Optical Flow Estimation in the Presence of Fast or Discontinuous Motion

by

Yan Niu

Thesis submitted for the degree of

Doctor of Philosophy

in

Computer Science and Mathematics
University of Adelaide

2010

ADELAIDE
UNIVERSITY
AUSTRALIA

Contents

Abstract

This thesis focuses on the computation of optical flow, i.e., the motion perceived from a sequence of gradually changing images, as an estimate for the 2D velocity of the scene. Due to the large variety and high complexity of the motion types existing in practice, motion recovery requires the estimation process to be highly adaptive. This thesis investigates how to select and combine the reasoning rules, namely the optical flow constraints, according to the type of motion information detected. Moreover, the thesis extends optical flow computation to fast rotation, an important, frequent and challenging motion type, which has not been addressed much in the literature.

The thesis starts by proposing various measures, based on theory as well as heuristics, for motion inconsistency detection. This facilitates selecting only the optical flow constraints that are valid for each pixel. While this selection benefits pixels affected by inconsistent motion, the combination of different constraints also enhances flow recovery for pixels that have consistent motion.

Two frameworks are designed for the combination of flow constraints. One utilizes motion segmentation; and the other is close in spirit to Expectation-Maximization. Within these frameworks, new constraints are formulated and tested. Furthermore, the adaptive reasoning is generalized from translational motion to motion that includes fast rotation. The key concept that enables this generalization is the use of intrinsic directions in differential geometry.

Experimental results on a variety of benchmark sequences have demonstrated the ability of the proposed methods to improve the performance of existing techniques in several situations, including strong motion discontinuities and fast rotational motion.

Statement of Originality

This work contains no material that has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published written by another person, except where due reference has been made in the text.

I give consent to this copy of the thesis, when deposited in the University Library, being available for loan, photocopying and dissemination through the library digital thesis collection.

Signed

Date

Acknowledgments

In Adelaide, I have had 3 very happy years in my life. I have concentrated on my study well; I have improved my research skills; I have learned a lot about Computer Vision and Graphics; And I have seen more of the world; I have made many good friends from different countries; I know I could not have had them done just by myself. Therefore I am more than grateful to those people who have brought the happiness to my life.

One happiest thing here is to work with my supervisor Dr. Anthony Dick. I believe Anthony deserves the world's sincerest thanks. If my Ph.D study is an exploration of a terrain, Anthony is the one who leads my way through the journey, giving me maps, telescopes and the compass. He watches my steps, lifts me up and encourages me to keep going. He constructively gives me the confidence to do things that I was scared of. Following him, the exploration is very pleasant and joyful. It saddens me to think that sooner or later Anthony won't teach me any more.

And I sincerely appreciate my supervisor Professor Mike Brooks for his guidance, support, giving me chances and being warm to me, which makes me feel home and protected when I pursue my study as a foreigner.

Truly and really I am very proud to be Anthony and Mike's student.

Many thanks to the head of school Professor David Suter for always being nice to me; and our administrative manager Ms. Tracey Young for looking after me; and Ms. Sharon Liersch and Kathy Cooper for their kind assistance to make things easier. And I am very grateful to Dr. Chris Madden for doing the proof reading of my thesis - I know it is not fun. Thanks to Professor Wojciech Chojnacki, Professor Garry Newsam and Dr. Anders Eriksson for discussing optical flow with me and teaching me mathematics. Thanks to my colleague and close friend Alex Cichowski for patiently and promptly rescuing me from debugging every time when I need help.

Sincere thanks to the Australian IPRS scholarship, without which I would not have been able to study here with so much fun. I am also grateful for the Horald Scholarship and Cowan scholarship from Kathleen Lumley College and the conference funding, which allows me to see more and learn more. Thanks to Kathleen Lumley College, a wonderful place where I enjoy the leisure time with so many friends and have learned

a lot from them. Especially I am thankful to the master Felix, sectary Allyson and chef John.

I appreciate the university security staff members for always being so nice to me. They are Kym, Francisca, Tony, Bruce, Jeff and David. When I work at uni late at night, they even walk me back home. I still remember the Easter Eggs that David secretly left at my office table. I still keep the warping papers. I am also thankful to the shuttle bus drivers Bob and Michael, who drive me home warmly and safely when I work after office hours and remember my Birthday.

Thanks to Professor Wang ZhengXuan, my close friends Dr. Wang Xin and Dr. Guo XiaoXin back home, who help me a lot with the jobs I left in China so to let me concentrate better on my study overseas. Thanks to my closest friend Ms Yao ZhiYun, who has given me many tips about daily life from Japan.

And I also hope to take the chance to thank my International Bridging Program lecturer Michelle Picard and the Manager of the International Student Center Patricia Anderson.

I probably have forgotten to mention some people who helped me here and there at the time of writing. I am full of gratitude to those who have said “leave this to me for now” or “don’t worry about that” or “I will be there to have a look”...; and those who made me feel the world is gentle.

Most of all, I thank my parents. To me, their love, care, support, encouragement, understanding, patience and confidence in me are the whole world. Although research is rewarding, only being loved and supported during research allows me to truly enjoy what I am doing. Baba and Mama, I love You!

I dedicate this thesis to my parents

Bibliography

Niu, Y., Dick, A. & Brooks, M. (2009). A new inconsistency measure for linear systems and two applications in motion analysis, *International Conference on Image and Vision Computing New Zealand.* **Receiver of Best Paper Award**.

Niu, Y., Dick, A. & Brooks, M. (2008). A new combination of local and global constraints for optical flow computation, *International Conference on Image and Vision Computing New Zealand.*.

Niu, Y., Dick, A. & Brooks, M. (2007). Discontinuity-preserving optical flow computation by a dynamic overdetermined system, *Digital Image Computing: Techniques and Applications*, Australia.

List of Figures

List of Tables

Notations

Image Intensity and Geometry

X,Y,Z	the world coordinate of a 3D point
x, y	spatial coordinates in the image plane
t	temporal coordinate in an image sequence
$E(x, y, t)$	image intensity function
$\vec{d}$	a general direction or the isophote (edge) direction, depending on the context
$\vec{n}$	the edge normal direction
$\mathcal{N}()$	the set of neighbours of a pixel
k	the kth pixel in a local patch, unless otherwise specified
$a^{(k)}$	the "a" of the kth pixel in the local patch
$a^{(c)}$	the "a" of the patch center
u	the horizontal component of the flow vector
v	the vertical component of the flow vector
$\vec{v}$	the flow vector, i.e, $\vec{v} = \begin{bmatrix} u & v \end{bmatrix}^T$
$\vec{V}$	the homogeneous flow vector, i.e, $\vec{V} = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$

Derivatives

E_x	first order partial derivatives of $E(x, y, t)$ with respect to variable x
E_{xx}	second order partial derivatives of $E(x, y, t)$ with respect to variable x
$\dot{E}$	total temporal derivative when compact notation is needed, $\dot{E} = \frac{dE}{dt}$
∇	spatial gradient vector
∇_3	spatial-temporal gradient vector
∇_d	gradient vector in an oriented coordinate system
Δ	Laplacian, $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$
$\mathrm{div}(\begin{bmatrix} a_1 \\ a_2 \end{bmatrix})$	divergence $\mathrm{div}(\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}) = \frac{\partial}{\partial x}(a_1) + \frac{\partial}{\partial y}(a_2)$
$\mathrm{div}_d(\begin{bmatrix} a_1 \\ a_2 \end{bmatrix})$	directional divergence $\mathrm{div}_d(\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}) = \frac{\partial}{\partial \vec{d}}(a_1) + \frac{\partial}{\partial \vec{n}}(a_2)$
$\partial_{\vec{d}}$	directional derivative in direction $\vec{d}$, i.e., $\frac{\partial}{\partial \vec{d}}$

H_{2D}	the spatial Hessian matrix $\begin{bmatrix} E_{xx} & E_{xy} \\ E_{xy} & E_{yy} \end{bmatrix}$
H_{3D}	the spatio-temporal Hessian matrix $\begin{bmatrix} E_{xx} & E_{xy} & E_{xt} \\ E_{xy} & E_{yy} & E_{yt} \\ E_{xt} & E_{yt} & E_{tt} \end{bmatrix}$
S_{2D}	the spatial Structure tensor $\begin{bmatrix} \sum E_x^2 & \sum E_x E_y \\ \sum E_x E_y & \sum E_y^2 \end{bmatrix}$
S_{3D}	the spatio-temporal Structure tensor $\begin{bmatrix} \sum E_x^2 & \sum E_x E_y & \sum E_x E_t \\ \sum E_x E_y & \sum E_y^2 & \sum E_y E_t \\ \sum E_x E_t & \sum E_y E_t & \sum E_t E_t \end{bmatrix}$

Vector and Matrix Operation

$\underline{0}$	a matrix whose elements are all zeros
$\vec{a}$	"a" is a vector
$\vec{X}$	the motion parameter vector in the local system $A\vec{X} = \vec{b}$
$\|\|\|_1$	the l_1 norm of a vector
$\|\|\|_2$	the l_2 norm of a vector
$[]^T$	the transpose of a vector or a matrix
$\left\langle \vec{a}, \vec{b} \right\rangle$	inner product of vector $\vec{a}$ and $\vec{b}$
$A * B$	the convolution of A and B
A^+	the pseudo-inverse of matrix A
$N(A)$	the null space of matrix A
$col()$	the column space
$span(\vec{a}_1, \cdots, \vec{a}_n)$	the space spanned by basis vectors $\vec{a}_1, \cdots, \vec{a}_n$
$\perp$	orthogonal
G_σ	Gaussian smoothing kernel of standard deviation σ
$K_{\frac{\partial}{\partial \vec{d}}}$	directional differencing filter
$A \sim B$	matrix A can be transformed to B by elementary matrix operations

Miscellaneous

$\mathfrak{E}$	the error functional
$\alpha_1, \cdots, \alpha_6$	affine motion model parameters
ϵ	a small positive number to prevent the denominator from being 0
λ	Lagrange multiplier
λ_i	the ith eigenvalue or singular value of a matrix
ω	weight
τ	index of the iteration stage, unless otherwise specified
ζ,η	the axes of the locally oriented coordinate frame
T	threshold, unless otherwise specified
δa	the refinement or increase of a in an iteration process
R	the set of real numbers
R^+	the set of positive real numbers

Chapter 1

Introduction

It is only with the heart that one can see rightly; what is essential is invisible to the eye.
Antoine de Saint-Exupry, The Little Prince

1.1 Problem Overview

1.1.1 Problem Statement

Consider a 3D scene that is viewed by a camera, where some or all of the scene is moving relative to the camera, as in Figure 1.1. The problem addressed in this thesis is to infer the 2D projection of that motion from a sequence of images by analysing variations in the intensity of pixels in each image.

Clearly, this is an ill-posed problem, as intensity changes between images may be due not only to motion but also lighting and other factors. In this thesis we assume that all perceived intensity changes are due to motion, and our aim is to find the motion field that best explains these changes. Such a motion field is called apparent motion or *optical flow*.

Even assuming motion is the only cause for brightness pattern change, recovering the cause from the observation is an *inverse problem*. The typical difficulty associated with an inverse problem is the ill-posedness (Hadamard 1902)[1]: the solution is often not unique. In terms of optical flow recovery, the intensity variation of a point is insufficient to indicate its motion uniquely. In fact, the core problem of optical flow recovery is to find out what constraint(s) can be integrated additionally and how to integrate,

[1]The term *ill-posedness* has different meanings in different context, such as the instability of the solution, the sensitivity to noise and the difficulty to find the optimal solution. Throughout this dissertation, we follow its original definition by (Hadamard 1902).

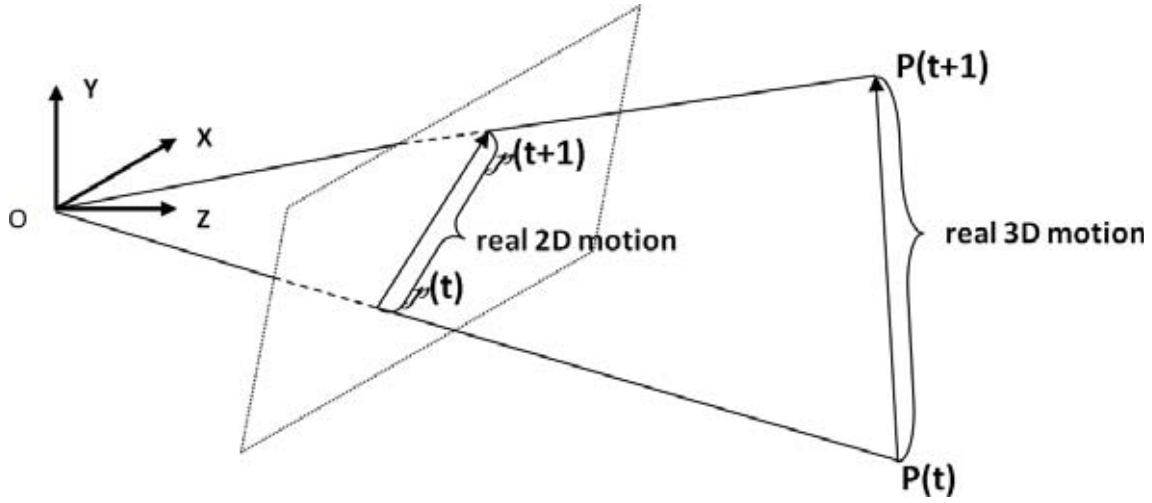


Figure 1.1. An illustration of the real motion of a scene point P at time t and $t+1$, in the world and in the image plane.

such that (i) the problem is solvable; (ii) the solution is unique; (iii) the solution estimates the 2D motion as accurately as possible.

1.1.2 Why Would We Want To Do This?

Although optical flow is not necessarily equivalent to the 2D projection of the 3D motion, it is a crucial, and most of the time, a reasonably satisfactory approximation of the actual 2D motion.

In many biological vision systems, recovering apparent motion seems to be adequate to accomplish low level tasks. For example, the behavioural research conducted on honeybees (Srinivasan 1992) shows that bees compute the apparent speed of the retinal images, i.e., optical flow, to gauge distance and guide steering. Similar insect optical flow computation has been simulated in robot navigation to estimate ego-motion and avoid obstacles, by computing the apparent shape change of objects. Optical flow is also an important part of the human visual system, as evidenced by our tendency to perceive motion when watching a sequence of static images; for example on television or using a "flipbook".

Moreover, if the recovered optical flow is an accurate estimation of the true 2D motion, more in-depth understanding of the scene is possible. For example, the reconstruction of the motion and structure can be obtained from the estimated 2D motion across frames. In the framework of video surveillance, the target object can be segmented and tracked based on the 2D motion estimated. In the field of medical image processing,

medical photos taken at different time instances can be reliably registered. Optical flow has also been widely applied to many other areas, such as video coding (mpeg4), face recognition and human action detection.

1.1.3 Why Is This A Difficult Problem?

It might be a common life experience that even if we are only given two frames of a scene with moving objects, we can easily extract salient features from both frames and quickly tell which feature in one frame moves to where in the other frame, if the features maintain their appearance in both frames. Next we can reason effortlessly about how other parts of the scene have to move to make physical sense. The "reasoning rules" and the "sense" come from our prior experience accumulated in life.

The process seems straightforward; however, it is difficult to have a machine vision algorithm accomplish a similar task. Firstly, it is difficult to precisely and quantitatively describe the "reasoning rules" and "sense" applied in a human brain. Secondly, the reasoning in a human brain is sophisticated, each particular occasion has its own rules of reasoning. The human brain adapts to different occasions easily, whereas it is hard for a machine vision algorithm to tell what events are happening in the scene and which set of rules should be selected accordingly. Typical difficult examples for optical flow computation are shadows, reflection, transparency, occlusion and motion boundaries, the occurrence of which can be immediately deduced by the human brain but confounds computer vision systems. Furthermore, it is hard to enunciate all motion types by numerical models, due to the variety and complexity of real motions in practice.

