Detecção Robusta de Movimento de Câmera em Vídeos por Análise de Fluxo Ótico Ponderado

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Rodrigo Minetto e aprovada pela Banca Examinadora.

Campinas, 17 de agosto de 2007.

Prof. Dr. Neucimar Jerônimo Leite
(Orientador)

Prof. Dr. Jorge Stolfi
(Co-orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Miriam Cristina Alves – CRB8a / 5098

Minetto, Rodrigo

M662d Detecção robusta de movimento de câmera em vídeos por análise de fluxo ótico ponderado / Rodrigo Minetto -- Campinas, [S.P. :s.n.], 2007.

Orientadores : Neucimar Jerônimo Leite; Jorge Stolfi

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Processamento de imagens. 2. Videodigital. 3. Reconhecimento de padrões. I. Leite, Neucimar Jerônimo. II. Stolfi, Jorge. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Título em inglês: Robust detection of camera motion by weighted optical flow analysis

Palavras-chave em inglês (Keywords): 1. Image processing. 2. Digital video. 3. Pattern recognition.

Área de concentração: Processamento de imagens

Titulação: Mestre em Ciência da Computação

Banca examinadora: Prof. Dr. Neucimar Jerônimo Leite (IC-UNICAMP)
Prof. Dr. Silvio Jamil Ferzoli Guimarães (PUC-Minas)
Prof. Dr. Alexandre Xavier Falcão (IC-UNICAMP)

Data da defesa: 17/08/2007

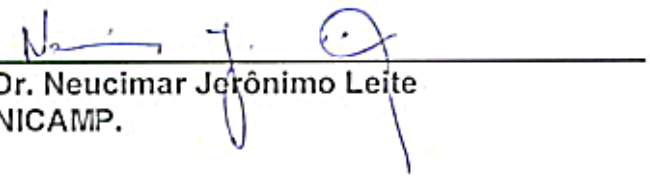
Programa de Pós-Graduação: Mestrado em Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 17 de agosto de 2007, pela Banca examinadora composta pelos Professores Doutores:

Prof. Dr. Silvio Jamil Ferzoli Guimarães
PUC - Minas Gerais.

Prof. Dr. Alexandre Xavier Falcão
IC - UNICAMP.



Prof. Dr. Neucimar Jerônimo Leite
IC - UNICAMP.

Instituto de Computação
Universidade Estadual de Campinas

Detecção Robusta de Movimento de Câmera em Vídeos por Análise de Fluxo Ótico Ponderado

Rodrigo Minetto[1]

Agosto de 2007

Banca Examinadora:

- Prof. Dr. Neucimar Jerônimo Leite
  (Orientador)
- Prof. Dr. Silvio Jamil Ferzoli Guimarães
  Pontifícia Universidade Católica de Minas Gerais
- Prof. Dr. Alexandre Xavier Falcão
  Universidade Estadual de Campinas
- Prof. Dr. Ricardo da Silva Torres (Suplente)
  Universidade Estadual de Campinas

[1]Suporte financeiro de: Bolsa do CNPq (processo 132409/2006-2) 2006–2007.

© Rodrigo Minetto, 2007.
Todos os direitos reservados.

Resumo

Nosso objetivo nesta dissertação é a detecção robusta de movimento de câmera (tilt, pan, roll e zoom) em vídeos. Para tanto, desenvolvemos um algoritmo original para esta tarefa, baseado em um ajuste ponderado de mínimos quadrados de um fluxo ótico, onde um procedimento iterativo é utilizado para melhorar o peso de cada vetor. Além da detecção de movimento de câmera, nosso algoritmo fornece uma análise quantitativa precisa e confiável dos movimentos. Este também fornece uma segmentação grosseira de cada quadro em duas regiões, "objeto" e "fundo", correspondentes às partes estacionárias e com movimento na cena, respectivamente. Experimentos com vídeos reais mostram que o algoritmo é rápido e eficaz, mesmo para cenas com movimento substancial de objetos.

Abstract

Our goal in this dissertation is the reliable detection of camera motion (tilt, pan, roll and zoom) in videos. We propose an original algorithm for this task based on weighted least-square fitting of the optical flow, where an iterative procedure is used to improve the weight of each flow vector. Besides detecting camera motion, our algorithm provides a precise and reliable quantitative analysis of the movements. It also provides a rough segmentation of each frame into two regions, "foreground" and "background", corresponding to the moving and stationary parts of the scene, respectively. Tests with real videos show that the algorithm is fast and effective, even for scenes with substantial object motion.

Agradecimentos

Eu agradeço a Deus em primeiro lugar, pela força necessária para a conclusão dessa dissertação.

Aos meus pais Jovino e Doraci, e familiares Rosane, Gildo, Angelita, Marcos, Jhony, Ariane, Fabrícia e Verediane pelo apoio, incentivo, carinho e paciência nos momentos mais difíceis.

Aos professores Jorge Stolfi e Neucimar J. Leite pela orientação, apoio e paciência durante todo o período em que estive na Unicamp, sem os quais não seria possível a realização deste mestrado.

A todos os amigos do Instituto de Computação, dos quais não citarei nomes com receio de esquecer alguém.

A Hélio Pedrini e aos amigos do PET da Universidade Federal do Paraná pelo apoio.

Agradeço também a contribuição de todos os professores e funcionários do Instituto de Computação, e todos aqueles que contribuíram para a realização deste projeto.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Símbolos

Capítulo 2

$\mathbb{V}$	Espaço de cores
$\mathbb{R}$	conjunto dos números reais
J, K	imagens
H	vídeo
t	tempo
$\mathfrak{D}$	domínio espacial de uma imagem
$\mathfrak{D}J$	Domínio de uma imagem J
$\mathfrak{D}H$	domínio de cada quadro do vídeo H
$\mathfrak{I}H$	intervalo de tempo coberto pelo vídeo H
(x, y)	coordenadas de um ponto no sistema da imagem (2D)
p, q	pontos na imagem
$f = (f_x, f_y)$	deslocamento ótico em x e em y
$v = (v_x, v_y)$	fluxo ótico diferencial
$\|f\|$	comprimento de um vetor de deslocamento ótico
dt	intervalo pequeno de tempo
$O(dt^2)$	termos de ordem ≥ 2
$ldist$	função de discrepância local
$drms$	discrepância local dada pela raiz da soma dos quadrados das diferenças
Ω	janela para discrepância local (subconjunto de $\mathbb{R}^2$)
$\Omega + p$	janela Ω deslocada pelo vetor p
δ	deslocamento entre pixels vizinhos
r	deslocamento relativo a um ponto p

Capítulo 3

∇H	gradiente espacial de H
H_t	derivada temporal de um vídeo ($\frac{\partial H}{\partial t}$)
J_x, J_y	derivada espacial da imagem J ($\frac{\partial J}{\partial x}$, $\frac{\partial J}{\partial y}$)
G	matriz hessiana
λ', λ''	autovalores
ℓ	número de escalas de resolução
J^ℓ	escala de resolução ℓ

$\mathfrak{r}$	resíduo de casamento entre correspondências
n, m	dimensões de uma imagem

Capítulo 4

X, Y, Z	coordenadas de um ponto no sistema da cena (3D)
F	distância focal
T_x^*, T_y^*, T_z^*	velocidade de translação absolutas (tracking, craning, dollying)
R_x^*, R_y^*, R_z^*	velocidades angulares absolutas (tilt, pan, roll)
R_f^*	velocidades angulares absolutas de zoom
$\dot{F}$	velocidade diferencial do zoom
$P, \dot{P}$	ponto da cena e velocidade, respectivamente
$\dot{X}, \dot{Y}, \dot{Z}$	velocidades em X, Y, Z de um ponto P
T_x, T_y, T_z	velocidade de translação aparente (tracking, craning, dollying)
R_x, R_y, R_z	velocidades angulares aparentes (tilt, pan, roll)
R_f	velocidades aparente de zoom
t_x, t_y, t_z	fluxos ideais para tracking, craning, dollying
r_x, r_y, r_z e r_f	fluxos canônicos de tilt, pan, roll e zoom, respectivamente

Capítulo 5

κ	escore de confiabilidade do KLT
ϵ	constante de ajuste de pesos
μ	média ponderada de um fluxo ótico
σ	desvio padrão de um fluxo ótico ponderado
w	pesos
d	fluxo ótico de resíduo $d = (f - \tilde{f})$
$\langle \cdot \vert \cdot \rangle$	produto escalar contínuo para dois fluxos óticos
$\langle\langle \cdot \vert \cdot \rangle\rangle$	produto discreto contínuo para dois fluxos óticos
$\vert \cdot \vert$	norma contínua de um fluxo ótico
$\Vert \cdot \Vert$	norma discreta de um fluxo ótico
$\tilde{f}$	fluxo ótico ajustado
Q	erro quadrático de d

Capítulo 6

M	câmera estática
S	câmera em movimento
V	medida de movimentação da câmera
D	magnitude de d
E_i	evento i
pr	precisão
rc	revocação
V_+	positivos verdadeiros
F_+	positivos falsos
F_-	negativos falsos

Capítulo 1

Introdução

1.1 Contexto e motivação

O volume crescente de vídeos digitais criou a necessidade de métodos eficientes para navegação, busca e visualização do seu conteúdo. Para auxiliar o usuário a localizar e recuperar informações relevantes, o passo fundamental consiste da divisão do vídeo em segmentos coerentes [1], segundo algum critério que depende da aplicação.

A segmentação automática ou semi-automática de vídeo é um problema computacional bastante estudado [1, 2, 3]. Os movimentos de câmera (figura 1.1) são freqüentemente relevantes para essa tarefa, pois revelam indiretamente mudanças no foco de interesse do operador [4].

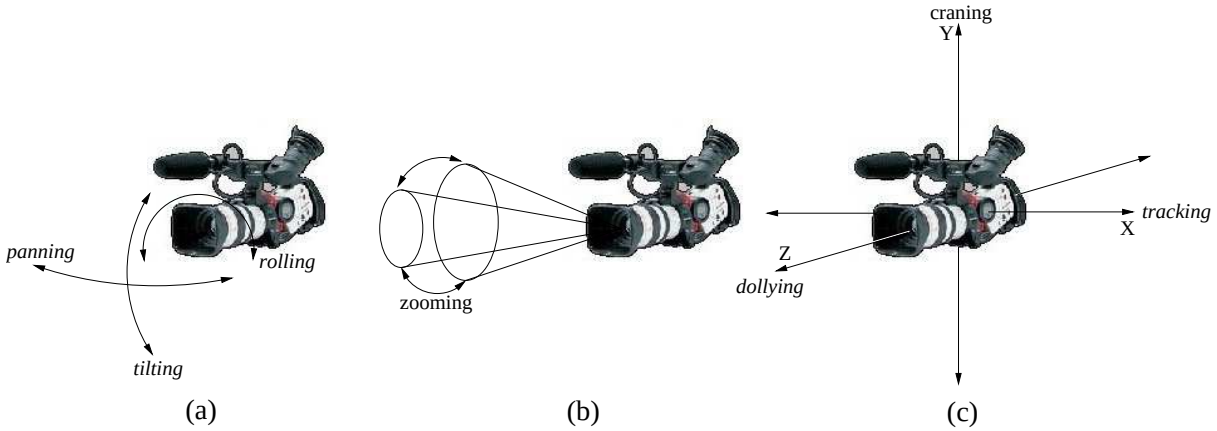


Figura 1.1: Movimentos básicos de câmera.

Para certos tipos de filmagens, tais como videoconferências, os movimentos de câmera são a principal característica para segmentação. Um exemplo disto são os registros de

vídeo das reuniões do Conselho Universitário (CONSU) da UNICAMP (figura 1.2). Tais gravações são freqüentemente feitas como uma única *tomada* [5] contínua, por uma única câmera em posição fixa, em que um operador aponta sucessivamente para os diversos oradores. Um algoritmo robusto para detectar tais movimentos de câmera permite a organização da gravação em diversos clipes de vídeo, um para cada orador — uma tarefa tediosa e demorada quando realizada manualmente. A grande dificuldade neste problema é distinguir movimentos da câmera do movimento de objetos e pessoas na cena [6], tal como o orador movimentando sua cabeça e mãos, pessoas andando no ambiente que causam a oclusão da câmera, movimento parasita (por exemplo, o balanço das cortinas do auditório devido ao vento), etc.

Figura 1.2: Quadros selecionados de uma reunião do conselho universitário da UNICAMP. Quadros A-G compreendem um segmento durante o qual a câmera está estacionária. Quadros H-W compreendem a transição para um outro orador, consistindo basicamente de um zoom-out, um pan para a direita, e um zoom-in. Quadro X é o início do próximo segmento.

1.2 Nossa contribuição

Nesta dissertação, desenvolvemos um método original [7] para detectar automaticamente movimentos de câmera em vídeos digitais. O algoritmo aqui apresentado se aplica principalmente a vídeos criados por uma câmera com posição fixa que pode ser apontada em várias direções através de movimentos de rotação (*tilt*, *pan* e *roll*), e mudanças de ampliação da lente (*zoom*), ilustrados na figura 1.1(a,b). Entretanto, como discutiremos, o

algoritmo também pode ser aplicado como uma etapa para análise de vídeos onde há movimentos de translação da câmera (*tracking*, *craning* e *dollying*), ilustrados na figura 1.1(c).

O algoritmo proposto provê uma análise precisa e confiável da quantidade de movimento existente, a saber, a quantidade de pan, tilt, zoom e roll entre dois quadros consecutivos quaisquer. Este consegue ainda, distinguir movimentos de câmera de movimentos de objetos na cena. Pela sua generalidade, acreditamos que ele poderá ser adequadamente utilizado em uma ampla gama de outras aplicações.

Para fins de segmentação de vídeos de conferência, como os do CONSU, o problema relevante é distinguir trechos em que a câmera está completamente estacionária de trechos com movimento de câmera, mesmo que de baixa velocidade. Nos testes realizados com vídeos reais, nosso algoritmo conseguiu discriminar estas duas classes com taxas de acerto de 90% em média.

1.3 Visão geral do método proposto

O método proposto consiste de dois passos principais aplicados para cada par consecutivo de quadros. O primeiro passo determina o *fluxo ótico* [8], que pode ser interpretado como a velocidade com que a cena "flui"através de cada pixel do quadro. O segundo passo analisa esse fluxo ótico para obter uma estimativa do movimento de câmera entre esses dois quadros.

Para o primeiro passo, utilizamos o algoritmo de Kanade-Lucas-Tomasi (KLT) [9], que deriva do trabalho de Lucas e Kanade [10]. Escolhemos este algoritmo porque produz fluxos de boa qualidade e, além disso, determina um escore de confiabilidade para cada vetor de fluxo, que é um dado importante para o nosso método.

Para o segundo passo, desenvolvemos um procedimento original, baseado no *método dos mínimos quadrados ponderado* [11], para decompor o fluxo ótico observado em uma combinação de quatro fluxos óticos ideais, correspondentes aos movimentos básicos de câmera (pan, tilt, zoom e roll). A diferença entre dois fluxos (resíduo) é atribuída a movimento de objetos na cena, artefatos de quantização no vídeo e falhas de rastreamento. Uma característica importante do nosso método, que chamaremos de *WOFF* (da abreviação de *weighted optical flow fitting*), é um procedimento iterativo para ajustar os pesos utilizados na análise por mínimos quadrados, com o objetivo de descartar os vetores mais discrepantes. Os pesos que resultam deste processo permitem, como sub-produto, separar a imagem em "fundo" (estático) e "objetos" (em movimento).

1.4 Organização

Esta dissertação está estruturada em oito capítulos organizados da seguinte forma: além deste Capítulo que contém uma breve introdução sobre a motivação da segmentação de vídeos por movimento de câmera, o Capítulo 2 apresenta o conceito do fluxo ótico e suas propriedades. O Capítulo 3 contém descrições de algoritmos para o cálculo do fluxo ótico. No Capítulo 4 é realizada uma análise detalhada dos modelos canônicos de fluxos óticos gerados por movimentos de translação, rotação e zoom da câmera. A descrição de alguns algoritmos encontrados na literatura para computar os parâmetros de movimento da câmera (pan, tilt, roll e zoom), e a descrição completa do algoritmo proposto estão no Capítulo 5. No Capítulo 6 são realizados experimentos com nove vídeos reais para avaliar o desempenho do algoritmo proposto. Já no Capítulo 7, são realizados experimentos para justificar os passos principais do algoritmo. Finalmente, o Capítulo 8 traz as conclusões e trabalhos futuros.

Capítulo 2

Fluxo Ótico

A maioria das técnicas existentes para determinar movimentos de câmera baseia-se na análise do fluxo ótico [12], o deslocamento aparente de partes da imagem entre quadros sucessivos do vídeo. Neste capítulo definimos mais precisamente este conceito e estudamos algumas de suas propriedades.

2.1 Imagens e vídeos

Nesta dissertação, definimos uma imagem J como sendo a função definida em um retângulo $\mathfrak{D}J$ do plano $\mathbb{R}^2$ com valores em algum espaço de cores $\mathbb{V}$. Vamos supor que o domínio $\mathfrak{D}J$ é sempre centrado na origem e que o maior vetor a partir da origem é 1.0, veja a figura 2.1.

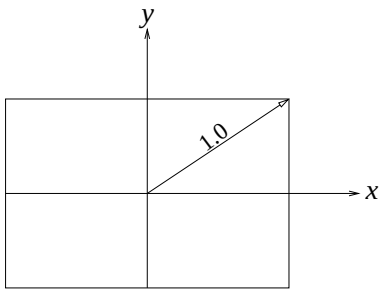


Figura 2.1: O domínio $\mathfrak{D}J$ de uma imagem.

Definimos o domínio desta maneira para simplificar as fórmulas (vide capítulo 5) e para conseguir que a magnitude dos movimentos da câmera seja independente da resolução da imagem. Na prática uma imagem J é representada por uma *imagem digital*, uma matriz de amostras (*pixels*) tomadas sobre uma grade regular que cobre o domínio $\mathfrak{D}J$; sendo

que cada amostra é arredondada (*quantizada*) para um conjunto finito de valores. Nesta dissertação, vamos supor que a amostragem tem a mesma resolução nos dois eixos.

Para obter o valor $J(x, y)$ da imagem em um ponto arbitrário, é portanto necessário realizar algum tipo de interpolação entre os valores do pixels próximos ao ponto (x, y). O método de interpolação mais apropriado depende da maneira como os pixels são amostrados, da natureza e magnitude dos erros de amostragem, e da eficiência computacional desejada. Em [13] encontra-se uma discussão sobre métodos de interpolação para imagens.

Tipicamente, o conjunto $\mathbb{V}$ é o intervalo $[0, 1]$ para imagens monocromáticas ou o cubo unitário $[0, 1] \times [0, 1] \times [0, 1]$ para imagens coloridas.

Analogamente, vamos supor que um vídeo H é uma função de um paralelepípedo $\mathfrak{D}H \times \mathfrak{I}H \subset \mathbb{R}^3$ para $\mathbb{V}$, onde $\mathfrak{D}H$ é o domínio das imagens e $\mathfrak{I}H$ é o intervalo de tempo coberto pelo vídeo. Tipicamente, um vídeo é representado por uma seqüência de *quadros* (imagens) tomadas nos instantes discretos $t_1, t_2, \ldots, t_m$. Assim, para obter o valor $H(x, y, t)$ do vídeo em um ponto arbitrário (x, y) e em um instante arbitrário t é necessário realizar algum tipo de interpolação entre os pixels próximos a (x, y) de quadros próximos a t.

2.2 Definições

2.2.1 Correspondência ótica

Dado um ponto $p \in \mathfrak{D}J$ de uma imagem J, definimos seu *correspondente ótico* em uma imagem K como sendo um ponto $q \in \mathfrak{D}K$ tal que os valores de J na vizinhança de p são maximalmente similares aos valores de K na vizinhança de q. Veja figura 2.2.

Este conceito é normalmente aplicado a duas imagens da mesma cena obtidas em momentos distintos, e/ou por câmeras em posições diferentes. Denotaremos este conceito por

$$J(p) \approx K(q). \tag{2.1}$$

O critério de similaridade usado na equação (2.1) depende do contexto, mas usualmente inclui similaridade dos valores $J(p)$ e $K(q)$ das duas imagens e possivelmente de outras propriedades nas proximidades dos pontos p e q, tais como derivadas ou textura.

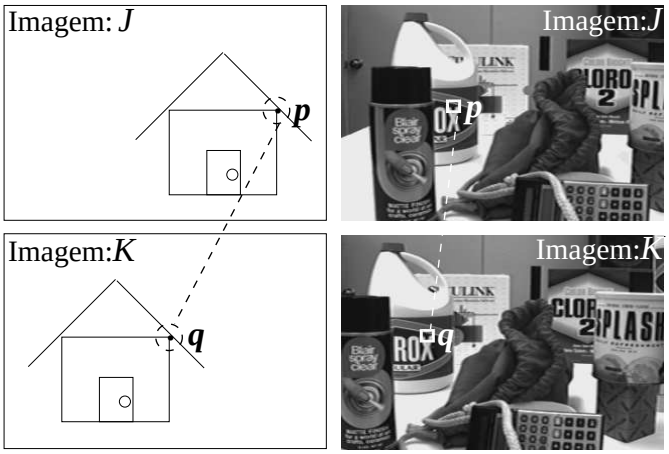


Figura 2.2: Correspondência ótica.

Critérios de (dis)similaridade local

O critério de semelhança local usado para definir a correspondência ótica é geralmente expresso por uma função de *discrepância local*, que denotaremos por $ldist(J, p, K, q)$, que mede a dissimilaridade das imagens J e K em uma certa vizinhança dos pontos p e q.

A vizinhança pode ser definida por um subconjunto Ω de $\mathbb{R}^2$. Tipicamente, Ω é um círculo ou quadrado centrado na origem que cobre alguns pixels da imagem relativo aos pontos de interesse p e q.

Uma medida de discrepância local muito usada é a *raiz da soma dos quadrados das diferenças* [14], que denotaremos por $drms$, do inglês *root of mean square difference*,

$$drms(J, p, K, q) = \sqrt{\frac{\int_\Omega [J(p+r) - K(q+r)]^2 dr}{\int_\Omega dr}}. \tag{2.2}$$

Para imagens amostradas, costuma-se usar a aproximação discreta

$$drms(J, p, K, q) = \sqrt{\frac{\sum_{r \in \Omega_\delta} [J(p+r) - K(q+r)]^2}{|\Omega_\delta|}}. \tag{2.3}$$

onde Ω_δ denota o conjunto dos vetores que pertencem à janela Ω cujas coordenadas são múltiplos de δ; e $|\Omega_\delta|$ é o número desses vetores.

Outros exemplos de funções de discrepância local que podem ser usadas na equação (2.1) são a *correlação normalizada* [14] e a *distância de entropia* [14].

2.2.2 Deslocamento ótico e fluxo ótico

O vetor $f = q - p$ é chamado de *deslocamento ótico de p entre J e K*. A função que associa a cada ponto de p de $\mathfrak{D}J$ seu deslocamento ótico $f(p)$ é chamada de *fluxo ótico*.

Ou seja, o fluxo ótico de uma imagem J para uma imagem posterior K é uma função f que, para cada ponto $p \in \mathfrak{D}J$, associa um vetor de deslocamento $f(p) \in \mathbb{R}^2$ tal que $p + f(p) \in \mathfrak{D}K$, e

$$J(p) \approx K(p + f(p)). \tag{2.4}$$

Para ilustrar um fluxo ótico f, vamos amostrá-lo em um conjunto fixo de pontos $p_1, p_2, \ldots, p_n$, produzindo uma lista de vetores $f_1, f_2, \ldots, f_n$; e representar esses dados como na figura 2.3. Cada vetor f_i é indicado por um segmento de reta com origem no ponto p_i, cuja direção é a do vetor f_i e cujo comprimento é proporcional a $|f_i|$.

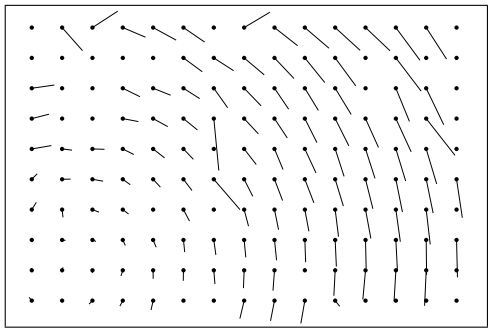


Figura 2.3: Representação de um fluxo ótico.

2.2.3 Fluxo ótico diferencial

Uma situação importante é quando as duas imagens são tão próximas (no tempo ou na posição da câmera) que os deslocamentos $f(p)$ são da ordem do passo de amostragem δ (o tamanho dos pixels). Nessa situação, podemos considerar as duas imagens J e K como sendo quadros próximos de um vídeo contínuo H; ou seja,

$$J(p) = H(p, t) \qquad \text{e} \qquad K(p) = H(p, t + dt) \tag{2.5}$$

para todo ponto $p \in \mathfrak{D}H$, algum instante t e algum intervalo pequeno de tempo dt. Para simplificar a notação, denotaremos por x e y as coordenadas de p e por f_x e f_y as componentes do deslocamento $f(p)$ entre os instantes t e $t+dt$. Portanto, a equação (2.4) pode ser re-escrita como

$$H(x, y, t) \approx H(x + f_x(x, y, t, dt), y + f_y(x, y, t, dt), t + dt). \tag{2.6}$$

Para intervalos suficientemente pequenos, o deslocamento f (se for significativo) é proporcional ao intervalo de tempo dt, isto é

$$f(x, y, t) \approx v(x, y, t)dt + O(dt^2) \tag{2.7}$$

para algum vetor $v(x, y, t)$ que independe de dt, onde $O(dt^2)$ são os termos de ordem ≥ 2. Chamamos a função v de *fluxo ótico diferencial* de H no ponto (x, y) e instante t. O fluxo ótico diferencial é dado pela fórmula

$$\begin{aligned} v = v(p,t) &= (v_x(x,y,t), v_y(x,y,t)) \\ &= \lim_{dt\to 0}\left(\frac{f(p,t,dt)}{dt}\right) = \lim_{dt\to 0}\left(\frac{f_x(x,y,t,dt)}{dt}, \frac{f_y(x,y,t,dt)}{dt}\right). \end{aligned} \tag{2.8}$$

A equação (2.6) pode então ser re-escrita como

$$H(x, y, t) \approx H(x + v_x dt, y + v_y dt, t + dt). \tag{2.9}$$

Note que o fluxo ótico é medido em unidades de distância na imagem, enquanto que o fluxo ótico diferencial é medido em unidades de distância/tempo. (Entretanto, devemos observar que muitos autores usam o termo "fluxo ótico" insdiscriminadamente para os dois conceitos.)

2.3 Aplicações do fluxo ótico

Considere o que acontece quando um objeto se movimenta entre as imagens J e K. Em condições favoráveis (que incluem iluminação uniforme, ausência de ruído, dentre outras), os valores dos pixels pertencentes ao objeto permanecem relativamente constantes durante o movimento. O mesmo ocorre quando J e K são imagens da mesma cena estática tomadas de ângulos diferentes. Portanto, o deslocamento ótico pode ser usado para determinar o movimento relativo dos objetos da cena e da câmera.

As aplicações do deslocamento ótico, que decorrem desta propriedade, são extremamente variadas: interpretação de cenas [15], segmentação de objetos em movimento [16], acompanhamento (*tracking*) de objetos [17], codificação eficiente de vídeo [18], vigilância eletrônica, etc.

Uma aplicação importante do fluxo ótico é *o estéreo geométrico* [19] — a determinação da terceira dimensão a partir de duas imagens tomadas de ângulos diferentes — que, por sua vez, é essencial para aplicações de visão robótica, como navegação exploratória [20], detecção e predição de trajetórias de objetos em movimento e avaliação de tempo para colisão [20].

2.4 Problemas da definição de fluxo ótico

2.4.1 Ambigüidades na correspondência ótica

A definição da correspondência ótica é inerentemente ambígua, pois, para cada ponto p na primeira imagem, pode haver vários pontos q na segunda imagem com similaridades locais iguais ou muito próximas.

A figura 2.4 exemplifica o problema. A cena consiste de um quadrado de lado 0.7 e cor uniforme, movendo-se diagonalmente, cujos contornos nas imagens J e K estão indicados em linha cheia e tracejada, respectivamente. O critério de similaridade local considera janelas circulares de raio 0.05, indicadas na figura com linha cheia (na imagem J) e tracejada (na imagem K). Observe-se que o deslocamento ótico está bem definido no ponto b, enquanto que é ambíguo nos pontos a, c e d. Observe-se também que nos pontos a e c, a ambigüidade é unidimensional (se estende ao longo de uma linha), enquanto que no ponto d a ambigüidade é bidimensional (sobre uma região).

