Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e Computação
Departamento de Comunicações

Identificação e Transformação de Valores Aberrantes como Medida de Confiabilidade do Método das Diferenças para Estimativa de Fluxo Óptico em Seqüências de Imagens

Letícia Rittner
Orientador: Prof. Dr. Luiz César Martini

Campinas, SP – Brasil
Janeiro, 2004

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e Computação
Departamento de Comunicações

Identificação e Transformação de Valores Aberrantes como Medida de Confiabilidade do Método das Diferenças para Estimativa de Fluxo Óptico em Seqüências de Imagens

Letícia Rittner
Orientador: Prof. Dr. Luiz César Martini

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação (FEEC) da Universidade Estadual de Campinas (UNICAMP), como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Elétrica.

Campinas, SP – Brasil
Janeiro, 2004

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e Computação
Departamento de Comunicações

Identificação e Transformação de Valores Aberrantes como Medida de Confiabilidade do Método das Diferenças para Estimativa de Fluxo Óptico em Seqüências de Imagens

Letícia Rittner

Banca examinadora:

Prof. Dr. Luiz César Martini
DECOM/FEEC/UNICAMP

Profa. Dra. Nina Sumiko Tomita Hirata
IME/USP

Prof. Dr. Roberto de Alencar Lotufo
DCA/FEEC/UNICAMP

Prof. Dr. João Baptista Tadanobu Yabu-uti
DECOM/FEEC/UNICAMP

Campinas, SP – Brasil

Janeiro, 2004

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Rittner, Letícia
R518i Identificação e transformação de valores aberrantes como medida de confiabilidade do método das diferenças para estimativa de fluxo óptico em seqüências de imagens / Letícia Rittner. --Campinas, SP: [s.n.], 2004.

Orientador: Luiz César Martini.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Processamento de imagens. 2. Valores estranhos - estatística. 3. Movimento. I. Martini, Luiz César. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Resumo

Um problema fundamental no processamento de seqüências de imagens é a estimativa de fluxo óptico (ou velocidade da imagem). Uma vez computadas, as medidas de velocidade da imagem podem ser usadas em uma variedade de tarefas, desde recuperação de movimento 3-D até dedução de estruturas e segmentação de objetos. Este trabalho implementa de forma didática o algoritmo proposto por Lucas e Kanade para estimativa de fluxo óptico, permitindo sua utilização em estudos e aplicações diversos. O método escolhido foi o desenvolvido por LK, por ter sido considerado o de melhor desempenho por estudos que compararam diversos métodos. Uma vez que estes estudos basearam-se principalmente em resultados obtidos com a análise de imagens sintéticas, este trabalho propõe uma modificação do algoritmo LK utilizando valores aberrantes, com o objetivo de obter resultados mais confiáveis, não só para seqüências de imagens sintéticas, mas principalmente em seqüências de imagens reais.

Abstract

A fundamental problem in the processing of image sequences is the measurement of optical flow (or image velocity). Once computed, the measurements of image velocity can be used for a wide variety of tasks, such as recovering three-dimensional motion, deducing structure and segmenting objects. This work implements in a didactic way the Lucas and Kanade algorithm, allowing it to be used in several applications. The LK algorithm was chosen because it was considered the one with best performance by some performance studies. Since these studies were based on results using sintetic image sequences, this work proposes a modification in the LK algorithm using outliers, in order to obtain more reliable results not only with sintetic sequences, but specially with real image sequences.

“Tudo o que é inteligente já foi pensado, só o que podemos tentar é pensá-lo de novo.”

Johann Wolfgang von Goethe

Aos meus pais.

Ao professor Dr. Luiz César Martini, não só por sua orientação, mas pela confiança em mim depositada e pela paciência e compreensão ante as surpresas do caminho, meu muito obrigado.

Agradecimentos

Ao professor Dr. Roberto de Alencar Lotufo, por sua generosa colaboração.

Aos colegas Wilson, Rangel, Tatiana e Silvia, que fizeram das horas de estudo juntos momentos agradáveis e muito proveitosos e principalmente ao Jorge, por compartilhar comigo não só seu conhecimento, mas também sua sabedoria e alegria de viver.

Aos meus pais, Carmem Lucia Rittner e Roberto Rittner Neto, pelo exemplo de vida e pelo apoio incondicional.

Ao meu marido Ivan, por acreditar em meu sonho e possibilitar realiza-lo e à minha filha Beatriz, por me motivar, mesmo sem saber que o fazia.

À minha irmã Marilia e ao meu cunhado Edison pela ajuda nos momentos de desânimo.

A todos aqueles, amigos e familiares, que me apoiaram e incentivaram nos momentos difíceis e comemoraram comigo cada vitória conquistada ao longo deste trabalho.

À Capes, pelo apoio financeiro.

Sumário

Lista de figuras

Lista de tabelas

Glossário

Arquivos-m	M-files
Bigodes	Whiskers
Bordas	Edges
Caixa de ferramentas	Toolbox
Campo de movimento	Motion field
Campo de velocidade	Velocity field
Características	Features
Casar	To match
Correspondência	Matching
Especularidades	Specularity
Estrutura através do movimento	Structure from motion
Filtros sintonizados em velocidade	Velocity-tuned filters
Gráfico de caixa	Boxplot
Gráfico de dispersão	Scatterplot
Gráfico Q-Q	Quantile-Quantile plots
Medidas de confiabilidade	Confidence measures
Mínimos quadrados	Least-square
Problema da abertura	Aperture problem
Sistema de captação da imagem	Viewing sistem
Sobreposição do espectro	Aliasing
Superfície Lambertiana	Lambertian surface
Valores aberrantes	Outliers
Valor limite	Threshold

Abreviaturas

IQR Interquartil range

LK Lucas e Kanade

LS Least-squares

QQ Quantil-Quantil

RMS Root mean square

SNR Signal noise ratio

Capítulo 1 – Introdução

1.1 Motivação

Uma relevante contribuição da Visão Computacional é extrair descritores do mundo real a partir de uma imagem ou de uma seqüência de imagens. Captar imagens através de uma câmera é um processo não-destrutivo, fácil e barato e que em conjunto com a Visão Computacional pode substituir sensores, radares e outros equipamentos mais caros e sofisticados. A aplicação pretendida é o que determina quais descritores deverão ser extraídos das imagens.

Por exemplo, uma técnica conhecida como *estrutura através do movimento* (“structure from motion”) torna possível extrair informações do ambiente e da câmera através de uma seqüência de imagens. A indústria cinematográfica usa esta técnica para construir modelos tridimensionais computadorizados de prédios e outras construções, para serem usados em cenas de incêndios, explosões etc.

A área médica é uma fonte inesgotável de aplicações para a Visão Computacional. Depois das imagens serem captadas através de técnicas como a Ressonância Magnética ou Tomografia Computadorizada, elas precisam ser analisadas para se chegar a um diagnóstico ou para monitorar a evolução de pacientes após cirurgia. Esta análise, antes manual, agora pode ser feita de forma automática, graças à Visão Computacional. Podemos encontrar programas desenvolvidos especialmente para auxiliar a análise de imagens cardíacas, por exemplo, permitindo a medição do fluxo sanguíneo e a reconstrução e análise de estruturas anatômicas do coração.

Um terceiro grupo de aplicações compreende a interpretação de imagens de satélites. A análise destas imagens pode ser utilizada para fins militares (cálculo do tempo para colisão; reconhecimento, rastreamento e destruição de alvos), fins de pesquisa na área agrícola (estudo e acompanhamento do cultivo de grãos), fins meteorológicos etc.

A caracterização e o entendimento do movimento humano na dança e no atletismo, por exemplo, tem permitido o aprimoramento de técnicas de treinamento e o desenvolvimento de equipamentos para tais atividades.

Até mesmo a área de transmissão de dados utiliza-se da predição de movimento para métodos de compressão de arquivos de imagens.

1.2 Descrição do problema

Um problema fundamental no processamento de seqüências de imagens é a estimativa de fluxo óptico (ou velocidade da imagem). O objetivo é calcular uma aproximação do *campo de movimento* ("motion field") bidimensional – uma projeção da velocidade tri-dimensional de cada ponto da superfície observada sobre a superfície da imagem – através de padrões espaciais e temporais de intensidade das imagens. Uma vez computadas, as medidas de velocidade da imagem podem ser usadas em uma variedade de tarefas, desde recuperação de movimento tridimensional até dedução de estruturas e segmentação de objetos.

O fluxo óptico tem sido foco de uma série de estudos, não só devido à sua indiscutível utilidade, mas também por apresentar sérios desafios computacionais. Todas as formulações requerem fortes pressupostos iniciais para tornar a estimativa de fluxo óptico um problema bem postulado, mas estas hipóteses podem levar a resultados imprecisos em algumas situações. Assim, métodos que pressupõem constância da intensidade luminosa apresentam resultados mais confiáveis para cenas em ambientes fechados do que para seqüências de imagens com grandes variações de luz e sombra.

Muitos métodos para estimativa do fluxo óptico foram propostos e outros continuam a aparecer. Alguns trabalhos surgiram na tentativa de avaliar e comparar o desempenho destes diversos métodos. Dentre eles, Barron, Fleet *et al.* [1,2] realizaram uma análise empírica de nove algoritmos de estimativa de fluxo óptico. Os algoritmos foram testados em cinco seqüências de imagens sintéticas, onde o fluxo óptico real era conhecido e em quatro seqüências de imagens reais, para as quais não se tinha informação do fluxo óptico. O estudo comparativo se preocupou com a precisão, confiabilidade e densidade dos resultados obtidos pelos diversos métodos. Dos diferentes métodos testados, concluiu-se que os métodos de maior confiabilidade foram o método diferencial de primeira ordem de Lucas e Kanade [3], usualmente chamado de método LK, e o método baseado em fase proposto por Fleet e Jepson [4].

Outro estudo de desempenho de algoritmos de fluxo óptico foi conduzido por Galvin, McCane *et al.* [5], onde um algoritmo “ray tracer” (técnica para geração de imagens, comentada no Capítulo 2) foi modificado para permitir a obtenção do campo de movimento real para cenas complexas. Os mapas de fluxo resultantes foram comparados com os mapas obtidos através de oito algoritmos de estimativa de fluxo óptico, para três cenas sintéticas, porém complexas. Este estudo mostrou que o método desenvolvido por Lucas e Kanade tem desempenho superior aos demais, apesar de apresentar mapas de fluxo esparsos.

1.3 Objetivos

Como dito anteriormente, o processamento de imagens contendo movimento passa normalmente pela etapa de estimativa de fluxo óptico, independente do que se quer alcançar com este processamento. O presente trabalho tem como objetivo implementar, de forma didática, o algoritmo proposto por Lucas e Kanade (LK) para estimativa de fluxo óptico, permitindo sua utilização em estudos e aplicações diversos. Dessa forma, os interessados em trabalhar com segmentação de imagens através do movimento, análise de imagens médicas, rastreamento de alvos e demais aplicações já citadas, poderiam concentrar seus esforços no desenvolvimento da aplicação escolhida, não precisando investir parte de seu tempo entendendo e implementando a etapa de estimativa de fluxo óptico.

O método escolhido para ser implementado foi o desenvolvido por LK, por ter sido considerado o de melhor desempenho por estudos que compararam diversos métodos. Uma vez que estes estudos basearam-se principalmente em resultados obtidos com a análise de imagens sintéticas, este trabalho propõe uma modificação do algoritmo LK, com o objetivo de obter resultados mais confiáveis, não só para seqüências de imagens sintéticas, mas principalmente em seqüências de imagens reais.

1.4 Metodologia

O método proposto por Lucas e Kanade, também conhecido por método das diferenças, possui algumas variantes (ponto-a-ponto, média, mínimo erro quadrático e mínimo erro quadrático ponderado). Para entendermos a razão de ser destas variantes, implementamos desde o algoritmo mais simples até o mais complexo. Desta forma, utilizando-se uma mesma seqüência de imagens, podemos observar a evolução dos mapas de fluxos obtidos para cada uma destas instâncias. Ainda como uma última variante implementamos o método das diferenças utilizando a identificação e transformação de valores aberrantes como medida de confiabilidade.

Todos os algoritmos foram implementados no ambiente de desenvolvimento do MATLAB. A *caixa de ferramentas* (“toolbox”) de Processamento de Imagens do MATLAB torna a manipulação de arquivos de imagens rápida e ágil. Além disso ela disponibiliza um ambiente flexível e descomplicado para testar idéias e algoritmos, com recursos gráficos interessantes para comparar resultados e apresentá-los.

Para melhor organização e entendimento do sistema desenvolvido, optamos por implementar cada etapa do algoritmo através de uma “function” desenvolvida no MATLAB. Desta forma, as “functions” podem ser estudadas de forma isolada por aqueles que quiserem entender apenas uma etapa do processamento.

Já a etapa de testes do sistema foi implementada através de “scripts”, de forma a permitir a escolha de valores diferentes para as variáveis de testes, sem necessidade de se alterar as funções dos algoritmos. Para se executar uma seqüência completamente diferente de testes, basta escrever um novo “script”, sem qualquer alteração das “functions” que compõem o sistema.

1.5 Organização da dissertação

O Capítulo 2 deste trabalho trata dos conceitos básicos relacionados ao fluxo óptico e descreve os principais métodos existentes para estimá-lo. Apresenta também dois estudos de desempenho destes métodos e resume suas conclusões.

O Capítulo 3 descreve o método LK para estimativa do fluxo óptico, considerado o de melhor desempenho pelos estudos apresentados no Capítulo 2. Cada uma das diferentes variantes do método é descrita e seus pontos fracos são apontados.

O Capítulo 4 faz uma introdução à teoria de valores aberrantes, utilizada para análise de dados em trabalhos científicos das mais diversas áreas. É esta teoria de valores aberrantes que nos ajudará na tarefa de propor uma variante do método LK.

O Capítulo 5 apresenta uma proposta de modificação do método LK, introduzindo uma etapa de identificação e transformação de valores aberrantes como medida de confiabilidade.

No Capítulo 6, o método LK é apresentado em sua forma matricial, de maneira a deixá-lo mais próximo de sua implementação. Em seguida, são descritos os algoritmos desenvolvidos em MATLAB que implementam cada uma das instâncias do método LK e também o método proposto no Capítulo 5.

Os “scripts” dos testes, as imagens utilizadas para realizá-los e seus resultados estão relatados no Capítulo 7, com os mapas de fluxo óptico para cada seqüência de imagens estudada utilizando cada uma das variantes do método.

Finalmente, são apresentadas no Capítulo 8 as conclusões deste trabalho e discutidos os desdobramentos para possíveis trabalhos futuros.

Capítulo 2 - Revisão bibliográfica

Se tivermos uma seqüência de imagens no tempo, e nelas existirem objetos em movimento ou se a câmera estiver se movendo, informações importantes sobre o conteúdo dessas imagens podem ser obtidas analisando-se as diferenças entre as imagens (diferenças essas causadas pelo movimento). Por exemplo, dada uma imagem de um carro em movimento, decidindo-se quais pixels na imagem representam o movimento pode-se identificar os pixels pertencentes ao carro e os pertencentes ao fundo estático [6].

Estudando o movimento em detalhe podemos responder perguntas tais como:

- Quantos objetos em movimento existem na imagem?
- Em que direção estão se movendo?
- Quão rápido estão se movendo?

Um problema fundamental no estudo do movimento em seqüências de imagens é a estimativa de *fluxo óptico* (ou velocidade da imagem). O objetivo é calcular uma aproximação do *campo de movimento* ("motion field") bidimensional – uma projeção da velocidade tridimensional de cada ponto da superfície observada sobre a superfície da imagem – através de padrões espaciais e temporais de intensidade das imagens. Uma vez computadas, as medidas de velocidade da imagem podem ser usadas em uma variedade de tarefas, desde recuperação de movimento tridimensional até dedução de estruturas e segmentação de objetos.

Antes de iniciarmos a descrição e discussão dos métodos de estimativa de fluxo óptico, precisamos definir claramente o que é *fluxo óptico* e o que é *campo de movimento*. Além disso, tentaremos esclarecer porque nem sempre o fluxo óptico corresponde ao campo de movimento, ou ainda, quais as condições necessárias para que o fluxo óptico corresponda ao campo de movimento.

2.1 Conceitos

2.1.1 Velocidade tridimensional

Velocidade tridimensional é a velocidade real dos objetos contidos na seqüência de imagens. A câmera através da qual estão sendo captadas as imagens também possui uma velocidade tridimensional.

2.1.2 Campo de Movimento (“Motion field”)

Quando um objeto se move em frente a uma câmera, há uma mudança correspondente na imagem. Então, se um ponto $\boldsymbol{p_0}$ de um objeto se move com uma velocidade $\boldsymbol{v_0}$, à sua imagem $\boldsymbol{p_i}$ pode ser associada uma velocidade $\boldsymbol{v_i}$ que indica seu movimento no plano da imagem. O conjunto destes vetores de velocidade ($\boldsymbol{v_{is}}$) forma o *campo de movimento*[1].

A figura a seguir mostra o primeiro e segundo quadros de uma seqüência de imagens, onde um quadrado se desloca na diagonal, e o respectivo campo de movimento. Para todos os pixels pertencentes ao fundo estático, os vetores de velocidade são nulos (representados por pontos) enquanto que para os pixels pertencentes ao quadrado, os vetores de velocidade apontam para a direção do movimento (diagonal) e são representados por setas. O conjunto formado pelos pontos (vetores de velocidade nulos) e pelas setas (vetores de velocidade não nulos) representam o campo de movimento.

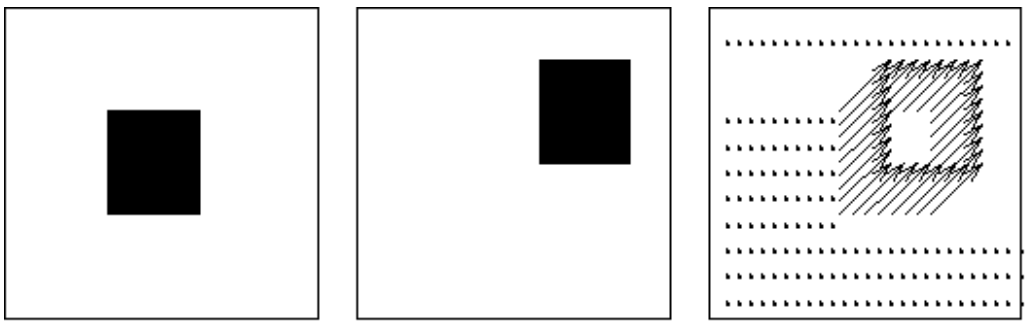


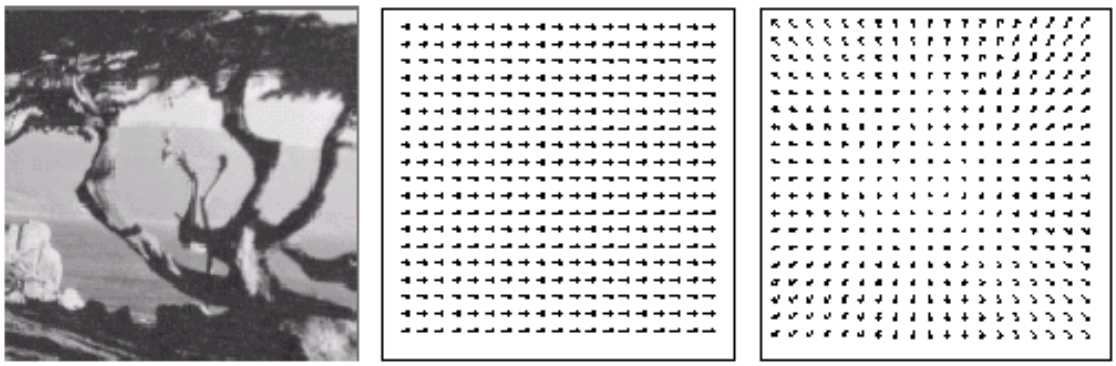
Figura 2-1 : *O primeiro e o segundo quadros mostram um quadrado em um movimento de translação. O terceiro quadro representa o campo de movimento correspondente.*

[1] Conceito e ilustração apresentados por Owens em suas notas de aula [7].

2.1.3 Fluxo óptico

De acordo com Horn e Schunck [8], o *fluxo óptico* de uma seqüência de imagens é “o movimento aparente do padrão de intensidade luminosa”. Ou ainda, o fluxo óptico é o *campo de velocidade* (“velocity field”) que transforma uma imagem na próxima imagem da seqüência.

O fluxo óptico estimado para cada pixel possui uma magnitude e uma direção e é representado por um vetor. O conjunto de vetores desenhados sobre a imagem é chamado de *mapa de fluxo*. Na figura 2-2 são apresentados dois exemplos de mapa de fluxo. O primeiro mapa representa o movimento contido na seqüência “Translating tree”, onde a câmera se desloca para a esquerda, causando a impressão de que a árvore está se deslocando para a direita. O segundo mapa de fluxo refere-se à seqüência “Diverging tree”, onde a câmera se aproxima da árvore, dando a impressão de que a árvore avança para frente.



(a) (b) (c)

Figura 2-2 : *(a) Um dos quadros pertencentes a ambas as seqüências “Translating Tree” e “Diverging Tree” (b) mapa de fluxo óptico da seqüência “Translating Tree” (c) mapa de fluxo óptico da seqüência “Diverging Tree”.*

2.1.4 Densidade de mapas de fluxo

Para cada pixel em uma imagem podemos teoricamente calcular um vetor de velocidade correspondente. Na prática, apenas alguns métodos de estimativa de fluxo óptico são capazes de obter vetores para todos os pixels. A maioria dos métodos trata a imagem em blocos, estimando um vetor de fluxo para cada conjunto de pixels. Quanto mais vetores de velocidade um mapa de fluxo contiver, mais denso ele é. Mapas de fluxo com poucos vetores de velocidade são ditos pouco densos, ou ainda, esparsos.

A figura 2-3 ilustra este conceito de densidade de mapa de fluxo através de dois exemplos. Enquanto o primeiro exemplo mostra um mapa de fluxo esparso (vetores de velocidade apenas para alguns blocos de pixels da imagem), o segundo mapa é um exemplo de mapa de fluxo denso (um vetor de velocidade para cada bloco de pixels).

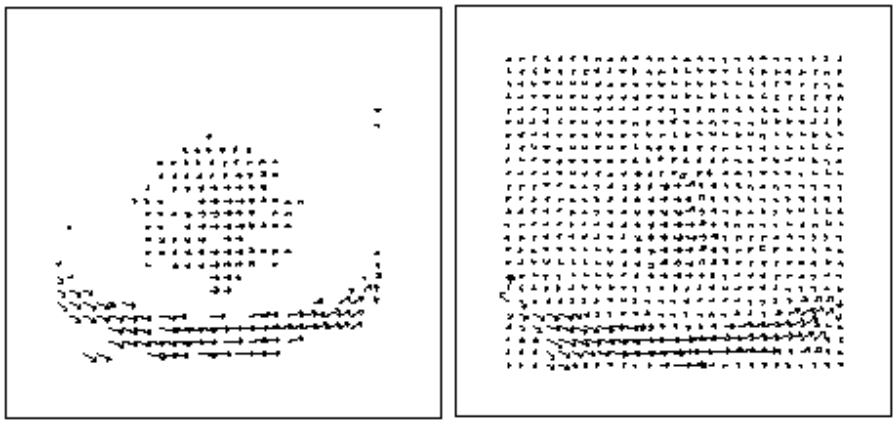


Figura 2-3 : *Dois exemplos de mapas de fluxo para a mesma seqüência de imagens, obtidos por métodos diferentes. O primeiro é um exemplo de mapa esparso, enquanto que o segundo é um mapa de fluxo denso.*

2.2 *Campo de movimento* versus *fluxo óptico*

Na maioria dos casos, o *fluxo óptico* corresponde ao *campo de movimento*, ou seja, à projeção bidimensional do movimento real tri-dimensional dos objetos contidos nas imagens. No entanto, a relação entre o fluxo óptico no plano da imagem e as velocidades dos objetos no mundo tridimensional não é necessariamente óbvia. Quando uma figura em transformação é projetada em uma tela parada, a imagem vista é de movimento. Por outro lado, um objeto em movimento pode levar a um padrão de intensidade luminosa constante.

Considere, por exemplo, uma esfera uniforme que exibe uma sombra porque seus elementos na superfície estão orientados em diferentes direções. Ao ser rotacionada, o fluxo óptico é igual a zero, pois a sombra não se move com a superfície da esfera [8].

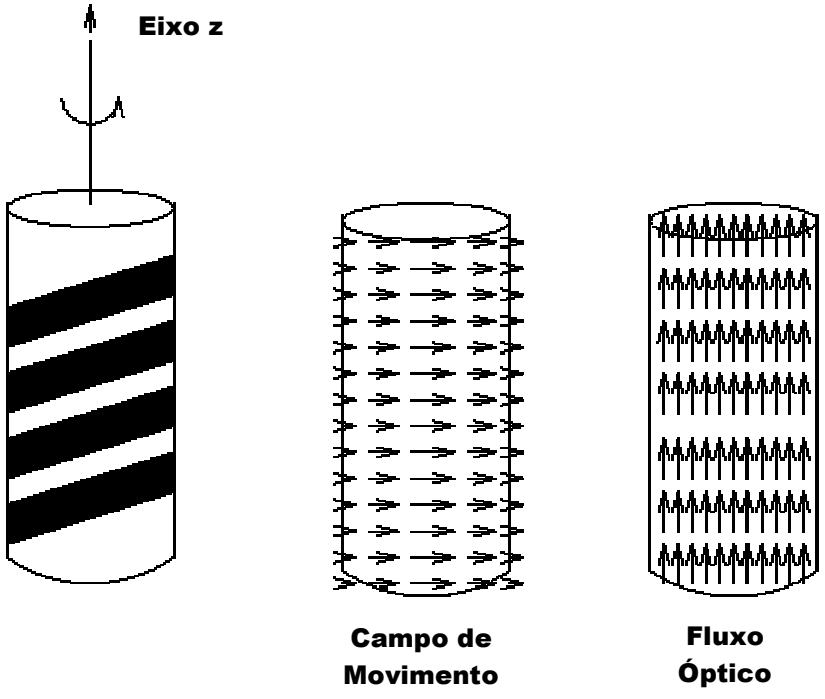


Figura 2-4 : *Uma espiral girando em torno de seu eixo vertical é um bom exemplo de Campo de Movimento e Fluxo Óptico que não coincidem.*

Outro exemplo onde o campo de movimento e o *fluxo óptico* são diferentes está ilustrado na figura 2-4. O cilindro com uma espiral pintada nele está sendo rotacionado para a direita, em torno do eixo z. A velocidade tridimensional, portanto, é a que está representada pela seta em torno do eixo z. O campo de movimento resultante, neste caso, aponta para a direita, conforme mostram as setas sobre o cilindro no meio da figura. Estas setas representam as velocidades tridimensionais projetadas no plano da imagem. As setas desenhadas sobre o terceiro cilindro apontam na direção do fluxo óptico (para cima). Ele representa o movimento observado na imagem e não corresponde ao campo de movimento.

Para que o fluxo óptico corresponda exatamente ao movimento da imagem, algumas condições têm que ser satisfeitas:

- Constância de intensidade luminosa. Isto é, os pixels representando pontos de um objeto em movimento possuem a mesma intensidade ao longo de toda a sequência;

- Superfície Lambertiana (“lambertian surface reflectance”). Ou seja, a luminância é constante em uma dada superfície, qualquer que seja a direção de observação (difusor perfeito).

- Translação paralela ao plano da imagem

O grau com que estas condições são satisfeitas determina a precisão com que o fluxo óptico representa o movimento.

2.3 Estimando o fluxo óptico

Essencialmente, detectar movimentos bidimensionais pode envolver o processamento de cenas onde a câmera se move em um ambiente contendo objetos estacionários e não-estacionários. Além disso, eventos visuais tais como oclusões, objetos transparentes e não-rígidos aumentam a complexidade da estimativa do fluxo óptico. Segundo Mitiche e Bouthemy [9], de uma ampla perspectiva podemos distinguir três principais fontes de dificuldades na estimativa de fluxo óptico:

- Para cada mudança na intensidade luminosa de uma imagem não há um único movimento que a explique. Esta ambigüidade pode ser mais crítica quando métodos locais são utilizados (definição de método local se encontra no item 2.4.1);

- Movimentos relativos entre o sistema de captação da imagem e o ambiente não são as únicas fontes de mudança de intensidade luminosa. As condições de iluminação, variações nas características das superfícies e ruídos dos sensores podem contribuir também para estas mudanças;

- Modelos físicos, que implícita ou explicitamente estão embutidos no processo de estimativa, são freqüentemente complexos demais para serem usados na prática.

Mesmo satisfazendo as condições para que o fluxo óptico corresponda ao movimento tridimensional, é importante salientar que pode ainda haver perdas de informação por redução dimensional (câmera se afasta da cena – "zoom out") e devido ao *problema da abertura* ("aperture problem"), que faz com que apenas a componente do fluxo óptico na direção do gradiente local da imagem possa ser determinada. Não é possível medir a componente tangencial ao gradiente da imagem (veja figura 2-5).

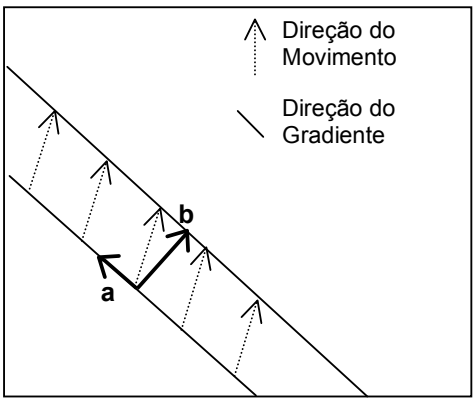


Figura 2-5: *Os vetores a e b representam as componentes normal e perpendicular do movimento. Devido ao problema da abertura, apenas b pode ser estimada.*

O *problema da abertura* já foi exaustivamente estudado e exemplos de métodos que se preocuparam em solucioná-lo podem ser encontrados nos trabalhos de Haralick e Lee [11] e de Ullman [10]. Não há consenso entre os pesquisadores sobre sua relevância na prática, pois há autores que afirmam se tratar de um problema teórico, pouco encontrado em seqüências de imagens reais. De qualquer forma, este problema pode ser melhor entendido através de alguns exemplos (figuras 2-6 e 2-7).