The difficulty also lies in the fact that the flow computation is limited by the video quality. Although human vision system can reconstruct motion from a blurry or noisy video sequence, these factors can be catastrophic for a computer vision algorithm. More importantly, to enable a machine to perceive motion, it entails that the temporal variation is "slower" than the spatial variation. In other words, the displacement of a point should be small enough to be reflected from the intensity values sampled at integer points. Today, large displacement recovery is still one of the most challenging problems in computer vision.

Even if a computer vision system can emulate human motion perception completely, it may still fail to estimate the true motion accurately. One well-known example is the

barber-pole illusion, where the true motion is horizontal but the perceived motion is vertical. However, given only the video sequence, the best motion information that one can expect is the apparent motion. The ambiguity of the apparent motion adds to the difficulty.

1.2 Thesis Overview

1.2.1 Contributions of This Thesis

The main contributions of this thesis are presented in Chapter 3 - 7. These chapters describe novel techniques for optical flow computation in the presence of discontinuous motion and fast rotation. The novelty of this work over previous methods is in the following aspects.

- It proposes new measures to detect the presence of multiple motions; and designs more adaptive mathematical reasoning models to inhibit the distortion from the inconsistent motions.
- It explores the complementary effects of different reasoning rules, as well as the combination of them, to overcome ambiguity and preserve motion boundaries simultaneously.
- It generalizes the traditional mathematical formulation, which favors translational motion with minimal apparent shape change, to rotation. The generalization provides a solution to flow computation with fast rotation, which causes large displacement that challenges existing flow computation methods.

1.2.2 Thesis Outline

The remainder of the thesis is organized as follows.

Chapter 2 overviews the literature of optical flow techniques, some challenging problems and potential solutions.

Chapter 3 proposes measures to detect the presence of inconsistency in a local area and predict the reliability of the recovered flow as an estimation of the real 2D motion.

Chapter 4 proceeds to detecting, removing or inhibiting the outliers that cause the inconsistency of a local computation system. In this chapter, various functions are investigated to detect and reject the outliers based on the inconsistency exhibited in spatial position, intensity, intensity variation and motion itself.

Chapter 5 describes a new strategy that integrates global computation with local computation to overcome the possible ambiguity inherited from the local computation. The complementary effects of local and global computation are explored. By linking the two computation techniques with motion segmentation, motion boundaries are well preserved by local computation and oversmoothing is avoided in global regularization. In each segmented region, local flow and global regularization are connected by the global subspace constraint, which effectively corrects the ambiguous flow vectors obtained in local computation and improves the robustness of global regularization.

Chapter 6 starts investigating the important yet difficult problem of recovering fast rotation, which has not been addressed much in the literature. This chapter presents a solution by a global computation technique. Different from previous global computation, the proposed method formulates the computation functional on an adaptive coordinate system, which is always locally oriented to the intrinsic directions of the underlying image structure. This chapter also discusses computing the intrinsic directions and directional derivatives in ways that are different from the conventions to achieve better performance. Numerical schemes to obtain the optimal solution are derived in the locally oriented coordinate system.

Chapter 7 goes one step further on fast rotation recovery by designing a new combinational scheme of local and global computation. Unlike the combination strategy designed for multiple motions in Chapter 5, segmentation of the motion field that contains rotation is difficult. Alternatively, in this chapter, the local and global computation is combined in a similar spirit to the Expectation-Maximization optimization. The customizing of both steps to preserve motion boundaries and recover fast rotation is demonstrated.

Chapter 8 summarizes the main points of the thesis, and discusses about future research.

Chapter 2

Differential Optical Flow Techniques for Motion Estimation

This chapter reviews the literature of optical flow computation, especially by differential techniques. Following the conventional taxonomy of local and global computation, this chapter reviews the representative methods in each category, and the research effort on combining them.

Along with the review, research difficulty is explained and used to motivate this thesis.

Optical flow, as an important measure to approximate image motion, has been intensively investigated for many applications including image registration, video coding, structure and motion reconstruction, object tracking. During 3 decade's intensive investigation, techniques such as overdetermined linear systems (e.g., (Lucas and Kanade 1981), (Campani and Verri 1990)), partial differential equations (e.g., (Horn and Schunck 1981), (Brox *et al.* 2004)), robust statistics (e.g.,(Simoncelli 1993a), (Roth and Black 2007), (Black and Anandan 1996)) and machine learning (e.g., (Sun *et al.* 2008), (Li and Huttenlocher 2008)), to name but a few, have been applied to accomplish the task. Several comparison studies have been conducted on the most influential contemporary techniques (e.g., (Barron *et al.* 1994), (Stiller and Konrad 1999), (McCane *et al.* 2001), (Baker *et al.* 2007)). In an early seminal survey (Barron *et al.* 1994), local differential methods (Lucas and Kanade 1981), (Uras *et al.* 1988) and phase-based method (Fleet and Jepson 1990) were shown to achieve lower average errors than the region-matching methods (Anandan 1989) and the energy-based method (Heeger 1988) on a variety of synthesized sequences simulating object translation, 3D camera motion and multiple moving objects. On top of that, they also demonstrate better visual results on real data sequences that depict 3D camera motion, object rotation and multiple independent motion. Moreover, within the class of differential techniques, local computation methods by Lucas-Kanade and Uras et al. have higher accuracy than the global computation of (Horn and Schunck 1981) and (Nagel 1983a) on some test sequences. According to the most recent survey (Baker *et al.* 2009), which compares the state-of-the-art methods comprehensively, today's most successful methods still belong to the class of differential techniques. However, global methods have been substantially improved from Horn-Schunck's original formulation, and now play a dominant role in flow computation (Baker *et al.* 2009).

This dissertation employs differential techniques, both local and global, for optical flow computation in the challenging cases of fast rotation and discontinuous motion. In this chapter, the research context and related works in the literature are reviewed. The research difficulty is explained and used to motivate the work presented in this thesis.

2.1 Problem Statement

Let $P = [X, Y, Z]^T$ be a moving point in the 3D world coordinate frame, and $p = [x, y, t]^T$ be its projection to the image plane in the image coordinate frame at time instance t. Recovering the *2D motion* of p, i.e., $\vec{v} = [\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}]$ from the brightness pattern

$E(x, y, t)$, is crucial to understand the motion and structure in the scene. The link between image intensity and the 2D motion is the total temporal derivative of the image intensity $\frac{dE}{dt}$ and the chain rule,

$$\frac{dE}{dt} = E_x x_t + E_y y_t + E_t. \tag{2.1}$$

Here and throughout the thesis, subscript x,y and t denote the corresponding spatial or temporal partial derivatives. Under circumstances that

- surface reflectance is Lambertian;
- pointwise light source is far away;
- photometric distortion is ignorable,

it is reasonable to assume that the projected intensity of P at different time instances remains constant (Trucco and Verri 2006), thus

$$\frac{dE}{dt} = E_x x_t + E_y y_t + E_t = 0. \tag{2.2}$$

Eq.2.2 is the so-called Brightness Constancy Equation (BCE). The 2D vector $[u, v]^T$ that satisfies the BCE, i.e.,

$$E_x u + E_y v + E_t = 0,$$

is the motion that can be perceived (or recovered) when E remains constant. This vector is the *optical flow* (p.195, (Trucco and Verri 2006)). It is shown in (Trucco and Verri 2006) that optical flow and 2D motion are not equivalent except in special cases that,

- the motion is translational, or,
- the illumination direction is parallel to the angular velocity.

Nevertheless, the two terms tend to be used interchangeably in modern literature (Baker *et al.* 2009), (Weickert *et al.* 2006). For example, the newly generated Middlebury datasets use the 2D motion ground truth as the flow ground truth. In this context, optical flow can also be interpreted as finding p's true correspondence at time instance $t + 1$, by only knowing that the correspondence has the same intensity as p. Unfortunately, this intensity constraint is generally insufficient, as there may be multiple points with the same intensity, whereas the true correspondence is unique. This is typically

an ill-posed inverse problem. To overcome the ambiguity, other prior knowledge or assumptions have to be integrated into the framework. Differential techniques, which formulate the prior constraints by a set of partial differential equations, have shown promising effectiveness. Following the traditional taxonomy of local and global computation, this section reviews how the two category of methods postulate the prior assumptions/constraints and recover the flow vectors.

2.2 Local Computation

The first local flow computation method is proposed by Lucas-Kanade (Lucas and Kanade 1981), who assumed that neighbouring pixels in a local patch have the same velocity. This assumption suggests that a true pair of correspondence have the same neighbourhood. Thus the BCE at each pixel contributes one constraint to the patch's flow vector. Let $E_x^{(k)}$, $E_y^{(k)}$ and $E_t^{(k)}$ denote the spatial and temporal partial derivatives of the kth pixel ($k = 1, \cdots, N$)) in the local area. The collection of the BCE at all pixels constructs a linear system

$$\begin{bmatrix} E_x^{(1)} & E_y^{(1)} \\ \vdots & \vdots \\ E_x^{(k)} & E_y^{(k)} \\ \vdots & \vdots \\ E_x^{(N)} & E_y^{(N)} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -E_t^{(1)} \\ \vdots \\ -E_t^{(k)} \\ \vdots \\ -E_t^{(N)} \end{bmatrix}. \tag{2.3}$$

The flow computation is thus converted to a classical problem of solving $A\vec{X} = \vec{b}$. If A has full rank, the intensity information in the local neighbourhood is sufficient to identify a unique correspondence for p in the next frame. The flow of p can be easily obtained by the least squares error solution, which minimizes the penalty function $\left\| A\vec{X} - \vec{b} \right\|_2$. Subsequent local computation approaches improve Lucas-Kanade's by modifying the BCE or the motion model, as described in the following sections.

2.2.1 Extension of the Invariant Features

Research has been conducted to relax the assumption of constant brightness to other features. For example, (Uras *et al.* 1988) assumes that pixel p's correspondence should have the same gradient ∇E. Mathematically, by applying the chain rule to $d\nabla E/dt =$

0, one obtains

$$\begin{bmatrix} E_{xx} & E_{xy} \\ E_{xy} & E_{yy} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} E_{xt} \\ E_{yt} \end{bmatrix} \tag{2.4}$$

If the system has full rank, the flow vector can be solved directly from the equation. In (Tistarelli 1996), the combination of brightness and gradient constancy is discussed. Compared to brightness constancy assumption, the gradient constancy is more robust to additive lighting changes. However, as gradient is always computed as the horizontal-vertical intensity variation based on the image grid, the gradient does not remain constant when the horizontal and vertical neighbourhood is changed during the motion. A typical instance that fails the assumption is fast rotation. In (Laskov and Kambhamettu 2003), the authors derived the change of p's Gaussian curvatures from K to K' before and after motion. Basically, the change can be expressed by

$$K' = K \cdot F \left(div \begin{pmatrix} u \\ v \end{pmatrix}, \nabla \begin{pmatrix} u \\ v \end{pmatrix}, \vec{n} \right),$$

where $F()$ is a function of the flow divergence, flow gradient and surface unit normal $\vec{n}$ (see the original work for details). Intuitively, the idea is to find p's correspondence in the next frame by knowing its Gaussian curvature K'. The relaxation of constancy assumption offers more flexibility for non-rigid motion. However, as the Gaussian curvature is based on the second order derivatives, it is more sensitive to noise than gradient. Thus the first order derivatives in the intrinsic directions (e.g., isophote and normal directions) are preferable, as they are more robust to noise, additional illumination change and rotation. Surprisingly, although such features have been widely applied to stereo matching and demonstrated effectiveness, they have not been considered for optical flow. To combat the difficulty caused by fast rotation, Section 7.2 describes a rotational invariant feature, which consists of first-order derivatives in intrinsic directions. Experiments conducted on real sequences of human motion with fast rotation show promising results.

2.2.2 Affine or Higher-Order Prior Assumption

Lucas-Kanade's constant flow assumption is rather restrictive, therefore research effort has been put into the assumption of locally affine model (Campani and Verri 1990) (Shi and Tomasi 1994). This assumption relates the kth pixel's velocity $[u^{(k)}, v^{(k)}]^T$ to the

velocity $[u, v]^T$ of the patch center by

$$\begin{bmatrix} u^{(k)} \\ v^{(k)} \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix} \begin{bmatrix} x^{(k)} - x \\ y^{(k)} - y \end{bmatrix}, \tag{2.5}$$

By substituting Eq.2.5 to each BCE associated with the k th pixel, i.e.,

$$E_x^{(k)} u^{(k)} + E_y^{(k)} v^{(k)} + E_t^{(k)} = 0,$$

an additional linear constraint of $[u, v]^T$ is obtained. They constitute a linear system in the form of $A\vec{X} = \vec{b}$, i.e.,

$$\begin{bmatrix} E_x^{(1)} & E_y^{(1)} & E_x^{(1)}\Delta x^{(1)} & E_x^{(1)}\Delta y^{(1)} & E_y^{(1)}\Delta x^{(1)} & E_y^{(1)}\Delta y^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ E_x^{(k)} & E_y^{(k)} & E_x^{(k)}\Delta x^{(k)} & E_x^{(k)}\Delta y^{(k)} & E_y^{(k)}\Delta x^{(k)} & E_y^{(k)}\Delta y^{(k)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ E_x^{(N)} & E_y^{(N)} & E_x^{(N)}\Delta x^{(N)} & E_x^{(N)}\Delta y^{(N)} & E_y^{(N)}\Delta x^{(N)} & E_y^{(N)}\Delta y^{(N)} \end{bmatrix} \begin{bmatrix} u \\ v \\ u_x \\ u_y \\ v_x \\ v_y \end{bmatrix} = - \begin{bmatrix} E_t^{(1)} \\ \vdots \\ E_t^{(k)} \\ \vdots \\ E_t^{(N)} \end{bmatrix}. \tag{2.6}$$

where $\vec{X}$ is the vector of the 6 motion parameters. In (Shi and Tomasi 1994), it is further concluded that affine model is preferable to estimate the large displacement accumulated across frames, although it is not necessarily better than the constant flow model to recover the small displacement between two consecutive frames. Therefore, it is suggested that affine model should be applied to monitor the tracking loss of a feature point, whereas the local flow computation should be based on the constant flow model. In this work, to respect motion with large deformation and high acceleration, most local computation systems are based on the affine flow model.

Eq.2.5 can also be interpreted as the first order Taylor expansion of $[u^{(k)}, v^{(k)}]$. Higher order Taylor expansions of $[u^{(k)}, v^{(k)}]$ and $E^{(k)}$ in flow computation have been discussed in (Otte and Nagel 1995). However, due to the sensitivity to noise of the differencing operator, partial derivatives with an order higher than 2 should generally be avoided in practice.

As above, using either brightness constancy or gradient constancy, either piecewise constant flow or affine flow, local computation integrates these assumptions into a linear system $A\vec{X} = \vec{b}$, where A and $\vec{b}$ contain the spatial and temporal variation of intensity respectively. It is possible that the linear system has no solution or the solution is not unique. In the scope of optical flow, these correspond to the motion boundary

problem and the aperture problem. Numerous attempts have been made to find solutions. The following subsections briefly review the research effort dedicated to each problem.

2.2.3 Handling the Aperture Problem of $A\vec{X} = \vec{b}$

Initially, research concern about local computation focused on the case that $A\vec{X} = \vec{b}$ has multiple solutions, when A is rank deficient. This happens when the spatial variation in A does not provide enough information to find the pixel's correspondence uniquely in the next frame. The situation is worsened if A is contaminated by noise and shows full rank, which is unfortunately almost inevitable in practice. In this case, a unique solution can be obtained by the pseudo-inverse of A, but it is generally erroneous.