Este problema é chamado na literatura de *problema da abertura* [21] em alusão de que ele decorre do fato de que a imagem está sendo examinada através de uma janela estreita (a vizinhança Ω).

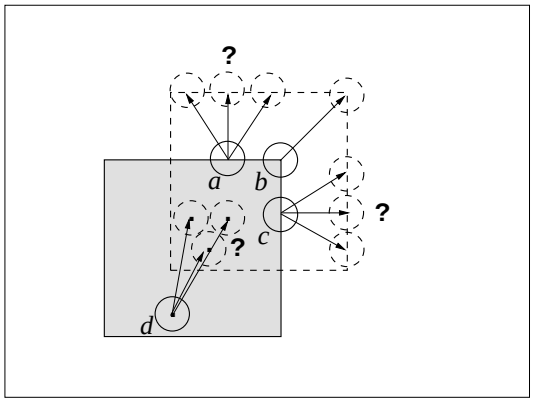


Figura 2.4: Ambigüidade do fluxo ótico.

2.4.2 Problema da oclusão

Outro problema na definição de fluxo ótico ocorre quando há *oclusão parcial* [21], isto é, quando alguma parte da cena é visível em apenas uma das imagens. Por exemplo, considere um ponto p do fundo que é visível na imagem J, mas coberto na imagem K por um objeto em movimento. Nesta situação, o correspondente ótico de p, embora

seja definido, é essencialmente inútil. Reciprocamente, um ponto q do fundo que é visível apenas na imagem K não pode ser destino de nenhum deslocamento ótico. Veja figura 2.5.

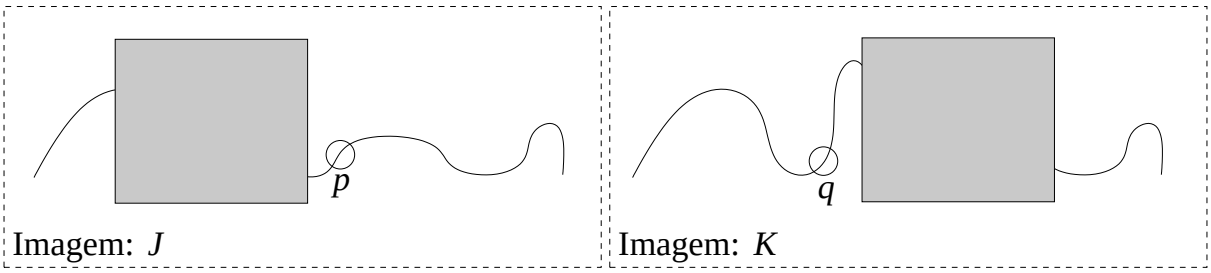


Figura 2.5: Problema da oclusão parcial. O deslocamento ótico do ponto p de J não tem sentido, e nenhum ponto de J corresponde ao ponto q de K.

2.4.3 Problema da incoerência espacial

Outro problema [21] na definição de fluxo ótico está ilustrado na figura (2.6). Os deslocamentos óticos entre as imagens J e K que melhor satisfazem a equação (2.4) são os indicados com linhas cheias na figura 2.6(c). Entretanto, uma interpretação plausível para esse par de imagens é que as cinco manchas escuras pertencem a um único objeto que está se deslocando para a direita; sendo que as diferenças entre as manchas c e n, e entre e e p são devidas a erros de amostragem, ruído, pequenas deformações do objeto, etc. Nesse caso, os casamentos mais corretos seriam os indicados pelas linhas tracejadas, figura 2.6(d).

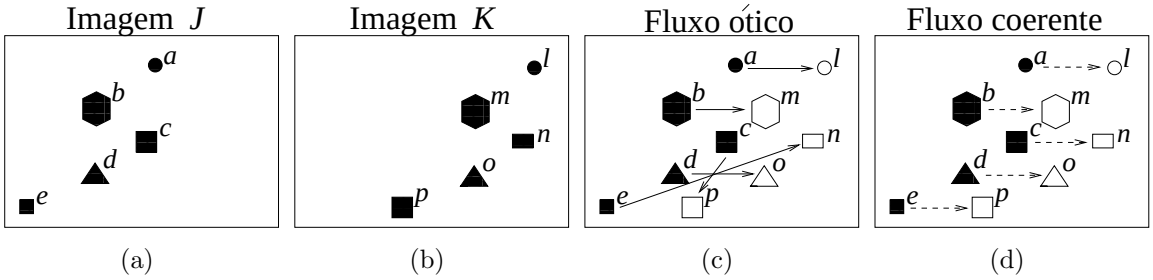


Figura 2.6: Incoerência espacial do fluxo ótico.

Capítulo 3

Cálculo do fluxo ótico

Os algoritmos para o cálculo do deslocamento ótico podem ser classificados em três categorias, segundo o tipo de problema que eles resolvem. Algoritmos para *correspondência ótica geral*, figura 3.1, possibilitam localizar o correspondente ótico q de um ponto p dado, com deslocamento $q - p$ de tamanho arbitrário, entre duas imagens J e K.

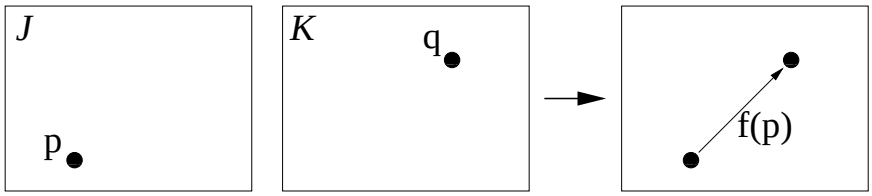


Figura 3.1: Problema de cálculo da correspondência ótica geral.

Algoritmos para cálculo do *fluxo ótico geral* basicamente resolvem o problema da correspodência ótica para um conjunto de pontos $p_1, \ldots, p_n \in \mathfrak{D}J$, possivelmente procurando satisfazer critérios de coerência entre os deslocamentos calculados.

Algoritmos para *fluxo ótico diferencial*, figura 3.2, usam análise diferencial para determinar deslocamentos pequenos entre duas imagens próximas (por exemplo, dois quadros consecutivos de um vídeo).

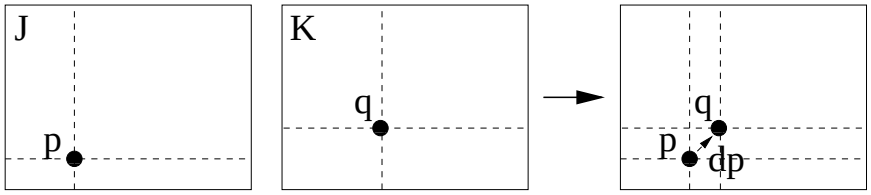


Figura 3.2: Problema de cálculo do deslocamento infinitesimal: dado p, calcular um deslocamento pequeno dp tal que $J(p) \approx K(p + dp)$).

Algoritmos de *ajuste infinitesimal* determinam pequenas correções para um deslocamento dado, com base no cálculo diferencial (figura 3.3).

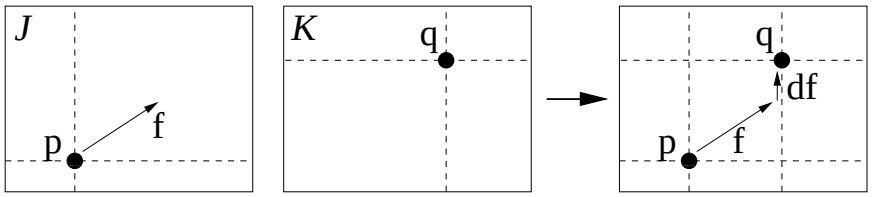


Figura 3.3: Problema de ajuste infinitesimal de deslocamento ótico: dados p e f, tal que $J(p) \approx K(p+f)$, encontrar um deslocamento pequeno df tal que $J(p) \approx K(p+f+df)$).

Finalmente, algoritmos de *ajuste local* determinam uma correção de magnitude limitada para um deslocamento dado, que garante apenas que o resultado é localmente ótimo.

O problema do ajuste infinitesimal pode ser considerado um caso particular (aplicação) do problema do fluxo ótico diferencial, pois basta substituir a imagem dada J por uma imagem J' que é obtida de J transladando-se cada ponto da mesma pelo deslocamento f dado, isto é, $J'(p) = J(p-f)$ para todo p. Muitos algoritmos para ajuste local aplicam um ajuste diferencial repetidamente, até que este seja nulo. Em outras palavras, procuram um mínimo local para a dissimilaridade pelo método de descida mais íngreme (descida pelo gradiente). Assim, algoritmos originalmente descritos para ajuste infinitesimal (como Lucas-Kanade) serão apresentados aqui em forma simplificada, como algoritmos para cálculo do fluxo ótico diferencial.

Neste capítulo examinaremos quatro algoritmos em detalhe. O método da busca exaustiva (seção 3.1) resolve o problema do deslocamento ótico geral. O algoritmo de Horn e Schunck (seção 3.2.1) determina o fluxo ótico procurando atender um critério de suavidade global. O algoritmo de Lucas e Kanade (seção 3.2.2) determina o fluxo ótico diferencial (originalmente para ajuste local de fluxo). Finalmente, o algoritmo KLT (seção 3.3) determina o fluxo ótico geral de maneira eficiente por técnicas multi-escala.

3.1 Busca exaustiva

A maneira mais simples de calcular o correspondente ótico $q \in K$ de um ponto $p \in J$ é através de uma busca exaustiva, em todo o domínio da imagem K. Veja o algoritmo 3.1.

Neste algoritmo, denotamos por $\mathfrak{D}_\varepsilon K$ uma grade discreta de pontos no domínio $\mathfrak{D}K$ com passo ε. Tipicamente ε é a distância δ entre pixels vizinhos; mas é comum usar-se

Algoritmo 3.1 - *CorrespondênciaOtica-BuscaExaustiva* $(J, p, K, \Omega, ldist, \varepsilon)$

Dados: Imagens J e K; ponto $p \in \mathfrak{D}J$; janela $\Omega \subseteq \mathbb{R}$; função de distância local $ldist$; passo $\varepsilon > 0$.
Resultado: o correspondente ótico $q \in \mathfrak{D}_\varepsilon K$ mais similar a p.

1. $dq \leftarrow +\infty$;
2. **para todo** $w \in \mathfrak{D}_\varepsilon K$ **faça**
3. $\quad dw \leftarrow ldist(J, p, K, w)$;
4. $\quad$ **se** $(dw < dq)$ **então**
5. $\quad\quad dq \leftarrow dw; \quad q \leftarrow w$;
6. **retorne** q;

$\varepsilon < \delta$ para obter o deslocamento $q - p$ com precisão subpixel.

A grande desvantagem do algoritmo 3.1 é seu alto custo computacional, pois os passos $3-5$ são executados N vezes, onde N é o número de pixels na imagem. Outra desvantagem do algoritmo 3.1 é a *falta de coerência* (seção 2.4.3) do fluxo ótico encontrado desta forma.

3.2 Cálculo do fluxo ótico diferencial

O fluxo ótico diferencial de um vídeo pode ser calculado de maneira inteiramente local, por cálculo diferencial. Como na seção 2.2, vamos considerar inicialmente o fluxo ótico entre duas imagens J e K que são dois quadros próximos do vídeo H; ou seja, $J(p) = H(p, t)$ e $K(p) = H(p, t + dt)$ para todo p e para algum t. Lembramos que o objetivo é determinar um vetor $v = (v_x, v_y)$ tal que

$$H(x, y, t) \approx H(x + v_x dt, y + v_y dt, t + dt). \tag{3.1}$$

O vetor v depende do ponto (x, y) e do instante t. Expandindo o lado direito da equação (3.1) pela série de Taylor em relação a $v_x dt$, $v_y dt$ e dt obtemos

$$H(x, y, t) \approx H(x, y, t) + v_x dt \frac{\partial H}{\partial x}(x, y, t) + v_y dt \frac{\partial H}{\partial y}(x, y, t) + dt \frac{\partial H}{\partial t}(x, y, t) + O(dt^2) \tag{3.2}$$

onde $O(dt^2)$ representa um resíduo que depende quadraticamente de dt. Desconsiderando esse resíduo, omitindo (x, y, t), cancelando o termo $H(x, y, t)$, e dividindo os demais termos por dt obtemos

$$\frac{\partial H}{\partial x} v_x + \frac{\partial H}{\partial y} v_y + \frac{\partial H}{\partial t} = 0. \tag{3.3}$$

A equação (3.3) é conhecida como *restrição do fluxo ótico* [8]. Podemos reescrevê-la como

$$v \cdot \nabla H + \frac{\partial H}{\partial t} = 0 \tag{3.4}$$

onde

$$\nabla H = \left(\frac{\partial H}{\partial x}, \frac{\partial H}{\partial y} \right) \tag{3.5}$$

é o *gradiente espacial* de H no instante t.

A equação (3.4) tem duas incógnitas (v_x, v_y), possuindo assim infinitas soluções. Portanto, considerando apenas as derivadas até primeira ordem, só é possível determinar uma componente do fluxo ótico v: especificamente, a componente que é paralela ao gradiente espacial ∇H (veja figura 3.4). A componente perpendicular a ∇H é totalmente indeterminada. Este fenômeno é a versão diferencial da ambigüidade unidimensional descrita na seção 2.4.1.

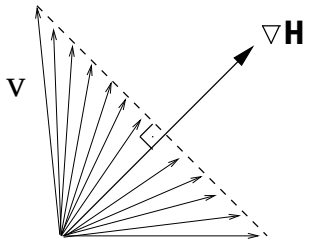


Figura 3.4: Explicação geométrica para a equação de restrição do fluxo ótico. Se algum dos vetores v indicados satisfaz a equação (3.4), então todos eles satisfazem.

Além disso, nas regiões em que ∇H é nulo, a equação (3.4) não fornece nenhuma informação sobre v. Esta é a versão infinitesimal da ambigüidade bidimensional.

Vários algoritmos foram propostos para contornar o problema da ambigüidade unidimensional [8, 10, 22, 23]. Uma análise comparativa foi feita por Barron et al [24]. Destacamos a seguir as duas soluções mais conhecidas e importantes, o algoritmo de Horn e Schunck (seção 3.2.1) e o de Lucas e Kanade (seção 3.2.2).

3.2.1 O algoritmo de Horn e Schunck (HS)

Berthold K.P. Horn e Brian G. Schunck [8] descreveram um método para o cálculo do fluxo ótico que procura contornar o problema da ambigüidade unidimensional usando a hipótese de que o fluxo ótico $v(x, y)$ varia suavemente com a posição (x, y). Mais precisamente, eles introduzem duas métricas de erro, $\mathcal{E}_F$ e $\mathcal{E}_S$. A métrica $\mathcal{E}_F$ quantifica o grau de violação

da restrição do fluxo ótico (3.4), ou seja

$$\mathcal{E}_F = \iint_{\mathfrak{D}H} \left(\nabla H \cdot v + \frac{\partial H}{\partial t} \right)^2 dxdy. \tag{3.6}$$

A métrica $\mathcal{E}_S$ quantifica a variação espacial do fluxo ótico,

$$\mathcal{E}_S = \iint_{\mathfrak{D}H} \left(\frac{\partial v_x}{\partial x} \right)^2 + \left(\frac{\partial v_x}{\partial y} \right)^2 + \left(\frac{\partial v_y}{\partial x} \right)^2 + \left(\frac{\partial v_y}{\partial y} \right)^2 dxdy. \tag{3.7}$$

Intuitivamente, esta integral mede a "energia de flexão" do gráfico de v_x e v_y em função de x e y.

O algoritmo de Horn e Schunck procura então um fluxo ótico v que minimiza uma *função energia* que é uma combinação das equações (3.6) e (3.7):

$$\mathcal{E}_T = \mathcal{E}_F + \lambda \mathcal{E}_S, \tag{3.8}$$

onde λ é um peso especificado pelo usuário. O valor ótimo do peso λ depende de fatores como erro de quantização e nível de ruído presentes no vídeo H [8].

Técnicas de cálculo variacional permitem reduzir o problema de minimizar a energia $\mathcal{E}_T$ a um sistema de equações diferenciais parciais sobre o campo v. Na implementação de Horn e Schunck, o fluxo v é calculado nos pontos de uma grade regular. Assim, as equações diferenciais podem ser aproximadas por métodos de diferenças finitas, resultando em um sistema de equações lineares com $2m$ incógnitas e $2m$ equações, onde m é o número de pontos da grade. Este sistema pode ser resolvido pelo método de relaxação [11] que é um caso particular do algoritmo iterativo de Gauss-Seidel.

A abordagem de Horn e Schunk contorna o problema da ambigüidade unidimensional porque combina informações extraídas de diferentes partes do objeto em um único fluxo coerente. Em particular, ela automaticamente preenche o fluxo nas partes da imagem onde não há informação (∇H nulo) por interpolação dos fluxos determinados na borda do objeto. A abordagem possui no entanto algumas desvantagens. Uma delas é a imposição da suavidade do fluxo ótico mesmo onde o fluxo é descontínuo, por exemplo na fronteira entre um fundo estático e um objeto em movimento [25].

3.2.2 O algoritmo de Lucas e Kanade (LK)

Bruce D. Lucas e Takeo Kanade [10, 26] propuseram outra abordagem para resolver o problema da ambigüidade unidimensional. A idéia é substituir a informação local usada pela fórmula (3.4) (gradiente ∇H e derivada temporal de H) por informação extraída dos valores de H em uma janela Ω cobrindo vários pixels.

Sejam J e K as imagens do vídeo H entre dois instantes próximos t e $t+dt$, respectivamente. Lembramos que o objetivo é calcular um vetor v para um ponto dado $p \in \mathfrak{D}J$ de tal forma

$$J(p - vdt) \approx K(p) \tag{3.9}$$

O algoritmo de Lucas e Kanade escolhe o vetor v que minimiza a soma dos quadrados dos erros entre as imagens J e K numa vizinhança $\Omega + p$ em torno do ponto p, ou seja, minimiza

$$\mathcal{D}(v) = \sum_{r \in \Omega + p} [J(r - vdt) - K(r)]^2. \tag{3.10}$$

A expansão de Taylor de primeira ordem do termo $J(r - vdt)$ na equação (3.10) em relação a dt produz

$$\mathcal{D}(v) = \iint_{\Omega+p} \left[J(r) - J_x(r)v_x dt - J_y(r)v_y dt + O(dt^2) - K(r)\right]^2 dr \tag{3.11}$$

onde $J_x = \left(\dfrac{\partial J}{\partial x}\right)$ e $J_y = \left(\dfrac{\partial J}{\partial y}\right)$. Desprezando os termos quadráticos em dt, a minimização da função $\mathcal{D}(v)$ em (3.11) é um problema de mínimos quadrados, cuja solução pode ser obtida da condição que as derivadas $\mathcal{D}(v)$ em relação a v_x e v_y sejam nulas. Isto é

$$\frac{\partial \mathcal{D}}{\partial v_x} = 0, \qquad \frac{\partial \mathcal{D}}{\partial v_y} = 0 \tag{3.12}$$

onde

$$\begin{aligned} \frac{\partial \mathcal{D}}{\partial v_x} &= -2dt \iint_{\Omega+p} (J(r) - J_x(r)v_x dt - J_y(r)v_y dt - K(r))J_x(r)dr \\ \frac{\partial \mathcal{D}}{\partial v_y} &= -2dt \iint_{\Omega+p} (J(r) - J_x(r)v_x dt - J_y(r)v_y dt - K(r))J_y(r)dr \end{aligned} \tag{3.13}$$

No que segue, omitiremos o argumento (r) de $J(r), J_x(r)$ e $J_y(r)$, para simplificar a notação. Rearranjando os termos da equação (3.13), separando o termo constante, e dividindo as equações por $2dt^2$, obtemos

$$\begin{aligned} \left(\iint_{\Omega+p} {J_x}^2 dr\right) v_x + \left(\iint_{\Omega+p} J_y J_x dr\right) v_y &= \frac{1}{dt} \iint_{\Omega+p} (J - K)J_x dr \\ \left(\iint_{\Omega+p} J_y J_x dr\right) v_x + \left(\iint_{\Omega+p} {J_y}^2 dr\right) v_y &= \frac{1}{dt} \iint_{\Omega+p} (J - K)J_y dr. \end{aligned} \tag{3.14}$$

Podemos representar o sistema da equação (3.14) em notação matricial

$$Gv = b \tag{3.15}$$

onde

$$G = \iint_{\Omega+p} \begin{bmatrix} {J_x}^2 & J_x J_y \\ J_x J_y & {J_y}^2 \end{bmatrix} dr, \quad b = \frac{1}{dt} \iint_{\Omega+p} \begin{bmatrix} (J-K)J_x \\ (J-K)J_y \end{bmatrix} dr. \tag{3.16}$$

Portanto, o fluxo ótico diferencial v pode ser obtido resolvendo-se o sistema 3.15,

$$v = G^{-1}b. \tag{3.17}$$

Esta solução é válida se e somente se a matriz G é inversível; o que é equivalente a dizer que o vídeo H possui gradiente significativo na vizinhança Ω do ponto p.

O algoritmo 3.2.2 é a versão de ajuste infinitesimal iterativo de Lucas e Kanade cujo objetivo é aplicar um ajuste local a um deslocamento dado f tal que $J(p) \approx K(p+f(p))$. Observe-se que, conforme a proposta original de Lucas e Kanade, a matriz G não é recalculada a cada iteração.

Algoritmo 3.2.2 - *AjusteLocalLucasKanade* $(J, K, J_x, J_y, p, f, \Omega, \tau, m)$

Dados: imagens J e K, derivadas J_x e J_y de J, ponto p, deslocamento ótico inicial f e janela Ω, limiar τ e número máximo de iterações m.

Resultado: deslocamento ótico ajustado g, matriz hessiana G.

---

1. iter $\leftarrow 0$;

Calcule estimativa da Hessiana de J no ponto p:

2. $G \leftarrow \sum_{r \in p+\Omega} \begin{bmatrix} (J_x(r))^2 & J_x(r)J_y(r) \\ J_x(r)J_y(r) & (J_y(r))^2 \end{bmatrix}$;

3. $g \leftarrow f$;

4. **faça** {

Calcule lado direito da equação (3.15):

5. $b \leftarrow \sum_{r \in p+\Omega} \begin{bmatrix} (J(r) - K(r+g))J_x(r) \\ (J(r) - K(r+g))J_y(r) \end{bmatrix}$;

6. **se** G é não inversível {

7. **retorne** $(0,0), G$;

8. }

Calcule o fluxo ótico diferencial por Lucas-Kanade:

9. $v \leftarrow G^{-1}b$;

Estimativa para a próxima iteração:

10. $g \leftarrow g + v$;

11. iter $\leftarrow$ iter $+1$;

12. } **enquanto** (iter $\leq m$) **ou** $(dg < \tau)$;

13. **retorne** g, G;

O passo 9 é executado iterativamente até que v seja menor que um limiar ou até que seja atingido um número m dado de iterações. A cada nova iteração o deslocamento v é acrescentado ao deslocamento calculado na iteração anterior.

A acurácia do rastreamento realizado pelo fluxo ótico de LK depende da existência de detalhes característicos salientes na vizinhança dos pontos a serem rastreados. Assim, se a vizinhança de um pixel for uniforme, ocorre uma ambigüidade bidimensional, e o algoritmo não conseguirá calcular o fluxo diferencial v pois a matriz G será nula. Da mesma forma, se a vizinhança for uniforme em uma determinada direção, ocorre ambigüidade unidimensional e a matriz G será singular.

Uma estimativa da confiabilidade do deslocamento encontrado pelo LK para o problema da correspondência no caso de movimentos puramente translacionais está justamente no cálculo da matriz inversa de G, equação (3.17). Para a matriz G ser inversível, a região do ponto deve possuir gradientes ao longo das direções x e y não triviais, assemelhando-se a um canto de um objeto/região [9].

Mais precisamente, suponha que os dois autovalores λ' e λ'' da matriz G são nulos (ou próximos a zero). Isso significa que a vizinhança do ponto p tem valor constante (ambigüidade bidimensional). Se apenas um dos autovalores λ' ou λ'' for nulo (ou próximo a zero), então o brilho da região analisada varia ao longo de uma única direção (ambigüidade unidimensional). Se λ' e λ'', forem ambos significativos o fluxo ótico no ponto p está bem definido (mas ainda pode estar incorreto, por exemplo se houver oclusão (seção 2.4.1)).

A figura (3.5) mostra seis pontos de uma imagem J, escolhidos de tal forma a ilustrar os casos descritos. Para cada ponto são calculados os autovalores λ' e λ'' da matriz G. As vizinhanças dos pontos 1 e 6 possuem características que permitem um rastreamento com confiança. Os pontos 2 e 5 possuem ambigüidade unidimensional (variação de brilho em uma direção). Finalmente, os pontos 3 e 4 possuem ambigüidade bidimensional (brilho constante).

3.3 O algoritmo de Kanade, Lucas e Tomasi (KLT)

O algoritmo de Takeo Kanade, Bruce D. Lucas e Carlos Tomasi [9, 27] (KLT) objetiva calcular deslocamentos óticos de tamanho arbitrário mas sem o custo do método da busca exaustiva.

3.3.1 Descrição do algoritmo

A idéia do KLT é resolver o problema em múltiplas escalas de resolução, começando com uma versão de baixa resolução J^ℓ da imagem dada J e passando sucessivamente a versões mais detalhadas $J^{\ell-1}, J^{\ell-2}, \ldots, J^0 = J$. A coleção $J^0, J^1, \ldots, J^\ell$ é chamada de *pirâmide*

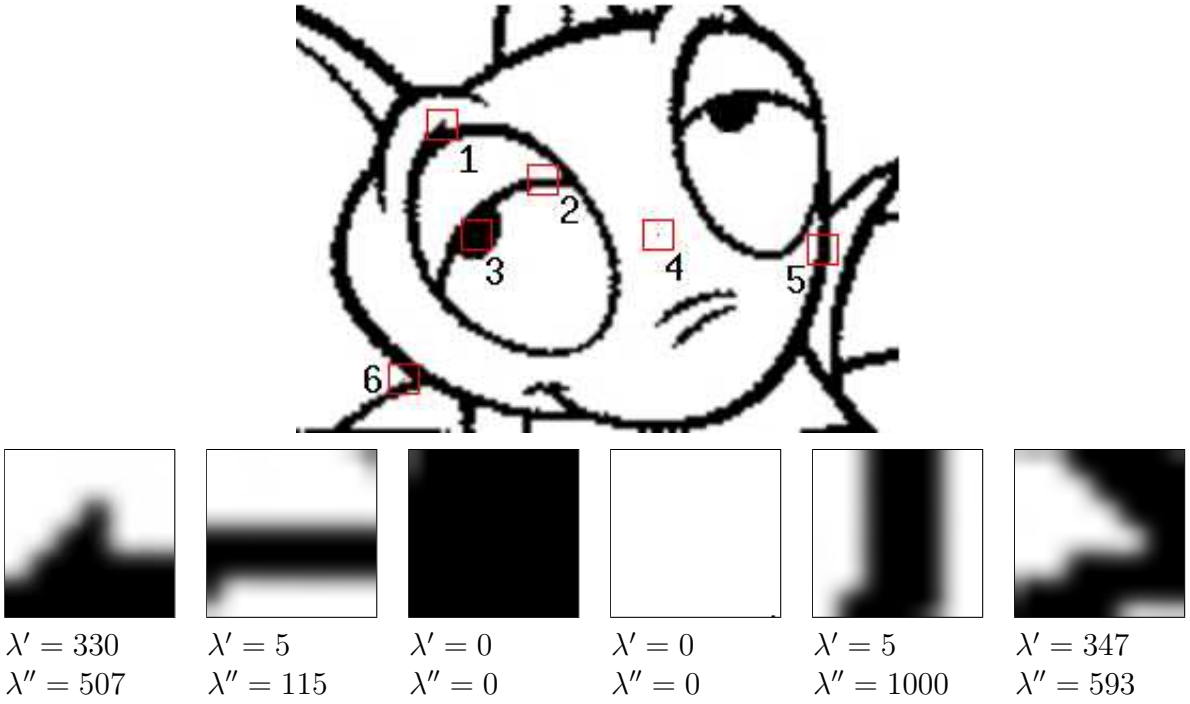


Figura 3.5: Confiabilidade de rastreamento: seis pontos selecionados de uma imagem, (no alto), suas vizinhanças (meio), e os autovalores λ' e λ'' da matriz G (embaixo).

de imagens. Veja a figura 3.6. Cada imagem J^s tem resolução $1/c$ em cada eixo, relativo

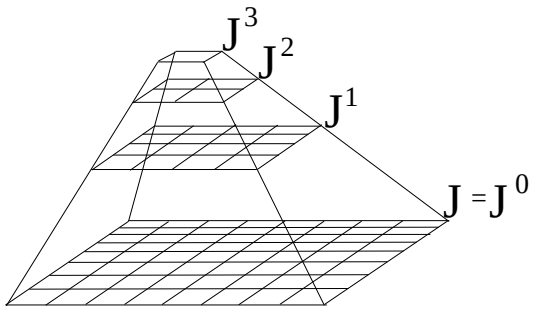


Figura 3.6: Uma pirâmide de imagens para o algoritmo KLT.

a resolução da imagem J^{s-1}; onde c é uma constante, geralmente entre dois e quatro. Portanto, a dimensão de J^s é a dimensão de J^0 dividida por c^s. Supondo que a versão mais grosseira, tem apenas um pixel, conclui-se que $\ell = \log_c m$, onde m é a maior dimensão de J^0 em pixels.

