FourierStudio

Uma ferramenta para edição de imagens coloridas no domínio da frequência *

André Roberto Gouveia do Amaral Leitão
André Wilson Brotto Furtado
Fernando da Cunha Andrade Neto
Gustavo Danzi de Andrade
José Edson de Albuquerque Filho

ABSTRACT

Este documento tem como objetivo apresentar à comunidade científica uma nova ferramenta para a edição de imagens no domínio da frequência: o FourierStudio. Os autores deste trabalho acreditam que o uso desta ferramenta pode facilitar o desafio de editar imagens, automatizando parte do processo. Uma descrição comparativa de outras ferramentas da mesma área, assim como detalhes mais específicos da implementação do FourierStudio também são apresentados.

1. INTRODUÇÃO

O FourierStudio é uma ferramenta intuitiva para a aplicação de filtros no domínio da freqüência. Este processo de filtragem é bastante utilizado para eliminação de ruídos periódicos. Embora este tipo de filtragem seja bastante disseminada no meio acadêmico, poucas ferramentas oferecem esse serviço de uma maneira agradável. O FourierStudio visa preencher esta lacuna.

A base conceitual do FourierStudio é a Transformada de Fourier, proposta por Jean Baptiste Joseph Fourier. Fourier (21/03/1768 - 16/05/1830) viveu na França, na época da Revolução Francesa, e teve participação ativa nos acontecimentos políticos da época. Junto com a política, Fourier também teve grande participação nas pesquisas matemáticas, tanto na área pura quanto na aplicada.

A transformada de Fourier é uma das várias transformações que representam uma função de uma forma diferente. Esta transformação tem sido amplamente aplicada em diferentes áreas de pesquisa, como física quântica, ótica, e processamento de imagens. A sua aplicação no processamento de imagens está em descrever uma imagem em uma representação conveniente, modificá-la nesta representação e transformá-la de volta para a representação original.

Existem várias aplicações da transformada de Fourier em processamento de imagens, como filtragem, processamento de padrões, borramento e realce do contornos, e, mais recentemente, produção de marcas d'água em imagens.

2. ANTECEDENTES

A manipulação de imagens no domínio da freqüência é utilizada em muitos programas de edição gráfica, tendo-se mostrado um poderoso método de filtragem de imagens. Entretanto, como esta representação de imagens possui algumas propriedades de simetria, o editor da imagem no domínio da freqüência deve garantir que tais propriedade sejam preservadas durante a edição, o que dificulta a sua utilização em aplicativos convencionais.

Uma das ferramentas disponíveis para a edição de imagens no domínio da freqüência é o Image Studio, desenvolvido por Carlos André Cavalcante Pessoa, da Universidade Federal de Pernambuco, Centro de Informática. Esta ferramenta possui uma interface bastante intuitiva, acessível a qualquer usuário básico da área de processamento de imagens. O Image Studio também possui funcionalidades muito interessantes, como a possibilidade de calibragem da função de visualização pelo usuário e a realização de operações de adição e multiplicação sobre regiões do espectro. O aplicativo lida apenas com imagens no formato bitmap em tons de cinza, mas permite a gravação das imagens editadas e da imagem do espectro da transformada. O editor da ferramenta garante a simetria das operações realizadas, e permite a seleção de regiões livres; entretanto, todas as seleções têm o formato quadricular, não sendo possível a seleção de linhas ou círculos, por exemplo. Outra limitação da ferramenta é permitir apenas a visualização do espectro do domiínio da freqüência, não possibilitando visualização da fase.

Foi analisada tambem a ferramenta Quantum Image, da Aragon System. Concebida para processamento de imagens, ela possui diversos filtros e funcionalidades, entre elas a aplicação da Transformada de Fourier e de sua inversa. A ferramenta disponibiliza internamente um editor de espectro limitado, que apenas possibilita ao usuário zerar pontos do espectro, utilizando diversos tipos de pincéis (linhas, círculos e retângulos).