To predict the occurrence of such errors, confidence measures have been proposed to evaluate the possibility that a system may suffer from the aperture problem. Lucas-Kanade use the least significant eigenvalue of the structure tensor (Lucas and Kanade 1981). Uras et al. use the condition number of the 2D Hessian matrix (Uras *et al.* 1988); Bertero et al. and Barron et al. use the determinant (Bertero *et al.* 1988) (Barron *et al.* 1994); and Simoncelli use the trace (Simoncelli 1993b). Except for the condition number measure, the ambiguity caused by the system's scale is generally neglected by these measures. More specifically, systems $A\vec{X} = \vec{b}$ and $\alpha A\vec{X} = \alpha\vec{b}(\alpha \neq 0)$ have the same solution manifolds and are always solved by the same pseudo-solution. Hence the confidence measure of a solution should be independent of the scaling factor α. However, the smallest eigenvalue measure, the trace measure and the determinant measure are all affected by α, which consequently leads to ambiguity.

The common aim of these measures is to detect the deficiency of spatial variation in the local patch. If detected, one approach to obtain a unique solution is to integrate other constraints, such as regularized variation of the flow field, with the system, e.g., (Bertero *et al.* 1988). This approach falls in the category of joint local and global computation, which is to be introduced in Section 2.4.

2.2.4 Handling the Inconsistency of $A\vec{X} = \vec{b}$

As discussed in the previous section, to obtain dense flow recovery by local computation, the patch has to be large enough to include salient intensity variation. However,

it is thus more likely to include pixels that have inconsistent motions into the system. In this case, A may have full rank, but $A\vec{X} = \vec{b}$ does not necessarily have a meaningful solution. As a consequence, flow estimation by the LSE is of low fidelity. Therefore, such systems should also be identified and replaced before it is applied to flow recovery. However, the only attempt to measure the system inconsistency is Haussecker et al.'s *coherency measure* and *corner measure* (Haussecker *et al.* 1998), which are specific to 3D structure tensors. A general inconsistency measure of $A\vec{X} = \vec{b}$ is proposed in Chapter 3, based on the rank increase from the coefficient matrix A to the augmented matrix $[A|\vec{b}]$ without computing the ranks of both matrices explicitly. A related line of work on rank increase measure is (Shechtman and Irani 2007), which is invented in the framework of human-behavior recognition. The difference between the two measures will be articulated in Section 3.3.

If $A\vec{X} = \vec{b}$ is detected as not consistent, the constraints from outliers should be inhibited. This can be done by weighting down the corresponding constraints. To save computational cost, these constraints may be weighted by zeros, as suggested in (Baker and Matthews 2004). Research on the weighting functions applied to local flow computation is quite limited. In (Baker and Matthews 2004), the pixels are weighted by the inverse of noise variance in case of spatially varying white Gaussian noise; and by the gradient magnitude if the noise is uniform. This weighting strategy requires the prior knowledge of the noise model. Furthermore, compared to outliers, noise is the secondary cause of the errors in the LSE estimation. A pairwise affinity-based weighting function is studied in (Ren 2008), based on the intensity contrast accumulated along the path linking the two pixels. This weighting function neglects the temporal variation, which is the direct cue for motion inconsistency. In Chapter 4, different possibilities of weighting functions, based on occlusion index, motion boundary, and intensity contrast, are discussed and tested intensively on benchmark sequences.

2.3 Global Computation

Unlike local computation, by which one linear system only recovers one pixel's flow, global computation recovers all pixels flow vectors by minimizing one energy function. Most global computation techniques assume that the flow field is smooth to some degree (e.g., (Horn and Schunck 1981), (Brox *et al.* 2004), (Ju *et al.* 1996)). Such techniques are conventionally referred to as *variational flow computation*, because calculus of variation is commonly used in such formulation to find the global optimal

solution. There are also some global methods that assume the scene is rigid and the camera is moving (e.g., the fundamental matrix constraint (Wedel *et al.* 2008), subspace constraint (Irani 2002)). Such prior assumptions are referred to as *rigidity priors* in (Baker *et al.* 2009). In this dissertation, both spatial smoothness constraint and subspace constraint are investigated for discontinuous motion estimation.

2.3.1 Variational Flow Techniques

Horn-Schunck's global smoothness assumption lays down the foundations of variational flow computation. The prior assumption of this method states that the flow field should vary slowly, such that the flow variation at point (x, y) measured by $|\nabla u|^2 + |\nabla v|^2$ is close to 0. Therefore, the deviation from this prior assumption summed over the whole region Ω (usually the image) leads to an error term

$$\mathfrak{E}_{prior} = \int\int_{\Omega} \left(u_x^2 + u_y^2 + v_x^2 + v_y^2 \right) dxdy, \tag{2.7}$$

which should be minimized. On the other hand, the deviation from the BCE aggregated over Ω leads to another error term

$$\mathfrak{E}_{data} = \int\int_{\Omega} \left(E_x u + E_y v + E_t \right)^2 dxdy, \tag{2.8}$$

to be minimized. Horn-Schunck formulated the dense flow recovery as minimizing the combination of the two error terms, balanced by a constant λ,

$$\mathfrak{E} = \mathfrak{E}_{data} + \lambda \mathfrak{E}_{prior}. \tag{2.9}$$

By calculus of variation, the optimal flow field satisfies the associated Euler-Lagrange equation pair,

$$\begin{aligned} E_x^2 u + E_x E_y v &= \lambda \Delta u - E_x E_t \\ E_x E_y u + E_y^2 v &= \lambda \Delta v - E_y E_t, \end{aligned} \tag{2.10}$$

where Δ is the Laplacian. Horn-Schunck discretized the Laplacian by the difference between the flow vector and the average of adjacent pixels' flow vectors, i.e.,

$$\Delta u = \bar{u} - u, \qquad \Delta v = \bar{v} - v, \tag{2.11}$$

where $\bar{u}$ and $\bar{v}$ mean the average of adjacent pixels' flow components. Following the discretization, standard numerical schemes can be applied straightforwardly to find

the solution. In (Horn and Schunck 1981), the authors used Gauss-Seidel iteration. At iteration step $\tau + 1$, the flow vector is updated by

$$\begin{aligned} u^{\tau+1} &= \bar{u}^{\tau} + E_x \left(E_x \bar{u}^{\tau} + E_y \bar{v}^{\tau} + E_t\right) / \left(\lambda + E_x^2 + E_y^2\right) \\ v^{\tau+1} &= \bar{v}^{\tau} + E_y \left(E_x \bar{u}^{\tau} + E_y \bar{v}^{\tau} + E_t\right) / \left(\lambda + E_x^2 + E_y^2\right) \end{aligned} \tag{2.12}$$

until convergence. The Euler-Lagrange equation 2.10 can be viewed as the steady state of a pair of diffusion equations; and the numerical iteration Eq.2.12 can be intuitively viewed as a diffusion process, by which the flow is "diffused" from the neighbouring pixels to the current pixel. Horn-Schunck's seminal method sketches an outline for variational flow computation. Subsequent research effort has been dedicated to remedy the data term's sensitivity to light changes and the smoothness term's capability of handling motion discontinuity; also to penalty functions that are more robust to motion outliers and motion models that are more flexible, as described in the following subsections.

Data Term

Similar to local computation, gradient constancy has been applied as a data term that is robust to additive illumination changes. The new data term can be given in either the form of (Brox *et al.* 2004)

$$\mathfrak{E}_1 = \int \int_{\Omega} \left(\nabla E(x, y, t) - \nabla E(x + u, y + v, t + 1)\right)^2 dxdy; \tag{2.13}$$

or its linearized form in (Weickert *et al.* 2006)

$$\mathfrak{E}_2 = \int \int_{\Omega} \left[\left(E_{xx}u + E_{xy}v + E_{xt}\right)^2 + \left(E_{xy}u + E_{yy}v + E_{yt}\right)^2\right] dxdy. \tag{2.14}$$

Because gradient is not rotational invariant, (Weickert *et al.* 2006) suggests using the gradient magnitude $|\nabla E| = \sqrt{E_x^2 + E_y^2}$, which is rotational invariant. The corresponding data term becomes

$$\mathfrak{E}_3 = \int \int_{\Omega} \left(\frac{\partial |\nabla E|}{\partial x} u + \frac{\partial |\nabla E|}{\partial y} v + \frac{\partial |\nabla E|}{\partial t}\right)^2 dxdy. \tag{2.15}$$

However, as the gradient magnitude discards one dimension of information, it is less effective if rotation is *not* involved. The SIFT (Lowe 2004) feature, which is rotational and scale invariant, has been used as the data term posed for scene matching in (Liu *et al.* 2008). However, since SIFT is based on histogram accumulated over image

region and not differentiable, this data term is not suitable for precise flow computation. Invariance of higher-order derivatives, such as Laplacian or the determinant of Hessian, has been discussed in (Weickert *et al.* 2006). Clearly such features are more sensitive to noise compared to gradient. Chapter 6 of this dissertation presents a solution based on the gradient in an oriented coordinate frame, the axes of which are always locally aligned with the local isophote and normal directions. The corresponding data term can be shown invariant to additive illumination change and rotation.

Discontinuity-Preserving Regularization

By penalizing the flow variation equally in horizontal and vertical directions, Horn-Schunck's method does not take motion boundary into consideration. This can also be seen from the diffusion process, in which a pixel's flow depends *equally* on its neighbours'. To respect motion discontinuity, the flow should only be diffused from neighbours that have consistent motion. The adaptive diffusion is generally implemented by weighting the flow's dependency on its neighbours. Many approaches define the weighting functions by the intensity similarity (e.g., (Nagel 1983a), (Wedel *et al.* 2009)), such that the diffusion along local edge direction is smooth, whereas diffusion in the normal direction is impeded. For example, the smoothness term used in (Wedel *et al.* 2009) has the form of

$$\mathfrak{E}_3 = \int\int_{\Omega} \exp\left(-\alpha \left|\nabla E\right|^{\beta}\right) \left(\left|\nabla u\right| + \left|\nabla v\right|\right) dxdy.$$

where α and β are pre-defined parameters controlling the dependency of the flow smoothness on the image smoothness. Compared to Horn-Schunck's uniform penalizing, this weighted penalty function enforces the diffusion to the direction in which intensity varies smoothly. The corresponding regularization process is *image-driven*, as the diffusion is steered by the underlying image structure. As image boundaries are not necessarily motion boundaries, (Weickert and Schnörr 2001) argues that oversegmentation may result, and hence the diffusion should be steered by the flow recovered from previous iteration steps. The steering is generally implemented by defining the weights as a monotonically decreasing function of the flow gradient. Such regularization is called *flow-driven*. The drawback with flow-driven regularization is that the recovery errors are easily propagated and enlarged in further iterations, especially around motion boundaries. Recently, joint image- and flow-driven regularization has also been investigated to avoid artifacts of oversegmentation and oversmoothing (Sun *et al.* 2008), (Zimmer *et al.* 2009). In (Sun *et al.* 2008), the flow is projected to the

intrinsic directions of the local patch, and the dependency between neighbouring pixels is controlled by the flow smoothness. Briefly, in this scheme, the diffusion direction is image-driven, whereas the diffusion speed is flow-driven. The anisotropic diffusion of (Zimmer *et al.* 2009) employs a similar idea, but the diffusion directions conform with the structure of the *data terms* rather than the local patch. The smoothness term presented in Chapter 6 is also driven by image and flow jointly, but in a different way from previous works. By orienting the coordinate frame to the isophote-normal directions, the diffusion coincides with the image structure, whereas the diffusion speed hinges on the flow affinity in both directions.

Beside image-driven and flow-driven approaches, multiple cue-driven diffusion have been proposed in (Xiao *et al.* 2006). The authors show that the updating procedure, for a simple example Eq.2.12, can be divided into two sub-procedures. One is from $\vec{v}^{\tau}$ to $\vec{v}^{\tau'}$ subject to the data term; and the other is from $\vec{v}^{\tau'}$ to $\vec{v}^{\tau+1}$ subject to the smoothness term. The latter updating procedure, which estimates $\vec{v}^{\tau+1}$ by the weighted average of $\vec{v}^{\tau'}$ of neighbouring pixels, can also be viewed as bilateral filtering or anisotropic convolution. Based on this observation, the authors of (Xiao *et al.* 2006) extend the flow- and image-driven convolution to multiple cue-driven, with the convolution kernels given by spatial proximity, intensity similarity, flow affinity and explicit occlusion labeling. A similar two-stage updating procedure is adopted in Chapter 7 of this dissertation, with the weighting options discussed for both sub-procedures.

Robust Penalty Functions

As shown in Eq. 2.7 and Eq. 2.8, Horn-Schunck formulate each energy term by the $L2$ norm of the residual error function. The solution that minimizes the energy functional is optimal in the sense of least squares error. From the statistics point of view, although least squares fits are robust to noise, they are notoriously sensitive to outliers (i.e., pixels that have a different type of motion). To alleviate the effect of outliers, (Black and Anandan 1996) suggests using robust norms to replace the $L2$ norm in the energy functional, for example the Lorentzian

$$\mathfrak{E}(r,\sigma) = \int\int_{\Omega} \log(1 + \frac{1}{2}\left(\frac{r(x,y)}{\sigma}\right)^2)dxdy,$$

where $r(x,y)$ is the residual error function at position (x,y), σ is the scale parameter.

Energy functionals defined by the $L1$ norm of the error function are also known to be more robust to outliers than the $L2$ norm. Replacing the $L2$ norm by the $L1$ norm, the

energy functional becomes

$$\mathfrak{E}(r,\sigma) = \int\int_{\Omega} |r(x,y)|\, dxdy.$$

The optimal solution of the energy functional has least absolute error. However, unlike $L2$-minimization, $L1$-minimization does not have a close form solution. One main category of solutions use an auxiliary variable to link the data term and the smoothness term, and minimizing each term by a sub-optimization procedure (e.g., (Trobin *et al.* 2008), (Wedel *et al.* 2008)). Another widely used category of solutions approximate the $L1$-norm by the square root of the $L2$ norm (e.g., (Sand and Teller 2006),(Zimmer *et al.* 2009),(Brox *et al.* 2004)) i.e.,

$$\mathfrak{E}(r,\epsilon) = \int\int_{\Omega} |r(x,y)|\, dxdy \approx \int\int_{\Omega} \left(\sqrt{r^2(x,y)+\epsilon}\right) dxdy, \tag{2.16}$$

where ϵ is a small positive value. This solves the non-differentiability but also induces non-linearity to the Euler-Lagrange equations. The nonlinearity is generally removed by an inner iteration, in a similar fashion to iteratively re-weighted L2 minimization, which is explained with details in Section 6.4.3.

Although the $L1$ norm is robust to outliers, (Werlberger *et al.* 2009) argues that it induces the "staircase" artifacts to the recovered flow. The authors therefore suggest using the Huber norm (Huber 1973), which was first applied for flow regularization by Shulman-Herve in (Shulman and Herve 1989), and is defined as a mix of the $L1$ and the $L2$ norm,

$$\mathfrak{E}(r,\epsilon) = \begin{cases} \int\int_{\Omega} \frac{r^2(x,y)}{2\epsilon} dxdy & |r(x,y)| \le \epsilon \\ \int\int_{\Omega} \left(|r(x,y)| - \frac{\epsilon}{2}\right) dxdy & \text{else,} \end{cases}$$

where ϵ is a predefined threshold. Superior flow computation accuracy of the Huber norm to the $L1$ norm is reported in (Werlberger *et al.* 2009).

Chapter 4, 5 and 7 of this dissertation employ the $L2$-norm of the deviation errors to define the energy functional, because outliers are detected and repressed explicitly in these chapters; whereas Chapter 6 adopts the approximated $L1$-norm (Eq.2.16), since the flow computation is unified in one global formulation, and lacks a particular step to handle outliers.