No algoritmo KLT, o deslocamento ótico encontrado em cada escala J^s é multiplicado por c, resultando em um deslocamento para a escala seguinte J^{s-1}; e este sofre um ajuste infinitesimal, levando em conta os detalhes da imagem que existem apenas em J^{s-1} mas não em J^s. Utiliza-se o algoritmo de LK (seção 3.2.2) para realizar este ajuste. A resolução

da escala inicial é escolhida de modo que os deslocamentos esperados sejam infinitesimais.

Como o trabalho em cada escala é proporcional ao número n de pontos dados, o custo total do algoritmo é $O(n\log m)$.

Tal como descrito na literatura, o algoritmo KLT recebe uma lista de pontos $p = (p_0, p_1, \ldots, p_n) \in J$ como parâmetro de entrada e devolve uma lista de pares $(p'_0,\ f(p_0))$, $(p'_1, f(p_1))$, $\ldots$, $(p'_m, f(p_k)) \in J$, que contém apenas os k melhores pontos da lista p, onde tais pontos são determinados conforme os autovalores de G (para mais detalhes veja a equação 3.17 e a discussão a respeito). Na nossa formulação (algoritmo 3.3.1), o procedimento não elimina nenhum ponto, mas devolve os parâmetros usados nesse critério de eliminação.

3.3.2 Alcance do algoritmo KLT

Supondo que a cada cálculo do fluxo ótico o ajuste máximo no deslocamento é de d_m, então o deslocamento máximo que o algoritmo pode computar é dado por

$$d_f = (c^\ell + c^{\ell-1} + \cdots + c + 1)d_m = \frac{c^{\ell+1} - 1}{c - 1} d_m. \tag{3.18}$$

O algoritmo LK pode ser aplicado apenas enquanto a janela $\Omega + q$ está totalmente dentro do domínio da imagem K, o que impede que pixels localizados nas bordas ou fronteiras da imagem tenham seu fluxo calculado, pois ao ajustá-los no centro da janela parte desta se localiza fora do domínio da imagem. Aplicando o algoritmo da pirâmide tal faixa de pixels a serem descartados aumenta, pois o tamanho da janela continua o mesmo mas as dimensões de cada imagem da pirâmide são reduzidas a um fator de 2^s a cada decomposição. A faixa de pixels rejeitada a partir de cada borda da imagem é dada pela fórmula

$$c^\ell \lfloor \zeta \rfloor \tag{3.19}$$

onde ζ é o raio da janela Ω utilizada no algoritmo de LK.

Por exemplo, ao se utilizar uma janela de 7×7 pixels, $\zeta = 3$, e $c = 2$ (redução a um fator de 2), a faixa de pixels rejeitada pelo método é de

$$2^2 3 = 12 \tag{3.20}$$

ou seja, 12 linhas e 12 colunas de pixels contadas a partir de cada borda da imagem.

Algoritmo 3.3.1 - *KLT* $(J, K, n, p, \Omega, c, \tau, m)$

Dados: imagens J e K; n pontos; vetor com n pontos $p_1, \ldots, p_n$ pertencentes a imagem J; janela Ω; c é a amostragem realizada; limiar τ e número máximo de iterações m.
Resultado: fluxo ótico $f_1, \ldots, f_n$, resíduo $\mathfrak{r}_1, \ldots, \mathfrak{r}_n$ e autovalores $\lambda'_1, \ldots, \lambda'_n$, $\lambda''_1, \ldots, \lambda''_n$ de G para cada um dos n pontos.

Construa as representações em pirâmide de J e K com $l+1$ níveis de resolução:
1. $J^0, \ldots, J^\ell$ e $K^0, \ldots, K^\ell$;
Calcule a derivada de $J^0, \ldots, J^l$ nas direções x e y:
2. $J^0_x, \ldots, J^\ell_x$ e $J^0_y, \ldots, J^\ell_y$;
3. **para** $i \leftarrow 0$ **até** n **faça** {
Inicialmente, o deslocamento ótico f_i para p_i é nulo:
4. $f_i \leftarrow 0$;
5. **para** $s \leftarrow l$ **até** 0 **faça** {
Calcule o deslocamento ótico para o próximo nível da pirâmide:
6. $f_i \leftarrow cf_i$;
Calcule a posição de p_i no nível s da pirâmide:
7. $p^s_i \leftarrow p_i/c^s$;
Calcule o ajuste infinitesimal do deslocamento:
8. $f_i, G_i \leftarrow AjusteLocalLucasKanade(J^s, K^s, J^s_x, J^s_y, p^s_i, f_i, \Omega, \tau, m)$;
9. **se** G não é inversível {
10. **então** termina a malha s;
11. }
12. }
13. **se** G não é inversível {
14. **então** $\mathfrak{r}_i = \infty$, $\lambda'_i \leftarrow \lambda''_i \leftarrow 0$
15. }
16. **senão** {
Calcule o resíduo de casamento entre o ponto p_i e v_i:
17. $$\mathfrak{r} \leftarrow \frac{\sqrt{\sum_{r \in p_i + \Omega} \left(J(r) - K(r + f_i)\right)^2}}{|\Omega|};$$
18. $\lambda'_i, \lambda''_i \leftarrow$ autovalores (G_i);
19. }
20. }
21. **retorne** $f, \mathfrak{r}, \lambda', \lambda''$;

3.3.3 Construindo a pirâmide

A maneira correta de se obter múltiplas escalas de resolução, $J^0, J^1, \ldots, J^\ell$, de uma imagem J, onde J^0 é a imagem original J e cada J^s é uma versão em escala reduzida de J^{s-1}, é aplicar um filtro passa-baixas seguido de re-amostragem na resolução desejada [28]. O objetivo da filtragem é remover as componentes de Fourier de frequência mais elevada, que seriam convertidas em componentes de frequência mais baixa pela amostragem [29] (fenômeno conhecido como "*aliasing*").

A filtragem e redução de imagens é um problema bastante estudado, e há bastantes algoritmos para esse fim [14, 30]. Estes métodos levam tempo proporcional ao número de pixels na imagem, de modo, que a redução do nível J^s para J^{s+1} tem tempo proporcional a $(m/c^s)^2$. Portanto, o tempo total para construção da pirâmide é proporcional a $m^2(1 + \frac{1}{c^2} + \frac{1}{c^4} + \cdots + \frac{1}{c^{2\ell}})$ ou seja $O(m^2)$.

3.4 Extração do fluxo ótico de vídeos MPEG

Uma alternativa aos algoritmos do HS, LK e KLT é utilizar o fluxo ótico presente em vídeos codificados no padrão MPEG [31, 32].

Neste padrão, informalmente, cada quadro do vídeo é dividido em regiões, e para cada região é calculado um vetor de deslocamento ótico em relação ao quadro anterior e/ou ao quadro posterior. Estes vetores permitem reconstruir os quadros do vídeo de maneira bem econômica. Assim, em princípio, o fluxo ótico pode ser extraído diretamente do vídeo MPEG codificado sem a necessidade de reconstruir e comparar quadros do mesmo.

Há muitos detalhes do formato MPEG que dificultam sua extração. Entre outras, os vetores só estão presentes em alguns quadros. Além disso, como os vetores de movimento do MPEG são escolhidos de forma a obter altas taxas de compressão, eles podem divergir consideravelmente do movimento real dos objetos na cena. Algumas desvantagens da utilização destes vetores são listadas em [33]. Apesar destes problemas, esta alternativa é interessante devido ao baixo custo computacional, e à popularidade do padrão MPEG.

Em todo caso, algoritmos que utilizam o fluxo ótico do MPEG geralmente necessitam de uma etapa de "filtragem" em que vetores de movimento discrepantes são removidos. Vários autores descreveram algoritmos para esse fim, como Lee e Hayes [12], Oh, Sankuratri e Tavanapong [34], Kuhn [33], Ewerth et al [35]. No entanto, mesmo com tais filtragens, o fluxo ótico extraído do MPEG pode ser bastante ruidoso.

Capítulo 4

Fluxo de movimentos de câmera

A detecção automática de movimentos da câmera em vídeos genéricos baseia-se no princípio de que estes movimentos geralmente produzem fluxos óticos característicos nas imagens, bem distintos dos fluxos resultantes de movimentos de objetos na cena. Neste capítulo estudaremos os fluxos óticos gerados por tais movimentos.

4.1 Projeção perspectiva plana

A maioria das câmeras fotográficas e de vídeo formam a imagem através da *projeção perspectiva plana* [36]. As fórmulas dessa projeção ficam bastante simplificadas quando as coordenadas de pontos da cena são dadas em relação a um *sistema canônico da câmera*. Veja a figura 4.1.

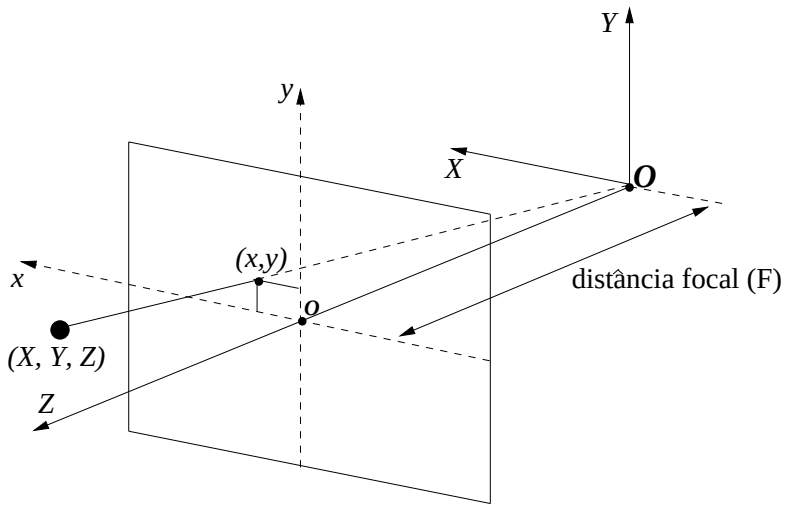


Figura 4.1: O sistema canônico para projeção perspectiva plana.

Neste sistema, a origem O é o centro da projeção (posição da câmera); o eixo Z é o eixo da câmera, perpendicular ao plano da imagem e orientado para a frente; o eixo X é paralelo ao eixo horizontal da imagem, orientado para a direita; e o eixo Y aponta para o alto da imagem. A distância entre o plano da imagem e a origem O é chamada de *distância focal* e denotada por F.

Os pontos projetados são descritos em relação ao *sistema canônico da imagem*, cuja origem o é a projeção ortogonal do ponto O no plano da imagem, e cujos eixos x, y são paralelos aos eixos X, Y da câmera. Vamos supor as mesmas unidades de medidas nos dois sistemas.

Com estas convenções, verifica-se que a projeção (x, y) de um ponto da cena com coordenadas (X, Y, Z) é dada pelas fórmulas

$$x = F\frac{X}{Z} \qquad y = F\frac{Y}{Z}. \tag{4.1}$$

4.2 Movimento da câmera

Como o sistema canônico (O, X, Y, Z) está fixo na câmera, um movimento da câmera com uma cena estacionária é equivalente a um movimento rígido da cena com a câmera estacionária. Nesta seção, vamos analisar esta equivalência e determinar o seu efeito na imagem projetada.

4.2.1 Taxas de movimento de câmera

Instantâneamente, todo movimento de câmera é uma combinação de uma translação, com certa velocidade linear, e uma rotação, com certa velocidade angular.

A translação instântanea é determinada por três parâmetros T_x^*, T_y^* e T_z^*, as componentes da velocidade linear da câmera (em unidades de distância por unidade de tempo), medidas no sistema da câmera. A rotação instantânea pode ser decomposta na soma de três rotações instantâneas da câmera em torno dos eixos X, Y e Z, cujas velocidades angulares denotaremos por R_x^*, R_y^* e R_z^* (em radianos por unidade de tempo). Além disso, a distância focal F pode estar variando; denotaremos por R_f^* a "velocidade de zoom" da câmera, ou seja a derivada de F em relação ao tempo $R_f^* = \dfrac{dF}{dt}$. Veja a figura 4.2.

Por convenção, as velocidades R_x^*, R_y^* e R_z^* são positivas quando a rotação é no sentido dado pela regra da mão esquerda (indicado na figura 4.2). Obviamente, um valor negativo

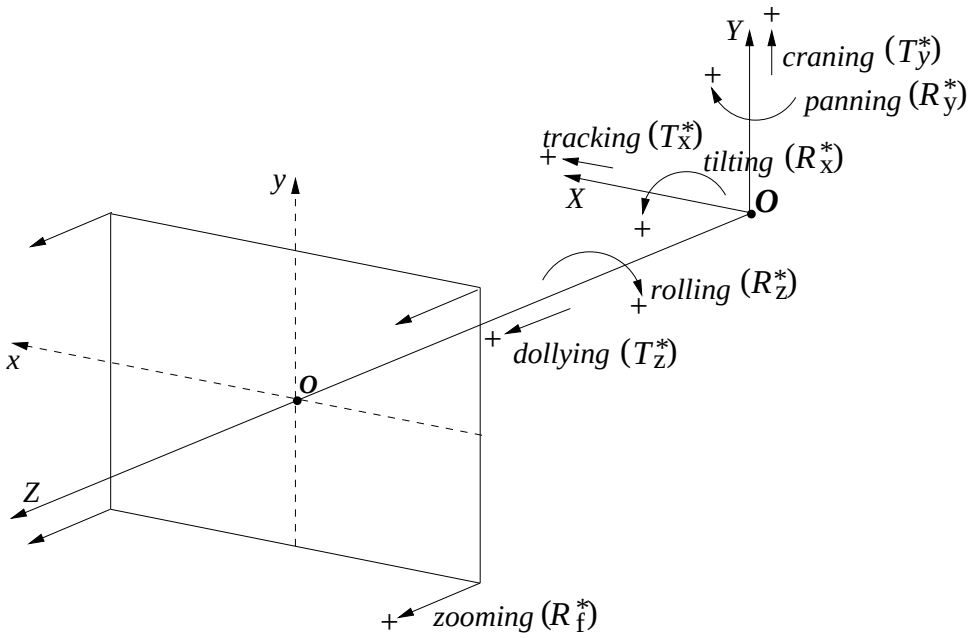


Figura 4.2: Velocidades que definem o movimento instantâneo da câmera.

Vel.	Nome	Sentido Positivo
T_x^*	tracking	p/ direita
T_y^*	craning	p/ cima
T_z^*	dollying	p/ frente
R_x^*	tilting	p/ baixo
R_y^*	panning	p/ direita
R_z^*	rolling	anti-horário
R_f^*	zooming	ampliar

Tabela 4.1: Nomes tradicionais e sentido positivo das velocidades instantâneas da câmera.

para uma velocidade significa movimento no sentido oposto. Na produção de cinema e/ou televisão, estas componentes têm nomes tradicionais listados na tabela 4.1.

Movimento na cena e na imagem

O movimento de um ponto da cena ao longo do tempo pode ser descrito por uma função $P(t) = (X(t), Y(t), Z(t))$ do tempo t. Seja $p(t) = (x(t), y(t))$ a projeção na imagem do ponto $P(t) = (X(t), Y(t), Z(t))$ da cena. Pela equação (4.1), a posição $p(t)$ é

$$p(t) = F(t)\left(\frac{X(t)}{Z(t)}, \frac{Y(t)}{Z(t)}\right). \tag{4.2}$$

ou seja,

$$x(t) = F(t)\frac{X(t)}{Z(t)}, \quad y(t) = F(t)\frac{Y(t)}{Z(t)}. \tag{4.3}$$

O deslocamento desse ponto em um intervalo infinitesimal dt pode ser descrito pela fórmula

$$P(t+dt) = P(t) + \frac{\partial P}{\partial t}(t)dt \tag{4.4}$$

onde $\frac{\partial P}{\partial t} = \left(\frac{\partial X}{\partial t}, \frac{\partial Y}{\partial t}, \frac{\partial Z}{\partial t}\right)$ é a velocidade do ponto no instante t, ambos no sistema da câmera. Para simplificar a notação, omitiremos o argumento t (tempo) nas fórmulas seguintes, e usaremos um ponto superior para indicar derivadas em relação ao termo t, $\dot{V} = \frac{\partial V}{\partial t}$.

O fluxo ótico esperado é a derivada de p em relação ao tempo, ou seja,

$$\begin{aligned} v_x &= \frac{dx}{dt} = F\left(\frac{\dot{X}}{Z} - \frac{X}{Z}\frac{\dot{Z}}{Z}\right) + \dot{F}\frac{X}{Z} \\ v_y &= \frac{dy}{dt} = F\left(\frac{\dot{Y}}{Z} - \frac{Y}{Z}\frac{\dot{Z}}{Z}\right) + \dot{F}\frac{Y}{Z}. \end{aligned} \tag{4.5}$$

4.2.2 Velocidade da cena devida a movimento de câmera

Conforme explicado por Longuet-Higgins e Prazdny [37], as velocidades $\dot{X}, \dot{Y}$ e $\dot{Z}$ do ponto, geradas pelos movimentos da câmera da figura 4.2 são dadas pela fórmulas

$$\begin{aligned} \dot{X} &= -T_x^* \quad\quad + R_z^* Y - R_y^* Z \\ \dot{Y} &= -T_y^* - R_z^* X \quad\quad + R_x^* Z \\ \dot{Z} &= -T_z^* + R_y^* X - R_x^* Y \end{aligned} \tag{4.6}$$

ou, em notação matricial,

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} -T_x^* \\ -T_y^* \\ -T_z^* \end{bmatrix} + \begin{bmatrix} 0 & R_z^* & -R_y^* \\ -R_z^* & 0 & R_x^* \\ R_y^* & -R_x^* & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \tag{4.7}$$

Substituindo as fórmulas de $(\dot{X}, \dot{Y}, \dot{Z})$, conforme a equação (4.6), em (4.5) obtemos a velocidade aparente v do ponto na imagem:

$$\begin{aligned} v_x &= F\left(\frac{-T_x^* + R_z^* Y - R_y^* Z}{Z} - \frac{X}{Z}\left(\frac{-T_z^* + R_y^* X - R_x^* Y}{Z}\right)\right) + \dot{F}\frac{X}{Z} \\ v_y &= F\left(\frac{-T_y^* - R_z^* X + R_x^* Z}{Z} - \frac{Y}{Z}\left(\frac{-T_z^* + R_y^* X - R_x^* Y}{Z}\right)\right) + \dot{F}\frac{Y}{Z}, \end{aligned} \tag{4.8}$$

ou seja

$$\begin{aligned} v_x &= -F\frac{1}{Z}T_x^* + F\frac{Y}{Z}R_z^* - F\frac{Z}{Z}R_y^* - F\frac{X}{Z}\left(-\frac{1}{Z}T_z^* + \frac{X}{Z}R_y^* - \frac{Y}{Z}R_x^*\right) + \dot{F}\frac{X}{Z} \\ v_y &= -F\frac{1}{Z}T_y^* - F\frac{X}{Z}R_z^* + F\frac{Z}{Z}R_x^* - F\frac{Y}{Z}\left(-\frac{1}{Z}T_z^* + \frac{X}{Z}R_y^* - \frac{Y}{Z}R_x^*\right) + \dot{F}\frac{Y}{Z}. \end{aligned} \tag{4.9}$$

Observando que $x = F\frac{X}{Z}$, $y = F\frac{Y}{Z}$ e $R_f^* = \dot{F}$, então

$$\begin{aligned} v_x &= -F\left(\frac{1}{Z}T_x^*\right) + R_z^* y - R_y^* F - x\left(-\frac{1}{Z}T_z^* + \frac{x}{F}R_y^* - \frac{y}{F}R_x^*\right) + R_f^*\left(\frac{x}{F}\right) \\ v_y &= -F\left(\frac{1}{Z}T_y^*\right) - R_z^* x + R_x^* F - y\left(-\frac{1}{Z}T_z^* + \frac{x}{F}R_y^* - \frac{y}{F}R_x^*\right) + R_f^*\left(\frac{y}{F}\right), \end{aligned} \tag{4.10}$$

e, finalmente,

$$\begin{aligned} v_x &= \frac{x}{Z}T_z^* - \frac{F}{Z}T_x^* + \frac{xy}{F}R_x^* - F\left(1 + \frac{x^2}{F^2}\right)R_y^* + yR_z^* + \left(\frac{x}{F}\right)R_f^* \\ v_y &= \frac{y}{Z}T_z^* - \frac{F}{Z}T_y^* - \frac{xy}{F}R_y^* + F\left(1 + \frac{y^2}{F^2}\right)R_x^* - xR_z^* + \left(\frac{y}{F}\right)R_f^*. \end{aligned} \tag{4.11}$$

Observe que, nestas fórmulas, v_x e v_y são funções lineares das velocidades T_x^*, T_y^*, T_z^*, R_x^*, R_y^*, R_z^* e R_f^*. Na figura 4.3 mostramos um exemplo de fluxo ótico definido pela equação (4.11).

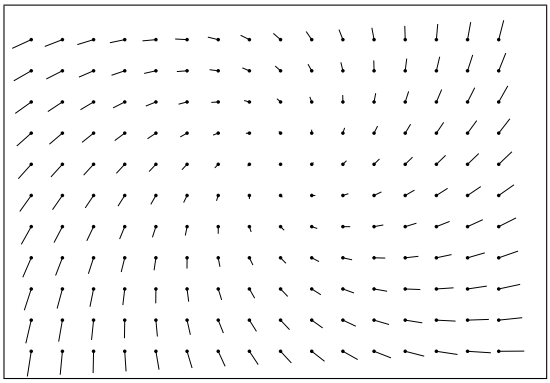


Figura 4.3: Fluxo ótico previsto para movimento de câmera com $T_x^* = T_z^* = 0$, $T_y^* = Z = 1$, $R_y^* = -1$, $R_x^* = R_z^* = -2$, $R_f^* = 2$ e $F = 7$.

4.2.3 Velocidades aparentes da câmera

Como a distância focal F é muitas vezes desconhecida, é interessante definir as *velocidades aparentes da câmera* como

$$\begin{aligned} T_x = T_x^* F, \quad T_y = T_y^* F, \quad T_z = T_z^*, \\ R_x = R_x^* F, \quad R_y = R_y^* F, \quad R_z = R_z^*, \quad R_f = \frac{R_f^*}{F}. \end{aligned} \tag{4.12}$$

Desta forma as equações (4.11) ficam

$$\begin{aligned} v_x &= \frac{x}{Z}T_z - \frac{T_x}{Z} + \frac{xy}{F^2}R_x - \left(1 + \frac{x^2}{F^2}\right)R_y + yR_z + xR_f \\ v_y &= \frac{y}{Z}T_z - \frac{T_y}{Z} - \frac{xy}{F^2}R_y + \left(1 + \frac{y^2}{F^2}\right)R_x - xR_z + yR_f. \end{aligned} \tag{4.13}$$

Utilizaremos daqui em diante as velocidades aparentes, e portanto a equação (4.13) para caracterizar o movimento da câmera.

4.2.4 Fluxo para câmera em posição fixa

Um caso particular importante é quando a posição da câmera é fixa, isto é, $T_x^* = T_y^* = T_z^* = 0$. Neste caso, as fórmulas (4.13) reduzem-se a

$$\begin{aligned} v_x &= \frac{xy}{F^2}R_x - \left(1 + \frac{x^2}{F^2}\right)R_y + yR_z + xR_f \\ v_y &= -\frac{xy}{F^2}R_y + \left(1 + \frac{y^2}{F^2}\right)R_x - xR_z + yR_f. \end{aligned} \tag{4.14}$$

4.2.5 Fluxo para campo estreito

A análise anterior é exata para qualquer geometria. Entretanto, quando a distância focal F é grande comparada com o raio do domínio da imagem, ou seja, quando ($F \gg 1$), vários termos da fórmula (4.13) se tornam insignificantes em comparação com erros devidos a outras fontes, e podem ser desprezados. Nesse caso, a fórmula (4.13) se torna

$$\begin{aligned} v_x &= \frac{x}{Z}T_z - \frac{T_x}{Z} - R_y + yR_z + xR_f \\ v_y &= \frac{y}{Z}T_z - \frac{T_y}{Z} + R_x - xR_z + yR_f. \end{aligned} \tag{4.15}$$

Estas fórmulas são as *aproximações de campo estreito* [38] para o fluxo de movimento de câmera.

Mesmo quando a distância focal F é relativamente pequena, a aproximação de campo estreito ainda pode ser adequada para a análise. É o caso do passo de ajuste de mínimos quadrados, do nosso método (seção 5.2.2). A aproximação (4.15) também é útil nos casos em que o valor de F é desconhecido (que é o caso dos vídeos utilizados nesta dissertação).

4.3 Fluxos canônicos para rotação e zoom

Os *fluxos canônicos de movimento de câmera* são os quatro fluxos óticos previstos pela equação (4.13), quando uma cena estática é filmada com uma câmera em posição fixa realizando os movimentos de tilt, pan, roll e zoom, respectivamente, com velocidade unitária. Denotaremos estes fluxos por r_x, r_y, r_z e r_f, respectivamente.

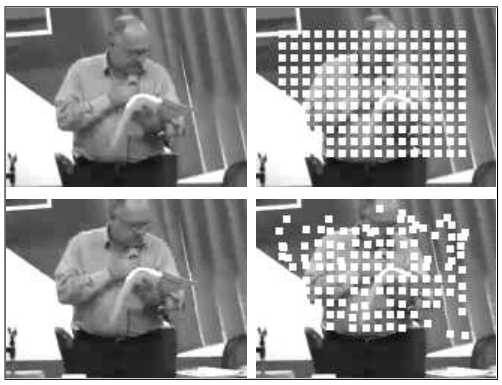
Estes fluxos são descritos nas seções 4.3.1— 4.3.4. Nas figuras destas seções apresentaremos: (a) dois quadros consecutivos de um vídeo durante o movimento de câmera em questão; (b) o fluxo ótico correspondente, calculado pelo KLT; (c) o fluxo previsto pela equação (4.13) com $F = 8$; (d) e o fluxo de campo estreito previsto pela equação (4.15).

4.3.1 Fluxo canônico devido a Tilt

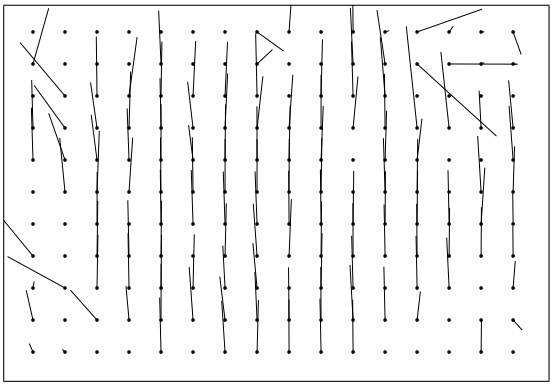
As componentes de velocidade (v_x, v_y) de cada ponto (x, y) do fluxo ótico canônico referente unicamente ao tilt, são

$$r_x(x, y) = \left(\frac{xy}{F^2}, \left(1 + \frac{y^2}{F^2}\right)\right). \tag{4.16}$$

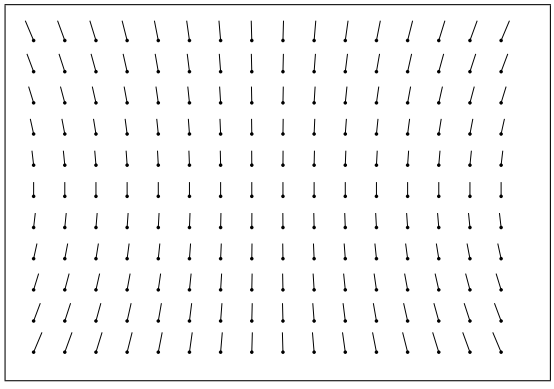
Para campo estreito os termos $\frac{xy}{F}$ e $\frac{y^2}{F}$ são desprezíveis, portanto a fórmula reduz-se a $r_x(x, y) = (0, 1)$. Veja a figura 4.4.



