagem real no domínio espacial. Sendo assim, os filtros devem obedecer o fato das imagens complexas em questão serem hermitianas. Assim, quando calcularmos a transformada inversa obteremos uma imagem complexa com parte imaginária nula (próxima de zero), ou melhor, uma imagem real.

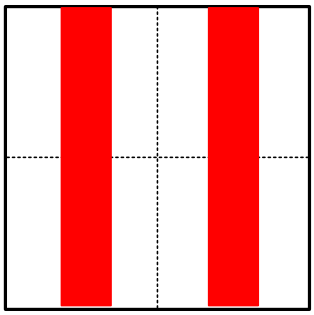

Figura 3-4 Retangular Vertical

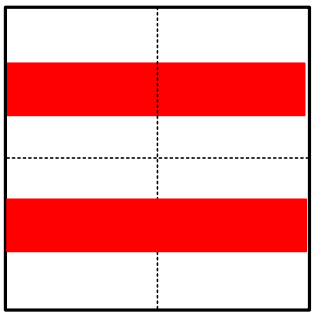

Figura 3-5 Retangular Horizontal

Quando a imagem possue freqüências espaciais muito bem determinadas em ambas as direções estas freqüências se manifestam no domínio da freqüência através de pontos brilhantes, ou melhor, uma região brilhante ao redor de um ponto mais intenso, um pico. Para isolar estes picos necessitamos de um filtro que envolva os mesmos, o filtro da Figura 3-6 cumpre exatamente esse papel. Observe também sua característica hermitiana.

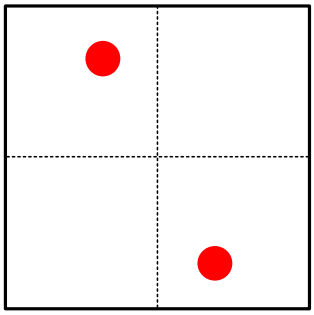

Figura 3-6 Oval Não Centrada na Origem

Estes picos comumente se manifestam de forma periódica. O filtro da Figura 3-7 simplifica o trabalho de repetirmos o filtro anterior para cada um dos picos presentes na imagem.

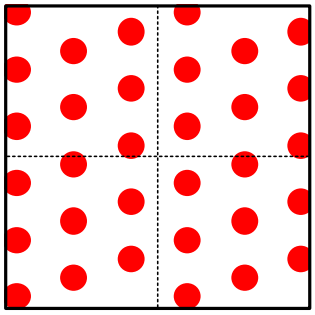

Figura 3-7 Oval Periódica

Algumas imagens podem ter freqüências distribuídas ao longo de uma direção inclinada. Estas freqüências aparecem com o formato de um setor angular como na Figura 3-8.

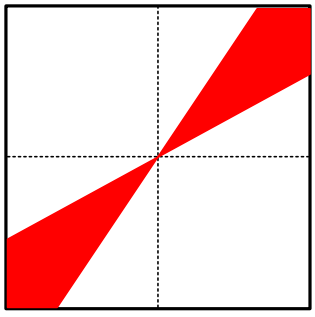

Figura 3-8 Setor Angular

Quando se deseja incluir toda informação próxima à origem, utiliza-se o filtro da figura Figura 3-9.

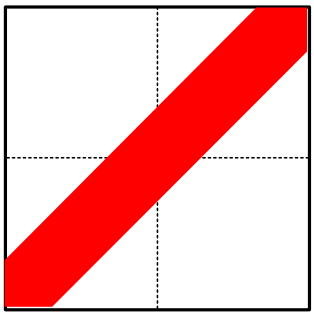

Figura 3-9 Retangular Inclinado Centrado na Origem

Portanto, vimos que os mais conhecidos fenômenos podem ser modelados por filtros simples de serem criados e manipulados, pois podem ser utilizados objetos geométricos bem conhecidos. Naturalmente, todos devem poder possuir uma borda suave durante a aplicação sobre a imagem. Matematicamente, a combinação de um ou mais filtros implica na união da área delimitada pelos mesmos. A área demilitada pela combinação dos filtros pode sobrepor a imagem durante a aplicação, atuando como um filtro que “passa” freqüências, ou o complemento desta área sobrepõe, atuando como um filtro que “rejeita” freqüências.

O método de filtragem com estes filtros é o mesmo descrito na Figura 3-2. A imagem original no domínio da freqüência é multiplicada pela imagem formada pelo uso de um ou mais filtros sobre a mesma.

A seqüência completa de filtragem pode ser vista na Figura 3-10. Deve-se aplicar a transformada de Fourier direta sobre a imagem que se deseja filtrar, criar e combinar os filtros que melhor isolem o fenômeno desejado, aplicá-los sobre a imagem complexa, calcular a transformada de Fourier inversa e por fim obtem-se a imagem original filtrada.

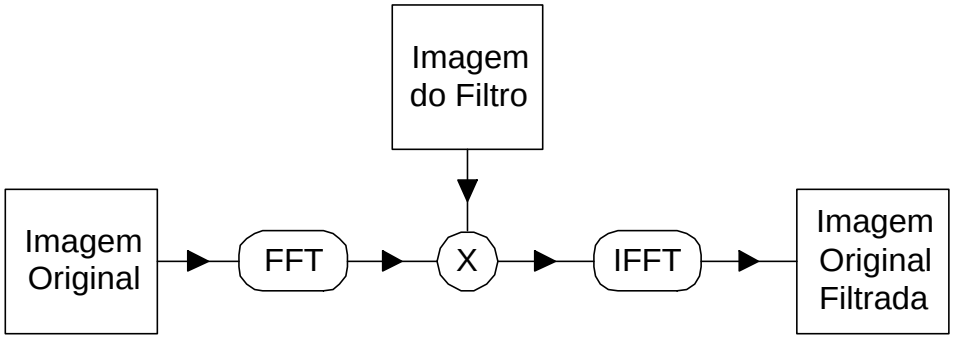

Figura 3-10 Seqüência Completa de Filtragem

8. IMPLEMENTAÇÃO

9. O SPID

O SPID é um sistema de processamento de imagens digitais dedicado à aplicações científicas. Isto significa que o tratamento quantitativo da imagem é preferencial ao tratamento qualitativo. Portanto, não se encontrará nele funções de manipulação de imagens coloridas que prezem pela fidelidade na representação da cor, ou seja, cor ou tom de cinza, no nosso caso, é apenas uma representação visual para a informação contida na imagem.

Assim, uma imagem digital é essencialmente uma matriz de *pixels*, onde cada elemento contém um valor numérico. Temos agora que balancear a precisão numérica desejada com o tamanho em bits do *pixel*. Afinal, imagens digitais consomem uma boa quantidade de memória.

A Tabela 8-1 mostra os tipos de pixel disponíveis e seus respectivos tamanhos. Para imagens coloridas são utilizados três valores respectivos às componentes de vermelho (red), verde (green) e azul (blue) da cor (RGB). Este é o único modelo de cor utilizado. Não se leva em consideração o referencial do sistema de coordenada de cor.

Multiplicando os valores em bytes por 3 para imagens coloridas RGB podemos chegar a imagens bem grandes. Portanto é preciso saber escolher muito bem o tipo de *pixel* com que se irá trabalhar.

O SPID classifica os tipos de imagem com que trabalha de maneira mais intuitiva. As imagens podem ser coloridas usando o padrão RGB ou em tons de cinza. Cada componente ou o tom pode ter uma precisão inteira, inteira positiva, ponto flutuante e complexa, onde um número complexo é visto como dois números de ponto flutuante. A precisão inteira, com ou sem sinal, pode ter 8, 16 ou 32 *bits* por *pixel*. A precisão de ponto flutuante pode ter 32 ou 64 *bits* por *pixel*.

Nome	Domínio	Bits / Bytes
byte integer	-128 ; 127	8 / 1
unsigned byte integer	0 ; 255	8 / 1
integer	-32.768 ; 32.767	16 / 2
unsigned integer	0 ; 65.535	16 / 2
long integer	-2.147.483.648 ; 2.147.483.647	32 / 4
unsigned long integer	0 ; 4.294.967.295	32 / 4
IEEE floating point	$3{,}4x10^{-38}$; $3{,}4x10^{+38}$	32 / 4
IEEE floating point double precision	$1{,}7x10^{-308}$; $1{,}7x10^{+308}$	64 / 8
complex	(*real*, *real*)	64 / 8
double complex	(*double real*, *double real*)	128 / 16

Tabela 8-1 Tipos de *Pixel*

As imagens devem ser visualizadas em algum dispositivo gráfico. Os dipositivos disponíveis trabalham somente com *pixels* do tipo *unsigned byte integer*, ou inteiro positivo de 8 bits por *pixel*.