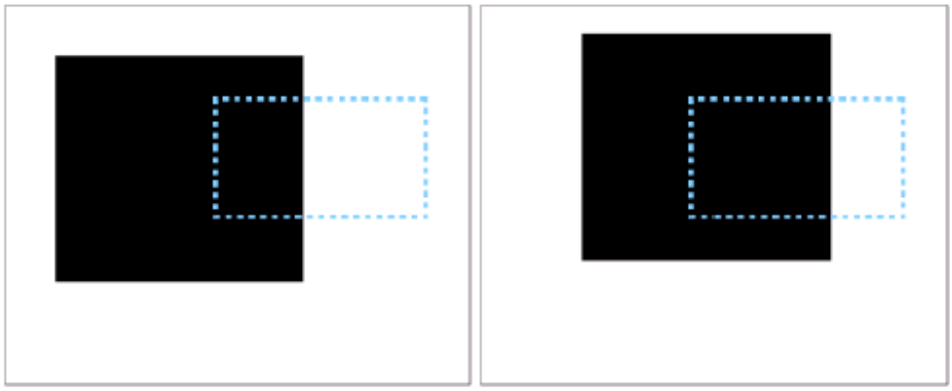


Figura 2-6: *O retângulo pontilhado indica a parte visível da imagem. Mesmo que o quadrado escuro se mova diagonalmente, um observador examinando a parte visível da imagem vê apenas um movimento para a direita.*

A figura 2-6 mostra dois quadros de uma seqüência de imagens. Nesta seqüência, o quadrado preto se move na diagonal (para a direita e para cima). O retângulo pontilhado representa a parte visível da imagem (limitada pela lente que está captando a imagem). Devido a esta limitação, tem-se a impressão de que o quadrado preto está se movendo apenas para a direita. A componente do movimento para cima não pode ser percebida.

A figura 2-7 mostra uma linha que se move diagonalmente (para baixo e para a direita). No primeiro quadrado, o movimento percebido é o representado pelo vetor V_n, pois o campo de visão se encontra limitado pela lente do equipamento que está capturando a imagem (círculo pontilhado). O segundo quadrado mostra o mesmo movimento, mas amplia o campo de visão. Neste caso, sem a limitação da lente, pode-se perceber o movimento real (representado pelo vetor V).

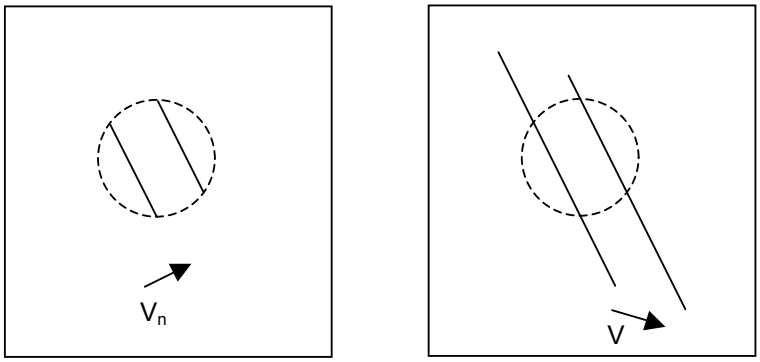


Figura 2-7: *No primeiro quadro vemos uma linha que se move na direção V_n. O círculo representa, por exemplo, a lente de uma câmera, que limita nosso campo de visão. Já no segundo quadro, aumentando nosso campo de visão, percebemos que na verdade a linha se move na direção V.*

Generalizando o *problema da abertura:* em uma região fortemente texturizada, é fácil determinar o fluxo óptico porque há gradientes de valores altos em várias direções em um grande número de pixels. Em uma região de baixa textura, como por exemplo um padrão de cor constante, não há informação local de fluxo e só é possível determinar o fluxo óptico impondo restrições não locais, ou seja, importando informação de regiões mais texturizadas [12].

2.4 Métodos existentes

Muitos métodos para estimativa do fluxo óptico foram propostos e outros continuam a aparecer. Os modelos computacionais existentes para estimativa do fluxo óptico podem ser agrupados em três classes, segundo Beauchemin e Barron [13]: *métodos diferenciais baseados em intensidade, métodos de filtragem baseados em freqüência e métodos baseados em correspondência.* Os limites entre as classes apresentadas não são sempre bem claros, e há métodos que podem ser incluídos em mais de uma classe.

Ao agrupar os diversos métodos de estimativa de fluxo óptico em classes, Barron, Fleet *et al*. [1,2] perceberam que, a despeito de suas diferenças, muitas dessas técnicas podem ser vistas conceitualmente em termos de três estágios de processamento:

1. pré-filtragem ou suavização com filtros passa-baixa/passa-faixa, de modo a extrair a estrutura do sinal de interesse;

2. extração de medidas básicas, como por exemplo, derivadas espaço-temporais;

3. integração destas medidas para produzir um campo de fluxo bidimensional.

A seguir são descritas brevemente as diferentes classes de métodos de estimativa de fluxo óptico. Cada classe tem seus pontos fracos e fortes e apresentam melhor desempenho para um determinado tipo de seqüência de imagens. [2]

[2] Detalhes sobre pontos fortes e fracos das diferentes classes de métodos de estimativa de fluxo óptico , bem como análise de seu desempenho em diversas seqüências de imagens podem ser encontrados nos trabalhos de Aggarwal e Nandhakumar [14], de Smith [15] (análise das classes de métodos de forma generalizada), de Barron, Fleet *et al*. [1,2] e Beauchemin e Barron [13] (análise de métodos específicos).

2.4.1 Métodos diferenciais

Métodos diferenciais calculam a velocidade da imagem através das derivadas espaço-temporais de sua intensidade. O domínio da imagem é considerado contínuo, portanto diferenciável, no espaço e no tempo.

Os métodos diferenciais podem ser globais ou locais, de primeira ou segunda ordem. Métodos globais utilizam, além das derivadas, uma restrição adicional, normalmente um termo de suavização. Métodos locais usam informação das velocidades normais na vizinhança para realizar uma minimização de erro quadrático. Na verdade, o tamanho da vizinhança para obter uma estimativa da velocidade determina se o método é global ou local.

Exemplos de métodos diferenciais: Hildreth [16], Horn e Schunck [8], Lucas e Kanade [3,17] e Nagel [18].

2.4.2 Métodos baseados em freqüência

Uma segunda classe de técnicas de estimativa de fluxo óptico é baseada no uso de *filtros sintonizados em velocidade* (“velocity-tuned filters”). Estas técnicas usam filtros no domínio de Fourier, e conseguem estimar o movimento em imagens para as quais métodos de correspondência (“matching”) falhariam. Por exemplo, pode ser difícil de se detectar o movimento randômico de pontos através de métodos baseados em correspondência, enquanto que no domínio da freqüência, a energia resultante pode ser facilmente extraída para estimar o movimento.

Exemplos de métodos baseados em freqüência: Adelson e Bergen [19], Fleet e Jepson [4], Grzywacz e Yuille [20], Heeger [21] e Watson e Ahumada [22].

2.4.3 Métodos baseados em correspondência ou em características

Diferenciação numérica é muitas vezes impraticável em uma seqüência de poucos quadros ou em sinais com uma relação sinal-ruído muito pobre. Nestes casos, métodos diferenciais ou baseados em freqüência podem não ser apropriados e técnicas de correspondência têm que ser consideradas.

Esta é a classe de métodos mais fácil de ser entendida. O método procura *características* (“features”), como, por exemplo, cantos, bordas e outras estruturas facilmente localizáveis em imagens bidimensionais, e segue estas características à medida que elas se movem quadro a quadro. Este tipo de método envolve dois estágios. Primeiro, escolher algumas características e encontrá-las em dois ou mais quadros consecutivos. Depois, casar (“to match”) estas características entre os quadros. No caso mais simples e comum, dois quadros são usados e dois conjuntos de características são casadas para resultar em um único conjunto de vetores de movimento.

Exemplos de métodos baseados em correspondência: Anandan [23,24], Little, Bulthoff *et al*. [25] e Singh [26,27].

2.5 Estudos comparativos entre os métodos

Barron, Fleet *et al*. [1,2] compararam o desempenho de alguns dos métodos de estimativa de fluxo óptico, enfatizando a precisão e a densidade de medidas. Nove técnicas foram implementadas, incluindo instâncias de métodos diferenciais, métodos baseados em correspondência e métodos baseados em freqüência. Isto permitiu uma comparação do desempenho de técnicas conceitualmente diferentes, bem como uma comparação de diferentes métodos com abordagens conceitualmente similares. Tanto seqüências de imagens reais quanto sintéticas foram utilizadas para testar os métodos.

Os métodos testados foram Horn e Schunck [8], Lucas e Kanade [3,17], Uras, Girosi *et al*. [28], Nagel [29], Anandan [23,24], Singh [26,27], Heeger [21], Waxman, Wu *et al*. [30] e Fleet e Jepson [4].

É importante ressaltar em que condições estes testes de desempenho foram feitos. Em primeiro lugar, assumiu-se que a *sobreposição de espectro* ("aliasing") temporal e espacial não era um problema severo e que as imagens (ou as versões filtradas das imagens) eram diferenciáveis. Em segundo lugar, foram utilizadas seqüências relativamente simples, sem grande quantidade de oclusões, *especularidades*[3] ("specularities"), movimentos múltiplos, etc. Ou seja, as medidas quantitativas de desempenho deveriam ser consideradas como limites superiores da precisão esperada sob condições mais gerais. E finalmente, a maioria das implementações consideradas envolveu apenas uma única filtragem, sendo que múltiplas filtragens poderiam levar a melhores resultados, principalmente no caso de Lucas e Kanade e Fleet e Jepson.

Dos métodos testados, concluiu-se que o mais confiável é o método diferencial de primeira ordem Lucas e Kanade e o segundo mais confiável é o método baseado em freqüência Fleet e Jepson. Em termos gerais, Barron, Fleet *et al.* concluíram que:

- dentre os métodos diferenciais os métodos locais mostraram-se superiores aos métodos globais, tanto em precisão, quanto em eficiência computacional. Além disso, métodos locais são mais robustos com relação a ruídos de quantização.

- métodos diferenciais de segunda ordem são capazes de produzir mapas de fluxo precisos e relativamente densos. Tais métodos só apresentaram um problema de consistência: obtém-se um bom resultado para seqüências de imagens transladadas, enquanto que para deformações geométricas ("zoom in" e "zoom out"), tendem a se degradar mais rapidamente do que os métodos de primeira ordem.

[3] Especularidades são pequenas manchas brilhantes em uma superfície observada, como resultado da interação entre material, forma e ponto de vista. Elas se movem pela superfície à medida em que a câmera se move. Exemplos: superfícies plásticas, metais escovados, pinturas ou roupas brilhantes.

- métodos baseados em freqüência mostraram-se muito sensíveis à presença de sobreposição de espectro na seqüência de imagens por causa da sintonia em freqüência dos filtros utilizados neste tipo de método. Outro ponto fraco apontado foi o alto custo computacional, pois os métodos envolvem um grande número de filtros.

- Métodos baseados em correspondência produzem bons resultados quando tratam de movimentos com altas velocidades. Quando a seqüência de imagens envolve deslocamentos de subpixels (menos de um pixel por quadro), o fluxo estimado é pobre e não corresponde ao fluxo real.

Galvin, McCane *et al.* [5] também conduziram um estudo de desempenho de oito algoritmos de estimativa de fluxo óptico, seis dos quais já haviam sido testados no estudo de Barron, Fleet *et al*. Neste caso, os algoritmos foram testados utilizando-se seqüências de imagens sintéticas, porém mais complexas do que as utilizadas no trabalho de Barron.

Primeiro, foram gerados os valores base dos campos de movimento para as imagens escolhidas utilizando-se um "ray tracer"[4] modificado [31]. Os mapas de fluxo resultantes foram comparados com os mapas obtidos através dos oito algoritmos de estimativa de fluxo óptico estudados.

Os oito métodos de estimativa de fluxo óptico testados nesse estudo de desempenho foram os desenvolvidos por Horn e Schunck [8], Lucas e Kanade [3,17], Uras, Girosi *et al*. [28], Nagel [18,29], Anandan [23,24] e Singh [26,27] (já comparados no trabalho de Barron), além dos métodos propostos por Camus [32] e por Proesmans, Van Gool *et al* [33].

[4] Ray-tracing é uma técnica utilizada para geração de imagens que simula o percurso dos raios luminosos desde o observador até os objetos que os refletem/transmitem (percurso inverso ao verificado na natureza). Algoritmos de Ray-Tracing foram desenvolvidos a partir de um trabalho inicial da Apple publicado em 1968 e são bastante populares para a produção de cenas realistas.

Para a maioria das imagens o algoritmo baseado em Lucas e Kanade apresentou o melhor desempenho. Ele produz, segundo Galvin, McCane *et al*, de forma consistente, mapas de fluxo precisos, tem um baixo custo computacional e boa tolerância a ruído. Seu ponto fraco é a baixa densidade dos mapas de fluxo gerados por ele, em comparação com outros métodos.

De maneira geral, as técnicas diferenciais obtiveram melhores resultados do que as técnicas baseadas em correspondência, apesar de mostrarem-se mais suscetíveis ao problema da abertura. Contudo, nas cenas mais complexas, esta diferença não foi perceptível, indicando que talvez o problema da abertura não seja tão significante na prática quanto na teoria.

MÉTODO			Estudos comparativos			
			Barron, Fleet *et al.*		Galvin, McCane *et al.*	
Classe		Desenvolvido por	testado	resultado	testado	resultado
Diferenciais	1ª Ord.	Horn e Schunck	*		*	bom
		Lucas e Kanade	*	**melhor**	*	**melhor**
		Proesman			*	bom
	2ª Ord.	Nagel	*		*	
		Uras	*	bom	*	
Corresp.		Anandan	*		*	
		Singh	*		*	
		Camus			*	
Frequência		Heeger	*			
		Waxman, Wu e Bergholm	*			
		Fleet e Jepson	*	melhor		

Tabela 2-1: *Resumo dos resultados obtidos nos 2 estudos de desempenho de métodos de estimativa de fluxo óptico. Podemos ver que em ambos estudos o método proposto por Lucas e Kanade apresentou o melhor desempenho.*

Uma vez que o método diferencial desenvolvido por Lucas e Kanade foi o que apresentou melhor desempenho em ambos os estudos, entendemos que seria de grande valia implementá-lo de forma didática, de modo que ele pudesse ser utilizado em aplicações e estudos diversos.

Capítulo 3 – O método das diferenças

Neste capítulo descreveremos o método proposto por Lucas e Kanade, também chamado de método das diferenças. Inicialmente, apresentaremos a idéia básica contida no método das diferenças sem preocupação com o formalismo matemático [34]. Em seguida descreveremos matematicamente o método das diferenças como abordado por Lucas [17]: primeiro em sua formulação unidimensional, com suas variações, e em seguida mostraremos como se evolui para a formulação bidimensional.

3.1 Visão Geral do método das diferenças

Suponha que tenhamos uma imagem com um gradiente espacial de 2 unidades por pixel na direção do eixo x em uma pequena região:

quadro 1

23	25	27	29	31
23	25	27	29	31
23	25	27	29	31

gradiente = 2 unidades/pixel

Agora suponha que a mesma região da imagem no próximo quadro seja:

quadro 2

17	19	21	23	25
17	19	21	23	25
17	19	21	23	25

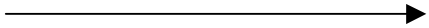


imagem se moveu 3 pixels nesta direção

valores de cada pixel caíram 6 unidades

Ou seja, se o gradiente espacial era 2 unidades/pixel e a imagem foi deslocada em 3 pixels, portanto a mudança total foi de 2 X 3 = 6. Podemos escrever isto de outra forma:

$$\text{distância movida (por quadro)} = \frac{\text{diferença de valores dos pixels}}{\text{gradiente espacial}} = \frac{\text{gradiente temporal}}{\text{gradiente espacial}}$$

Esta abordagem parte de alguns pressupostos:

- as diferenças nos níveis de cinza são devidos somente ao movimento, e não há diferenças devido à iluminação, reflexos etc;
- o fluxo óptico é constante dentro do bloco;
- os deslocamentos são pequenos;

Uma vez compreendido o conceito básico contido no método das diferenças, fica mais fácil entender agora sua formulação matemática.

3.2 Definições

Uma imagem é uma função ***I(p)*** de um vetor ***p***. O vetor ***p*** representa uma posição na imagem e ***I(p)*** contém o valor do pixel naquela posição. Para imagens usuais, ***I(p)*** é uma função com valores escalares, normalmente definida para uma região retangular limitada.

Em estudos de movimento, normalmente existirão duas imagens, denominadas de I_1 e I_2. Freqüentemente existirá a necessidade de se comparar estas duas imagens, portanto uma métrica de diferença entre imagens deverá ser utilizada. A mais comum destas métricas, e a que será utilizada nesta tese, é a norma L2 (ou norma Euclidiana), representada pelo símbolo ***E*** (de erro) e definida por:

$$E = \sum_{p} \left(I_2(p) - I_1(p) \right)^2 \qquad \textit{eq. 3-1}$$

Neste caso, ***p*** varre todos os pontos das regiões das imagens que estão sendo comparadas.

3.3 Correspondência ("Matching")

O problema tradicional de correspondência pode ser definido da seguinte forma: dadas duas imagens $I_1(p)$ e $I_2(p)$, sendo que $I_1(p) = I_2(p+h)$, determinar o vetor h de disparidade entre elas. Em várias situações reais, não é possível atender à relação apresentada, e uma reformulação do problema é necessária: encontrar o vetor h de disparidade de forma que, $I_1(p)$ e $I_2(p+h)$ sejam o mais próximo possível.

Ou seja, queremos encontrar um h que minimiza determinada medida de diferença entre $I_1(p)$ e $I_2(p+h)$. A resolução deste problema nos leva a determinar dois parâmetros globais, que são os componentes h_x e h_y do vetor de disparidade.

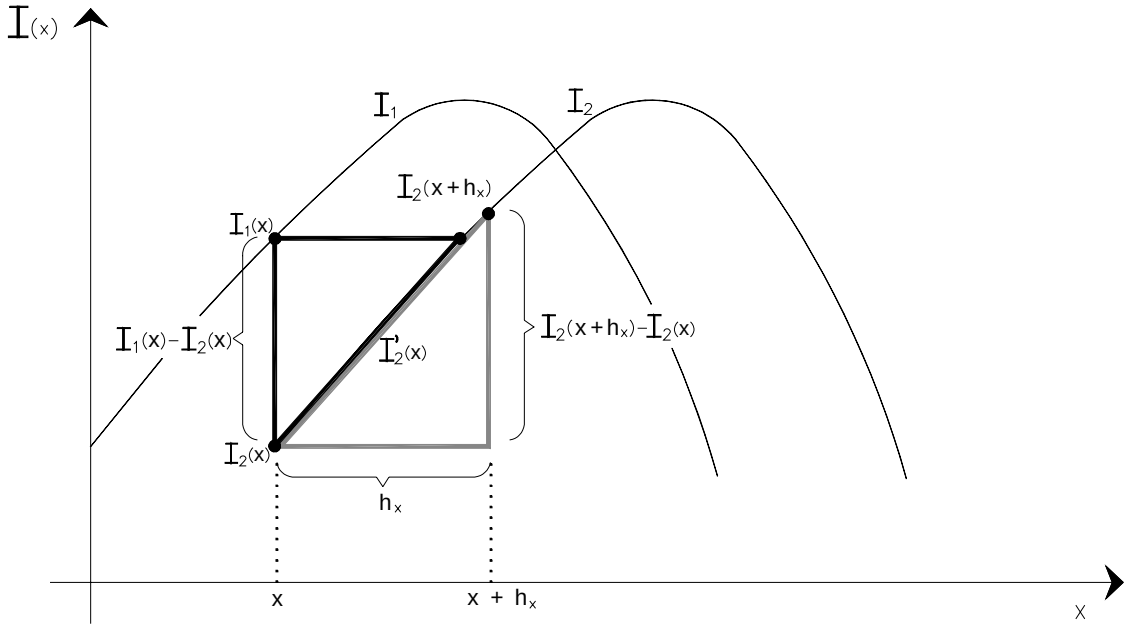


Figura 3-1 : *Ilustração das imagens unidimensionais I_1 e I_2 e as aproximações necessárias para o cálculo da disparidade h_x. Se as curvas I_1 e I_2 fossem contínuas, teríamos um h_x tal que $I_2(x+h_x)=I_1(x)$, porém como x assume valores discretos, $I_2(x+h_x) \cong I_1(x)$ e queremos justamente encontrar h_x que minimiza a diferença.*

3.4 O método das diferenças

A partir dos conceitos apresentados acima podemos entender agora o método das diferenças. O método é baseado na hipótese de que a diferença entre as imagens $I_1(p)$ e $I_2(p)$ em um determinado ponto p pode ser explicada, através de uma aproximação linear, pela disparidade h entre as imagens e pelo gradiente espacial de intensidade da imagem. A relação é dada por:

$$I_1(p) - I_2(p) \approx h_x D_x I_2(p) + h_y D_y I_2(p) \qquad \textit{eq. 3-2}$$

Onde h_x e h_y são os componentes do vetor disparidade h e D_x e D_y denotam diferenciação parcial com relação a x e y.

Como mostra a *eq.3-2*, cada ponto p resulta em uma restrição linear. Já que neste caso estamos resolvendo para dois valores h_x e h_y, precisaremos de pelo menos dois pontos p para obter uma solução única. A figura 3-2 mostra claramente esta questão: para cada mudança na intensidade luminosa não há um único movimento que a explique.

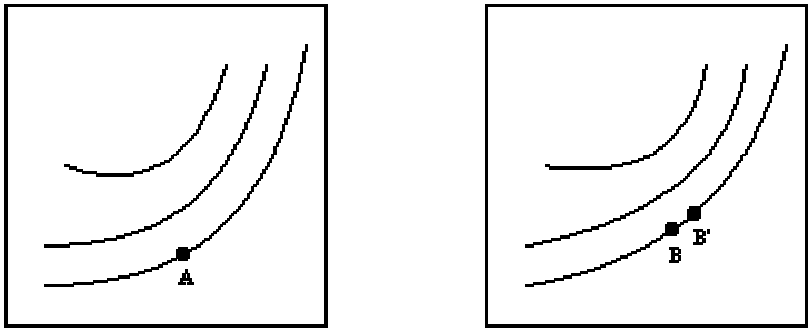


Figura 3-2 : *As linhas são contornos de mesma intensidade. A dificuldade é dizer se a parte da cena representada pelo ponto A na primeira imagem moveu-se para o ponto B ou B', ou ainda para qualquer outro ponto de mesma intensidade na segunda imagem.*

Na prática, uma técnica de minimização do erro quadrático permite combinarmos mais pontos do que parâmetros, minimizando os efeitos de ruído e de alguma forma melhorando a aproximação embutida na *eq.3-2* :

$$\sum_{p}(I_1(p)-I_2(p)-h_x D_x I_2(p)-h_y D_y I_2(p))^2 \qquad \textit{eq. 3-3}$$

A minimização é direta. O conjunto de pontos *p* usados na somatória acima é escolhido de uma das seguintes formas: se estivermos resolvendo em busca de parâmetros globais (parâmetros do movimento), combinamos as informações obtidas de pontos característicos da imagem inteira. Estes pontos característicos devem ser selecionados por estarem próximos às bordas ou em posições pouco afetadas por erros fotométricos, como os causados por ruído ou efeito especular. Se estivermos resolvendo o problema em busca de parâmetros locais, combinamos as informações de cada ponto na vizinhança de um determinado ponto, obtendo assim os parâmetros relativos àquele ponto.

3.5 Algoritmos genéricos

Nesta seção apresentaremos uma série de algoritmos genéricos para estimativa de fluxo óptico utilizando o método das diferenças. Os diferentes algoritmos partem de diferentes pressupostos quanto ao tipo de transformação que modela as mudanças na imagem, e são de complexidade e precisão variáveis.

Como já mencionado anteriormente, os algoritmos apresentados são de dois tipos: *global* ou *local*. Os algoritmos do tipo *global* estão interessados em computar um conjunto de parâmetros globais que descrevem a transformação entre duas imagens, como translação, rotação ou mudança de escala. Já os do tipo *local* estão interessados em calcular parâmetros de transformação similares, só que em escala local. Ou seja, assumimos que uma vizinhança em torno de um ponto na imagem original é mapeada em uma vizinhança em torno de algum ponto da imagem transformada através de uma

transformação do tipo translação, rotação ou mudança de escala. Neste caso, estaremos calculando um campo de parâmetros, como por exemplo um campo de vetores locais representando translações, etc. Na maioria dos casos, um dado algoritmo terá duas versões muito similares, uma *global* e outra *local.*

Freqüentemente a contribuição de cada ponto no cálculo é ponderada por uma função *w(x)*. Possíveis funções de ponderação *w(x)* são discutidas de forma teórica, uma vez que a utilidade desta ponderação ainda é uma questão aberta, segundo Lucas [17].

3.5.1 Algoritmos unidimensionais

3.5.1.1 Ponto-a-ponto

Esta forma do algoritmo é a mais simples e a menos efetiva, mas é importante para entender as versões mais complexas. Ela parte de duas imagens unidimensionais I_1 e I_2, que se relacionam entre si através de:

$$I_1(x) = I_2(x+h) \qquad \textit{eq. 3-4}$$

O objetivo é calcular a disparidade *h*. Utilizando-se uma série de Taylor truncada, obtemos uma estimativa do comportamento da imagem próxima ao ponto *x*:

$$I_1(x) = I_2(x+h) \approx I_2(x) + hI_2^{'}(x)$$

Portanto

$$h \approx \frac{I_1(x) - I_2(x)}{I_2^{'}(x)} \qquad \textit{eq. 3-5}$$

Como a equação acima refere-se a I_2' mas possuímos apenas amostras de I_2 para valores discretos de *x*, obtemos a derivada através de uma diferença. Isto revela a origem do nome "método das diferenças": a diferença entre valores da imagem juntamente com a derivada da imagem, estimada também por uma diferença, leva a uma estimativa da disparidade entre as imagens.

Note que esta diferença depende de *x*, e portanto deveríamos escrever:

$$I_1(x) = I_2(x + h(x)) \approx I_2(x) + h(x)I_2'(x)$$

$$h(x) \approx \frac{I_1(x) - I_2(x)}{I_2'(x)} \qquad \textit{eq. 3-6}$$

A partir da relação acima definimos $\hat{h}(x)$:

$$\hat{h}(x) = \frac{I_1(x) - I_2(x)}{I_2'(x)} \qquad \textit{eq. 3-7}$$

Esta aproximação só pode ser considerada precisa naqueles pontos onde *h* é pequeno ou onde I_2 é praticamente linear.

3.5.1.2 Média

A imprecisão da *eq.3-7* pode ser reduzida tirando uma média de um grupo de valores de *x*. Se estamos calculando um valor global de disparidade para a imagens I_1 e I_2, a média é calculada usando-se todos os valores de *x*:

$$\hat{h} = \frac{1}{N}\sum_x \frac{I_1(x) - I_2(x)}{I_2(x)} \qquad \textit{eq. 3-8}$$

Esta fórmula calcula um parâmetro *h* válido para todo *x*, por isso é dito um algoritmo global. No entanto, se quisermos calcular um "mapa de disparidades", então um algoritmo local é necessário. A versão local da *eq.3-8* é obtida tomando-se:

$$\hat{h} = \frac{1}{N} \sum_{x' \text{próx}.x} \frac{I_1(x) - I_2(x)}{I_2(x)} \qquad \textit{eq. 3-9}$$

Onde *N* é o número de pontos *x'* na vizinhança de *x* (para uma dada definição de vizinhança). Isto corresponde a calcular a *eq.3-8* para cada *sub-imagem* de I_1 e I_2 em torno de *x*. Na verdade, a *eq.3-7* e a *eq.3-8* são casos limites da *eq.3-9*, onde a *vizinhança de x* passa a ser *apenas o ponto x* (*eq.3-7*) ou *todos os pontos* (*eq. 3-8*).

3.5.1.3 Mínimo erro quadrático

Infelizmente, a disparidade calculada pelos algoritmos apresentados até aqui sofre de dois problemas. Primeiro, não há nenhum jeito óbvio de generalizá-los para duas dimensões; a aproximação linear bidimensional correspondente a *eq. 3-2* é:

$$I_1(x,y) \approx I_2(x,y) + h_x(x,y) D_x I_2(x,y) + h_y(x,y) D_y I_2(x,y) \qquad \textit{eq. 3-10}$$

Onde h_x e h_y são as componentes *x* e *y* do campo de disparidade e D_x e D_y são os operadores de derivação parciais em *x* e *y*. Esta equação fornece uma restrição linear em $h_x(x,y)$ e $h_y(x,y)$ em cada ponto. Infelizmente a *eq. 3-10* não tem solução única para $h_x(x,y)$ e $h_y(x,y)$. Ou seja, uma restrição é insuficiente para determinar duas componentes. Nas técnicas de estimativa de fluxo óptico este problema é normalmente resolvido acrescentando-se uma restrição adicional a h_x e h_y, como por exemplo, h_x e h_y devem satisfazer algum critério de suavização.

Além deste problema, há o fato de que $\hat{h}(x)$ é apenas uma aproximação de $h(x)$ quando $I_1(x)=I_2(x+h(x))$, e não é exatamente o que está sendo calculado quando $I_1(x)$ é apenas uma aproximação de $I_2(x+h(x))$.

Ambos problemas são resolvidos por uma abordagem um pouco diferente. Suponhamos que (no caso unidimensional) o objetivo é calcular para cada ponto x uma disparidade estimada $\hat{h}(x)$ que minimiza alguma medida de diferença entre $I_1(x)$ e $I_2(x+\hat{h}(x))$ para cada x' em uma pequena vizinhança de x. Isto é, imaginemos calcular uma disparidade $\hat{h}(x)$ que faz a sub-imagem de I_1 próxima a x tão similar à sub-imagem de I_2 próxima a $x+\hat{h}(x)$ quanto possível ou seja , devemos calcular para cada x a disparidade $\hat{h}(x)$ que minimiza o erro:

$$E_x = \sum_{x' próx.x} (I_2(x'+\hat{h}(x)) - I_1(x'))^2 \qquad \textit{eq. 3-11}$$

Para encontrar o mínimo da função diferenciamos com relação a $\hat{h}(x)$ e igualamos a zero, obtendo:

$$\hat{h}(x) = \frac{\sum_{x' próx.x} (I_1(x') - I_2(x')) I_2'(x')}{\sum_{x' próx.x} I_2'(x')^2} \qquad \textit{eq. 3-12}$$

3.5.1.4 Mínimo erro quadrático ponderado

O efeito dos pontos de estimativa imprecisa no resultado final pode ser reduzido ou eliminado. A questão é tentar imaginar o que causou este efeito, para então escolher uma função de ponderação $w(x)$ de forma a minimizá-lo. O que Lucas sugere em sua tese [17] é uma função $w(x)$ que deve ser grande quando a diferença entre as derivadas de I_1 e I_2 é pequena e deve ser pequena quando a diferença é grande.