*Este trabalho foi desenvolvido para a disciplina de Computação Gráfica do curso de Ciências da Computação da Universidade Federal de Pernambuco. Maiores informações podem ser obtidas em *http://www.cin.ufpe.br/~awbf/fourierstudio*

Dentro do processamento de imagens, uma das grandes áreas de aplicação da transformada de Fourier é a filtragem de imagens obtidas por sensoriamento remoto, em virtude dos freqüentes ruídos periódicos encontrados nessas imagens. As técnicas de sensoriamento remoto permitem a obtenção de dados sobre todos os lugares do globo terrestre, através de sensores orbitais (satélites) ou fotográficos (aerotransportados), ou seja, sem que haja contato direto com os objetos ou fenômenos. Como os sensores ópticos podem inserir distorções no processo de geração de imagens digitais, torna-se necessário a utilização de técnicas eficientes para a eliminação de tais ruiídos.

No entanto, como a implementação e manipulação da transformada de Fourier possui algumas dificuldades, algumas ferramentas de sensoriamento remoto não oferecem esta funcionalidade.

Uma dessas ferramentas é o SPRING [12], Sistema de Processamento de Informações Georeferenciadas, desenvolvido pelo INPE (Instituto Nacional de Pesquisas Espaciais). Focada mais no geoprocessamento de dados, esta ferramenta possui excelentes recursos de armazenagem e manipulação de dados geográficos. Na área de processamento de imagens, a ferramenta oferece muitas opções de processamento, como filtragem espacial, operações aritméticas e eliminação de ruídos; porém nenhuma dessas técnicas se utiliza das propriedades da transformada de Fourier. A eliminação de ruído, por exemplo, é baseada apenas em limiares inferior e superior, isto é, um pixel é considerado ruído caso seu tom de cinza seja maior ou menor do que os níveis de cinza de seus vizinhos e a diferença seja maior do que os limiares escolhidos pelo usuário.

Outra importante ferramenta de sensoriamento remoto é o ENVI, The Environment for Visualizing Images, desenvolvido pela Research Systems, subsidiária da Eastman Kodak Company. Esta ferramenta possui um grande número de funcionalidades para a edição de imagens, incluindo a transformada de Fourier. O ENVI utiliza o algortimo FFT, Fast Fourier Transform, que otimiza o tempo de execução do cálculo da transformada. O ENVI permite a edição de imagens coloridas, podendo distinguir as operações entre cada espectro de cor. Outro conveniente da ferramenta é lidar com um grande número de formatos de imagens, incluindo imagens geradas por satélites. Por outro lado, o ENVI é uma ferramenta "burocrática", uma vez que o usuário precisa definir muitos parâmetros para efetivar a filtragem de uma imagem. O editor do ENVI, embora possibilite muitas opções de seleções, também não oferece uma interface amigável, dificultando sua utilização por usuários não-acadêmicos, isto é, para aplicações não-científicas. Por fim, o ENVI permite um conjunto limitado de operações sobre o espectro de imagens, não suportando somas e multiplicações de constantes.

3. FUNDAMENTOS TEÓRICOS DA FERRAMENTA

3.1 Imagem digital

Uma imagem digital pode ser considerada uma função real

$$f : S \longrightarrow \mathbb{R}$$

onde $S = [0, \cdots, N-1] \times [0, \cdots, M-1]$ é a grade ou matriz em que a imagem está definida. Geralmente, esta grade é um subconjunto de pixels do monitor do computador, e cada pixel desse subconjunto corresponde a uma posição na matriz. Embora tenhamos definido que a imagem de um elemento da matriz (ou pixel) é um valor real, na prática podemos restringir esse valor a um conjunto finito, que equivale às diversas variações de tonalidade que o monitor (ou outro dispositivo qualquer) pode representar. Para os monitores coloridos, é comum definir o contra-domínio de uma imagem como um vetor no $\mathbb{R}^3$ em que cada componente indica a tonalidade de uma das cores básicas: vermelho (R), verde (G), e azul (B). Cada componente pode assumir um valor inteiro, entre 0 e 255, onde 0 representa ausência do componente e 255 representa a máxima tonalidade do componente. Desta forma, a cor branca é representada por $(255, 255, 255)$ e a cor preta por $(0, 0, 0)$. Outras representações são possíveis, mas nos restringiremos a lidar com essa.