Prior Assumption of Affine Flow

Horn-Schunck's smoothness constraint states that $|\nabla u|^2 + |\nabla v|^2$ should be as small as possible. In the extreme case, u_x,u_y,v_x,v_y are all zeros, which correspond to the situation of constant flow. In other words, Horn-Schunck's smoothness constraint favors

constant flow. However, in many applications, the motion field has large divergence, curl or deformation (Cipolla and Blake 1997), and hence large first order derivatives of flow should be allowed. To adapt flow computation to this more general scenario, regularization based on affine motion model has been applied (e.g., (Ju *et al.* 1996), (Nir *et al.* 2008), (Trobin *et al.* 2008)). A direct extension to affine prior is to assume that the 6 motion parameters of a patch s should vary smoothly. Following the notation in (Ju *et al.* 1996), this parameter vector is denoted by

$$\vec{a}(s) = [a_1, \cdots, a_6]^T.$$

The affine prior of (Ju *et al.* 1996) states that

$$\sum_{t \in G(s)} \Psi\left(\|\vec{a}(s) - \vec{a}^*(t)\|\right) \tag{2.17}$$

should be as small as possible, where $G(s)$ means the neighbouring patches, $\vec{a}^*(t)$ is the motion parameter of patch t but aligned w.r.t the center of s.

Alternatively, (Nir *et al.* 2008) proposes minimizing the spatial-temporal variation of $\vec{a}$ directly, i.e.,

$$\mathfrak{E}_{smooth} = \int\int_{\Omega} \Psi\left(\sum_{i=1}^{6} \left\|\widetilde{\nabla} a_i\right\|^2\right) dxdy, \tag{2.18}$$

where $\widetilde{\nabla}$ is the spatial-temporal gradient, and Ψ presents the image region.

(Trobin *et al.* 2008) argues that the second order derivative operators are not orthogonal and possibly induces bias. Instead, the authors used a decorrelated second order derivatives operator $\|\diamond u\|$ and $\|\diamond v\|$, which should be zero for affine flow. In particular, $\|\diamond u\|$ is defined by

$$\|\diamond u\| = \sqrt{\frac{1}{3}}\sqrt{\left(u_{xx} + u_{yy}\right)^2 + 2\left(u_{xx} - u_{yy}\right)^2 + 8\left(u_{xy}\right)^2}, \tag{2.19}$$

and $\|\diamond v\|$ is defined similarly. With $\|\diamond u\| + \|\diamond v\|$ measuring the flow deviation from being affine, the smoothness term is expressed as

$$\mathfrak{E}_{smooth} = \int\int_{\Omega} (\|\diamond u\| + \|\diamond v\|) dxdy. \tag{2.20}$$

The disadvantage of this operator is its non-differentiability. To tackle the problem, the optimization in (Trobin *et al.* 2008) resorts to solving a dual variable, which entails another iteration process and incurs computational cost. In this dissertation, the global computation in Section 6.3.3 employs affine motion prior for fast rotation recovery, and the regularization is in a similar spirit to (Nir *et al.* 2008). Satisfactory performance of this regularization scheme has been obtained in our empirical study.

2.3.2 Rigid Motion Constraints

Motion rigidity constraint is another important global constraint in the literature of flow computation. More particularly, if the relative motion between the camera and the scene is rigid, then optical flow computation is equivalent to stereo disparity estimation. Moreover, all pixels' displacement vectors are subject to the *Epipolar Constraint* (p152, (Trucco and Verri 2006)), which is embodied by the fundamental matrix. Integrating the epipolar constraint or the fundamental matrix prior to the flow computation has been investigated in (Valgaerts *et al.* 2008) and (Wedel *et al.* 2008). While (Valgaerts *et al.* 2008) estimates the fundamental matrix and the flow jointly, (Wedel *et al.* 2008) requires the fundamental matrix to be either known or estimated online. Improved quantitative performance is reported on sequences with rigid scenes. However, according to (Baker *et al.* 2009), the method of (Wedel *et al.* 2008) "does poorly on the non-rigid scenes". Recently, (Wedel *et al.* 2009) proposes adapting the rigidity constraint to more general scenarios by using a weight term to toggle the fundamental matrix prior, based on the likelihood that the motion is rigid. The success of this approach has been confirmed by the evaluation conducted in (Baker *et al.* 2009).

Under the same assumption of rigid scene viewed by a moving camera, (Irani 2002) points out that the trajectories of all pixels across multiple frames reside in a very low rank (≤ 9) subspace. This constraint thus unifies the trajectories of all pixels by a small number (≤ 9) of basis vectors. The author also demonstrated that the displacement recovered under this subspace constraint is robust to the aperture problem. Based on this theory, Chapter 5 derives a new subspace constraint for the flow matrix, and proposes a new strategy to regularize the local flow computation in problematic areas.

2.4 Local and Global Computation in Tandem

As shown in previous sections, the local and global computations are fundamentally different from several perspectives.

- First, they are different in the way that neighbouring pixels interact through the constraints. Local methods constrain a pixel's flow by the surrounding pixels' *intensity* values; whereas global methods constrain a pixel's flow directly by the neighbouring pixels' *flow* vectors.

- Moreover, the two categories formulate the computation differently. Local methods integrate the flow constraints of one pixel into a linear system, and solve each pixel's system *independently*. In other words, the flow recovery accuracy at one pixel is not affected by the flow vectors recovered at other pixels; whereas global methods aggregate the constraint residuals over the whole field, and seek for an *overall* optimal solution. Hence the flow recovery at one pixel relies on the flow recovery at other pixels.

- While a linear system can be solved by standard numerical schemes for Least Squares Error (LSE) or Least Absolute Error (LAE) solutions for the local system, diffusion or diffusion-reaction processes are needed to find the optimal solution for global computation.

Both methods thus have strength and weakness in different aspects.

- Local computation suffers the *aperture problem*. In contrast to this situation, global methods have high accuracy in smooth ares, as the flow can be smoothly diffused from neighbouring pixels.

- Local computation is more robust to noise, as been observed by (Galvin *et al.* 1998) and (Barron *et al.* 1994) from experiments conducted on sequences with noise added. Moreover, the recovery error at one pixel does not interact with the flow recovery of other pixels; whereas the diffusion process of global methods may propagate one pixel's error to other pixels.

The complementary effects of the two methods have been studied. In (Uras *et al.* 1988), the flow obtained by local computation is post-processed by a regularization procedure. Briefly, this approach partitions the image into 8×8 blocks. In each block, the most reliable flow vector is used as the flow for all pixels. Although this subsampling regularization is not global, its effectiveness illustrates the necessity to exploit motion consistency for performance improvement.

In fact, regularizing ill-posed local computation by global regularization is a typical solution to overcome the ill-posedness. The integration of local computation and global regularization has been discussed in (Bertero *et al.* 1988), in a general context of solving inverse problems in early vision. It is further applied to optical flow computation in (Bruhn *et al.* 2005), which substitutes the data term in Horn-Schunck functional by

Lucas-Kanade's local system. In the simplest settings, the Horn-Schunck functional is modified to

$$\mathfrak{E} = \int\int_{\Omega} \left[\left\| A \begin{bmatrix} u \\ v \end{bmatrix} - \vec{b} \right\|_2^2 \right] + \lambda \left[(\|\nabla u\|^2 + \|\nabla v\|^2) \right] dxdy, \tag{2.21}$$

where A are $\vec{b}$ are defined in Eq.2.3, and λ is the constant balancing the two terms. In this new energy functional, the data term improves the functional's robustness, and the smoothness term diffuses the flow into the area where local flow system fails. The work also extends the formulation to anisotropic diffusion and spatio-temporal regularization. However, as this combination scheme is still formulated as a global computation, the error propagation effect still exists in the diffusion process.

In (Ohta 1991), a separate global regularization is designed to stabilize the local flow results. In this method, a local computation recovers the flow field $[u_0, v_0]$; For each recovered vector, the reliability indices $\vec{r}_1$ and $\vec{r}_2$, expressed by the most and least reliable directions scaled by the level of reliability, are quantitatively evaluated from the image intensity. A global optimization functional

$$\mathfrak{E}(u,v) = \int\int_{\Omega} \left[\left(\vec{r}_1 \cdot \begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} \right)^2 + \left(\vec{r}_2 \cdot \begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} \right)^2 \right] + \alpha \left[\|\nabla u\|^2 + \|\nabla v\|^2 \right] dxdy \tag{2.22}$$

is minimized to accomplish the stabilization. As pointed out by the author, the stabilization is more effective in textured regions than in smooth regions. Presumably this is because local flow computation in smooth regions is error-prone, and the global stabilization diffuses the errors to other pixels. This suggests that correcting the unreliable flow before the spatial regularization is desirable.

The subspace constraint offers a good solution to this problem. On one hand, in (Irani 2002), the application of subspace projection has demonstrated its capability to correct the erroneous flow vectors arising from the aperture problem. On the other hand, it provides a denoised input for the global regularization. However, the aggressive assumption of a unique and rigid motion between the scene and camera limits its application. In Chapter 5 of this dissertation, the problem is addressed by segmenting the image into regions with consistent motion, which allows the global constraints to be applied without ambiguity.

2.5 Summary

In recent years, much effort has been dedicated to improving both local and global flow computation techniques. However, several fundamental problems remain. Although local computation suffers the aperture problem, it is robust to noise, as one pixel's flow recovery does not affect other flow vectors. If outliers can be effectively detected and rejected from the linear system, the local patch can be safely enlarged to overcome the aperture problem. However, not much work has been done in this area. This is addressed in Chapter 3 and Chapter 4 of the thesis.

The complementary effects of local and global computation, although it has been noticed and discussed in the literature, has only attracted limited research attention. As a global constraint, the subspace constraint can effectively correct the erroneous flow result from the local computation and facilitates the spatial regularization. However, it has not been widely applied, due to the restrictive assumption of the rigid motion pattern. If local computation, subspace projection and spatial regularization should be combined without ambiguity, the motion discontinuity will be preserved at motion boundaries, while the motion consistency will be maintained in the interior of an object. Such a combination scheme, is yet to be investigated. A possible solution is proposed in Chapter 5 of this thesis.

Most top-performing flow computation methods favor translational or constant flow, by assuming constant gradient or minimal flow variation. These methods are not suitable to recover fast rotation. This thesis presents two methods to respect fast rotation in Chapter 6 and Chapter 7.

Chapter 3

Measuring Local Motion Inconsistency

Optical flow registers pixels that are projected from the same world point onto consecutive frames by assuming that the point's projected intensity remains constant. However, this assumption in isolation is almost always insufficient to pinpoint a pixel's true correspondence in the next frame. This insufficiency has to be solved by some supplementary "prior" assumptions. Commonly, prior assumptions assume the flow field is smooth to some degree, and therefore neighbouring pixels' intensity or velocity can be utilized to infer the flow vector uniquely. Such assumptions are applicable in most image areas apart from the motion boundaries. As a result, flow computation in the presence of motion discontinuity remains a challenging problem. Aiming at motion discontinuity preserving flow computation, this chapter discusses how to detect motion inconsistency using evidence from the local intensity variation pattern. Based on the theoretical analysis of solution existence of a linear system, new measures are proposed to detect motion inconsistency from one pixel, between two pixels and among a group of pixels.

The previous chapter has shown that motion boundaries are the primary challenge for current optical flow techniques. This is because the computation has to postulate motion consistency to recover an optical flow vector meaningfully and uniquely for each pixel. As a consequence, flow recovery is always problematic around motion boundaries, where spatially adjacent pixels may move independently without any consistency. To overcome the difficulty, numerous techniques have been proposed. They generally involve one or a combination of several following procedures:

- detect motion boundaries (e.g., (Xu *et al.* 2008), (Lei and Yang 2009), (Xiao *et al.* 2006));
- measure or predict the reliability of the assumed flow computation model, i.e., flow confidence (e.g., (Ohta 1991), (Uras *et al.* 1988),(Bruhn and Weickert 2006));
- relate a pixel's velocity only to those pixels that have consistent motion patterns (e.g., (Sun *et al.* 2008), (Zimmer *et al.* 2009)).

This chapter investigates the connection between flow consistency, motion boundary, computation fidelity and the intensity variation, in order to handle these tasks efficiently. Motion inconsistency detected from one pixel, between two pixels and among a group of pixels are proposed. These detection measures will be applied in the flow computation techniques presented in subsequent Chapters.

3.1 Motion Inconsistency of One Pixel

In this work, we limit the discussion to the scope that motion is the only cause for the image intensity change in the temporal dimension. If the motion of a scene point is coherent, the temporal variation of its image intensity is closely related to the spatial variation. For example, the basic optical flow constraint states that E_t is a linear combination of E_x and E_y. Conversely, if the intensity temporal variation is independent of the spatial variation, the independence can be attributed to unexpected motion such as occlusion or discontinuity. Pixels affected by inconsistent motion are generally outliers in a local system. In this section, we show that the inconsistency may be detected from one pixel's second order partial derivatives. The detection strategy is suitable for the following motion models.

1. The projected point's intensity remains invariant in the sequence, and the spatial-temporal variation of the flow vector vanishes;

2. The projected point's spatio-temporal gradient remains invariant in the sequence.

In the first model, the brightness constancy assumption provides the baseline constraint of the flow vector at a pixel. This can be written as,

$$\frac{dE(x,y,t)}{dt} = E_x u + E_y v + E_t = 0. \tag{3.1}$$

By taking partial derivatives at both sides of Eq.3.1 with respect to x, y, and t, an underdetermined linear system for 8 unknowns is obtained,

$$\begin{bmatrix} E_{xx} & E_{xy} & E_x & E_y & 0 & 0 & 0 & 0 \\ E_{xy} & E_{yy} & 0 & 0 & E_x & E_y & 0 & 0 \\ E_{xt} & E_{yt} & 0 & 0 & 0 & 0 & E_x & E_y \end{bmatrix} \vec{X} = - \begin{bmatrix} E_{xt} \\ E_{yt} \\ E_{tt} \end{bmatrix}. \tag{3.2}$$

where $\vec{X} = [u,\ v,\ u_x,\ v_x,\ u_y,\ v_y,\ u_t,\ v_t]^T$. As the optical flow components are assumed to have vanished spatio-temporal variation, Eq. 3.2 degenerates to

$$\begin{bmatrix} E_{xx} & E_{xy} \\ E_{xy} & E_{yy} \\ E_{xt} & E_{yt} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} E_{xt} \\ E_{yt} \\ E_{tt} \end{bmatrix}. \tag{3.3}$$

It can be straightforwardly verified that the second model also leads to Eq. 3.3, which can also be regarded as an extension of (Uras *et al.* 1988) from the spatial domain to the spatio-temporal domain (see Section 2.2).

If Eq. 3.3 has at least one solution, there exists a flow vector that conforms with the motion assumptions. Furthermore, $[E_{xt}\ E_{yt}\ E_{tt}]^T$ is a linear combination of $[E_{xx}\ E_{xy}\ E_{xt}]^T$ and $[E_{xy}\ E_{yy}\ E_{yt}]^T$. Thus by elementary matrix operations, the 3D Hessian matrix H_{3D} can be transformed to

$$H_{3D} = \begin{bmatrix} E_{xx} & E_{xy} & E_{xt} \\ E_{xy} & E_{yy} & E_{yt} \\ E_{xt} & E_{yt} & E_{tt} \end{bmatrix} \rightarrow \begin{bmatrix} E_{xx} & E_{xy} & 0 \\ E_{xy} & E_{yy} & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} H_{2D} & 0 \\ O & 0 \end{bmatrix}.$$

Therefore, H_{3D} and H_{2D} have the same rank. Conversely, if the rank of H_{3D} is higher than H_{2D}, the motion model has no solution. In other words, there does not exist a flow vector that meets the motion assumptions. That is, the rank increase from a pixel's H_{2D} to H_{3D} indicates the presence of inconsistent motion. To the best of our knowledge at the time of writing, this connection between the Hessian matrices rank increase and motion inconsistency has not been discussed in the literature.