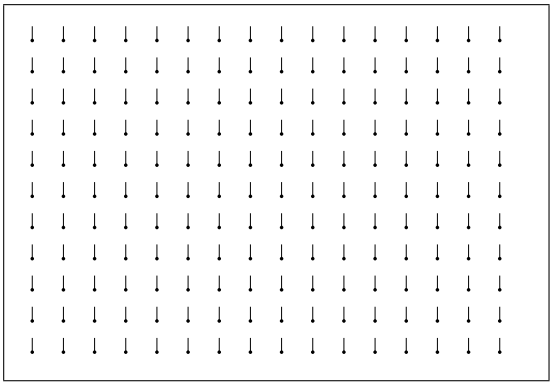
(a) quadros e pontos correspondentes



(b) fluxo KLT



(c) fluxo canônico r_x



(d) aproximação de campo estreito

Figura 4.4: Fluxos óticos devidos a movimento de tilt.

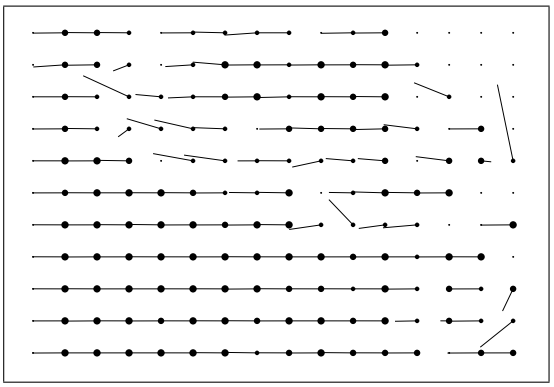
4.3.2 Fluxo canônico devido a Pan

O fluxo canônico de pan é

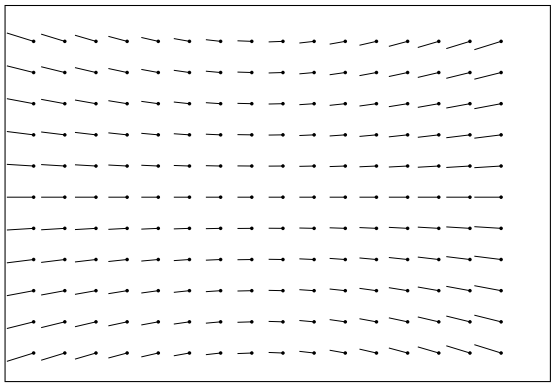
$$r_y(x,y) = \left(-\left(1 + \frac{x^2}{F^2}\right), -\frac{xy}{F^2} \right). \tag{4.17}$$

A aproximação de campo estreito é $r_y(x,y) = (-1, 0)$. Veja figura 4.5.

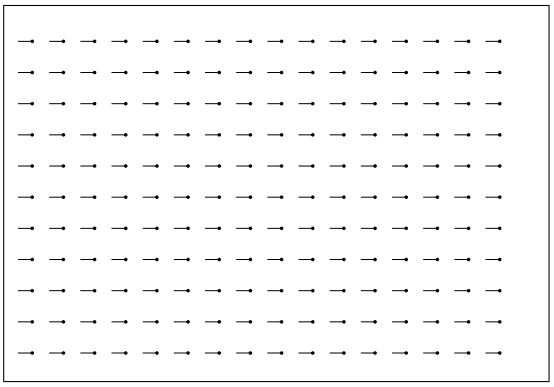
(a) quadros e pontos correspondentes



(b) fluxo KLT



(c) fluxo canônico r_y



(d) aproximação de campo estreito

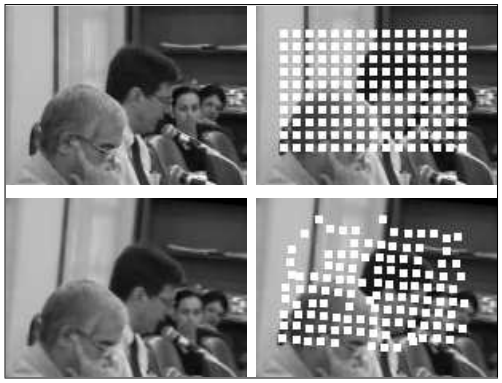
Figura 4.5: Fluxos óticos devidos a movimento de pan.

4.3.3 Fluxo canônico devido a Roll

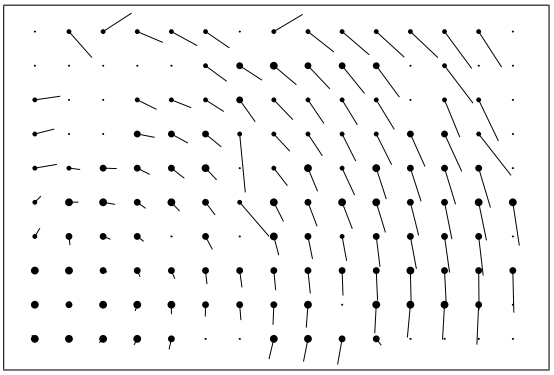
O fluxo ótico canônico de roll é

$$r_z = (y, -x). \tag{4.18}$$

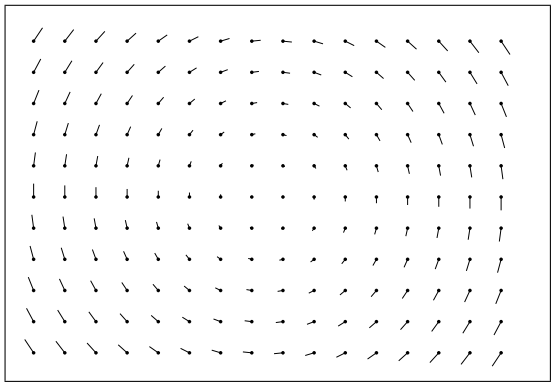
Observe que esta fórmula não depende da distância focal F e portanto se aplica também na aproximação de campo estreito. Veja figura 4.6.



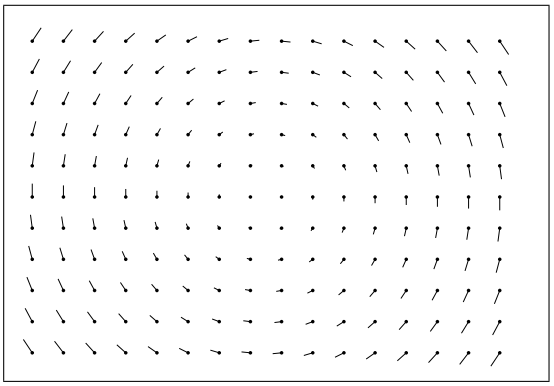
(a) quadros e pontos correspondentes



(b) fluxo KLT



(c) fluxo canônico r_z



(d) aproximação de campo estreito

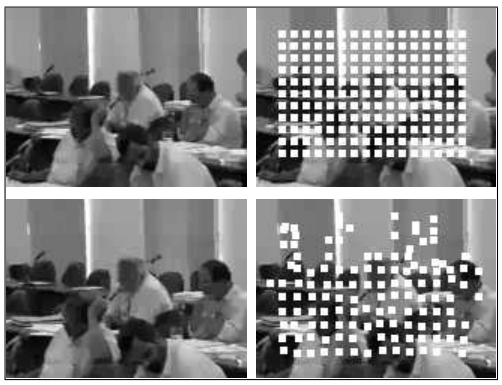
Figura 4.6: Fluxos óticos devidos a movimento de roll.

4.3.4 Fluxo canônico devido a Zoom

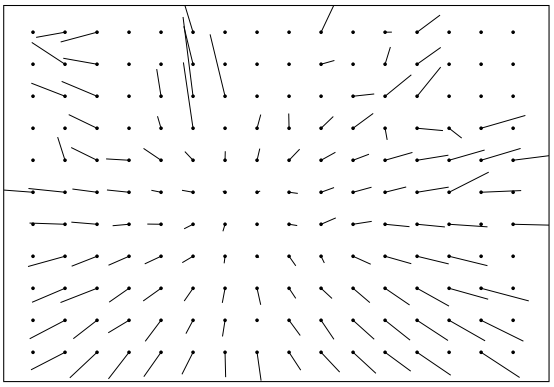
O fluxo canônico de zoom é

$$r_f = (x, y). \tag{4.19}$$

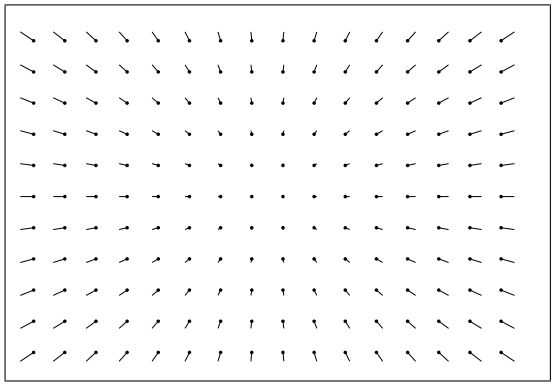
A mesma fórmula se aplica na aproximação de campo estreito. Veja figura 4.7.



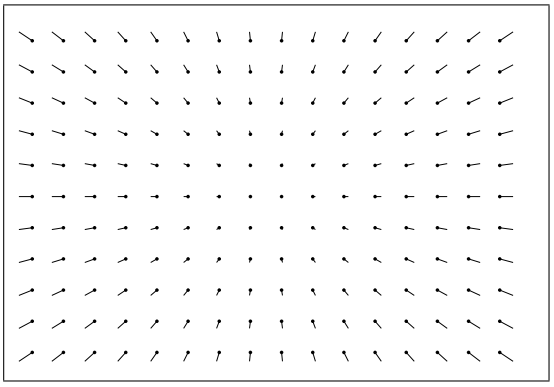
(a) quadros e pontos correspondentes



(b) fluxo KLT



(c) fluxo canônico r_f



(d) aproximação de campo estreito

Figura 4.7: Fluxos óticos devidos a movimento de zoom.

4.4 Fluxos canônicos para translação

O fluxo ideal devido a translação depende da profundidade Z de cada ponto da cena. Portanto os fluxos devidos a cada componente de velocidade (T_x, T_y, T_z) não são únicos. Como o algoritmo proposto não detecta ainda tais movimentos estes não são detalhados nesta dissertação.

Capítulo 5

Análise do Fluxo Ótico

O objetivo da análise é extrair do fluxo ótico os parâmetros de movimento da câmera — a velocidade do tilt, pan, roll e zoom ao longo do vídeo. Tipicamemte, procura-se determinar os parâmetros de movimento que proporcionam o melhor ajuste entre o fluxo observado e o fluxo predito. A qualidade do ajuste pode ser utilizada como uma medida da confiabilidade dos valores obtidos para esses parâmetros. Neste capítulo apresentamos três soluções encontradas na literatura (seção 5.1), e nossa solução original (seção 5.2). Entretanto, deve-se observar que é impossível a distinção entre pan e tracking horizontal ou tilt e tracking vertical, de maneira única, quando o algoritmo é aplicado em vídeos com campo de visão estreito [38]. Isto se deve ao fato de vetores de fluxo devidos a pan ou tilt no cenário especificado, serem perfeitamente paralelos tal como são os vetores devidos a tracking (veja seção 4.3).

5.1 Trabalhos relacionados

Algumas abordagens interessantes para a análise do fluxo ótico são as de Srinivasan et al (1997) [38], Lee et al (2002) [12] e Oh et al (2003) [34]. Apesar de mais recentes, os dois últimos são mais simples e limitados que o primeiro e, por este motivo, serão descritos antes dele.

5.1.1 Algoritmo de Lee e Hayes (LH)

Em 2002, Sangkeun Lee e Monson Hayes [12] propuseram um método que considera apenas seis movimentos simples (pan, tilt e zoom nos dois sentidos) excluindo combinações simultâneas como tilt e pan.

O método consiste em comparar o fluxo ótico observado com os seis fluxos-modelo (pan para esquerda e direita, tilt para baixo e para cima e zoom-in e zoom-out) descritos no capítulo 4 (para campo de visão estreito). Antes da comparação, os dois fluxos são reduzidos a vetores unitários com 16 direções possíveis $e_1, \ldots, e_{16}$. Pontos onde a magnitude do fluxo é menor que um limiar fixo são excluídos da comparação. A métrica utilizada na comparação é

$$d(h, g_j) = \sum_{k=1}^{16} |h[k] - g_j[k]| \tag{5.1}$$

onde $h[k]$ é o número de pontos de amostragem na imagem cujo fluxo normalizado tem direção e_k, e $g_j[k]$ é a contagem análoga para o fluxo padrão j. O algoritmo atribui o movimento j ao par de quadros para o qual $d(h, g_j)$ é mínimo. A magnitude do movimento é calculada a partir da média das magnitudes dos vetores do fluxo observado.

Lee e Hayes propuseram este método para analisar especificamente fluxos óticos extraídos de vídeos MPEG. Para reduzir o ruído desse fluxo, eles aglomeram os macroblocos do MPEG em regiões quadradas de mesmo tamanho e sem intersecção. Para cada região é calculado um vetor de fluxo, que é a média dos vetores de deslocamento dos macroblocos que a compõem.

A principal desvantagem deste método é que ele não considera combinações de movimento. Outras desvantagens são que ele reduz para $(0, 0)$ os vetores de fluxo normalizados em áreas da região onde estes são pequenos e ignora módulo dos vetores e, portanto, despreza informação presente no fluxo observado.

5.1.2 Algoritmo de Oh, Sankuratri e Tavanapong (OST)

Outro algoritmo simples mas limitado foi proposto por JungHwan Oh, Praveen Sankuratri e Wallapak Tavanapong [34] em 2003. O método usa apenas três vetores v_1, v_2 e v_3 do fluxo ótico, amostrados em três pontos apropriados p_1, p_2 e p_3.

Eles utilizaram a observação de Bouthemy et al [6], de que o fluxo ótico ideal $v = (v_x, v_y)$ gerado por movimento de câmera, descrito pela equação (4.11) pode ser representado como uma função afim da posição $p = (x, y)$:

$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} a_2 & a_3 \\ a_5 & a_6 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a_1 \\ a_4 \end{pmatrix} \tag{5.2}$$

onde a_1, a_2, a_3, a_4, a_5 e a_6 são coeficientes que dependem dos parâmetros de movimento da

câmera. Mais precisamente

$$R_x = a_4, \quad R_y = a_1, \quad R_z = \frac{1}{2}(a_5 - a_3) \quad \text{e} \quad R_f = \frac{1}{2}(a_2 + a_6). \tag{5.3}$$

Aplicando a equação (5.2) aos três pares p_i, v_i, e rearranjando os termos, obtemos o sistema linear 6×6

$$\begin{bmatrix} 1 & x_1 & y_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_1 & y_1 \\ 1 & x_2 & y_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_2 & y_2 \\ 1 & x_3 & y_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_3 & y_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = \begin{bmatrix} v_{x1} \\ v_{y1} \\ v_{x2} \\ v_{y2} \\ v_{x3} \\ v_{y3} \end{bmatrix}. \tag{5.4}$$

Os parâmetros R_x, R_y, R_z e R_f são então obtidos resolvendo-se este sistema e aplicando-se as fórmulas 5.3.

Os autores aplicam este método especificamente ao fluxo obtido dos vetores de movimento do MPEG. Na implementação, os vetores v_1, v_2, v_3 são os três vetores mais dominantes do fluxo MPEG.

A principal desvantagem do método é que este usa somente três vetores do fluxo ótico e, portanto, é muito sensível a movimentos de objetos na cena. Além disso, nem sempre os vetores dominantes são a melhor escolha. Por exemplo, no caso do roll ou zoom um objeto em movimento pode gerar um vetor dominante mesmo ocupando uma fração pequena da imagem.

5.1.3 Algoritmo de Srinivasan, Venkatesh e Hosie (SVH)

Mandyam Srinivasan, Sventha Venkatesh e Robin Hosie [38] apresentaram em 1997 um algoritmo para determinar os parâmetros T_x, T_y, R_x, R_y, R_z e R_f para uma cena estática. Os autores supõem que a velocidade de translação em Z (*dollying*) é nula ($T_z = 0$), de modo que o efeito da translação reduz-se ao campo

$$\frac{1}{Z_i}(T_x, T_y). \tag{5.5}$$

Como T_x e T_y são constantes para todo ponto da imagem, este campo tem vetores paralelos, cuja magnitude varia conforme a profundidade Z da cena.

Supondo-se que os parâmetros R_x, R_y, R_z e R_f estão corretos, o resíduo $d = f - \tilde{f}$ entre o fluxo observado e o modelo estático (sem translação) deve ser devido unicamente

ao termo de translação (equação 5.5) e a erros na determinação do fluxo ótico. Portanto, todos os vetores d_i deveriam ser aproximadamente paralelos.

Seja $\bar{d}$ a média dos vetores d_i, $\bar{d} = \frac{1}{n}\sum d_i$. Na suposição de que T_z é zero, todos os vetores deveriam apontar para a direção de $\bar{d}$. Portanto, uma medida da incorretude dos parâmetros R_x, R_y, R_z e R_f é a soma $\Theta = \sum |d_i||\theta_i|$ dos valores absolutos dos desvios angulares θ_i entre cada d_i e a média $\bar{d}$, ponderados pelo módulo $|d_i|$ de cada resíduo. Note que Θ não depende de T_x e T_y e que $\Theta = 0$ se e somente se existe uma escolha de T_x e T_y e das profundidades Z_i que faz o fluxo ótico previsto pelos parâmetros R_x, R_y, R_z R_f coincidir com o observado.

Portanto, podemos determinar R_x, R_y, R_z e R_f mesmo sem conhecer T_x e T_y encontrando os valores desses quatro parâmetros que minimizem a função Θ. Para a minimização, Srinivasan et al [38] utilizaram o algoritmo de otimização não linear de Nelder-Meade [11].

Uma limitação do algoritmo SVH é que ele só permite quantidades limitadas de movimento de objetos na cena. Além disso, o algoritmo de otimização de Nelder-Meade apresenta um elevado custo computacional e pára facilmente em mínimos locais.

O algoritmo de SVH é utilizado para determinar os parâmetros de câmera em muitos outros algoritmos [33, 35], que diferem na maneira como os vetores de fluxo são obtidos e filtrados antes da análise.

5.2 Algoritmo proposto

Em vista das limitações dos algoritmos anteriores, desenvolvemos um algoritmo original [7] que denominaremos WOFF, (abreviação de *weighted optical flow fitting*). Veja algoritmo 5.2. Nosso algoritmo objetiva extrair do fluxo ótico os parâmetros de movimento da câmera R_x, R_y, R_z e R_f (tilt, pan, roll e zoom), supondo que a câmera é estacionária ($T_x = T_y = T_z = 0$). Suas características principais são:

1. a entrada é um fluxo ótico amostrado, cujos vetores são ponderados de acordo com sua confiabilidade;

2. o algoritmo usa um procedimento de mínimos quadrados ponderado para obter o fluxo ideal que melhor se ajusta ao fluxo observado.

3. os pesos dos vetores do fluxo são ajustados iterativamente com base na análise estatística do fluxo residual, para distingüir entre movimentos da câmera e movimentos

de objetos na cena.

O algoritmo está descrito formalmente abaixo (algoritmo 5.2) e será explicado nas seções 5.2.1—5.2.5.

Algoritmo 5.2 - *WOFF* (p, f, κ, n)

Dados: uma lista p de n pontos, um fluxo ótico f amostrado em p, e uma lista κ de escores de confiabilidade.

Resultado: parâmetros de movimento da câmera R_x, R_y, R_z, R_f e pesos finais $w_1, \ldots, w_n$ para os pontos.

Ajuste inicial de pesos:

para $i \in \{0, 1, \ldots, n-1\}$ **faça** {

  $w_i \leftarrow \kappa_i / \sqrt{|f_i|^2 + \varepsilon^2}$;

}

repita {

  convergiu $\leftarrow$ **verdadeiro**;

  $R_x, R_y, R_z, R_f \leftarrow$ AnáliseMínimosQuadrados (f, p, w);

  Cálculo dos resíduos:

  para $i \in \{0, 1, \ldots, n-1\}$ **faça** $d_i \leftarrow f_i - \tilde{f}(R_x, R_y, R_z, R_f)(p_i)$;

  $\mu \leftarrow$ Média (d, w); $\sigma \leftarrow$ Desvio (d, w, μ);

  Ajuste iterativo:

  para $i \in \{0, 1, \ldots, n-1\}$ **faça** {

    se $(w_i \neq 0)$ e $(|d_i| > 3\sigma)$ **então** {

      $w_i \leftarrow 0$;

      convergiu $\leftarrow$ **falso**;

    }

  }

} **enquanto** (**não** convergiu);

retorne R_x, R_y, R_z, R_f, w;

5.2.1 Fluxo ótico ponderado

A entrada do algoritmo é uma lista de vetores $f_1, f_2, \ldots, f_n$ resultantes da amostragem de um fluxo ótico f em n pontos dados $p_1, p_2, \ldots, p_n$ da imagem; e uma lista de escores $\kappa_1, \kappa_2, \ldots, \kappa_n$, positivos ou nulos, que exprimem a confiabilidade desses vetores. Os vetores do fluxo podem ser obtidos por qualquer um dos algoritmos descritos no capítulo 3. O escore κ_i deve ser elevado para pontos p_i onde a textura local na primeira imagem é facilmente rastreável, e é localizada na segunda imagem pelo algoritmo de fluxo ótico, com alta confiabilidade.

A figura 5.1 mostra uma entrada típica do algoritmo. Nesta imagem e no restante da dissertação, a área do círculo negro em cada ponto p_i é proporcional ao escore κ_i ou peso w_i do ponto.

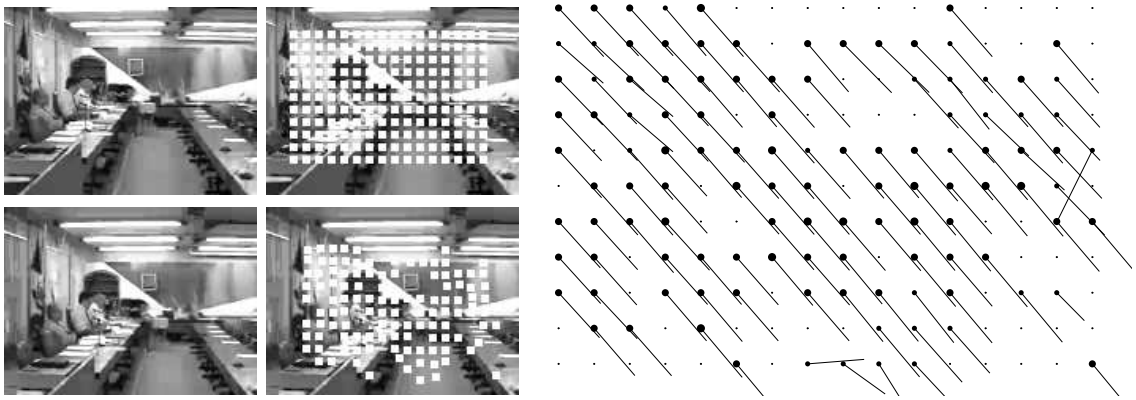


Figura 5.1: Fluxo ótico em um vídeo do Conselho Universitário da Unicamp. Na primeira coluna são mostrados dois quadros consecutivos do vídeo. Na coluna do meio são indicados por quadrados brancos os pontos de amostragem $p_1, p_2, \ldots, p_n$ (imagem no alto) e os pontos correspondentes $p_i + f(p_i)$ rastreados pelo algoritmo KLT (imagem de baixo). A figura à direita mostra os vetores $f_1, f_2, \ldots, f_n$ (traços) e os pesos $w_1, w_2, \ldots, w_n$ (círculos negros).

5.2.2 Determinação dos parâmetros de movimento

O passo central do algoritmo proposto (passo 6 do algoritmo 5.2) é aproximar o fluxo ótico f entre dois quadros por uma combinação linear $\tilde{f}$ de fluxos canônicos, dada pela equação

$$\tilde{f}(p) = \tilde{f}(R_x, R_y, R_z, R_f)(p) = R_x r_x(p) + R_y r_y(p) + R_z r_z(p) + R_f r_f(p) \tag{5.6}$$

para todo $p \in \mathfrak{D}J$.

Os coefficientes R_x, R_y, R_z, R_f são calculados diretamente, através do procedimento de mínimos quadrados ponderado [11]. Para tal propósito, definimos o *produto escalar* de dois fluxos a e b, com a *função de peso* w, como

$$\langle a|b\rangle = \frac{\int_D w(p)(a(p) \cdot b(p))\, dp}{\int_D w(p)}. \tag{5.7}$$

A versão discreta desta fórmula, supondo que as imagens são amostradas nos pontos $p_1, p_2, \ldots, p_n$ é

$$\langle\langle a|b\rangle\rangle = \frac{\sum_{i=1}^{n} w_i(a_i \cdot b_i)}{\sum_{i=1}^{n} w_i}. \tag{5.8}$$

Definimos também a *norma de um fluxo* f com *função de peso* w como sendo

$$|f| = \sqrt{\langle f|f\rangle} \tag{5.9}$$

na versão contínua, ou na versão discreta

$$\|f\| = \sqrt{\langle\langle f|f\rangle\rangle}. \tag{5.10}$$

As fórmulas (5.7) e (5.8) obviamente satisfazem as definições de produto escalar e norma, desde que os pesos w_i sejam todos positivos. (Quando alguns dos pesos w_i são zero, $\|f\|$ ainda é uma *semi-norma*.)