Como um dispositivo colorido RGB necessita de uma grande quantidade de memória, foram criados dispositivos coloridos indexados. Nestes dispositivos o valor do pixel é um índice para uma tabela de valores RGB. A partir disso criaram-se as imagens coloridas com palette de 256 e 16 cores.

Alguns dispositivos mais antigos só permitiam a utilização de duas cores, estas imagens são chamadas de binárias.

As imagens que podem ser visualizadas diretamente em dispositivos gráficos são classificadas no SPID como *bitmaps*. As que não o são, possuem um mecanismo de varredura de valores máximo e mínimo para serem utilizados pela normalização durante a obtenção de um *bitmap* para visualização. Estas imagens são chamadas científicas.

O mecanismo de varredura é chamado de *survey* e possue três tipos: nenhum (é realizado o *clipping* de cor), uma cruz central ou toda a imagem. Estas variações permitem um *survey* mais rápido.

No caso de imagens complexas existe ainda um outro problema que consiste na conversão de complexo para real antes do *survey*. Esta conversão está disponível de várias maneiras no SPID.

Podem ser observados: o logaritmo da magnitude, a magnitude ao quadrado, a magnitude, a fase[1], a parte real[1] e a parte imaginária[1].

Conhecidas as imagens com que iremos trabalhar é necessário um modo de armazenamento para as mesmas. Adotou-se o padrão TIFF, ver [ALDUS 92], que vem se estabelecendo no mercado como o padrão mais adotado por sistemas de processamento de imagens científicos ou não. Devido a sua específicação genérica, o TIFF permite o armazenamento de todos os tipos de imagens mencionados com excessão da imagem complexa. Mas, sua versatilidade é tal que pudemos acrescentar uma definição ao TAG 339 - **Data Sample Format**, onde **ValueOffset** igual a 10 indica um valor complexo composto por dois valores IEEE *floating point*, o primeiro representando a parte real e o segundo representando a parte imaginária. Uma outra opção seria usar **SamplesPerPixel** igual a 2, mas isto invalidaria o uso de uma imagem colorida RGB complexa.

A FFT exige que a imagem seja quadrada com lados uma potência de 2. Também imporemos a condição de que a imagem seja científica.

O SPID permite a conversão entre os diversos tipos de imagem disponíveis e permite a seleção de uma área de operação sobre a imagem. Esta área pode ser forçada a um quadrado como também um quadrado com lados uma potência de 2. A operação de FFT automaticamente identifica que há uma selação na imagem e só utiliza aquela parte da imagem original para criar a imagem da transformada.

Como todos esses recursos o SPID possue um ambiente pronto e adequado para o processamento desejado neste trabalho. Nos resta apenas a criação e manipulação das máscaras, explicadas no tópico **15 As Máscaras**. Aqui o termo máscara é utilizado no lugar de filtro, pois este nome é mais adequado para o contexto de um ambiente interativo, onde o que se está fazendo é aplicando uma máscara sobre a imagem.

Em sendo um programa para o ambiente **Microsoft® Windows™**, o SPID possue diversos recursos comuns a programas deste ambiente. O mais importante destes recursos talvez seja o fato do SPID ser um aplicativo MDI (**Multiple Document Interface**), isso permite com que sejam abertas várias imagens ao mesmo tempo, possibilitando a comparação visual entre elas.

---

[1] Aqui estamos falando do módulo do valor, pois a FFT possue um problema de oscilação do sinal do valor. Teríamos que na verdade também dispor de uma outra opção sem o módulo.

O **Windows** também possue um gerenciador de hipertextos dedicado a implementação de um On-line Help. Este hipertexto pode ser acessado de dentro do SPID de quatro maneiras: a partir de algum item do menu de Help; a partir da tecla F1 pressionada sobre algum menu; a partir do **HelpCursor**; ou a partir do botão de Help de algum diálogo.

Pode-se abrir uma ou mais imagens de uma só vez de duas maneiras: através do diálogo de **FileOpen**, onde também pode-se obter informações sobre o arquivo antes de abri-lo e pode ser visto um **Preview** da imagem; ou através do mecanismo de **Drag** & **Drop** através do **FileManager**. Também pode-se abrir uma imagem através do uso do **Clipboard** ou associando a extensão do arquivo no **FileManager** ao SPID e executando um *double-click* sobre o nome do arquivo.

A imagem visualizada no SPID pode sofrer um aumento ou redução (*zoom*) apenas visual para um melhor enquadramento da mesma. Existem 16 níveis de *zoom in* e 16 de *zoom out*. Esta mesma visualização pode ser enviada para a impressora, onde a imagem aparecerá no centro do papel.

As operações no SPID podem ser feitas sobre a imagem original ou sempre criando uma nova imagem. Quando a operação é feita sobre a imagem original o SPID possue o recurso de **Undo**, ou seja, desfazer a última operação. Aproveitando a informação do **Undo** após este realizado, fica disponível a operação de **Redo**, que refaz a última operação.

Além do menu, como elemento de interface com o usuário existem ainda cinco janelas que facilitam a operação do programa.

A **SpeedBar,** localizada no topo ou nas laterais da área cliente da janela principal, fornece botões para o acesso mais rápido a itens do menu.

A **StatusBar**, localizada na parte de baixo da área cliente, é usada de duas maneiras. A primeira é para dicas sobre o item de menu selecionado ou sobre o botão abaixo do cursor do *mouse* na **SpeedBar**. A segunda é para o contador da porcentagem de processamento de uma determinada operação sendo realizada, ao lado do contador um texto indica qual a operação.

As outras três janelas ficam localizadas em qualquer posição e não são limitadas à área cliente da janela principal. A **ToolBox** fornece ferramentas de manipulação da imagem através do uso do mouse, tais como a criação das máscaras, a seleção de uma área, o *zoom*, etc... A **InformationWindow** fornece dados sobre a imagem abaixo do cursor do *mouse*, tais como: tipo de pixel, posição, valor do pixel, dimensões e posição de um objeto. A **ResultWindow** possue um

editor de texto independente de qualquer imagem para onde podem ser enviadas informações textuais, que podem então ser editadas livremente. Um exemplo é a obtenção da intensidade de pontos da imagem através de um *right click* sobre a mesma.

10. Arquitetura Interna

O SPID é dividido em duas grandes partes: gerenciamento e operações de imagens; e interface. A parte de interface é naturalmente limitada para uso em ambiente **Microsoft® Windows™**. Já a primeira parte pode ser portada para outros sistemas operacionais, contanto que possuam um compilador de C++ que inclua *templates*.

11. Classes de Imagens

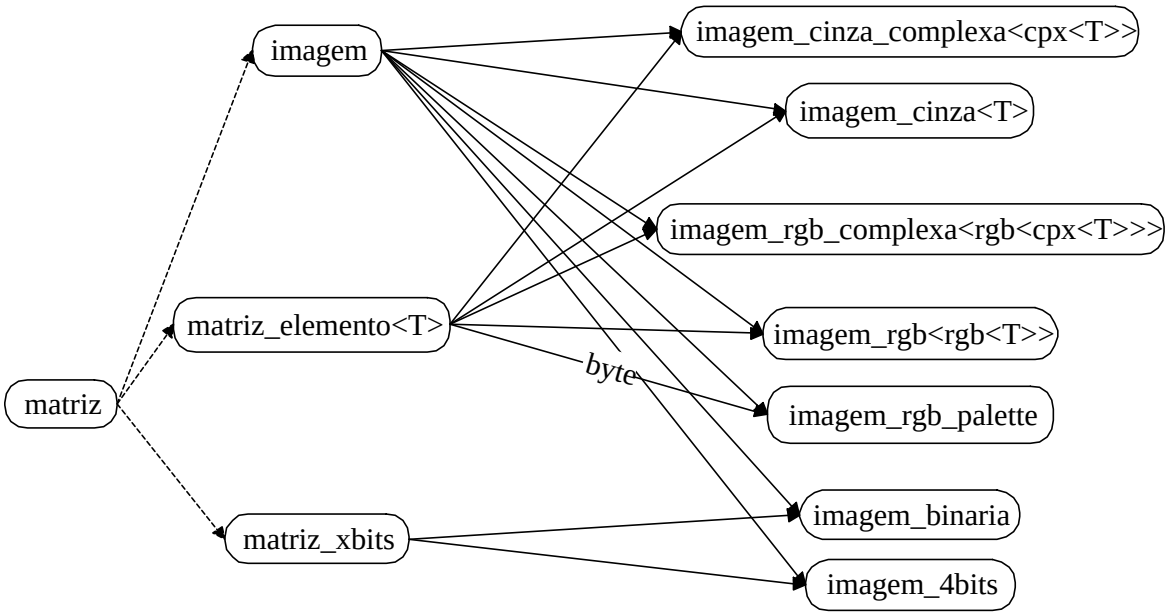

Figura 8-1 Classes de Imagens

Imagens são objetos simples. No entanto, devido ao fato de termos vários tipos de imagens, a manipulação destas se torna complicada se não houver uma disponibilidade de classes adequada.

