Gustavo Andre Jorge Rodrigues, Gustavo Balen Meneghel

Técnicas de Problemas Inversos (PI) aplicadas ao tratamento de imagens

Trabalho de formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de bacharel em Engenharia.

São Paulo
2006

Gustavo Andre Jorge Rodrigues, Gustavo Balen Meneghel

nota final 9,6
(nove e seis)

Técnicas de Problemas Inversos (PI) aplicadas ao tratamento de imagens

Trabalho de formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de bacharel em Engenharia.

Área de concentração:
Controle e automação

Orientador:
Prof. Dr. Alexandre Kawano

São Paulo
2006

Ficha Catalográfica

Técnicas de Problemas Inversos (PI) aplicadas ao tratamento de imagens. São Paulo, 2006. 38 p.

Trabalho de formatura — Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica.

1.Processamento de imagens 2.Equações do calor 3.Convoluções 4.Transformada de Fourier (Análise numérica) 5.Entropia (Matemática aplicada) I.Meneghel, Gustavo Balen - Rodrigues, Gustavo André Jorge II.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos III.t.

Resumo

Neste trabalho exploraremos algumas soluções para tratamento de imagens usando técnicas da área dos Problemas Inversos. Contrariamente ao que é estudado nos cursos clássicos de Engenharia, procuraremos investigar as causas, ou condições iniciais a partir de suas conseqüências. Concretamente, dada uma imagem obtida por algum processo, procuraremos melhorá-la a ponto de ser possível extrair algumas informações. Temos como objetivo direto a produção de programas de computador capazes de tratar uma imagem ou extrair informações de um arquivo fornecido. Indiretamente, objetivamos o aprendizado de técnicas matemáticas fundamentais usadas na solução de problemas inversos. O início deste trabalho consistiu na obtenção de conhecimento teórico e busca por material didático. A segunda parte consistiu no entendimento e discussão para uma filtragem do material, onde definimos as seguintes etapas: estudo da solução da equação da difusão, em sua forma como convolução contra um núcleo, fazendo uma analogia entre a difusão do calor e a possibilidade de dispersão de cores em uma imagem; estudo da FFT em duas dimensões e sua aplicação em equações de convolução; estudo do teorema da convolução inversa e sua aplicação no tratamento de imagens; estudo do método de Máxima Entropia, que se baseia no teorema de Bayes (que é usado na inferência estatística para atualizar estimativas da probabilidade de que diferentes hipóteses sejam verdadeiras, baseado nas observações e no conhecimento de como essas observações se relacionam com as hipóteses.). A terceira parte deste trabalho foi o desenvolvimento e implementação dos algoritmos estudados nas fases anteriores, e o presente desenvolvimento do software para a execução do método da Máxima Entropia (MME). Este método ocupa uma posição privilegiada entre os métodos de reconstrução de imagens, pois trabalha a partir de informações incompletas e com ruído. Ela é a única forma consistente de combinar diferentes informações em uma única imagem.

Palavras-chave: Tratamento de imagens, equação da difusão, convolução inversa, FFT

Sumário

Lista de Figuras

1 INTRODUÇÃO 2

2 MOTIVAÇÃO 3

3 OBJETIVOS 5

4 CRONOGRAMA 7

5 TRABALHO 8

5.1 Levantamento do material de estudo 8

5.2 Decisão dos softwares de apoio 12

5.3 Inicio dos testes utilizando Matlab 13

5.4 Análise de Fourier 13

5.4.1 Fundamentação teórica 13

5.4.2 Implementação desta teoria 15

5.4.3 Uma abordagem um pouco mais realista 18

5.5 Regularização 18

5.5.1 Fundamentação teórica 19

5.5.2 Implementação em software 24

5.6 Implementação do código em C 25

6 ZOOM DIGITAL 27

7 MÉTODO DA MÁXIMA ENTROPIA 28

7.1 Fundamentação teórica 29

7.2 Escolha do melhor método de solução 31
7.3 Implementação 33
7.4 Dificuldades 33

8 RESULTADOS **35**

9 CONCLUSÕES **37**

10 REFERÊNCIAS BIBLIOGRÁFICAS **38**

Lista de Figuras

3.1 Filtro Passa Baixa 6

3.2 Filtro Passa Alta 6

5.1 Borrando uma Imagem 13

5.2 Função filtro G (t variáveis) 16

5.3 Convoluindo imagens (t variáveis) 17

5.4 Imagem sem regularização 18

5.5 Imagens com regularização ($\delta variáveis$) 24

6.1 Foco na Gaiola 27

7.1 Imagem inicial e Imagem tratada mais escura 30

8.1 Borrando uma Imagem 35

8.2 Borrando uma Imagem 36

8.3 Borrada e volta com erros aleatóreos 36

1 INTRODUÇÃO

Neste projeto discutiremos técnicas básicas e introdutórias de processamento de imagens e quando for o caso exemplificando-as diretamente em uma imagem. Devemos lembrar que na prática o processamento de imagens é uma técnica extremamente dependente do problema que queremos resolver. Muitos dos procedimentos usados em processamento de imagens ou visão por computador podem ser caracterizados de rudimentares ou mesmo artesanais quando comparados ao complexo sistema visual humano, pois serão específicos para cada aplicação.

Normalmente as técnicas de processamento de imagens estão baseadas em métodos matemáticos que permitem descrever quantitativamente imagens das mais diversas origens. Uma imagem pode, de alguma forma, ser descrita independentemente do que ela representa e, a princípio todos os parâmetros que tem uma característica bidimensional ou topológico são convenientes. Em cada objeto definido em um espaço 2D nós podemos efetuar medidas de superfície, perímetros, comprimentos, espessura, posição, etc, e em seguida deduzir grandezas estatísticas de uma forma automática. É importante ressaltar que a análise automática é imprescindível quando queremos efetuar transformações sucessivas na imagem.

2 MOTIVAÇÃO

O Processamento de imagens é certamente uma área em crescimento. Diversos temas científicos são abordados e em alguns casos de caráter interdisciplinar. Entre eles podemos citar: a compreensão de imagens, a análise em multi-resolução e em multi-freqüência, a análise estatística, a codificação e a transmissão de imagens, etc. Mas o que faz do processamento de imagens uma disciplina tão particular e complexa ? O que faz com que não tenhamos ainda sistemas de alta performance de reconhecimento de caracteres ou de formas mais complexas ? O termo "Processamento de Imagens"vem na realidade do Processamento de Sinais. Os sinais, como as imagens, são na realidades um suporte físico que carrega no seu interior uma determinada informação. Esta informação pode estar associada a uma medida (neste caso falamos de um sinal em associação a um fenômeno físico), ou pode estar associada à um nível cognitivo (neste caso falamos de conhecimento). Processar uma imagem consiste em transformá-la sucessivamente com o objetivo de extrair mais facilmente os dados nela presente. Cabe neste momento fazer uma comparação entre o Processamento de Imagem e área de Computação Gráfica, técnica que encontramos freqüentemente aplicadas através de seqüências animadas na televisão ou em filmes de cinema. A Computação Gráfica parte de uma informação precisa para obter uma imagem ou um filme.

O Processamento de Imagens parte da imagem (de uma informação inicial que é geralmente captada por uma câmera) ou de uma seqüência de imagens para obtermos a "informação". Deste ponto de vista o Processamento de Imagens e a Computação Gráfica são exatamente métodos opostos, mas isto não quer dizer que as técnicas envolvidas em cada caso não possam ser as mesmas ou pelo menos complementares. É evidente que neste sentido processar uma imagem, como é feito pelo sistema visual humano (SVH), é extremamente complexo. Realizar as mesmas tarefas que o SVH, com a ajuda de máquinas, exige por antecedência uma compreensão "filosófica"do mundo ou dos conhecimentos humanos. Esta característica faz com que o processamento de imagens seja, atualmente, uma disciplina com extrema dependência do sistema no qual ele está associado, não

existindo no entanto uma solução única e abrangente para todos os problemas. Daí a não existência, até o momento, de sistemas de análise de imagens complexos e que funcionem para todos os casos.

A analise quantitativa e a interpretação de imagens representa atualmente um ponto de apoio importante em diversas disciplinas científicas. Tal é o caso por exemplo na ciência dos materiais, na biofísica, na medicina, na física da matéria condensada, etc. Na realidade a diversidade de aplicações do processamento de imagens, está associada diretamente a análise da informação que falamos acima. Pois em todas estas disciplinas estamos na realidade em busca de informações quantitativas que representem um fenômeno estudado.

Quando observamos do ponto de vista da ótica, uma imagem é um conjunto de pontos que convergem para formar um todo, mas podemos dizer de uma maneira mais ampla que uma imagem é o suporte para efetuarmos troca de informações. O termo imagem estava inicialmente associado ao domínio da luz visível, porém atualmente é muito freqüente ouvirmos falar de imagens quando uma grande quantidade de dados estão representados sob a forma bidimensional (por exemplo: as imagens acústicas, sísmicas, de satélites, infravermelhas, magnéticas etc). Os métodos recentes de exploração automática desta informação permitiu o desenvolvimento de técnicas complexas, que podem ser globalmente classificadas em duas grandes linhas. A primeira está associada a uma análise da informação e a segunda representa as técnicas que permitam obter uma melhoria (do termo em inglês "ENHANCEMENT") significativa da imagem.