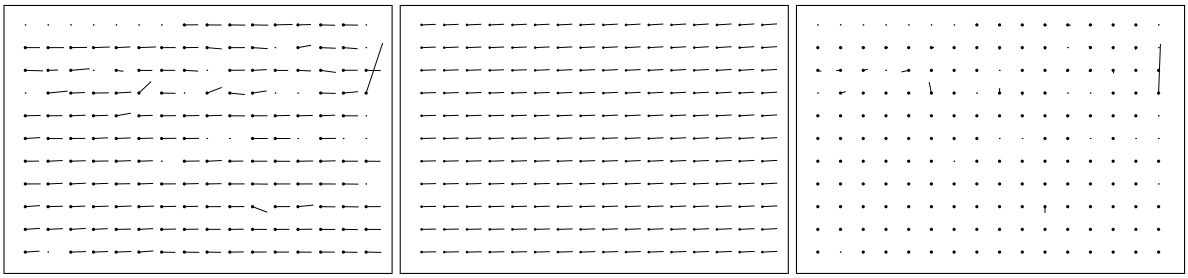
O objetivo então é determinar os valores de R_x, R_y, R_z e R_f que minimizam a discrepância $d = f - \tilde{f}(R_x, R_y, R_z, R_f)$ entre o fluxo dado f e o fluxo ideal $\tilde{f}$ da equação (5.6). A discrepância pode ser medida pela média ponderada dos quadrados dos módulos dos vetores d_i, isto é,

$$Q(R_x, R_y, R_z, R_f) = \|d\|^2 = \left\langle\left\langle f - \tilde{f}|f - \tilde{f}\right\rangle\right\rangle. \tag{5.11}$$

Conforme a teoria do método dos mínimos quadrados, os valores de R_x, R_y, R_z, R_f que minimizam Q são determinados pelo sistema linear de equações

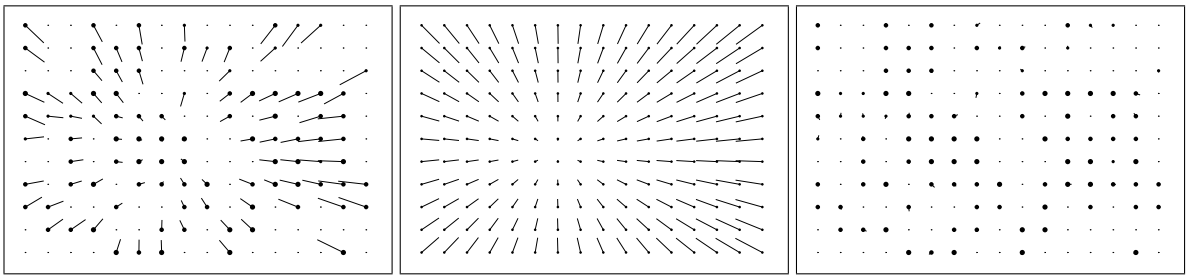
$$\begin{bmatrix} \langle\langle r_x|r_x\rangle\rangle & \langle\langle r_x|r_y\rangle\rangle & \langle\langle r_x|r_z\rangle\rangle & \langle\langle r_x|r_f\rangle\rangle \\ \langle\langle r_y|r_x\rangle\rangle & \langle\langle r_y|r_y\rangle\rangle & \langle\langle r_y|r_z\rangle\rangle & \langle\langle r_y|r_f\rangle\rangle \\ \langle\langle r_z|r_x\rangle\rangle & \langle\langle r_z|r_y\rangle\rangle & \langle\langle r_z|r_z\rangle\rangle & \langle\langle r_z|r_f\rangle\rangle \\ \langle\langle r_f|r_x\rangle\rangle & \langle\langle r_f|r_y\rangle\rangle & \langle\langle r_f|r_z\rangle\rangle & \langle\langle r_f|r_f\rangle\rangle \end{bmatrix} \begin{bmatrix} R_x \\ R_y \\ R_z \\ R_f \end{bmatrix} = \begin{bmatrix} \langle\langle \mathrm{f}|r_x\rangle\rangle \\ \langle\langle \mathrm{f}|r_y\rangle\rangle \\ \langle\langle \mathrm{f}|r_z\rangle\rangle \\ \langle\langle \mathrm{f}|r_f\rangle\rangle \end{bmatrix} \tag{5.12}$$

As figuras 5.2 e 5.3 mostram dois exemplos desta análise.



(a) (b) (c)

Figura 5.2: Exemplo da análise de mínimos quadrados para o fluxo observado durante um pan da câmera; (a) um fluxo amostrado f, com os pesos w_i indicados pelo tamanho dos pontos; (b) fluxo $\tilde{f}$ ajustado a f pelo método dos mínimos quadrados ponderados; (c) resíduo $d = f - \tilde{f}$.



(a) (b) (c)

Figura 5.3: Exemplo da análise de mínimos quadrados de um fluxo observado durante uma combinação de zoom-out e pan para a direita; (a) um fluxo alinhado f, com os pesos w_i indicados pelo tamanho dos pontos; (b) fluxo $\tilde{f}$ ajustado a f pelo método dos mínimos quadrados ponderados; (c) resíduo $d = f - \tilde{f}$.

Note que a matriz (5.12) é simétrica, portanto basta calcular quatorze produtos escalares distintos $\langle\langle \cdot | \cdot \rangle\rangle$; sendo que cada produto escalar implica em $3n$ multiplicações e $3n$ somas. A solução do sistema (5.12) pode ser encontrada pelo método de Gauss-Seidel em tempo constante (independente de n).

5.2.3 Pesos iniciais

Neste trabalho, usamos o algoritmo KLT para calcular o fluxo ótico, como descrito na seção 3.3. Em particular, modificamos a implementação padrão do KLT de tal forma que ele retorne, além do deslocamento $f(p_i)$, de cada ponto de amostragem, também os dois autovalores $\lambda_1(p_i)$ e $\lambda_2(p_i)$ (passo 18 do algoritmo 3.3.1) da matriz hessiana G, assim como a discrepância local $\mathfrak{r}(p_i)$ correspondente a esse deslocamento (passo 17 do algoritmo 3.3.1). O *escore de confiabilidade* κ_i de cada ponto foi definido como a razão

$$\kappa_i = \frac{\min(\lambda_1(p_i), \lambda_2(p_i))}{\mathfrak{r}(p_i)}. \tag{5.13}$$

Intuitivamente, se κ_i é alto, o ponto é fácil de ser localizado e seu casamento na imagem posterior é muito bom; nesse caso a probabilidade do vetor de fluxo ótico f_i estar correto é alta.

5.2.4 Ajuste dos pesos iniciais

O método dos mínimos quadrados (5.12) funciona bem se a cena é estacionária. Corpos que se movem (como na figura 5.4) introduzem erros nos parâmetros ajustados R_x, R_y, R_z e R_f, pois o procedimento dos mínimos quadrados produz uma média entre o fluxo devido

ao movimento da câmera e o devido ao movimento dos objetos. Este não é um problema significativo se o objeto que se move ocupa uma pequena fração da imagem e/ou sua velocidade é pequena se comparada ao fluxo de movimento da câmera. No entanto, se a cena contém objetos que se movem rapidamente, o fluxo desse objeto pode facilmente dominar o fluxo ajustado $\tilde{f}$.

Com o objetivo de aliviar este problema, definimos os pesos w_i (passos 1—3 do algoritmo 5.2) como sendo os escores de confiabilidade κ_i fornecidos como entrada, divididos pelo tamanho dos vetores de fluxos correspondentes f_i, ou seja

$$w_i = \frac{\kappa_i}{\sqrt{|f_i|^2 + \varepsilon^2}}. \tag{5.14}$$

Nesta fórmula, ε é uma pequena constante, introduzida para evitar divisão por zero ou por números muito pequenos. Observe que o peso w_i será no máximo κ_i/ε.

Note que a equação (5.14) aumenta o peso relativo de deslocamentos pequenos, enquanto reduz os pesos de deslocamentos grandes. A figura 5.4 mostra o efeito desta correção. A justificativa para esta correção é que pequenos vetores de fluxo são geralmente mais significativos, estatisticamente, que os grandes. Se o fluxo amostrado f contém um conjunto grande C de pontos com pequenos vetores de fluxo e um conjunto menor de pontos com fluxos grandes, a explicação mais provável é que os primeiros são devidos a movimento de câmera, e os segundos a movimentos de objetos. A hipótese alternativa — que os movimentos da câmera e do objeto se cancelam, de modo que C são pontos do objeto — é menos provável, pelo menos para os tipos de vídeos que estamos considerando neste trabalho.

5.2.5 Ajuste iterativo dos pesos

O método dos mínimos quadrados supõe implicitamente que os desvios entre um fluxo observado f e um fluxo ideal $\tilde{f}$ são devidos a ruídos aditivos independentes com distribuição Gaussiana. Com esta suposição, o método dos mínimos quadrados pode ser justificado em termos do *princípio da máxima verossimilhança*.

Tal suposição pode ser adequada para cenas estáticas com texturas ricas, onde se espera que as principais fontes de erros no fluxo f sejam devidas a erros de medida dos sensores da câmera, e erros de quantização na conversão analógico-digital. Tal como dito, para cenas com objetos se movimentando esta suposição é substancialmente incorreta, pois todos os vetores do fluxo na região da imagem coberta por esses objetos são completamente substituídos por vetores espúrios, todos apontando aproximadamente na mesma direção.

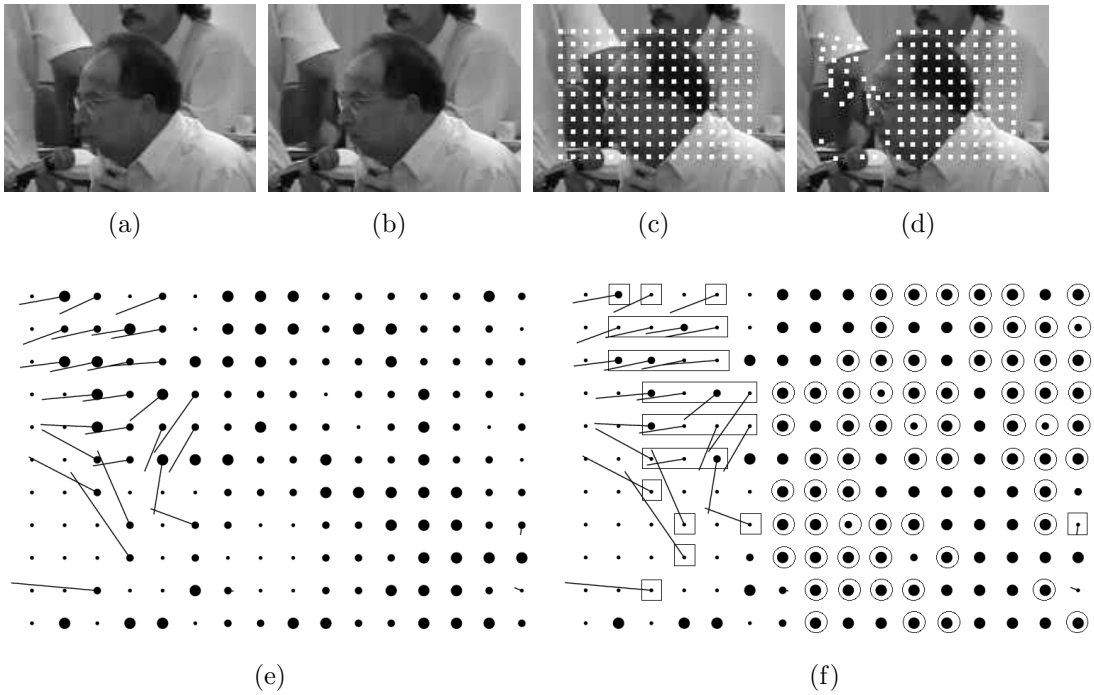


Figura 5.4: Efeito do ajuste inicial dos pesos: No alto, dois quadros consecutivos (a, b), os pontos de amostragem (c) e os pontos correspondentes encontrados pelo algoritmo do KLT em (d). Abaixo, os vetores de fluxo ótico e as confiabilidades κ_i (círculos pretos) (e), e os pesos ajustados w_i (f). Note que o peso dos vetores no braço que tem deslocamentos maiores foi diminuído (indicados por retângulos) em relação aos pesos dos pontos na cabeça e no fundo. No caso dos pontos circundados, houve um aumento do respectivo peso.

Para solucionar este problema, executamos diversas iterações do procedimento de ajuste por mínimos quadrados (passos 4—15 do algoritmo 5.2) ajustando os pesos a cada iteração de forma a excluir de consideração os vetores do fluxo que parecem ser devidos a objetos se movimentando (passos 9—14). Informalmente, em cada iteração fazemos $w_i \leftarrow 0$ para qualquer vetor do fluxo f_i que tenha um desvio considerável em relação ao fluxo ajustado $\tilde{f}_i$. Então, o próximo passo do ajuste por mínimos quadrados provavelmente produzirá um fluxo que é mais similar aos vetores com $w_i \neq 0$ que aqueles com $w_i = 0$.

Mais precisamente, após o cálculo dos parâmetros por mínimos quadrados, computamos a média ponderada μ dos vetores de discrepância entre o fluxo f e o fluxo de movimento de câmera previamente ajustado $\tilde{f}$:

$$\mu = \frac{\sum_{i=1}^{n} w_i (f_i - \tilde{f}_i)}{\sum_{i=1}^{n} w_i}. \tag{5.15}$$

Computamos também o desvio padrão

$$\sigma = \sqrt{\frac{\sum w_i \left| f_i - \tilde{f}_i - \mu \right|^2}{\sum w_i}}. \tag{5.16}$$

Todo ponto p_i cuja discrepância $f_i - \tilde{f}_i$ difere da média μ em mais que 3σ é eliminado dos dados, através do anulamento de seu peso w_i. O procedimento de ajuste por míminos quadrados é então aplicado novamente com estes novos pesos para computar R_x, R_y, R_z e R_f. Este processo é repetido até que nenhum ponto seja eliminado por este critério.

De acordo com os testes realizados, este procedimento geralmente converge em poucas iterações para um fluxo $\tilde{f}$ que casa com o fluxo das partes estáticas da cena — contanto que essas partes ocupem ao menos metade da imagem. Os vetores de fluxo que correspondem a objetos em movimento geralmente terminam com peso $w_i = 0$.

A figura 5.5 mostra a eliminação de vetores de fluxo conforme este procedimento iterativo. Podemos notar que alguns dos pontos a serem rastreados na pessoa que se move tem peso w_i alto. Como o movimento não cobre uma área maior que a metade da imagem, esses vetores são eliminados em cada passo da iteração, figura 5.5(f,g), sendo que na figura 5.5(h) somente os pontos que pertecem a cena estática são preservados.

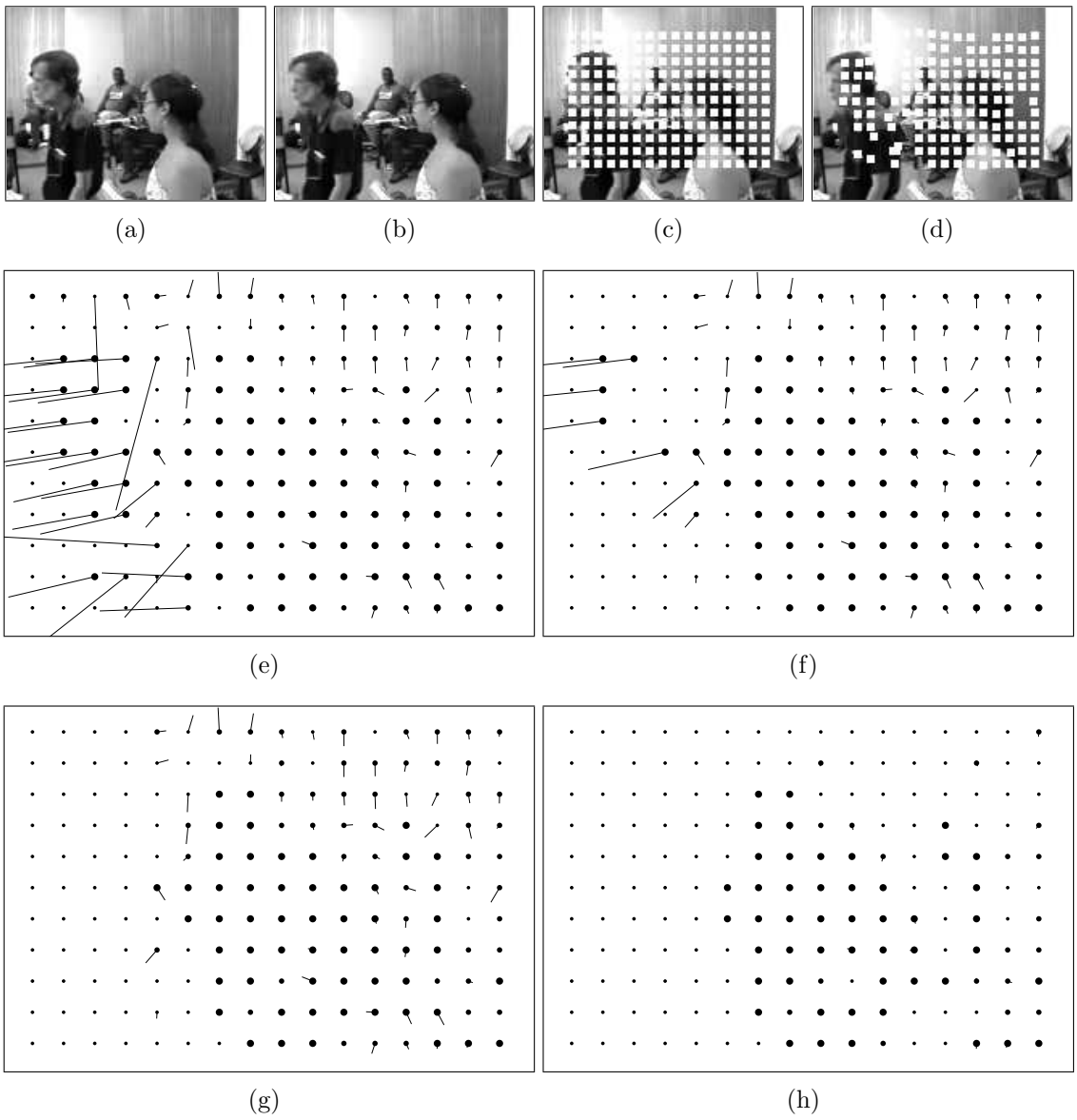


Figura 5.5: Eliminação de vetores de fluxo ótico devido a movimentos da cena entre dois quadros (a-b), os pesos iniciais w_i (e), e o efeito de três iterações do ajuste iterativo de pesos (f-h).

5.3 Comparação com outros métodos

Ao contrário dos métodos LH (seção 5.1.1) e OST (seção 5.1.2), o método aqui proposto utiliza toda a informação contida no fluxo de entrada, e é portanto mais robusto em relação a ruído. Ao contrário do algoritmo SVH (seção 5.1.3), nosso algoritmo permite objetos em movimento na cena, desde que não dominem a área da imagem.

Por outro lado, o algoritmo proposto, tal como apresentado, não considera tracking (T_x, T_y, T_z). Entretanto, ele pode ser usado como uma etapa de um algoritmo iterativo, similar ao algoritmo SVH. Nesta proposta, a cada iteração o nosso método é usado para (re)calcular os quatro parâmetros R_x, R_y, R_z e R_f, supondo estimativas correntes para os parâmetros T_x, T_y e T_z e para a profundidade Z de cada ponto. Em seguida, os parâmetros T_x e T_y são obtidos analisando-se as direções dos vetores do resíduo $d = f - \tilde{f}$, supondo R_x, R_y, R_z e R_f fixos. Finalmente, a profundidade Z de cada ponto é calculada como em SVH.

Acreditamos que esta abordagem seria muito mais eficiente que a implementação original em SVH, porque a determinação de R_x, R_y, R_z e R_f é direta (por mínimos quadrados), e a determinação de T_x, T_y é uma minimização não linear em duas variáveis, em vez de seis. Além disso, o ajuste iterativo dos pesos no nosso algoritmo deve permitir usar esta abordagem mesmo com movimento de objetos na cena, ao contrário de SVH.

Capítulo 6

Testes da detecção de movimento

Neste capítulo relatamos testes experimentais da capacidade do algoritmo em detectar e discernir os movimentos de câmera. Esta característica é essencial para a aplicação motivadora deste trabalho, segmentação estrutural de vídeos de conferências.

6.1 Vídeos de teste

Nos experimentos utilizamos três vídeos de reuniões de Câmaras do Conselho Universitário (CONSU) da UNICAMP, [39, 40, 41], e seis vídeos obtidos da internet com gravações de ambientes externos (paisagens, feira, etc) [42, 43, 44, 45, 46, 47]. As informações detalhadas sobre os vídeos utilizados são mostradas na tabela 6.1.

Os vídeos do Conselho Universitário são registros ininterruptos de reuniões que se estedem por várias horas. Eles foram filmados por uma equipe da universidade, em cor, com uma única câmera (modelo Panasonic PV-DV401). A resolução original era de 320×240 pixels no formato RealPlayer [48]. Para a análise, convertemos esses vídeos em um conjunto de imagens em níveis de cinza de 8 bits no formato Portable Gray Map (PGM), extraídos do vídeo com a ferramenta ffmpeg [49].

Todos os vídeos foram reduzidos em escala. A fim de reduzir o nível de ruídos de quantização e compressão usamos aproximadamente a mesma resolução espacial para facilitar a interpretação dos resultados. Os vídeos das reuniões do Conselho Universitário, em particular, tiveram sua resolução linear reduzida à metade, para 160×120 pixels. Os demais vídeos foram reduzidos para um terço do original, de 480×324 para 160×108 pixels.

Vídeo	Tamanho		Formato	Dimensões	Trans.	Pares
	(minutos)	(quadros)		(quadro)	de Cenas	M
consu1 [39]	00:39:27	59140	RealPlayer	160×120	80	1360
consu2 [40]	00:11:04	16600	RealPlayer	160×120	23	688
consu3 [41]	01:38:38	147959	RealPlayer	160×120	192	4459
montanha1 [42]	00:00:38	341	AVI	160×108	3	119
península [43]	00:00:17	195	AVI	160×108	2	50
montanha2 [44]	00:00:16	183	AVI	160×108	2	45
cidade [45]	00:00:19	247	AVI	160×108	2	109
feira [46]	00:00:21	261	AVI	160×108	4	122
fazenda [47]	00:00:14	177	AVI	160×108	1	53

Tabela 6.1: Informações sobre os vídeos utilizados nos experimentos.

6.1.1 Gabaritos

Para avaliar a performance do método proposto, todos os vídeos utilizados foram segmentados manualmente na íntegra. Todos os pares de quadros consecutivos em cada vídeo foram classificados manualmente nas duas classes, pares com *movimento de câmera* (M) e pares com *câmera estacionária* (S). Os vídeos do CONSU prestam-se muito bem a esta classificação, pois a câmera normalmente permanece estática enquanto um membro do Conselho está falando, e sofre movimentos de pan, tilt e zoom (seqüenciais ou combinados) quando o orador muda. Assim, há uma distinção bem definida entre as classes M e S.

6.1.2 Processamento de cada vídeo

Para cada vídeo, todos os pares de quadros consecutivos foram analisados pelo algoritmo WOFF (seção 5.2). Para cada par de quadros calculamos o fluxo ótico com o algoritmo KLT modificado (seção 3.3), com janela Ω de dimensões 7×7 pixels. O fluxo foi amostrado em uma grade regular de 16×11 pontos p_i, cobrindo a imagem inteira menos uma margem de segurança de 14 pixels ao redor das bordas da imagem. Veja a figura 6.1.

O algoritmo WOFF foi então aplicado a esses fluxos, para computar os quatro coeficientes R_x, R_y, R_z e R_f do fluxo ajustado.

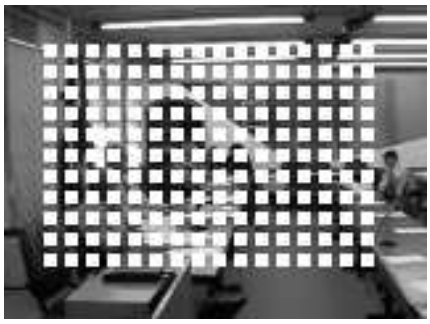


Figura 6.1: Grade de amostragem com 16×11 pontos utilizada nos testes.

6.2 Exemplos de movimentos de câmera e de objetos

Nas figuras 6.2, 6.3 e 6.4 mostramos os gráficos dos quatro coeficientes (R_x, R_y, R_z e R_f) calculados pelo algoritmo WOFF para três trechos de vídeo onde ocorrem movimentos significativos de câmera e/ou de cena. Em cada figura, mostramos os quadros correspondentes a oito eventos da seqüência.

Os vídeos da figuras 6.2 e 6.3 (**consu2** e **consu3**, respectivamente) mostram exemplos representativos do trabalho de câmera típico das reuniões do Conselho Universitário, que consiste de um zoom-out no orador que acabou o discurso (pico negativo no R_f) um pan para localizar o próximo orador (pico no R_y), e finalmente, um zoom-in para enquadrar o orador que iniciou o discurso (pico positivo no R_f). Geralmente, o início do movimento do pan ocorre simultaneamente com um zoom-in ou zoom-out.

Note-se que, quando o movimento de câmera é muito rápido (como no evento E2 da figura 6.2 e nos eventos E2 e E3 da figura 6.3), o fluxo ótico pode ter erros grandes, o que torna a discrepância D elevada, veja os picos no gráfico referente a D para os referidos eventos. A discrepância é elevada também quando há muita oclusão (como nos eventos E6 e E7 da figura 6.2), pois quase todos os pontos a serem rastreados são perdidos.

Na figura 6.4 mostra-se toda a seqüência do vídeo **feira**. A gravação mostra uma feira ao ar livre, onde o trabalho de câmera contém diversos movimentos lentos e médios, veja a curva do pan e do zoom (R_y e R_f). Alguns defeitos como perda de foco aparecem ao longo do vídeo, veja evento E1.

Observe nestas figuras que as curvas das magnitudes de tilt, pan, roll e zoom indicam algumas vezes movimentos combinados de câmera. Por exemplo, o evento E3 da figura 6.2 com uma combinação de zoom e pan.

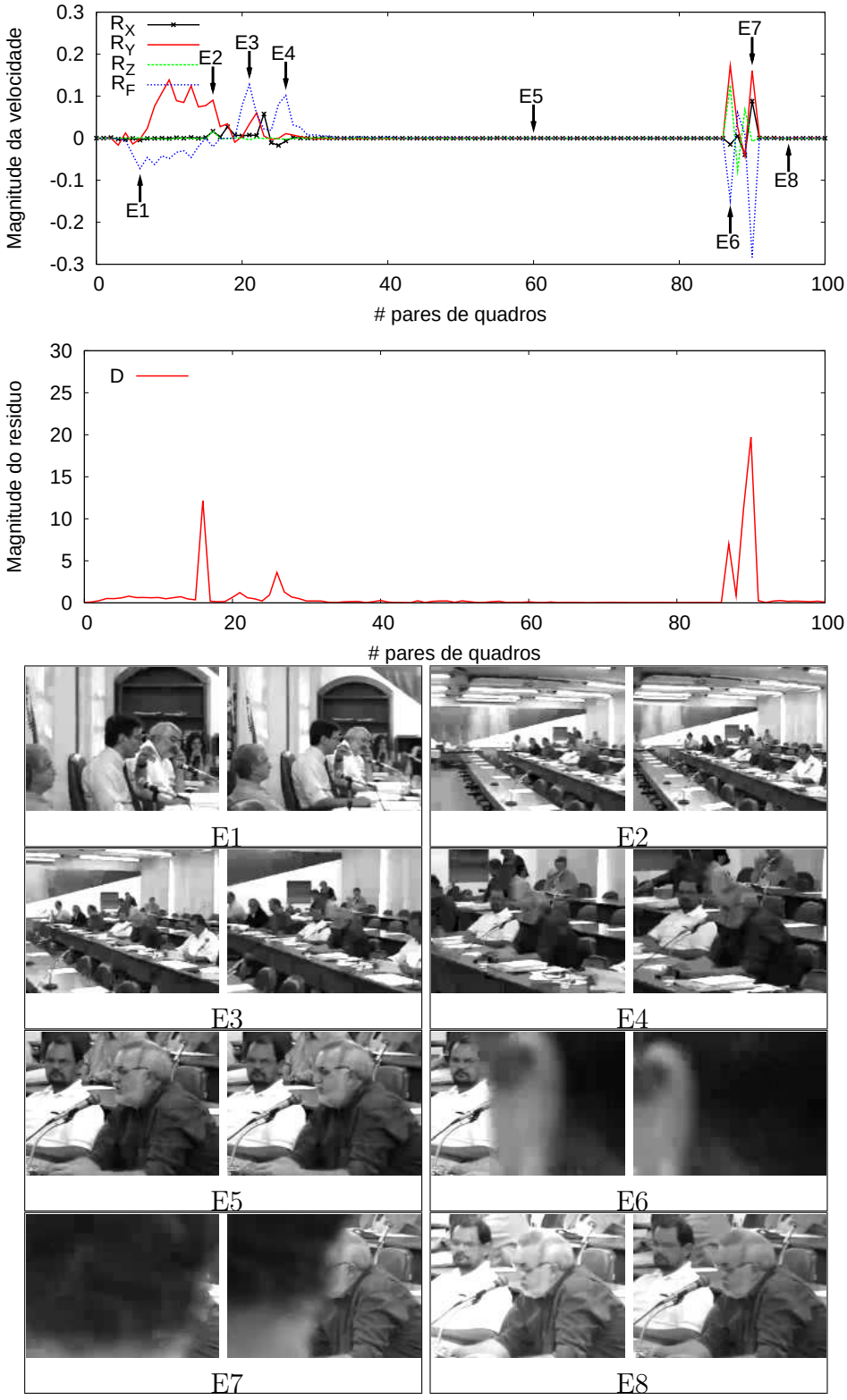


Figura 6.2: No primeiro gráfico são mostradas as variações das velocidades de R_x, R_y, R_z e R_f para um segmento com 200 quadros do vídeo **consu2** (100 pares). As amplitudes em cada uma das curvas indica o movimento referido tal como pode ser visto nos eventos E1—E8. O segundo gráfico mostra o resíduo $D = ||d|| = ||f - \tilde{f}||$ em cada análise.