O processo de criação da Figura 8-1 vai nos ajudar a entender porque chegamos a uma árvore de herança tão singular.

Em primeiro lugar, como já dito, imagem é uma matriz, onde cada elemento representa a cor numa determinada coordenada. Surge a classe **matriz**, ainda sem a definição de qual tipo de pixel estamos tratando. Portanto, uma classe virtual.

Em um computador a menor unidade que podemos alocar é um byte. Portanto, surge a necessidade de duas classes: a **matriz_elemento<elemento>**, para gerenciar uma matriz com elementos[2] múltiplos de um byte; e a **matriz_xbits**, para gerenciar uma matriz com elemento menor que 8 bits. Observe as linhas tracejadas na figura indicando uma herança virtual. Isso será necessário para o que pretendemos.

A partir da **matriz_xbits** chegamos à **imagem_4bits**, que gerencia uma imagem com 16 tons de cinza ou 16 cores indexadas, e à **imagem_binaria**, que gerencia uma imagem com *pixel* possuindo apenas dois valores, preto e branco.

As classes **imagem_cinza** e **imagem_rgb** gerenciam todos os tipos de imagens, exceto as complexas, em tons de cinza e coloridas rgb, respectivamente. A herança da classe **matriz** é feita diferenciadamente para a classe **imagem_rgb** pois cada elemento agora são três. Para isso criou-se a classe **rgb** que também recebe um *template* possibilitando esta simplicidade. As classes **imagem_cinza_complexa** e **imagem_rgb_complexa** são colocadas em separado para que não haja conflito dentro dos membros das classes **imagem_cinza** e **imagem_rgb**. Também foi criada a classe **cpx** também paramétrica, simplificando a criação das classes de imagens complexas.

A classe **imagem_rgb_palette** só necessita de um byte para o elemento da matriz pois as cores são indexadas. Assim, é colocada em separado.

E por fim, para em tempo de execução escolhermos com que tipo de imagem iremos trabalhar, vem a necessidade da classe virtual **imagem**. Novamente, esta classe herda propriedades da classe virtual **matriz**. Assim, completamos o ciclo de herança virtual mostrado na Figura 8-1, que é a base que sustenta todo este trabalho. A classe **imagem** também contém inúmeros elementos comuns a todos os tipos de imagem, tais como número de bits por *pixel*, precisão, descrição, resolução, escala.

---

[2] O tipo passado como parâmetro na criação do template para a classe matriz deve possuir operações aritméticas e atribuição.

Para o gerenciamento de todos esses tipos de imagem foi criada a classe **ImageManager**, que possue funções para criação e conversão de imagens. A partir de uma descrição[3] genérica o **ImageManager** realiza a escolha da classe de imagem adequada aquela descrição. A conversão entre imagens é separada em três blocos: conversão entre bitmaps; conversão rgb-cinza e vice-versa; mudança de precisão. Durante a conversão rgb-cinza o pixel mantém a sua precisão; a mudança de precisão pode ser feita entre quaisquer dos tipos básicos vistos na Tabela 8-1; a conversão mais complexa é entre *bitmaps*, pois os algoritmos de conversão rgb 24 bits para 256 e 16 cores, e os de binarização não são imediatos e únicos.

12. Operações

A metodologia de operação sobre imagens é um pouco mais complicada. Principalmente por combinar os mecanismos de ligação dinâmica (*dinamic binding*) e métodos virtuais. A idéia aqui foi separar as operações dos métodos das classes. Para isso foi quebrado o conceito de encapsulamento. Mas dessa maneira pode-se concentrar a implementação de uma determinada operação.

A Figura 8-2 mostra o ponto de partida para a execução de uma determinada operação. O fato de Im ter sido criada como uma imagem_cinza<byte> fará com que o método OpGlobal da classe imagem_cinza<byte> seja chamado.

```
imagem* Im = imagem_cinza<byte>(Linhas, Colunas);
...
teste Op(Param);
imagem* NewIm = Im->OpGlobal(Op);
```

Figura 8-2 Trecho do Programa Principal

Na Figura 8-3 vemos que OpGlobal chama o método Apply da classe operador_global. Mas como esse método é declarado virtual, o método Apply da classe teste, que foi passada como parâmetro originalmente, é chamado (Figura 8-4).

O método Apply da classe teste finalmente realiza a operação e retorna uma nova imagem como resultado. Em resumo, a implementação de uma operação nova requer apenas que implementemos os métodos Apply para cada um dos tipos de imagem existentes. Se uma

[3] A descrição inclue: rgb ou cinza, precisão, número de bits por *pixel*, palette se existente.

determinada operação não tem sentido para um determinado tipo de imagem, o método Apply respectivo deve retornar NULL.

```
class imagem
      {
      ...
      virtual  imagem* OpGlobal(operador_global&) =
0;
      ...
      };

template <class elemento>
class imagem_cinza: public matriz, public imagem
      {
      ...
      imagem* OpGlobal(operador_global&);
      ...
      };

template <class elemento>
imagem* OpGlobal(operador_global& Op)
      {
      return Op.Apply(*this);
      }
```

Figura 8-3 Método OpGlobal da Classe imagem

No método Apply é muito comum a utilização de *templates* para simplificar a implementação da operação. Mas, para algumas imagens *bitmaps* isso pode não ser possível e a operação é implementada no próprio método Apply.

```
class operador_global
      {
      ...
      virtual imagem* Apply(const imagem_cinza<byte>&) = 0;
      ...
      };

class teste: public operador_global
      {
      teste(tipo_param_teste param);
      ...
      imagem* Apply(const imagem_cinza<byte>&);
      ...
      };
```

Figura 8-4 Método Apply da Classe operador_global

13. Classes de Arquivos

Figura 8-5 Classes de Arquivos de Imagem

Arquivos que contém imagens são normalmente arquivos binários e possuem uma estrutura interna bem determinada. As classes mais à direita na Figura 8-5 correspondem aos padrões mencionados anterormente. A classe **arquivo_imagem** é necessária para a escolha do tipo de arquivo em tempo de execução.

14. Interface

Utilizou-se a biblioteca de classes OWL[4] para o ambiente **Microsoft Windows**, que foi fornecida com o compilador utilizado, o **Borland C++ 4.0** [BORLAND 93].

A partir de classes da OWL criam-se as classes específicas do **SPID**. Este trabalho é ainda simplificado utiizando-se um recurso do compilador chamado **ApplicationExpert**, que cria todo o esqueleto do programa, com relação a interface.

Estas classes criadas são atualizadas através do **ClassExpert**, que também permite a criação de novas classes de interface, sempre baseadas em classes da OWL.

15. As Máscaras

Como visto, as máscaras são criadas a partir de uma ferramenta da **ToolBox**. Após selecionar a ferramenta o usuário executa um *click* & *drag* sobre a imagem e a máscara é automaticamente criada e acrescentada à lista de anotações. Uma anotação é qualquer objeto que possa ser colocado sobre a imagem. As máscaras são um tipo de anotação.

---

[4] Object Windows Library

Logo após a criação de uma anotação a **ToolBox** retorna ao seu estado *default*, a **CrossCursor**. Com esta ferramenta o usuário pode selecionar uma ou mais anotações para copiá-las para outra imagem, apagá-las, duplicá-las, etc...

A **CrossCursor** também permite modificar a posição e o tamanho de uma anotação. Deve-se observar o estado do cursor para identificar que tipo de operação pode ser realizada: *resize* ou *move*. Cada anotação reagirá de acordo com o seu formato. Os valores de posição e tamanho de uma anotação podem ser dados manualmente após a execução de um *right-click* sobre a mesma.

As anotações sobre uma determinada imagem podem ser salvas em um arquivo texto usando uma descrição suscinta para cada uma. Esta mesma descrição pode ser enviada para a **ResultWindow**. Da mesma forma pode-se ler um arquivo texto com esta descrição para criar as anotações desejadas sobre uma imagem.

As máscaras foram escolhidas a partir dos estudos do tópico **7 Os Filtros**.