3 OBJETIVOS

Ao final do trabalho, desejamos ter produzido programas de computador capazes de tratar uma imagem e assim extrair informações úteis para uma determinada aplicação. Dentro desse contexto temos diversas aplicabilidades possíveis, vou citar algumas: Síntese da Fala a partir do Movimento Facial, Processamento Digital de Imagens e Conservação-Restauração de Bens Culturais, Uma Estratégia de "Recozimento Simulado"(Simulated Annealing) para Detecção de Agrupamentos Espaciais de Formato Irregular, Recuperação de Informação com Base no Conteúdo Visual, Navegação Robótica Aérea Baseada em Visão - Requisitos de Processamento de Imagens para Projeto e Implementação, etc.

Indo além, nós vamos apenas melhorar uma imagem, mas como também "voltá-la no tempo", como foi desenvolvido no primeiro semestre.

Visamos o aprendizado de técnicas matemáticas fundamentais usadas na solução e tratamento de problemas inversos. Ainda nesse ideal, nosso projeto nos propiciará uma revisão e aprofundamento em ferramentas matemáticas que são de usos gerais.

Deixemos claro que nosso trabalho não é um caso de filtros que absorvem ruído e assim dizem fazer um tratamento de imagem. Estamos aqui propondo uma manipulação das informações contidas de forma a rearranjá-las chegando a uma imagem de maior aproveitamento da informação, sem perda do que estava contido inicialmente. Explicaremos então qual o princípio dos filtros: primeiramente fazemos a transformada bidimensional de Fourier da imagem, em seguida jogamos o centro ou as bordas dessa matriz fora (filtro passa-baixa ou passa-alta, zerando a outra porção), e por último fazemos a inversa da transformada para visualizar os efeitos:

Figura 3.1: Filtro Passa Baixa

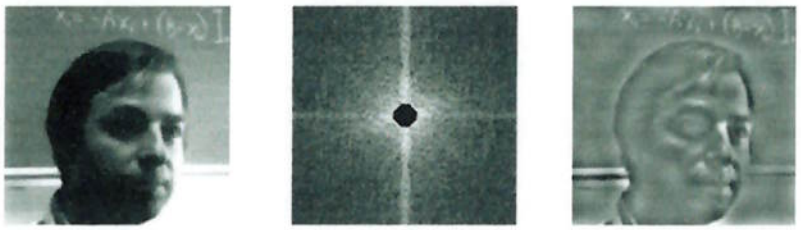


Figura 3.2: Filtro Passa Alta

No centro vemos a visualização das transformadas de Fourier, sendo que as partes negras representam o que foi apagado (filtrado) da imagem. A direita a inversa dessa transformada.

Podemos ver aqui nitidamente que estamos jogando parte da informação da nossa imagem fora, o que certamente não é um tratamento de imagem e sim um "aparamento"dela, como se estivéssemos cortando os cantos "feios".

4 CRONOGRAMA

ATIVIDADES 2º SEMESTRE 2006	Mês/Semana				
	Agosto	Setembro	Outubro	Novembro	Dezembro
Estudo do novo problema					
Coleta de material para estudo e trabalho					
Início dos teste com programas (MATLAB/excel)					
Desenv. de software p/ process. de imagens					
Testes e aplicações					
Documentação					

5 TRABALHO

O andamento dos trabalhos pode ser divido como a seguir:

5.1 Levantamento do material de estudo

Em todas as fases do nosso trabalho seguimos um planejamento base: primeiro levantamos material didático sobre os assuntos pretendidos, em seguida temos a análise e discussão para a filtragem, e finalmente delineamos os caminhos a serem tomados. O estudo inicial da problemática deste trabalho levou-nos a conclusão que a melhor forma para manipularmos uma imagem seria em forma de uma matriz de números onde cada ponto da matriz com sua respectiva coordenada representa um pixel da imagem. Isso posto, começamos a pesquisar os diferentes formatos e tipos de codificação (armazenamento) de imagens.

1. Tipos de dados

Descrevem o tipo de dados contidos num arquivo gráfico. Existem dois tipos: raster e vetor.

Raster: também conhecido como Bitmap. A informação gráfica é descrita como um conjunto de pixels, normalmente organizados da maneira como são vistos no monitor ou numa impressão: da esquerda para a direita e de cima para baixo. Cada pixel tem uma localização específica e uma cor atribuída a ele. Exemplos de programas que usam imagens bitmap: Photoshop, Photo Paint, Microsft Paint.

Vetor: a informação gráfica é descrita em termos de equações matemáticas ou de objetos discretos, tais como círculos, linha, retângulos, etc. Usado normalmente em programas de CAD e desenho. Gráficos do Adobe Illustrator são vetorizados.

2. Métodos de codificação de imagem

Também conhecidos como modos da imagem. Descrevem como os pixels são

organizados e representados dentro da memória do computador.

Preto e Branco (1 bit): também conhecido como Bitmap, pois cada pixel é representado por um único bit. Neste sistema, cada pixel pode assumir o valor 0 (preto) ou 1 (branco). Programas de manipulação costumam oferecer 4 modos de conversão para preto e branco: linha artística (50% Threshold), ordenado, difusão de erro e meio-tom. Dentre todos os modos, o Bitmap é o que resulta em imagem com menor tamanho. Este modo é usado para dar saída em duas cores.

Escala de Cinza (8 bits): sistema que usa 256 níveis de cinza por pixel, ou um byte por pixel. O valor 0 corresponde ao preto, e o valor 255 ao branco. No Photoshop, uma escala de cinza é representada como um porcentual de preto (K): 0 equivale ao branco e 100% equivale ao preto. Este modo é o recomendado para armazenar imagens em preto e branco guardando tons contínuos.

Cor Indexada (de 1 a 8 bits): também conhecido como 256 cores. Neste modo, cada pixel assume um valor presente numa paleta de 256 cores. Existem vários tipos de paletas. Programas de manipulação permitem customizar uma paleta e criar uma nova fazendo uma amostragem da imagem a ser convertida. Este modo é útil para aplicações multimídia e para publicar na Web.

RGB (24 bits): sistema que usa três cores por pixel, permitindo reproduzir até 16,7 milhões de cores. Cada cor é representada em 8 bits (1 byte), permitindo 256 níveis ou valores por cor. O valor (0, 0, 0) de R, G e B equivale ao preto, e o valor (255, 255, 255) de R, G e B equivale ao branco. Estas três cores são conhecidas como primárias; o sistema é baseado na combinação da luz emitida por três fontes de luz. Esta combinação é chamada aditiva.

3. Métodos de compressão de dados

São algoritmos matemáticos (concretizados na forma de programas) que visam reduzir o tamanho original de uma imagem usando alguma forma de codificação/ decodificação. Durante o processo de codificação, os valores dos pixels (R,G e B, por exemplo) são traduzidos para um código próprio ao método de compressão. Métodos de compressão podem ou não acarretar perda de informação (i.e., qualidade) da imagem.

LZW (Lempel-Ziv-Welch): foi desenvolvido em 1984 para compactar dados em discos magnéticos. Hoje em dia, métodos como este se tornaram populares pelo seu uso em programas de microcomputadores (Doublespace, Zip, etc).

Em 1987, a Compuserve criou o formato GIF e passou a usar o LZW como método de compressão. Em 1988, a Aldus lançou a versão 5 do formato TIFF,

onde compressão LZW era uma opção. A compressão LZW é obrigatória no formato GIF e uma opção do TIFF. A sua maior característica é não acarretar alguma perda de informação. Em contrapartida, nenhum método de compressão sem perdas permite um grau de compressão maior do que em torno de 2:1 (o JPEG pode atingir 100:1).

O LZW funciona muito bem com imagens gráficas, onde a quantidade de cores é discreta e onde existem muitas áreas com tons constantes.

JPEG (Joint Photographic Experts Group): foi criado em 1990 pelo comitê que deu o nome à este método de compressão. O JPEG foi projetado para comprimir imagens de sujeitos reais (tais como fotos), tanto coloridas quanto em escala de cinza. O JPEG tem como característica intrínseca a perda de qualidade da imagem, ou seja, uma imagem descomprimida não é exatamente igual à original. Por outro lado, permite taxas de compressão muito mais elevadas do que métodos sem perda. O JPEG permite armazenar imagens true color (24 ou 32 bits por pixel); o GIF, também muito usado na Internet, armazena apenas 8 bits/píxel. Uma imagem JPEG pode ser progressiva, como no formato GIF.

No JPEG, o grau de compressão pode ser controlado. Quanto mais compressão, menor o tamanho do arquivo. Porém, quanto maior a compressão, maior será a perda de informação. O JPEG é muito eficiente em imagem de tons contínuos, tais como fotografias, e menos eficiente em gráficos ou line art, onde a quantidade de tons diferentes é menor. O JPEG permite graus de compressão de 10:1 a 20:1 sem perdas visíveis na qualidade da imagem. Graus de compressão de 30:1 a 50:1 podem ser atingidos com perda moderada de qualidade, enquanto imagem com qualidade baixa podem ser geradas permitindo uma compressão de 100:1.

4. Formatos de arquivos

Quando imagens são capturadas eletronicamente por câmaras digitais, scanners, etc, ou geradas por programas, elas são sempre transferidas para um computador, onde ficam armazenadas em arquivos. Diferentes fabricantes de equipamentos digitais e programas de computador desenvolveram uma grande quantidade de formatos de arquivos. Os formatos de arquivos descrevem como as imagens são organizadas dentro do disco ou da memória do computador.

Existe um número muito grande de formatos de arquivos, e o fotógrafo deve conhecer ao menos superficialmente vários deles para poder trabalhar com programas de computador, com a Internet e transportá-los de um local à outro.