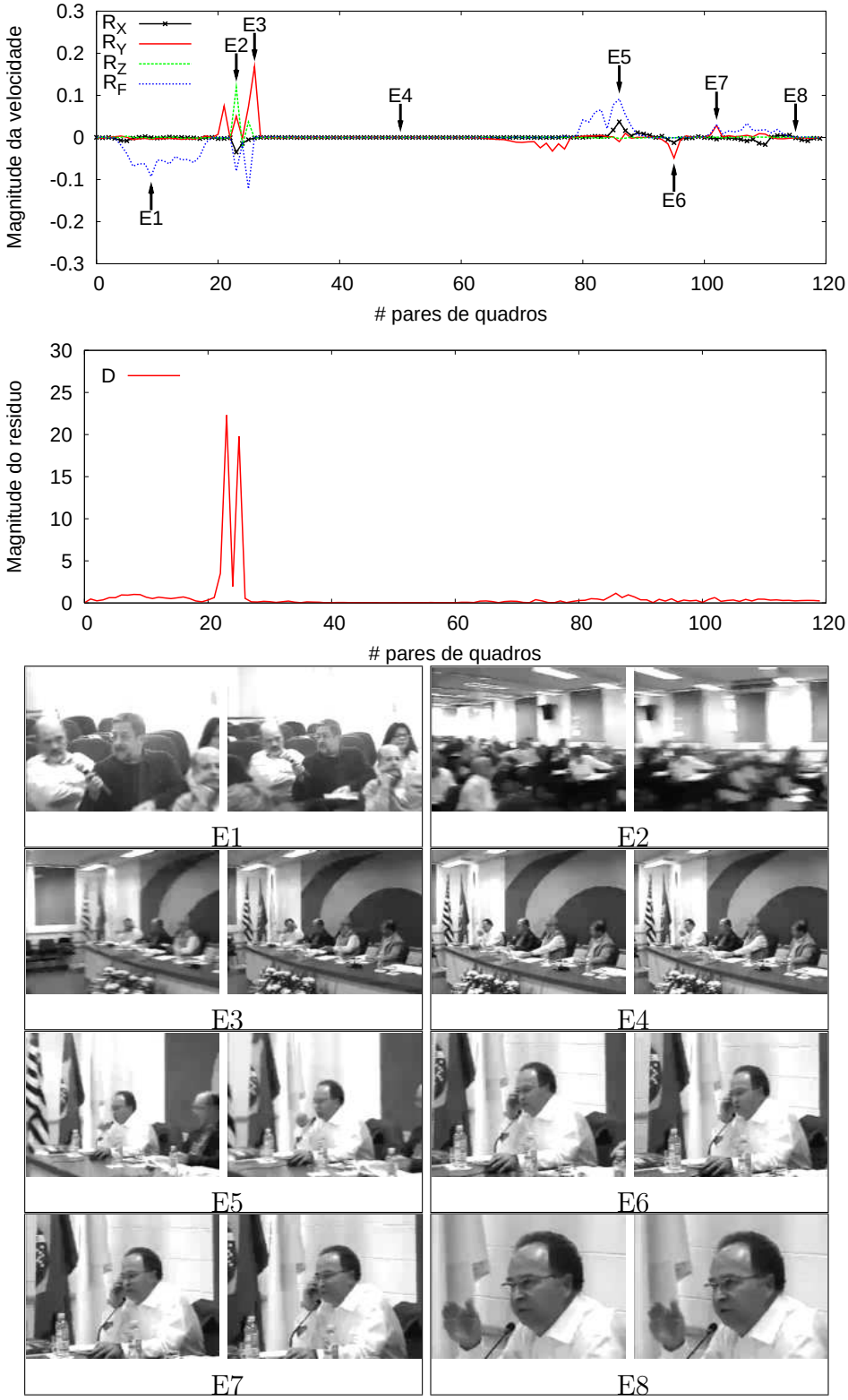


Figura 6.3: Outro gráfico com variações das velocidades de R_x, R_y, R_z, R_f, resíduo D e oito eventos selecionados de um segmento com 240 quadros do vídeo **consu3**.

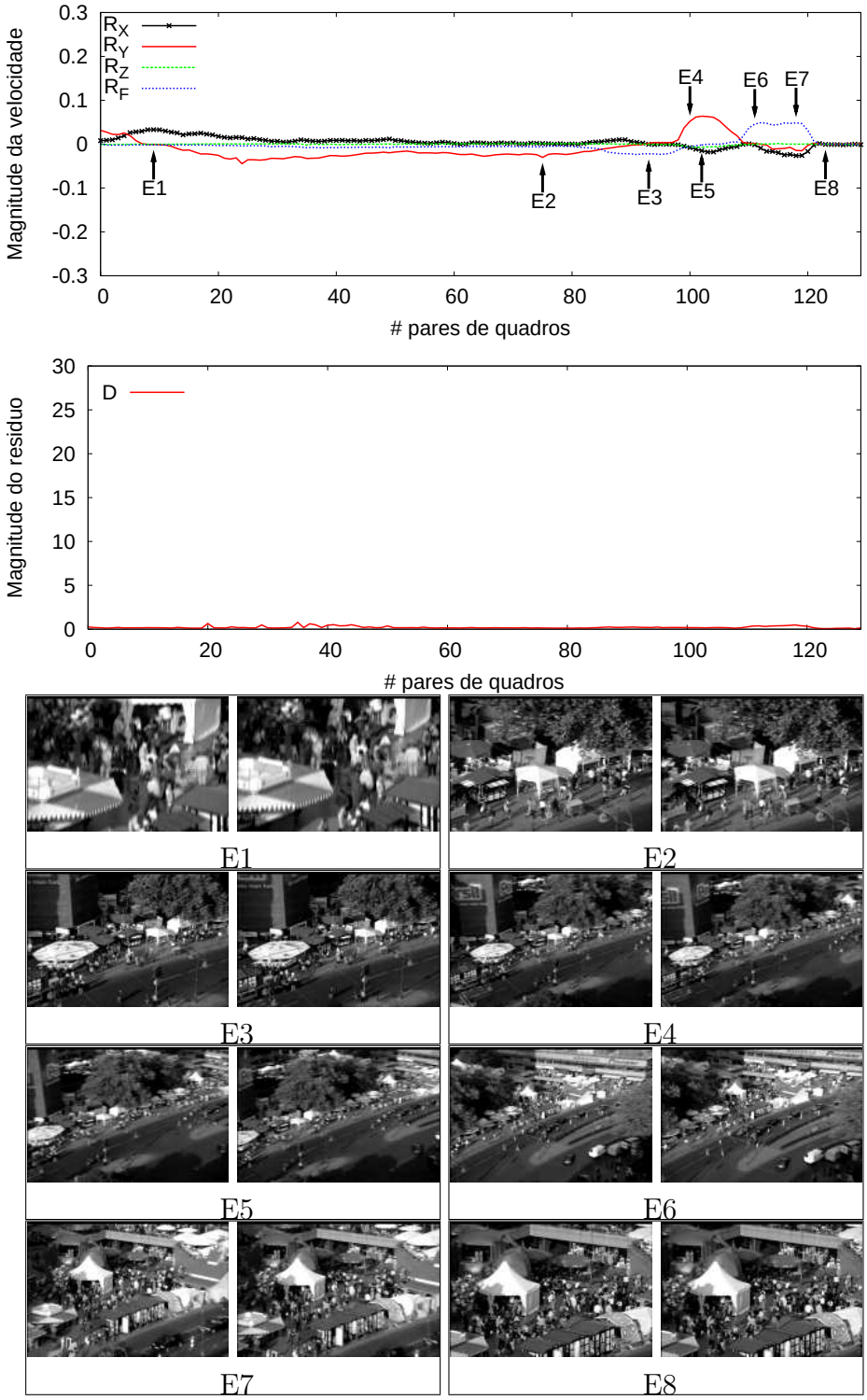
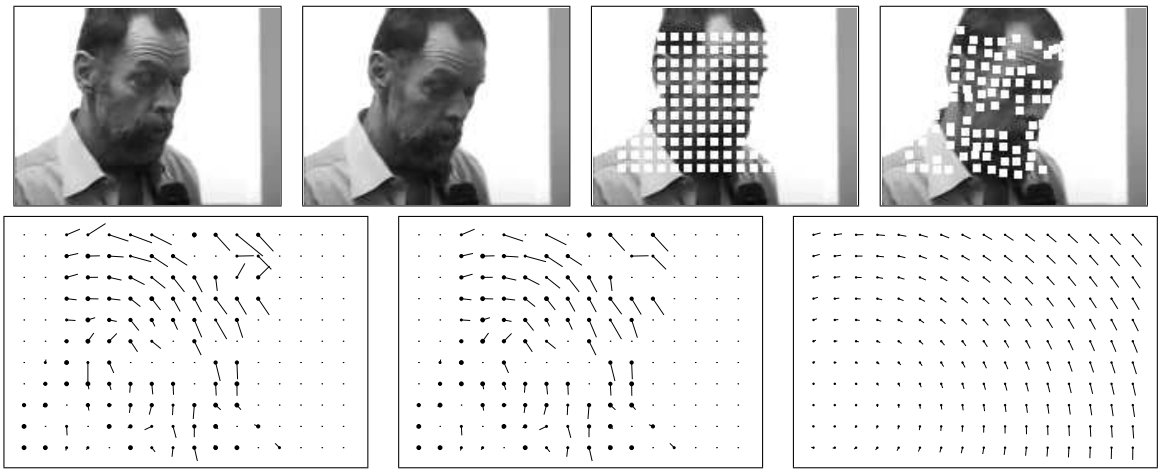


Figura 6.4: Outro gráfico com variações das velocidades de R_x, R_y, R_z, R_f e resíduo D do vídeo **feira** na íntegra, e oito eventos selecionados.

6.2.1 Discussão dos exemplos

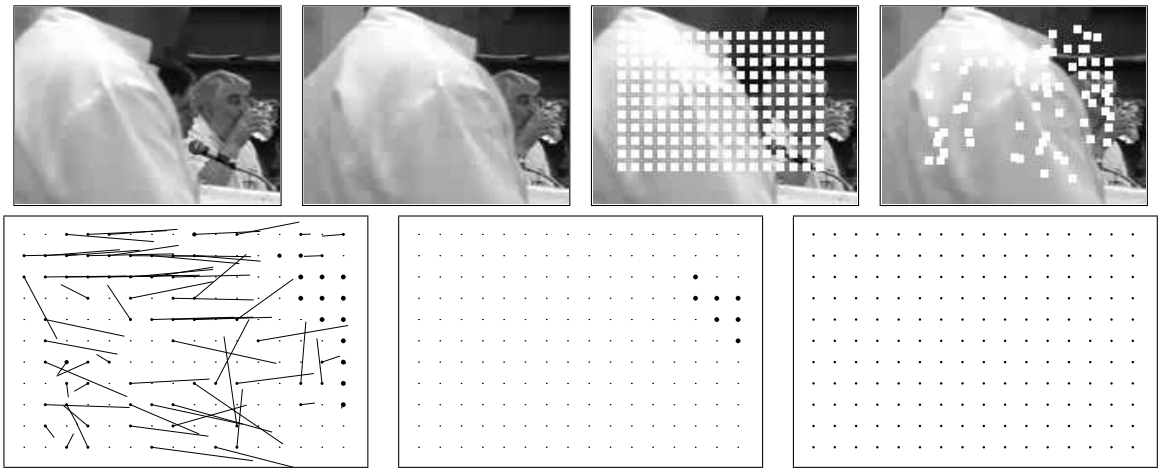
Uma possível fonte de erros da análise do algoritmo proposto ocorre quando uma componente movimenta-se sobre uma cena pobre em textura (veja figura 6.5). Como o rastreamento dos pontos amostrados na cena (parede branca no caso do exemplo) é muito sujeito a erros devido ao problema da ambigüidade uni e bidimensional (veja seção 2.4.1), seus vetores de deslocamento não influenciam significativamente o fluxo estimado $\tilde{f}$. Por outro lado, se o objeto em movimento é rico em textura, caso do exemplo, seus vetores de deslocamento são mais confiáveis e, portanto, acabam dominando o fluxo estimado $\tilde{f}$.



fluxo f com escores κ_i | fluxo f com pesos finais | fluxo ajustado $\tilde{f}$

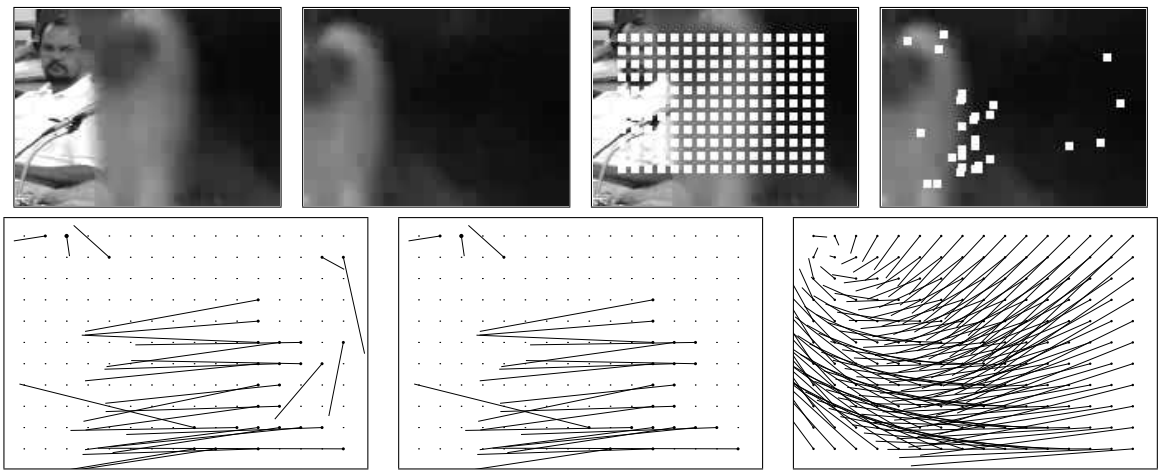
Figura 6.5: Movimento em cena estática pobre em textura causa análise incorreta do fluxo.

Outra fonte de erros é quando o objeto que se move ocupa uma grande fração da imagem, como no caso de oclusões. Neste caso pode ocorrer duas situações; a primeira causa um fluxo ótico desorganizado devido a alta velocidade e falta de textura do objeto em movimento, que faz com que a iteração elimine todos os vetores do fluxo. Veja a figura 6.6. Entretanto, pode acontecer que o objeto em movimento da câmera produza um fluxo organizado, ocupando mais da metade da imagem, que será interpretado como movimento de câmera na direção oposta. Veja figura 6.7.



fluxo f com escores κ_i | fluxo f com pesos finais | fluxo ajustado $\tilde{f}$

Figura 6.6: Movimento de cena produzindo fluxo caótico, cujos vetores são eliminados no ajuste iterativo.



fluxo f com escores κ_i | fluxo f com pesos finais | fluxo ajustado $\tilde{f}$

Figura 6.7: Movimento de cena produzindo um fluxo coerente, que é confundido com movimento de câmera.

6.3 Detecção automática de movimentos de câmera

A experiência mostra que a iteração converge para o fluxo ótico que é devido ao movimento de câmera, descartando a parte dos dados devido a movimento de objetos na cena. Portanto, a movimentação da câmera pode ser detectada pelo fato dos parâmetros R_x, R_y, R_z e R_f terem valores significativos.

6.3.1 Detecção por limiar no movimento

Mais precisamente definimos $V = \sqrt{R_x + R_y + R_z + R_f}$ como sendo uma medida de movimentação da câmera (ou magnitude do fluxo $\tilde{f}$). A presença de movimento seria detectada pela condição $V > V_{\min}$ onde $V_{\min}$ é um limiar dado, que depende da magnitude dos erros no fluxo observado.

Este critério pode não ser suficiente pois em cenas com câmera estática e objetos em movimento a iteração pode terminar com um fluxo $\tilde{f}$ diferente de zero. Para distingüir estes casos, consideramos também a magnitude $D = \left\| f - \tilde{f}(R_x, R_y, R_z, R_f) \right\|$ da discrepância (resíduo) entre o fluxo f calculado pelo KLT e o fluxo ajustado $\tilde{f}$. Pares de quadros com $D > D_{\min}$ são considerados classe S, independentemente do parâmetro V.

Nas figuras 6.8, 6.9 e 6.10 são mostrados gráficos padrões em escala logarítmica das quantidades V e D para os vídeos **consu1**, **consu2** e **consu3**, respectivamente.

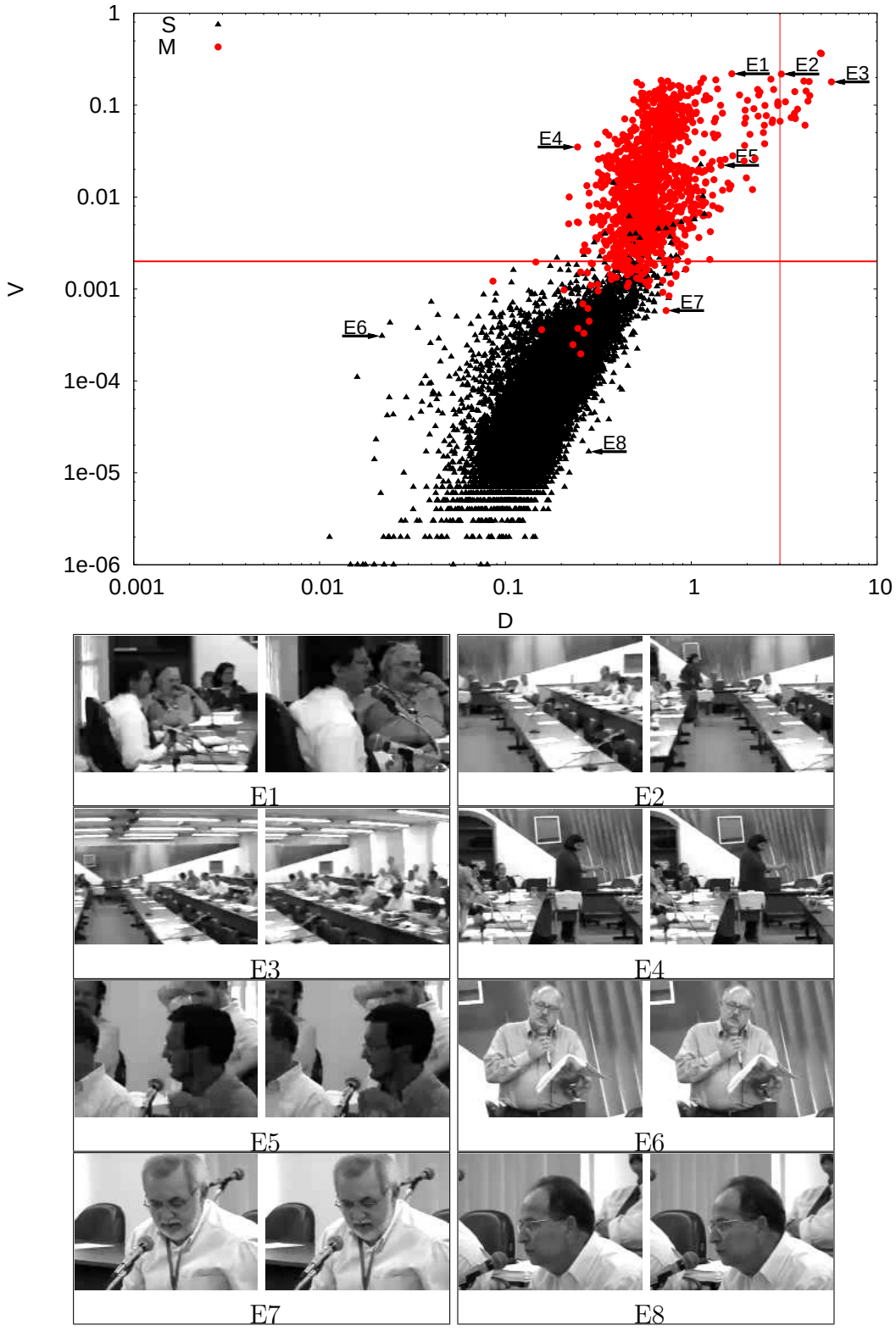


Figura 6.8: Gráfico da velocidade de câmera estimada V pelo erro residual D para todos os pares de quadro do **consu1**. Os pares de quadros do tipo S (câmera estacionária) e M (câmera em movimento) são representados por triângulos e círculos, respectivamente. Oito eventos, indicados por setas, mostram o que representam.

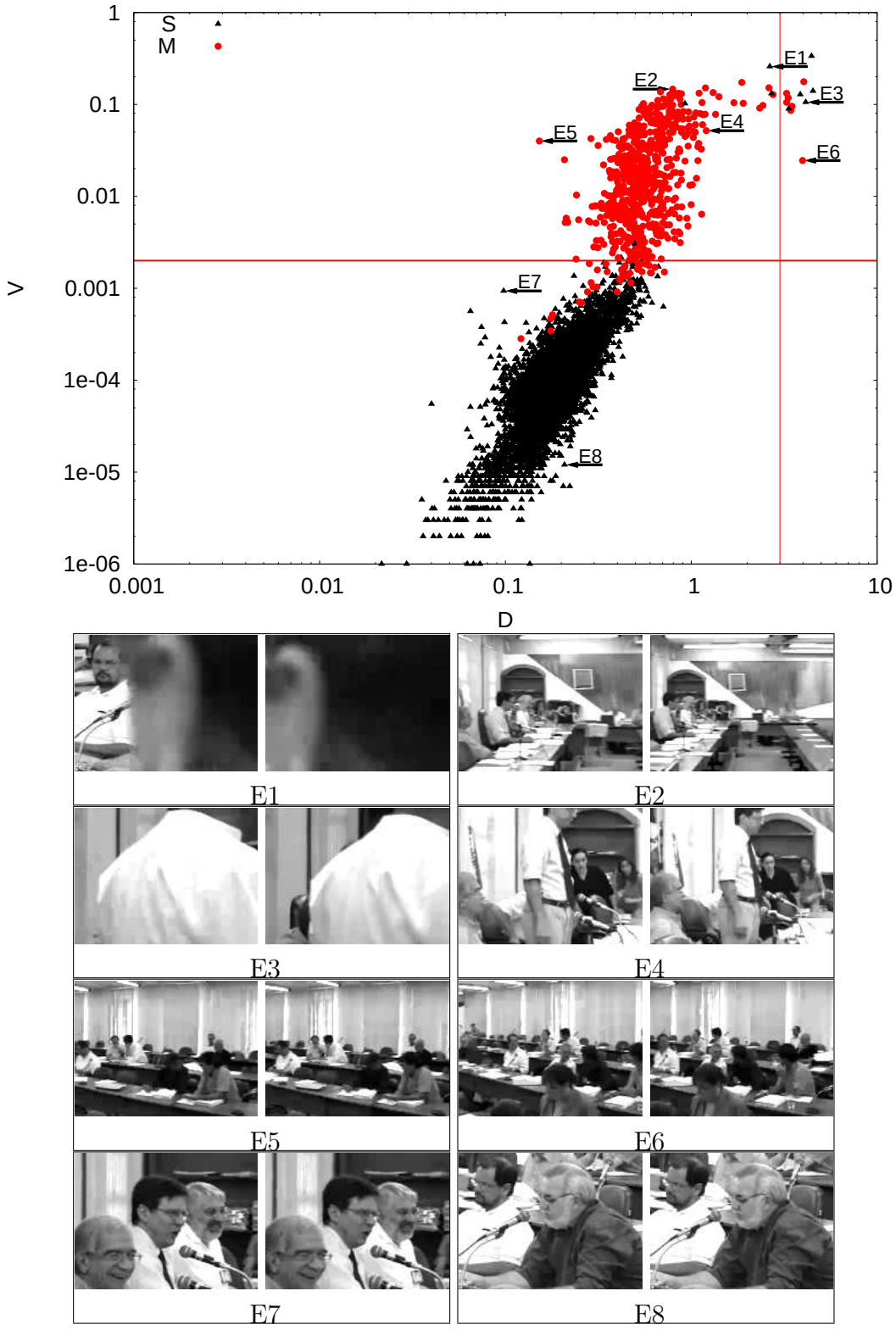


Figura 6.9: Discriminação de movimento com a utilização do algoritmo WOFF no vídeo **consu2**.

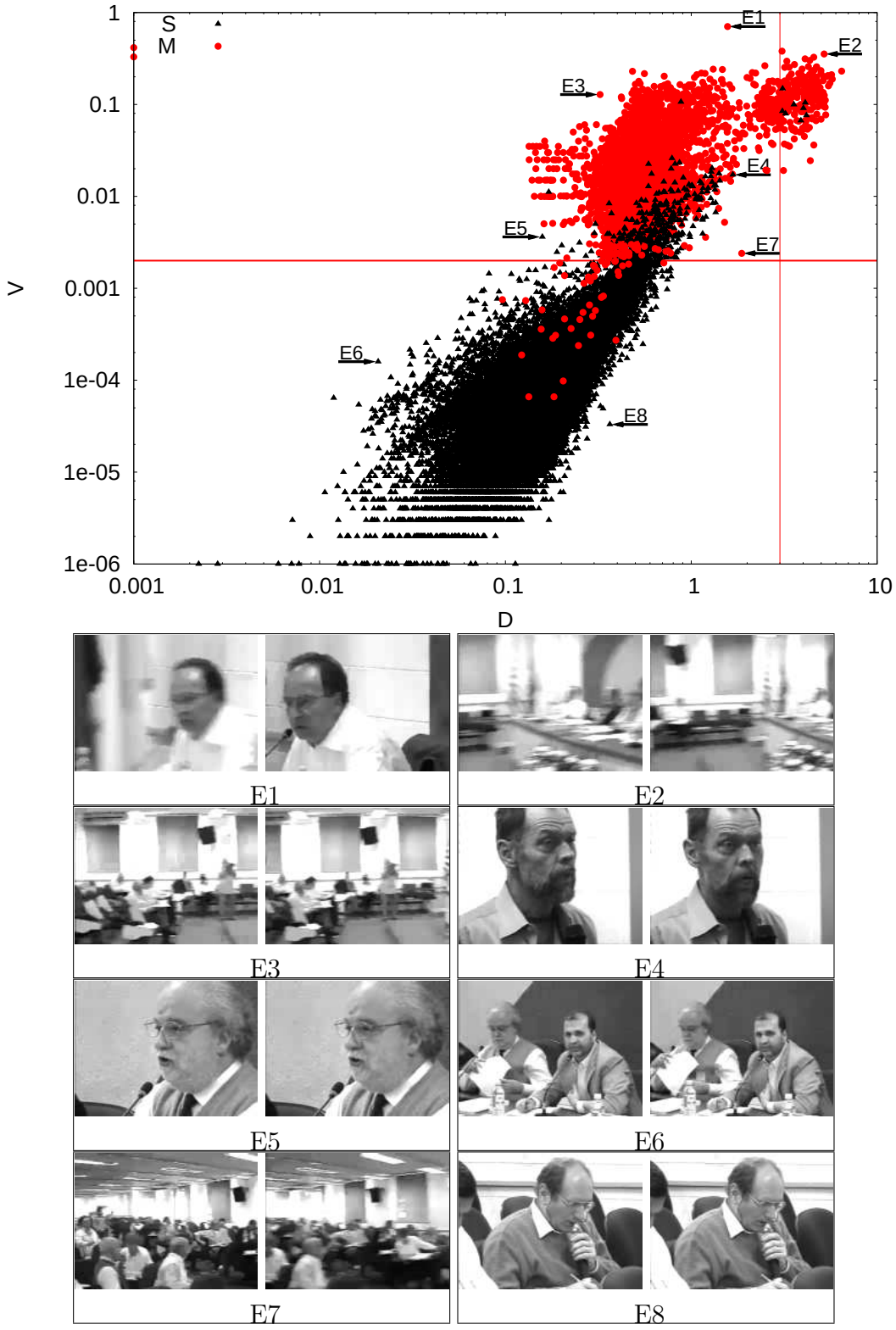


Figura 6.10: Outro exemplo, da discriminação de movimento com a utilização do algoritmo WOFF no vídeo **consu3**.

6.3.2 Discussão da discriminação automática

Observe que as duas classes M e S podem ser separadas facilmente com um simples limiar em V. Ou seja, pares de quadros com $V \geq 0.002$ são quase sempre M, enquanto pares com $V \leq 0.002$ são quase sempre S. O limiar em V foi escolhido de forma a maximizar os acertos na separação entre as classes M e S.