O **Setor Circular Centrado na Origem** (*band mask*) é descrito pelos raios interno e externo. Tanto o círculo externo quanto o interno podem ser manipulados. Mas ambos permanecerão centrados na origem. A operação de mover a máscara fará com que o setor circular se mova mantendo a largura de banda constante e os dois círculos centrados na origem.

A **Oval Não Centrada na Origem** (*twin oval mask*) é descrita pela posição do centro da elipse, e pelo tamanho dos semi-eixos vertical e horizontal. As ovais (elipses) podem navegar à vontade pela imagem. Embora, sejam dependentes entre sí devido à simetria da imagem complexa hermitiana. Assim, quando uma se move numa direção, a outra realiza o movimento simétrico. Seus eixos podem ser modificados independentemente. Pode-se forçar com que os eixos mantenham sua proporcionalidade pressionado-se a tecla <**Ctrl**>, e com que a elipse seja um círculo pressionando-se as teclas <**Ctrl**> e <**Shift**> simultaneamente ao modificá-la.

A **Oval Periódica** (*Periodic mask*) é descrita pela peridiocidade, pela posição do centro da elipse de manipulação, e pelo tamanho dos eixos vertical e horizontal desta elipse. Naturalmente todas as elipses são dependentes entre sí. A peridiocidade é modificada pressionando-se a tecla <**Alt**> e movendo a máscara. As outras características seguem o caso anterior.

A **Retangular Vertical** (*horizontal band pass mask*) é descrita pelas posições das verticais externa e interna. As duas faixas verticais são naturalmente dependentes. Pode-se movê-las

mantendo-se a largura de banda constante ou pode-se manipular a vertical externa ou a vertical interna para modificar o tamanho.

A **Retangular Horizontal** (*vertical band pass mask*) é similar a anterior, só que é descrita pelas posições das horizontais externa e interna. A manipulação da máscara é equivalente à anterior.

A **Retagular Inclinada Centrada na Origem** (*diagonal band pass mask*) é descrita pela largura da banda e pelo ângulo que a reta que passa pela origem faz com o eixo horizontal. Neste caso ao mover a máscara o ângulo da faixa retangular será modificado. A lateral da faixa também pode ser modificada para aumentar ou reduzir a banda.

O **Setor Angular** (*wedge mask*) é descrito pelos ângulos externo e interno do setor angular. Neste caso, ao mover a máscara estar-se-á modificando o ângulo do setor angular em relação aos eixos. O ângulo interno pode ser modificado movendo-se uma das laterais do setor.

16. EXEMPLOS

As figuras 5-1, 5-2 e 5-3 foram incluídas com o objetivo de demonstrar a corretude da FFT utilizada e de mostrar algumas propriedades da imagem obtida com a Transformada de Fourier. As figuras seguintes exemplificam a metodologia de filtragem usando alguns dos filtros propostos.

Um pulso bidimensional pode ser representado de várias maneiras. Apresentamos duas delas, a primeira com duração igual em todas as direções a partir da origem e a segunda com duração igual ao longo de cada eixo, ou seja, um círculo (Figura 16-1) e um quadrado (Figura 16-2), respectivamente. Sabemos que a FT de um pulso é uma função sinc[5], como visto na Figura 3-2. Na FT da imagem com o círculo a sinc se manifestará em todas as direções. Já na FT da imagem com o quadrado, a sinc se manifestará apenas na direção vertical e na horizontal. No caso bidimensional estamos observando a função de um ponto de vista superior, como se a imagem estivesse no plano X-Y e estivessemos sobre o eixo Z olhando para baixo.

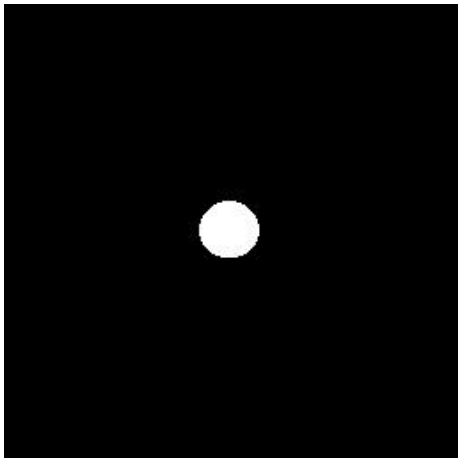

Figura 16-1 FFT de um Círculo

[5] Lembre-se de que estamos observando a magnitude do valor complexo, o que faz com que a sinc pareça ter mais picos positivos do que deveria.

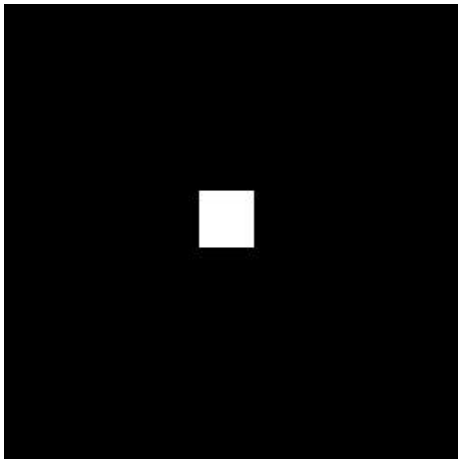

Figura 16-2 FFT de um Quadrado

A Figura 16-3 mostra um grid e sua transformada. O grid pode ser interpretado como um trem de impulsos bidimensional, desconsiderando a linha contínua de freqüência zero. Sendo assim sua transformada é também um trem de impulsos (parte imaginária zero, uma função, ou melhor imagem real), mas com uma periodicidade diferente. É interessante observar que as linhas contínuas são de certa maneira consideradas ao observarmos que os valores da cruz central são maiores que os valores internos aos quadrantes.

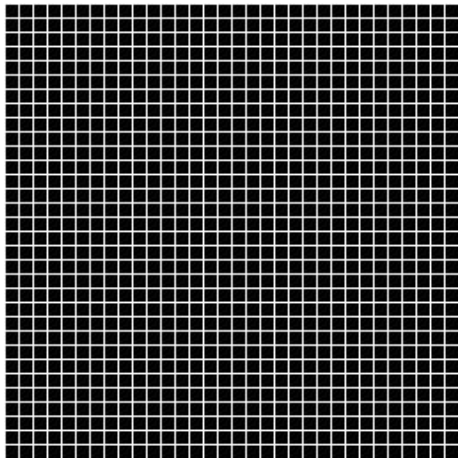

Figura 16-3 FFT de um Grid

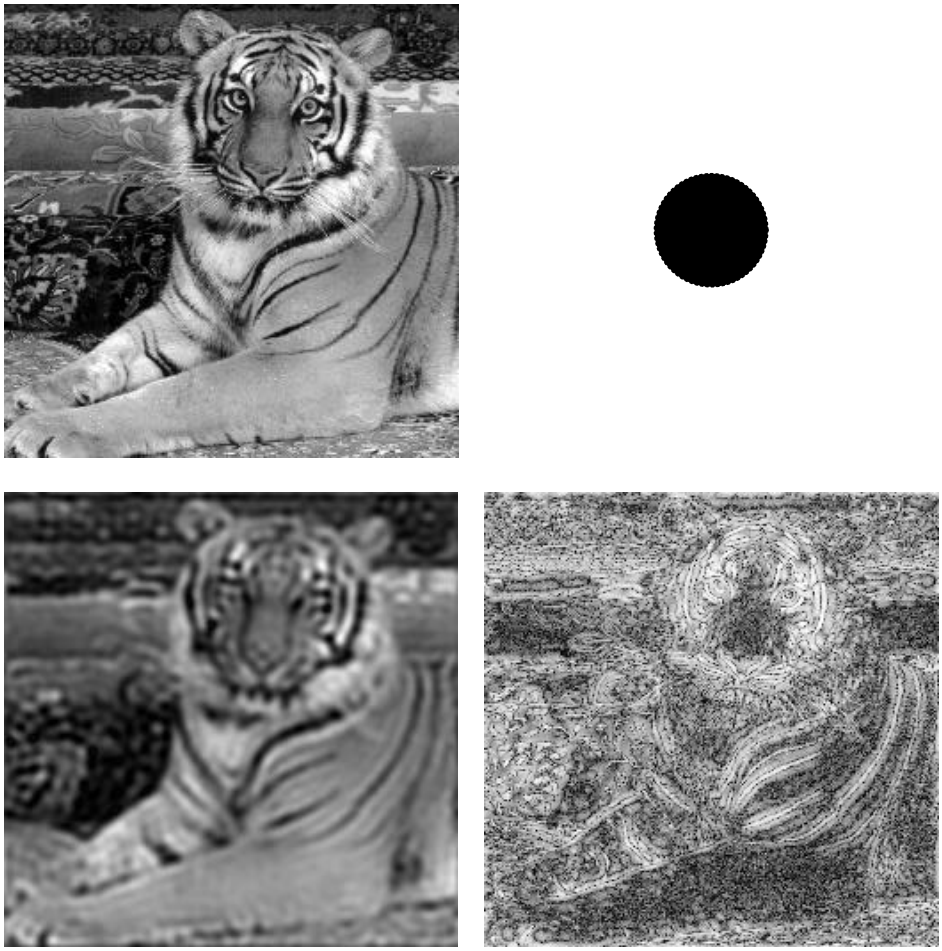

Figura 16-4 Filtragem de Baixas e Altas Freqüências

Utilizando um filtro passa baixa bidimensional, como visto na Figura 16-4, obtemos uma imagem “borrada”, ou seja ocorre uma perda de detalhes que são compostos de altas freqüências. Aplicando o inverso deste filtro explicitamos os detalhes perdidos na imagem anterior.