3.2 Motion Inconsistency Between Two Pixels

The previous section shows that motion inconsistency may be reflected in a pixel's intensity variation pattern. In many applications, it is also important to detect whether this pixel has flow that is consistent with another pixel in its neighbourhood. If the pair of pixels have consistent motion, both pixels' intensity information can be utilized to improve the accuracy of the computed optical flow. Otherwise, the estimation can become severely skewed. A related line of work is the pairwise affinity term proposed in (Ren 2008). This is based on intensity contrast and is used to group pixels that are likely to have consistent motion. This method, although it exploits spatial variation, neglects the temporal variation information of the pixels, which is the direct cue for motion. This section proposes estimating pairwise motion affinity by the 3D Hessian matrices of the two pixels, which contains both spatial and temporal variation. The detection strategy is suitable for motion that can be locally modeled by

$$H_{3D}\vec{V} = \vec{f},$$

where $\vec{f}$ is a vector of small magnitude. For instance, the two motion models discussed in the previous section fit in the discussion of this section.

Let p and c be two pixels which satisfy the brightness constancy equation. Thus each pixel is associated with a linear system, specifically $H_{3D}(p)\,\vec{V}(p) = \vec{f}(p)$ and $H_{3D}(c)\,\vec{V}(c) = \vec{f}(c)$. If the two pixels have similar flow, i.e., $\vec{V}(p) \approx \vec{V}(c)$, this indicates that the two linear systems have common solutions. Mathematically, a necessary and sufficient condition for the two systems to have common solutions is that the two solution manifolds $\Omega(c)$ and $\Omega(p)$ have non-trivial intersection (i.e. not only zero), where

$$\Omega(c) = \left\{ H_{3D}^{+}(c)\vec{f}(c) + N(H_{3D}(c)) \right\}$$
$$\Omega(p) = \left\{ H_{3D}^{+}(p)\vec{f}(p) + N(H_{3D}(p)) \right\}.^{2}$$

Note that if the flow varies slowly, $\left\|\vec{f}(c)\right\|$ and $\left\|\vec{f}(p)\right\|$ should vanish, and hence the intersection of the solution manifolds is dominated by the intersection of $N(H_{3D}(c))$ and $N(H_{3D}(p))$. Therefore, a lack of non-trivial intersection of $N(H_{3D}(c))$ and $N(H_{3D}(p))$ generally indicates flow inconsistency at p and c.

In the following discussion, we assume that the eigenvectors of $H_{3D}(p)$ are $\vec{e}_1^{\,p}$, $\vec{e}_2^{\,p}$ and $\vec{e}_3^{\,p}$; and their corresponding eigenvalues are sorted such that $|\lambda_1^p| \geq |\lambda_2^p| \geq |\lambda_3^p|$. The

[2] $N(\bullet)$ is the null space of the matrix, and $^{+}$ means the pseudo-inverse of the matrix.

same notation is used for $H_{3D}(c)$, with the superscript replaced by c. The intersection of the two solution manifolds thus falls into one of the following four cases:

Case 1: Rank$(H_{3D}(p)) = 1$ **and** Rank$(H_{3D}(c)) = 1$**.**

In this case,

$$N(H_{3D}(c)) = \text{Span}\{\vec{e}_2^c, \vec{e}_3^c\}, N(H_{3D}(p)) = \text{Span}\{\vec{e}_2^p, \vec{e}_3^p\}.$$

Since the vector space in our problem is $\mathbf{R}^3$, the intersection of the two planes is non-trivial.

Case 2: Rank$(H_{3D}(p)) = 1$ **and** Rank$(H_{3D}(c)) = 2$**.**

In this case

$$N(H_{3D}(c)) = \text{Span}\{\vec{e}_3^c\}, N(H_{3D}(p)) = \text{Span}\{\vec{e}_2^p, \vec{e}_3^p\}.$$

The two null spaces have trivial intersection if

$$N(H_{3D}(c)) = \text{Col}(H_{3D}(p)) \perp N(H_{3D}(p)). \tag{3.4}$$

where Col$(H_{3D}(p))$ is the column space of $(H_{3D}(p))$, and is spanned by $\vec{e}_1^p$. Thus Eq.(3.4) reads

$$\text{Span}\{\vec{e}_3^c\} = \text{Span}\{\vec{e}_1^p\}.$$

Therefore, if $\vec{e}_3^c$ is parallel to $\vec{e}_1^p$, the two solution manifolds have trivial intersection only.[3]

Case 3: Rank$(H_{3D}(p)) = 2$

In this case

$$N(H_{3D}(p)) = \text{Span}\{\vec{e}_3^p\}.$$

Either Rank$(H_{3D}(c)) = 1$ or Rank$(H_{3D}(c)) = 2$, so we have

$$\text{Span}\{\vec{e}_1^c\} \subset \text{Col}(H_{3D}(c)) \perp N(H_{3D}(c)), \text{Span}\{\vec{e}_3^p\} = N(H_{3D}(p)).$$

Therefore, if $\vec{e}_1^c$ is parallel to $\vec{e}_3^p$, Null$(H_{3D}(p)) \perp$ Null$(H_{3D}(c))$, which indicates that the two solution manifolds have trivial intersection.

Case 4: Rank$(H_{3D}(p)) = 3$. In this case, $N(H_{3D}(p))$ must have only trivial intersection with $N(H_{3D}(c))$.

[3]Eigenvectors whose inner product is greater than 0.95 are considered parallel.

As analyzed above, the lack of non-trivial intersection of $\Omega(c)$ and $\Omega(p)$ provides evidence for inconsistent motion. Therefore in practice, if this is detected from the eigenvectors of $H_{3D}(c)$ and $H_{3D}(p)$, the influence of pixels p and c on each other's flow computation should be inhibited. Note that the discussion for the 4 cases still holds if p and c are swapped.

3.3 Motion Consistency among a Group of Pixels

Detecting motion inconsistency between *a pair of* pixels is important, yet the spectral decomposition of all pixels is expensive. As consistency is more likely to happen in most image regions, it is thus more efficient to detect motion inconsistency among *a group of* pixels, and use the pixel-wise or pairwise inconsistency detection to reject outliers only in the groups that exhibit inconsistency. For this purpose, this section proposes a continuous quantity that measures the inconsistency of a generic linear system. When this measure is applied to a local flow computation system, it is able to identify the motion inconsistency in the local patch. Furthermore, it also predicts the fidelity of the flow recovered by the system.

3.3.1 An Inconsistency Measure for Linear Systems

Let $A\vec{X} = \vec{b}$, where $A \in \mathbb{R}^{m\times n}$, $\vec{b} \in \mathbb{R}^{m\times 1}$ denote a generic linear system. This is a crucial model in a number of computer vision problems including optical flow, as shown in Chapter 2. In many cases, the linear system is ill-posed in the sense that:

1. The solution of the system is not unique. This happens when the system is deficient in independent constraints. Generally the deficiency can be detected by A's rank-deficiency or its infinite condition number. In practice, the existence of noise ensures that matrix A generally has full rank. Therefore real applications do not tend to have rank deficiency, but rather are "well-posed but ill-conditioned", as their condition numbers are large but finite.

2. The solution of the system does not exist. This occurs where the system has "extra" constraints that are mutually contradictory. In this case, the pseudo-solution is severely distorted, as the system contains at least one constraint that is inconsistent with the true vector $\vec{X}$. It is thus important to detect the inconsistency

before the pseudo-solution is applied. Therefore if the inconsistency is detected, the linear system is not an appropriate model for the problem.

Mathematically, a necessary and sufficient condition for the system to be consistent is that the coefficient matrix A and the augmented matrix $[A|\vec{b}]$ have the same rank. Thus it seems plausible to detect inconsistency by counting the number of the non-trivial singular values of each matrix and taking the difference. However, in real applications, due to the existence of noise, A and $[A|\vec{b}]$ are always of full rank. Furthermore, if $m > n$, both matrices have rank equal to the number of their columns. Hence there is always rank increase from A to $[A|\vec{b}]$. To reduce the expected effect of noise on the rank estimation, a threshold has to be defined to model the noise level. As pointed out by Shechtman-Irani in (Shechtman and Irani 2007), a wrong threshold may lead to wrong rank detection. Furthermore, rank comparison leads to a binary decision of consistency or inconsistency, which is inflexible. Shechtman-Irani suggested indexing the rank comparison by *one* continuous quantity, which is further thresholded to make the rank increase decision. This idea is novel in the sense that it unifies the rank comparison to one quantity rather than comparing two individual rank numbers. Also, thresholding a continuous rank-increase measure is more sensible and direct than thresholding the eigenvalues. However Shechtman-Irani's rank increase measure is limited to 3D and 2D structure tensors.

This section proposed a different sufficient and necessary condition for a general system $A\vec{X} = \vec{b}$ to be consistent.

Theorem 1 *Define $\vec{s} = A^T\vec{b}$, $G = A^TA$, and let real numbers $\lambda_1 \geq \cdots \geq \lambda_n$ be the sorted eigenvalues of G and $\vec{e}_1, \cdots, \vec{e}_n$ be the corresponding eigenvectors. Let θ defined by*

$$\theta = \begin{cases} 0 & \text{if all eigenvalues are zero} \\ \sum_{i=1}^{r} \frac{\left(\vec{s}^T \cdot \vec{e}_i\right)^2}{\lambda_i} & \text{if } G \text{ has } r \text{ nonzero eigenvalues} \end{cases} \tag{3.5}$$

System $A\vec{X} = \vec{b}$ is consistent if and only if $\left\|\vec{b}\right\|_2^2 = \theta$.

Proof If all the eigenvalues of G are zero, both G and A have rank zero. In this case, system $A\vec{X} = \vec{b}$ is consistent if and only if $\vec{b}$ is a zero vector, i.e.,

$$rank(A) = rank([A|\vec{b}]) = 0 \Leftrightarrow \left\|\vec{b}\right\|_2^2 = 0. \tag{3.6}$$

If G has $r \geq 1$ eigenvalues, both G and A have rank r. Let $A = USV^T$ be the singular value decomposition of A, where U and V are orthogonal matrices and the diagonal matrix S contains the non-zero singular values of A, i.e., $\sqrt{\lambda_1}, \sqrt{\lambda_2}, \cdots, \sqrt{\lambda_r}$.

Denote the ith column of U by $\vec{u}_i$. $rank(A) = rank([A|\vec{b}])$ if and only if $\vec{b}$ is in the column space of A, which is spanned by the first r columns of U. That is,

$$\begin{aligned} rank(A) &= rank([A|\vec{b}]) \\ &\Updownarrow \\ \left\|\vec{b}\right\|_2^2 &= \sum_{i=1}^{r} (\vec{b}^T \vec{u}_i)^2; \end{aligned} \tag{3.7}$$

Note that $G = A^T A = VS^2V^T$, and hence matrix V contains the eigenvectors of G, i.e.,

$$V = [\vec{e}_1 \quad \vec{e}_2 \quad \cdots \quad \vec{e}_n]. \tag{3.8}$$

It can then be verified that

$$A\vec{e}_i = USV^T\vec{e}_i = \sqrt{\lambda_i}\vec{u}_i. \tag{3.9}$$

By the definition of $\vec{p}$, Eq.3.9 leads to

$$\vec{s}^T\vec{e}_i = \vec{b}^T A\vec{e}_i = \sqrt{\lambda_i}\vec{b}^T\vec{u}_i. \tag{3.10}$$

Combining with Eq. 3.7, one can obtain

$$\sum_{i=1}^{r} \frac{(\vec{s}^T\vec{e}_i)^2}{\lambda_i} = \sum_{i=1}^{r} (\vec{b}^T\vec{u}_i)^2 = \left\|\vec{b}\right\|_2^2 \Leftrightarrow rank(A) = rank([A|\vec{b}]) \tag{3.11}$$

Together Eq.3.6 and Eq.3.11 prove that

$$rank(A) = rank([A|\vec{b}]) \Leftrightarrow \left\|\vec{b}\right\|_2^2 = \theta.$$

This proves Theorem 1.

The theorem has a direct corollary as stated below.

Corollary 2 *For any vector $\vec{b}$ and θ defined as in Theorem 1, $\left\|\vec{b}\right\|_2^2 \geq \theta$.*

Proof Here we use the same notations as in the proof of Theorem 1. With the columns of U forming an orthogonal basis of $\mathbb{R}^{m\times m}$, any vector $\vec{b} \in \mathbb{R}^{m\times 1}$ satisfies

$$\left\|\vec{b}\right\|_2^2 = \sum_{i=1}^{m} (\vec{b}^T\vec{u}_i)^2 \geq \sum_{i=1}^{r} (\vec{b}^T\vec{u}_i)^2. \tag{3.12}$$

Substituting Eq.3.11 into the R.H.S. of the inequality 3.12, the following inequality can be easily verified.

$$\left\|\vec{b}\right\|_2^2 \geq \theta. \tag{3.13}$$

Theorem 1 states that $\left\|\vec{b}\right\|_2^2$ can not take any arbitrary value if the system is expected to have solution(s). In other words, for the system to be solvable, $\left\|\vec{b}\right\|_2^2$ must equal an ideal value θ. Theoretically, if the constraints in $A\vec{X} = \vec{b}$ are strictly consistent, then $\left\|\vec{b}\right\|_2^2 - \theta$ is exactly zero, and vice versa. However, due to the existence of noise, each constraint is a noisy observation of the true underlying data. Therefore, it is more practical to allow $\left\|\vec{b}\right\|_2^2 - \theta = n_\epsilon$, where n_ϵ is a noise signal of small magnitude. Otherwise, a large magnitude of $\left\|\vec{b}\right\|_2^2 - \theta$ is evidence for rank increase from the coefficient matrix to the augmented matrix, and hence the inconsistency of the constraints can be inferred. Therefore our COnstraint INconsistency (COIN) measure is defined by

$$m = \left\|\vec{b}\right\|_2^2 - \theta. \tag{3.14}$$

The COIN measure depends continuously on the underlying data, which is important to indicate the degree of inconsistency of a linear system faithfully. It also has low computational cost compared to direct rank comparison, as it uses the eigendecomposition A^TA only; whereas the rank comparison of A and $[A|\vec{b}]$ requires the SVD of both matrices. It is worth noting that θ, which is the projection of $\vec{b}$ to the column space of A, coincides with the least squares solution of the system $A\vec{X} = \vec{b}$. In practice, it can therefore also be obtained as $AA^+\vec{b}$, where A^+ is the Penrose-Moore pseudoinverse, or by other methods for calculating a least squares solution and residual.

3.3.2 An Example of the Lucas-Kanade System

Recall the Lucas-Kanade linear system introduced in Chapter 2. In this linear system, A^TA and $[A|\vec{b}]^T[A|\vec{b}]$ are 2D and 3D structure tensors, i.e.,

$$\begin{aligned} A^TA &= \begin{bmatrix} \sum E_xE_x & \sum E_xE_y \\ \sum E_xE_y & \sum E_yE_y \end{bmatrix}; \\ \left[A|\vec{b}\right]^T\left[A|\vec{b}\right] &= \begin{bmatrix} \sum E_xE_x & \sum E_xE_y & \sum E_xE_t \\ \sum E_xE_y & \sum E_yE_y & \sum E_yE_t \\ \sum E_xE_t & \sum E_yE_t & \sum E_tE_t \end{bmatrix}, \end{aligned} \tag{3.15}$$

where the summation is taken over the local area. Let $\lambda_1 \geq \lambda_2$ be the eigenvalues of the 2D structure tensor, and $\vec{e}_1, \vec{e}_2$ be the corresponding eigenvectors. The COIN measure in this example is given by

$$m = \begin{cases} \sum E_t^2 & \text{if } rank(A) = 0; \\ \sum E_t^2 - \sum_{i=1}^{r} \frac{(\vec{p} \cdot \vec{e}_i)^2}{\lambda_i} & \text{if } rank(A) = r; \end{cases} \tag{3.16}$$

where $\vec{p} = \left[\sum E_x E_t \ \sum E_y E_t\right]$.