3.2 Transformada de Fourier

A base matemática do FourierStudio está na Transformada de Fourier. A transformada de Fourier pertence ao conjunto de transformações que utilizam funções como base para descrever outras funções. No caso de Fourier, esta base é composta por senos e cossenos.

Dada uma função $f : \mathbb{R} \longrightarrow \mathbb{R}$ qualquer, integrável, f pode ser escrita como uma série de senos e cossenos, na forma:

$$F(u) = \int_{-\infty}^{\infty} f(x) \exp(-j2\pi ux)dx \qquad (1)$$

A função F é denominada de *Transformada de Fourier da função f*.

Como pode ser visto na equação acima, a função F é complexa ($F : \mathbb{R} \longrightarrow \mathbb{C}$), isto é, seu resultado é um número complexo, mesmo que a entrada seja real. Uma das característica mais importantes da teoria de Fourier é a possibilidade de se recuperar f a partir de sua transformada F, através da relação:

$$f(x) = \int_{-\infty}^{\infty} F(u) \exp(j2\pi xu)du \qquad (2)$$

Está função chama-se *Transformada Inversa de Fourier da função F*.

Foi apresentada a Transformada Contínua de Fourier, mas para o processamento digital de imagens, utilizamos a *Transformada Discreta de Fourier*, em que as integrais das equações (1) e (2) são substituídas por somatórios sobre os dados amostrados.

3.3 Transformada Discreta de Fourier

Dada uma função $f : [0, \cdots, N-1] \longrightarrow \mathbb{R}$ qualquer, integrável, f pode ser escrita como uma série de senos e cossenos, na forma:

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \exp(\frac{-j2\pi ux}{N}) \qquad (3)$$

e $x \in [0, \cdots, N-1]$. A função inversa de F é

$$f(x) = \sum_{u=0}^{N-1} F(u) \exp(\frac{j2\pi xu}{N}) \quad (4)$$

e $x \in [0, \cdots, N-1]$.

De uma maneira menos formal, a transformada de Fourier calcula os valores da amplitude e da fase de várias funções seno e cosseno, que somadas resultam na função original. A quantidade de funções senos e cossenos, no caso da transformanda discreta, é determinada pelo número de amostragens obtidas.

Na prática, não se implementa a transformada de Fourier a partir de sua definição formal. Tal implementação possui complexidade de N^2, sendo bastante ineficiente no processamento de imagens, em que o número de amostragens é grande. A implementação mais comum é o algoritmo FFT (Fast Fourier Transform) que reduz o tempo de execução para a ordem de $N \log(N)$, aproveitando-se das equivalências de algumas operações com números complexos. O algortimo FFT utiliza a abordagem básica *dividir para conquistar*, que divide o problema original de tamanho N em dois problemas menores, e mais fáceis de resolver, de tamanho $\frac{N}{2}$

A base matemática fundamental do algoritmo FFT é a seguinte equação, obtida através da re-arrumação da definição da Transformada Discreta:

$$F(u) = \sum_{x=0}^{\frac{N}{2}-1} f_1(x) W_{\frac{N}{2}}^{m} k + W_N^k \sum_{x=0}^{\frac{N}{2}-1} f_2(x) W_{\frac{N}{2}}^{m} k \quad (5)$$

onde

$$W_N^n k = \exp -j\frac{2\pi nk}{N} \quad (6)$$

e f_1 são as amostragens pares e f_2 as amostragens ímpares do conjunto inicial:

$$f_1(x) = f(2x) f_2(x) = f(2x+1) \quad (7)$$

para $x \in [0, \cdots, N-1]$

Esta relação utiliza duas transformadas de tamanho $\frac{N}{2}$ que consomem menos tempo de computação do que uma transformada de tamanho N. Aplicando esta mesma relação para as transformadas resultandes (de tamanho $\frac{N}{2}$), obtemos o tempo de excecução de $N \log(N)$.

As funções citadas anteriormente são unidimensionais. Entretanto, uma imagem digital é uma função bidimensional. Poderíamos aplicar as transformadas de Fourier bidimensionais nas imagens, mas a melhor abordagem é aplicar o algoritmo FFT a todas as linhas da imagem, e aplicar novamente o FFT a cada coluna da nova representação da imagem gerada.