3.5.2 Algoritmos bidimensionais

3.5.2.1 Mínimo erro quadrático

No caso bidimensional, o escalar *x* é substituído pelo vetor bidimensional *p*:

$$p = \lfloor p_x \quad p_y \rfloor$$

A *eq. 3.6* pode ser reescrita da seguinte forma:

$$I_1(p) = I_2(p + h(p)) \approx I_2(p) + h_x(p) D_x I_2(p) + h_y(p) D_y I_2(p) \qquad \textit{eq. 3-13}$$

onde D_x significa $\partial / \partial p_x$ e D_y significa $\partial / \partial p_y$. A versão bidimensional da *eq. 3.11* é:

$$Ep = \sum_{p^1 \, próx.\, p} (I_2(p' + h(p)) - I_1(p'))^2 \qquad \textit{eq.3.14}$$

Substituindo a *eq. 3.13* em *3.14* chegamos ao erro estimado:

$$Ep \approx \sum_{p^1} (I_2(p') + h_x(p) D_x I_2(p') + h_y(p) D_y I_2(p') - I_1(p))^2 \qquad \textit{eq.3.15}$$

Diferenciando com relação a $h_x(p)$ e $h_y(p)$ chegamos a um sistema de duas equações com duas incógnitas:

$$\begin{cases} h_x(p) \Sigma_1 + h_y(p) \Sigma_2 + \Sigma_3 = 0 \\ h_x(p) \Sigma_4 + h_y(p) \Sigma_5 + \Sigma_6 = 0 \end{cases}$$

$$h_x(p) = \frac{\Sigma_5 \Sigma_3 - \Sigma_2 \Sigma_6}{\Sigma_2^2 - \Sigma_1 \Sigma_5} \qquad \textit{eq. 3-16}$$

$$h_y(p) = \frac{\Sigma_2 \Sigma_3 - \Sigma_1 \Sigma_6}{\Sigma_2^2 - \Sigma_1 \Sigma_5} \qquad \textit{eq. 3-17}$$

onde

$$\Sigma_1 = \sum_{p' prox.p} (D_x I_2(p'))^2 \qquad \Sigma_3 = \sum_{p' prox.p} (I_2(p') - I_1(p')) D_x I_2(p')$$

$$\Sigma_5 = \sum_{p' prox.p} (D_y I_2(p'))^2 \qquad \Sigma_6 = \sum_{p' prox.p} (I_2(p') - I_1(p')) D_y I_2(p')$$

$$\Sigma_2 = \Sigma_4 = \sum_{p' prox.p} D_x I_2(p') D_y I_2(p')$$

3.5.2.2 Mínimo erro quadrático ponderado

Assim como no caso unidimensional, o algoritmo de erro quadrático mínimo pode ser melhorado usando-se uma função de ponderação *w*.

*Capítulo 4 – Valores aberrantes (“*Outliers”)

É inevitável em um trabalho científico experimental, que alguns dos dados obtidos não tenham o comportamento esperado. Estes dados, com valores não usuais, ou seja, valores completamente diferentes do restante dos dados, são chamados *valores aberrantes* (“outliers”). As fontes prováveis de *valores aberrantes* são: erros (de captação dos dados, de análise dos dados, de cálculo etc) e eventos raros.

Qualquer que seja a origem dos *valores aberrantes*, é necessário identificá-los e, às vezes, até segregá-los, para que o resultado da pesquisa não seja prejudicado por eles. Na presença de valores aberrantes, testes estatísticos baseados em médias e variâncias ficam distorcidos, assim como estimativas de coeficientes de regressão que minimizam a soma de erros quadráticos.

4.1 Definição de valores aberrantes

Observações que apresentam um grande afastamento das restantes ou são inconsistentes com elas são habitualmente designadas por aberrantes. Um valor aberrante é caracterizado pela sua relação com as restantes observações que fazem parte da amostra. O seu distanciamento em relação a estas observações é fundamental para fazer sua caracterização. Essas observações são também designadas por observações “anormais”, “contaminantes”, “estranhas” ou “extremas” [35].

4.2 Identificação de valores aberrantes

Primeiro é necessário visualizar os dados de forma a identificar algum tipo de distribuição ou padrão. Algumas ferramentas de visualização de dados podem ser bastante úteis, como por exemplo um *gráfico de dispersão* (“scatterplot”) ou um *gráfico Q-Q* (“Quantile-Quantile Plots”). Um *gráfico de caixa* (“boxplot”) também é uma ferramenta muito útil, pois não assume nenhuma distribuição específica nem tampouco requer uma estimativa inicial de média ou desvio padrão. Valores extremos em relação ao resto dos dados são facilmente identificados.

4.2.1 Identificação através do gráfico de caixa

Construir um gráfico de caixa é a forma mais usual para se identificar valores aberrantes. Para construir um gráfico de caixa é necessário conhecer o primeiro, o segundo e o terceiro quartil[5] dos dados. Estes valores definem a *caixa*. O terceiro quartil é a linha superior da caixa, o primeiro quartil é a linha inferior da caixa e o segundo quartil (ou seja, a mediana) é representada por uma linha desenhada no meio da caixa.

[5] Quartis são valores que dividem os dados em 4 grupos contendo um número igual de observações. Um conjunto de dados ordenados possui 3 quartis, onde o 2º quartil é igual à mediana, o 1º quartil é o dado entre o primeiro valor e a mediana (25% dos dados estão abaixo deste valor e 75% estão acima) e o 3º quartil é o dado entre a mediana e o último valor (75% dos dados estão abaixo deste valor e 25% estão acima).

A caixa também tem *bigodes* (“whiskers”) que se estendem acima e abaixo da caixa. O máximo comprimento de cada bigode é normalmente de 1,5 vez a distância interquartil (IQR)[6]. Então, o bigode acima da caixa é desenhado até o dado que não excede 1,5 vez IQR a partir do 3º quartil. Qualquer dado maior do que este valor deve ser marcado como um valor aberrante. O mesmo princípio se aplica para o bigode abaixo do 1º quartil: qualquer dado menor do que 1,5 vez IQR marcado abaixo do 1º quartil é considerado também um valor aberrante [36].

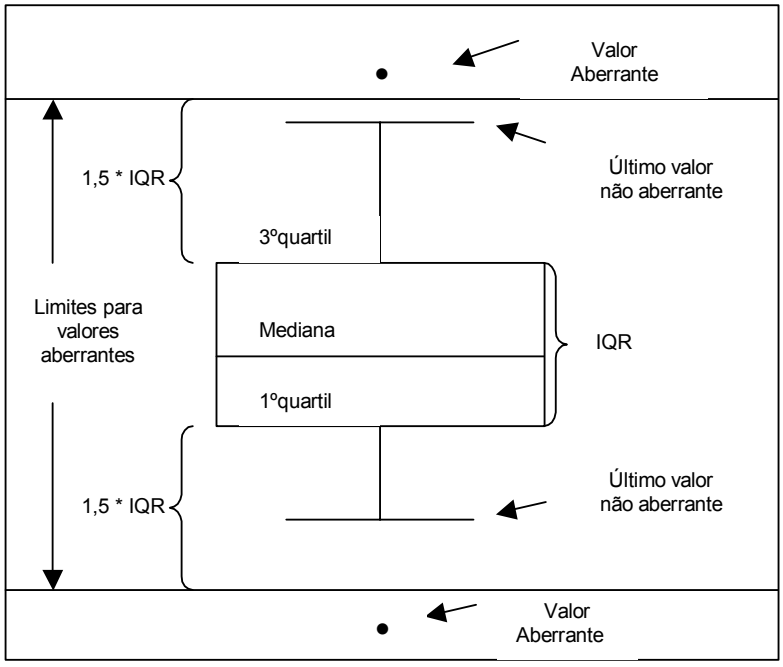


Figura 4-1 : *Esquema de um gráfico de caixa. Nele podemos identificar todos os elementos deste tipo de gráfico: a caixa, com o 1º e o 3º quartis, o 2º quartil (ou a mediana), os bigodes e dois exemplos de valores considerados aberrantes (um maior do que bigode superior e um menor do que o bigode inferior).*

[6] A distância inter-quartil (IQR), como o nome já diz, é a distância entre o 1º e o 3º quartis. É razoavelmente simples calcular IQR e então usar um múltiplo dela para definir que valores serão considerados aberrantes. O valor típico utilizado é de 1,5 vez IQR.

4.2.2 Identificação através do gráfico Q-Q

O gráfico Q-Q é uma técnica gráfica para determinar se dois conjuntos de dados vem de populações com distribuições similares. É um gráfico dos quantis[7] do primeiro conjunto de dados contra os quantis do segundo conjunto de dados. Uma linha de referência de 45 graus também é plotada. Se os dois conjuntos vem de uma população com a mesma distribuição, os pontos deveriam cair próximos à linha de referência.

Este tipo de gráfico é útil também para detectar valores aberrantes. Em um gráfico de dois conjuntos de dados com a mesma distribuição (todos os pontos próximos à linha de referência), os valores aberrantes aparecem como pontos que estão muito longe dos demais (e da curva de referência) [37,38].

A figura 4-2 mostra um exemplo de gráfico Q-Q. Vemos que ele é formado por:

- Eixo horizontal – quantis estimados para o conjunto de dados 1

- Eixo vertical - quantis estimados para o conjunto de dados 2.

Ambos os eixos estão nas unidades de seus respectivos dados. Ou seja, para cada ponto no gráfico Q-Q, sabemos que o quantil é o mesmo para os dois conjuntos, mas não qual o valor real do quantil. Como os dois conjunto de dados podem não ter a mesma quantidade de valores, o número de valores plotados no gráfico é igual à quantidade de dados da menor amostra.

[7] Por quantil entende-se a fração (ou percentual) de pontos abaixo de um determinado valor. Ou seja, o quantil 0.3 (ou 30%) é o ponto onde 30% dos dados estão abaixo deste valor e 70% dos dados estão acima deste valor.

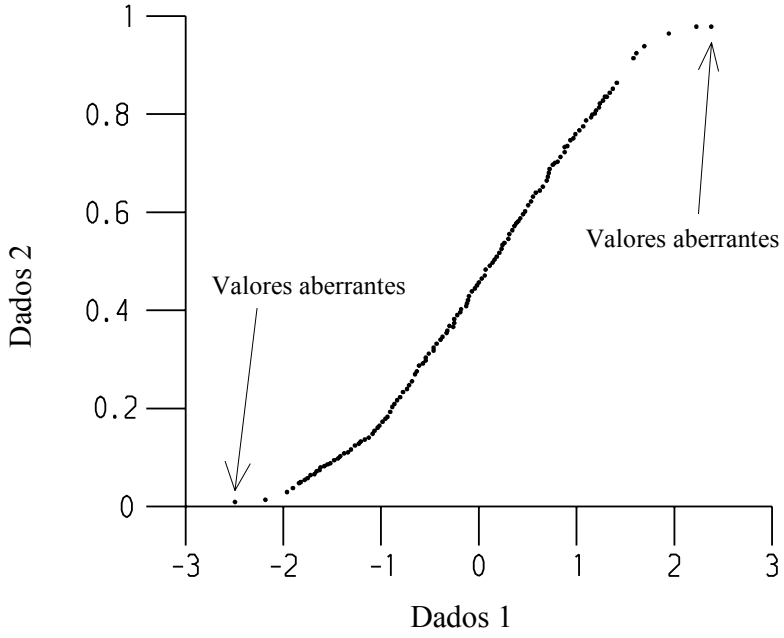


Figura 4-2 : *Um exemplo de gráfico Q-Q, com a identificação de possíveis valores aberrantes, tanto no início quanto no final da curva.*

4.3 Tratamento dos valores aberrantes

Uma vez identificados os valores aberrantes, a questão é decidir o que fazer com eles. Nem ignorá-los nem apagá-los parecem ser boas soluções. Por outro lado, se não fizermos nada e mantivermos os valores aberrantes junto com os demais dados, os resultados obtidos através destes dados ficarão comprometidos [35, 39].

Algumas possibilidades de tratamento de valores aberrantes:

Transformação – transformar os dados é uma maneira de suavizar o impacto dos valores aberrantes, uma vez que raízes quadradas ou logaritmos, por exemplo, reduzem grandes valores muito mais do que reduzem pequenos valores. Contudo, as transformações podem não se encaixar à teoria ou o modelo que está sendo estudado. Além disso, as transformações mais utilizadas requerem dados não negativos ou maiores do que zero, e nem sempre são a solução adequada.

Acomodação – Outra saída é utilizar para a análise de dados métodos robustos com relação a valores aberrantes. Métodos estatísticos não-paramétricos se encaixam nesta categoria e podem ser utilizados na presença de valores aberrantes.

Eliminação – Apenas como último recurso deve-se eliminar os dados identificados como valores aberrantes, e somente se eles forem proveniente de erros que não podem ser corrigidos ou estão tão longe dos demais dados que distorceriam qualquer inferência estatística. Às vezes, pode ser interessante construir o modelo com e sem os valores aberrantes, para poder comparar o efeito destes últimos no estudo que está sendo desenvolvido.

Capítulo 5 – O método proposto

No Capítulo 3 descrevemos o método das diferenças e suas várias instâncias: *ponto-a-ponto*, *média*, *mínimo erro quadrático* e *mínimo erro quadrático ponderado*. Baseados nos estudos já realizados a respeito do desempenho deste método citamos alguns de seus pontos fracos e limitações. Neste capítulo apresentamos uma proposta de melhoria do método das diferenças, utilizando a identificação e transformação de valores aberrantes (descrita no capítulo 4) como medida de confiabilidade.

5.1 Medidas de confiabilidade

Como já visto anteriormente existe uma grande variedade de métodos de estimativa de fluxo óptico. O estudo de Barron, Fleet *et al*. [1,2] escolheu alguns destes métodos para comparar desempenhos, métodos cujos campos de movimento produzidos apresentaram uma precisão que varia drasticamente dependendo da estrutura do sinal e do movimento bidimensional contido na seqüência de imagens utilizada.

Como conclusão de seu trabalho, Barron, Fleet *et al*. defenderam fortemente a necessidade de se utilizar medidas de confiabilidade, ou seja, alguma maneira de determinar a integridade e precisão das velocidades estimadas. Essas medidas devem ser usadas como valores limites (“thresholds”) para extrair subconjuntos de velocidades estimadas. Outra possibilidade é lançar mão desta medida de confiabilidade para dar peso às velocidades já calculadas.

Estas técnicas parecem funcionar bem, pois permitem isolar os resultados mais confiáveis, descartando ou dando menor peso aos dados menos confiáveis. Segundo o estudo de Barron, elas produzem mapas de fluxo mais esparsos, porém mais precisos. Todos os algoritmos que apresentaram os melhores desempenhos em seu estudo tinham em comum o fato de fazer uso de alguma medida de confiabilidade.

Algumas das medidas de confiabilidade sugeridas por alguns autores e testadas por Barron, Fleet *et al*. [1,2] e Beauchemin e Barron [13] foram:

- o menor autovalor da matriz de mínimos quadrados [40];
- o determinante da matriz de Hessian [30];

- a magnitude de gradientes locais da imagem [24];

- os autovalores da matriz de covariância [27].

Todas estas medidas traduzem de alguma forma erros contidos nas estimativas de fluxo óptico. Por exemplo: o menor autovalor da matriz de mínimos quadrados, segundo Barron, reflete a ocorrência do problema da abertura, bem como erros significativos nas medidas do gradiente.

No presente trabalho, propomos uma abordagem diferente. Partindo do fato de que o método das diferenças não é capaz de estimar o fluxo óptico de forma confiável quando há variações de luminosidade entre um quadro e outro da seqüência de imagens, oclusões ou ainda regiões de baixa textura, escolhemos o conceito estatístico de valores aberrantes, já apresentado no capítulo anterior, para identificar estes valores não confiáveis. Uma vez tendo identificado os valores aberrantes, definimos qual o tratamento que deverá ser dado a eles, de forma a obter um mapa de fluxo para a imagem toda que não tenha sofrido a influência destes valores.

5.2 Identificação dos valores aberrantes

Como apresentado no Capítulo 4, é inevitável em um trabalho científico experimental que alguns dos dados obtidos não tenham o comportamento esperado. Estes dados, com valores não usuais, ou seja, valores completamente diferentes do restante dos dados, são chamados valores aberrantes. Qualquer que seja a origem dos valores aberrantes é necessário identificá-los e, às vezes, até segregá-los, para que o resultado da pesquisa não seja prejudicado por eles. Na presença de valores aberrantes, testes estatísticos baseados em médias e variâncias ficam distorcidos.

Quando lidamos com estimativa de fluxo óptico em seqüências de imagens, as fontes prováveis de valores aberrantes nos mapas de fluxo óptico são: reflexos, sombras, transparência, oclusões, pouca textura, problema da abertura etc. E para que tenhamos mapas de fluxo confiáveis, precisamos identificar estes valores, independente de sua origem, e transformá-los ou segregá-los. Desta forma, estaremos buscando obter mapas de fluxo mais precisos, não só para imagens sintéticas, mas principalmente para imagens reais, que apresentam uma ou mais das fontes de erro citadas acima.

Na implementação do método das diferenças, portanto, após a estimativa de fluxo óptico pelo algoritmo ponto-a-ponto, incluímos uma etapa de identificação de valores aberrantes. A intenção é identificar quais os pixels cujos gradientes estão nos levando a uma estimativa de fluxo óptico absurda. Uma vez identificados os pixels "problemáticos", adotamos alguma medida para corrigir seus respectivos gradientes, para depois estimar o fluxo óptico através de qualquer uma das instâncias do método das diferenças.

Escolhemos o gráfico de caixa para nos auxiliar nesta etapa de identificação dos valores aberrantes, pois como explicado no Capítulo 4, é uma técnica simples e bastante utilizada para este fim.

5.3 Tratamento dos valores aberrantes

Conforme descrito no Capítulo 4, uma vez identificados os valores aberrantes, há basicamente três tratamentos possíveis: *transformação*, *acomodação* ou *eliminação*.

A *acomodação* foi a primeira opção de tratamento a ser descartada, pois pressupõe a utilização de métodos estatísticos para análise dos dados, que sejam robustos em relação aos valores aberrantes. Desta forma os valores identificados como aberrantes poderiam ser mantidos juntamente com os demais. Mas como nosso interesse nos

dados é sua visualização gráfica e não sua análise estatística, não nos pareceu de grande valia adotar este tipo de tratamento.

A *eliminação* dos valores aberrantes também foi descartada, uma vez que, como já dito no cap. 4, deve ser utilizada apenas como último recurso. No nosso caso, estaríamos eliminando um número diferente de pontos para cada bloco, o que nos deixaria com um número variável de pontos para recalcular o fluxo óptico através de outro algoritmo.

Fizemos a opção, portanto, pela *transformação* dos dados, de forma a suavizar o impacto dos valores aberrantes. E, para transformar os fluxos considerados aberrantes, alteramos os valores dos gradientes espaciais, pois são eles as origens das distorções. Alterando os valores dos gradientes espaciais para cada pixel cujo vetor de fluxo correspondente foi considerado um valor aberrante, estamos alterando também os novos fluxos estimados através de qualquer algoritmo, já que todos eles baseiam-se nos gradientes espaciais.

Capítulo 6 – Implementação do método Lucas e Kanade

Como já dito anteriormente, a escolha do método a ser implementado foi baseada em dois estudos que compararam diversos métodos de estimativa de fluxo óptico e concluíram que o método desenvolvido por LK apresentou o melhor desempenho. Uma vez que estes estudos basearam-se principalmente em resultados obtidos com a análise de imagens sintéticas, este trabalho propõe uma modificação do algoritmo LK, com o objetivo de obter resultados mais confiáveis, não só para seqüências de imagens sintéticas, mas principalmente em seqüências de imagens reais.

6.1 Decisões de projeto

6.1.1 Algoritmos a serem implementados

Como apresentado no Capítulo 3, o método das diferenças é baseado na idéia de encontrar a velocidade da imagem através de seus gradientes espaciais e temporais. Então, em sua forma básica, bastaria calcular os gradientes espaciais e temporais para todos os pixels e fazer a divisão de um pelo outro para obtermos a estimativa de fluxo óptico. Por estimar um vetor de fluxo para cada pixel da imagem, esta primeira instância do método das diferenças é conhecida como *ponto-a-ponto*. Mas esta versão do método tem suas deficiências, então recorre-se a estratégia de calcular a média dos fluxos estimados para toda a vizinhança do ponto. Esta nova versão, denominada de algoritmo da *média*, não resolve ainda a questão da disparidade calculada ser apenas uma aproximação da disparidade real, fato que pode ser melhorado utilizando-se outra medida de vizinhança que não a média, como por exemplo o *mínimo erro quadrático*.

Para entendermos melhor esta evolução do método das diferenças, implementamos todas suas variantes. Desta forma, utilizando uma mesma seqüência de imagens, podemos observar o comportamento dos mapas de fluxo obtidos para cada uma destas instâncias.

O último algoritmo implementado é uma variante do método das diferenças proposta neste trabalho, que utiliza o conceito de valores aberrantes como medida de confiabilidade, conforme já apresentada no Capítulo anterior.

6.1.2 Ferramentas de trabalho

Todos os algoritmos foram implementados utilizando-se o ambiente de desenvolvimento do MATLAB. A escolha desta linguagem se deve ao fato da mesma apresentar uma *caixa de ferramentas* (“toolbox”) de Processamento de Imagens que nos permite

manipular arquivos de imagens de forma rápida e ágil. Com o uso desta caixa de ferramentas é possível redimensionar, rotacionar e alterar o padrão de cores das imagens. A estrutura de matrizes multidimensionais é um recurso interessante para quem precisa trabalhar com seqüências de imagens de uma única vez. E como estas seqüências de imagens são obtidas de diversas fontes distintas, o fato do MATLAB ser capaz de trabalhar com diversos formatos de arquivos nos poupa do inconveniente de converter arquivos, o que pode trazer perdas para o processamento destas imagens.

Além de todas as facilidades já citadas anteriormente, podemos acrescentar também que o MATLAB disponibiliza um ambiente flexível e descomplicado para testar idéias e algoritmos, com recursos gráficos interessantes para comparar resultados e apresentá-los. Os algoritmos desenvolvidos com o uso do MATLAB ficam gravados em *arquivos-m* ("m-files") que podem ser compartilhados com outros usuários e utilizados em conjunto ou separadamente por outras aplicações.

6.1.3 Metodologia

O MATLAB trabalha com dois tipos de arquivos-m[8]: *function* e *script* [41].

Script

Scripts são o tipo mais simples de arquivo-m, pois não possuem nem entrada nem saída. Eles são úteis para automatizar uma seqüência de comandos, como por exemplo, cálculos repetitivos e testes a serem executados. Scripts operam sobre os dados existentes no ambiente de trabalho do MATLAB, ou podem criar novos dados ou variáveis. Qualquer variável que um script cria na área de trabalho, permanece na mesma após o término da execução do script, para que possa ser usada em cálculos subseqüentes.

[8] Arquivo-m é um arquivo texto contendo todos os comandos que seriam digitados na linha de comandos do MATLAB. Pode ser criado usando-se o editor do MATLAB ou outro editor de texto. A denominação "Arquivo-m" vem do fato deste tipo de arquivo receber um nome + a extensão .m .

Function

> Functions são arquivos-m que aceitam argumentos de entrada e retornam argumentos de saída. Elas operam nas variáveis dentro de seu ambiente, separado da área de trabalho. Variáveis criadas por uma função não ficam acessíveis após a mesma terminar sua execução, a menos dos argumentos de saída.

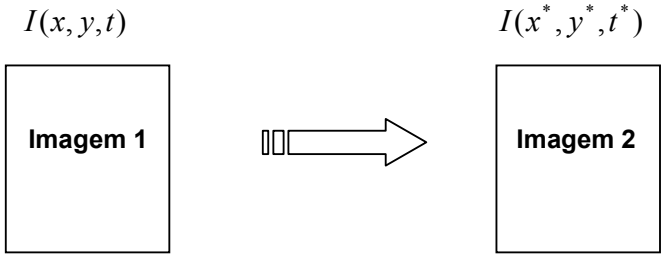
Para melhor organização e entendimento do sistema desenvolvido, optamos por implementar cada etapa do algoritmo através de uma *function* desenvolvida no MATLAB. Desta forma, as *functions* podem ser estudadas de forma isolada por aqueles que quiserem entender apenas uma etapa do processamento.

Já a etapa de testes do sistema foi implementada através de *scripts*, de forma a permitir a escolha de valores diferentes para as variáveis de testes, sem necessidade de se alterar as *functions* dos algoritmos. Para se executar uma seqüência completamente diferente de testes, basta escrever um novo *script*.

6.2 Formulação matricial do método das diferenças

Antes de iniciarmos a descrição de como foi feita a implementação do método das diferenças, vamos reescrever as equações do Capítulo 3 na forma matricial. Primeiro, para que possamos ter uma visão clara de como o método manipula as imagens de forma a estimar o fluxo óptico, já que as imagens são representadas por matrizes. Segundo, para conseguir estabelecer uma relação direta entre a teoria do método e os algoritmos desenvolvidos em MATLAB.

Dadas duas imagens de uma seqüência:



Vamos supor que a intensidade luminosa *I(x,y,t)* é deslocada por uma distância *dx,dy* em um intervalo *dt*. Isto significa que a intensidade luminosa de cada ponto não varia a medida que ele se desloca. Podemos escrever, portanto, para um ponto *(x,y)* no instante *t*:

$$I(x^*, y^*, t^*) = I(x, y, t)$$

$$I(x + dx, y + dy, t + dt) = I(x, y, t)$$

Expandindo o lado esquerdo em Série de Taylor, e desprezando os termos de ordens superiores, temos:

$$I(x, y, t) + dx\frac{\partial I}{\partial x} + dy\frac{\partial I}{\partial y} + dt\frac{\partial I}{\partial t} = I(x, y, t) \qquad \textit{eq. 6-1}$$

Podemos agora fazer as seguintes substituições na eq. 6-1:

$$u = \frac{dx}{dt} \text{ e } v = \frac{dy}{dt}$$

e

$$E_x = \frac{\partial I}{\partial x} \ , \ E_y = \frac{\partial I}{\partial y} \text{ e } E_t = \frac{\partial I}{\partial t}$$

onde E_x é o gradiente espacial em x, E_y é o gradiente espacial em y e E_t é o gradiente temporal. Temos então:

$$E_x u + E_y v + E_t = 0 \qquad \textit{eq 6-2}$$

Para estimarmos o fluxo óptico ponto-a-ponto basta fazermos a divisão direta:

$$u = {E_t}/{E_x} \text{ e } v = {E_t}/{E_y} \qquad \textit{eq 6-3}$$

Da mesma forma, para obtermos um resultado mais apurado através da média dos valores dos gradientes calculados para os pontos de um bloco da imagem, temos o seguinte cálculo:

$$u = \frac{1}{n}\sum_{bloco} {E_t}/{E_x} \text{ e } v = \frac{1}{n}\sum_{bloco} {E_t}/{E_y} \qquad \textit{eq 6-4}$$

onde n é o número de pontos do bloco (por exemplo: em um bloco 8X8, n=64).

Mas, se levarmos em conta que as imagens são discretas, tanto no espaço como no tempo, não existem x^*,y^* e t^* inteiros , tal que $I(x^*, y^*, t^*) = I(x, y, t)$. Então, o erro pode ser escrito como

$$E(x, y, t) = \sum_{x,y} [E_x u + E_y v + E_t]^2$$

Como queremos minimizar o erro, vamos encontrar o mínimo da função *E(x,y,t)*, diferenciando-a em relação a *u* e *v*:

$$\sum_{x,y} 2E_x^2 u + 2E_x E_y v + 2E_x E_t = 0$$

$$\sum_{x,y} 2E_y^2 v + 2E_x E_y u + 2E_y E_t = 0$$

Reescrevendo as equações acima na forma matricial, chegamos ao cálculo do fluxo óptico pelo algoritmo do mínimo erro quadrático:

$$\begin{bmatrix} \sum E_x^2 & \sum E_x E_y \\ \sum E_x E_y & \sum E_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum E_x E_t \\ \sum E_y E_t \end{bmatrix} \qquad \textit{eq 6-5}$$

Pode-se ainda lançar mão de uma função de ponderação *W*, que dará mais peso aos valores mais confiáveis e menos peso aos valores não confiáveis. Esta definição de valores confiáveis ou não confiáveis é baseada na observação de possíveis fontes de erro, como por exemplo, regiões (ou pontos) onde os gradientes são muito pequenos.

A formulação acima, acrescentando-se a máscara *W*, fica então:

$$\begin{bmatrix} \sum W^2 E_x^2 & \sum W^2 E_x E_y \\ \sum W^2 E_x E_y & \sum W^2 E_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum E_x E_t \\ \sum E_y E_t \end{bmatrix} \qquad \textit{eq 6-6}$$

Na prática[9]:

- Calcula-se os gradientes espaciais E_x e E_y , utilizando-se para isso uma máscara Gaussiana (Apêndice 1);

[9] Seqüência baseada nas anotações de Calway [42].

- Em seguida calcula-se o gradiente temporal E_t, subtraindo-se um quadro da seqüência de imagens do quadro anterior;

- Monta-se a matriz:

$$A = \sum_{Bloco} (\nabla E(x,y))(\nabla E(x,y))^T$$

onde

$$\nabla E(x,y) = [E_x, E_y]^T$$

- Calcula-se o inverso da matriz A (A^{-1})

- Calcula-se:

$$b = -\sum_{Bloco} \left[E_x E_t, E_y E_t\right]^T$$

- Por último, estima-se o movimento, através da multiplicação matricial:

$$\hat{v} = A^{-1} b$$

Importante: note que *A* se torna uma matriz singular (portanto, não inversível) quando $\nabla E(x,y)$ é constante ao longo do bloco (insuficiente variação de intensidade luminosa, o que não permite a determinação do movimento).