By applying Theorem 1 to this example and noting that

$$rank(A^T A) = rank(A), rank([A|\vec{b}]^T [A|\vec{b}]) = rank([A|\vec{b}]),$$

it can be concluded that *the Lucas-Kanade system is consistent if and only if the 2D and 3D structure tensors have same rank.*

It has been noted that if the 3D structure tensor has full rank, then the local spatial-temporal patch has *no* coherent motion (e.g., (Haussecker *et al.* 1998)). In (Shechtman and Irani 2007), Shechtman and Irani generalized this observation by pointing out that, if the 3D structure tensor has higher rank than the 2D structure tensor, then there is no coherent motion in the patch. The rationale of the generalization, however, is based on a case by case study of all possible ranks which the structure tensors may have (Section 4, (Shechtman and Irani 2007)). This example sheds a different light on Shechtman and Irani's theory. That is, the 3D structure tensor's rank increase, which implies lack of coherent motion, is just another expression of the inconsistency among the optical flow constraints in Lucas-Kanade's system.

3.3.3 Measuring flow confidence and motion boundary by COIN

Since the advent of Lucas-Kanade flow computation, many sophisticated local flow models have been proposed (e.g., (Uras *et al.* 1988), (Shi and Tomasi 1994)). Commonly, they are modeled as solving a linear system $A\vec{X} = \vec{b}$, where A and $\vec{b}$ contain the spatial and temporal derivatives of E at several positions, and $\vec{X}$ is the unknown vector of motion parameters (see Chapter 2 for details). The discussion on the general constraint inconsistency thus is applicable to these local optical flow computation systems. In the following, we discuss two applications of the COIN measure in the context of optical flow computation. These are motion boundary detection and flow confidence measure.

Motion Boundary Detection

Where the prior assumption for a local patch is valid for the underlying data, the optical flow constraints are consistent. Conversely, if the linear system of these optical flow constraints show clear inconsistency, it generally marks the violation of the prior assumption. One primary reason for the violation is the presence of motion boundary, as spatially adjacent pixels may move with completely unrelated velocities. Thus the COIN measure can be applied as a simple motion boundary detector, with the large COIN value signaling a high probability of a motion boundary's presence.

In the literature, Shechtman-Irani proposed detecting motion inconsistency by their rank-increase measure from 2D- to 3D structure tensor, in the framework of human behavior recognition. As explained in Section 3.3.2, measuring rank increase from 2D- to 3D structure tensor is equivalent to measuring the constraint consistency of the Lucas-Kanade optical flow computation system.

The COIN and the rank-increase measure are related in the sense that, if the linear system $A\vec{X} = \vec{b}$ degenerates to Lucas-Kanade's optical flow computation (Lucas and Kanade 1981), both measures detect motion inconsistency by comparing the continuous rank difference between the 2D structure tensor and 3D structure tensor. However, the two measures are fundamentally different in the following aspects. First, the proposed measure is not limited to Lucas-Kanade's system. The arrays A and $\vec{b}$ can be generated by one or several prior models that best suit the underlying data. whereas the rank-increase measure is limited to 2D and 3D structure tensor. Second, the two measures have different definitions rooted from different mathematical theories. The rank-increase measure is based on the *interlacing property* of symmetric matrices (pp.396, (Golub and Van Loan 1996)), and is defined as

$$\Delta r = \frac{\det(T_{3D})}{\det(T_{2D})\lambda_l},$$

where T_{2D} and T_{3D} denote the Lucas-Kanade 2D and 3D structure tensor (Lucas and Kanade 1981), and λ_l is the largest eigenvalue of T_{3D}. This definition does not take "divided by zero" into consideration; whereas the proposed measure is based on the projection of $\vec{b}$ to the column space of A, and avoids zeros in the denominator. Third, the rank increase measure works well if the camera motion is negligible relative to the object motion, but results in false positive detection if the camera undergoes fast

motion and the background lacks spatial texture. In this case, both λ_l and $\frac{\det(T_{3D})}{\det(T_{2D})}$ correspond to the temporal eigenvalue (Lindeberg *et al.* 2004). As a consequence, the measure value is near to 1, which indicates motion inconsistency even if the background undergoes uniform motion. In contrast, the proposed measure is robust to fast camera motion.

Flow Confidence Measurement

If a local patch contains motion boundaries, the system is prone to collect constraints that are valid for some neighbouring pixels but invalid for the center pixel, as they are from different motion segments. Consequently the estimated solution is unreliable. In general, the more inconsistent the two motion patterns, the more distorted the estimation. As explained in Section 2.2.4, a practical approach is to predict the unreliability, and replace the invalid system by an appropriate model.

In this work, we propose measuring the flow confidence based on the COIN measure. As discussed in Chapter 2, a proper confidence measure should be free of the scaling ambiguity. Therefore, $\vec{b}$ is normalized to unit norm and A is scaled accordingly, if $\left\|\vec{b}\right\|_2 \neq 0$ (if $\left\|\vec{b}\right\|_2 = 0$, A and $[A|\vec{b}]$ must have the same rank). Therefore our confidence measure is defined as

$$\kappa = \begin{cases} 1, & \text{if } \left\|\vec{b}\right\|_2 = 0. \\ 1 - \frac{m}{\left\|\vec{b}\right\|_2^2}, & \text{otherwise.} \end{cases} \tag{3.17}$$

where m is the COIN measure. By Corollary 2, it can be verified that $\kappa \in [0, 1]$, where a small $\kappa \to 0$ indicates the low confidence of the computed flow.

Different from the confidence measures reviewed in Section 2.2.3, κ measures the flow confidence by the non-existence of the system solution. The *coherency measure* and *corner measure* (Haussecker *et al.* 1998) serve the same purpose, however, they are specific to 3D structure tensors.

Research has also been carried out on measuring the certainty of a flow vector *after* flow computation. In (Ohta 1990) and (Bruhn and Weickert 2006), such certainty measures are defined as the inverse of specially designed energy terms associated with particular optical flow techniques. In (Kondermann *et al.* 2007) and (Kondermann *et al.* 2008), the fidelity of a computed flow vector is measured by its conformity to neighbouring flow vectors, according to a linear subspace or a statistical model. These measures, although they can be applied to the flow obtained by any method, require training

data to learn the parameters of the models. Moreover, they can only be applied to dense flow field computation. The normalized COIN measure predicts the reliability of a flow vector before it is recovered, which is important for the algorithm to discard inappropriate constraints before flow computation. Furthermore, it does not require dense precomputation of flow vectors in the neighbourhood.

3.4 Experimental Results

3.4.1 Measuring Flow Confidence

The normalized COIN measure is compared to other classical confidence measures on benchmark sequences[4]: **RubberWhale**, **Hydrangea**, **Grove2**, **Grove3** (Baker *et al.* 2007), **Short Street** (McCane *et al.* 2001) and **Yosemite**. The measures are least significant eigenvalue (Lucas and Kanade 1981), determinant (Barron *et al.* 1994), inverse condition number (Uras *et al.* 1988) and the corner measure (Haussecker *et al.* 1998), with large value indicating high confidence.

To have a fair comparison with the corner measure, which is specific to the 3D structure tensor, Lucas-Kanade's flow technique is employed to recover the flow field. The quantitative evaluation follows the routine of "sparsification" used in (Bruhn and Weickert 2006) and (Kondermann *et al.* 2008). That is, given the confidence scores over the whole image, the $n\%$ ($n = 0, \cdots, 99$) pixels that have the lowest scores are removed, and the average flow error of the remaining pixels are computed.[5] The end-point error e at a pixel (i, j) is given by the Euclidean distance between the true flow $\vec{g}(i, j)$ and the computed flow $\vec{v}(i, j)$, i.e.,

$$e(i, j) = \|\vec{g}(i, j) - \vec{v}(i, j)\|_2 .$$

If the scoring scheme works well, the removal is expected to decrease the average error of the remaining flow vectors. In the ideal case, pixels that have the smallest confidence scores are exactly the pixels that have the largest errors. This ideal case provides an optimal score $C_{opt}(i, j) = 1 - e(i, j) / \max_{(i,j)}(e(i, j))$ as a benchmark (Kondermann *et al.* 2008).

[4] Properties of each test sequence used in this thesis are described with details in Appendix A

[5] On RubberWhale and Hydrangea, some pixels' ground truth flows are unknown, which are coded as the black color. These pixels are excluded from the reported error statistics.

As some measures (the least significant eigenvalue, determinant and inverse condition number) depend on the ill-condition of the system (the aperture problem), whereas others (the corner measure and the normalized COIN) depend on the inconsistency of the system (the motion boundary problem), it seems that different choices of the local patch size may favor different measures – a small patch is more likely to suffer the aperture problem than the motion boundary problem, whereas a large patch is in the opposite situation. To test this factor, the algorithm is implemented by varying the patch size from 7×7 to 19×19 gradually. The experiments show that on all sequences except **RubberWhale**, the ranking of different measures is not affected by the patch size selection. On **RubberWhale** sequence, the comparison between the corner measure and the normalized COIN measure is slightly affected by the size of the patch, when $n > 70\%$. Fig. 3.1 plots the average errors during the sparsification course on **RubberWhale**, **Hydrangea** and **Yosemite**, where flow is computed on 7×7, 13×13 and 19×19 patches. As the patch size does not affect the comparison on most sequences, Fig. 3.2 presents the results on **Grove2**, **Grove3** and **Street** for patch size 7×7 only.

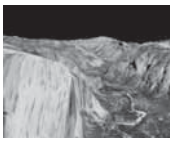
These experimental results show that the normalized COIN measure outperforms the other measures uniformly on **Hydrangea**, **Grove2**, **Grove3** and **Street**. On **RubberWhale**, the normalized COIN measure is only inferior to the corner measure over a portion $(< 30\%)$ of the whole range when the local patch size is 7×7. With the patch size increased, this portion is reduced, and the performance difference gets smaller. When the patch size reaches to 19×19, the normalized COIN measure performs better. On **Yosemite**, most measures (except the inverse condition number) perform comparably. Most likely this is because **Yosemite** does not contain as much motion inconsistency as other sequences. Experiments are also conducted on **Urban2** and **Urban3**. However, the motion of these two sequences are too large to be recovered by the Lucas-Kanade method with reasonable accuracy, so the confidence measure on the computed flow does not make much sense.

3.4.2 Motion Boundary Detection

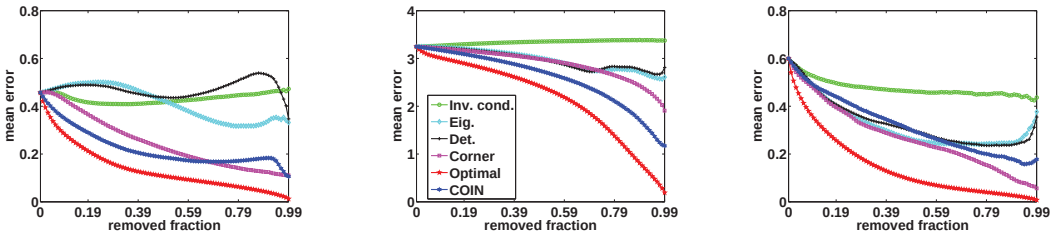
To quantitatively evaluate the performance of the COIN measure on motion boundary detection, it is necessary to compare the detected motion boundary to the ground truth. Instead of manually labeling the frames, in this work the ground truth is generated from the optical flow of benchmark sequences. Specifically, at each pixel (i, j), the

RubberWhale

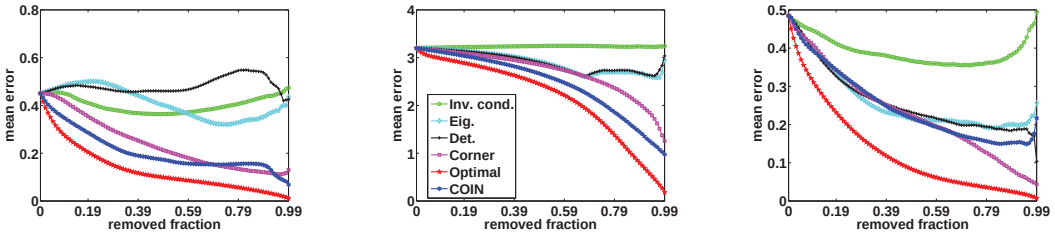
Hydrangea



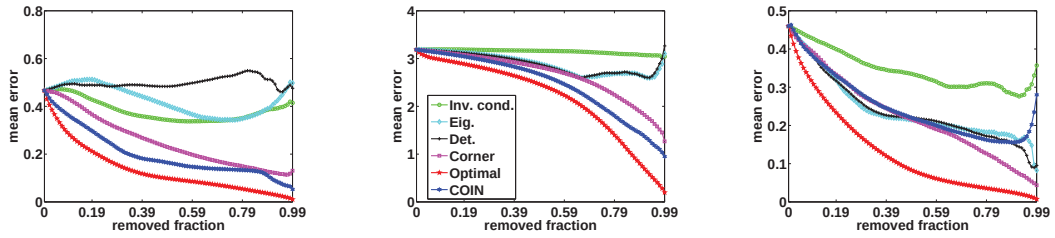
Yosemite



local patch size 7×7



local patch size 13×13



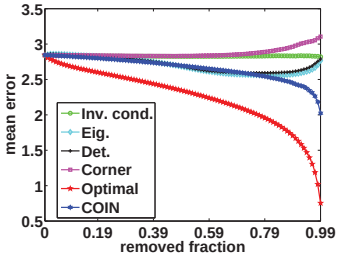
local patch size 19×19

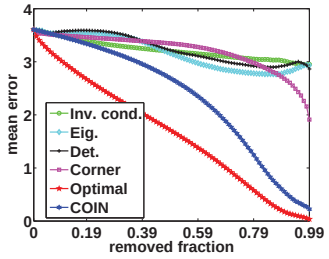
Figure 3.1. Comparison of various measures on **RubberWhale**, **Hydrangea** and **Yosemite**. The lower curve indicates better performance, where the lowest one corresponds to the theoretically ideal performance. All plots have the same legends.

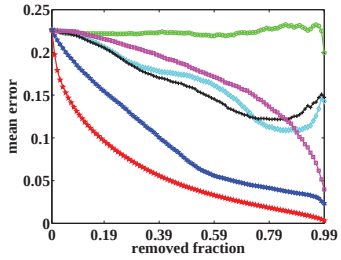
Grove2

Grove 3

Short Street







local patch size 7×7

Figure 3.2. Comparison of various confidence measures on **Grove2**, **Grove3** and **Short Street**. The lowest curve indicates the theoretically ideal performance. The COIN measure achieves best performance on all three sequences.

spatial variation of the ground truth flow $(u(i,j), v(i,j))$ is computed by

$$\gamma(i,j) = |u_x(i,j)| + |u_y(i,j)| + |v_x(i,j)| + |v_y(i,j)| . \tag{3.18}$$

Next, $\gamma(i,j)$ is thresholded over the whole image, with the partial derivatives in Eq.3.18 are approximated by central differencing (forward or backward differencing at image boundaries). Fig 3.3 shows the ground truth flow and the motion boundary obtained from widely used benchmark sequences.

The COIN measure is compared to the rank-increase measure by Shechtman-Irani. Particularly, at each pixel, the 2D and 3D structure tensors are computed from the intensity. The COIN measure (Eq. 3.16) and the rank-increase measure are calculated from the tensors' eigen-systems. In both cases, a large value indicates a high level of motion inconsistency. The thresholds are varied for both measures, and the pixels that pass the thresholding are labeled as detected motion boundary pixels. The recall-precision rates (P.138, (Olson 2008)) are computed for both detection scores, as demonstrated in Fig 3.3. Experiments are conducted on **RubberWhale**, **Hydrangea**, **Grove2**, **Grove3**, **Urban2**, **Urban3** and **Short Street**.