GIF (Graphics Interchange Format) É provavelmente o formato de arquivos gráficos mais popular. Foi criado pela Compuserve para a transmissão de imagens do tipo bitmap pela Internet. A primeira versão do GIF surgiu em 1987 (GIF87a). Em 1989 a Compuserve lançou a especificação GIF89a, que implementava o recurso da cor transparente.

Imagens GIF são sempre comprimidas e codificadas pela especificação LZW. A sua característica mais marcante é suportar apenas 8 bits por pixel, no máximo; se você necessitar de 24 ou 32 bits por pixel, use JPEG ou TIFF. Apesar desta limitação, o GIF ainda é o formato mais popular para armazenar imagens de baixa resolução. Outro aspecto importante é que o formato GIF embute um método de compressão (LZW) sem perda de informação: uma imagem GIF pode ser lida e gravada infinitas vezes e sempre será idêntica à original. Esta é uma vantagem do LZW sobre o JPEG, que sempre acarreta em perda de informação.

JFIF (JPEG File Interchange Format) Estritamente falando, o JPEG especifica apenas um método de compactação de imagens. O padrão JFIF foi criado para que os programas de manipulação possam ler trocar dados de um modo compatível.

O método de compressão usado pelo JPEG acarreta em perda da qualidade da imagem, ou seja, uma imagem comprimida em JPEG, quando descomprimida, nunca será igual à original. Por outro lado, o método de compressão do JPEG permite taxas de compressão mais altas do que o LZW (usado pelo GIF e pelo TIFF): usando um mecanismo que permite ao usuário alterar um nível de qualidade, a taxa de compressão do JPEG pode chegar à 20:1. Internamente, o JPEG usa um procedimento matemático complexo, que examina e agrupa blocos de 8x8 à 64x64 pixels, fazendo uma operação de média dos valores de cor.

TIFF (Tagged Image File Format) É um formato de arquivos que praticamente todos os programas de imagem aceitam. Foi desenvolvido em 1986 pela Aldus e pela Microsoft numa tentativa de criar um padrão para imagens geradas por equipamentos digital. O TIFF é capaz de armazenar imagens true color (24 ou 32 bits) e é um formato muito popular para transporte de imagens do desktop para bureaus, para saídas de scanners e separação de cores.

O TIFF permite que imagens sejam comprimidas usando o método LZW e permite salvar campos informativos (caption) dentro do arquivo. No Photoshop, use o comando File Info do menu File para preencher tais campos informativos.

BMP (Windows bitmap) É o formato gráfico nativo do Windows da Micro-

soft. É capaz de armazenar cores em até 24 bits, e muito popular em ambiente PC: alguns programas como o Paint aceitam somente este formato. Devido à popularidade do Windows, muitos programas, inclusive em Macintosh, suportam o formato BMP.

5. Formato e modo escolhido

Após esta dissecação do material a ser utilizado para desenvolvimento deste trabalho chegamos a conclusão que usaríamos o formato JPEG por ser de comum uso e por utilizar o modo RGB que facilitaria a implementação do nosso software.

5.2 Decisão dos softwares de apoio

Primeiramente pesquisamos possíveis ferramentas (softwares) que pudessem nos ajudar a desenvolver o nosso projeto. Nós precisávamos de visualização das imagens, também disponibilizar essas imagens em matrizes para aplicações matemáticas, e finalmente algumas ferramentas de apoio para verificar as implementações realizadas. Assim comparando dentre os softwares disponíveis chegamos a conclusão que o MATLAB seria a melhor escolha, e aquela que unifica todas as funcionalidades das quais iríamos precisar inicialmente.

O MATLAB foi desenvolvido no início da década de 80 por Cleve Moler, no Departamento de Ciência da Computação da Universidade do Novo México, EUA. É um "software"interativo de alta performance voltado para o cálculo numérico. Integra análise numérica, cálculo com matrizes, processamento de sinais e construção de gráficos em ambiente fácil de usar, onde problemas e soluções são expressos somente como eles são escritos matematicamente, ao contrário da programação tradicional.

O MATLAB possui ferramentas poderosas e amigáveis para a visualização de dados, bem de acordo com a filosofia de um laboratório de dados: fácil de experimentar e fácil de averiguar. O MATLAB permite também que matrizes sejam visualizadas como imagens. Também é possível ler imagens de arquivos tipo BMP, JPEG, TIF, GIF ... e guardar essa imagem em uma matriz, o que possibilita processamento e tratamento de imagens.

Por esses motivos escolhemos o programa para nos auxiliar durante o desenvolvimento do trabalho. Ele é de vital importância já que várias de nossas análises são de caráter subjetivo (referência humana) e ele facilmente mostra o resultado de uma matriz como imagem.

5.3 Inicio dos testes utilizando Matlab

Em várias partes do nosso trabalho tivemos que seguir o desenvolvimento de forma empírica, os seja, tentativa e erro/acerto. Dessa forma sempre que era iniciada uma etapa, testes eram feitos visando coleta de informações sobre o comportamento do resultado.

Assim, começamos a trabalhar com a idéia de "borrar"uma imagem e para tanto fizemos o nosso primeiro algoritmo que calculava a média ponderada entre cada pixel e seus oito pixels vizinhos, levando em conta as distâncias de cada pixel com o central, sendo que os das diagonais influenciam menos que os horizontais e verticais.

Podemos ver a seguir um exemplo real aplicado as colunas da Torre de Pizza:

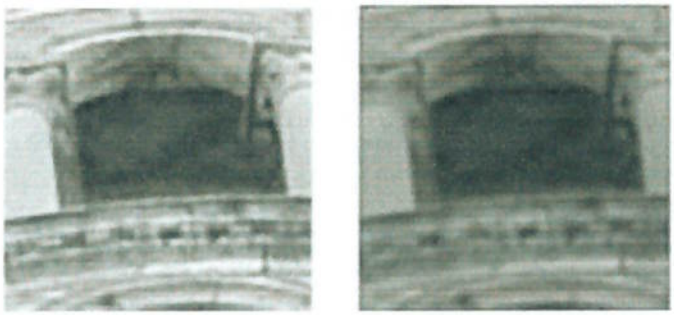


Figura 5.1: Borrando uma Imagem

Estes testes foram e continuam sendo de vital importância para o entendimento das teorias e possíveis modificações do trabalho, pois elucidam o que matematicamente seria muito difícil e pouco provável de se entender.

5.4 Análise de Fourier

A análise de Fourier é uma ferramenta matemática largamente usada tanto para Problemas Inversos, como para processamento de imagens, capaz de transformar uma distribuição de dados numéricos em um espectro de freqüências, e assim, podemos compreender a complexidade e comportamento de qualquer sinal.

5.4.1 Fundamentação teórica

Começamos relembrando a definição de transformada de Fourier de uma função:

$$f : \Re^2 \to \Re, f \in L^2(\Re^2) : \\ \mathrm{F}(f)(\xi) = \int f(\xi) \cdot \exp[-i.2.\pi.x \circ \xi] dx, \tag{5.1}$$

$\xi = (\xi_1, \xi_2)\, x = (x_1, x_2)$ em que $x \circ \xi$ é o produto escalar entre os vetores e .

Quando estamos em um domínio discreto (para facilitar quadrado) de lado 2N + 1, isto é, colocaremos a coordenada (0; 0) no centro do quadrado, e para cada direção contamos N casas, temos que fazer algumas alterações. A primeira é quanto às fronteiras. Lembrando que na transformada de Fourier as funções se estendem até o infinito, devemos também estender nosso domínio. Para tanto, colocaremos repetições da mesma imagem tanto para os lados como para cima e para baixo. Teremos então um "ladrilhamento" de imagens. O período da função da imagem será 2N +1 tanto na coordenada x1 como na coordenada x2. Vamos, então, precisar apenas de oito cópias, uma para cada posição com relação á imagem central, N, S, L, O, NE, NO, SE, SO.

Sabemos de cursos anteriores que quando a função é periódica, a transformada de Fourier é substituída pela transformação com a qual obtemos a série de Fourier. Nesse caso, já fazendo a adaptação para o nosso caso que é discreto, a transformação é:

$$\mathrm{F}(f)(\xi_1, \xi_2) = \sum_{x_1=-N}^{N} \sum_{x_2=-N}^{N} f(x_1, x_2) \cdot \exp\left[-i.\frac{2.\pi}{2.N+1}.(x_1, x_2) \circ (\xi_1, \xi_2)\right] \tag{5.2}$$

A transformada inversa é:

$$f(x_1, x_2) = \frac{1}{(2.N+1)^2} \cdot \sum_{\xi_1=-N}^{N} \sum_{\xi_2=-N}^{N} \mathrm{F}(f)(\xi_1, \xi_2) \cdot \exp\left[-i.\frac{2.\pi}{2.N+1}.(x_1, x_2) \circ (\xi_1, \xi_2)\right] \tag{5.3}$$

A imagem borrada é obtida por uma convolução:

$$\tilde{f} = f * g \tag{5.4}$$

em que a função filtro g é dada por

$$g(\xi_1, \xi_2) = \frac{1}{(2\sqrt{\pi.t})^2} \cdot e^{\frac{-|\xi_1, \xi_2|^2}{4.t} \cdot c} \tag{5.5}$$

Essa função pode ser facilmente discretizada quando colocada no centro de um quadrado, e conseqüentemente, teremos sua transformada de Fourier discreta. A

convolução $h = f * g$ entre duas funções discretas é calculada por

$$h(x_1, x_2) = \sum_{\xi_1=-N}^{N} \sum_{\xi_2=-N}^{N} f(x_1 - \xi_1, x_2 - \xi_2) \cdot g(\xi_1, \xi_2) \tag{5.6}$$

Observação 1.1. Aqui entra o ladrilhamento periódico. Quando fazemos a subtração de coordenadas, aparecem números para fora da imagem original.