Para vizualizar a imagem resultante da aplicação da transformada de Fourier, isto é, uma matriz de números complexos, utilizaremos duas representações.

Para o número complexo $c = a + bi$, denominamos a a sua parte real e b sua parte imaginária.

Definimos o módulo, amplitude ou espectro do número complexo como:

$$\|c\| = \sqrt{a^2 + b^2}$$

Definimos a fase do número complexo como:

$$\theta(c) = \tan^{-1} \frac{b}{a}$$

4. EXPLORANDO O FOURIERSTUDIO

Como introduzido anteriormente, o FourierStudio é uma ferramenta para a edição de imagens coloridas no domínio da frequência, que automatiza o processo de eliminação de ruídos periódicos através da edição da imagem do espectro.

Inicialmente, o FourierStudio solicita o nome de um arquivo bitmap para iniciar a edição. O usuário pode escolher entre visualizar o espectro (módulo) ou a fase (amplitude) de cada número complexo resultante da aplicação da Transformada de Fourier.

A Edição da imagem pode ser feita de várias formas, usando retângulos, circulos, retas, linnhas e até mesmo ponto a ponto. O FourierStudio provê ainda um auxílio para detecção de ruídos. Esse auxílio permite uma seleção mais apurada de ruídos específicos.

As áreas de edição fornecem ao usuário a possibilidade não só de zerar componentes da imagem, mas atenuá-los ou destacá-los através da adição ou multiplicação por constantes.

Como a ferramenta oferece formas de edição separada das cores, um módulo de Ajuste de cores é disponibilizado ao usuário depois da aplicação da transformada inversa. O objetivo desse ajuste é permitir que o usuário possa voltar às cores originais da imagem, uma vez que componentes podem ser perdidos durante a edição.

Qualquer imagem pode ser gravada em arquivo para posterior visualização e um zoom é disponibilizado para facilitar a busca por ruídos.

5. IMPLEMENTAÇÃO

Dois itens serviram como alicerce para a implementação do FourierStudio. O primeiro deles, a linguagem de programação *C++*, não será abordada mais profundamente neste documento, pois não é desejada uma descrição de detalhes do código-fonte da aplicação. Entretanto, informações sobre a relevância desta linguagem para o desenvolvimento do FourierStudio podem ser encontradas na seção Dificuldades Encontradas.

Além da linguagem de programação *C++*, o desenvolvimento do FourerStudio foi viabilizado pelo uso de bibliotecas gráficas específicas desta mesma linguagem. A seguir, estas bibliotecas são descritas em maiores detalhes.

5.1 OpenGL

A OpenGL[9] (*Open Graphics Library*) é uma biblioteca gráfica que provê ao usuário diversas funcionalidades para a manipulação de primitivas gráficas no espaço, podendo ser utilizada também em aplicações bidimensionais, como é o caso do FourierStudio. Devido ao seu rico repertório de operações, a OpenGL é considerada por alguns como uma verdaderia *linguagem gráfica*. Entre algumas de suas vantagens, destacam-se independência de plataforma, estabilidade, facilidade de uso e, principalmente, o modo eficiente como a biblioteca implementa seus efeitos visuais, como renderização ou mapeamento de texturas.

Um dos principais motivos responsáveis pela popularidade da OpenGL é sua facilidade de aplicação em ferramentas que necessitam lidar com o espaço tridimensional. Muitas ferramentas CAD ou de realidade virtual, por exemplo, usufruem das funcionalidades oferecidas pela biblioteca.

Felizmente, muito do ferramental oferecido pela OpenGL para o gerenciamento de ambientes 3D pode ser aproveitado por aplicações que utilizam apenas o espaço bidimensional. Em primeiro lugar, a OpenGL disponibiliza operações que lidam diretamente com pixels, independentemente das dimensões espaciais. Além disso, o programador pode abstrair o fato de estar trabalhando num espaço tridimensional se sempre considerar que a coordenada de profundidade (Z) é zero [1].