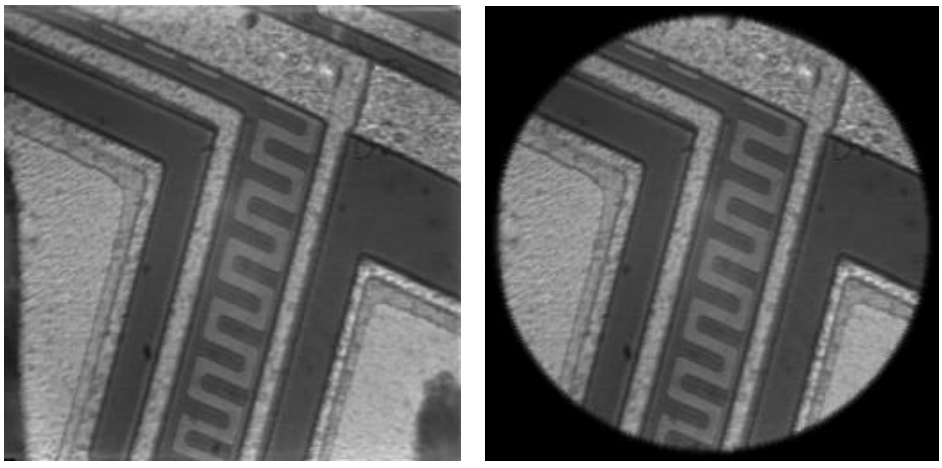

Figura 16-5 Eliminação da Cruz Central

Como visto na seção 4 Transformada de Fourier a transformada de uma imagem com bordas distintas gera a cruz central vista na Figura 16-5. Aplicando-se uma máscara na imagem original que elimina essa discontinuidade, obtemos uma imagem complexa da trasnformada sem a cruz central, o que permite uma melhor visualização dos fenômenos ocorridos no domínio da freqüência. Este procedimento é pouco usado pois é necessário uma interferência na imagem original muitas vezes não desejada.

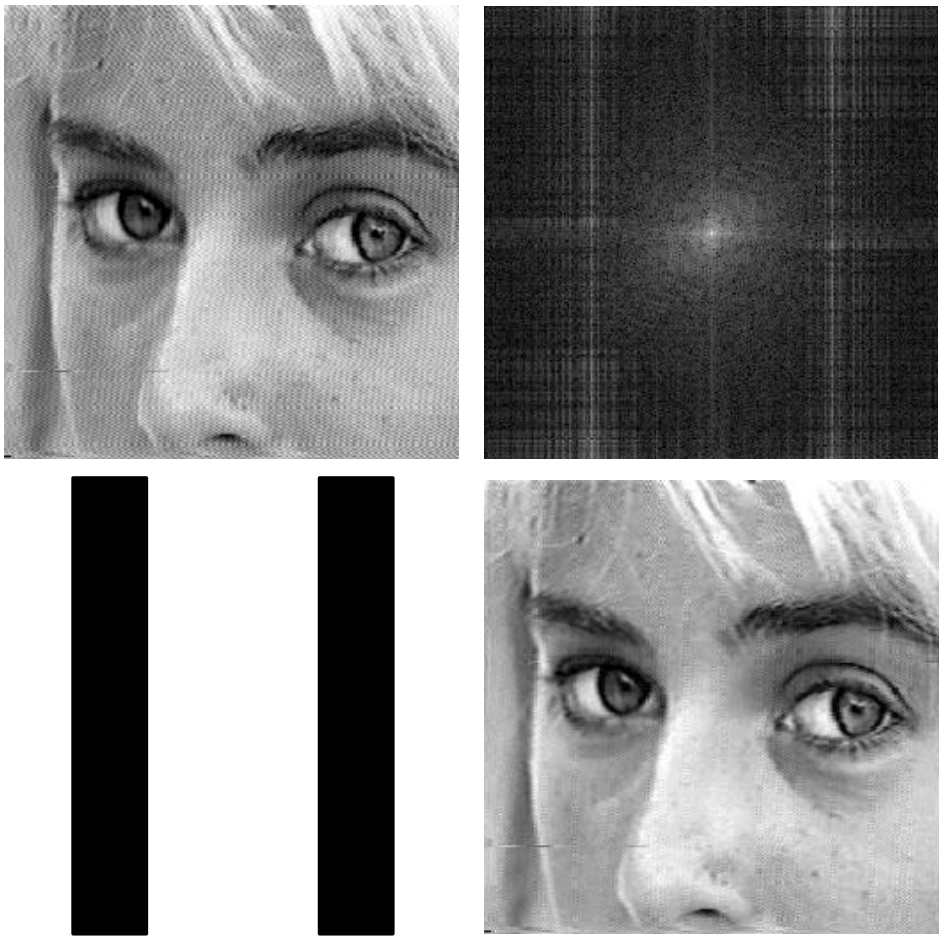

Figura 16-6 Filtragem de um Ruído com Periodicidade Horizontal

Esta imagem foi capturada de um digitalizador de sinal de vídeo quando a imagem no vídeo não estava bem posicionada gerando um ruído horizontal com uma periodicidade bem definida. A escolha do filtro é feita percebendo-se esta periodicidade e aplicando um filtro retangular vertical (*horizontal band pass*). A imagem resultante da filtragem é vista sem o ruído em questão.

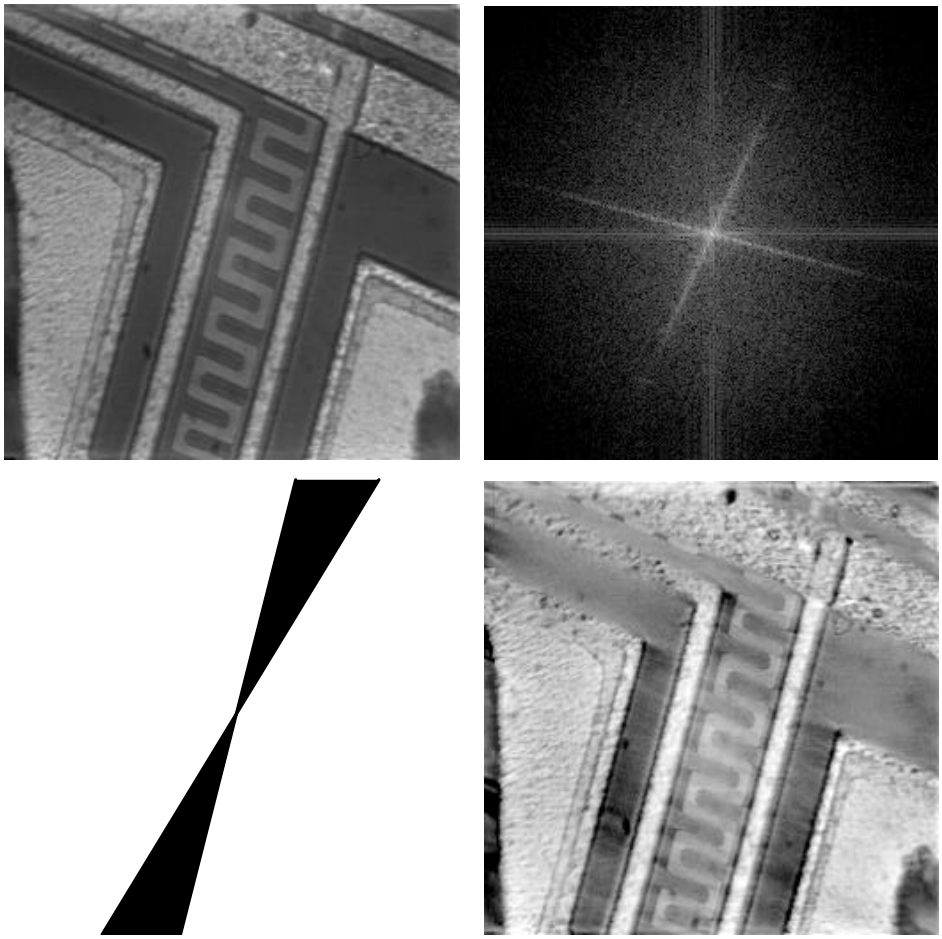

Figura 16-7 Filtragem de Linhas Diagonais em um Chip

A imagem do chip vista na figura possue inúmeras linhas diagonais em duas direções, que se manifestam no domínio da freqüência como o X visto na imagem da transformada. Cada um dos traços corresponde a um grupo de linhas diagonais. Isolando apenas um dos grupos e preservando o resto da imagem complexa, obtemos a imagem original sem aquele conjunto de linhas diagonais.