6.3 Etapas de processamento

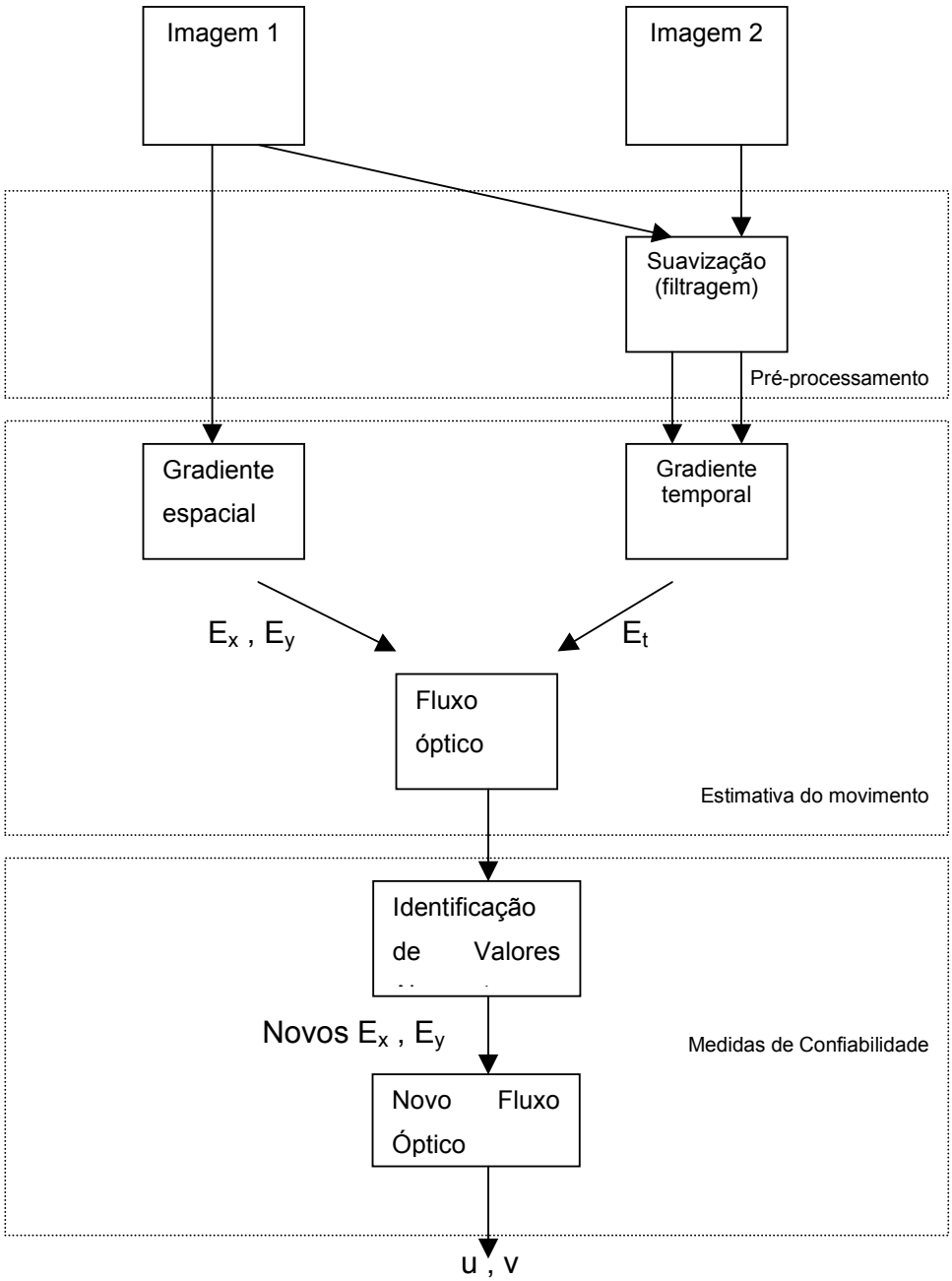


Figura 6-1 : *O sistema desenvolvido consiste basicamente de uma etapa de pré-processamento, seguida da etapa de estimativa de movimento propriamente dita e por fim de uma etapa de identificação e transformação de dados não confiáveis.*

6.3.1 Pré-processamento

Antes de iniciarmos a estimativa de movimento propriamente dita, as imagens foram suavizadas através de uma filtragem utilizando-se uma Gaussiana (Apêndice 1). Esta suavização é descrita por Lucas [17] como uma etapa fundamental para o cálculo do gradiente temporal das imagens. Sem ela, o pressuposto de linearidade adotado na aproximação por uma Série de Taylor não é garantido (as imagens devem ser localmente lineares), o que pode introduzir erros significativos na estimativa de fluxo óptico. Beauchemin e Barron [13] concluíram em seu trabalho que uma suavização da seqüência de imagens utilizando uma Gaussiana traz resultados mais precisos para o método LK.

6.3.2 Estimativa do movimento

Para cada instância do algoritmo: ponto-a-ponto, média, mínimo erro quadrático, mínimo erro quadrático ponderado e mínimo erro quadrático com identificação e transformação de valores aberrantes, uma função de estimativa de fluxo óptico foi gerada. Todas seguem a mesma seqüência de cálculo, diferindo entre si apenas no final: inicialmente os gradientes espaciais (E_x e E_y) são calculados para a primeira imagem (I_1). Em seguida, calcula-se o gradiente temporal, através da diferença entre as imagens I_1 e I_2 (já suavizadas). Por último, estima-se o fluxo óptico, seja trabalhando ponto-a-ponto ou em blocos (dependendo do algoritmo em questão).

6.3.3 Medidas de confiabilidade

Uma vez calculado o fluxo óptico, utilizamos o conceito de valores aberrantes para minimizar erros e distorções e tornar os resultados mais confiáveis. Começamos identificando os valores aberrantes através da construção do gráfico de caixa, a partir dos vetores de fluxo estimados através do algoritmo ponto-a-ponto. A seqüência de cálculo é a seguinte:

- Estimativa do fluxo óptico através do algoritmo ponto-a-ponto;
- Cálculo do primeiro, segundo e terceiro quartis dos fluxos estimados;
- Cálculo da distância interquartil (IQR);
- Cálculo dos bigodes (1,5 vez IQR).

Qualquer dado maior do que 1,5 vez IQR a partir do 3^o quartil deve ser marcado como um valor aberrante. O mesmo princípio se aplica para dados menores do que 1,5 vez IQR abaixo do 1^o quartil.

Em seguida, para cada ponto que apresentou um vetor de fluxo considerado um valor aberrante, substituímos os valores de seus gradientes espaciais (E_x e E_y) pelas medianas dos gradientes calculadas para cada bloco. Desta forma, ao recalcularmos o fluxo óptico, já estamos livres das influências dos valores aberrantes sobre os resultados.

6.4 Funções de Implementação

As principais *functions* desenvolvidas em MATLAB responsáveis por executar as etapas descritas anteriormente estão listadas abaixo, com um breve resumo do que cada uma faz (o código integral em MATLAB pode ser encontrado no Apêndice 2):

GAUSS: Dado um desvio padrão (sigma), gera uma Gaussiana unidimensional.

DGAUSS: Dado um desvio padrão (sigma), gera a derivada de uma Gaussiana unidimensional.

GRADIENT: Dado um par de imagens, calcula os gradientes espaciais e o gradiente temporal.

FLOW: Dados os gradientes espaciais e o gradiente temporal, estima o fluxo óptico através do algoritmo ponto-a-ponto.

FLOW_AVERAGE: Dados os gradientes espaciais e o gradiente temporal, estima o fluxo óptico através do algoritmo da média.

FLOW_LSQUARE: Dados os gradientes espaciais e o gradiente temporal estima o fluxo óptico através do algoritmo do mínimo erro quadrático.

FLOW_WEIGHT: Dados os gradientes espaciais, o gradiente temporal e a função de ponderação W, estima o fluxo óptico através do algoritmo do mínimo erro quadrático ponderado.

FLOW_OUTLIERS:Dados os gradientes espaciais e o gradiente temporal, estima o fluxo óptico através do algoritmo do mínimo erro quadrático após a identificação e transformação de valores aberrantes.

OUTLIERS: Dados os vetores de fluxo calculados através do algoritmo ponto-a-ponto, identifica e transforma valores aberrantes, calculando novos valores para os gradientes espaciais.

6.5 Funções auxiliares

Algumas funções específicas que fazem parte da caixa de ferramentas de Processamento de imagens e da caixa de ferramentas Estatísticas do MATLAB, foram utilizadas na implementação do método LK com a seguinte finalidade (mais detalhes encontram-se no Apêndice 3):

QUIVER: dadas as componentes (u,v) das velocidades estimadas, representa-as como vetores (setas) nos pontos (x,y).

BLKPROC: dados uma imagem, uma função e o tamanho do bloco, executa a função dada, em blocos, ao longo da imagem.

HIST: dados os fluxos estimados para um bloco da imagem, desenha o histograma das magnitudes dos fluxos.

BOXPLOT: dados os fluxos estimados para um bloco da imagem, desenha o correspondente gráfico de caixas.

SET "YDir": altera o sentido do eixo Y .

Capítulo 7 – Testes e resultados

O objetivo principal dos testes realizados foi comprovar o funcionamento dos algoritmos implementados e comparar os mapas de fluxo obtidos através destes algoritmos conhecidos com os mapas de fluxo obtidos pelo algoritmo modificado (identificação e transformação de valores aberrantes).

Em nenhum momento os testes realizados tiveram a intenção de comparar desempenho entre os algoritmos propostos por LK, uma vez que isto já foi objeto de estudo. Podemos dizer que testamos a implementação realizada, e não os algoritmos em si.

7.1 Testes executados

7.1.1 Scripts de execução dos testes

Para definir e executar as seqüências de testes foram desenvolvidos *scripts* contendo os parâmetros de teste, a seqüência de execução das funções criadas e o formato dos gráficos responsáveis por apresentar os resultados obtidos.

Principais *scripts*:

testa1e2: testa os algoritmos *ponto-a-ponto* e *média*

testa3e4: testa os algoritmos *mínimo erro quadrático* e *mínimo erro quadrático ponderado*

testa4e5: compara os resultados dos *algoritmos mínimo erro quadrático ponderado* e *mínimo erro quadrático após identificação e transformação dos valores aberrantes*

Estes *scripts* foram executados inúmeras vezes, com alterações nos valores das variáveis e nas seqüências de imagens escolhidas. Nem todas as execuções destes scripts estão sendo consideradas. Os resultados apresentados neste Capítulo foram os que melhor ilustraram as questões discutidas no presente trabalho.

7.1.2 Imagens utilizadas

As imagens escolhidas para testar os algoritmos implementados foram as mesmas utilizadas por Barron, Fleet *et al* [1,2] e disponibilizadas na internet[10] exatamente com o objetivo de permitir uma comparação de desempenho entre diferentes métodos e implementações.

Ao todo, Barron, Fleet *et al.* utilizaram 4 imagens sintéticas e 4 imagens reais. Enquanto que trabalhar com imagens sintéticas, livres de oclusões, transparência, etc, nos permite obter um desempenho máximo (situação otimista), trabalhando com imagens reais temos o desempenho do método, senão para o "pior caso", para o caso onde diversas fontes de erro estão presentes.

Como o objetivo deste trabalho não é testar o desempenho do método em comparação com outros métodos, mas validar a implementação do mesmo e comprovar seu aprimoramento, escolhemos apenas duas imagens, uma sintética e uma real, que são respectivamente *"Translating tree" e "Hamburg Taxi"*.

Esta escolha se deve não só ao fato de uma seqüência ser sintética e a outra real mas principalmente pela natureza distinta dos movimentos representados em cada uma delas. Enquanto que na seqüência *"Translating tree"* é a câmera que se movimenta, na outra seqüência a câmera fica estática e são os objetos contidos nela que se movem, em direções distintas e com velocidades diversas.

Na seqüência "Translating tree" a câmera se move na direção normal ao seu plano de visão ao longo do eixo x, com velocidades entre 1,73 e 2,26 pixels/quadro, sempre paralelas ao eixo x. A figura 7-1 mostra um quadro desta seqüência e seu respectivo mapa de fluxo óptico (teórico).

[10] As seqüências de imagens foram disponibilizadas por John Barron e podem ser obtidas na internet no endereço ftp://ftp.csd.uwo.ca/pub/vision até o presente momento.

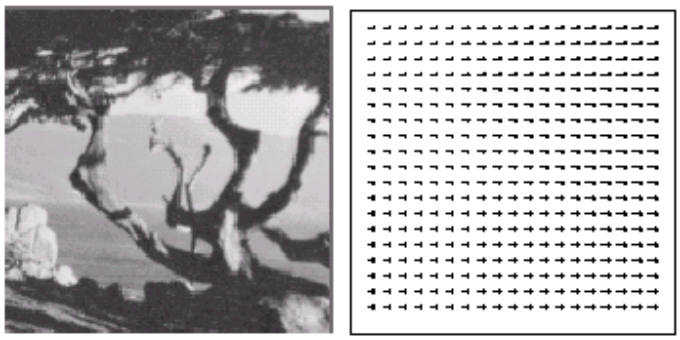


Figura 7-1 : *Um dos quadros pertencente à seqüência "Translating Tree" e seu respectivo mapa de fluxo óptico (teórico).*

A seqüência "Hamburg Taxi" mostra uma cena de rua com quatro objetos se movendo: um táxi virando a esquina a uma velocidade de 1 pixel/quadro, um carro no canto inferior esquerdo, se movendo da esquerda para a direita a uma velocidade de 3 pixels/quadro, um utilitário no canto inferior direito, se movendo da direita para a esquerda também a uma velocidade de 3 pixels/quadro e um pedestre no canto superior esquerdo andando a uma velocidade de 0,3 pixels/quadro. Nesta seqüência a câmera que captou as imagens se encontra parada. Um quadro desta seqüência e seu respectivo mapa de fluxo se encontram apresentados na figura abaixo (figura 7-2).

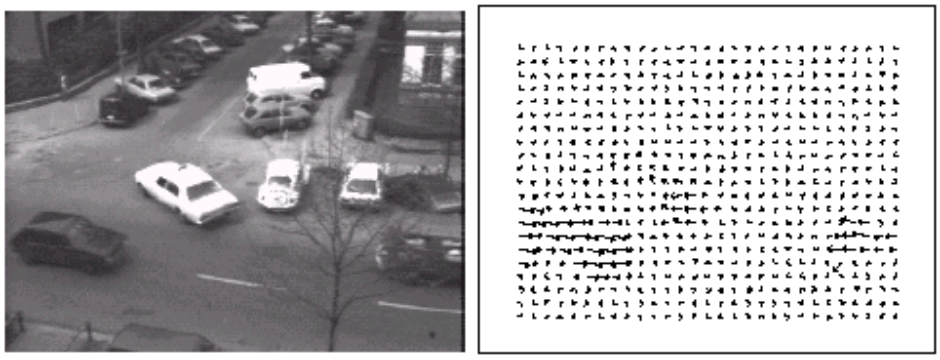


Figura 7-2: *Um dos quadros da seqüência "Hamburg Taxi" e seu respectivo mapa de fluxo óptico (estimado pela implementação de Barron do método Anandan)* [11].

[11] Os exemplos de mapas de fluxo para estas imagens também foram obtidas na internet no endereço ftp://ftp.csd.uwo.ca/pub/vision.

Como os quadros que compõem a seqüência “Hamburg Taxi“ (190X256) são bem maiores do que os da seqüência “Translating tree” (150X150), cortamos os da primeira seqüência para que ambas tivessem o mesmo tamanho, nos preocupando em manter o táxi em movimento dentro da imagem final a ser utilizada, pois decidimos concentrar nele as observações de fluxo. As vantagens de termos ambas as imagens do mesmo tamanho é não termos que alterar os scripts de teste e também obtermos mapas de fluxo com o mesmo número de vetores, o que facilita qualquer comparação que quisermos fazer.

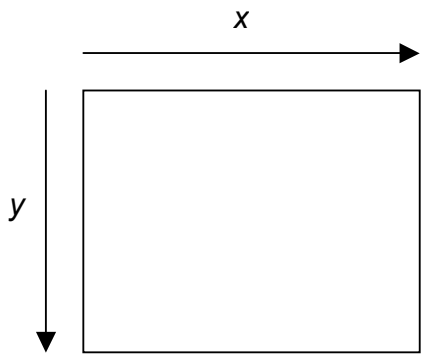
7.1.3 Formato dos arquivos

Inicialmente tentamos utilizar as imagens em formato JPEG, mas os resultados obtidos ficaram muito aquém, o que nos fez concluir que as perdas por compressão nos levaram a erros inadmissíveis durante os cálculos do fluxo óptico. Nossa escolha então foi a de utilizar as imagens em formato binário, o que nos possibilitou obter resultados muito mais próximos dos esperados.

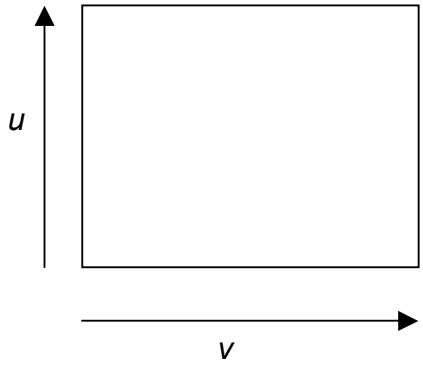
7.1.4 Convenções e parâmetros utilizados

O tamanho de todas as imagens utilizadas foi de 150 X 150 pixels. Para executar os algoritmos que operam em blocos, escolhemos blocos de tamanho 8X8 pixels. Este tamanho foi escolhido por se tratar de um tamanho comumente utilizado em Processamento de Imagens. Além disso, se dividirmos a imagem de tamanho 150 X 150 em blocos de tamanho 8 X 8, obtemos imagens contendo 19 X 19 vetores (cada vetor representa um bloco), o que nos proporciona uma boa visualização dos mapas de fluxo. Vale a pena ressaltar que a divisão de 150 por 8 não é exata, o que faz com que os últimos blocos sejam menores que os demais. Neste caso, a função **blkproc** completa a matriz com zeros até um múltiplo de 8 (veja Apêndice 3).

Para os eixos das imagens, adotamos a convenção abaixo, por ser a normalmente utilizada pelo MATLAB:



Já os mapas de fluxo apresentados foram gerados utilizando-se a função **quiver** (Veja Apêndice 3), onde *u* e *v* representam as componentes da velocidade nas seguintes direções:



Esta diferença entre o sentido do eixo y (na imagem) e o sentido de u (definido pela função quiver) faz com que seja necessário inverter o sentido do eixo y antes de desenhar os gráficos de velocidade utilizando a opção **'reverse'** da função **set** (Apêndice 3). Desta forma, *y* e *u* ficam com a mesma orientação.

Tanto os valores de desvio padrão da máscara Gaussiana utilizada na etapa de suavização, quanto a máscara de ponderação utilizada no algoritmo de mínimo erro quadrático ponderado foram retirados do trabalho de Barron, Fleet et al. [1,2].

7.2 Testa1e2

O primeiro grupo de testes foi realizado através do script *testa1e2* e se refere aos algoritmos ponto-a-ponto e média. O roteiro das principais tarefas executadas pelo script, com as variáveis de entrada está listado abaixo:

Tarefa	Variáveis de entrada
Lê os arquivos de imagens	Nomes dos arquivos
Limita o cálculo a um bloco da imagem	Blocox e Blocoy
Mostra a imagem e o bloco selecionado	
Chama a função GRADIENT	Desvio padrão
Executa a função FLOW	
Desenha o mapa de fluxo resultante para o bloco selecionado	Blocox e Blocoy
Executa a função FLOW_AVERAGE	
Desenha o vetor resultante para o bloco selecionado sobre o mapa já existente	Blocox e Blocoy

Tabela 7-1 : *Roteiro de tarefas executadas pelo script testa1e2.*

A primeira tarefa do script testa1e2 é, após ler as imagens, estimar o fluxo óptico das seqüências escolhidas através do algoritmo ponto-a-ponto. Como este algoritmo produz um vetor de fluxo óptico para cada pixel da imagem, teríamos como resultado um mapa de fluxo com 22500 vetores (no caso de uma imagem de tamanho 150X150, por exemplo), que seria de difícil visualização tanto na tela quanto na folha impressa. Então, ao invés de executarmos o algoritmo para a imagem inteira, executamos o

mesmo apenas para um bloco da imagem de tamanho 8X8. Desta forma, obtemos um mapa de fluxo de apenas 64 vetores, muito mais inteligível.

Para visualizar melhor como o algoritmo da média pode melhorar o resultado obtido através do algoritmo ponto-a-ponto, calculamos ainda o fluxo pelo algoritmo da média para o mesmo bloco e desenhamos o vetor de fluxo resultante sobre o mesmo mapa de fluxo já obtido anteriormente (lembrando que o algoritmo da média obtém um único vetor para cada bloco 8X8).

7.2.1 Resultados obtidos

A primeira execução do script testa1e2 utilizou a seqüência “Translating tree” e, o bloco escolhido foi o (8,15). As figuras resultantes foram as seguintes:

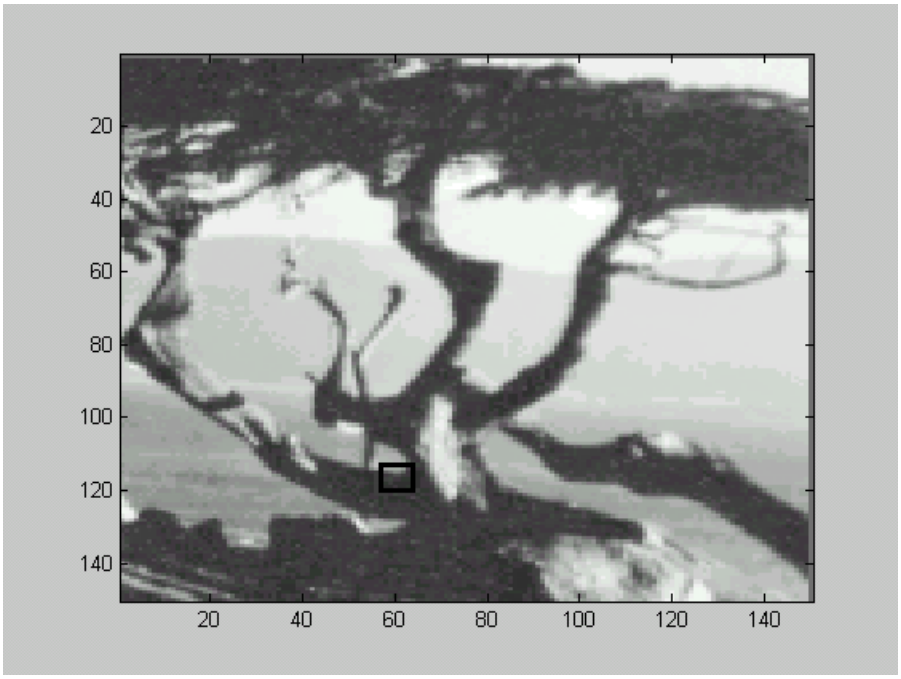


Figura 7-3 : *Um dos quadros da seqüência “Translating tree” com o bloco escolhido assinalado por um retângulo (blocox=8 e blocoy=15).*

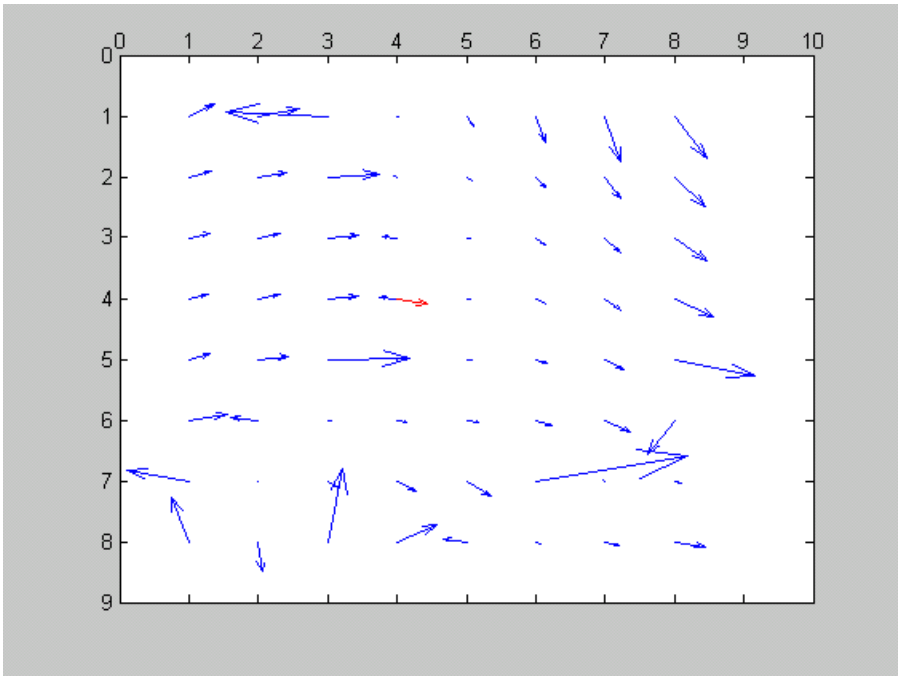


Figura 7-4 : *Mapa de fluxo resultante utilizando-se o algoritmo ponto-a-ponto (em azul) e utilizando-se a média (em vermelho).*

A figura 7-3 mostra uma das imagens da seqüência “Translating tree” utilizada no teste. Nela o bloco de tamanho 8X8 para o qual o fluxo óptico foi estimado está identificado através de um retângulo preto.

A figura 7-4 contém o mapa de fluxo obtido para os dois algoritmos: em *azul* vemos os vetores calculados *ponto-a-ponto* e em *vermelho* o vetor de fluxo óptico calculado pela *média*. Podemos observar que os fluxos individuais apontam, em sua maioria, para a direção do movimento e apresentam intensidades semelhantes (resultado próximo ao da figura 7.1). Porém, há ainda um número considerável de pontos cujos fluxos apontam em uma direção diferente e/ou apresentam magnitudes muito distintas.

Como vimos no capítulo 6, a equação (na formulação matricial) para estimativa do fluxo óptico através do algoritmo ponto-a-ponto contém o gradiente espacial em seu denominador (eq. 6-3). Isto explica porque, sempre que o gradiente espacial for muito pequeno ou igual a zero (estamos falando de regiões homogêneas), o resultado da

divisão será um vetor de intensidade exagerada, em comparação com os demais vetores.

Observando o vetor em vermelho, que representa a média dos fluxos calculados individualmente, vemos que sua direção é bem semelhante à do fluxo óptico real. Isto porque, como qualquer cálculo baseado na média, ele mostra a tendência da maioria dos vetores. Ou seja, neste caso o algoritmo da média foi capaz de apresentar um resultado muito bom, atenuando o efeito dos pontos cujos vetores de fluxo apresentaram direções e magnitudes completamente dissociadas do movimento real.

Em seguida, executamos novamente o script *testa1e2* utilizando a mesma seqüência "Translating tree", só que agora para um novo bloco (9,7). As figuras resultantes estão apresentadas a seguir:

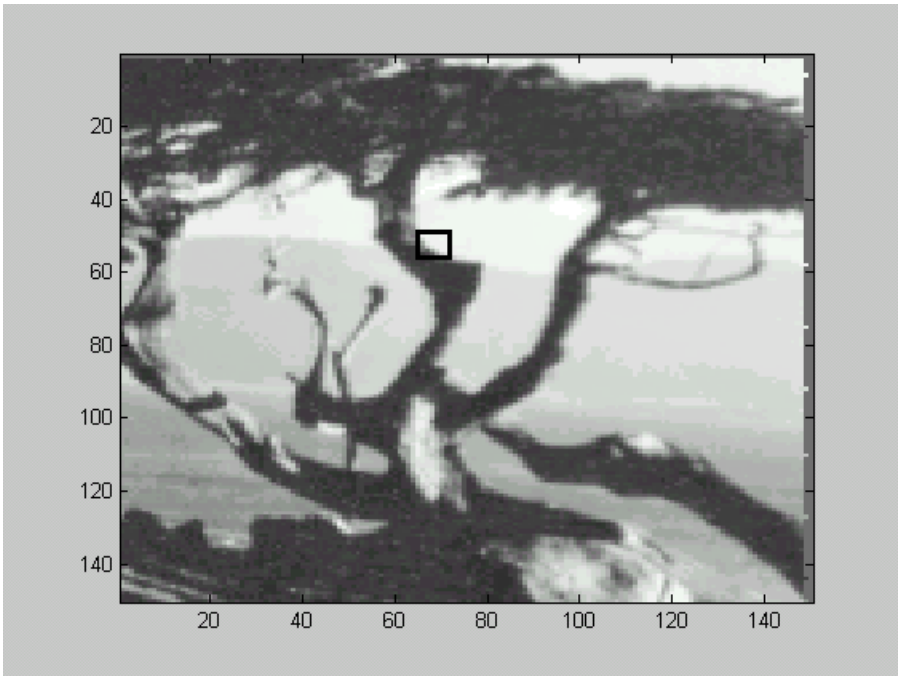


Figura 7-5 : *Um dos quadros da seqüência "Translating tree" com o novo bloco escolhido assinalado por um retângulo (blocox=9 e blocoy=7).*

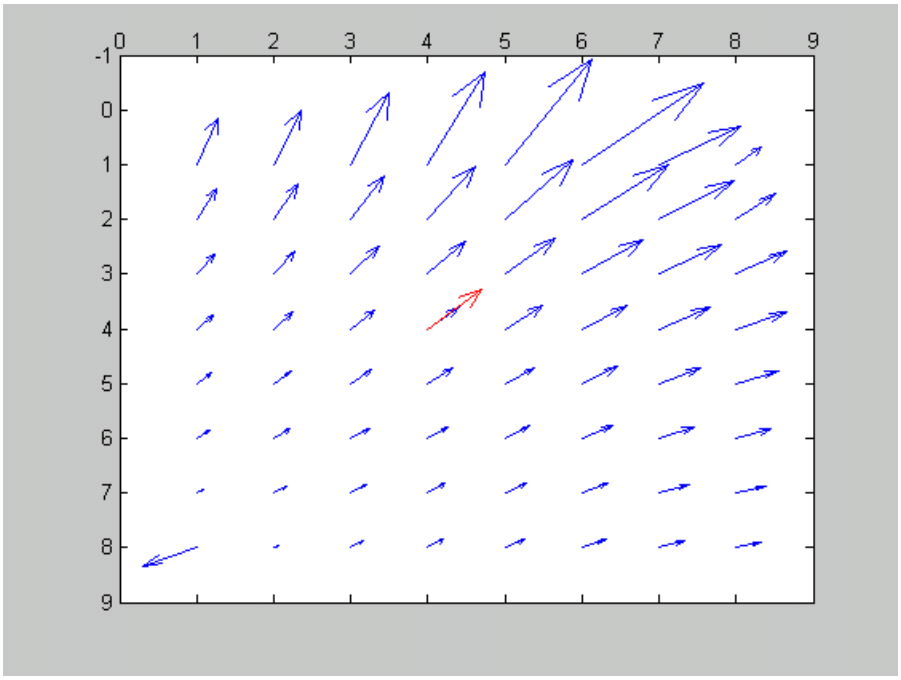


Figura 7-6 : *Mapa de fluxo resultante utilizando-se o algoritmo ponto-a-ponto (em azul) e utilizando-se a média (em vermelho).*

A figura 7-5 mostra o mesmo quadro da seqüência “Translating tree” com o novo bloco escolhido e na figura 7-6 vemos o mapa de fluxo obtido para os dois algoritmos: os vetores em azul são os fluxos calculados *ponto-a-ponto* e o vetor em vermelho representa o fluxo óptico calculado pela *média*.