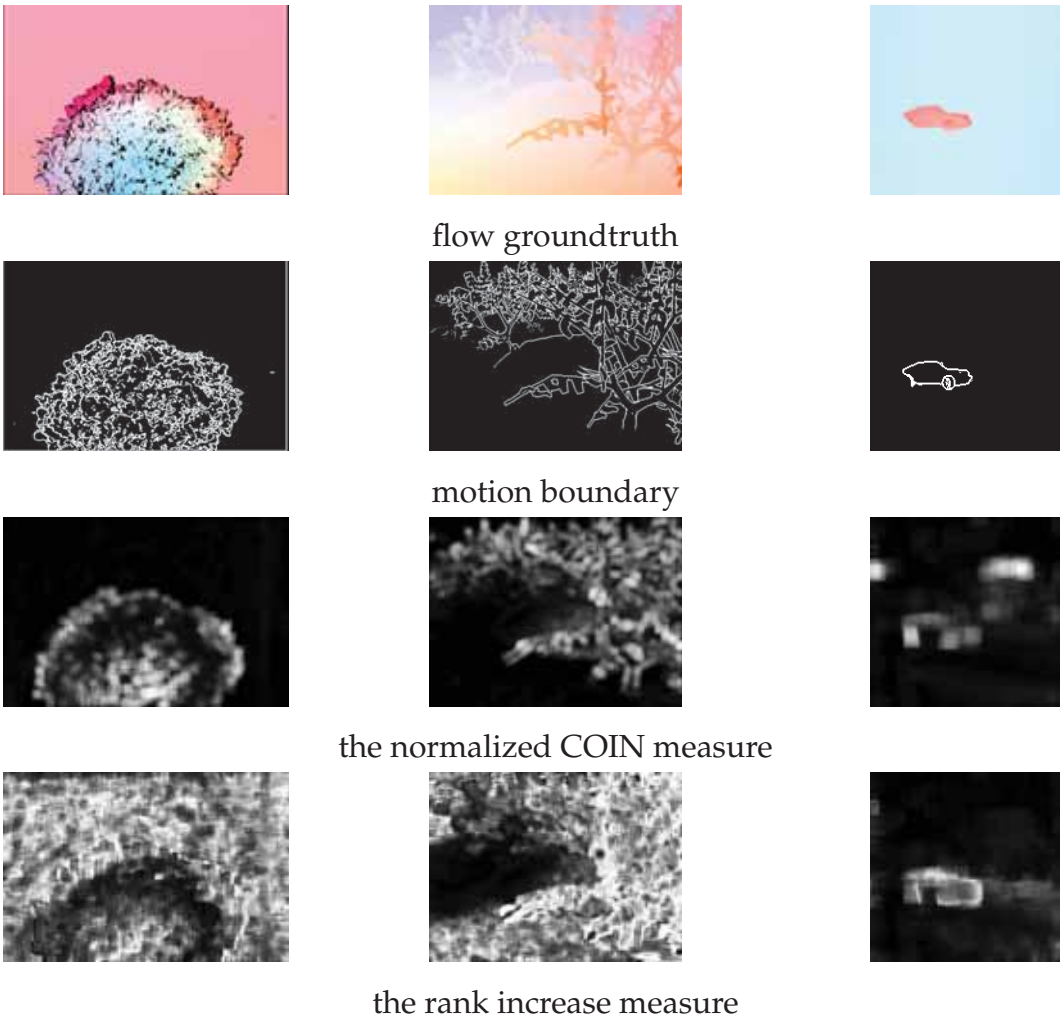


Figure 3.3. Motion boundary detection results on **Hydrangea**, **Grove3** and **Short Street**. First row: the color-coded flow ground truth; Second row: the motion boundaries obtained from the ground truth flow; Third and forth row: image maps of the normalized COIN measure and the rank increase measure; the intensity has been scaled for better illustration. Fifth row: The recall-precision curves of the motion boundary detection by the normalized COIN measure and the rank increase measure; the higher curve indicates better performance.

On all sequences except **Short Street**, the COIN measure achieves superior performance. Note that on sequence **Hydrangea**, the rank increase measure's detection is rather unsatisfactory. The reason is as analyzed in Section 3.3.3. The background in the sequence undergoes a motion at about 4 pixels/frame, but lacks texture compared to the moving hydrangea. This problem is avoided on sequence **Short Street**, as the background's velocity ($\approx$ 1 pixel/frame) is slower than the foreground ($\approx$ 2 pixels/frame), but it has richer spatial textures. Unlike the rank-increase measure, the COIN measure performs stably even when the camera undergoes fast motion.

3.5 Summary

This chapter has investigated how to find evidence of motion inconsistency from the intensity variation pattern of one, two or multiple pixels. The first part of the chapter has shown that, the presence of occlusion or motion boundaries may be signaled by the rank increase from a pixel's spatial Hessian matrix to its spatio-temporal Hessian matrix. The second part of the chapter has examined the pairwise motion inconsistency reflected in the eigen-systems of their spatio-temporal Hessian matrices. Finally, the third part of this chapter has proposed a continuous measure for inconsistency in linear system of equations. This measure, when applied to a system of linear optical flow constraints, can be used to predict the confidence with which flow can be recovered at each pixel. Analysis suggests and experimental results confirm that it outperforms other measures that have been previously used for this task, and that it can be used as a simple and effective motion boundary detector. The motion inconsistency detection methods proposed in this chapter can be used in later chapters to preserve motion boundaries in the framework of local optical flow computation.

Chapter 4

Discontinuity-Preserving Local Flow Computation

In the previous chapter, an inconsistency measure has been proposed to signal the presence of motion outliers in a local flow system. This chapter investigates how to inhibit or remove these outliers from a detected system. To this end, a variety of weighting functions are studied to down-weight pixels that have inconsistent patterns such as spatial position, intensity, intensity variation and motion. As these weighting functions are not limited to a particular local computation model, analysis on them assists the research conducted in further chapters.

As has been seen in the previous chapters, the common dilemma that local computation methods face is the generalized aperture problem (Jepson and Black 1993). Therefore, to achieve dense flow computation with motion discontinuity preserved by local methods, the local region has to be large enough to contain adequate intensity variation. Thus operations such as motion boundary detection, outliers removal and robust estimation are necessary. In the previous chapter, the COIN measure is introduced to detect systems that suffer motion outliers. This chapter goes one step further by detecting the outliers and inhibiting their influence. Section 3.2 proposes a pixel-wise and a pairwise motion inconsistency detection methods, which can be developed to reject outliers in a local model (as will be verified experimentally in Chapter 5). However, as they require the eigen-decomposition of the spatio-temporal Hessian matrices, it is computationally expensive if the algorithm is implemented pyramidally (as explained later). Therefore in this chapter, several low cost weighting functions suitable for pyramidal implementation are studied. In particular, the optical flow constraints are weighted dynamically in the process of multi-level multi-stage warping. In each warping stage, various weighting functions are applied to weight down outliers indicated by inconsistent patterns such as spatial position, intensity, intensity variation or motion. These weight functions adapt the local computation model $A\vec{X} = \vec{b}$ to $WA\vec{X} = W\vec{b}$, where the diagonal matrix W collects the weights for each pixel. The least squares solution of the weighted system estimates the flow vector.

To facilitate the explanation and description, the discussion of the weighting functions is carried out on an example local computation model that assumes gradient constancy and the local affine motion. Nevertheless, these weighting functions are not limited to any particular local computation model. Moreover, they can be (and some of them have been) applied to global computation.

4.1 The Linear System Model

In this chapter, the local computation model assumes that within a local patch: 1) the 3D gradient of a pixel, i.e., $[\ E_x \quad E_y \quad E_t\ ]^T$ is preserved between two consecutive frames; 2) the flow field is locally affine.

Let k index the kth pixel of the patch (in scan order) at coordinate position $[x^{(k)}, y^{(k)}]$, $\nabla_3 E^{(k)}$ be the 3D gradient vector, and $[u^{(k)}, v^{(k)}]$ be the flow vector of the pixel. The

first assumption means

$$\frac{d\nabla_3 E^{(k)}}{dt} = \frac{\partial \nabla_3 E^{(k)}}{\partial x} u^{(k)} + \frac{\partial \nabla_3 E^{(k)}}{\partial y} v^{(k)} + \frac{\partial \nabla_3 E^{(k)}}{\partial t} = 0; \tag{4.1}$$

while the second assumption gives

$$\begin{aligned} u^{(k)} &= \alpha_1 x^{(k)} + \alpha_2 y^{(k)} + \alpha_3 \\ v^{(k)} &= \alpha_4 x^{(k)} + \alpha_5 y^{(k)} + \alpha_6, \end{aligned} \tag{4.2}$$

where α_i's are the affine motion parameters.

Denote the flow of the patch center by $(u^{(c)}, v^{(c)})$, and the coordinate position by $[x^{(c)}, y^{(c)}]$. Similar to the kth pixel, $(u^{(c)}, v^{(c)})$ can be expressed by

$$\begin{aligned} u^{(c)} &= \alpha_1 x^{(c)} + \alpha_2 y^{(c)} + \alpha_3 \\ v^{(c)} &= \alpha_4 x^{(c)} + \alpha_5 y^{(c)} + \alpha_6, \end{aligned} \tag{4.3}$$

Eq.4.2 and Eq.4.3 lead to

$$\begin{aligned} u^{(k)} &= u^{(c)} + \alpha_1 \Delta x^{(k)} + \alpha_2 \Delta y^{(k)} \\ v^{(k)} &= v^{(c)} + \alpha_4 \Delta x^{(k)} + \alpha_5 \Delta y^{(k)}, \end{aligned} \tag{4.4}$$

where $\Delta x^{(k)} = x^{(k)} - x^{(c)}$ and $\Delta y^{(k)} = y^{(k)} - y^{(c)}$. By substituting Eq.4.4 to Eq.4.1, a system of 3 equations for the patch center's motion $\vec{X} = \begin{bmatrix} u & v & \alpha_1 & \alpha_2 & \alpha_4 & \alpha_5 \end{bmatrix}$ is obtained at the kth pixel,

$$\begin{bmatrix} \frac{\partial \nabla_3 E^{(k)}}{\partial x} & \frac{\partial \nabla_3 E^{(k)}}{\partial y} & \frac{\partial \nabla_3 E^{(k)}}{\partial x}\Delta x^{(k)} & \frac{\partial \nabla_3 E^{(k)}}{\partial x}\Delta y^{(k)} & \frac{\partial \nabla_3 E^{(k)}}{\partial y}\Delta x^{(k)} & \frac{\partial \nabla_3 E^{(k)}}{\partial y}\Delta y^{(k)} \end{bmatrix} \vec{X} = -\frac{\partial \nabla_3 E^{(k)}}{\partial t}. \tag{4.5}$$

Or in an extended form

$$\begin{bmatrix} E_{xx}^{(k)} & E_{xy}^{(k)} & E_{xx}^{(k)}\Delta x^{(k)} & E_{xx}^{(k)}\Delta y^{(k)} & E_{xy}^{(k)}\Delta x^{(k)} & E_{xy}^{(k)}\Delta y^{(k)} \\ E_{xy}^{(k)} & E_{yy}^{(k)} & E_{xy}^{(k)}\Delta x^{(k)} & E_{xy}^{(k)}\Delta y^{(k)} & E_{yy}^{(k)}\Delta x^{(k)} & E_{yy}^{(k)}\Delta y^{(k)} \\ E_{xt}^{(k)} & E_{yt}^{(k)} & E_{xt}^{(k)}\Delta x^{(k)} & E_{xt}^{(k)}\Delta y^{(k)} & E_{yt}^{(k)}\Delta x^{(k)} & E_{yt}^{(k)}\Delta y^{(k)} \end{bmatrix} \vec{X} = - \begin{bmatrix} E_{xt}^{(k)} \\ E_{yt}^{(k)} \\ E_{tt}^{(k)} \end{bmatrix}. \tag{4.6}$$

Briefly, Eq.4.5 can be written as $A^{(k)}\vec{X} = \vec{b}^{(k)}$, and the collection of all the linear constraints for $\vec{X}$ at all pixels form a linear system in the form of $A\vec{X} = \vec{b}$.

4.2 Weighting Functions for the Local Model

Given that the patch size is large enough to generate an overdetermined system, if all pixels have the same motion parameter $\vec{X}$ (with a small amount of noise allowed), the standard Least Squares Error (LSE) or Least Absolute Error (LAE) solutions of the system generally estimates $\vec{X}$ robustly. However, if the motion of some pixels is significantly different from $\vec{X}$, the system is invalid and results in severely skewed estimation. Unfortunately, such distortion happens frequently around motion boundaries. To improve the computation accuracy in such areas, the linear system should inhibit the constraints by pixels that have a motion pattern inconsistent with $\vec{X}$. One approach to the inhibition is to weight the pixels differently, by finding evidence for motion inconsistency from the intensity variation. Preferably, the kth pixel should be weighted heavily if its motion is consistent with the patch center; and be weighted lightly otherwise. The simplest weighting function is based on the kth pixel's distance from the patch center (e.g., (Tomasi and Kanade 1991)). More sophisticated schemes base the weighting functions on the intensity affinity (e.g., (Ren 2008)), motion affinity (e.g., (Niu *et al.* 2008)), occlusion and motion boundary indexing (e.g., (Niu *et al.* 2007),(Sand and Teller 2006)). In this work, we are going to look at 6 different possibilities for applying weighting functions to the individual system $A^{(k)}\vec{X} = \vec{b}^{(k)}$.

4.2.1 Spatial Proximity Term

It has been a common practice to weight a pixel by its spatial distance from the patch center (p.197, (Trucco and Verri 2006)). Following this convention, the first weighting function studied in this chapter is based on the normal distribution of the spatial proximity. In particular, the weight for the kth pixel is given by

$$\omega^{(k)}_{distance} = \exp\left(-\frac{(\Delta x^{(k)})^2 + (\Delta y^{(k)})^2}{2\sigma_s^2}\right), \tag{4.7}$$

where $\sigma_s = 4$ in this work.

4.2.2 Pairwise Intensity Affinity Term

Significant intensity contrast between neighbouring pixels is an important cue for object boundary. Therefore it has been used widely in previous works for the weighting/segmenting purpose. In (Niu *et al.* 2007), the kth pixel is weighted by a hard

thresholding function

$$\omega^{(k)} = \begin{cases} 1 & \text{if } \left| I^{(k)} - I^{(c)} \right| < T \\ 0 & \text{otherwise.} \end{cases} \tag{4.8}$$

where $I^{(k)}$ and $I^{(c)}$ are the intensity values of the kth pixel and the patch center, and T is $1/8$ of the total intensity levels summed over the color channels. In (Sand and Teller 2006), in the context of bilateral filtering, a continuous weight function in the form of normalized distribution is given by

$$\omega^{(k)} = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(I^{(k)} - I^{(c)} \right)^2}{2\sigma^2} \right) \tag{4.9}$$

where the standard deviation σ is defined to be 20. In (Ren 2008), a soft affinity term is defined as an exponential function of the boundary energy along the path linking the two pixels. This affinity term, as it requires *probability-of-boundary* detection (Martin *et al.* 2004) to find the boundary, is computationally more expensive than the direct comparison of intensity values. In this work, combining (Niu *et al.* 2007) and (Sand and Teller 2006), the pairwise intensity affinity term is defined by

$$\omega^{(k)}_{intensity} = \begin{cases} \exp\left(-\frac{\left| I^{(k)} - I^{(c)} \right|}{\sigma} \right) & \text{if } \left| I^{(k)} - I^{(c)} \right| < 2\sigma \\ 0 & \text{otherwise.} \end{cases} \tag{4.10}$$

This weighting term is basically a continuous function with hard thresholding. The σ is defined to be $1/16$ of the the total intensity levels summed over the color channels. For example, $\sigma = 16$ for 8-bit gray-scale sequences.