A imagem borrada foi obtida com a convolução acima. Há um teorema que afirma que se $h = f * g$, então

$$F\tilde{f} = Ff * Fg$$

Assim, podemos obter a imagem original não borrada primeiro calculando e depois aplicando a transformada inversa, $F\left(\frac{F\tilde{f}}{Fg}\right)^{-1}$.

5.4.2 Implementação desta teoria

Esta foi uma parte crucial no desenvolvimento deste trabalho de graduação, pois envolveu não apenas a compreensão desta teoria matemática como também a compreensão das ferramentas, no caso Matlab, que poderíamos usar (vide tópico 3). A elaboração deste programa foi dividida em quatro partes, expostas a seguir.

I) Transformada de Fourier e Transformada inversa de Fourier de uma matriz

A primeira parte da implementação desta teoria, portanto, consistiu na elaboração de um software que lê uma imagem em formato jpeg e faz a TDF2D, Transformada Discreta de Fourier Bidimensional, ou 2D-DFT (em inglês). Esta transformada gera a matriz de freqüências da imagem lida. Posteriormente o software faz a TDF2D inversa com a matriz de freqüências gerada anteriormente, e o resultado como esperado tem que ser a matriz que representa a imagem nítida.

Exemplo do funcionamento do software com imagens de 5x5 pixels:

a) Matriz da imagem Original:

$$\begin{pmatrix} 0 & 16 & 12 & 140 & 188 \\ 68 & 86 & 128 & 152 & 192 \\ 150 & 12 & 182 & 56 & 172 \\ 122 & 138 & 122 & 148 & 16 \\ 80 & 112 & 40 & 52 & 52 \end{pmatrix}$$

b) Matriz da TDF2D da imagem original:

$$\begin{pmatrix} 0,2176-0,1053i & -0,3400+0,0286i & -0,1623-0,0280i & 0,0128-0,0445i & 0,2510+0,0002i \\ 0,0642+0,0301i & 0,4144+0,1019i & 0,3743-0,0878i & -0,2990+0,1918i & 0,5320-0,1821i \\ 0,0676-0,0821i & -0,0756+0,2926i & 2.436+0,0000i & -0,0756-0,2926 & 0,0676+0,0821i \\ 0,5320+0,1821i & -0,2990-0,1918i & 0,3743+0,0878i & 0,4144-0,1019i & 0,0642-0,0301i \\ 0,2510-0,0002i & 0,0128+0,0445i & -0,1623+0,0280i & -0,3400-0,0268i & 0,2176+0,1053i \end{pmatrix}$$

c) Matriz da TDF2D inversa da matriz anterior:

$$\begin{pmatrix} 0,0000 & 16,0000 & 12,0000 & 140,0000 & 188,0000 \\ 68,0000 & 86,0000 & 128,0000 & 152,0000 & 192,0000 \\ 150,0000 & 12,0000 & 182,0000 & 56,0000 & 172,0000 \\ 122,0000 & 138,0000 & 122,0000 & 148,0000 & 16,0000 \\ 80,0000 & 112,0000 & 40,0000 & 52,0000 & 52,0000 \end{pmatrix}$$

Portanto, verificamos que a matriz a) é igual a matriz c) atestando a funcionalidade deste software no cálculo da TFD2D e de sua inversa.

II) Geração da função filtro G em forma de matriz e respectiva TDF2D

Nesta etapa da implementação criamos uma matriz seguindo a Equação do Calor que utiliza as coordenadas da matriz para discretizar esta função filtro. A seguir podemos verificar a função filtro para uma imagem de 40x40 pixels com diferentes tempos t (variável que define a intensidade do "borramento") de difusão.

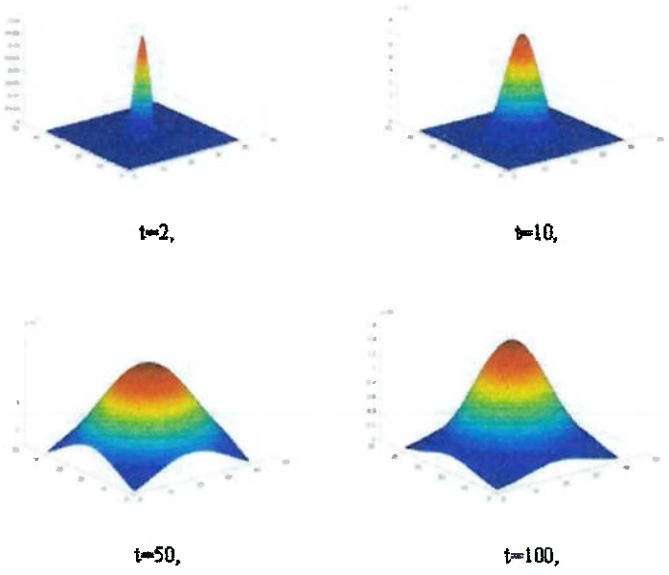


Figura 5.2: Função filtro G (t variáveis)

III) Convolução da função filtro G com a imagem nítida e respectiva TDF2D

Seguindo a fundamentação teórica, elaboramos a parte do software que é res-

ponsável pela convolução entre a função filtro G com a imagem nítida. Aqui observamos o "borramento" que segundo o item anterior, com o aumento de t temos que a imagem é cada vez mais borrada. A seguir podemos verificar a imagem resultante da convolução para uma imagem de 40x40 pixels com diferentes tempos de difusão:

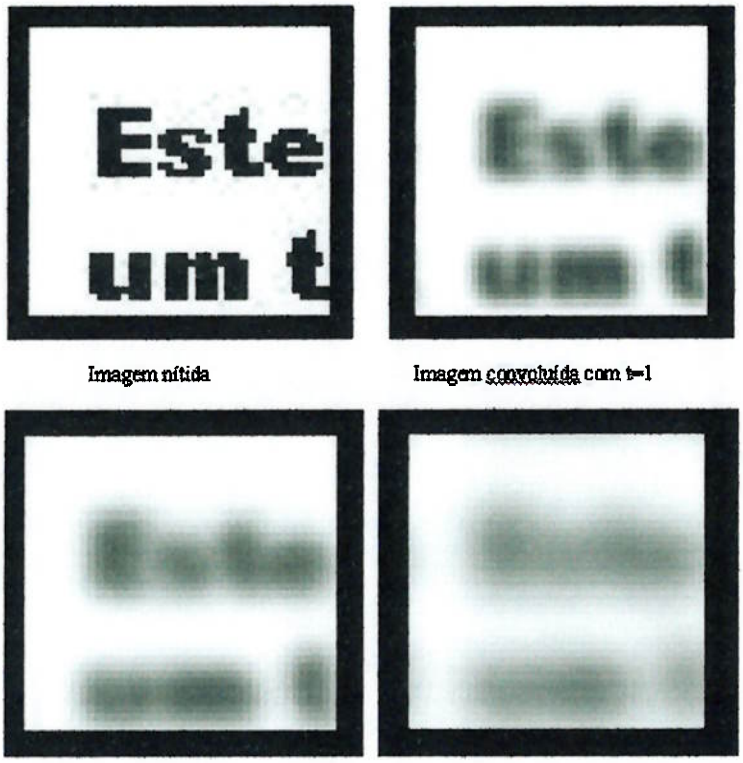


Figura 5.3: Convoluindo imagens (t variáveis)

Com a convolução funcionando o software aplica a TDF2D (item I) para que obtenhamos a transformada da imagem convoluída, e, o processo possa seguir adiante.

IV) Divisão entre TDF2D da convolução com a TDF2D da função filtro G e inversa

Nesta etapa, o programa inicia a divisão termo a termo entre a matriz da TDF2D da convolução com a TDF2D da função filtro G para que possamos obter a transformada da matriz nítida e finalmente efetuar a TDF2D inversa para assim obter com êxito os resultados utilizando as transformadas e provando que o software funciona.

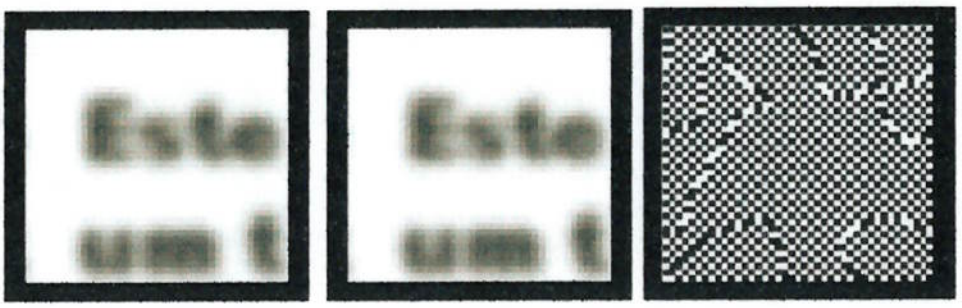
5.4.3 Uma abordagem um pouco mais realista

Sabemos que o escopo deste trabalho não é simplesmente provar que através de uma teoria matemática conseguimos borrar uma imagem e retorná-la para a imagem nítida mas sim melhorar imagens de forma geral, e também "voltar no tempo". Tendo uma imagem desconhecida, isto é, uma imagem que possua erros aleatórios e que tenha sido gerada por um filtro desconhecido vamos utilizar essa nossa "máquina do tempo".