Este bloco ilustra bem alguns dos pontos fracos do algoritmo ponto-a-ponto e do algoritmo da média. Em primeiro lugar, apenas a componente do fluxo óptico na direção do gradiente local da imagem pode ser determinada (Capítulo 2 - *problema da abertura)*, por isso todos os fluxos apontam na direção perpendicular à borda do galho. Além disso, como os fluxos individuais apontam todos na mesma direção, a qual não corresponde ao movimento real da imagem, a média não apresenta um resultado melhor, pois só faz confirmar a direção dos fluxos calculados individualmente. Ou seja, este é um típico exemplo onde o algoritmo da *média* não é capaz de obter melhores resultados do que o algoritmo *ponto-a-ponto*.

O próximo passo foi executar o mesmo script testa1e2, desta vez utilizando a seqüência “Hamburg Taxi” e escolhendo o bloco (8,14). Os resultados obtidos estão apresentados a seguir:

Figura 7-7 : *Um dos quadros da seqüência “Hamburg Taxi” com o bloco escolhido assinalado por um retângulo (blocox=8 e blocoy=14).*

A figura 7-7 mostra uma das imagens da seqüência “Hamburg Taxi” utilizada no teste. Nela o bloco de tamanho 8X8 para o qual o fluxo óptico foi estimado está identificado através de um retângulo preto. Note que escolhemos um bloco pertencente ao táxi, justamente por ser um dos objetos que se move nesta seqüência.

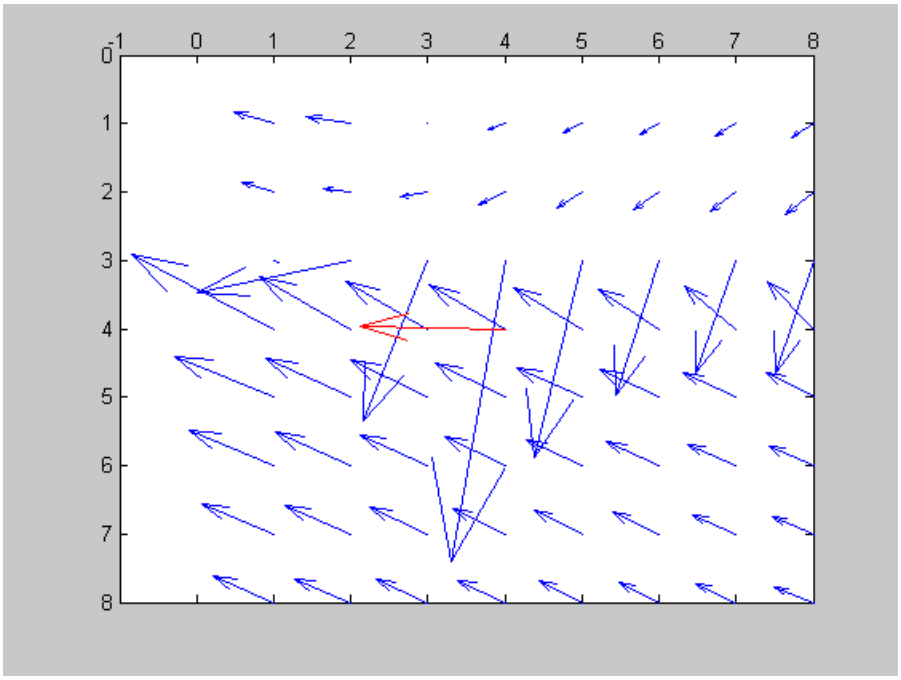


Figura 7-8 : *Mapa de fluxo resultante utilizando-se o algoritmo ponto-a-ponto (em azul) e utilizando-se a média (em vermelho).*

A figura 7-8 mostra o mapa de fluxo obtido para os dois algoritmos: *ponto-a-ponto* e *média*. Podemos observar que os fluxos individuais apontam, em sua maioria, para a direção do movimento e apresentam intensidades semelhantes (resultado próximo ao da figura 7-2, na região do bloco escolhido). Há apenas alguns pontos localizados na borda superior do bloco cujos fluxos apontam em uma direção diferente e alguns destes vetores possuem magnitudes muito superiores aos fluxos dos demais pontos. Isto se deve ao fato da maioria dos pixels se encontrar em uma região de transição entre a lataria branca do táxi e a sombra da parte traseira. Os pixels nesta região assumem uma variação gradativa de intensidade luminosa, que garante gradientes espaciais diferentes de zero e possibilitam uma estimativa razoável do fluxo óptico. Já os pixels da borda superior do bloco se encontram em uma região homogênea, onde provavelmente o gradiente espacial é igual a zero. Desta forma, os fluxos calculados para estes pixels fogem dos valores esperados.

Podemos verificar também que neste caso o algoritmo da média só faz piorar o resultado do fluxo estimado para este bloco, pois enquanto a maioria dos fluxos individuais apresenta valores e direções coerentes, apenas incorretos em uma das bordas do bloco, a média calculada aponta para uma direção diferente da do movimento, pois é bastante influenciada pelos fluxos incorretos de magnitudes muito superiores às da maioria.

Uma nova execução do script testa1e2 para a seqüência “Hamburg Taxi”, só que agora para o bloco (6,14) apresentou os seguintes resultados:

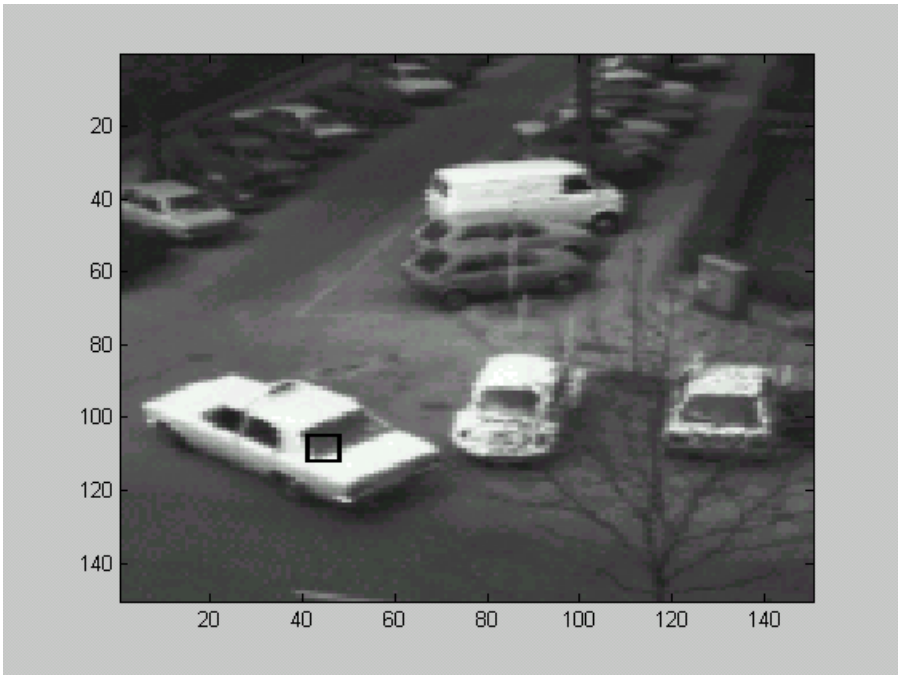


Figura 7-9 : *Um dos quadros da seqüência “Hamburg Taxi” com o novo bloco escolhido assinalado por um retângulo (blocox=6 e blocoy=14).*

A figura 7-9 mostra o mesmo quadro da seqüência “Hamburg Taxi” com o novo bloco escolhido e na figura seguinte vemos os mapas de fluxo obtidos para os algoritmos *ponto-a-ponto* e *média*.

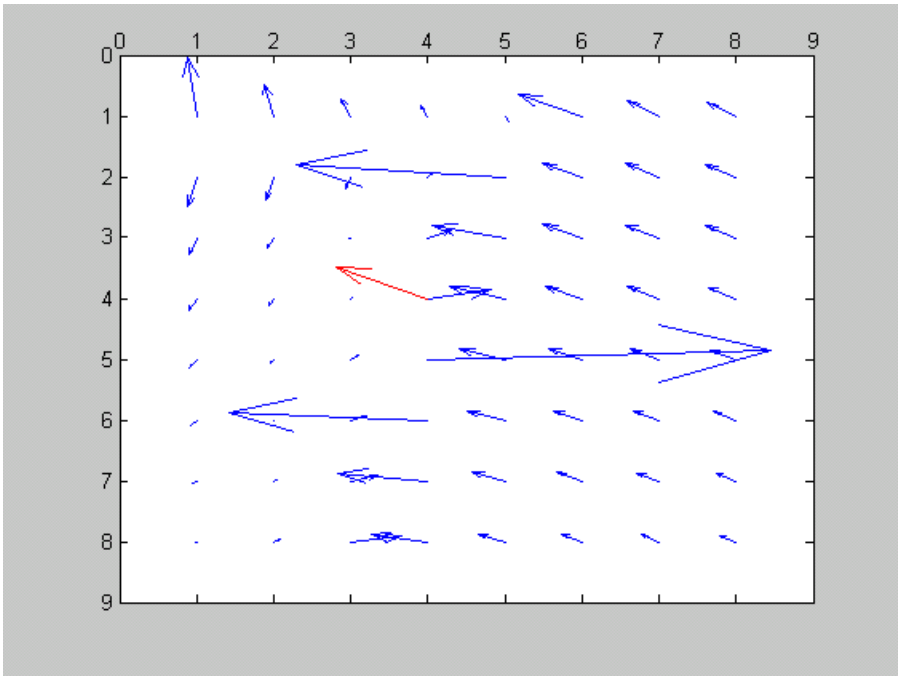


Figura 7-10 : *Mapa de fluxo resultante utilizando-se o algoritmo ponto-a-ponto (em azul) e utilizando-se a média (em vermelho).*

Observando a figura 7-10 vemos novamente o problema da abertura interferindo no mapa de fluxo obtido através do algoritmo *ponto-a-ponto*, assim como no exemplo da figura 7-6. Como apenas a componente do fluxo óptico na direção do gradiente local da imagem pode ser determinada, os fluxos estimados, neste caso, apontam na direção perpendicular às bordas da janela do táxi. Por isso vemos na metade esquerda do bloco, os fluxos apontando para baixo e na metade direita do bloco os fluxos apontando para a esquerda. Mais uma vez, a limitação do algoritmo quanto ao problema da abertura não nos permite obter um bom resultado para os fluxos individuais. Já a *média*, mostrada em vermelho, acaba apontando na direção do movimento, o que neste caso, não passa de uma coincidência, pois ela é resultante da combinação dos vetores perpendiculares a uma e a outra borda da janela.

7.2.2 Comentários

De maneira geral, a realização dos testes nos permitiu comprovar que o algoritmo ponto-a-ponto tem algumas deficiências. A primeira delas é a dificuldade de estimar o fluxo óptico em regiões de gradientes espaciais pequenos (regiões de baixa textura). Nestes pontos o algoritmo obtém como resultado fluxos de direção e magnitude incorretos. Outra deficiência é a vulnerabilidade em relação ao problema da abertura. Devido a este problema, somente a componente do fluxo perpendicular ao gradiente local pode ser determinada, o que faz com que, em regiões com gradiente local em uma única direção (uma borda, por exemplo), todos os fluxos apontem na direção perpendicular a este gradiente, independente da direção do movimento.

Já o algoritmo da média mostra-se um pouco mais robusto, pois, na maioria das vezes, é capaz de desprezar os fluxos discrepantes estimados para as regiões com baixa textura, já que mostra a tendência da maioria dos fluxos. Ele só se mostra ineficaz nos casos em que a região homogênea é muito grande dentro do bloco. Com relação ao problema da abertura, o algoritmo da média também se mostrou vulnerável a ele, pois o mesmo só faz confirmar a direção da maioria dos fluxos calculados.

7.3 Testa3e4

O próximo grupo de testes foi realizado através do script *testa3e4,* com o objetivo de calcular o fluxo óptico através dos algoritmos *mínimo erro quadrático* e *mínimo erro quadrático ponderado*. O roteiro das principais tarefas executadas pelo script, com as variáveis de entrada, está listado abaixo:

Tarefa	Variáveis de entrada
Lê os arquivos de imagens	Nomes dos arquivos
Chama a função GRADIENT	Desvio padrão
Executa a função FLOW_LSQUARE	
Desenha o mapa de fluxo resultante	
Executa a função FLOW_WEIGTH	Máscara W (para ponderação)
Desenha o mapa resultante sobre o mapa já existente	

Tabela 7-2 : *Roteiro de tarefas executadas pelo script testa3e4.*

A primeira tarefa do script testa3e4 é, após ler as imagens, estimar o fluxo óptico das seqüências escolhidas através do algoritmo de *mínimo erro quadrático*.

O mapa de fluxo resultante é apresentado em uma figura. Em seguida o script calcula o fluxo óptico, desta vez utilizando o algoritmo de *mínimo erro quadrático com ponderação*. Para podermos comparar os resultados e concluir se a ponderação de fato melhora a estimativa, desenhamos o mapa de fluxo resultante sobre o mapa de fluxo obtido anteriormente (utilizando cores diferentes para cada um dos mapas).

7.3.1 Resultados obtidos

A primeira execução do script testa3e4 utilizou a seqüência “Translating tree” e as figuras resultantes foram as seguintes:

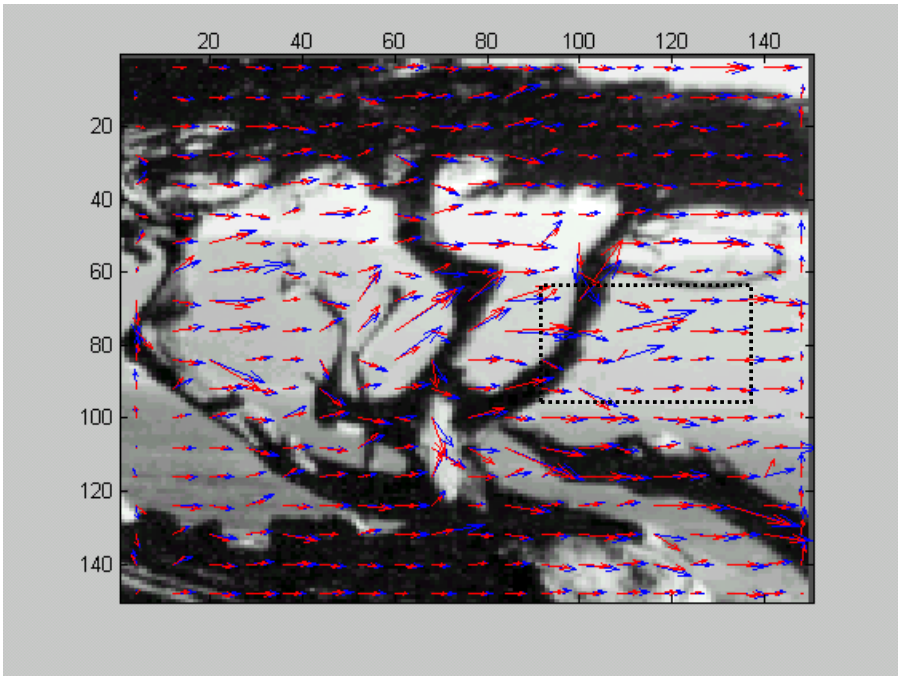


Figura 7-11 : *Um dos quadros da seqüência “Translating tree” e os mapas de fluxo resultantes utilizando-se o mínimo erro quadrático (em azul) e utilizando-se o mínimo erro quadrático ponderado (em vermelho). O retângulo tracejado indica a região ampliada e mostrada na figura seguinte.*

A figura 7-11 mostra uma das imagens da seqüência “Translating tree” utilizada no teste com os mapas de fluxo obtidos para os dois algoritmos: em azul vemos os vetores calculados através do *mínimo erro quadrático* e em vermelho os vetores estimados pelo *mínimo erro quadrático ponderado*. Observando os mapas de fluxo resultantes, podemos dizer, de maneira geral, que os dois algoritmos obtiveram resultados semelhantes. Enquanto nas bordas da imagem, todos os vetores de fluxo apontam na direção do movimento e apresentam intensidades similares, na região central da imagem, os vetores estimados apresentam distorções (direções e magnitudes variadas).

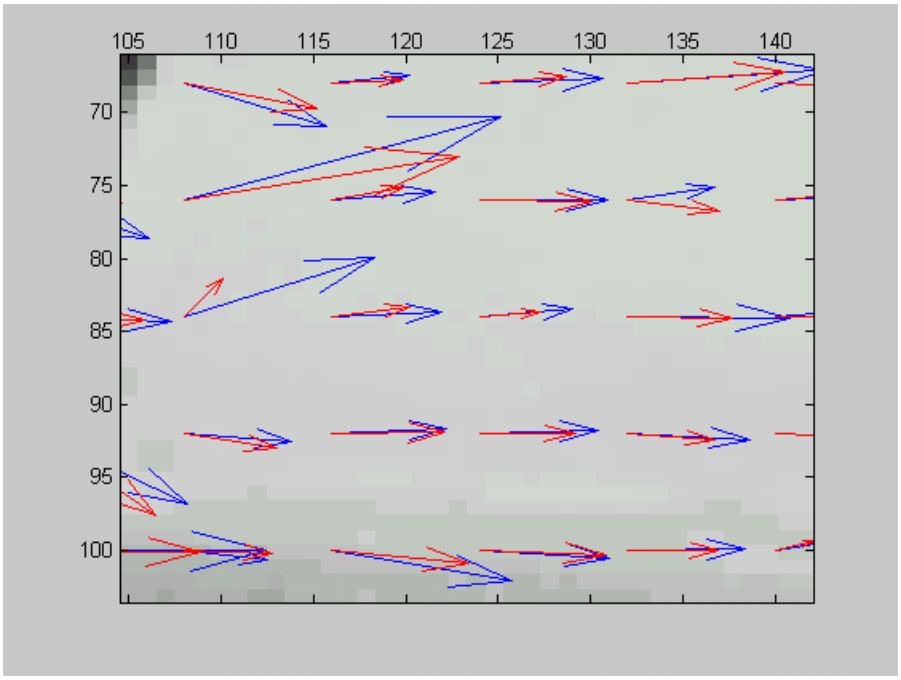


Figura 7-12 : *Parte do mapa de fluxo (ampliado).*

Para podermos analisar melhor os resultados obtidos, inserimos ainda uma nova figura (figura 7-12) contendo apenas uma parte dos mapas de fluxo já mostrados anteriormente. Isto nos permite aproximar a imagem e observar os vetores de fluxo ampliados. Nesta condição, vemos que os vetores em vermelho apontam quase sempre na mesma direção que os vetores em azul, às vezes com uma pequena diferença, porém, em todos os casos os vetores vermelhos apresentam uma magnitude menor do que os em azul. Isto nos mostra que o algoritmo de *mínimo erro quadrático ponderado* atenua as discrepâncias do algoritmo sem ponderação, fazendo com que o fluxo óptico estimado se aproxime mais do fluxo óptico real. Porém, quando as distorções são muito grandes, mesmo com a atenuação realizada pela ponderação, o resultado fica longe do esperado.

Em seguida, executamos o mesmo script testa3e4, desta vez utilizando a seqüência "Hamburg Taxi" e os resultados obtidos estão apresentados a seguir:

Figura 7-13 : *Um dos quadros da seqüência "Hamburg Taxi" e os mapas de fluxo resultantes utilizando-se o mínimo erro quadrático (em azul) e utilizando-se o mínimo erro quadrático ponderado (em vermelho). O retângulo tracejado indica a região ampliada e mostrada na figura seguinte.*

Neste teste executado com imagens da seqüência "Hamburg Taxi" podemos observar que os mapas de fluxo obtidos são semelhantes entre si, e conseguem mostrar o movimento do táxi fazendo a curva (figura 7-13). Porém, estes mapas apresentam valores discrepantes para vários pontos cujos fluxos deveriam ser nulos (regiões onde não há movimento). Isto se deve ao fato de estarmos trabalhando com uma imagem real, onde variações de luminosidade contradizem os pressupostos do método LK (veja Cap.2).

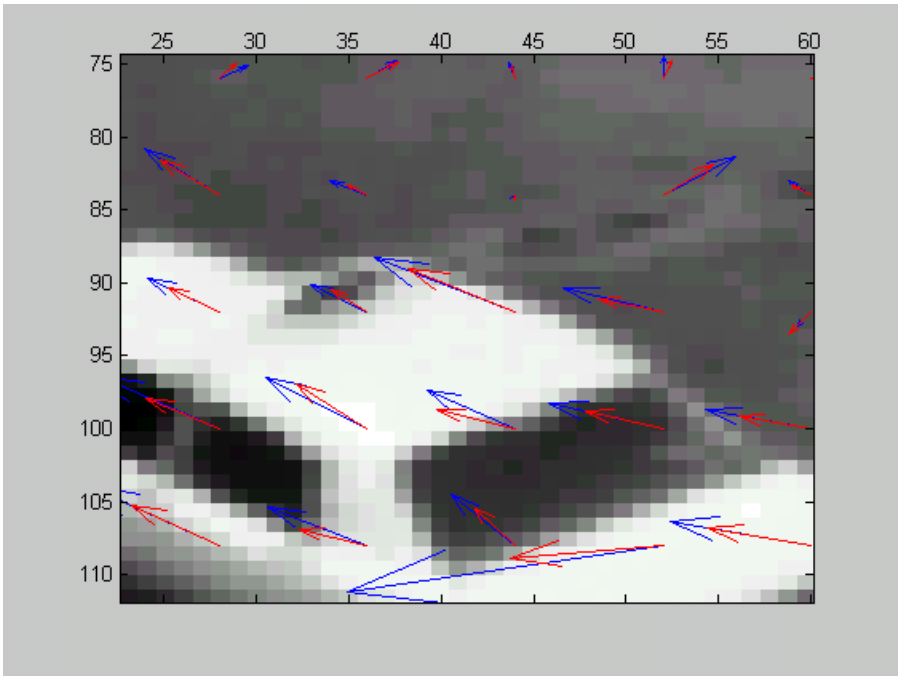


Figura 7-14 : *Parte do mapa de fluxo (ampliado).*

De qualquer forma, olhando mais de perto os resultados obtidos (figura 7-14) vemos que também neste caso o algoritmo com ponderação apresenta uma melhoria de desempenho em comparação ao algoritmo sem ponderação. Os vetores de fluxo estimados através dos dois algoritmos quase sempre coincidem em direção, mas, além de um pouco mais alinhados com os movimentos, os vetores calculados pelo algoritmo com ponderação (em vermelho) não apresentam magnitudes discrepantes, como alguns dos vetores estimados pelo algoritmo sem ponderação (em azul).

7.3.2 Comentários

Nesta seqüência de testes realizada, as principais conclusões a que podemos chegar é que a função de ponderação atenua as discrepâncias encontradas nos mapas de fluxo estimados através do mínimo erro quadrático. Os resultados obtidos para ambos os algoritmos se aproximam bem do esperado, porém há ainda algumas regiões da imagem para as quais os fluxos obtidos não condizem com o movimento. Isto se deve provavelmente à falta de informação de gradiente espacial, em blocos de intensidade constante, por exemplo.

Além disso, os resultados para a seqüência de imagens reais são bastante ruidosos, e, em regiões onde o fluxo deveria ser nulo, podemos encontrar valores de fluxo relativamente altos. As dificuldades encontrada neste caso são as variações de luminosidade ao longo da seqüência de imagem, além de efeitos de sombra e oclusões.

7.4 Testa4e5

O último conjunto de testes foi realizado através do script *testa4e5* nos algoritmos de mínimo erro quadrático ponderado e mínimo erro quadrático após identificação e transformação de valores aberrantes. Segue abaixo o roteiro das principais tarefas executadas pelo script com as variáveis de entrada:

Tarefa	Variáveis de entrada
Lê os arquivos de imagens	Nomes dos arquivos
Escolhe um bloco da imagem como exemplo	Blocox e Blocoy
Mostra a imagem e o bloco selecionado	
Chama a função GRADIENT	Desvio padrão
Executa a função FLOW_WEIGHT e desenha o mapa de fluxo resultante	Máscara W (para ponderação)
Executa a função FLOW_OUTLIERS	
Executa a função FLOW para o bloco selecionado e mostra o mapa resultante	Gradientes espaciais e temporais, Blocox e Blocoy
Desenha o gráfico de caixa para os vetores de fluxo estimados	
Desenha o histograma para estes vetores	
Executa a função FLOW para o bloco selecionado após transformação dos valores aberrantes e redesenha o mapa sobre o anterior	Novos gradientes espaciais e temporais, Blocox e Blocoy
Desenha o gráfico de caixa e o histograma para os novos vetores de fluxo estimados	
Desenha os fluxos resultantes para o bloco escolhido, antes e após transformação de valores aberrantes	

Tabela 7-3: *Roteiro de tarefas executadas pelo script testa4e5*

A primeira tarefa do script testa4e5 é, após ler as imagens, estimar o fluxo óptico das seqüências escolhidas através do algoritmo de *mínimo erro quadrático ponderado*. O mapa de fluxo resultante é desenhado sobre a imagem utilizada. Em seguida o script calcula o fluxo óptico, desta vez utilizando o algoritmo que identifica e transforma valores aberrantes. Isto é feito estimando-se os fluxos através do algoritmo *ponto-a-ponto* para cada bloco da imagem. Com o mapa resultante para um bloco, constrói-se o gráfico de caixa e identificam-se os valores aberrantes. Para recalcular estes fluxos, os gradientes espaciais relativos a estes pontos são substituídos pela mediana dos gradientes espaciais do bloco. Depois de corrigirmos todos os gradientes espaciais, estimamos o fluxo óptico para a imagem inteira utilizando o algoritmo de *mínimo erro quadrático*.

7.4.1 Resultados obtidos

Para a execução do script testa4e5 utilizamos inicialmente a seqüência “Translating tree” e escolhemos o bloco (8,15) para mostrar as etapas intermediárias do algoritmo. Esta escolha nos permite fazer algumas comparações ao longo do exemplo, uma vez que este bloco já foi usado no primeiro teste do script testa1e2. Os resultados obtidos estão apresentados a seguir.

A primeira figura (figura 7-15) mostra uma das imagens da seqüência “Translating tree” utilizada no teste com os mapas de fluxo obtidos para os dois algoritmos: em azul vemos os vetores calculados através do *mínimo erro quadrático ponderado* e em vermelho os vetores estimados pelo *mínimo erro quadrático após eliminação de valores aberrantes*. Observando os mapas de fluxo resultantes notamos que os resultados são próximos, porém os vetores em vermelho estão mais alinhados (na direção do movimento) e poucos apresentam magnitudes muito distintas dos demais. Isto fica mais claro na figura seguinte (figura 7-16), onde observamos apenas o mapa obtido após eliminação dos valores aberrantes.