Além de ser uma aplicação inserida no contexto bidimensional, o FourierStudio também é específico em outro aspecto: seu domínio de funcionalidades está basicamente restrito à manipulação de pixels. Dessa forma, o principal conjunto de recursos oferecidos pela OpenGL aproveitado pela ferramenta consiste em operações para visualização e edição de pixels.

A possibilidade de converter diretamente um array de bytes da memória em pixels na tela é uma funcionalidade da OpenGL bastante explorada no FourierStudio. Esta operação[2], além de garantir rapidez na vizualização de imagens, pode ser combinada com outras operações[3] para permitir que cada pixel seja multiplicado por uma escala e adicionado por um fator (ou bias) antes de ser exibido na tela. Isto é extremamente útil quando está armazenado em memória um array de bytes no formato *RGB*, mas se deseja visualizar apenas um dos três componentes. A OpenGL disponibiliza ainda efeitos de *blending*[4] (mixagem de cores) que permitiu a visualização de um determinado componente em tons de cinza.

Outro recurso bastante interessante provido pela OpenGL para a vizualização de pixels consiste em funções específicas para o mecanismo de *zoom*. Em alto nível, o programador pode especificar como o efeito de *zoom* será aplicado, definindo o fator do *zoom* e o conjunto alvo de pixels.[5] Este recurso foi amplamente explorado pelo FourierStudio: todas as janelas da ferramenta que apresentam alguma imagem oferecem-no. Isto é útil não apenas para conferir diferenças entre a imagem original e a imagem filtrada, como também para obter detalhes do espectro ou da fase resultantes da Transformada de Fourier.

Por fim, a OpenGL é uma biblioteca formidável para renderização dos mais variados tipos de primitivas, como linhas ou polígonos, o que é bastante desejável em ferramentas de edição de imagens como o FourierStudio. De fato, muitas das funcionalidades oferecidas pelo FourierStudio utilizam algum tipo de primitiva gráfica. A funcionalidade *Pintar* da ferramenta, por exemplo, permite que o "pincel" se apresente no formato de retangulo, círculo, linha ou "mão livre". Estas duas últimas opções nada mais são do que uma série de quadrados organizados de forma coerente para satisfazer o efeito desejado. A funcionalidade *Selecionar*, por sua vez, também está disponível na forma de círculos ou retângulos. Por fim, a funcionalidade *Zoom* necessita da renderização de linhas para que o usuário tenha uma noção da região na qual está sendo aplicado o *zoom*[6].

Infelizmente, alguns problemas foram encontrados em relação ao *raster* utilizado pela OpenGL. Um *raster* ou rasterizador, a grosso modo, pode ser considerado como uma entidade que define a posição, na tela, em que os pixels são desenhados. O *raster* da OpenGL é considerado inválido se o mesmo estiver associado a uma posição que esteja abaixo ou à esquerda do canto inferior esquerdo da janela corrente. Entretanto, em certas situações, é bastante interessante que o *raster* seja definido nessas condições, para que pixels de coordenadas maiores que as dimensões da janela possam ser desenhados na mesma[7]. A resposta para esta dificuldade foi encontrada no próprio *site* oficial da OpenGL, em que o problema é reconhecido e uma solução não trivial é apresentada[8].

5.2 GLUT

A OpenGL é uma ferramenta que gerencia a parte gráfica de uma aplicação, porém não define um mecanismo para gerenciar janelas e entradas do usuário. Desse modo, um *toolkit* auxiliar quase sempre é usado em conjunto com aplicações OpenGL, de modo a prover uma série de funcionalidades que a complemente.

Para esta tarefa, no FourierStudio, foi utilizado o GLUT (*Graphics Library Utility Toolkit*)[10]. Esta escolha se deveu ao fato de que a integração GLUT/OpenGL é bastante difundida na comunidade de programadores OpenGL, o que

[1] Esta pode ser uma alternativa interessante, visto que, ao simular um ambiente bidimensional a partir de um tridimensional em OpenGL, o programador ainda vai dispor de funcionalidades específicas da OpenGL para ambientes tridimensionais. Por exemplo, um *zoom* em uma imagem bidimensional pode ser obtido alterando a coordenada de profundidade (Z), recurso disponível apenas em 3D.