O desenvolvimento de programas capazes de reconhecer ou, pelo menos, se aproximar do filtro real utilizado será escopo para o segundo semestre porém, considerando o filtro G conhecido, proveniente da equação do calor podemos supor erros aleatórios à imagem já borrada e assim verificar a capacidade de nosso software para tratar uma imagem parcialmente conhecida.

Para isso propusemos erros da ordem de +2 ou -2 aleatoriamente (ordem de 1% da grandeza usual, 0-255). A surpresa residiu no fato que um erro dessa magnitude pudesse fazer com que o software deixasse de funcionar. Após pesquisas e principalmente do apoio do nosso orientador, chegamos à conclusão que seria necessário entrarmos em uma nova área da Matemática conhecida como Teoria da Regularização.

A seguir a imagem convoluída com a adição de erros aleatórios e o resultado da tentativa de melhoramento da imagem:



(1) (2) (3)

1) Imagem convoluída; 2) Imagem convoluída com erros aleatórios; 3) Imagem da TDF2D inversa

Figura 5.4: Imagem sem regularização

5.5 Regularização

Como visto anteriormente, quando utilizamos nosso software para borrar uma imagem com um filtro conhecido, conseguimos voltar esta imagem borrada para

a imagem inicial, porém quando adicionamos um erro aleatório de aproximadamente 1% à imagem borrada, a inversa "estoura". Neste momento, se torna necessária a utilização de métodos matemáticos para evitar que tais comportamentos ocorram e consigamos retornar imagens às quais conhecemos os filtros utilizados para o "borramento" e desconhecemos os erros adicionados. Estes métodos fazem parte da Teoria da Regularização.

Portanto vamos compreender melhor a parte desta teoria que nos interessa.

5.5.1 Fundamentação teórica

Sabemos que x^{ϵ} em muitas situações, a equação: $A * x = y$ é instável, no sentido que o conhecimento de y^{ϵ} medido em torno de $Y_0 = A * x_0$, com $\|y_\epsilon - y_0\| < \epsilon$ fornece uma solução correspondente muito longe da resposta correta x^0, mesmo quando $A * x = y_0$ tem solução única.

Exemplo: Propagação da solução da equação do calor ou da difusão.

Uma maneira de estabilizarmos a solução X^{ϵ} é usando a regularização de Tikhonov.

Na formulação geral, A:x → Y, X e Y Hilbert, $B : D(B) \rightarrow$ z, z Hilbert, A, operador linear contínuo, $A(x) \notin Y$ densamente, B, operador linear fechado, são dados, $D(B) \subset x$. O $x^{\epsilon} \in x$ procurado é aquele que minimiza o funcional $F_t(x) = \|A * x - y^{\epsilon}\|_y^2 + t\|B * x\|_Z^2$; $x \in D(B)$ com t dado.

Obs: O objetivo é obtermos uma estimativa de X_0; sendo que $A * x_0 = y_0$, dado o conhecimento de y^{ϵ} (uma medida física) com $\|y_\epsilon - y_0\| < \epsilon$.

A solução existe e é única, se algumas condições estiverem satisfeitas. Veja que se existir solução, ela é única, pois se $x^{\epsilon,t}$ é $F_t(x^{\epsilon,t}) = \underbrace{inf}_{x \in D(B)} \left\{F_{t(x)}\right\}$, então

$$F_t(x) - F_t\left(x^{\epsilon,t}\right) = \left\|A * x - A * x^{\epsilon,t}\right\|^2 + t * \left\|B * x - B * x^{\epsilon,t}\right\|^2 \tag{5.7}$$

Isso significa que se temos 2 candidatos que realizam o mínimo:

$$F_t\left(x_1^{\epsilon,t}\right) = F_t\left(x^{\epsilon,t}\right) \tag{5.8}$$

então,

$$F_t\left(x_1^{\epsilon,t}\right) - F_t\left(x^{\epsilon,t}\right) = 0 \Rightarrow \left\|A * x_1^{\epsilon,t} - A * x^{\epsilon,t}\right\| = 0 \Rightarrow x_1^{\epsilon,t} = x^{\epsilon,t} \tag{5.9}$$

pois A é injetor.

Para provar que existe uma solução, temos que colocar uma hipótese:

Hipótese: Existe $C > 0$ tal que

$$\|x\|^2 \leq C * \left(\|A * x\|_y^2 + \|B * x\|_z^2\right)$$

, $\forall x \in D(B)$ (5.10)A questão é provar que o mínimo é atingido:

Seja $x_n \rightharpoonup x^{\epsilon,t}$, temos

$$\left\|A * x - A * x^{\epsilon,t}\right\|^2 + t * \left\|B * x - B * x^{\epsilon,t}\right\|^2 \to 0 \Rightarrow \left\{B * x_n \to B * x^{\epsilon,t}, x_n \rightharpoonup x^{\epsilon,t}\right\} \tag{5.11}$$

Mas B é fechado. Logo, $x^{\epsilon,t} \in D(B)$.

Uma escolha padrão para Bé a identidade.

Nesse caso, $I : x \rightharpoonup x \quad (D(B) = D(I) = x)$, I é fechado e vale a hipótese.

ma outra opção é escolhermos Ba identidade algébrica, mas que remete $x \in Z$.

$$I : x \rightharpoonup z = H'$$

é uma escolha comum na propagação inversa do calor.

Recuperação da nitidez da imagem

A imagem dita borrada é obtida pela operação:

$$y^0 = g * x_0,$$

onde y^0 é a imagem borrada e x^0 é a imagem nítida.

O que temos é uma imagem borrada corrompida por erros y^ε. Para fixar as idéias, podemos pensar que $y^\varepsilon = y^0 + \varepsilon \cdot a$, em que a é uma função com $\max |a| = 1$ aleatória.

Assim, $\|y^\varepsilon - y^0\|_Y \leq \varepsilon$.

Pela teoria da regularização vista acima, obteremos uma aproximação da imagem original pela minimização do funcional:

$$F_t(x) - F_t(x^{\epsilon,t}) = \|g * x - y^\varepsilon\|^2 + \delta \cdot \|B \cdot x\|_Z^2 \tag{5.12}$$

Adotemos $B = I$, a identidade $I : X \rightharpoonup X$

Como X e Y são Hilbert. Na verdade, $X = Y = L^2$.

$$F\delta(x) =< G.x - y^{\varepsilon}, G.x - y^{\varepsilon} > +\delta < x, x > \tag{5.13}$$

em que G é a operação $g*$.

Para buscar o mínimo de $F\delta$:

$$\frac{d}{dx}F\delta(x)(h) = 2 < G.x - y^{\varepsilon}, G.h > +\delta.2 < x, h >, \quad h \in x \tag{5.14}$$

$$= \frac{d}{dx}F\delta(x)(h) = 2 < G * G.x - G * y^{\varepsilon}, h > +\delta.2 < x, h > \tag{5.15}$$

No Ponto de mínimo, $\frac{d}{dx}F\delta(x)(h) = 0 \forall h \in x$.

Então, $G * G.x + \delta x = G * y^{\varepsilon}$,

$G*$, a adjunta de G, é dada por:

$< G * \phi, \psi >=< \phi, G\psi >$, isto é:

$$< \phi, G\psi >=< \phi, G * \psi >= \int_{-\infty}^{+\infty} \phi(x) \int_{-\infty}^{+\infty} g(x - y).\psi(y)dydx = \tag{5.16}$$

$$= \int_{-\infty}^{+\infty} \psi(y) \int_{-\infty}^{+\infty} \phi(x).g(x - y)dxdy = < G * \phi, \psi > \tag{5.17}$$

Portanto, $G * \phi = \int_{-\infty}^{+\infty} g(x - y).\phi(x)dx$

$G*$é a convolução com $\overset{v}{g}$

($g$com o sinal do argumento trocado, i.e, $\overset{v}{g}(x) = g(-x)$)

Voltando à equação que define o ponto de mínimo:

$$G * G.x + \delta x = G * y^{\varepsilon} \tag{5.18}$$

Substituindo $G*$ e G, temos :

$$\overset{v}{g} * g * x + \delta x = \overset{v}{g} * y^{\varepsilon} \tag{5.19}$$

Fazendo a transformada de Fourier:

$$\Im(\overset{v}{g}).\Im(g).\Im(x) + \delta.\Im(x) = \Im(\overset{v}{g}).\Im(y^{\varepsilon}) \tag{5.20}$$

Portanto:

$$\Im(x) = \frac{\Im(\overset{v}{g})}{\Im(\overset{v}{g})..\Im(g) + \delta}.\Im(y^{\varepsilon}) = \frac{\Im(\overset{v}{g}).\Im(g)}{\Im(\overset{v}{g}).\Im(g) + \delta}.\frac{\Im(y^{\varepsilon})}{\Im(g)} \tag{5.21}$$

Mas $\Im(\overset{v}{g})$ pode ser escrita em função de $\Im(g)$de uma forma bem interessante:

$$\Im(\overset{v}{g}) = \int_{-\infty}^{+\infty} \overset{v}{g}(x)e^{-i.x.w}dx = \int_{-\infty}^{+\infty} g(-x)e^{-i.x.w}dx == \int_{-\infty}^{+\infty} g(x)e^{+i.x.w}dx = \overline{\Im(g)} \tag{5.22}$$

Logo:

$$\Im(\overset{v}{g}).\Im(g) = \|\Im(g)\|^2 = \int_{-\infty}^{+\infty} |\Im(g)(w)|^2 \, dw \tag{5.23}$$

Então nossa formula de recuperação da imagem original será:

$$\Im(x) = \frac{\|\Im(g)\|^2}{\|\Im(g)\|^2 + \delta}.\frac{\Im(y^{\varepsilon})}{\Im(g)} \tag{5.24}$$

Porque antes estourava e agora não estoura mais? (ou pelo menos, temos a esperança de não mais?)