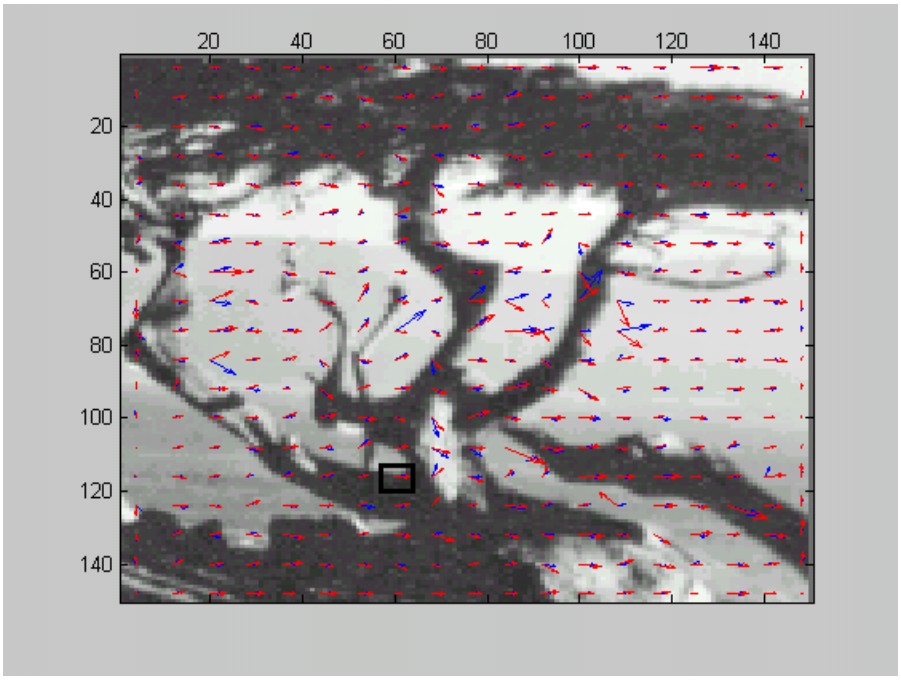


Figura 7-15 : *Um dos quadros da seqüência "Translating tree" e os mapas de fluxo resultantes utilizando-se o mínimo erro quadrático ponderado (em azul) e utilizando-se o mínimo erro quadrático após identificação e transformação dos valores aberrantes (em vermelho). O bloco identificado por um retângulo (blocox=8 e blocoy=15) foi o utilizado para ilustrar todas as etapas do algoritmo.*

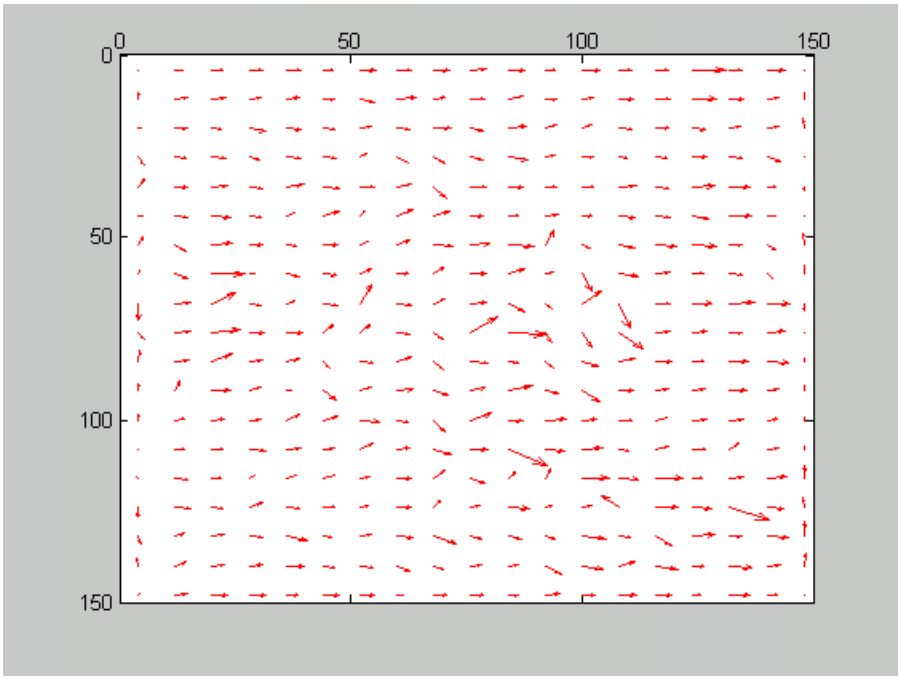


Figura 7-16 : *O mesmo mapa de fluxo obtido através do algoritmo de mínimo erro quadrático com identificação e transformação de valores aberrantes (em vermelho), só que desta vez mostrado sozinho.*

Para entender melhor o método proposto de identificação e transformação de valores aberrantes selecionamos um bloco da imagem (identificado pelo retângulo preto na figura 7-15) e geramos algumas figuras para este bloco, dadas a seguir:

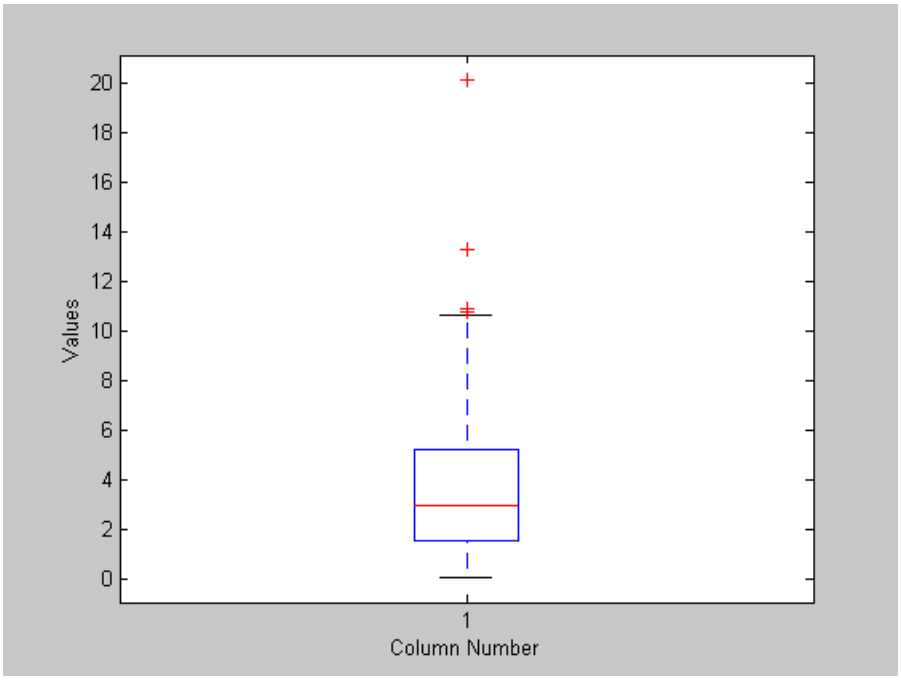


Figura 7-17 : *O gráfico de caixa ("boxplot") mostra a caixa (1º e 3º quartis em azul), a mediana (linha vermelha dentro da caixa), os limites para os valores aberrantes (linhas horizontais em preto) e os valores aberrantes identificados pelas cruzes em vermelho.*

A primeira coisa a fazer quando se quer tratar valores aberrantes é identificá-los (conforme visto no Capítulo 4). No nosso algoritmo usamos gráficos de caixa para realizar tal tarefa. A figura 7-17 mostra o gráfico de caixa para o bloco selecionado (3,11). Nela podemos ver que a mediana dos fluxos individuais do bloco se encontra em torno de 1, o 1º quartil próximo a 1 e o 3º.quartil próximo a 5. Isto nos dá uma distância intrequartil (IQR) de 4.

Conforme já visto no Capítulo 4, os valores cuja distância do 1º quartil ou do 3º quartil for superior a 1,5*IQR, ou seja, 1º quartil-1,5*IQR ou 3º quartil+1,5*IQR, serão considerados valores aberrantes. Neste caso, os valores abaixo de 0 (já que não há valores negativos, o bigode inferior fica limitado em zero) e os acima de 11 foram considerados valores aberrantes e estão identificados no gráfico por cruzes em vermelho.

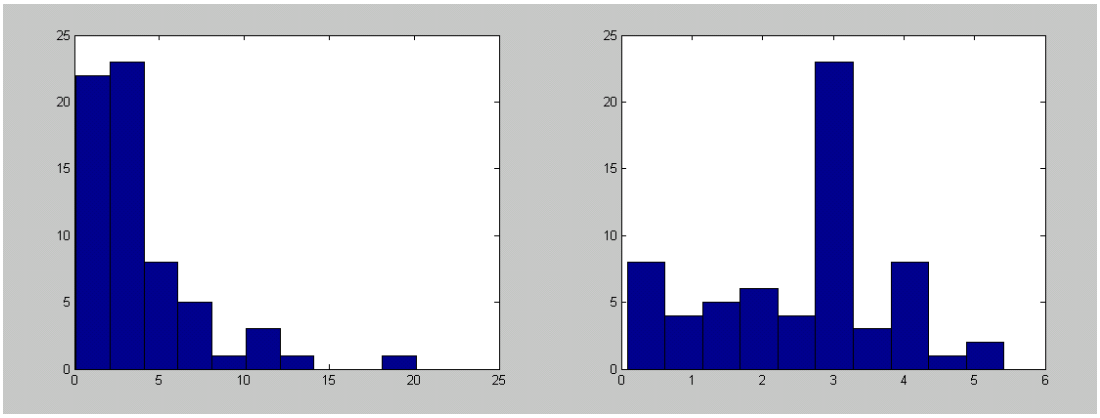


Figura 7-18 : *Histogramas das magnitudes dos vetores de fluxo estimados para o bloco (8,15) antes e após a identificação e transformação dos valores aberrantes.*

O primeiro histograma da figura 7-18 (antes da identificação dos valores aberrantes) confirma este raciocínio. Ele nos mostra a distribuição das magnitudes dos vetores de fluxo estimados para o bloco selecionado. A escala deste histograma divide os vetores da seguinte forma: na primeira coluna estão representados os vetores cujas magnitudes se encontram entre 0 e 2. A segunda coluna apresenta os vetores cujas magnitudes ficam entre 2 e 4, e assim sucessivamente até o último grupo, que apresenta magnitudes entre 18 e 20. Vemos claramente que a grande maioria dos vetores possui magnitude inferior a 8 (58 dos 64 vetores pertencem ao primeiro grupo) e que temos somente 6 vetores com magnitude superior a 8: um vetor entre 8 e 10, dois vetores entre 10 e 12, 1 vetor entre 12 e 14 e um vetor entre 18 e 20.

Ou seja, intuitivamente, sem calcular a mediana ou a distância interquartil, diríamos que pelo menos estes 6 vetores, com magnitude superior à 8 são valores aberrantes. É claro que, com a ajuda do gráfico de caixas, temos uma visão mais detalhada do que acontece e podemos identificar com mais segurança os valores aberrantes.

Uma vez identificados os valores aberrantes, passamos para a etapa de transformação dos dados, de forma a suavizar o impacto dos valores aberrantes. Mais especificamente, para cada ponto que apresentou um vetor de fluxo considerado um valor aberrante, substituímos os valores de seus gradientes espaciais (E_x e E_y) pelas medianas dos gradientes calculadas para cada bloco. A partir daí, recalculamos o fluxo óptico, desta vez utilizando o algoritmo de mínimo erro quadrático sem ponderação (entendemos que a eliminação de valores aberrantes está sendo apresentada como uma alternativa para a ponderação). O mapa de fluxo resultante foi apresentado nas figuras 7-15 e 7-16.

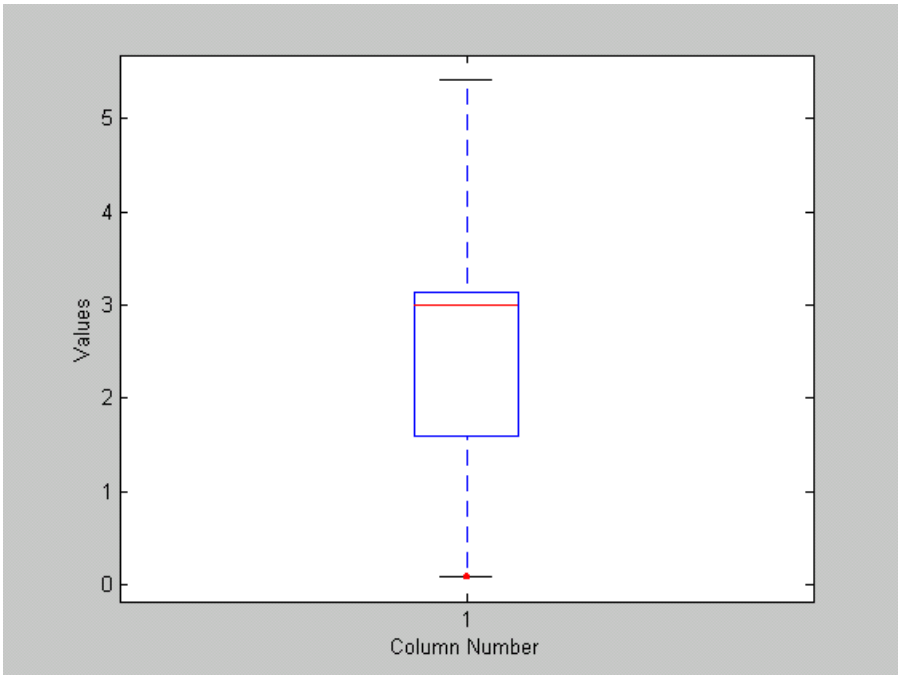


Figura 7-19 : *O novo gráfico de caixa mostra que não há mais valores aberrantes*

O segundo histograma da figura 7-18 apenas nos mostra que após a eliminação dos valores aberrantes a distribuição das magnitudes dos vetores de fluxo passa a ser mais homogênea e não há, aparentemente, valores aberrantes. Isto fica comprovado pela figura 7-19, onde um novo gráfico de caixa foi construído, agora com os vetores de fluxo óptico após eliminação dos valores aberrantes.

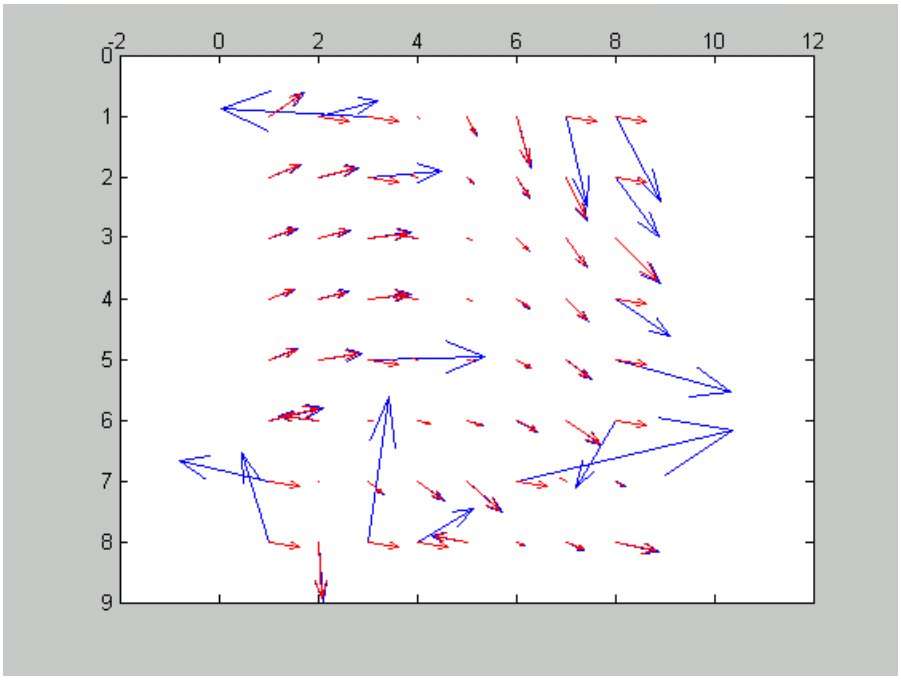


Figura 7-20 : *Mapa de fluxo resultante utilizando-se o algoritmo ponto-a-ponto: antes da identificação e transformação de valores aberrantes(em azul) e após identificação e transformação de valores aberrantes (em vermelho).*

Os mapas de fluxo apresentados na figura 7-20 servem para entender melhor o método proposto: os vetores desenhados em azul são os estimados através do algoritmo ponto-a-ponto e são exatamente os mesmos mostrados em azul na figura 7-4, quando executamos o script testa1e2 e obtivemos o mapa de fluxo através dos algoritmos ponto-a-ponto e média para o bloco (8,15). Já os vetores em vermelho são os estimados também através do algoritmo ponto-a-ponto, só que esta estimativa é feita após a etapa de identificação e transformação dos valores aberrantes.

Ou seja, primeiro identificamos os valores aberrantes através do gráfico de caixa da figura 7-17, depois substituímos os valores dos gradientes espaciais correspondentes aos fluxos aberrantes pela mediana dos gradientes do bloco e, em seguida, recalculamos os vetores de fluxo óptico pelo algoritmo ponto-a-ponto. Estes novos vetores estimados (em vermelho) não são o resultado final do algoritmo proposto, e sim, um resultado intermediário para observarmos o efeito da transformação dos valores aberrantes nos vetores de fluxo.

Como já observamos anteriormente, os fluxos em azul apontam, em sua maioria, para a direção do movimento e apresentam magnitudes semelhantes (resultado próximo ao da figura 7.1). Porém, há ainda um número considerável de pontos cujos fluxos apontam em uma direção diferente e/ou apresentam magnitudes muito distintas. Isto se deve provavelmente à homogeneidade desta região do bloco, o que leva a gradientes espaciais muito pequenos (ou próximos a zero) e a fluxos muito elevados.

O mapa de fluxo em vermelho nos mostra que todos os vetores em azul, que apresentaram magnitudes muito grandes (em comparação com os demais) foram considerados aberrantes e recalculados. Nestes pontos os vetores em vermelho assumiram todos o mesmo valor e direção (pois foram todos estimados a partir do mesmo valor de gradiente).

O mapa de fluxo resultante (em vermelho) se aproxima muito mais do mapa de fluxo da figura 7-1 do que o mapa em azul, mesmo tendo sido estimado através do algoritmo ponto-a-ponto, que sabemos apresentar um desempenho muito inferior aos demais algoritmos.

Então, o que propomos é utilizar os novos gradientes, calculados após identificação dos valores aberrantes, para estimar novamente o fluxo óptico, desta vez utilizando para isso o algoritmo do mínimo erro quadrático. O mapa de fluxo resultante é o já apresentado na figura 7-15. Abaixo, redesenhamos os vetores obtidos para o bloco escolhido:

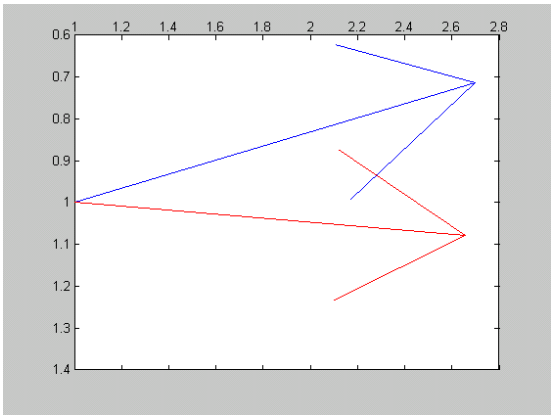


Figura 7-21 : *Os Fluxos resultantes para o bloco em questão. Em azul o vetor estimado pelo mínimo erro quadrático ponderado e em vermelho o fluxo calculado após a eliminação dos valores aberrantes (ampliação da figura 7-15).*

A figura 7-21 serve para concluir este primeiro exemplo, mostrando que com o algoritmo proposto conseguimos corrigir a direção do vetor de fluxo. Enquanto o vetor obtido através do mínimo erro quadrático ponderado (em azul) aponta em uma direção diferente da do movimento (veja figura 7-1), o vetor de fluxo resultante do método de identificação e transformação de valores aberrantes (em vermelho) aponta em uma direção mais próxima à do movimento. Neste exemplo, ambos os vetores possuem magnitudes semelhantes (aproximadamente 1,7 pixels/quadro) e muito próximas da velocidade já conhecida (entre 1,7 e 2,2 pixels/quadro).

Apesar da figura mostrar os vetores calculados pelos dois métodos apenas para o bloco em questão, podemos voltar à figura 7-15 e observar que este comportamento se repete para praticamente todos os blocos da imagem.

A seguir apresentamos um segundo exemplo, onde os mesmos algoritmos foram testados utilizando-se dois quadros da seqüência “Hamburg Taxi” e o bloco (6,14):

Figura 7-22: *Um dos quadros da seqüência "Hamburg Taxi" e os mapas de fluxo resultantes utilizando-se o mínimo erro quadrático ponderado (em azul) e utilizando-se o mínimo erro quadrático com eliminação de valores aberrantes (em vermelho). O bloco identificado por um retângulo (blocox=6 e blocoy=14) foi o utilizado para ilustrar a construção do gráfico de caixa.*

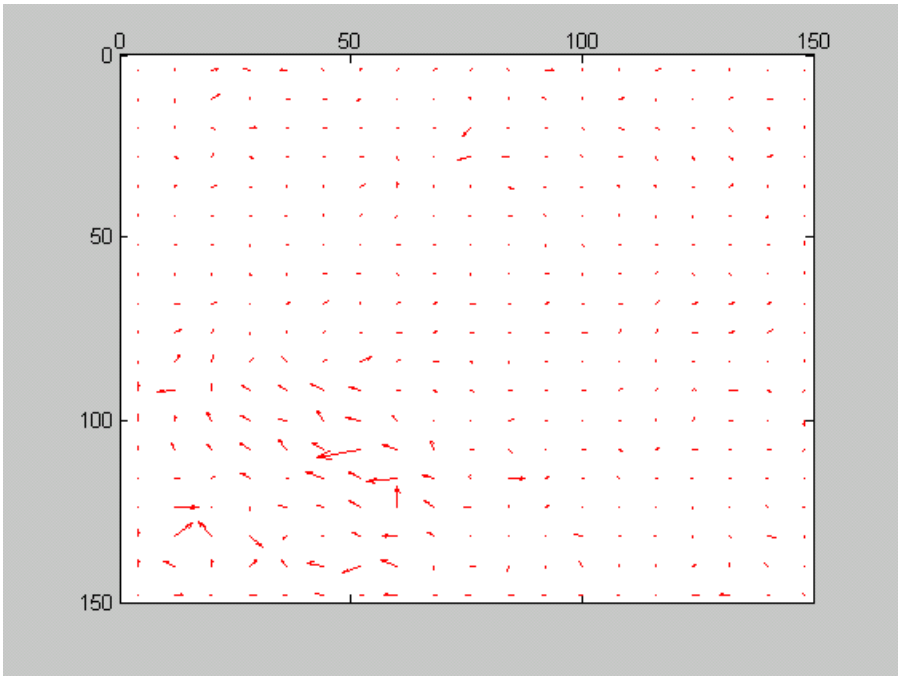


Figura 7-23 : *O mesmo mapa de fluxo obtido através do algoritmo de mínimo erro quadrático com eliminação de valores aberrantes (em vermelho), só que desta vez mostrado sozinho.*

Vemos na figura 7-22 um dos quadros da seqüência “Hamburg Taxi” utilizado neste segundo teste e os mapas de fluxo obtidos para os dois algoritmos: em azul os vetores calculados através do *mínimo erro quadrático ponderado* e em vermelho os vetores estimados pelo *mínimo erro quadrático após eliminação de valores aberrantes*. Observando estes mapas de fluxo percebemos que os mesmos são muito similares, porém os vetores em vermelho estão mais alinhados (na direção do movimento) e apresentam intensidades bem pequenas (próximas de zero) nas regiões onde não há movimento. Isto fica mais claro na figura seguinte (figura 7-23), onde observamos apenas o mapa obtido após eliminação dos valores aberrantes.

Para acompanhar a seqüência de identificação e transformação de valores aberrantes selecionamos um bloco da imagem (identificado pelo retângulo preto na figura 7-22) e geramos as figuras a seguir:

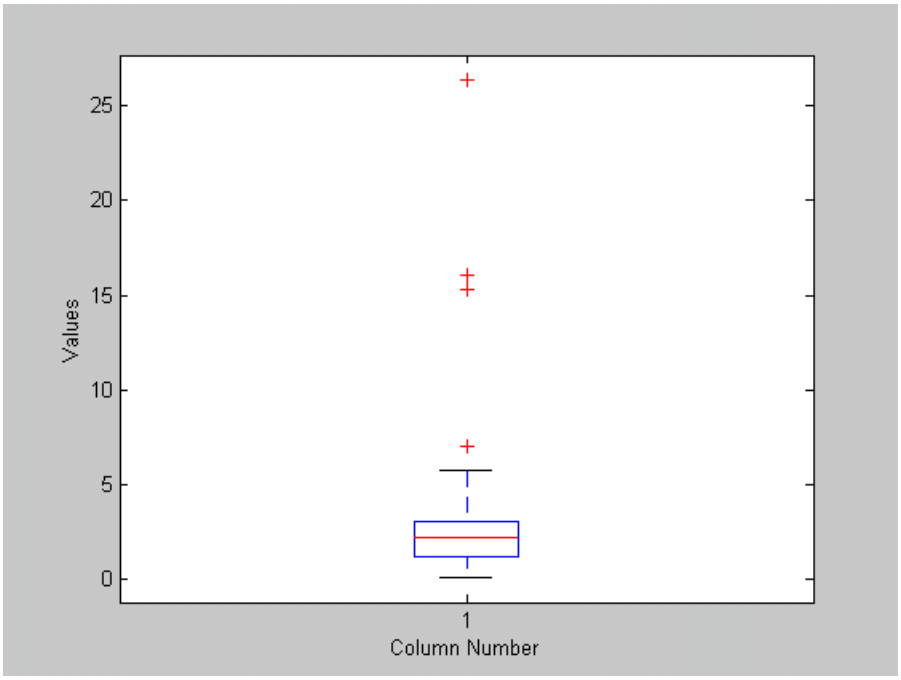


Figura 7-24 : *O gráfico de caixa (“boxplot”) mostra a caixa (1^o e 3^o quartis em azul), a mediana (linha vermelha dentro da caixa), os limites para os valores aberrantes (linhas horizontais em preto) e os valores aberrantes identificados pelas cruzes em vermelho.*

A figura 7-24 mostra o gráfico de caixa para o bloco selecionado (6,14). Nela podemos ver a mediana, o 1^{o} quartil, o 3^{o}.quartil e os bigodes (limites para os valores aberrantes). Nele identificamos os valores aberrantes, que também podem ser vistos no primeiro histograma da figura 7-25. Este histograma nos mostra que inicialmente os vetores apresentam em sua maioria valores em torno de 2 e que são poucos os vetores com magnitude superior a 8. É por isso que estes vetores são transformados através do algoritmo de valores aberrantes e o novo histograma nos mostra agora vetores com magnitude de no máximo 2.8.

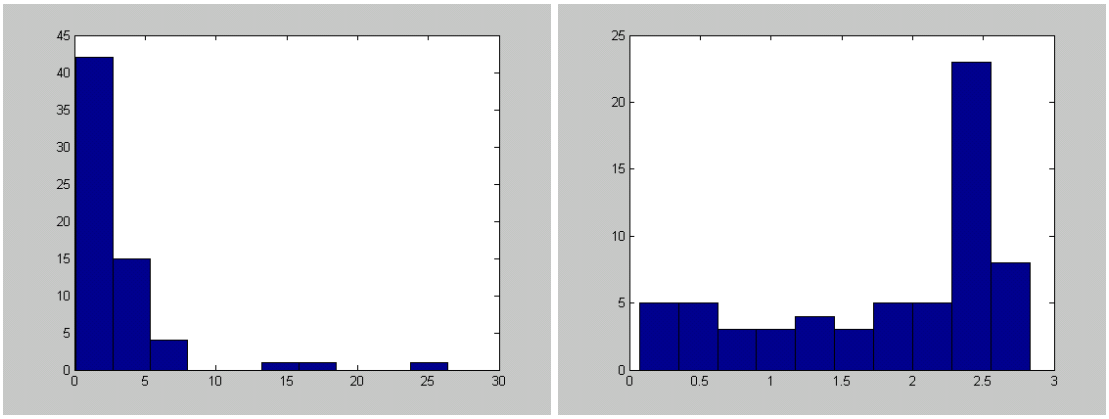


Figura 7-25 : *Histogramas das magnitudes dos vetores de fluxo estimados para o bloco (6,14) antes e após a identificação e transformação dos valores aberrantes.*

Isto fica comprovado pela figura 7-26, onde um novo gráfico de caixa foi construído, agora com os vetores de fluxo óptico calculados após eliminação dos valores aberrantes. O novo gráfico não apresenta valores aberrantes.

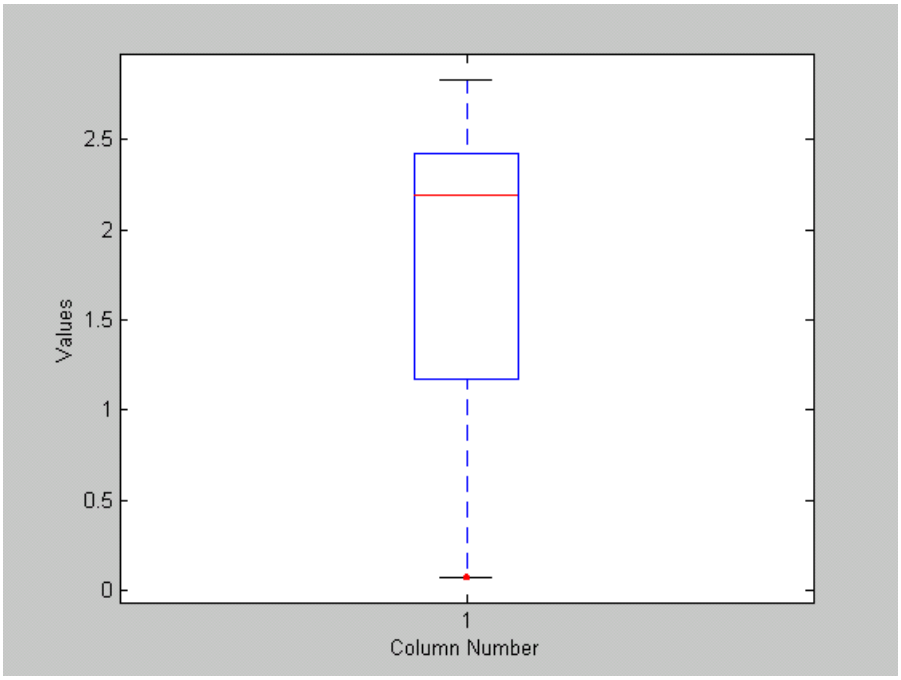


Figura 7-26 : *O novo gráfico de caixa mostra que não há mais valores aberrantes (cruzes em vermelho que deveriam se localizar abaixo da linha horizontal em zero ou acima da linha horizontal em 2.8 (linhas em preto, chamadas de bigodes).*

Na figura 7-27, os vetores desenhados em azul são os estimados através do algoritmo ponto-a-ponto e são exatamente os mesmos mostrados em azul na figura 7-10, quando executamos o script testa1e2 e obtivemos o mapa de fluxo através dos algoritmos ponto-a-ponto e média para o bloco (6,14). Já os vetores em vermelho são os estimados também através do algoritmo ponto-a-ponto, só que esta estimativa é feita após a etapa de identificação e transformação dos valores aberrantes.

Ou seja, primeiro identificamos os valores aberrantes através do gráfico de caixa da figura 7-24, depois substituímos os valores dos gradientes espaciais correspondentes aos fluxos aberrantes pela mediana dos gradientes do bloco, e em seguida recalculamos os vetores de fluxo óptico pelo algoritmo ponto-a-ponto. Estes novos vetores estimados (em vermelho) não são o resultado final do algoritmo proposto, e sim um resultado intermediário para observarmos o efeito da transformação dos valores aberrantes nos vetores de fluxo.