Os gráficos 6.8 e 6.9 mostram que o resíduo D é muito pequeno nos vídeos **consu1** e **consu2**. Já o gráfico 6.10 mostra um resíduo bem maior no vídeo **consu3** devido a diversos movimentos muito rápidos de câmera. Alguns destes eventos são mostrados na figura 6.10 (eventos E1, E2 e E7).

Entretanto, valor de 0.002 para o limiar em V foi ajustado especificamente para os nove vídeos analisados. Embora o fato de que os vídeos utilizados nos experimentos sejam muito distintos entre si, pode acontecer que dependendo de características da cena (dimensão do ambiente, luz, mobília, multidão, etc), posicionamento da câmera, resolução do vídeo e codificação, taxa de quadros, trabalho de câmera do operador, um valor melhor para V possa ser encontrado. O propósito das figuras 6.8, 6.9 e 6.10 é ilustrar a robustez do procedimento iterativo de mínimos quadrados, especialmente sua imunidade a grande quantidade de movimento de cena tipicamente observada em vídeos de conferências (oradores sentados e audiência).

A discriminação baseada em um único limiar em V falha em alguns pares S (com quantidades substanciais de movimento de cena, por exemplo, uma pessoa andando através do campo de visão da câmera). Em tais casos, o procedimento iterativo de mínimos quadrados acaba rejeitando os vetores de dados do fundo, e interpreta o fluxo ótico de uma pessoa se movimentando como sendo devido a um movimento de câmera. Assim, esses pares têm V elevado e, portanto, acabam sendo classificados como M (tal como o par E3 da figura 6.9).

Observamos, entretanto, que muitos destes falsos positivos têm um fluxo residual significativo, de forma que podem ser descartados com um segundo limiar em D, ou seja, pares com $V \geq 0.002$ mas $D \geq 3$ são mais prováveis de serem S e, portanto podem ser classificados como tais. Uma explicação para este fato é que a imagem de uma pessoa andando está geralmente fora de foco, assim, o algoritmo de fluxo ótico tem dificuldade em achar as correspondências corretas. Além do mais, uma pessoa está freqüentemente se movendo de tal forma que seu fluxo ótico não pode ser aproximado adequadamente por uma combinação linear de fluxos canônicos r_x, r_y, r_z e r_f.

Outros pares de quadro, por exemplo, os eventos E2 e E3 da figura 6.8, E6 da figura 6.9,

e E2 da figura 6.10 possuem um movimento muito rápido de câmera que causa uma mudança muito grande nos quadros do vídeo e, portanto, também possuem resíduo D elevado. Esses eventos são classificados como não movimento devido a dificuldade em separá-los de eventos como grandes oclusões.

Os eventos E4 e E5 da figura 6.10 são classificados erroneamente como M devidos aos problemas citados na seção (6.2.1), onde é possível ver claramente a pobreza de textura no fundo da cena.

6.3.3 Desempenho do detector de movimento de câmera

O desempenho deste método de detecção de movimento de câmera pode ser quantificado através das métricas de *precisão* (pr) e *revocação* (rc), definadas como

$$pr = \frac{V_+}{V_+ + F_+} \quad rc = \frac{V_+}{V_+ + F_-} \tag{6.1}$$

onde V_+ é o número de positivos verdadeiros (pares de quadro da classe M que foram corretamente identificados como tais), F_+ é o número de positivos falsos (pares S identificados como pares M) e F_- são os negativos falsos (pares M identificados como S). Os valores destas métricas para os nove vídeos de teste estão apresentados na tabela 6.2. Os pares de quadro da classe S corretamente identificados não foram utilizados no cálculo da performance

Video	V_+	F_+	F_-	pr	rc
consu1	1263	97	45	0.97	0.93
consu2	637	51	8	0.99	0.93
consu3	4182	277	450	0.90	0.94
montanha1	115	4	0	1.00	0.97
península	49	1	0	1.00	0.98
montanha2	45	0	0	1.00	1.00
cidade	109	0	0	1.00	1.00
feira	122	0	0	1.00	1.00
fazenda	53	0	0	1.00	1.00

Tabela 6.2: Precisão e revocação do WOFF para os vídeos da tabela 6.1.

6.3.4 Efeito da densidade de amostragem do fluxo

O desempenho do algoritmo e seu custo computacional dependem do número de pontos de amostragem p_i. Este efeito é ilustrado na tabela 6.3. Observa-se que o aumento do número de pontos acima de 05×07, apesar de aumentar proporcionalmente o custo, traz poucas melhorias nas taxas de precisão e abrangência. A explicação decorre de que o movimento de câmera existente no vídeo **consu2** pode ser expresso com um número pequeno de pontos amostrados. No entanto, isso pode não ser verdade para qualquer tipo de vídeo.

Números de pontos n	pr	rc	tempo(s)
$05 \times 07 = 35$	0.95	0.93	0.04
$11 \times 08 = 88$	0.96	0.94	0.05
$16 \times 11 = 176$	0.99	0.93	0.06
$32 \times 22 = 704$	0.98	0.95	0.10
$63 \times 43 = 2709$	0.97	0.95	0.26

Tabela 6.3: Variação da precisão, revocação e tempo de execução do WOFF no vídeo **consu2** em função do número n de pontos de amostragem do fluxo. Os tempos de execução foram medidos em um computador com processador Intel pentium 4 de 3.0GHz com 2GB de memória.

Capítulo 7

Testes das etapas do algoritmo

Neste capítulo realizamos uma série de experimentos para demonstrar a necessidade dos principais passos do algoritmo WOFF. Especificamente, tentamos responder as seguintes questões:

1. É importante atribuir pesos distintos aos vetores no ajuste de mínimos quadrados?
2. É importante fazer ajuste iterativo de pesos (passos 4—15 do algoritmo 5.2)?
3. É importante atribuir como pesos iniciais os escores de confiabilidade κ_i devolvidos pelo algoritmo KLT?
4. É importante ajustar o escore inicial κ_i, em função do tamanho do vetor f_i (passos 1—3 do algoritmo 5.2)?

Para responder estas questões, testamos quatro versões modificadas do algoritmo WOFF omitindo ou não os referidos passos. Nas seções 7.1— 7.4 detalhamos e discutimos estes testes, e comparamos os resultados com os do algoritmo WOFF completo. Para melhor ilustrar qualitativamente o efeito de cada passo, escolhemos um par de quadros consecutivos onde a eliminação do passo em questão tem efeitos marcantes. Na seção 7.5 apresentamos uma comparação quantitativa entre estas quatro versões e o algoritmo completo, usando o conjunto de vídeos do CONSU para todas as versões.

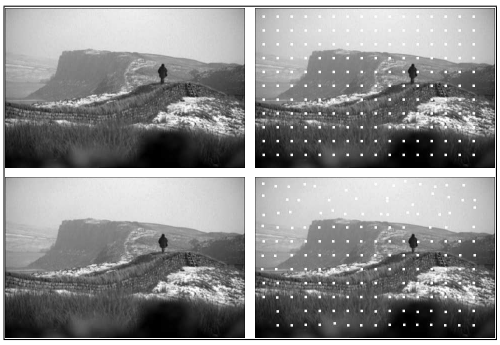
7.1 Teste 1: Importância dos pesos

A primeira questão é se vale a pena atribuir um peso distinto w_i a cada vetor de fluxo f_i. Ou seja, como é a performance do algoritmo se todos os fluxos tiverem o mesmo peso, digamos $w_i = 1$, durante toda a sua execução?

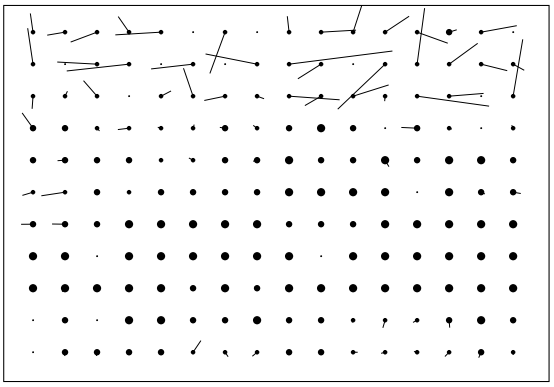
Para responder esta questão, testamos o algoritmo WOFF com os pesos iguais e constantes. Atribuímos escore $\kappa_i = 1$ para todo vetor f_i (exceto os que não são encontrados pelo algoritmo KLT que recebem $\kappa_i = 0$); eliminamos o ajuste inicial (passos 1—3) fazendo $w_i = \kappa_i$ para todos os vetores e eliminamos o ajuste iterativo dos pesos (passos 7—14).

A figura 7.1 mostra o resultado do algoritmo WOFF original, e a figura 7.2 o resultado da versão modificada. Em cada figura têm-se: (a) o par de quadros consecutivos analisado; (b) os vetores de fluxo f_i calculados pelo algoritmo KLT com os escores de confiabilidade κ_i; (c) os mesmos vetores de fluxo f_i com os pesos w_i após o ajuste de pesos (passos 1—3); (d) vetores de fluxo f_i, com os pesos w_i no final do algoritmo; (e) o fluxo $\tilde{f}$ ideal ajustado, amostrado nos pontos p_i; (f) os vetores do fluxo residual $d_i = |f_i - \tilde{f}_i|$. Para maior clareza, nas figuras (b), (c), (d) e (f), omitimos os vetores f_i que possuem pesos (κ_i ou $w_i = 0$) nulo.

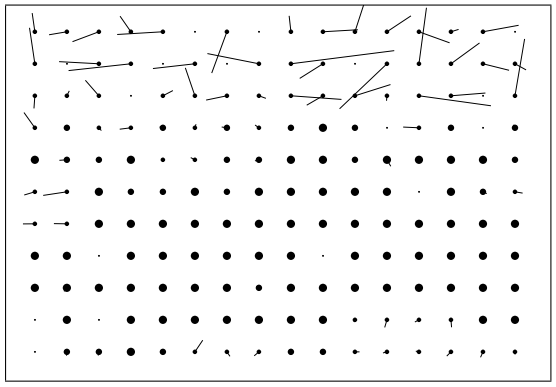
Na figura 7.1(d) notamos que o algoritmo WOFF completo elimina os vetores localizados na parte superior da imagem (no horizonte) cujos vetores f_i não são significativos devido à falta de textura. Na versão do algoritmo sem pesos, em constraste (figura 7.2), os vetores f_i no alto da imagem são usados no ajuste de mínimos quadrados, e acabam produzindo um fluxo ajustado $\tilde{f}$ não nulo — apesar de a câmera estar parada nesta cena.



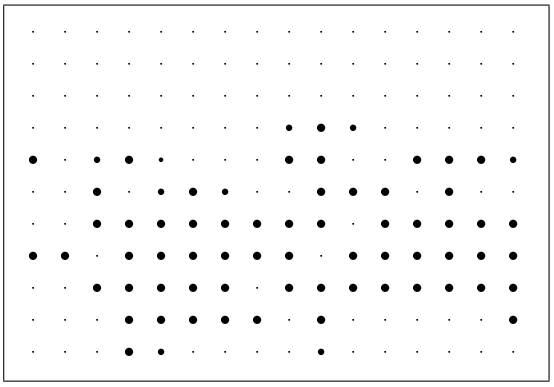
(a) Quadros



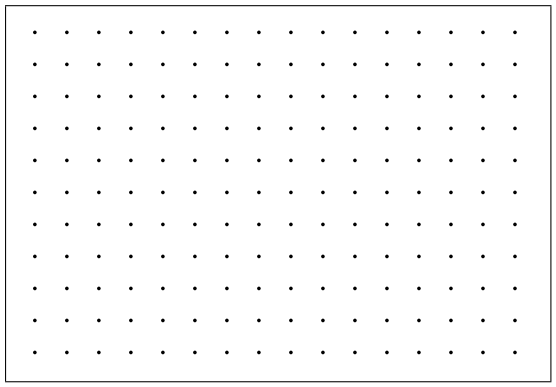
(b) fluxo ótico KLT com pesos κ



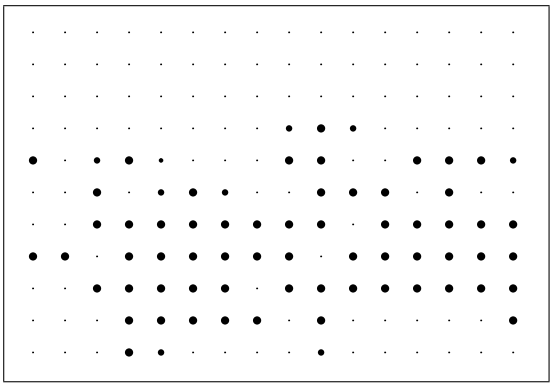
(c) fluxo ótico KLT com pesos w



(d) fluxo ótico com pesos finais

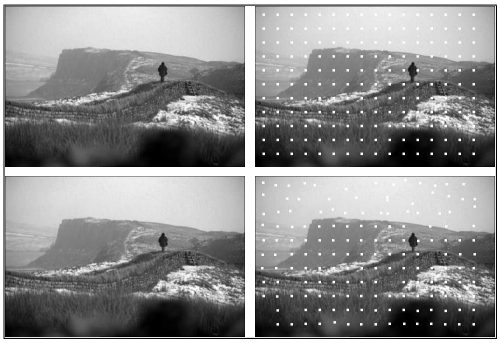


(e) fluxo ajustado $\tilde{f}$

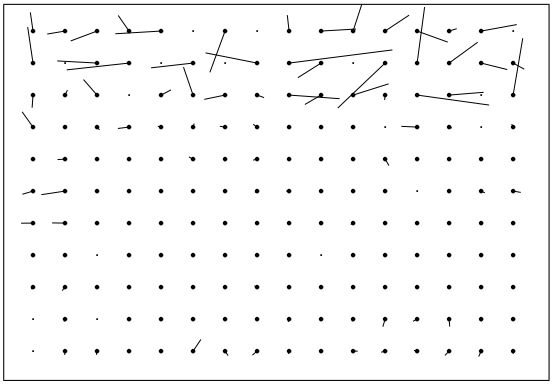


(f) resíduo d

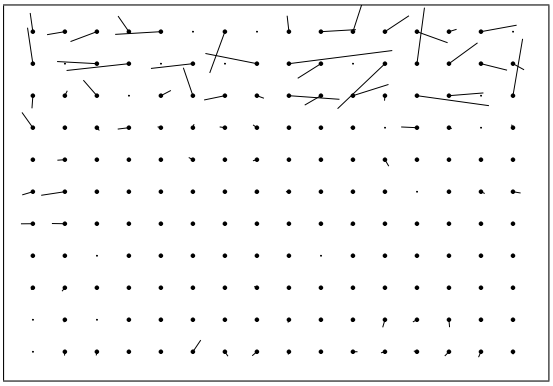
Figura 7.1: Resultado do algoritmo WOFF completo aplicado a dois quadros consecutivos do vídeo **montanha**.



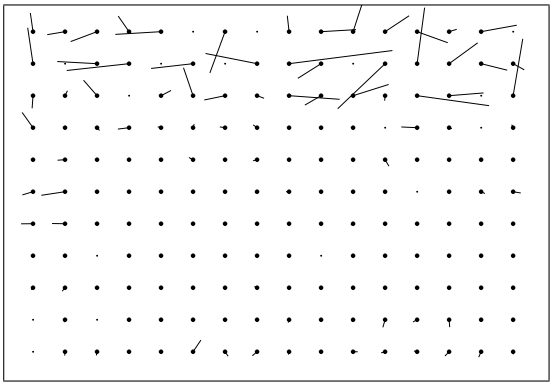
(a) Quadros



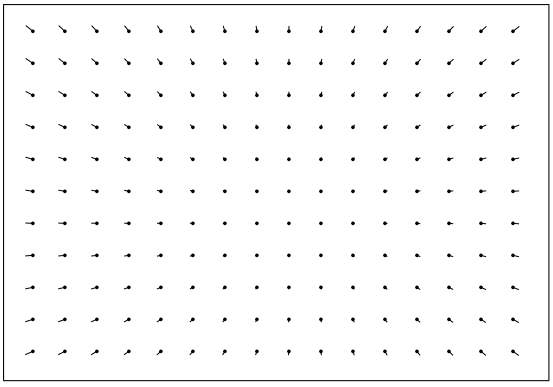
(b) fluxo ótico KLT com pesos κ



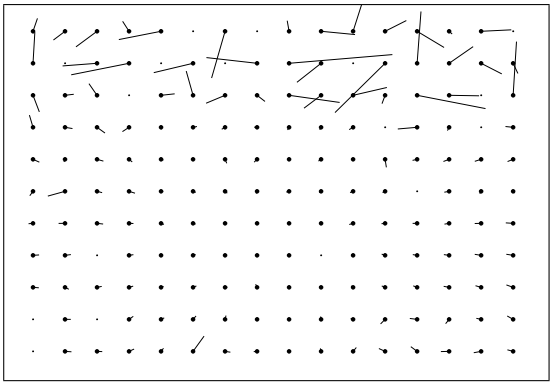
(c) fluxo ótico KLT com pesos w



(d) fluxo ótico com pesos finais



(e) fluxo ajustado $\tilde{f}$



(f) resíduo d

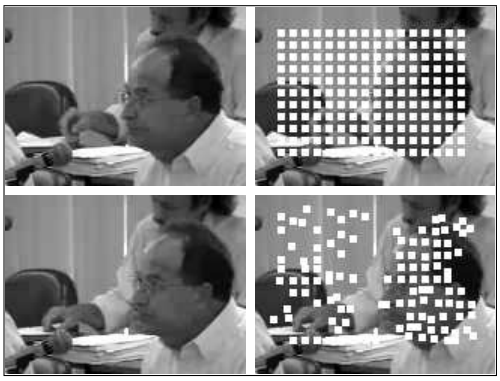
Figura 7.2: Resultados de uma versão do WOFF sem pesos iniciais, sem ajuste inicial dos pesos e sem o ajuste iterativo de pesos, para os mesmos quadros da figura 7.1.

7.2 Teste 2: Importância do ajuste iterativo

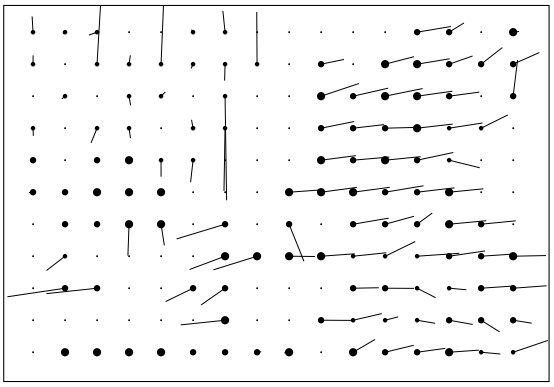
Tendo estabelecido que o uso de pesos w_i contribui para a qualidade dos resultados, investigaremos a seguir, o mérito do ajuste iterativo, isto é, será que o uso dos escores κ_i e o ajuste inicial são necessários para a obtenção de um fluxo $\tilde{f}_i$ correto?

Para responder esta questão, comparamos o algoritmo WOFF completo com uma versão modificada na qual o ajuste iterativo dos pesos (passos 7—14) foi suprimido. Os resultados são ilustrados nas figuras 7.3 e 7.4, respectivamente.

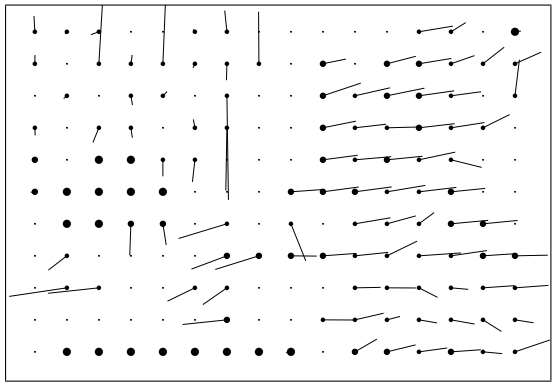
Pode-se observar que sem o ajuste iterativo os vetores, devido a pessoas em movimento, entram no ajuste de mínimos quadrados com pesos significativos, causando um erro no fluxo de movimento da câmera ajustado $\tilde{f}$.



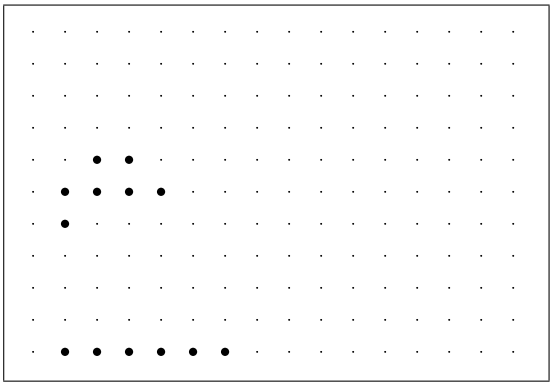
(a) Quadros



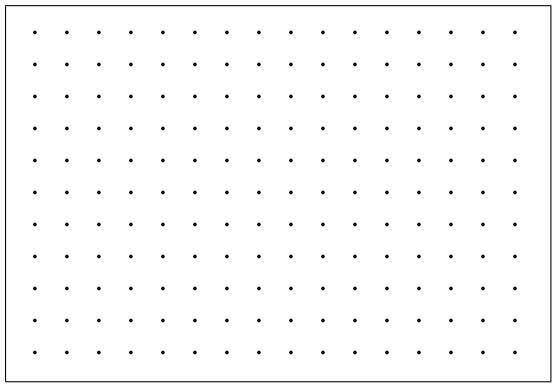
(b) fluxo ótico KLT com pesos κ



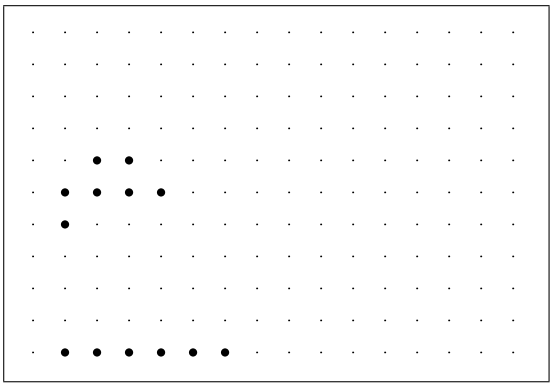
(c) fluxo ótico KLT com pesos w



(d) fluxo ótico com pesos finais

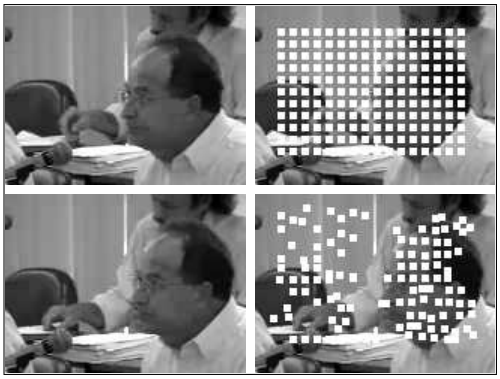


(e) fluxo ajustado $\tilde{f}$

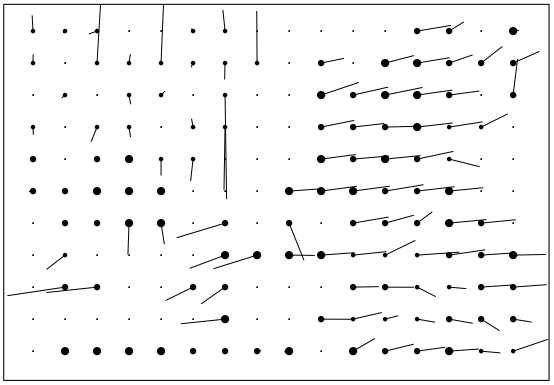


(f) resíduo d

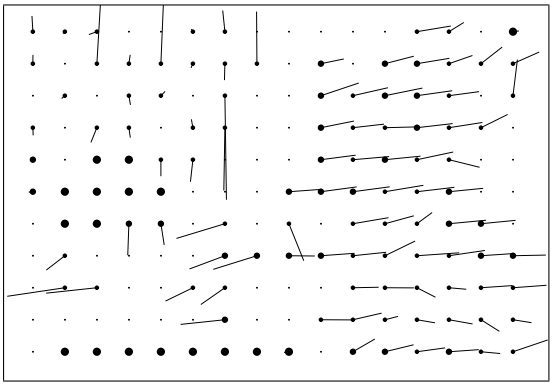
Figura 7.3: Resultado do algoritmo WOFF completo aplicado a dois quadros consecutivos do vídeo **consu2**.



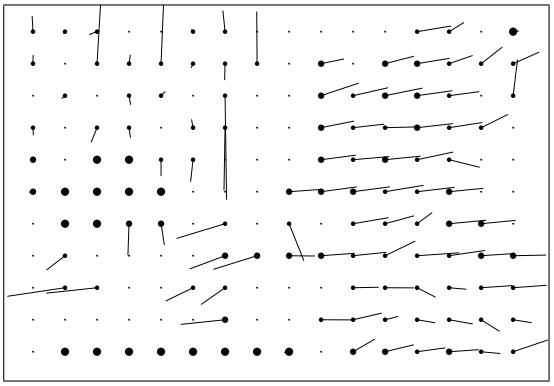
(a) Quadros



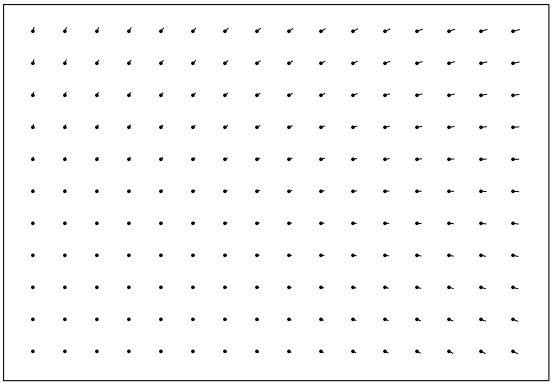
(b) fluxo ótico KLT com pesos κ



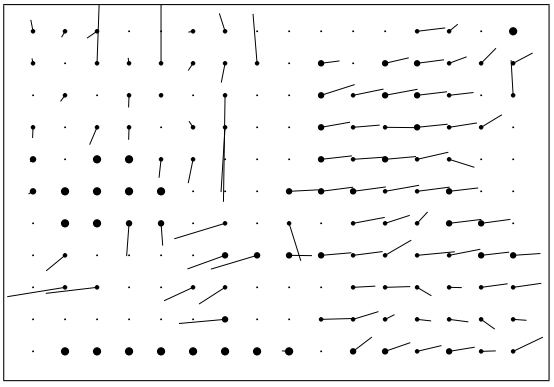
(c) fluxo ótico KLT com pesos w



(d) fluxo ótico com pesos finais



(e) fluxo ajustado $\tilde{f}$



(f) resíduo d

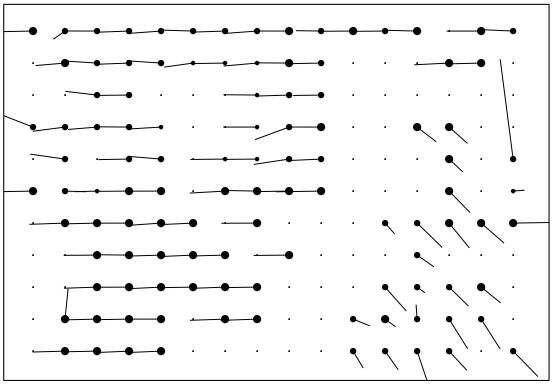
Figura 7.4: Resultados de uma versão do WOFF sem o ajuste iterativo de pesos, para os mesmos quadros da figura 7.3.

7.3 Teste 3: Importância do escore de confiabilidade

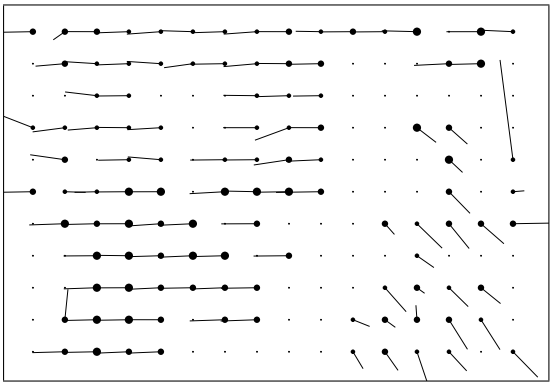
Considerando-se útil o ajuste iterativo dos pesos, cabe questionar se o uso dos escores de confiabilidade κ_i é necessário na inicialização dos pesos. Isto é, se considerarmos inicialmente todos os vetores rastreados pelo KLT com a mesma confiabilidade $\kappa_i = 1$ e efetuarmos apenas o ajuste inicial dos pesos w_i (passos 1—3) será que o ajuste iterativo continua convergindo para o mesmo fluxo $\tilde{f}$?