Resposta: A transformada de Fourier tem a propriedade que $\Im(f)(w) \overset{w\to\pm\infty}{\longrightarrow} 0$.

Isso pelo seguinte:

Supondo $f \in C_c^{\infty}$:

$$\left|(-i.w).\int_{-\infty}^{+\infty} f(x)e^{-i.x.w}dx\right| = \left|\int_{-\infty}^{+\infty} f(x)\left(\frac{d}{dx}\right)e^{-i.x.w}dx\right| =$$

$$= \left|-1.\int_{-\infty}^{+\infty}\left(\frac{d}{dx}\right)f(x)e^{-i.x.w}dx\right| = cte. \tag{5.25}$$

Portanto:

$$\left|\int_{-\infty}^{+\infty} f(x)e^{-i.x.w}dx\right| \leq \frac{cte.}{|w|} \Rightarrow \Im(f)(w) \overset{w\to\pm\infty}{\longrightarrow} 0 \tag{5.26}$$

Se $f \in C_c^{\infty}$, então o resultado vem fazendo uma aproximação de $f$por uma$\tilde{f} \in C_c^{\infty}$.

Voltando à questão do porque não estoura mais:

Antes, $\Im(x) = \frac{\Im(y^{\varepsilon})}{\Im(g)}$. Se a perturbação que nos fornece y^{ε} for tal que $\Im(y^{\varepsilon})(w) \overset{w\to\pm\infty}{\longrightarrow}$

0 decresce mais lentamente que o decrescimento de $\Im(g)(w) \overset{w\to\pm\infty}{\longrightarrow} 0$, $\Im(x) \longrightarrow +\infty$ e não existe função$x \in x$ que forneça uma imagem nítida.

Considerando agora o resultado que obtivemos:

$$\Im(x) = \frac{\|\Im(g)\|^2}{\|\Im(g)\|^2 + \delta} \cdot \frac{\Im(y^\varepsilon)}{\Im(g)} \tag{5.27}$$

Agora multiplicando $\frac{\Im(y^\varepsilon)}{\Im(g)}$, temos o fator $J = \frac{\|\Im(g)\|^2}{\|\Im(g)\|^2+\delta}$.

Note que $J \overset{w\to\pm\infty}{\longrightarrow} 0$, o que ajuda a fazer com que $\Im(x)$ fique bem comportada.

Outra opção de recuperação:

Podemos ainda aplicar a idéia de Tikhonov com $B = I : z \to z = H'$

$$I : L^2 \to H'$$

Assim o funcional $F\delta(x)$ fica:

$$F\delta(x) = \|g * x - y^\varepsilon\|^2 + \delta \cdot \left(\|x\|_x^2 + \|x'\|_x^2\right) \tag{5.28}$$

Fazendo a minimização do funcional:

$$\frac{d}{dx} F\delta(x)(h) = 2 < G.x - y^\varepsilon, G.h > +\delta.2\left(< x, h > - < x'', h >\right) \tag{5.29}$$

Forçando $\frac{d}{dx} F\delta(x)(h) = 0 \forall h \in x$, temos:

$$G * G.x - G * y^\varepsilon + \delta(x - x'') = 0, \tag{5.30}$$

$$G * G.x + \delta(x - x'') = G * y^\varepsilon$$

Fazendo a transformada de Fourier:

$$\left(\Im(\overset{v}{g}).\Im(g) + \delta.(1 + w^2)\right).\Im(x) = \Im(\overset{v}{g}).\Im(y^\varepsilon) \tag{5.31}$$

Portanto: $\Im(x) = \frac{\Im(\overset{v}{g})}{\Im(\overset{v}{g})..\Im(g)+\delta.(1+w^2)}.\Im(y^\varepsilon) =$

$$= \frac{\Im(\overset{v}{g}).\Im(g)}{\Im(\overset{v}{g}).\Im(g) + \delta.(1 + w^2)} \cdot \frac{\Im(y^\varepsilon)}{\Im(g)} = \frac{\|\Im(g)\|^2}{\|\Im(g)\|^2 + \delta.(1 + w^2)} \cdot \frac{\Im(y^\varepsilon)}{\Im(g)} \tag{5.32}$$

5.5.2 Implementação em software

Aparentemente estas mudanças podem apresentar certa complexidade porem a sua implementação foi simples considerando que já havíamos desenvolvido todo o software. A mudança se caracterizou apenas na inclusão de um fator na divisão entre os elementos da matriz de BDTF da imagem borrada e os respectivos elementos a matriz da BDTF do filtro, segundo suas coordenadas. Este fator é simplesmente um número, obtido pela divisão da norma ao quadrado da BDTF do filtro em cada ponto pela norma ao quadrado da BDTF do filtro acrescido de um ?, variável determinante do ponto ótimo nesta regularização. Em uma outra alternativa multiplicamos este ? por (1 + w2), onde w é a posição de cada elemento.

Resultados:

Como esperado, com a multiplicação da divisão pelo fator J(x1,x2, ?) onde ? é a variável que determina o ponto ótimo de resolução da imagem , conseguimos retornar uma imagem borrada por um filtro conhecido porém com a adição de um erro aleatório desconhecido.

A seguir podemos ver diferentes resoluções obtidas com a variação deste ?:

Figura 5.5: Imagens com regularização (*δvariáveis*)

Assim podemos verificar que para uma imagem de 40x40 pixels utilizando a equação do calor como filtro em um tempo de borramento, t=1 o ponto ótimo ocorre para ? = 0.001, isto é, para um ? muito pequeno 10E-6 a fórmula praticamente se cancela e a inversa ocorre como se não houvesse regularização. Após o ponto ótimo, para um ? maior do que 0.001 temos que a imagem não "estoura"porém inicia um novo borramento. Portanto o software está funcionando mesmo com a adição de erros aleatórios.

5.6 Implementação do código em C

A linguagem C foi criada por Dennis Ritchie, em 1972, no centro de Pesquisas da Bell Laboratories. Sua primeira utilização importante foi a reescrita do Sistema Operacional UNIX, que até então era escrito em assembly.

Em meados de 1970 o UNIX saiu do laboratório para ser liberado para as universidades. Foi o suficiente para que o sucesso da linguagem atingisse proporções tais que, por volta de 1980, já existiam várias versões de compiladores C oferecidas por várias empresas, não sendo mais restritas apenas ao ambiente UNIX, porém compatíveis com vários outros sistemas operacionais. O C é uma linguagem de propósito geral, sendo adequada à programação estruturada. Ela pertence a uma família de linguagens cujas características são: portabilidade, modularidade, compilação separada, recursos de baixo nível, geração de código eficiente, confiabilidade, regularidade, simplicidade e facilidade de uso.

Sendo assim tínhamos motivos mais que suficientes para querer ter uma versão do nosso software nessa linguagem, já que então deixaríamos o MATLAB de uso particular e seguiríamos para o C de livre acesso ao público.

Escolhemos a plataforma do Eclipse para rodar os programas em C, por sua interface amigável e um incrível gerenciamento das informações referentes ao código, e ainda por ser um programa livre (não necessita de licença). O compilador usado foi o Mingw 5.0.2 (gcc) de grande aceitação e fácil aquisição, que proporcionava as bibliotecas que iríamos utilizar (inclusive a de álgebra complexa).

As diferenças do versão para C em relação ao MATLAB são basicamente:

1-) Temos que fornecer um arquivo .txt que contenha a matriz que representa a imagem com suas dimensões.

2-) Não podemos visualizar as imagens geradas já que este é um recurso de

mais alto nível e não pretendemos implementá-lo em C.

3-) Para implementações mais complexas o MATLAB começa a ficar com o processamento lento pois faz uso de linguagem interpretada e assim desenvolvendo em C, linguagem compilada, conseguimos esses mesmos resultados em menor tempo, o que traz inúmeros benefícios.

6 ZOOM DIGITAL

Uma das propostas do primeiro semestre fora o desenvolvimento de um mecanismo para Zoom Digital. Vamos aqui esclarecer que nosso intuito foi de nos focarmos apenas no software que tivesse um meio para nos dizer o que é uma imagem nítida.

Assim, com base nos softwares anteriormente desenvolvidos e técnicas aprendidas, fizemos um algoritmo que compara a imagem atual com base na que já estava na memória, e então decide se deve aproximar ou retroceder o Zoom. A comparação é simples e feita através de um único resultado: o algoritmo percorre a matriz dos pixeis da imagem e faz uma subtração entre o pixel (i, j) onde está e cada um de seus 8 vizinhos, somando os resultados. O resultado maior indica qual imagem é a melhor. Porque? Pois se os pixeis vizinhos possuem uma menor diferença que a imagem da memória, significa que essa imagem está "borrada"(pois quando borramos totalmente uma imagem ela tende ao cinza, ou seja, todos os pixeis iguais).

Também podemos fazer o Zoom Digital funcionar para regiões e não para a matriz todo, isso é útil quando queremos focar em um objeto e não em toda a imagem que estamos vendo. Assim o algoritmo muda e ao invés de percorrer toda a matriz somente percorre a região específica.

Figura 6.1: Foco na Gaiola

7 MÉTODO DA MÁXIMA ENTROPIA

O método da Máxima Entropia tem provado ser de grande importância em reconstrução de imagens a partir de tipos diferentes de informações. Ele tem sido usado de maneira intensa na astronômia, onde lida rotineiramente com imagens de milhões de pixeis. Ele também tem ajudado em deconvoluções ópticas e reconstruções de tomografias.