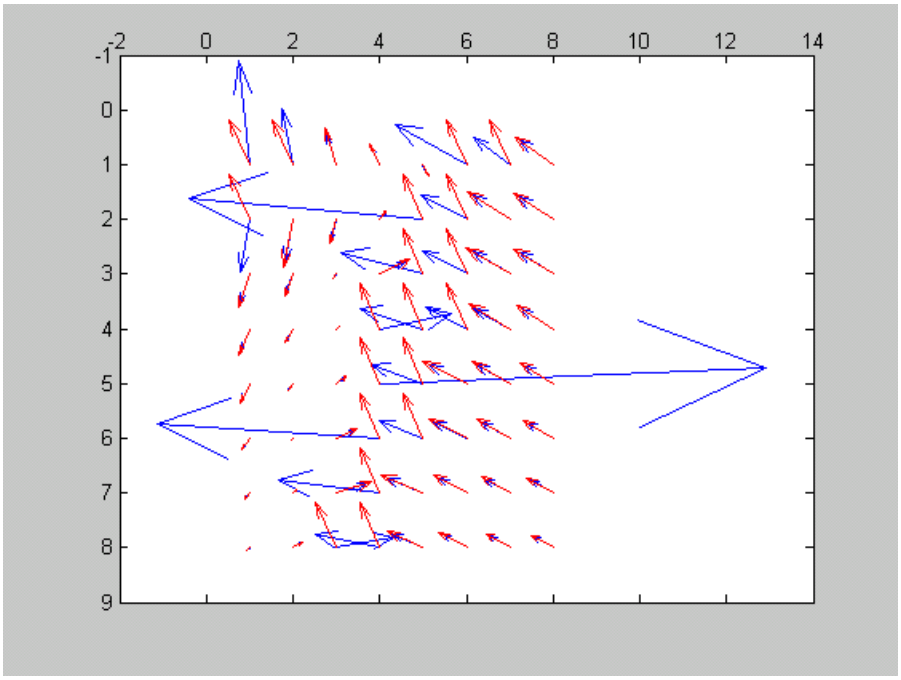


Figura 7-27 : *Mapa de fluxo resultante utilizando-se o algoritmo ponto-a-ponto: antes da identificação e transformação de valores aberrantes(em azul) e após identificação e transformação de valores aberrantes (em vermelho).*

Como já observamos anteriormente, metade dos vetores em azul apontam para a direção do movimento e apresentam magnitudes semelhantes, enquanto que a outra metade aponta para outra direção (perpendicular à borda da janela, como já discutido na figura 7-10). Há ainda um pequeno número de pontos cujos fluxos apontam em uma direção diferente e/ou apresentam magnitudes muito distintas.

O mapa de fluxo em vermelho nos mostra que todos os vetores em azul, que apresentaram magnitudes muito grandes (em comparação com os demais) foram considerados aberrantes e recalculados. Nestes pontos os vetores em vermelho assumiram todos o mesmo valor e direção (pois foram todos estimados a partir do mesmo valor de gradiente). Além disso, notamos que os vetores em vermelho se mostram um pouco mais alinhados entre si, porém um pouco “desviados” da direção do movimento. De qualquer forma, podemos considerar um bom resultado, uma vez que foi estimado através do algoritmo ponto-a-ponto, que sabemos apresentar um

desempenho muito inferior aos demais algoritmos. Quando recalculamos agora os fluxos, a partir dos novos valores de gradientes, pelo algoritmo do mínimo erro quadrático, a melhoria do mapa fica mais evidente, como pode ser visto nas figuras 7-22 e 7-23. A figura abaixo (figura 7-28) mostra os vetores obtidos desta forma para o bloco escolhido.

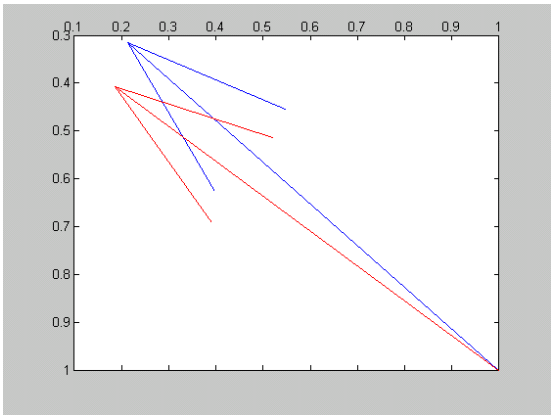


Figura 7-28 : *Os Fluxos resultantes para o bloco em questão. Em azul o vetor estimado pelo mínimo erro quadrático ponderado e em vermelho o fluxo calculado após a eliminação dos valores aberrantes (ampliação da figura 7-22).*

A figura 7-28 não nos permite tirar nenhuma conclusão, uma vez que ambos os vetores (o vetor obtido através do mínimo erro quadrático ponderado, em azul, e o vetor de fluxo resultante do método de identificação e transformação de valores aberrantes, em vermelho) apontam na mesma direção e apresentam praticamente a mesma magnitude. Como se trata de uma seqüência de imagem real, não conhecemos exatamente o fluxo real e não podemos afirmar qual dos dois vetores apresenta direção mais correta.

Se voltarmos à figura 7-22 poderemos observar que este comportamento se repete para vários blocos da imagem que se encontram em regiões onde o fluxo é nulo. Porém, há ainda regiões sem movimento, onde o algoritmo de eliminação de valores aberrantes não conseguiu corrigir o fluxo estimado, e os resultados não parecem muito melhores em comparação com o algoritmo de mínimo erro quadrático ponderado.

7.4.2 Comentários

Podemos tirar algumas conclusões observando os resultados obtidos neste exemplo.

Em primeiro lugar, o algoritmo proposto parece atenuar os efeitos de homogeneidade na estimativa de fluxo óptico. Fluxos aberrantes estimados para regiões de baixa textura são recalculados e o resultado final é um mapa de fluxo sem (ou com menos) pontos divergentes.

Já o problema da abertura parece ter ficado sem solução. Mapas de fluxo distorcidos devido a este problema (como é o caso da figura 7-27), cujos fluxos apontam na direção perpendicular ao gradiente local, não podem ser corrigidos através deste método. Isto porque estes fluxos, cujas componentes na direção normal ao movimento não foram possíveis de se obter, não são considerados aberrantes, pois às vezes representam a maioria dos pixels do bloco. Neste caso o método proposto pode até acabar considerando os poucos fluxos calculados corretamente como valores aberrantes.

Mesmo sabendo que a partir deste estudo qualitativo não podemos responder a questões quanto a efetividade do método, acreditamos que o mesmo não deva ser abandonado, pois:

- Os resultados obtidos parecem promissores;
- O método proposto abre uma gama de possibilidades a serem exploradas, utilizando o conceito de valores aberrantes, porém adotando novos tratamentos para os mesmos;
- Uma análise quantitativa pode vir a eliminar a subjetividade das conclusões e indicar as melhorias reais que o método proposto proporcionou.

Capítulo 8 – Conclusões e Trabalhos futuros

O fluxo óptico tem sido foco de uma série de estudos, não só devido à sua indiscutível utilidade, mas também por apresentar sérios desafios computacionais. O processamento de imagens contendo movimento passa normalmente pela etapa de estimativa de fluxo óptico, independentemente do que se quer alcançar com este processamento. Por isso é que encontramos ainda muitos trabalhos propondo novos métodos de estimativa de fluxo óptico ou melhorias em métodos já existentes.

8.1 Conclusões

O presente trabalho teve como objetivo não só implementar um método já existente de estimativa de fluxo óptico, permitindo sua utilização em estudos e aplicações diversos, como também propor uma modificação no algoritmo original, na tentativa de melhorar seu desempenho, principalmente em se tratando de estimativas em seqüências de imagens reais.

Observando os resultados obtidos nos testes podemos concluir que os algoritmos propostos por LK para estimativa do fluxo óptico foram implementados em MATLAB com sucesso. Os mapas de fluxo obtidos através da execução destes algoritmos se mostraram coerentes com os movimentos contidos nas imagens. Dessa forma, os interessados em trabalhar com segmentação de imagens através do movimento, análise de imagens médicas, rastreamento de alvos e demais aplicações já citadas, podem concentrar seus esforços no desenvolvimento da aplicação escolhida, utilizando as funções apresentadas neste trabalho sem necessariamente ter que entender e implementar a etapa de estimativa de fluxo óptico.

O presente trabalho serve também como ponto de partida para quem quiser entender melhor o método das diferenças para estimativa de fluxo óptico. A maneira como foram implementadas as diversas variantes do método das diferenças para estimativa de fluxo óptico permite que outros venham a entender passo a passo a evolução do método, suas variantes, seus pontos fortes e fracos. Este era um dos objetivos da tese e motivo pelo qual as várias instâncias do método foram implementadas, ao invés de somente o algoritmo de mínimo erro quadrático ou mínimo erro quadrático ponderado (variantes de melhor desempenho).

Outra contribuição deste trabalho foi uma proposta de modificação do método diferencial de estimativa de fluxo óptico originalmente apresentado por Lucas e Kanade, onde a teoria de valores aberrantes foi utilizada como medida de confiabilidade. Embora o conceito de medida de confiabilidade já tivesse sido apresentado por Lucas

[17] e defendido também por Barron, Fleet *et al*. [1,2] em seu estudo de desempenho de diversos métodos, a teoria de valores aberrantes ainda não havia sido utilizada para tal fim.

Esta modificação foi implementada e testada e os resultados obtidos mostram que a identificação de valores aberrantes e a correção dos valores dos gradientes espaciais para estimativa posterior do fluxo óptico através do algoritmo de mínimo erro quadrático é uma técnica bastante eficaz. Os mapas de fluxo obtidos a partir desta variante do método se aproximam mais dos mapas esperados, eliminando ruídos e valores indesejados.

Como era de se esperar, os resultados do método proposto obtidos com as imagens sintéticas mostraram uma melhoria mais significativa do que a melhoria obtida nos resultados com imagens reais. Isto nos faz pensar que a técnica de identificação e transformação de valores aberrantes é mais eficiente na correção de valores de fluxo não confiáveis devido a regiões de baixa textura do que devido a problemas encontrados em imagens reais, como oclusões, sombras ou ainda variações de luminosidade entre os quadros da seqüência de imagens.

8.2 Trabalhos futuros

Durante a realização desta pesquisa pudemos visualizar alguns possíveis trabalhos futuros.

A primeira possibilidade seria utilizar alguma medida de avaliação quantitativa para medir o desempenho do método proposto. Esta medida poderia ser:

- Para imagens sintéticas, onde o fluxo óptico correto é conhecido - erro absoluto, erro relativo, erro angular, RMS ou SNR

- Para imagens reais, onde não há informação do movimento – erro RMS de reconstrução de imagem

O desempenho do método proposto poderia desta forma ser comparado ao desempenho de todos os demais métodos já avaliados [13].

Outras possibilidades de trabalhos futuros seriam modificações no método proposto. Partindo da mesma idéia de utilizar o conceito de valores aberrantes como medida de confiabilidade, outros tratamentos para os valores identificados como não confiáveis poderiam ser testados. Enquanto que neste trabalho propusemos a substituição dos valores dos gradientes espaciais pela *mediana* dos gradientes calculada para o bloco em questão, outras substituições poderiam ser experimentadas, como, por exemplo, substituir os valores não confiáveis por uma média ponderada.

Como previsto na teoria de valores aberrantes, outro tratamento possível para estes valores após sua identificação é sua eliminação. Neste caso, uma alternativa seria eliminar os valores dos gradientes para os pontos identificados como não confiáveis e recalcular o fluxo óptico por algum dos algoritmos propostos por LK, ignorando os pontos para os quais os gradientes foram eliminados.

Para finalizar as sugestões de extensões deste trabalho, tentativas poderiam ser realizadas para se obter melhores resultados para seqüências de imagens reais. Antes de estimar o fluxo óptico através do método proposto, alguma técnica para eliminar as variações de luminância entre os quadros poderia ser aplicada. Os possíveis tratamentos de oclusões, sombras e transparência já propostos por outros autores [43,44,45,46] aplicados às imagens antes da etapa de identificação de valores aberrantes também poderia ser objeto de estudo e poderia contribuir para a melhoria de desempenho do método proposto, quando aplicado a seqüências de imagens reais.

Referências Bibliográficas

[1] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of Optical Flow Techniques.", University of Western Ontario, *Technical Report n° 229*, 1992.

[2] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of Optical Flow Techniques.", *International Journal of Computer Vision*, vol. 12, pp. 43-77, 1994.

[3] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision.", *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pp. 674-679, Vancouver, 1981.

[4] D. J. Fleet and A. D. Jepson, "Computation of component image velocity from local phase information.", *International Journal of Computer Vision*, vol. 5, pp. 77-104, 1990.

[5] B. Galvin, B. McCane, K. Novins, D. Mason, and S. Mills, "Recovering motion fields: An evaluation of eight optical flow algorithms.", *British Machine Vision Conference 98*, 1998.

[6] A. D. Marshall, "Vision Systems.", *World Scientific*, 1992. [http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/MARSHALL]

[7] R. Owens, "Computer Vision IT412 - Lecture 12.", *Lecture notes*, 1997. [http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/OWENS/LECT12]

[8] B. K. P. Horn and B. G. Schunck, "Determining optical flow.", *Artificial Intelligence*, vol. 17, pp. 185-203, 1981.

[9] A. Mitiche and P. Bouthemy, "Computation and analysis of image motion: A synopsis of current problems and methods.", *International Journal of Computer Vision*, vol. 19, pp. 29-55, 1996.

[10] S. Ullman, "The interpretation of visual motion.", *MIT Press,* Cambridge, London, 1979.

[11] R. M. Haralick and J. S. Lee, “The facet approach to optic flow.”, *Proceedings of Image Understanding Workshop*, Palo Alto, pp. 84-93,1982.

[12] M. G. Ross, "Exploiting texture-motion duality in optical flow and image segmentation.", *Master Thesis*, Massachusetts Institute of Technology, 2000.

[13] S. S. Beauchemin and J. L. Barron, "The computation of optical flow.", *Acm Computing Surveys*, vol. 27, pp. 433-467, 1995.

[14] J. K. Aggarwal and N. Nandhakumar, "On the computation of motion from sequences of images - A review.", *Proceedings IEEE*, vol. 76, pp. 917-935, 1988.

[15] S. M. Smith, "Reviews of optical flow, motion segmentation, edge finding and corner finding.", Oxford University, Oxford, *Technical Report TR97SMS1*, 1997. [http://fmrib.ox.ac.uk/~steve/review/review/review.html]

[16] E. C. Hildreth, "The computation of the velocity field.", *Proceedings of Royal Society London*, vol. B 221, pp. 189-220, 1984.

[17] B. D. Lucas, "Generalized Image Matching by the method of differences.", *PhD Dissertation*, Dept. of Computer Science, Carnegie-Mellon University, 1984.

[18] H. H. Nagel, "Displacements vectors derived from second-order intensity variations in image sequences.", *CGIP21*, pp. 85-117, 1983.

[19] E. H. Adelson and J. R. Bergen, "Spatiotemporal energy models for the perception of motion.", *Journal of the Optical Society of America*, vol. A 2, pp. 284-299, 1985.

[20] N. M. Grzywacz and A. L. Yuille, "A model for the estimate of local velocity by cells in the visual cortex.", *Proceedings of Royal Society London*, vol. B 239, pp. 129-161, 1990.

[21] D. J. Heeger, "Optical flow using spatiotemporal filters.", *International Journal of Computer Vision*, vol. 1, pp. 279-302, 1988.

[22] A. B. Watson and A. J. Ahumada Jr., "Model of human visual-motion sensing.", *Journal of the Optical Society of America*, vol. A 2, pp. 322-341, 1985.

[23] P. Anandan, "Measuring visual motion from image sequences.", Amherest, *PhD Dissertation*, University of Massachusetts, *COINS TR 87-21*,1987.

[24] P. Anandan, "A computational framework and an algorithm for the measurement of visual motion.", *International Journal of Computer Vision*, vol. 2, pp. 283-310, 1989.

[25] J. J. Little, H. H. Bulthoff, and T. A. Poggio, "Parallel optical flow using local voting.", *Proceedings IEEE International Conference on Computer Vision*, Tampa, pp. 454-459, 1988.

[26] A. Singh, "An estimation-theoretic framework for image-flow computation.", *Proceedings IEEE International Conference on Computer Vision*, Osaka, pp.168-177, 1990.

[27] A. Singh, "Optical flow computation: A unified perspective.", *IEEE Computer Society Press*, 1992.

[28] S. Uras, F. Girosi, A. Verri, and V. Torre, "A computational approach to motion perception.", *Biol. Cybern.*, vol. 60, pp. 79-97, 1988.

[29] H. H. Nagel, "On the estimation of optical flow: Relations between different approaches and some new results.", *Artificial Intelligence 33*, pp. 299-324, 1987.

[30] A. M. Waxman, J. Wu, and F. Bergholm, "Convected activation profiles and receptive fields for real time measurement of short range visual motion.", *Proceedings IEEE Computer Vision and Pattern Recognition*, Ann Arbor, pp. 717-723, 1988.

[31] C. Butcher, “Mirage. A Ray tracing package.” [http://Atlas.otago.ac.nz:800/students/butcher_cc/pages/mirage.html]

[32] T. Camus, "Real-time quantized optical flow.", *The journal of real-time Imaging*, vol. 3, pp. 71-86, 1997.

[33] M. Proesmans, L. Van Gool, E. Pauwels, and A. Osterlinck, "Determination of optical flow and its discontinuities using non-linear diffusion.", *European Conference on Computer Vision - ECCV’94*, vol. 2, pp. 295-304, 1994.

[34] D. Young, "Sussex Computer Vision - TEACH VISION 6.", 1994. [http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/YOUNG/vision6.html]

[35] M. M. C. Figueira, "Identificação de outliers.", *Millenium* (12), 1998. [http://www.ipv.pt/millenium/arq12.htm]

[36] M. E. Young, "How to Construct a Box Plot.", 2002.
[http://www.siu.edu/departments/cola/psycho/faculty/young/ResMethodsStuff/howtoboxplot.htm]

[37] “Quantile-Quantile Plot.”, NIST/SEMATECH e-Handbook of Statistical Methods, 2003.
[http://www.itl.nist.gov/div898/handbook/eda/section3/qqplot.htm]

[38] “Interpretation of Quantile-Quantile and Probability Plots.”, *QQPLOT Statement - SAS/QC User's Guide*, 1999.
[http://www.rz.tu-clausthal.de/qc/chap10/ sect10.htm]

[39] R. High, "Dealing with 'Outliers': How to Maintain Your Data's Integrity.", *Computing News*, 2000.
[http://cc.uoregon.edu/cnews/spring2000/outliers.html]

[40] E. P. Simoncelli, E. H. Adelson, and D. J. Heeger, "Probability distributions of optical flow.", *Proceedings IEEE of Computer Vision and Pattern Recognition*, Maui, 1991.

[41] “MATLAB Tutorial.”
[http://www.mathworks.com]

[42] A. Calway, “Motion estimation.”, *Image processing and Computer Vision – COM 530121*, Lecture notes, Bristol University, 2003.
[http://www.cs.bris.ac.uk/teaching/resources]

[43] V. Markandey and B. E. Flinchbaugh. “Multispectral constraints for optical flow computation.”, *Proceedings of International Conference on Computer Vision*, pp. 38-41, Osaka, 1990.

[44] N. Mukawa, “Estimation of shape, reflection, coefficients and illuminant direction from image sequences.”, *Proceedings of International Conference on Computer Vision*, pp. 507-512, Osaka, 1990.

[45] R. J. Woodham. “Multiple light source optical flow.”, *Proceedings of International Conference on Computer Vision*, pp. 42-46, Osaka, 1990.

Apêndice 1 – Máscaras Gaussianas

A função Gaussiana

Uma função gaussiana é dada pela expressão:

$$g_{2D}(r,\sigma) = \frac{1}{2\pi\sigma}\exp\left(\frac{r^2}{2\sigma^2}\right)$$

Com o desvio padrão (σ) da função gaussiana pode-se variar a amplitude da mesma. Um desvio padrão pequeno gerará uma função estreita com um pico pronunciado e um σ elevado proporcionará uma função com uma queda suave.

O termo fora da exponencial serve para que a área total da função seja sempre igual a 1. Obrigar que a área debaixo da função seja sempre igual a 1 garante que os valores filtrados permaneçam dentro da mesma faixa que os valores da imagem original.

Para determinar o tamanho da máscara discreta que se obtém a partir da função contínua (ou seja, o número de amostras que devem ser tomadas) pode-se utilizar o critério de amostrar somente o centro da função, onde tem-se valores significativos da mesma. Conforme se distancia da origem, a função se aproxima cada vez mais de zero e chegará um momento em que os valores da função serão tão próximos de zero que se pode desprezá-los sem que isso leve a cometer erros significativos nos cálculos.

Uma vez que a função se aproxima mais ou menos rapidamente de zero em função do valor de σ, é preciso calcular o tamanho da máscara também em função de σ. O critério habitual é tomar uma largura de filtro tal que sobre ele caia a maior parte da área da função. Por exemplo, tomando largura=5*σ assegura-se que a função está sendo amostrada cobrindo uma área de 98.76%. Outros autores consideram que a largura deve ser de 8 vezes o desvio padrão e já que os núcleos de convolução costumam ter um número impar de elementos, toma-se, por exemplo, a largura como o inteiro mais próximo a 8*σ +1.

Tamanhos maiores para a máscara fazem com que os cálculos sejam mais custosos em troca de não melhorar praticamente em nada os resultados. Tamanhos menores da máscara aceleram os cálculos mas pioram os resultados.

Pode-se reconstruir a função original a partir da função amostrada desde que o desvio padrão escolhido seja suficientemente grande (maior que 0,8). Para σ muito pequeno a reconstrução do sinal original é muito ruim.

O uso de máscaras Gaussianas para suavização de imagens

O uso de uma máscara Gaussiana para a suavização de imagens tem se tornado extremamente popular. Isto tem a ver com certas propriedades da Gaussiana (o teorema do limite central, produto com largura de banda mínima). Além disso, o filtro Gaussiano é separável, ou seja, o filtro bidimensional pode ser escrito como o produto de dois filtros Gaussianos unidimensionais:

$$h(x,y) = g_{2D}(x,y) = \left\{\frac{1}{\sqrt{2\pi\sigma}}\exp\left[-\left(\frac{x^2}{2\sigma^2}\right)\right]\right\}\cdot\left\{\frac{1}{\sqrt{2\pi\sigma}}\exp\left[-\left(\frac{y^2}{2\sigma^2}\right)\right]\right\}$$

$$= g_{1D}(x)\cdot g_{1D}(y)$$

Há quatro maneiras distintas de se implementar o filtro Gaussiano:

- Convolução, usando um número finito de amostras (N_0) do filtro Gaussiano como máscara da convolução.

$$g_{1D}[n] = \begin{cases} \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\left(\frac{x^2}{2\sigma^2}\right)\right] & |n| \le N_0 \\ 0 & |n| > N_0 \end{cases}$$

- Convolução repetitiva, usando um filtro uniforme como máscara da convolução.

$$g_{1D}[n] \approx u[n] \otimes u[n] \otimes u[n]$$

$$u[n] = \begin{cases} \frac{1}{(2N_0+1)} & |n| \le N_0 \\ 0 & |n| > N_0 \end{cases}$$

- Multiplicação no domínio da freqüência, já que a transformada de Fourier de uma Gaussiana é uma Gaussiana.

- Uso de um filtro recursivo, que tem resposta ao impulso infinita

O uso de máscaras Gaussianas para cálculo do gradiente

Um dos problemas fundamentais no processamento de imagens é calcular sua derivada espacial. Como a imagem digital não é uma função contínua das variáveis espaciais, mas somente uma função discreta de coordenadas espaciais inteiras, o cálculo de sua derivada pode ser considerado apenas uma aproximação da derivada espacial da imagem real (esta sim, contínua no espaço).

Como uma imagem é uma função bidimensional no espaço, primeiro, é preciso definir em que direção a derivada será tomada: se na direção horizontal , na direção vertical ou ainda em uma direção arbitrária que pode ser considerada uma combinação destas duas. Se usarmos h_x para denotar o filtro derivativo horizontal e h_y para denotar o filtro derivativo vertical, teremos um número de escolhas possíveis para h_x e h_y, sendo que filtros Gaussianos são os mais utilizados em processamento digital moderno.

Apêndice 2 – Functions e Scripts

As functions e scripts a seguir foram desenvolvidos em MATLAB, com o objetivo de implementar os algoritmos discutidos neste trabalho. Como explicado anteriormente, as *functions* podem ser estudadas de forma isolada por aqueles que quiserem entender apenas uma etapa do processamento, já que cada uma delas se refere a um algoritmo em particular.

Já os testes dos algoritmos foram realizados através de *scripts*, de forma a permitir a escolha de valores diferentes para as variáveis de testes, sem necessidade de se alterar as *functions* dos algoritmos. Para se executar uma seqüência completamente diferente de testes, basta escrever um novo *script*.