Esta filtragem pode ser usada para um deteção de bordas específica para linhas retas na imagem.

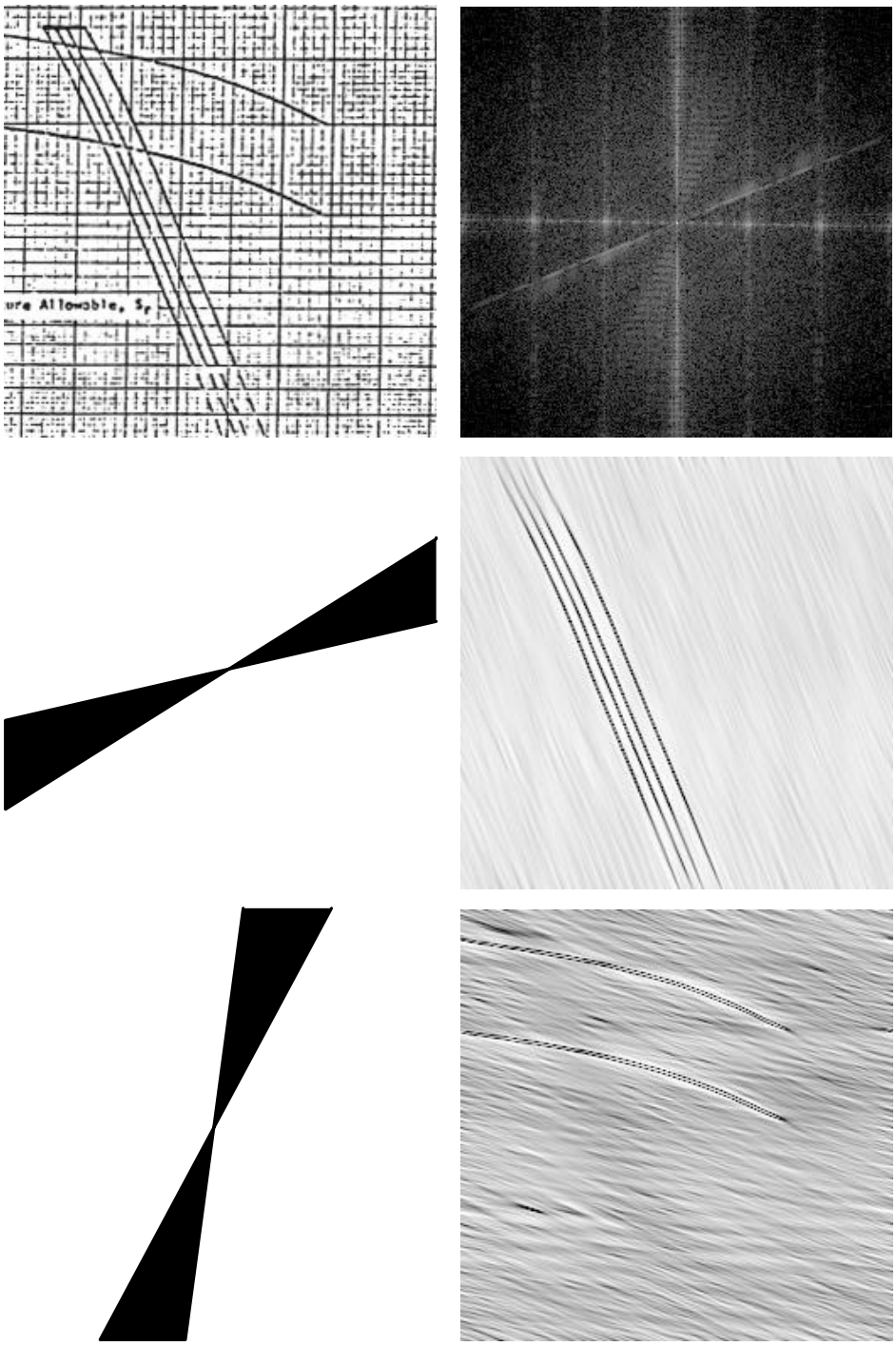

Figura 16-8 Filtragem de Curvas Sobre Grid Confuso.

A imagem inicial na seqüência da Figura 16-8 foi capturada através de um scanner e embora possua uma boa resolução a cópia original em papel não era bem definida gerando a imagem vista. O nosso interesse sobre esta imagem é isolar as duas classes de curvas que fornecem informações de cálculo, que uma vez no computador podem ser utilizadas por programas tornando o processo automático.

O grid, mesmo confuso, parece ter alguma propriedade periódica. Ao calcularmos a FT da imagem original observamos três manifestações distintas: linhas verticais que provavelmente provêem do grid, uma linha inclinada muito bem definida e uma região esparsa em um setor angular. Isolamos as duas últimas regiões e obtivemos o contorno das curvas desejadas.

A curva que possue tranformada uma linha inclinada bem definida, observando-se melhor, é muito semelhante a uma reta inclinada (uma linha diagonal), que como vimos na Figura 16-7 pode ser isolada. Já a outra curva se manifesta como se existissem diversas linhas diagonais[6] que combinadas formam a curva, por isso a formação da região esparsa em um setor angular.

[6] Pode-se até pensar em termos das derivadas ou das tangentes à curva em questão.