[2] implementada pela função *glDrawPixels*

[3] *glPixelTransferf*, *glPixelTransferfi*

[4] *glBlendFunc*

[5] *glPixelZoom*

[6] Na verdade, não se trata de um conjunto de linhas, e sim de um *loop* de linhas que terminam por formar um quadrado. Este recurso também é oferecido pela OpenGL

[7] Este problema, para ser melhor compreendido, exige um certo conhecimento do mecanismo de visualização da OpenGL. Recomendamos uma consulta à bibliografia para referências

[8] O site indica que é possível "enganar" a OpenGL sobre a posição corrente do *raster* através de chamadas à função *glBitmap*. Deste modo, mesmo sendo associado a uma posição abaixo e à esquerda do canto inferior esquerdo da janela corrente, o *raster* é considerado válido

não apenas a consolida como viável como também permite que um conjunto maior de referências sobre a mesma seja encontrado.

O gerenciamento de janelas realizado pelo GLUT é bastante explorado pelo FourierStudio. Infelizmente, o GLUT divide as janelas em apenas dois tipos: janelas principais e sub-janelas. Uma sub-janela possui tamanho e posição fixos em relação a uma janela principal, não permitindo que o usuário movimente livremente uma sub-janela qualquer. Além disso, a seleção de uma janela principal oculta todas as outras janelas principais que estejam em sua frente.

Estes dois fatores foram primordiais para decisão do projeto da interface gráfica do FourierStudio. Ao invés de haver uma janela central, responsável pelo controle e gerenciamento das demais janelas do programa, optou-se por um mecanismo de descentralização de janelas. Desse modo, não existe hierarquia alguma entre a janela principal e as janelas que englobam as imagens e demais mecanismos de interface com o usuário. Entretanto, cada imagem apresentada em uma janela principal está, na verdade, contida em uma sub-janela fixa, interna à janela principal correspondente. Alguns outros problemas específicos no gerenciamento de janelas podem ser encontrados na seção Dificuldades_Encontradas.

Em relação ao gerenciamento das entradas do usuário, o uso do GLUT se mostrou bastante oportuno. É importante deixar claro que, nesta seção, não estão sendo feitas referências à GUI (Graphic User Interface) da ferramenta, e sim às entradas do usuário aplicadas diretamente às imagens (como a seleção de uma área ou movimentação da lente de *zoom*, por exemplo). A implementação da GUI será devidamente detalhada na seção correspondente à biblioteca GLUI.

O *toolkit* GLUT possui diferentes funções de callback[9] disparadas de acordo com a interação do usuário com o programa. Apesar do alto volume de dados exigido por um programa de edição ao seu gerenciador de entradas do usuário, o GLUT se apresentou bastante estável. Entre os dois principais *callbacks* do GLUT utilizados encontram-se a detecção de cliques do mouse[10] e detecção de movimentação do mouse com o botão pressionado [11]. Juntamente com o *callback* que redesenha a janela [12], que não diz respeito ao processamento de entradas do usuário, esses dois *callbacks* foram os mais utilizados pelo FourierStudio.

5.3 GLUI

Apesar de prover uma série de mecanismos para lidar com entradas do usuário, o GLUT não oferece ao programador funções mais específicas para a interface gráfica (GUI) da aplicação. Como para uma ferramenta de edição de imagens é indispensável o uso de uma interface gráfica adequada, foi necessária a procura por opções de GUI compatíveis com OpenGL e GLUT. A biblioteca escolhida para esta tarefa foi a GLUI[11], que pode ser encarada como uma extensão do GLUT.

Uma das vantagens da GLUI é a compatibilidade total com o GLUT, permitindo uma fácil integração entre eles. A GLUI disponibiliza uma série de componentes de iterface gráfica, como *radio buttons*, *check boxes*, *listboxes* e botões simples.

Infelizmente, como será abordado em maiores detalhes no tópico Dificuldades_Encontradas, a GLUI não se apresentou como uma ferramenta muito madura, revelando problemas de eficiência.