Seu desenvolvimento é justificado pela inadequação de outros métodos, como Fourier, que são inadequados para tratar alguns tipos de imagem, mas este método nos permite incorporar informações extras e utilizar uma suposição do que aconteceu com a imagem.

Este método é baseado no teorema que só existe um meio de combinar diferentes dados numa única imagem de modo positivo. Ele sempre nos retorna a mais segura imagem possível, que possui estrutura suficiente para acomodar os seus dados.

O custo de se usar a Máxima Entropia é computacional: por volta de 100 vezes o tempo que um CPU usaria para um filtro linear. Mas a técnica é totalmente praticável em aplicações onde podemos despender alguns minutos.

Com dados lineares, este método nos retorna uma imagem única. Porém dados não-lineares podem ser ambíguos, e assim podemos ter mais de um máximo local de entropia. Em casos como este devemos introduzir informações extras.

Comumente, ele nos permite que recalibremos parâmetros instrumentais automaticamente para valores que nos direcionem a uma resposta. Em um caso extremo o método nos dá esperança em problemas de deconvolução cega.

Para que possamos realizar uma deconvolução cega, é necessário buscarmos métodos estatísticos. No problema em questão foi escolhido o método de Bayes para que seja possível encontrar uma imagem mais nítida.

7.1 Fundamentação teórica

Partimos de uma relação de probabilidade muito simples e conhecida na estatística:

$$P(A \cap B) = P(A/B) * P(B) = P(B/A) * P(A) \tag{7.1}$$

Esta regra nos diz que a probabilidade da intersecção entre o evento A e o evento B pode ser descrita pela probabilidade de ocorrer o evento A dado que conhecemos o evento B, multiplicada pela probabilidade do evento B ocorrer, e isto é igual à probabilidade do evento B ocorrer dado que conhecemos A, multiplicada pela probabilidade do evento B ocorrer.

No nosso caso, o evento A é encontrarmos a imagem mais nítida possível, e o evento B é a imagem borrada. Isto é, nosso objetivo, portanto, é encontrar uma imagem nítida dado que conhecemos a imagem borrada.

1^0 Passo:

Assim temos que A=$(\mu_1, \mu_2, ..., \mu_n)$ e B= $(x_1, x_2, ..., x_n)$. Aplicado a regra acima temos: $g(\mu_1, \mu_2, ..., \mu_n | x_1, x_2, ..., x_n) = f(x_1, x_2, ..., x_n | \mu_1, \mu_2, ..., \mu_n) * f(\mu_1, \mu_2, ..., \mu_n)$ onde:$f(x_1, x_2, ..., x_n | \mu_1, \mu_2, ..., \mu_n) * f(\mu_1, \mu_2, ..., \mu_n) = f(x|\mu) * f(\mu)$ e,

$$f(x|\mu) * f(\mu) = \left\{ \prod_{i=1}^{n} \left(\frac{1}{\sqrt{2.\pi}.\sigma} . \exp\left(-\frac{1}{2} . \left(\frac{x_i - \mu_i}{\sigma} \right)^2 \right) \right) \right\} * \prod_{i=1}^{n} (. \exp(-\mu_i . \ln(\mu_i))) \tag{7.2}$$

que é igual a:

$$f(x_1, x_2, ..., x_n | \mu_1, \mu_2, ..., \mu_n) * f(\mu_1, \mu_2, ..., \mu_n) = \tag{7.3}$$

$$= \prod_{i=1}^{n} \left(\frac{1}{\sqrt{2.\pi}.\sigma} . \exp\left(-\frac{1}{2} . \frac{x_i - \mu_i}{\sigma} - \mu_i . \ln(\mu_i) \right) \right) \tag{7.4}$$

que é o mesmo que:

$$f(x|\mu) * f(\mu) = \exp\left(\frac{1}{2} . \sum_{i=1}^{n} \left\{ \left(\frac{x_i - \mu_i}{\sigma} \right)^2 - \mu_i . \ln(\mu_i) \right\} \right) \tag{7.5}$$

Para encontrarmos a solução deste equacionamento devemos minimizar o expoente desta função, ficando com:

$$g(\mu_1, \mu_2, ..., \mu_n | x_1, x_2, ..., x_n) = \frac{1}{2} \cdot \sum_{i=1}^{n} \left\{ \left(\frac{x_i - \mu_i}{\sigma} \right)^2 - \mu_i . \ln(\mu_i) \right\} \quad (7.6)$$

Portanto fazendo a derivada temos:

$$\frac{\partial g}{\partial \mu_i} = -\left(\frac{x_i - \mu_i}{\sigma^2} \right) + \left(\ln(\mu_i) + \frac{\mu_i}{\mu_i} \right) \Rightarrow \quad \phi' = \frac{\mu_i - x_i}{\sigma^2} + \ln(\mu_i) + 1 \quad (7.7)$$

Para encontrar o ponto de mínimo fazemos $\phi' \Rightarrow 0$

Para resolvermos esta equação, que é linear utilizamos o Excel e desenvolvemos macros para efetuar este cálculo. O nosso sistema de solução utilizou um método iterativo partindo de um valor e testando na equação. Aquele valor que levou $\phi' \Rightarrow$ 0com uma precisão de 10e-5 foi o escolhido.

Abaixo podemos verificar que usando desvio padrão $\sigma = 1$, obtivemos uma imagem um poço mais escurecida.

Figura 7.1: Imagem inicial e Imagem tratada mais escura

Neste método inicial que desconsidera os valores e portanto, a influência dos pixels vizinhos sabíamos que o resultado seria a mesma imagem e dependendo de σ mudaríamos a tonalidade da imagem para mais escuro.

2^0 Passo:

A partir de então se mostrou necessário encontrar um método para relacionar cada pixel com todos os seus vizinhos. Escolhemos então, inicialmente, utilizar a equação do calor usada na fase inicial deste trabalho para tratar uma imagem borrada porém agora não mais utilizando Transformada de Fourier mas sim estatística Bayesiana no método da máxima entropia.

O equacionamento foi o seguinte:

$$\tilde{f}(x_1, x_2) = \iint_{\Omega} f(\xi_1, \xi_2) \cdot g(x_1 - \xi_1, x_2 - \xi_2) \quad (7.8)$$

Aqui temos uma convolução bidimensional entre a imagem nítida e a equação do calor. Onde Ω é o espaço bidimensional em questão.

Trabalhando com um sistema discreto temos:

$$\tilde{f}(x_1, x_2) = \sum_{\xi_1=-N}^{N} \sum_{\xi_2=-N}^{N} f(\xi_1, \xi_2) \cdot g(x_1 - \xi_1, x_2 - \xi_2) \tag{7.9}$$

Substituindo a função g da equação do calor chegamos a nossa função de borramento:

$$\tilde{f}(x_1, x_2) = \sum_{\xi_1=-N}^{N} \sum_{\xi_2=-N}^{N} f(\xi_1, \xi_2) \cdot \exp\left\{-\frac{1}{4 \cdot t}\left[(x_1 - \xi_1)^2 + (x_2 - \xi_2)^2\right]\right\} \tag{7.10}$$

Inserindo na solução de Bayes, utilizando o método da máxima entropia no espaço 2D temos:

$$g(\mu(\xi_1, \xi_2) | x(i,j)) = \frac{1}{2} \cdot \sum_{i=-N}^{N} \sum_{j=-N}^{N} \left\{ \left(\frac{x_{ij} - \tilde{f}(i,j)}{\sigma} \right)^2 + \mu_{ij} \cdot \ln(\mu_{ij}) \right\} \tag{7.11}$$

Analogamente ao primeiro passo, fazemos a derivada para encontrar o ponto de mínimo:

$$\frac{\partial g}{\partial \mu_i} = \sum_{i=-N}^{N} \sum_{j=-N}^{N} \left\{ \left(\frac{x_{ij} - \tilde{f}(i,j)}{\sigma^2} \right) \cdot \frac{1}{4 \cdot t} \cdot \sum_{\xi_1=-N}^{N} \sum_{\xi_2=-N}^{N} \cdot e^{-\frac{(i-m)^2 \cdot (j-n)^2}{4 \cdot t}} + \ln(\mu_i) + 1 \right\} \tag{7.12}$$

Esta equação nos leva a um sistema não linear de N equações, sendo necessário portanto utilizar algum método de solução.

7.2 Escolha do melhor método de solução

O desenvolvimento de um novo método para resolução de sistemas não lineares com "n" variáveis foge do escopo deste trabalho. Isto definido, se fez necessário encontrar métodos já prontos para implementar a equação XXX, em questão, suficientemente robustos para prover uma solução mesmo considerando todas os limites impostos em um tratamento de imagens, como uma banda de valores possíveis para a solução com limite superior de 255, branco e o limite inferior de

0, preto absoluto.

Outra característica a ser levada em consideração é o conjunto de ferramentas que dispomos para criação, testes e operação do programa implementado. No nosso caso foram selecionadas três plataformas principais, as quais poderiam ser usadas caso houvesse necessidade. São elas: Eclipse, Mathematica e Matlab.

Estes três softwares, embora bem diferentes serviriam para buscar a solução desejada cada um com suas especificidades. O Eclipse, plataforma para C e C++, poderia ser usado em conjunto com um compilador e bibliotecas nas quais encontraríamos as funções que resolvem sistemas não lineares. Tanto o Mathematica 5.0, como o Matlab 7.0 apresentaram-se como sistemas práticos e potentes para a complexidade exigida neste trabalho, tornando sua implementação mais fácil, e com isso a disponibilidade de tempo foi maior para testes e correções de erros nas fórmulas, encontrados no decorrer desta monografia.