Para responder esta questão, testamos o algoritmo WOFF completo com os escores κ_i calculados pelo algoritmo KLT (figura 7.5), e com escores fixos $\kappa_i = 1$ para pontos rastreados e 0 para pontos não rastreados (figura 7.6). Observamos que, apesar do ajuste iterativo de pesos, a segunda versão converge para um fluxo $\tilde{f}$ bastante incorreto.

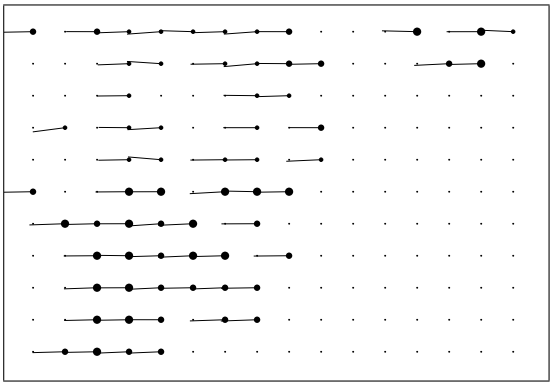
(a) Quadros



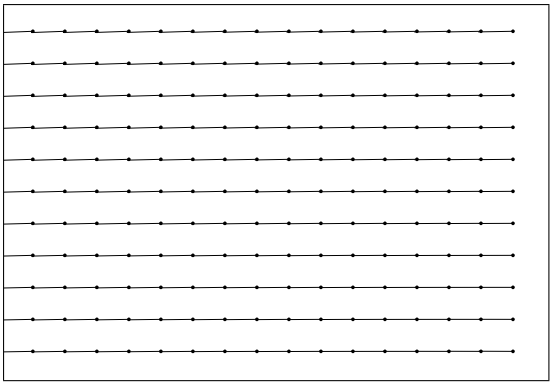
(b) fluxo ótico KLT com pesos κ



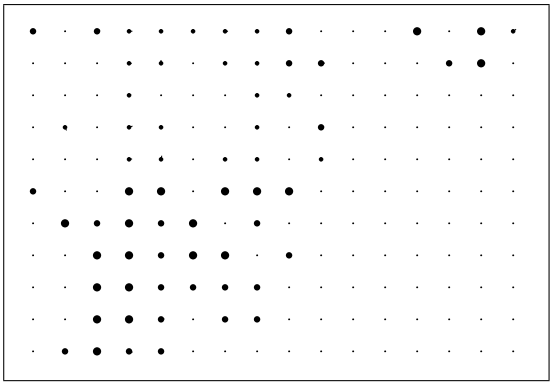
(c) fluxo ótico KLT com pesos w



(d) fluxo ótico com pesos finais



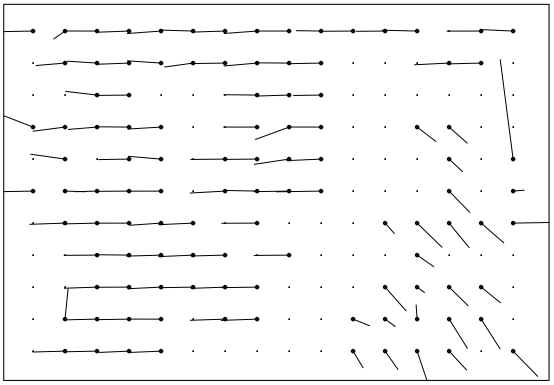
(e) fluxo ajustado $\tilde{f}$



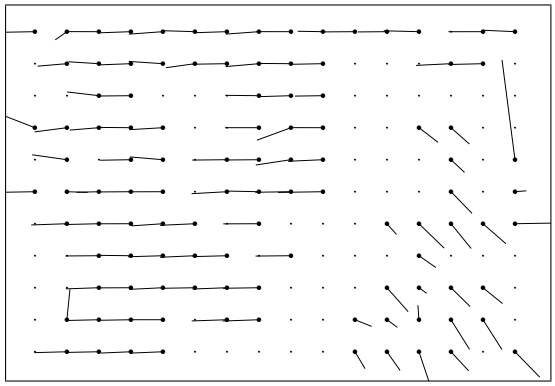
(f) resíduo d

Figura 7.5: Resultado do algoritmo WOFF completo aplicado a dois quadros consecutivos do vídeo **consu3**, com escores de confiabilidade κ_i fornecidos pelo algoritmo KLT.

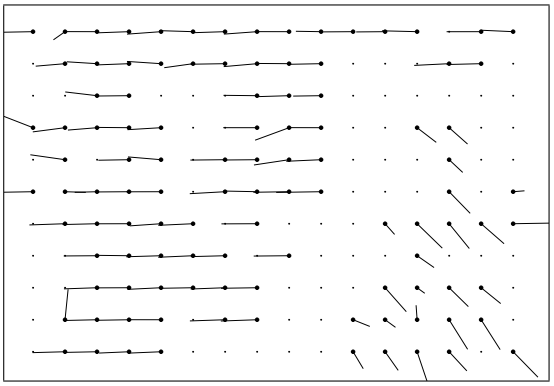
(a) Quadros



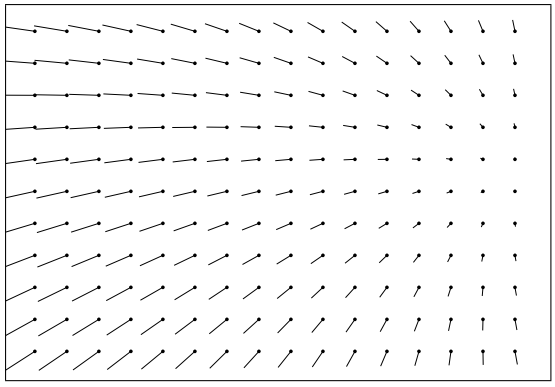
(b) fluxo ótico KLT com pesos κ



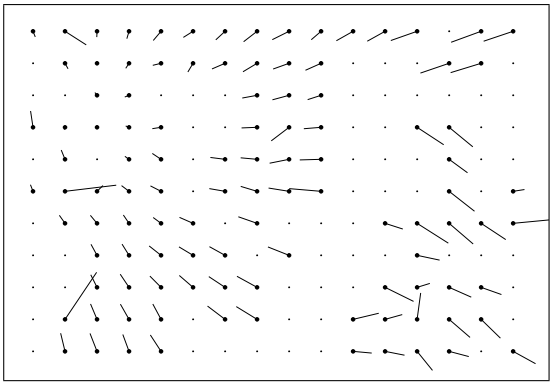
(c) fluxo ótico KLT com pesos w



(d) fluxo ótico com pesos finais



(e) fluxo ajustado $\tilde{f}$



(f) resíduo d

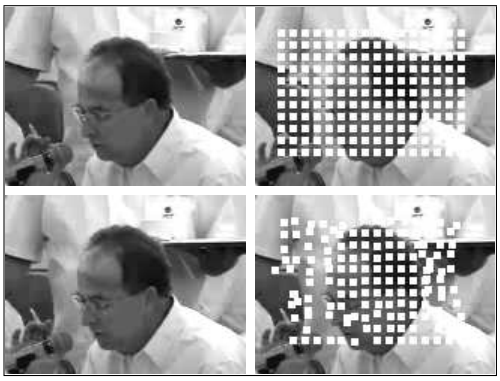
Figura 7.6: Resultados do algoritmo WOFF ignorando os escores de confiabilidade do algoritmo KLT, para os mesmos quadros da figura 7.5.

7.4 Teste 4: Importância do ajuste inicial de pesos

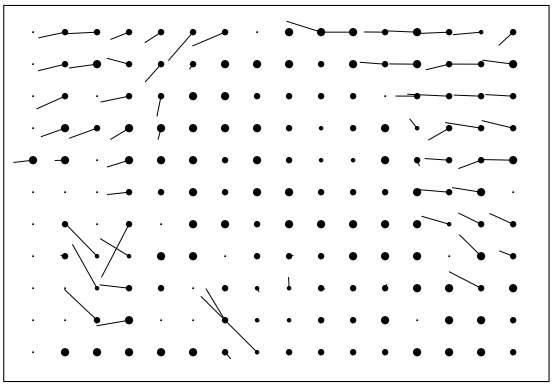
Finalmente, analisamos a importância do ajuste inicial de pesos w_i. Isto é, se os pesos iniciais dos vetores f_i forem os escores de confiabilidade κ_i retornados pelo algoritmo do KLT, sem o ajuste por comprimento (passos 1—3), será que o ajuste iterativo (passos 7—14) converge para o fluxo $\tilde{f}$ correto?

Para esta questão, testamos o algoritmo WOFF completo (figura 7.7), e uma versão modificada em que o ajuste inicial de pesos foi suprimido mantendo-se o ajuste iterativo (figura 7.8).

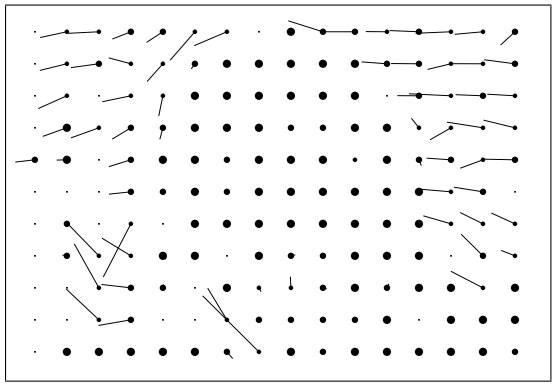
Observamos que neste exemplo, o algoritmo WOFF completo elimina os vetores devidos ao movimento da pessoa atrás do interlocutor, enquanto que a versão modificada não os elimina, o que produz um fluxo $\tilde{f}$ incorreto (figura 7.8 (f)).



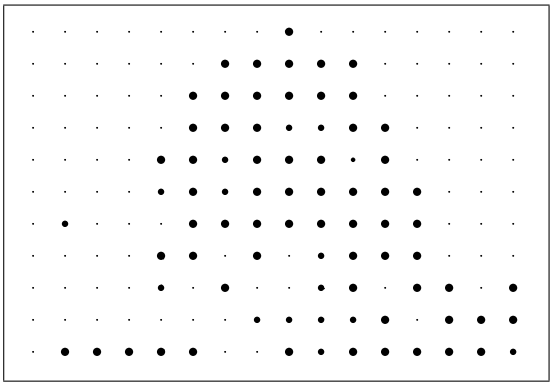
(a) Quadros



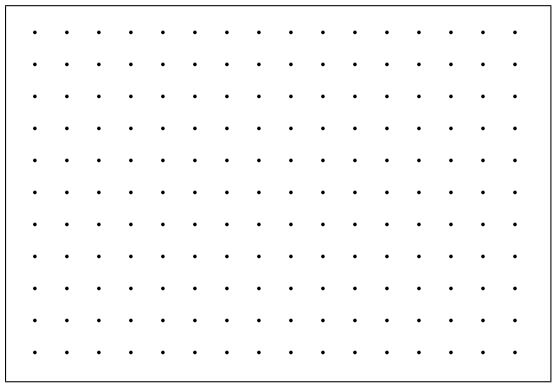
(b) fluxo ótico KLT com pesos κ



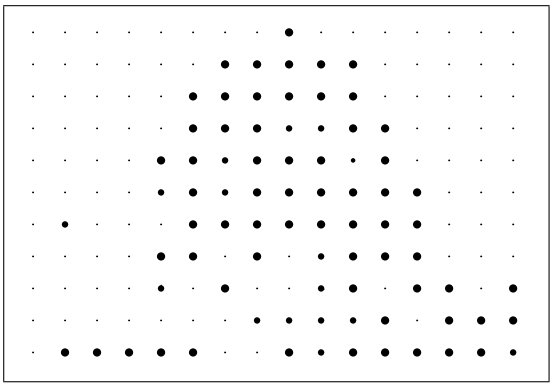
(c) fluxo ótico KLT com pesos w



(d) fluxo ótico com pesos finais

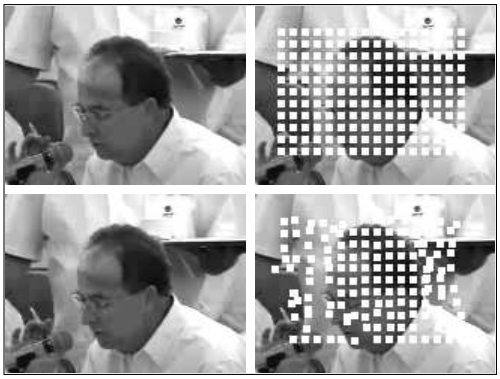


(e) fluxo ajustado $\tilde{f}$

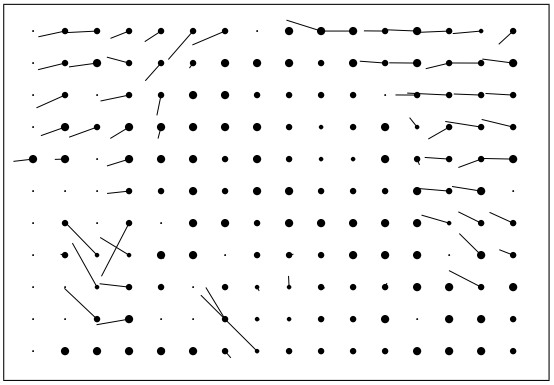


(f) resíduo d

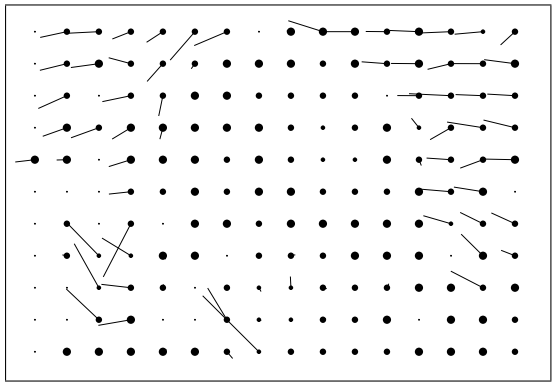
Figura 7.7: Resultado do algoritmo WOFF completo aplicado em dois quadros consecutivos do vídeo **consu1**.



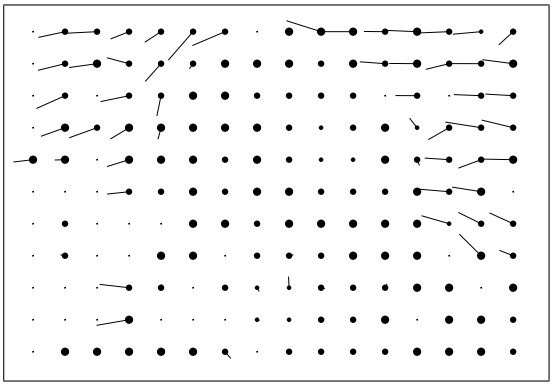
(a) Quadros



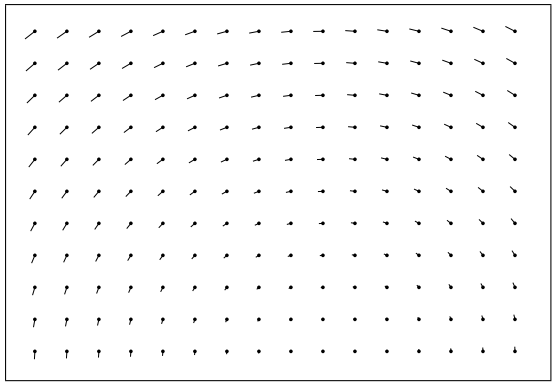
(b) fluxo ótico KLT com pesos κ



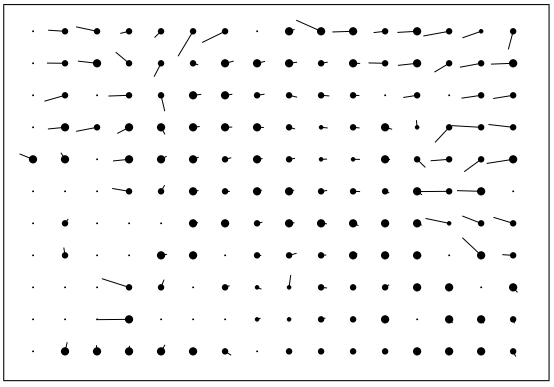
(c) fluxo ótico KLT com pesos w



(d) fluxo ótico com pesos finais



(e) fluxo ajustado $\tilde{f}$



(f) resíduo d

Figura 7.8: Resultados de uma versão do WOFF sem ajuste inicial dos pesos, para os mesmos quadros da figura 7.7.

7.5 Comparação quantitativa

Na tabela (7.1) são mostradas as taxas de precisão (pr) e abrangência (rc) da classificação dos vídeos **consu1**, **consu2** e **consu3**, para o algorito WOFF completo e para as quatro versões descritas anteriormente.

	WOFF		Teste1		Teste2		Teste3		Teste4	
Vídeo	pr	rc	pr	rc	pr	rc	pr	rc	pr	rc
consu1	0.97	0.93	0.22	0.94	0.90	0.94	0.58	0.97	0.97	0.83
consu2	0.99	0.93	0.28	0.95	0.98	0.94	0.67	0.98	0.99	0.88
consu3	0.90	0.94	0.30	0.90	0.86	0.93	0.42	0.96	0.94	0.85

Tabela 7.1: Performance para o algoritmo WOFF e suas quatro variantes.

Comparando o algoritmo completo (WOFF) com as versões modificadas (Teste1—Teste4), podemos concluir que as três etapas consideradas (uso de escores de confiabilidade, ajuste de pesos e ajuste iterativo de pesos) são importantes para a qualidade da classificação.

Em particular, a coluna Teste 1 mostra que a utilização de pesos aumenta a precisão, em média, de 30% para 90%. A coluna Teste 3 mostra que o uso dos escores κ_i são importantes para a taxa de precisão, e a coluna Teste 4 mostra que o ajuste inicial dos pesos é importante tanto para a taxa de precisão quanto para a abrangência.

Capítulo 8

Conclusões e Trabalhos Futuros

Nesta dissertação foi descrito um algoritmo original (WOFF), baseado na análise de fluxo ótico ponderado, para detectar movimentos de câmera (tilt, pan, roll e zoom). Este algoritmo realiza uma análise robusta da quantidade de movimento existente entre dois quadros consecutivos quaisquer e cujos pesos que resultam deste processo de ponderação permitem, como sub-produto, separar a imagem em "fundo" (estático) e "objetos" (em movimento). Nosso método utiliza toda a informação contida no fluxo de entrada, e é portanto mais robusto em relação a ruído e permite objetos em movimento na cena, desde que não dominem a área da imagem. Este também permite detectar movimentos combinados de câmera (por exemplo, pan e tilt simultâneos).

O algoritmo foi validado empiricamente em nove vídeos reais (três vídeos de reuniões de Câmaras do Conselho Universitário (CONSU) da UNICAMP, e seis vídeos obtidos da internet com gravações de ambientes externos). Nos testes realizados, o algoritmo conseguiu discriminar entre câmera em movimento e câmera estática com taxas de acerto acima de 90% — mesmo com a presença de movimento significativo de objetos e pessoas na cena.

Em um outro teste, realizamos uma série de experimentos para demonstrar a necessidade dos principais passos do algoritmo WOFF (importância dos pesos, do escore de confiabilidade, do ajuste inicial dos pesos, e do ajuste iterativo) para a segmentação de movimento de câmera. Concluímos que a utilização de pesos aumenta a precisão, em média, de 30% para 90%.

Relatamos, ainda, algoritmos importantes para o cálculo de fluxo ótico, assim como um estudo sobre fluxos canônicos devido a movimento de câmera e uma breve descrição de outros algoritmos para detecção de movimento de câmera.

Um projeto interessante é estender o algoritmo WOFF para situações com câmera em movimento (T_x, T_y ou T_z diferente de zero). O objetivo é diminuir o custo computacional das abordagens existentes para detectar esse tipo de movimento, e permitir a segmentação mesmo com movimento de objetos na cena.

Outro objetivo na mesma linha é utilizar a informação de fluxo direcional (quantidades de tilt, pan, roll e zoom retornadas pelo algoritmo) na composição de quadros do vídeo em imagens panorâmicas do ambiente e para realizar buscas de seqüências de imagens em banco de dados (busca por conteúdo).

Referências Bibliográficas

[1] Minerva Yeung, Boon-Lock Yeo, and Bede Liu. Extracting story units from long programs for video browsing and navigation. pages 296–305, june 1996.

[2] Arun Hampapur, Terry Weymouth, and Ramesh Jain. Digital video segmentation. In *Proceedings of the second ACM international conference on Multimedia*, pages 357–364, New York, USA, 1994. ACM Press.

[3] Irena Koprinska and Sergio Carrato. Temporal video segmentation: A survey. *Processing Image Communication, Elsevier Science*, 16(5):477–500, january 2001.

[4] Jae-Gon Kim, Hyun S. Chang, Jinwoong Kim, and Hyung-Myung Kim. Efficient camera motion characterization for MPEG video indexing. 2(30):1171–1174, august 2000.

[5] Yoshinobu Tonomura, Akihito Akutsu, Yukinobu Taniguchi, and Gen Suzuki. Structured video computing. *IEEE MultiMedia*, 1(3):34–43, 1994.

[6] Patrick Bouthemy, Marc Gelgon, and Fabrice Ganansia. A unified approach to shot change detection and camera motion characterization. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(7):1030–1044, 1999.

[7] Rodrigo Minetto, Neucimar J. Leite, and Jorge Stolfi. Reliable detection of camera motion based on weighted optical flow fitting. In *Proceedings of International Conference on Computer Vision Theory and Applications*, volume IV/MTSV, pages 435–440, Barcelona, Spain, march 2007.

[8] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, 1981.

[9] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, april 1991.

[10] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 674–679, april 1981.

[11] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1986.

[12] Sangkeun Lee and Monson H. Hayes III. Real-time camera motion classification for content-based indexing and retrieval using templates. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 4:3664–3667, 2002.

[13] Thomas M. Lehmann, Claudia Gonner, and Klaus Spitzer. Survey: interpolation methods in medical image processing. *IEEE Transactions on Medical Imaging*, 18(11):1049–1075, november 1999.

[14] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing, Boston, MA, USA, 2001.

[15] Shahriar Negahdaripour. Revised definition of optical flow: Integration of radiometric and geometric cues for dynamic scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(9):961–979, 1998.

[16] Stephen M. Smith and John M. Brady. Asset-2: Real-time motion segmentation and shape tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(8):814–820, 1995.

[17] Shinya Yamamoto, Yasushi Mae, Yoshiaki Shirai, and Jun Miura. Realtime multiple object tracking based on optical flows. *IEEE International Conference on Robotics and Automation*, 3:2328–2333, may 1995.

[18] John Watkinson. *MPEG-2*. Butterworth-Heinemann, Newton, USA, 1999.

[19] Ramesh Jain, Rangachar Kasturi, and Brian G. Schunck. *Machine vision*. McGraw-Hill, Inc., New York, USA, 1995.

[20] Mário Sarcinelli-Filho, Hans J. A. Schneebeli, Eliete M. O. Caldeira, Ricardo Carelli, Oscar H. Nasisi, and Carlos Sória. On the use of optical flow in mobile robot navigation: The search for a suitable algorithm. *Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems*, 2:922–925, august 2000.

[21] Barton L. Anderson and Pawan Sinha. Reciprocal interactions between occlusion and motion computations. volume 94, pages 3477–3480, april 1997.

[22] Hans-Hellmut Nagel. On the estimation of optical flow: relations between different approaches and some new results. *Artificial Intelligence*, 33(3):298–324, 1987.

[23] Sergio Uras, Federico Girosi, Alessandro Verri, and Vincent Torre. A computational approach to motion perception. *Biological Cybernetics*, 60(2):79–87, 1988.

[24] John L. Barron, David J. Fleet, and Steven S. Beauchemin. Performance of optical flow techniques. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 236–242, june 1992.

[25] Fabrice Heitz and Patrick Bouthemy. Multimodal estimation of discontinuous optical flow using markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1217–1232, december 1993.

[26] Bruce D. Lucas. *Generalized image matching by the method of differences*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, USA, july 1984.

[27] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, june 1994.

[28] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238, New York, USA, 1995. ACM Press.

[29] Anil K. Jain. *Fundamentals of digital image processing*. Prentice-Hall, Inc., Upper Saddle River, USA, 1989.

[30] Azriel Rosenfeld and Avinash C. Kak. *Digital Picture Processing*, volume 2. Academic Press, Inc., Orlando, FL, USA, 1982.

[31] Didier Le Gall. MPEG: a video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, 1991.

[32] Vikrant Kobla, David S. Doermann, King-Ip Lin, and Christos Faloutsos. Compressed-Domain video indexing techniques using DCT and motion vector information in MPEG video. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 200–211, 1997.

[33] Peter M. Huhn. Camera Motion Estimation Using Feature Points in MPEG Compressed Domain. *IEEE International Conference on Image Processing*, 3:596–599, september 2000.

[34] JungHwan Oh, Praveen Sankuratri, and Wallapak Tavanapong. Efficient measuring of various motions in mpeg videos. In *9th IASTED International Conference on Signal and Image Processing (SIP)*, pages 217–222, 2003.

[35] Ralph Ewerth, Martin Schwalb, Paul Tessmanny, and Bernd Freisleben. Estimation of arbitrary camera motion in MPEG videos. *17th International Conference on Pattern Recognition*, 1:512–515, 2004.

[36] James D. Foley and Andries Van Dam. *Fundamentals of interactive computer graphics*. Addison-Wesley Longman Publishing, Boston, USA, 1982.

[37] Hugh C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Royal Society of London Proceedings Series B*, 208:385–397, july 1980.

[38] Mandyam V. Srinivasan, Sventha Venkatesh, and Robin Hosie. Qualitative estimation of camera motion parameters from video sequences. *Pattern Recognition*, 30(4):593–606, 1997.

[39] Câmara de Ensino, Pesquisa e Extensão (CEPE), Sessão Ordinária 190. Site acessado no dia 13/06/2007, Abril 2005. `http://www.cameraweb.rei.unicamp.br/videos_consu_cad_cepe/cepe/2005/cepe-05042005/cepe-05042005.rm`.

[40] Câmara de Ensino, Pesquisa e Extensão (CEPE), Sessão Ordinária 179. Site acessado no dia 13/06/2007, Abril 2004. `http://www.cameraweb.rei.unicamp.br/videos_consu_cad_cepe/cepe/2004/cepe-06042004/cepe-06042004.rm`.

[41] Câmara de Ensino, Pesquisa e Extensão (CEPE), Sessão Ordinária 206. Site acessado no dia 14/06/2007, Setembro 2006. `http://www.cameraweb.rei.unicamp.br/videos_consu_cad_cepe/cepe/2006/cepe-05092006/cepe-05092006.rm`.

[42] Clip 180-39. Homem andando na montanha. Site acessado no dia 13/06/2007. `http://creative.gettyimages.com/source/frontdoor/defaultfilm.aspx`.

[43] Clip 457-23. Barco pesqueiro em um canal seguido de uma vista da península. Site acessado no dia 13/06/2007. `http://creative.gettyimages.com/source/frontdoor/defaultfilm.aspx`.

[44] Clip 478-10. Campo com a montanha Teranaki ao fundo. Nova Zelândia. Site acessado no dia 13/06/2007. `http://creative.gettyimages.com/source/frontdoor/defaultfilm.aspx`.

[45] Clip 665-86. Construções e telhados de uma cidade antiga de Praga, República Tcheca. Site acessado no dia 13/06/2007. `http://creative.gettyimages.com/source/frontdoor/defaultfilm.aspx`.

[46] Clip 545-9. Feira de rua em Berlim, Alemanha. Site acessado no dia 13/06/2007. `http://creative.gettyimages.com/source/frontdoor/defaultfilm.aspx`.

[47] Clip 628-11. Campo cultivado de uma fazenda e silos. Site acessado no dia 13/06/2007. `http://creative.gettyimages.com/source/frontdoor/defaultfilm.aspx`.

[48] Real Player. Site acessado no dia 30/06/2007. `http://brazil.real.com/player/`.

[49] FFmpeg. Site acessado no dia 30/06/2007. `http://ffmpeg.mplayerhq.hu/`.