Os comentários ao longo das *functions* e *scripts* listados se encontram em inglês para facilitar o intercâmbio destes arquivos entre a comunidade acadêmica.

```
function G = gauss(sig)

%  GAUSS Generate a gaussian vector.
%
%  G =  GAUSS( SIG )
%
%  Input:
%    SIG - Standard deviation.
%
%  Output:
%    G   - Gaussian vector centered on zero and std deviation SIG.
%
%    A Gaussian vector is a vector with a Gaussian distribution.
%    It can be used to generate test patterns or Gaussian filters
%    both for spatial and frequency domain.
%

% Size the vector appropriately.

x = floor(-4*sig):ceil(4*sig);

% Calculate the gaussian.

G =exp(-0.5*x.^2/sig^2);

% Scale the gaussian.

G = G/sum(G);
```

```
function dG = dgauss(sig)

%  DGAUSS Generate a derivative of a gaussian vector.
%
%  dG =  DGAUSS( SIG )
%
%  Input:
%    SIG - Standard deviation.
%
%  Output:
%    dG - Derivative of a Gaussian vector centered on zero and
%         std deviation equal to SIG.
%

% Size the vector appropriately.

x = floor(-4*sig):ceil(4*sig);

% Calculate the gaussian.

G = exp(-0.5*x.^2/sig^2);

% Scale the gaussian.

G = G/sum(G);

% Calculate the derivative of the gaussian.

dG = -x.*G/sig^2;
```

```
function [Ex,Ey,Et] = Gradient(i1,i2,sig)

%  GRADIENT Calculate spatial and temporal gradients of a pair of images.
%
%  [Ex, Ey, Et] =  GRADIENT ( I1, I2, SIG )
%
%  Input:
%    I1 - Input image. First image of a sequence.
%    I2 - Input image. Second image of a sequence.
%    SIG- Standard deviation.
%
%  Output:
%    Ex - Spatial gradient of I1 in X direction.
%    Ey - Spatial gradient of I1 in Y direction.
%    Et - Temporal gradient between I1 and I2.
%
%
%    Calculate spatial and temporal gradients of a pair of images.
%    Uses Gaussian kernels to do it. Pre-filter both images before
%    it calculates the temporal gradient, in order to attenuate
%    temporal aliasing and quantization effects in the input.
%

%Create Gaussian kernels to calculate spatial gradients
g = gauss(sig);
dg = dgauss(sig);
kx = g'*dg;
ky = dg'*g;
k = g'*g;

%Calculate spatial gradients
Ex=conv2(i1,kx,'same');
Ey=conv2(i1,ky,'same');

%Filter the images
i1_blurred=conv2(i1,k,'same');
i2_blurred=conv2(i2,k,'same');

%Calculate temporal gradient
Et=i1_blurred-i2_blurred;

return
```

```
function [u,v] = flow(Ex,Ey,Et)

%  FLOW Estimate optical flow using single point algorithm.
%
%  [u, v] =  FLOW( Ex, Ey, Et )
%
%  Input:
%    Ex - Spatial gradient of I1 in X direction.
%    Ey - Spatial gradient of I1 in Y direction.
%    Et - Temporal gradient between I1 and I2.
%
%  Output:
%    u - Component of the optical flow vector in Y direction.
%    v - Component of the optical flow vector in X direction.
%
%    Estimate the optical flow by the method of differences.
%    Given the spatial and temporal gradients, it estimates
%    the optical flow for each pixel.
%

%Initialize the optical flow vector;
 u = zeros(size(Ey));
 v = zeros(size(Ex));

%Estimate the optical flow;
 u = Et./Ey;
 v = Et./Ex;

 return
```

```
function [u,v] = flow_average(Ex,Ey,Et)

%  FLOW_AVERAGE Estimate optical flow using average algorithm.
%
%  [u, v] =  FLOW_MEAN( Ex, Ey, Et )
%
%  Input:
%    Ex - Spatial gradient of I1 in X direction.
%    Ey - Spatial gradient of I1 in Y direction.
%    Et - Temporal gradient between I1 and I2.
%
%  Output:
%    u - Component of the optical flow vector in Y direction.
%    v - Component of the optical flow vector in X direction.
%
%    Estimate the optical flow by the method of differences.
%    Given the spatial and temporal gradients, it estimates
%    the optical flow for each pixel and then take the average
%    of the result as the optical flow for each 8 X 8 block.
%

%Initialize the optical flow vector;
 u = zeros(size(Ey));
 v = zeros(size(Ex));

%Estimate the optical flow;
 u = Et./Ey;
 v = Et./Ex;

%Calculate the mean value of u and v for each 8 X 8 blocks;
 soma = inline('sum(sum(x))');
 u = (blkproc(u,[8 8],soma))/64;
 v = (blkproc(v,[8 8],soma))/64;

 return
```

```
function [u,v] = flow_lsquare(Ex,Ey,Et)

%  FLOW_LSQUARE Estimate optical flow using least-squares algorithm.
%
%  [u, v] =  FLOW_LSQUARE( Ex, Ey, Et )
%
%  Input:
%    Ex - Spatial gradient of I1 in X direction.
%    Ey - Spatial gradient of I1 in Y direction.
%    Et - Temporal gradient between I1 and I2.
%
%  Output:
%    u - Component of the optical flow vector in Y direction.
%    v - Component of the optical flow vector in X direction.
%
%    Estimate the optical flow by the method of differences.
%    Given the spatial and temporal gradients, it estimates
%    the optical flow for each 8 X 8 block using the
%    least-squares algorithm.
%

 %Prepare combined gradients
 Ex2 = (Ex.^2);
 Exy = (Ex.*Ey);
 Ey2 = (Ey.^2);
 Ext = (Ex.*Et);
 Eyt = (Ey.*Et);

 %Group gradients in 8 X 8 blocks;
 soma = inline('sum(sum(x))');
 Ex2_bloco = blkproc(Ex2,[8 8],soma);
 Ey2_bloco = blkproc(Ey2,[8 8],soma);
 Exy_bloco = blkproc(Exy,[8 8],soma);
 Ext_bloco = blkproc(Ext,[8 8],soma);
 Eyt_bloco = blkproc(Eyt,[8 8],soma);

%Initialize the optical flow vector;
 u = zeros(size(Ey2_bloco));
 v = zeros(size(Ex2_bloco));

%Calculate u and v for each block
 for k = 1:prod(size(Ex2_bloco))
     A = [Ex2_bloco(k) Exy_bloco(k); Exy_bloco(k) Ey2_bloco(k)];
     b = [Ext_bloco(k);Eyt_bloco(k)];
     aux = (inv(A)*b);
     v(k) = aux(1);
     u(k) = aux(2);
 end

 return
```

```
function [u,v] = flow_weight(Ex,Ey,Et,W2)

%  FLOW_WEIGHT Estimate optical flow using weighted least-squares algorithm.
%
%  [u, v] =  FLOW_WEIGHT( Ex, Ey, Et, W2 )
%
%  Input:
%    Ex - Spatial gradient of I1 in X direction.
%    Ey - Spatial gradient of I1 in Y direction.
%    Et - Temporal gradient between I1 and I2.
%    W2 - Window function
%
%  Output:
%    u - Component of the optical flow vector in Y direction.
%    v - Component of the optical flow vector in X direction.
%
%    Estimate the optical flow by the method of differences.
%    Given the spatial and temporal gradients, it estimates
%    the optical flow for each 8 X 8 block using the weighted
%    least-squares algorithm.
%

 %Prepare combined weighted gradients
 Ex2 = W2.*(Ex.^2);
 Exy = W2.*(Ex.*Ey);
 Ey2 = W2.*(Ey.^2);
 Ext = W2.*(Ex.*Et);
 Eyt = W2.*(Ey.*Et);

 %Group gradients in 8 X 8 blocks;
 soma = inline('sum(sum(x))');
 Ex2_bloco = blkproc(Ex2,[8 8],soma);
 Ey2_bloco = blkproc(Ey2,[8 8],soma);
 Exy_bloco = blkproc(Exy,[8 8],soma);
 Ext_bloco = blkproc(Ext,[8 8],soma);
 Eyt_bloco = blkproc(Eyt,[8 8],soma);

%Initialize the optical flow vector;
 u = zeros(size(Ey2_bloco));
 v = zeros(size(Ex2_bloco));

 %Calculate u and v for each block
 for k = 1:prod(size(Ex2_bloco))
     A = [Ex2_bloco(k) Exy_bloco(k); Exy_bloco(k) Ey2_bloco(k)];
     b = [Ext_bloco(k);Eyt_bloco(k)];
     aux = (inv(A)*b);
     v(k) = aux(1);
     u(k) = aux(2);
 end

 return
```

```
function [u,v,Ex,Ey] = flow_outliers(Ex,Ey,Et)

%  FLOW_OUTLIERS Estimate optical flow after eliminating outliers.
%
%  [u, v, Ex, Ey] =  FLOW_OUTLIERS( Ex, Ey, Et, W2 )
%
%  Input:
%    Ex - Spatial gradient of I1 in X direction.
%    Ey - Spatial gradient of I1 in Y direction.
%    Et - Temporal gradient between I1 and I2.
%    W2 - Window function
%
%  Output:
%    u - Component of the optical flow vector in Y direction.
%    v - Component of the optical flow vector in X direction.
%    Ex - Recalculated spatial gradient of I1 in X direction.
%    Ey - Recalculated spatial gradient of I1 in Y direction.
%
%    Estimate the optical flow by the method of differences.
%    Given the spatial and temporal gradients, it estimates
%    the optical flow for each 8 X 8 block using the single
%    point algorithm. Then identifies and eliminates outliers
%    and recalculate the gradients for those points. Estimate
%    again the optical flow without the outliers using the
%    least-squares algorithm.
%

 %Initialize vectors;
 soma = inline('sum(sum(x))');
 aux1 = (Ex.^2);
 aux2_bloco = blkproc(aux1,[8 8],soma);
 u = zeros(size(aux2_bloco));
 v = zeros(size(aux2_bloco));

 %Call the function 'outliers' for each block of the image
 for k = 1:prod(size(aux2_bloco))
     [blcy blcx]=ind2sub(size(aux2_bloco),k);
     if blcx~=19 & blcy~=19
         [Ex Ey]=outliers(Ex,Ey,Et,blcx,blcy);
     end
 end

 %Prepare combined weighted gradients
 Ex2 = (Ex.^2);
 Exy = (Ex.*Ey);
 Ey2 = (Ey.^2);
 Ext = (Ex.*Et);
 Eyt = (Ey.*Et);

 %Group gradients in 8 X 8 blocks;
 Ex2_bloco = blkproc(Ex2,[8 8],soma);
 Ey2_bloco = blkproc(Ey2,[8 8],soma);
 Exy_bloco = blkproc(Exy,[8 8],soma);
 Ext_bloco = blkproc(Ext,[8 8],soma);
 Eyt_bloco = blkproc(Eyt,[8 8],soma);

 %Calculate u and v for each block
```

```
for k = 1:prod(size(aux2_bloco))
    A = [Ex2_bloco(k) Exy_bloco(k); Exy_bloco(k) Ey2_bloco(k)];
    b = [Ext_bloco(k);Eyt_bloco(k)];
    aux = (inv(A)*b);
    v(k) = aux(1);
    u(k) = aux(2);
end

return
```

```
function [Ex,Ey] = outliers(Ex,Ey,Et,blcx,blcy)

%  OUTLIERS Identifies and eliminates outliers, calculating new values
%           for spatial gradients.
%
%  [Ex, Ey] =  OUTLIERS( Ex, Ey, Et, blcx, blcy )
%
%  Input:
%    Ex - Spatial gradient of I1 in X direction.
%    Ey - Spatial gradient of I1 in Y direction.
%    Et - Temporal gradient between I1 and I2.
%   blcx- X-coordenate of the choosen block.
%   blcy- Y-coordenate of the choosen block.
%
%  Output:
%    Ex - New spatial gradient of the input image in X direction,
%         after eliminating outliers.
%
%    Ey - New spatial gradient of the input image in Y direction,
%         after eliminating outliers.
%
%
%    Recalculates spatial gradientes after identifying and
%    eliminating outliers. It uses a boxplot graphic to
%    identify outliers. New values to Ex and Ey are calculated
%    using the median of the optical flow estimated for the
%    choosen block.
%
%


Exbloco=Ex(1+(blcy-1)*8:8+(blcy-1)*8,1+(blcx-1)*8:8+(blcx-1)*8);
Eybloco=Ey(1+(blcy-1)*8:8+(blcy-1)*8,1+(blcx-1)*8:8+(blcx-1)*8);
Etbloco=Et(1+(blcy-1)*8:8+(blcy-1)*8,1+(blcx-1)*8:8+(blcx-1)*8);

[u1 v1]=flow(Exbloco,Eybloco,Etbloco);

%Calculate IQR and identify outliers
c=complex(v1,u1);
maguv=abs(c);
anguv=angle(c);
[h valorh]=hist(maguv(:));
lim=valorh(10)/10;
indice=find(maguv>=1.5*iqr(maguv(:)));

%Calculate the median of u,v
mmag=median(median(maguv));
mang=median(median(anguv));
[vmed umed]=pol2cart(mang,mmag);

%Recalculate Ex and Ey using umed e vmed
Exbloco(indice)=Etbloco(indice)./vmed;
Eybloco(indice)=Etbloco(indice)./umed;
Ex(1+(blcy-1)*8:8+(blcy-1)*8,1+(blcx-1)*8:8+(blcx-1)*8)=Exbloco;
Ey(1+(blcy-1)*8:8+(blcy-1)*8,1+(blcx-1)*8:8+(blcx-1)*8)=Eybloco;
```

Script testa1e2

```
%This script tests the first two algorithms presented by Lucas:
%  - point-to-point
%  - media
%It uses two frames of the "Translating tree" image.


% Opens two frames from the image sequence and uses the "reshape"
% command to convert the binary image into a 150X150 matrix
fid=fopen('new2binarytreet.3');
f1=fread(fid);
f1=reshape(f1,150,150);
f1=f1';
figure(1);
imagesc(f1);
colormap(gray);
fid=fopen('new2binarytreet.4');
f2=fread(fid);
f2=reshape(f2,150,150);
f2=f2';

%Identifies the chosen block
blocox=13;
blocoy=8;
faux=f1;
faux(1+(blocoy-1)*8,1+(blocox-1)*8:8+(blocox-1)*8)=0;
faux(8+(blocoy-1)*8,1+(blocox-1)*8:8+(blocox-1)*8)=0;
faux(1+(blocoy-1)*8:8+(blocoy-1)*8,1+(blocox-1)*8)=0;
faux(1+(blocoy-1)*8:8+(blocoy-1)*8,8+(blocox-1)*8)=0;
figure(1);
hold on;
imagesc(faux);
colormap(gray);

%Calculates the gradients
[Ex Ey Et]=Gradient(f1,f2,1.5);

%Calculates the optical flow using the single point algorithm
%Shows it only for the chosen block
[u1 v1]=flow(Ex,Ey,Et);
u1b=u1(1+(blocoy-1)*8:8+(blocoy-1)*8,1+(blocox-1)*8:8+(blocox-1)*8);
v1b=v1(1+(blocoy-1)*8:8+(blocoy-1)*8,1+(blocox-1)*8:8+(blocox-1)*8);
figure(2);
quiver(v1b,u1b,4,'k');
set(gca,'YDir','reverse');
set(gca,'XAxisLocation','Top');

%Calculates the optical flow using the media algorithm
%Shows it only for the chosen block
[u2 v2]=flow_average(Ex,Ey,Et);
u2b=u2(blocoy,blocox);
v2b=v2(blocoy,blocox);
hold on;
x=4;
y=4;
quiver(x,y,v2b,u2b,'r');
```

Script testa3e4

```
%This script tests two of the algorithms presented by Lucas:
%  - least-squares
%  - weigted least-squares
%It uses two frames of the "Translating tree" image.


% Opens two frames from the image sequence and uses the "reshape"
% command to convert the binary image into a 150X150 matrix
fid=fopen('new2binarytreet.1');
f1=fread(fid);
f1=reshape(f1,150,150);
f1=f1';
figure(1);
imagesc(f1);
colormap(gray);
fid=fopen('new2binarytreet.2');
f2=fread(fid);
f2=reshape(f2,150,150);
f2=f2';

%Calculates the gradients
[Ex Ey Et]=Gradient(f1,f2,1.5);

%Calculates the optical flow using the least-squares algorithm
[u3 v3]=flow_lsquare(Ex,Ey,Et);
x=[4:8:150];
y=[4:8:150];
hold on;
quiver(x,y,v3,u3,2,'b');
set(gca,'YDir','reverse');
set(gca,'XAxisLocation','Top');
hold on;

%Creates the W function
W=[0.0625 0.25 0.375 0.25 0.0625];
W2=W'*W;
W2=imresize(W2,[8 8],'bilinear');
W2=repmat(W2,19,19);
W2=W2(1:150,1:150);

%Calculates the optical flow using the weighted least-squares algorithm
[u4 v4]=flow_weight(Ex,Ey,Et,W2);
quiver(x,y,v4,u4,2,'r');
```

Script testa4e5

```
%This script tests two algorithms:
%  - weigted least-squares
%  - least-squares after eliminating outliers
%It uses two frames of the "Translating tree" image.

% Opens two frames from the image sequence and uses the "reshape"
% command to convert the binary image into a 150X150 matrix
fid=fopen('new2binarytreet.5');
f1=fread(fid);
f1=reshape(f1,150,150);
f1=f1';
figure(1);
imagesc(f1);
colormap(gray);
fid=fopen('new2binarytreet.6');
f2=fread(fid);
f2=reshape(f2,150,150);
f2=f2';

%Identifies the chosen block
blocox=8;
blocoy=15;
faux=f1;
faux(1+(blocoy-1)*8,1+(blocox-1)*8:8+(blocox-1)*8)=0;
faux(8+(blocoy-1)*8,1+(blocox-1)*8:8+(blocox-1)*8)=0;
faux(1+(blocoy-1)*8:8+(blocoy-1)*8,1+(blocox-1)*8)=0;
faux(1+(blocoy-1)*8:8+(blocoy-1)*8,8+(blocox-1)*8)=0;
figure(1);
hold on;
imagesc(faux);
colormap(gray);

%Calculates the gradients
[Ex Ey Et]=Gradient(f1,f2,1.5);

%Creates the W function
W=[0.0625 0.25 0.375 0.25 0.0625];
W2=W'*W;
W2=imresize(W2,[8 8],'bilinear');
W2=repmat(W2,19,19);
W2=W2(1:150,1:150);

%Calculates the optical flow using the weighted least-squares algorithm
[u4 v4]=flow_weight(Ex,Ey,Et,W2);
x=[4:8:150];
y=[4:8:150];
figure(1);
hold on;
quiver(x,y,v4,u4,1,'b');
set(gca,'YDir','reverse');
set(gca,'XAxisLocation','Top');

%Eliminating outliers
[u5 v5 Exnovo Eynovo]=flow_outliers(Ex,Ey,Et);
hold on;
```

```
quiver(x,y,v5,u5,1,'r');

figure(2);
quiver(x,y,v5,u5,1,'r');
set(gca,'YDir','reverse');
set(gca,'XAxisLocation','Top');

%Calculates the optical flow using the single point algorithm
%Shows it only for the chosen block
[u1 v1]=flow(Ex,Ey,Et);
u1b=u1(1+(blocoy-1)*8:8+(blocoy-1)*8,1+(blocox-1)*8:8+(blocox-1)*8);
v1b=v1(1+(blocoy-1)*8:8+(blocoy-1)*8,1+(blocox-1)*8:8+(blocox-1)*8);
figure(3);
quiver(v1b,u1b,4,'b');
set(gca,'YDir','reverse');
set(gca,'XAxisLocation','Top');


%Uses a boxplot and a histogram to show the outliers
T=zeros(size(u1b));
c=complex(v1b,u1b);
mag=abs(c);
ang=180.*angle(c)./pi;
figure(4);
hist(mag(:));
figure(5);
boxplot(mag(:));

%Recalculates the optical flow using the single point algorithm
%Shows it only for the chosen block
[u1novo v1novo]=flow(Exnovo,Eynovo,Et);
u1novob=u1novo(1+(blocoy-1)*8:8+(blocoy-1)*8,1+(blocox-1)*8:8+(blocox-1)*8);
v1novob=v1novo(1+(blocoy-1)*8:8+(blocoy-1)*8,1+(blocox-1)*8:8+(blocox-1)*8);
figure(3);
hold on;
quiver(v1novob,u1novob,1,'r');

%Uses a boxplot and a histogram to show the new values
%(without outliers)
cnovo=complex(v1novob,u1novob);
magnovo=abs(cnovo);
angnovo=180.*angle(cnovo)./pi;
figure(6);
hist(magnovo(:));
figure(7);
boxplot(magnovo(:));

%Compares the estimated optical flow for the two algorithms
%for the chosen block
figure(8);
quiver(v4(blocoy,blocox),u4(blocoy,blocox),0,'b');
set(gca,'YDir','reverse');
set(gca,'XAxisLocation','Top');
hold on;
quiver(v5(blocoy,blocox),u5(blocoy,blocox),0,'r');
```

Apêndice 3 – Funções do MATLAB

As funções a seguir são funções já existentes no MATLAB e que foram bastante utilizadas nas *functions* e *scripts* desenvolvidos neste trabalho. Por se tratarem de funções específicas, a maioria pertencente à caixa de ferramenta estatística ou à caixa de ferramenta de Processamento de Imagens, estão listadas a seguir para facilitar o entendimento das *functions* e *scripts* listados no Apêndice 2.

A descrição das funções foi retirada da própria documentação do MATLAB, disponível na internet apenas em inglês [41].

quiver

Quiver or velocity plot

Syntax

```
quiver(U,V,U,V)
quiver(X,Y)
quiver(...,scale)
quiver(...,LineSpec)
quiver(...,LineSpec,'filled')
h = quiver(...)
```

Description

A quiver plot displays velocity vectors as arrows with components (`U`,`V`) at the points (`X`,`Y`).
For example, the first vector is defined by componets `U(1)`,`V(1)` and is displayed at the point `X(1)`,`Y(1)`.
`quiver(X,Y,U,V)` plots vectors as arrows at the coordinates specifide in each corresponding pair of elements in `X` and `Y`. The matirces `X`, `Y`, `U`, and `V` must all be the same size and contain corresponding position and velocity components.

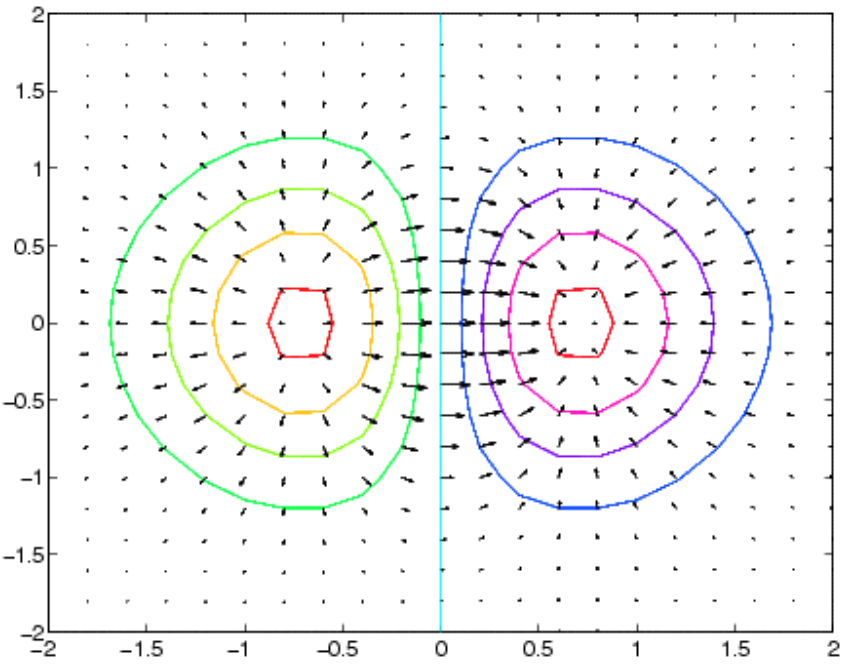
Expanding X and Y Coordinates

MATLAB expandes X and Y, if they are not matrices.
In this case, the following must be true:
`length(X) = n` and `length(Y) = m`, where `[m,n] = size(U) = size(V)`
The vector `X` corresponds to the columns of `U` and `V`, and vector `Y` corresponds to the rows of `U` and `V`.
`quiver(U,V)` draws vectors specified by `U` and `V` at equally spaced points in the x-y plane.
`quiver(...,scale)` automatically scales the arrows to fit within the grid and then stretches them by the factor `scale`.
`scale = 2` doubles their relative length and `scale = 0.5` halves the length. Use `scale = 0` to plot the velocity vectors without the automatic scaling.
`quiver(...,LineSpec)` specifies line style, marker symbol, and color. `quiver` draws the markers at the origin of the vectors.
`quiver(...,LineSpec,'filled')` fills markers specified by `LineSpec`.
`h = quiver(...)` returns a vector of line handles.

Examples

Plot the gradient field of the function.

```
[X,Y] = meshgrid(-2:.2:2);
Z = X.*exp(-X.^2 - Y.^2);
[DX,DY] = gradient(Z,.2,.2);
contour(X,Y,Z)
hold on
quiver(X,Y,DX,DY)
colormap hsv
grid off
hold off
```



boxplot

Box plots of a data sample

Syntax

```
boxplot(X)
boxplot(X,notch)
boxplot(X,notch,'sym')
boxplot(X,notch,'sym',vert)
boxplot(X,notch,'sym',vert,whis)
```

Description

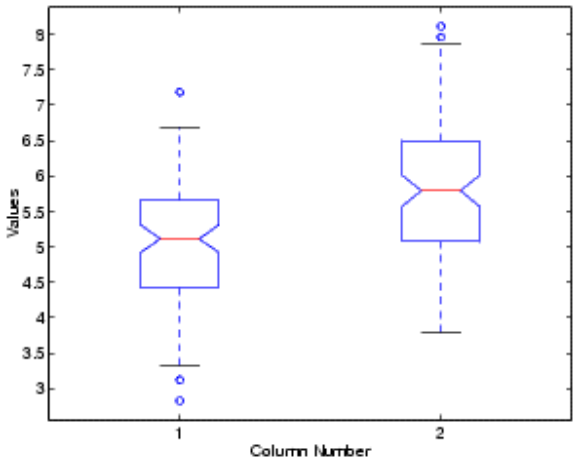
`boxplot(X)` produces a box and whisker plot for each column of `X`. The box has lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the box to show the extent of the rest of the data. Outliers are data with values beyond the ends of the whiskers. If there is no data outside the whisker, a dot is placed at the bottom whisker.
`boxplot(X,notch)` with `notch = 1` produces a notched-box plot. Notches graph a robust estimate of the uncertainty about the medians for box-to-box comparison. The default, notch `= 0`, produces a rectangular box plot.
`boxplot(X,notch,'sym')` where *sym* is a plotting symbol, affords control of the symbol for outliers. The default is `'+'`.
`boxplot(X,notch,'sym',vert)` with vert `= 0` creates horizontal boxes rather than the default vertical boxes (vert `= 1`).
`boxplot(X,notch,'sym',vert,whis)` enables you to specify the length of the "whiskers." whis defines the maximum length of the whiskers as a function of the inter-quartile range (default = 1.5). Each whisker extends to the most extreme data value within `whis` * IQR of the box. boxplot displays all data values beyond the whiskers using the plotting symbol, '*sym*'. To display all data values outside the box using '*sym*', set `whis = 0`.

Examples

This example produces box plots for the sample data, and accepts the default 1.5 * IQR for the length of the whiskers.

```
x1 = normrnd(5,1,100,1);
x2 = normrnd(6,1,100,1);
x = [x1 x2];
boxplot(x,1,'o',[],1.0)
```

The following figure shows the boxplot for the data with the length of the whiskers specified as 1.0. Points beyond the whiskers are displayed using 'o'.



blkproc

Implement distinct block processing for an image

Syntax

```
B = blkproc(A,[m n],fun)
B = blkproc(A,[m n],fun,P1,P2,...)
B = blkproc(A,[m n],[mborder nborder],fun,...)
B = blkproc(A,'indexed',...)
```

Description

`B = blkproc(A,[m n],fun)` processes the image `A` by applying the function `fun` to each distinct `m`-by-`n` block of `A`, padding `A` with zeros if necessary. `fun` is a function that accepts an `m`-by-`n` matrix, `x`, and returns a matrix, vector, or scalar `y`.

```
y = fun(x)
```

`blkproc` does not require that `y` be the same size as `x`. However, `B` is the same size as `A` only if `y` is the same size as `x`.

`B = blkproc(A,[m n],fun,P1,P2,...)` passes the additional parameters `P1,P2,...,` to `fun`.

`B = blkproc(A,[m n],[mborder nborder],fun,...)` defines an overlapping border around the blocks. `blkproc` extends the original `m`-by-`n` blocks by `mborder` on the top and bottom, and `nborder` on the left and right, resulting in blocks of size `(m+2*mborder)`-by-`(n+2*nborder)`. The `blkproc` function pads the border with zeros, if necessary, on the edges of `A`. The function `fun` should operate on the extended block.

The line below processes an image matrix as 4-by-6 blocks, each having a row border of 2 and a column border of 3. Because each 4-by-6 block has this 2-by-3 border, `fun` actually operates on blocks of size 8-by-12.

```
B = blkproc(A,[4 6],[2 3],fun,...)
```

`B = blkproc(A,'indexed',...)` processes `A` as an indexed image, padding with zeros if the class of `A` is `uint8` or `uint16`, or ones if the class of `A` is `double`.

Class Support

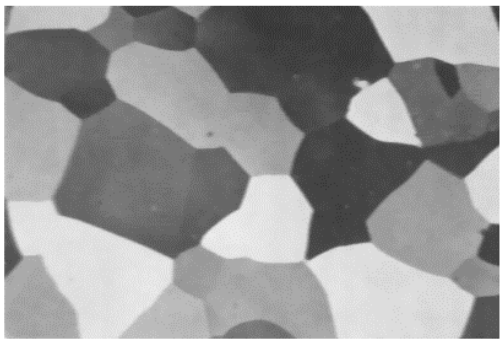
The input image `A` can be of any class supported by `fun`. The class of `B` depends on the class of the output from `fun`.

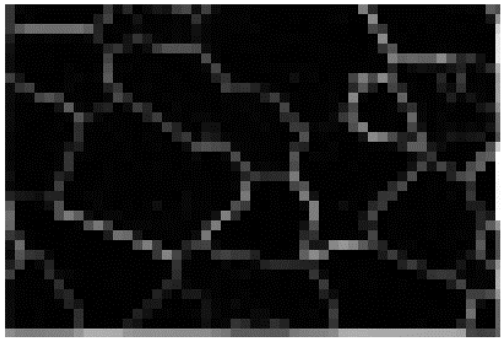
Example

```
fun can be a function_handle created using @.
fun can also be an inline object.
```

This example uses `blkproc` to set the pixels in each 8-by-8 block to the standard deviation of the elements in that block.

```
I = imread('alumgrns.tif');
fun = inline('std2(s)*ones(size(x))');
I2 = blkproc(I,[8 8],'std2(x)*ones(size(x))');
imshow(I)
figure, imshow(I2,[]);
```





hist

Histogram plot

Syntax

```
n = hist(Y)
n = hist(Y,x)
n = hist(Y,nbins)
[n,xout] = hist(...)
```

Description

A histogram shows the distribution of data values.
`n = hist(Y)` bins the elements in vector `Y` into 10 equally spaced containers and returns the number of elements in each container as a row vector. If `Y` is an `m`-by-`p` matrix, `hist` treats the columns of `Y` as vectors and returns a 10-by-`p` matrix `n`. Each column of `n` contains the results for the corresponding column of `Y`.
`n = hist(Y,x)` where `x` is a vector, returns the distribution of `Y` among `length(x)` bins with centers specified by `x`. For example, if `x` is a 5-element vector, `hist` distributes the elements of `Y` into five bins centered on the *x*-axis at the elements in `x`. Note: use `histc` if it is more natural to specify bin edges instead of centers.
`n = hist(Y,nbins)` where `nbins` is a scalar, uses `nbins` number of bins.
`[n,xout] = hist(...)` returns vectors `n` and `xout` containing the frequency counts and the bin locations. You can use `bar(xout,n)` to plot the histogram.
`hist(...)` without output arguments, `hist` produces a histogram plot of the output described above. `hist` distributes the bins along the x-axis between the minimum and maximum values of `Y`.

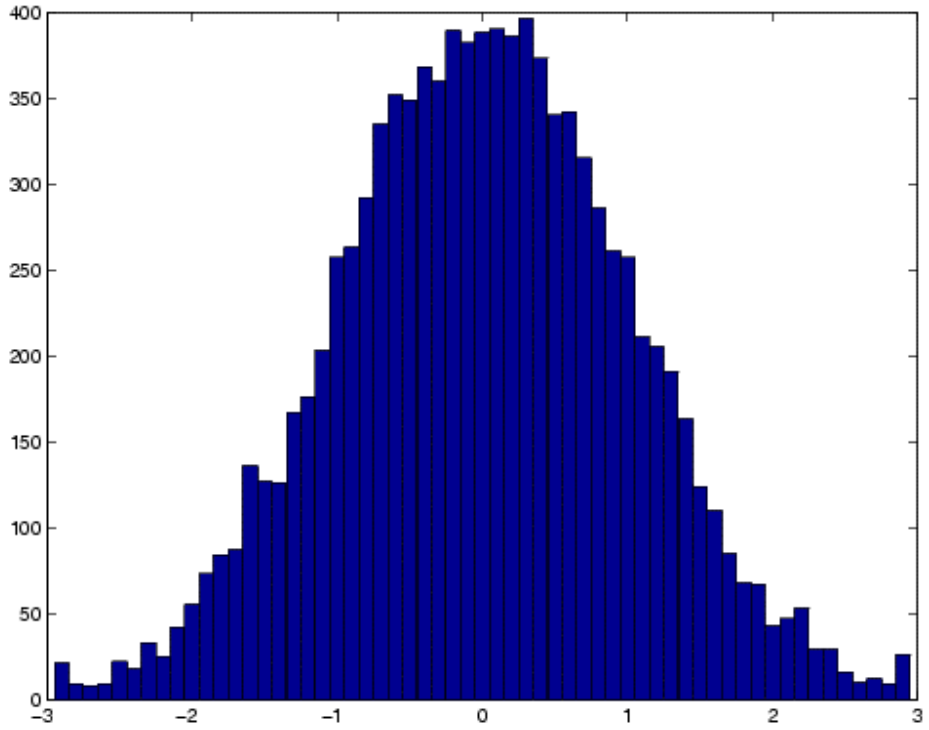
Remarks

All elements in vector `Y` or in one column of matrix `Y` are grouped according to their numeric range. Each group is shown as one bin.
The histogram's *x*-axis reflects the range of values in `Y`. The histogram's *y*-axis shows the number of elements that fall within the groups; therefore, the *y*-axis ranges from `0` to the greatest number of elements deposited in any bin.
The histogram is created with a patch graphics object. If you want to change the color of the graph, you can set patch properties. See the "Example" section for more information. By default, the graph color is controlled by the current colormap, which maps the bin color to the first color in the colormap.

Examples

Generate a bell-curve histogram from Gaussian data.

```
x = -2.9:0.1:2.9;
y = randn(10000,1);
hist(y,x)
```



set

Set object properties (for example: Axis Properties)

Syntax

set(H,'*PropertyName*',PropertyValue,...)

Remarks

You can use any combination of property name/property value pairs, structure arrays, and cell arrays in one call to set.

Examples

Set the YDir property of the current axes to reverse.

- set(gca,'YDir','reverse')

Axes Property Descriptions

Property names along with the types of values each accepts. Curly braces { } enclose default values.

XDir, **YDir**, **ZDir** {normal} | reverse

Direction of increasing values. A mode controlling the direction of increasing axis values. axes form a right-hand coordinate system. By default:

x-axis values increase from left to right. To reverse the direction of increasing *x* values, set this property to reverse.

- set(gca,'XDir','reverse')

y-axis values increase from bottom to top (2-D view) or front to back (3-D view). To reverse the direction of increasing *y* values, set this property to reverse.

- set(gca,'YDir','reverse')

z-axis values increase pointing out of the screen (2-D view) or from bottom to top (3-D view). To reverse the direction of increasing *z* values, set this property to reverse.

- set(gca,'ZDir','reverse')