Apesar destes dois últimos softwares possuírem bibliotecas diferentes, em ambas encontramos funções que minimizam a função-objetivo em questão. No caso do Mathematica 5.0, utilizamos a 'FindMinimum' e o no caso do Matlab 7.0, 'fminsearch', 'fmincon' e 'lqsnonlin'. Estas funções empregam respectivamente os métodos conhecidos como:

FindMinimum: É uma otimização que tem como método de resolução as seguintes opções: Gradiente Conjugado, Gradiente, Levenberg-Marquardt, Newton e Quasi-Newton.

Fminsearch: É uma otimização não linear sem restrições. Usa o método Simplex. Método de busca direta sem fazer uso de gradientes numéricos ou analíticos.

Fmincon: É uma otimização não linear sem restrições. Usa o método Simplex. Método de busca direta sem fazer uso de gradientes numéricos ou analíticos. Diferente da 'fminsearch' esta função busca resultados dentro de um limite pré-estipulado.

Lsqnonlin: Soluciona problemas não lineares de Mínimos Quadrados. Por padrão o Lsqnonlin utiliza um algoritmo de larga escala. Este algoritmo é um método que utiliza intervalos de confiança e é baseado no método de reflexão interna de Newton, utilizando PCG, gradientes conjugados pré-condicionados. Caso seja escolhido o algoritmo de escala média, o método utilizado será o de Levenberg-Marquardt. Como alternativa pode ser selecionado o método de Gauss-Newton que geralmente se mostra mais rápido em situações onde o resíduo é menor.

7.3 Implementação

O primeiro passo antes da implementação foi a compreensão, muito clara, sobre o funcionamento da função-objetivo, isto é, tínhamos uma equação com muitos índices que deveriam variar da forma correta para que a estruturação desta pudesse ocorrer sem erros e assim a função de solução do Matlab poderia utilizá-la e buscar como resultado uma matriz onde cada ponto representa um píxel de uma imagem que foi tratada e que portanto apresenta maior nitidez que a imagem inicial.

Implementamos em todos os métodos citados acima para que fosse possível a análise, e comparação dos resultados verificados a seguir.

7.4 Dificuldades

O principal problema encontrado durante todo o processo foi o "estouro"das matrizes. Isto é, os métodos não estavam convergindo ou estavam tendendo a infinito. O problema passou a ser verificado quando começamos a fazer testes alterando as variáveis principais como tempo, desvio padrão e o epsilon, fator responsável pelo peso, relevância do termo de regularização do método.

Os métodos utilizados para a solução do sistema de equações não lineares presente nesta etapa são métodos, em geral, de aproximações sucessivas e, portanto, fazem uso da matriz de constantes que multiplicam as variáveis em questão e calculam a sua inversa. Como sabemos que para calcular a inversa de uma matriz temos que calcular o determinante da mesma, resolvemos fazer o teste, e calculá-lo.

Efetuando, portanto o cálculo para t = 0.1 com uma matriz bem pequena, 2x2, apenas para que seja possível visualizar aqui temos como imagem nítida e borrada o seguinte:

$$Nitida = \begin{pmatrix} 245 & 100 \\ 223 & 217 \end{pmatrix} Borrada = \begin{pmatrix} 239 & 105 \\ 234 & 228 \end{pmatrix}$$

Calculando as constantes que multiplicam o sistema de quatro equações cada uma com quatro incógnitas, isto é, numero de equações = quadrado do comprimento da matriz Nítida, temos:

$$Constantes = \begin{pmatrix} 0.1153 & 0.0864 & 0.0864 & 0.1153 \\ 0.0864 & 0.0648 & 0.0648 & 0.0864 \\ 0.0864 & 0.0648 & 0.0648 & 0.0864 \\ 0.1153 & 0.0864 & 0.0864 & 0.1153 \end{pmatrix}$$

Assim, observamos que o determinante da matriz Constantes tende a zero desta forma podemos verificar o porque da matriz "estourar".

Isto significa que quando tendemos o tempo de borramento a zero na equação de borramento inicial o resultado tende a infinito.

Mostrou-se necessário então a necessidade de mudanças que coibissem esse comportamento. O resultado foi uma nova equação com um método de borramento que faz com que quando t tende a zero, o que significa que não houve difusão nenhuma. O valor de cada pixel em questão deve continuar o mesmo e não tender a infinito, isto é, o termo que contem t , quando t tende a zero, deve ir a um, cem porcento do valor do pixel inicial, e não à infinito.

A equação em questão portanto foi:

$$\tilde{\mu}[x,y] = \sum_{i=-1}^{+1}\sum_{j=-1}^{+1}[\int_{i-0,5}^{i+0.5}\int_{j-0,5}^{j+0.5}(\frac{1}{(2\sqrt{\pi t})^2}e^{-\frac{\xi^2+\varsigma^2}{4t}})d\xi d\varsigma]\mu[x+i,y+j] \quad (7.13)$$

8 RESULTADOS

O procedimento que seguimos no primeiro semestre foi repetido agora como forma de se atingir a solução desejada contendo, portanto, as seguintes etapas:

1- Obtenção de uma imagem borrada, a partir de uma nítida através do método utilizado e da equação do calor.

2- Tratamento da imagem borrada pelo respectivo método com objetivo de retorno à imagem nítida inicial.

3- "Borramento"da imagem nítida e posterior adição de erros aleatórios.

4- Tratamento da imagem obtida em (3) para verificação da robustez e estabilidade do sistema desenvolvido visando obter um melhoramento que se aproxime da imagem nítida inicial.

5- Aperfeiçoamento do software para que consiga melhorar qualquer imagem parcialmente difusa.(*)

(*) Observação: Na primeira etapa desta monografia fizemos uso da regularização de Thikonov para que o sistema ficasse estável.

Assim como no primeiro semestre, repetindo os passos, conseguimos borrar uma imagem nítida como podemos ver abaixo:

Figura 8.1: Borrando uma Imagem

Na segunda etapa, esta imagem borrada, portanto foi tratada pelo processo

aqui desenvolvido e desta forma foi possível retornar à imagem nítida anterior, isto é, dada a imagem borrada em questão, mesmo sem conhecer a sua versão nítida, seria possível atingir uma imagem mais nítida, como vemos abaixo:

Figura 8.2: Borrando uma Imagem

A terceira etapa correspondente à adição de erros à equação "borrada"que queríamos tratar, nos levou ao resultado como podemos verificar abaixo:

Figura 8.3: Borrada e volta com erros aleatóreos

Ao lado da imagem "borrada"com adição de erros aleatórios podemos ver que a resolução se mostrou estável à alterações, e o sistema portanto, robusto o suficiente para buscar a solução desejada.

É importante ressaltar que na primeira etapa desta monografia quando a tentativa de retornar uma imagem borrada com adição de erros aleatórios estourou, mostrou-se necessária uma regularização que no caso foi feita pelo método de Thikonov. Enquanto isso nesta segunda etapa, o método da máxima entropia já traz embutida a regularização através do termo logarítmico.

9 CONCLUSÕES

O trabalho se mostrou de grande abrangência e importância, já que ao longo de nosso estudo nos deparamos com inúmeras aplicações reais e teóricas. Os resultados foram positivos e concluímos com sucesso o projeto que haviamos nos proposto.

Através do desenvolvimento e implementação dos algoritmos tivemos que transpor barreiras como: programação em diferentes ambientes, diferentes metodologias de apoio foram usadas, baixo desempenho computacional para realizar os testes de validação.

Fazendo uma análise do projeto pode-se dizer que comparando os métodos desenvolvidos no primero semestre (Deconvolução atravéz da Transformada de Fourier com Regularização de Thikonov) com o do segundo (Método da Máxima Entropia), vemos que a questão tem que ser ponderada pelo uso final que se quer dar.

No primeiro caso temos um custo computacional de crescimento linear N, já no segundo vemos que este custo aumenta para N^2, além disso o segundo propicia resultados mais apurados (imagens de melhor qualidade). Sendo que para uma aplicação como a da CET (Companhia de Engenharia Tráfego) que pretende "scanear"placas com objetivo de aplicar multas para carros fora do rodízio, prefeririam um método mais rápido.

10 REFERÊNCIAS BIBLIOGRÁFICAS

Referências:

[1] "Digital Image Processing- R.C.Gonzalez / R.Woods - Addison Wesley - 1992.

[2] "Video Microscopy- S. Inoué - Plenun Press - Nova Iorque - 4Th edition - 1989

[3] "Computação Gráfica: Imagem- J. Gomes e L.Velho - Ed. IMPA - 1994

[4] "Vision- D. Marr - Freeman - Nova Iorque - 1982

[5] "Image Analysis and Computer Vision- A. Rosenfeld - Computer Vision Graphics and Image Processing - vol 59, p.367-404 - 1993

[6] "Visual Language and software development environment for image processing- J. Rasure, D.Argiro, T.Sauer e C.Williams - Jour. of Imaging System and Technology - vol.2, p. 183-199- 1990

[7] "Processamento de Imagens: Métodos e Análises- M. P. Albuquerque e M. P. Albuquerque - CBPF/MCT - 2000

[8] "Physics 707 Inverse Problems- S.M. Tan and Colin Fox, The University of Auckland - 2000

[9] "Maximum entropy method in image processing- GULL, S F; SKILLING, J IEE Proceedings. Vol. 131, Part F-Communications, no. 6, pp. 646-659. Oct. 1984