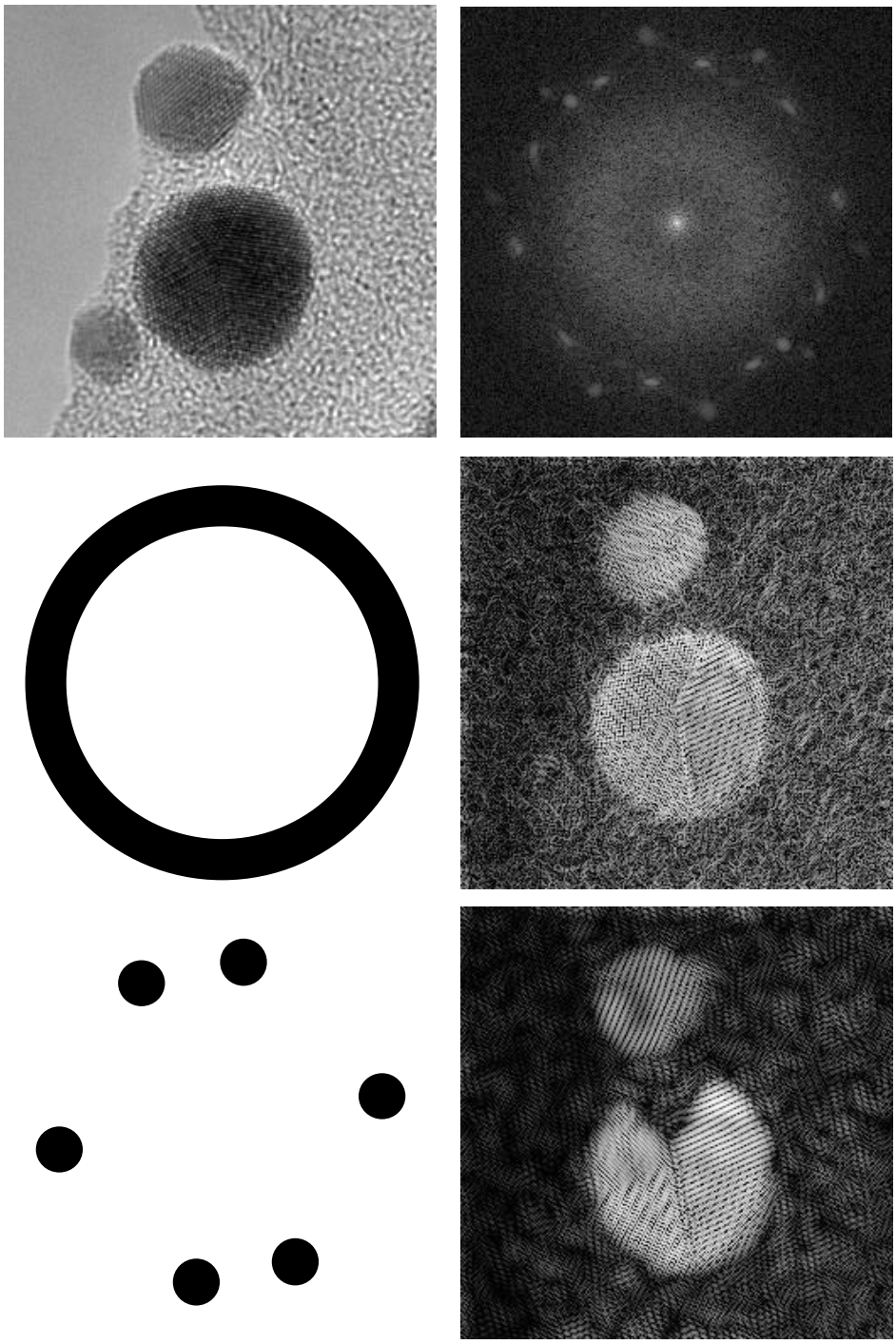

Figura 16-9 Filtragem de Átomos de Ouro sobre um Substrato Amorfo

A Figura 16-9 mostra uma seqüência de processamento de uma imagem de nano-partículas de ouro sobre um substrato de carbono amorfo. A estrutura cristalina é facilmente identificada na

imagem da transformada através dos pontos brilhantes de alta freqüência isolados da região de baixa freqüência.

Isolando todos os pontos vemos que o fundo amorfo praticamente desapareceu e que uma das partículas que não estava bem nítida também desapareceu.

Na segunda filtragem isolamos apenas os picos mais intensos e obtemos uma imagem semelhante mas com uma estrutura atômica bem diferente da anterior.

Isto nos indica que a escolha dos pontos a serem isolados é bastante crítica neste caso, devemos então tentar outras possibilidades até encontrar uma que realmente reflita a periodicidade desejada da imagem original.

Portanto, aparecerão casos em que a região a ser isolada na imagem da transformada não é trivial de ser encontrada, mesmo manifestando uma característica conhecida.

A imagem resultante, neste caso, pode ser utilizada para se reconstituir a estrutura atômica de uma parte da partícula. Recortando-se a imagem na região periódica de interesse e aplicando algoritmos específicos pode-se gerar padrões que reproduzam esta periodicidade. Escolhe-se visualmente o melhor padrão que se encaixa na imagem e com este padrão gera-se uma imagem sintética. Esta imagem é comparada com outras imagens sintéticas obtidas teoricamente. Isto permite definir melhor as propriedades das estruturas atômicas.

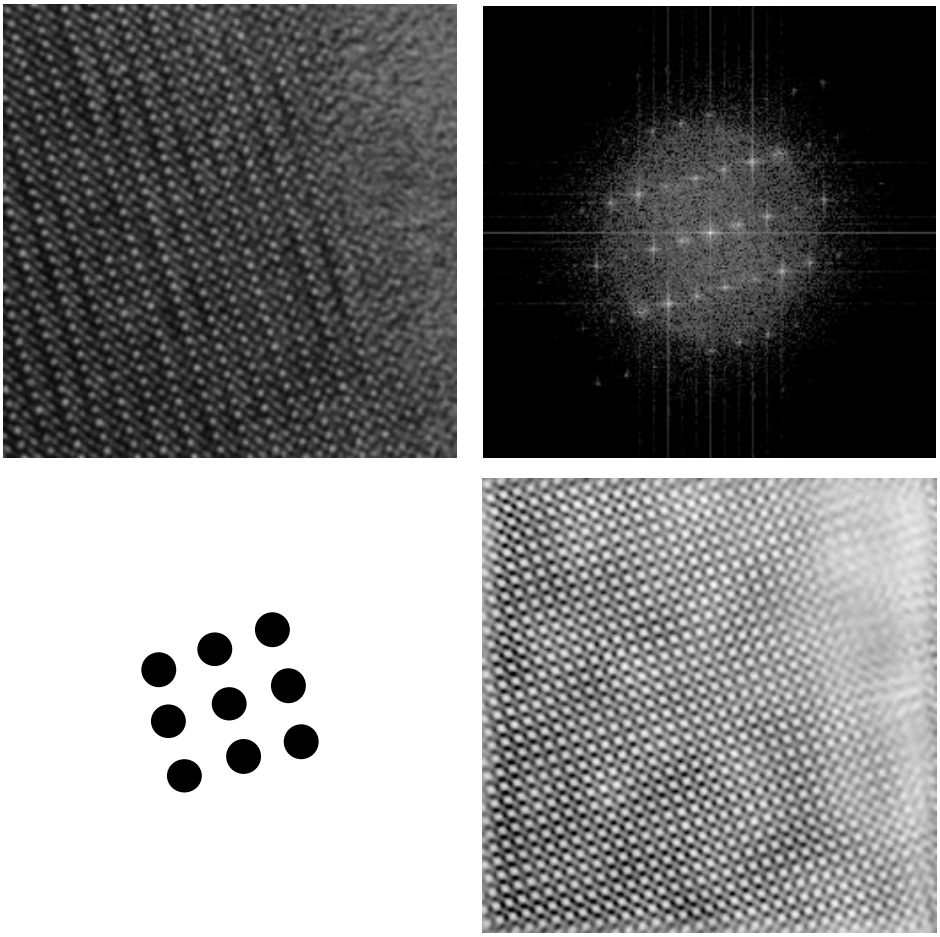

Figura 16-10 Filtragem de uma Estrutura Atômica

Este caso é semelhante ao anterior só que a estrutura pôde ser isolada com muito mais simplicidade. Foram utilizadas 5 máscaras *twin oval* sobre os picos mais intensos. A imagem resultante poderia ser melhorada se aplicassemos a *periodic*, pois estaríamos isolando um número maior de componentes que formam a estrutura periódica da imagem original.

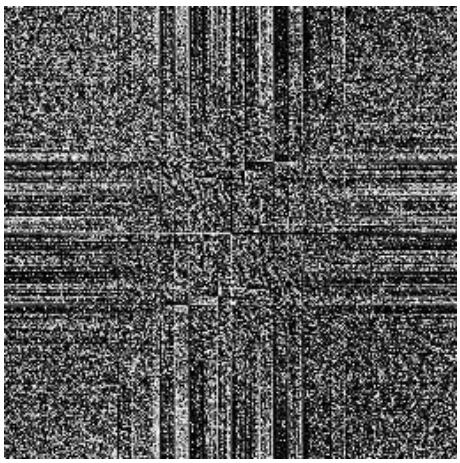

Figura 16-11 Visualização da Fase

É interesante mostrar aqui uma outra visualização da imagem da transformada da Figura 16-10, no caso a fase vista na Figura 16-11. Pois, é uma das pouca fases que possuem uma informação visual distinguível, embora ainda não saibamos como interpretá-la, convém mencioná-la. Poucos autores tentam fazer alguma análise sobre a fase da FT, pode se ver algo sobre o assunto em [CASTLEMAN 79].

17. CONCLUSÃO

Uma vez conhecidas as máscaras que seriam implementadas, o desenvolvimento deste trabalho decorreu em duas etapas.

Na primeira o SPID foi construído. Esta etapa foi naturalmente a mais longa e mais difícil levando em conta que deveríamos dar suporte para diversos tipos de imagem, acessar arquivos de imagem com padrão TIFF e criar uma metodologia de operação para imagens. Além disso, especificar uma inteface coerente com os diversos aplicativos para **Windows**, tornando seu uso o mais simples possível, dentro das qualidades e limitações deste ambiente.

Na segunda etapa foram efetivamente implementadas as máscaras, onde surgiram problemas específicos da criação das mesmas, mas que tomou um tempo muitas vezes menor.

Portanto, a criação do SPID foi muito bem aproveitada atingindo o seu objetivo de dar suporte para o desenvolvimento de aplicações em Processamento de Imagens. A partir de agora reduz-se a carga de desenvolvimento para criação de novas operações e simplifica-se a avaliação das mesmas.

O SPID foi testado em diversas máquinas do tipo IBM-PC compatível e obteve-se a Tabela 17-1, usando-se a FFT como padrão de medida. Os tempos incluem o fato de haver um contador no SPID, portanto, os tempos menores são mais influenciados pelo tempo de *display* da porcentagem de processamento e possuem menor precisão na tabela.