5.4 API Gráfica do Windows

Ao invés de utilizar uma implementação própria para oferecer determinadas funcionalidades, como carregar um arquivo bitmap (.bmp) do disco, o FourierStudio utiliza implementações já consagradas pela API gráfica do Windows para estas operações. Esta escolha é interessante por uma série de fatores.

Em primeiro lugar, as soluções oferecidas por esta API estão disponibilizadas há um tempo bastante razoável, mostrando-se maduras o suficiente em termos de corretude e estabilidade. Além disso, o desenvolvimento do FourierStudio pôde-se adequar melhor ao cronograma proposto, visto que um esforço maior pode ser dispendido para a implementação de funcionalidades mais específicas da ferramenta, como a identificação de ruídos periódicos. Por fim, a API do Windows integra-se de maneira bastante intuitiva com a linguagem *C++*.

Uma desvantagem desta solução consiste no fato de que, obviamente, a portabilidade da ferramenta é prejudicada. Entretanto, o FourierStudio foi implementado de forma modular e extensível o suficiente para que futuras implementações substituam a integração atual com a API do Windows, ou até mesmo passem a coexistir com ela.

é interessante ressaltar que, embora amplamente disponíveis e difundidos, os recursos oferecidos pela API do Widows precisaram ser compreendidos e efetivamente assimilados, de forma a garantir que eles seriam capazes de prover o esperado para o FourierStudio.

6. DIFICULDADES ENCONTRADAS

Primeiramente, a necessidade de um entendimento conciso da teoria da Transformada de Fourier, o algoritmo da Transformada Rápida de Fourier e também sua aplicação no contexto de processamento de imagens demandou um bom tempo de estudo da equipe.

Outra dificuldade diz respeito à natureza da edição no domínio da frequência. É necessário um conhecimento teórico sobre o significado do espectro e da fase da imagem transformada, e como elas devem ser editadas para conseguir os efeitos desejados na imagem resultante da filtragem. Isto apresenta-se primeiramente como uma dificuldade ao usuário, que deve possuir um estudo antecedente em processamento de imagens usando a Transformada de Fourier. Apresenta-se também como um desafio ao desenvolvedor, que deseja uma interface que provenha métodos e ferramentas de edição que não só facilitem ao usuário o processo, como também permita que

[9] Funções de callback associam um evento, como uma entrada do mouse ou do teclado ativada, a uma determinada ação, como renderizar um retângulo na posição do clique, por exemplo.
[10] glutMouseFunc
[11] glutMotionFunc
[12] glutDisplayFunc

ele se utilize ao máximo das propriedades da Transformada de Fourier aplicadas ao processamento de imagens.

Essa dificuldade foi superada com um estudo comparativo de algumas ferramentas existentes, o que resultou no desenvolvimento de uma interface com o melhor que cada uma das ferramentas analisadas provia.

Outra dificuldade que pode ser destacada diz respeito à linguagem de programação e às bibliotecas utilizadas no projeto.

A implementação da ferramenta em *C++* implicou em certas dificuldades no desenvolvimento de uma interface gráfica agradável e intuitiva para o usuário.

A pouca familiariedade com OpenGL, e sobretudo com APIs GLUT e GLUI, resultou em um maior tempo gasto no estudo dessas APIs.

O uso da API GLUT não implicou em maiores problemas na implementação. Pode ser destacada apenas a dificuldade no gerenciamento do fechamento e liberação de recursos das varias janelas usadas pelo programa.

A biblioteca GLUI, escolhida como solução para prover controles, isto é, botões, *radio buttons*, *listboxes*, estrutura de painéis, entre outros componentes interessantes para a interface da ferramenta, não apresentou uma performance ótima na exibição de seus componentes, além de alguns problemas de integração com a API GLUT.

A utilização de outra linguagem de programação ou APIs que provessem mais facilidades para o *design* da GUI, como Java ou Delphi, certamente teriam se mostrado mais adequadas para o desenvolvimento do FourierStudio.

Outra dificuldade resultante do uso de *C++* no projeto foi o gerenciamento de memória, por esta linguagem não prover qualquer tipo de gerenciamento automático, como um *garbage collector*, por exemplo.

7. TRABALHOS FUTUROS

Vários pontos poderiam ser trabalhados para próximas versões do Fourier Studio.

Pode-se destacar como ponto fundamental para a evolução da ferramenta a otimização da função de sugestão de seleções. Seria dada ênfase ao aumento da velocidade e precisão do algoritmo usado, bem como um estudo de outros algoritmos que pudessem ser aplicados a essa funcionalidade.

A evolução na implementação desse algoritmo poderia resultar em uma contribuição interessante para conseguir realizar filtragens no domínio da frequência realmente automatizadas. Tal processo teria aplicaçõs práticas na filtragem de ruídos e na recepção de sinais de TV digital, por exemplo.

Além disso, a ferramenta poderia ser extendida para trabalhar com diversos formatos de arquivos. Primeiramente, ela seria extendida para ler e salvar arquivos em formatos usualmente usados por quem trabalha com processamento de imagens, como JPEG, GIF e TIFF, por exemplo. Futuramente, poderia ser estudada a extensão da ferramenta para suportar também formatos de arquivos usados em ferramentas de sensoriamento remoto, como ENVI[14], PCI Geomatics[15] e ERDAS[16].

Ainda no tocante à leitura e gravação de arquivos, poderia ser feita uma melhoria na implementação dessas funções, as desacoplando da API do Windows, o que aumentaria a portabilidade da ferramenta.

A edição da imagem transformada também poderia ser melhorada em futuras versões. Algumas melhorias possíveis:

- Maior integração entre a edição e o zoom, i.e., editar a imagem aproximada.
- Aprimoramento do Undo
- Implementação do Redo (desfazer ações do Undo)
- Selecionar áreas à mão livre

O gerenciamento e uso de memória também pode ainda ser otimizado. Por exemplo, um estudo poderia ser feito para minimizar a estrutura com que o Fourier Studio trabalha internamente para gerenciar a representação das várias imagens exibidas em dado instante pelo programa.

8. CONCLUSÃO

A versão 1.0 do FourierStudio foi concebida visando a facilidade de uso na filtragem de imagens com ruídos periódicos. Esse propósito foi bem explorado e satsfeito com o auxílio da GLUI, que possibilitou facilidades na interface com o usuário. Na versão atual, a ferramenta é capaz de editar a imagem em cores separadas e até sugerir eliminação de ruído. As múltiplas formas de seleção e edição de áreas já tornam o FourierStudio uma ferramenta bastante útil na área de processamento de imagens.

9. REFERENCES

[1] JHNE, B., *Digital Image Processing & Concepts*, Algorithms and Scientific Applications, Springer-Verlag, 1995

[2] LIM, S., *Two-dimensionasignal and image Processing*, Prentice Hall, 1990

[3] JAIN, K., *Fundamentals of Digital Image Processing*, Prentice Hall, 1989

[4] MYLER, H. R.,WEEKS, A. R., *Computer Imaging Recipes in C*, Prentice Hall, 1993

[5] TOMAZ, G., DóRIA R., CAVALCANTI G., FRERY A., *Exploring the Discrete Fourier Transform in User-Friendly Tool*

[6] FFT Tutorial : http://www.spd.eee.strath.ac.uk/ interact/fourier/

[7] Image Filtering in the Frequency Domain : http://astronomy.swin.edu.au/ pbourke/analysis/imagefilter/

[8] Biografia de Joseph Fourier: http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Fourier.html

[9] Página oficial OpenGL : http://www.opengl.org

[10] Especificação da GLUT 3 :
http://www.opengl.org/developers/documentation/glut/

[11] Manual da GLUI 2.0 :
http://www.cs.unc.edu/řademach/glui/glui_manual_v2_beta.pdf

[12] Camara G, Souza RCM, Freitas UM, Garrido J,
SPRING: Integrating remote sensing and GIS by object-oriented data modelling, Computers &
Graphics, 20: (3) 395-403, May-Jun 1996

[13] Página oficial do SPRING:
http://www.dpi.inpe.br/spring

[14] Página oficial do ENVI :
http://www.ResearchSystems.com/envi

[15] Página oficial do PCI Geomatics :
http://www.pcigeomatics.com/product_ind/product.html

[16] Página oficial do ERDAS :
http://www.erdas.com/home.asp