Imagem 256 Tons de Cinza	486 DX/2 66 MHz	486 DX 33 Mhz
128x128	0,7s	0,7s
256x256	1,5s	2,5s
512x512	5,5s	9,5s
1024x1024	25s	45s

Tabela 17-1 Teste de Performance Usando a FFT

Todos os tempos visto são bastante aceitáveis para um trabalho diário gerando inúmeras FFT’s com uma resposta ao usuário satisfatória.

No capítulo **8.** Implementação todas as imagens de transformadas foram obtidas com o SPID. Foram ainda realizados dois outros testes importantes: a IFFT(FFT(imagem original)) deve

ser igual a imagem original e aplicando-se esta seqüência repetidamente este fato deve-se manter. Nos dois testes o SPID se mostrou correto e no caso em que é aplicada uma máscara a utilização de uma borda suave se mostrou essencial para evitar artefatos.

Embora satisfaça nossas necessidades algumas melhorias imediatas ao SPID serão necessárias. Um melhor algoritmo de conversão de imagens coloridas de 24 bits para imagens coloridas indexadas com 256 cores para substituir o algoritmo atual que tende a perder cores escuras na imagem. O algoritmo de **Median Cut** proposto por Paul Heckbert é uma excelente sugestão, [HECKBERT 82].

Com o tratamento de grandes imagens a compressão do arquivo passa a ser necessária. O padrão TIFF 6.0 inclue além dos conhecidos LZW e RLE, o padrão JPEG para compressão de imagens.

Quando se possue um *hardware* gráfico com profundidade de cor limitada, no caso somente 256 cores, a utilização de uma palette comum para todas as imagens pode melhorar a qualidade da comparação entre imagens coloridas. Com o tempo este recurso deve cair em desuso.

O fato da FFT só aceitar imagens com determinados tamanhos, pede também a criação de imagens com uma cor de fundo para servirem de base para uma imagem a ser colocada sobre a mesma.

As anotações no SPID não podem ser movidas todas ao mesmo tempo e algumas possuem uma manipulação ainda difícil. Deve-se melhorar de maneira a facilitar o uso ao máximo.

Ainda faltam muitas operações para o SPID se tornar um sistema de Processamento de Imagens prático e independente. Este trabalho teve como objetivo maior demonstrar a potencialidade do SPID de se tornar um sistema completo. Portanto, existe uma trilha grande a ser percorrida, mas onde os primeiros passos já foram dados e uma direção clara e concisa foi tomada. Resta agora que outros tenham interesse na mesma área e utilizem a especificação do SPID [SCURI 94] para completá-lo e estendê-lo.

18. REFERÊNCIAS BIBLIOGRÁFICAS

[ALDUS 92] Aldus Corporation "TIFF™ Revision 6.0 Final - June 3, 1992" Aldus Developers Desk, 1992.

[ALEGRETTE 92] R. Alegrette "Pixie - Programa para Visualização de Imagens Digitais em Ambiente Microsoft® Windows™" Trabalho de Fim de Curso (graduação), Departamento de Engenharia de Computação, PUC-Rio, 1992.

[BORLAND 93.1] Borland International Inc. "ObjectWindows for C++ 2.0, Programmer's Guide" 1993.

[BORLAND 93.2] Borland International Inc. "Borland C++ for Windows 4.0, User's Guide" 1993.

[BORLAND 93.2] Borland International Inc. "Borland C++ for Windows 4.0, Programmer's Guide" 1993.

[BRIGHAM 74] E. O. Brigham "The Fast Fourier Transform" Englewood Cliffs, Prentice Hall, 1974.

[CASTLEMAN 79] K. R. Castleman "Digital Image Processing" Prentice-Hall, Inc., Englewood Cliffs, 1979.

[GALUCIO 90] E. G. Galucio, S. Paciornik e R. A. Nunes "Descrição do Sistema IMAGO para Processamento Digital de Imagens" Anais do II Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens, SIBGRAPI III, 1990.

[HECKBERT 82] P. Heckbert "Color Image Quantization for Frame Buffer Display" Computer Graphics, July, vol. 10, #3, 1982.

[O'NEILL 88] M. A. O'Neill "Faster Than Fast Fourier" BYTE, April págs. 293-300, 1988.

[SCURI 91] A. E. Scuri "Pinta - Utilitário Genérico para Apresentação de Imagens em Telas Gráficas" Trabalho de Fim de Curso (graduação), Departamento de Engenharia Elétrica, PUC-Rio, 1991.

[SCURI 92] A. E. Scuri "Sistemas de Processamento de Imagens" Estudo Orientado, Departamento de Informática, PUC-Rio, 1992.

[SCURI 93] A. E. Scuri “SPID - Conversor de Imagens Digitais” Trabalho de Final de Programação (mestrado), Departamento de Informática, PUC-Rio, 1993.

[SCURI 94] A. E. Scuri “SPID - Um Sistema de Processamento de Imagens - Manual Técnico” Documentação Interna, ICAD, PUC-Rio, 1994.

19. BIBLIOGRAFIA

[BALDNER 92] J. H. Baldner e D. C. Muchaluat “Processamento de Imagems Coloridas” Trabalho do Curso de Introdução ao Processamento de Images Digitais, D.C.M.M., PUC-Rio, 1992.

[BARKAKATI 92] N. Barkakati “Object Oriented Programming in C++” SAMS, 1991.

[BORLAND 92] Borland International Inc. “Help Compiler” 1992.

[CHAVES 91] E. O. C. Chaves “Multimídia - Conceituação, Aplicações e Tecnologia” People Computação Ltda., 1991.

[DERRAIK 92] A. L. B. Derraik e Y. N. Nasser “Compressão de Arquivos de Imagem” Trabalho do Curso de Introdução ao Processamento de Images Digitais, D.C.M.M., PUC-Rio, 1992.

[EZZELL 92] B. Ezzell “Windows 3.1 Graphics Programming” Ziff-Davis Press, 1992.

[FOLEY 90] J. Foley, A. van Dam, S. Feiner e J. Hughes “Computer Graphics: Principles and Practice” Addison-Wesley, 1990.

[GATTASS 92] M. Gattass e F. Celes “Computação Gráfica Interativa com o GKS” Departamento de Engenharia Civil, PUC-Rio, 1992.

[GONZALEZ 87] R. C. Gonzalez & P. Wintz “Digital Image Processing” Addison Wesley, 1987.

[HP 89] Hewlett Packard Co. “An Introduction to Computer Graphics” 1989.

[NORTON 92] P. Norton & P. Yao “Programando em Borland C++ para Windows” Berkeley, 1992.

[PACIORNIK 91] S. Paciornik “Introdução ao Processamento Digital de Imagens e suas Aplicações” D.C.M.M. PUC-Rio, 1991.

[PETZOLD 92] C. Petzold “Programando para Windows 3” Markon Books, 1992.

[PRESS 89] W. H. Press, B. P. Flannery, S. A. Teukolsky e W. T. Vetterling “Numerical Recipes - The Art of Scientific Computing” Cambrige University Press, 1989.

[PRATT 78] W. K. Pratt “Digital Image Processing” John Wiley & sons Inc, 1978.

[SCHEIER 93] B. Scheier “Color Models” Dr. Dobb’s Journal, #202 July, pág 38-43, 1993.

[SCURI 93] A. E. Scuri “Introdução ao Processamento de Imagens Digitais” Documentação Interna, ICAD, PUC-Rio, 1992.

[STROUSTRUP 86] B. Stroustrup “The C++ Programming Language” Addison-Wesley Publishing Company, 1986.

[STROUSTRUP 91] B. Stroustrup e M. A. Ellis “The Annotated C++ Reference Manual” Addison-Wesley Publishing Company, 1990.

Filtros Interativos para Imagens Digitais no Domínio da Freqüência

Dissertação de Mestrado apresentada por Antonio Escaño Scuri em 14 de Setembro de 1994 ao Departamento de Informática da PUC-Rio, e aprovada pela Comissão Julgadora, formada pelos seguintes professores:

prof. Bruno Feijó (Orientador)
Dept^{o} Informática / PUC-Rio

prof. Sidnei Paciornik (Co-orientador)
D.C.M.M. / PUC-Rio

prof. Marcelo Gattass
Dept^{o} Informática / PUC-Rio

prof. Roberto Ierusalimschy
Dept^{o} Informática / PUC-Rio

Visto e permitida a impressão.

Rio de Janeiro, ____ de _____ de 1994.

prof. Luiz Fernando Gomes Soares
Coordenador dos Programas de Pós-graduação
do Centro Técnico Científico