4.2.3 Dynamic Occlusion and Boundary Detection

In order to handle large displacement, contemporary optical flow techniques generally implement the computation algorithm in the manner of pyramidal coarse-to-fine iterative refinement. Particularly, the Gaussian pyramid pair are built by subsampling the original consecutive frames. Initialized by zeros, the flow computation starts from the coarsest level and is propagated to the finer levels. Furthermore, in each level, the flow is refined by multiple stages. In each stage τ, the local patch (in the second subsampled image) centered at a pixel (x, y) is warped according to the current flow $(u^\tau(x, y), v^\tau(x, y))$. A refined temporal difference δE_t^τ is obtained between the patch

in the first image and the warped patch, which is input to the computation system and yields the flow refinement $(\delta u^\tau(x,y), \delta v^\tau(x,y))$. The flow at stage $\tau+1$ is then updated by

$$u^{\tau+1}(x,y) = u^\tau(x,y) + \delta u^\tau(x,y), \quad , v^{\tau+1}(x,y) = v^\tau(x,y) + \delta v^\tau(x,y).$$

Flow recovered in stage τ thus can be utilized to interpret the occurrence of occlusion or motion boundaries and helps reject outliers in stage $\tau+1$, as described in the following sub sections.

Occlusion Detection

Significant temporal intensity variation is generally assumed to be a consequence of occlusion, and therefore $|E_t|$ has been widely used to detect occlusion (e.g.,(Xiao *et al.* 2006),(Sand and Teller 2006)). However, this heuristic is not always true, as significant $|E_x|$ or $|E_y|$ may also lead to large $|E_t|$, even when the object is under a simple constant translation against a static background. (Niu *et al.* 2008) proposes using the normalized term $|E_t| \,/\, \left(|E_x| + |E_y|\right)$, which is bounded by $\max(|u|, |v|)$, given the brightness constancy assumption is valid. The occlusion likelihood weighting term is defined by:

$$\omega^{(k,\tau)}_{occlusion} = \begin{cases} 0 & \text{if } \left|\delta E_t^{(k,\tau)}\right| \,/\, \left(\left|E_x^{(k)}\right| + \left|E_y^{(k)}\right|\right) > T^\tau \\ 1 & \text{otherwise,} \end{cases} \tag{4.11}$$

The threshold T^τ can be given if prior knowledge of the maximum image velocity is available. In this work, as the flow computation algorithm is implemented in the manner of pyramidal coarse-to-fine refinement, where the increase of the flow should be smaller than 1 pixel, T^τ is defined to be 1.

In the framework of variational flow computation, occlusion detection based on intermediate flow has been discussed in (Sand and Teller 2006), which proposes using the positiveness of the flow divergence, i.e.,

$$\mathrm{div}(u,v) = u_x + v_y, \tag{4.12}$$

to label pixels affected by occlusion. This detector is studied in this work in the context for local flow computation. Following (Sand and Teller 2006), the divergence function is defined by

$$d^\tau(x,y) = \begin{cases} \mathrm{div}(u^\tau, v^\tau) & \text{if } \mathrm{div}(u^\tau, v^\tau) < 0 \\ 0 & \text{otherwise,} \end{cases} \tag{4.13}$$

The weighting function is defined by the Gaussian distribution function

$$\omega_{divergence}^{(k,\tau)} = \exp\left(-\frac{\left(d^{\tau}(x^{(k)}, y^{(k)})\right)^2}{2\sigma_d^2}\right) \tag{4.14}$$

where $\sigma_d = 0.3$, as defined in (Sand and Teller 2006). It is worth noting that the divergence of 2D velocity is a measure of the shape expansion (Cipolla and Blake 1997), therefore it is only suitable to detect the occlusion caused by scale change. The experimental results presented in Section 4.3 also illustrate this limitation.

Motion Boundary Detection

In (Niu *et al.* 2007), large temporal variation of the flow, i.e., $|u|_t + |v|_t$, is proposed as a low cost indicator of the motion boundary; and the residual error $|E_x u + E_y v + E_t|$ at each pixel is used to signal motion discontinuity during the dynamic selection of temporal neighbours. In this work, hard-thresholding weighting functions based on the detectors in (Niu *et al.* 2007) is studied.

$$\omega_{residual}^{(k,\tau)} = \begin{cases} 0 & \text{if } \left|E_x^{(k)} u^{(k,\tau)} + E_y^{(k)} v^{(k,\tau)} + E_t^{(k)}\right| > T^{\tau} \\ 1 & \text{otherwise,} \end{cases} \tag{4.15}$$

where T^{τ} is determined by the standard deviation of the residual errors over the whole image.

The difference between the flow recovered at the kth pixel and the patch center c reveals the motion inconsistency directly. This idea has been widely explored in many variational flow computation algorithms for flow-driven regularization (see Section 2.3.1); however, it has not been discussed for local computation. In this work, the weighting function based on this motion inconsistency indicator is studied, which is defined as below,

$$\omega_{consistency}^{(k,\tau)} = \exp\left(-\frac{(u^{(k,\tau)} - u^{(c,\tau)})^2 + (v^{(k,\tau)} - v^{(c,\tau)})^2}{2\sigma_m^2}\right). \tag{4.16}$$

Here σ_m is predefined to be 1, following (Sand and Teller 2006).

4.3 Performance Evaluation

In order to examine their numerical performance, the 6 weighting functions are tested on a variety of benchmark sequences **RubberWhale**, **Hydrangea**, **Dimetrodon**, **Grove2**,

Venus and **Long Street**. The weighted local computation is implemented in the pyramidal fashion as explained in Section 4.2.3.

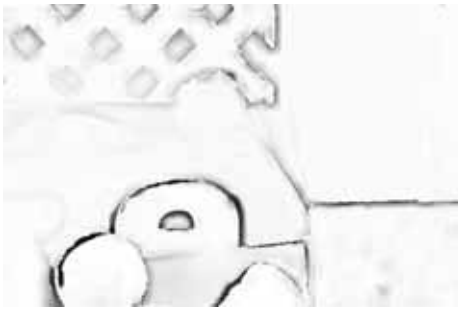
Table 4.1 lists the quantitative flow computation results measured by AAE, with the bold numbers indicating improvement of the corresponding weighting function on a particular sequence. The comparison of before-and-after applying each weighting function shows $\omega_{consistency}$ improves the flow computation accuracy on all the sequences. $\omega_{residual}$ works well on most sequences, except for **Dimetrodon**. The other dynamic weighting function $\omega_{divergence}$, which is based on the divergence of the recovered flow, only show slight improvement on sequence **Grove2**. Although not significant, improvement has been obtained by $\omega_{occlusion}$ on 4 out of 6 test sequences. $\omega_{intensity}$ remarkably reduces the error on **RubberWhale**, although a case-by-case tuned threshold may possibly achieve better accuracy on other sequences. It is interesting to note that, although weighting functions based on spatial proximity has been widely applied, the comparison conducted here shows that $\omega_{proximity}$ does not necessarily lead to better performance on all the sequences.

By comparing the responses of the sequences to the weighting functions, it can be seen that **Grove2** and **RubberWhale**, which contain the most substantial motion discontinuities among the test sequences, respond well to almost all the weighting functions. Sequence **Long Street** and **Hydrangea** only slightly respond to $\omega_{residual}$, $\omega_{occlusion}$ and $\omega_{consistency}$. Presumably, this is because in both sequences the large area of background undergoes smooth translation, and does not contain as much motion discontinuities as **Grove2** and **RubberWhale**. **Venus** responds significantly to $\omega_{residual}$ and $\omega_{consistency}$. The non-rigid motion sequence **Dimetrodon** responds to almost none of these weighting functions. Fig. 4.1 and Fig. 4.2 demonstrate the simulation results on benchmark sequences **RubberWhale** and **Grove2**. The quantitative performance evaluated by the average angular error (AAE) shows considerable improvement by the weighting function on **RubberWhale** sequence. The gray-scale coded angular error maps also confirm that the motion discontinuity is better preserved by the weighting function.

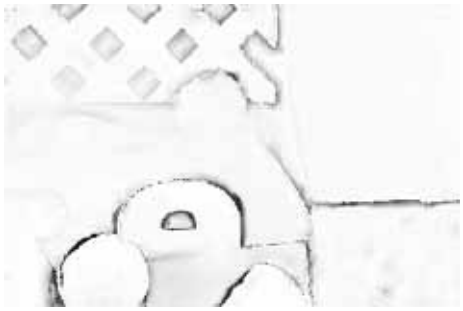
Due to the uniform good performance of $\omega_{residual}$ and $\omega_{consistency}$ and their robustness to the parameter selection, these two weighting functions are suggested for local flow computation. Although compared to these two functions, the improvement by $\omega_{occlusion}$ is to a less extent, it is also proposed as an option. Furthermore, the substantial improvement by $\omega_{intensity}$ on **RubberWhale** and **Grove2** suggests that this weighting function may significantly reduce distortion effects, if the parameter in Eq. 4.15

RubberWhale

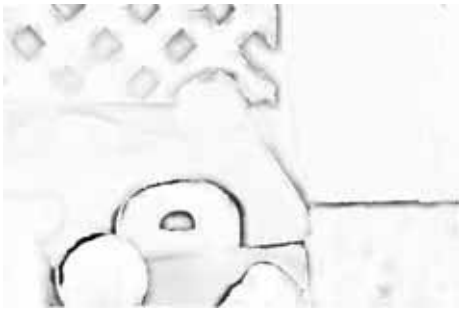
ground truth flow



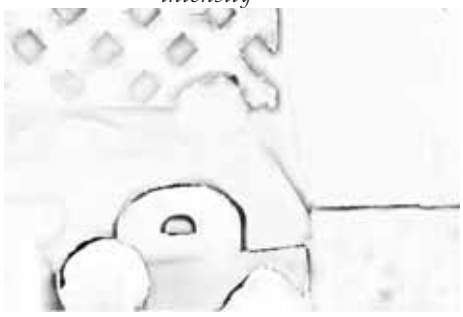
$\omega^{(k)} = 1$, AAE = 6.52



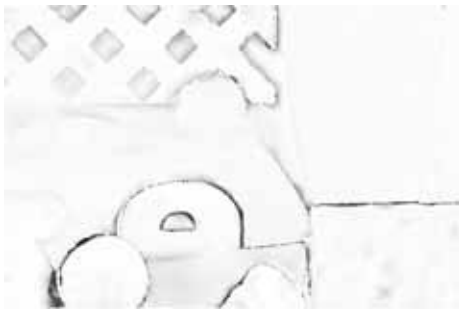
$\omega^{(k)} = \omega^{(k)}_{intensity}$, AAE = 5.47



$\omega^{(k)} = \omega^{(k)}_{residual}$, AAE = 6.26



$\omega^{(k)} = \omega^{(k)}_{consistency}$, AAE = 5.92



$\omega^{(k)} = \omega^{(k)}_{intensity} \cdot \omega^{(k)}_{residual} \cdot \omega^{(k)}_{consistency}$, AAE = 5.01

Figure 4.1. The performance of the weighting functions on **RubberWhale**. Motion boundaries are better preserved. Darker colour indicates higher average angular error (AAE) value.

Grove2

ground truth flow

$\omega^{(k)} = 1$, AAE = 3.97

$\omega^{(k)} = \omega^{(k)}_{intensity}$, AAE = 3.57

$\omega^{(k)} = \omega^{(k)}_{residual}$, AAE = 3.79

$\omega^{(k)} = \omega^{(k)}_{consistency}$, AAE = 3.63

$\omega^{(k)} = \omega^{(k)}_{intensity} \cdot \omega^{(k)}_{residual} \cdot \omega^{(k)}_{consistency}$, AAE = 3.30

Figure 4.2. The performance of the weighting functions on **Grove2**. Darker colour indicates higher average angular error (AAE) value.

	Venus	Dimetrodon	Street	RubberWhale	Hydrangea	Grove2
$\omega = 1$	9.42	2.59	3.57	6.52	4.22	3.97
$\omega_{proximity}$	9.58	2.96	4.36	**5.49**	4.47	**3.44**
$\omega_{intensity}$	**9.26**	2.66	3.85	**5.47**	4.41	**3.57**
$\omega_{occlusion}$	9.45	2.74	**3.53**	**6.36**	**4.13**	**3.85**
$\omega_{residual}$	**8.01**	2.67	**3.41**	**6.26**	**4.17**	**3.79**
$\omega_{divergence}$	9.71	2.60	3.57	6.56	4.25	**3.90**
$\omega_{consistency}$	**8.19**	**2.57**	**3.53**	**5.92**	**4.19**	**3.63**
combined	**7.86**	2.80	3.57	**5.01**	4.56	**3.30**

Table 4.1. An overview of the performance of different weighting functions on a variety of benchmark sequences, measured by the AAE achieved by using each weighting function. The bold numbers indicate that improvement is achieved by the corresponding weight function on the particular sequence. The last row shows the results obtained by combining $\omega_{intensity}$, $\omega_{residual}$ and $\omega_{consistency}$.

is properly defined, which is possible if the statistics of the image can be obtained. Based on the analysis, the final weighting function in this section is defined by the combination (multiplication) of $\omega_{intensity}$, $\omega_{residual}$ and $\omega_{consistency}$. Table 4.1 shows the performance on benchmark sequences. The individual constraint system provided by the kth pixel is thus adapted to $\omega^{(k)} A^{(k)} \vec{X} = \omega^{(k)} \vec{b}^{(k)}$; and the overall weighted linear system $WA\vec{X} = W\vec{b}$, where matrix W is diagonal and collects the weights, recovers the flow field by the standard LSE if the system is overdetermined. However, it should be noted that the weighted system can be rank-deficient, as pixels with distinguishable intensity pattern might be under-weighted. In this case, the window size is enlarged in the subsequent flow refinement steps. This is in a similar spirit to the adaptive window used in (Kanade and Okutomi 1991), which adjusts the local window size in each refinement step. In our empirical study, such a strategy is found to be sufficient to overcome the aperture problem and obtain a dense flow field.

4.4 Summary

Following the discussion in the previous chapter on how to detect inconsistent local systems, this chapter has investigated how to construct a consistent local computation system by selecting the neighbouring consistent pixels. 6 different possible selection (weighting) functions are empirically studied, focusing on (but not limited to) a local computation model that assumes gradient constancy and local affine motion. The selection functions based on the intensity similarity, dynamic residual error and motion consistency are found to be effective. The combined use of these weighting functions is suggested for further research on local flow computation.

Chapter 5

Complementary Combination of Local and Global Constraints

The previous chapter proposed a motion discontinuity preserving local flow computation method. In this chapter, we show that the local computation can be further regularized by global constraints to achieve enhanced performance. In particular, the local flow estimation is used to segment the image into regions of smooth motion. Within each region, global constraints are applied to reduce noise in local flow estimates while preserving motion boundaries. The main contributions in this framework are: (1) combining the local and global computation by motion segmentation; (2) the complementary use of global subspace and spatial smoothness constraints. Results on standard test sequences demonstrate improved accuracy in flow estimation, and analyse the role that each contribution plays in this improvement.

The flow computation method described in the previous chapter belongs to the category of local computation, which recovers each pixel's flow by an independent local linear system, and hence neighbouring pixels' flow vectors do not interact in the computation.[6] To some extent, this motion recovery model does not conform to principles in real practice, where neighbouring pixels generally move consistently. Recovering optical flow by exploiting this spatial consistency is the category of global computation. In this chapter, the local flow constraints are combined with the global constraints, which achieves enhanced performance by exploring the complementary effects of local and global computation.

In particular, an initial recovery of each pixel's flow is obtained by a local system based on the brightness invariance assumption. The resulting flow field is typically recovered accurately enough to enable consistent segmentation of independently moving objects. On each segmented object, two global motion constraints, the subspace constraint and spatial smoothness constraint, are integrated to further regularize the flow field. As the global constraints are only applied to the interior of a moving object, error propagation from the motion boundaries is effectively avoided.

The chapter is divided into 3 sections, with each section introducing one of the three steps, namely local computation, motion segmentation and global regularization. In each section, focusing on the particular topic, research concerns and difficulties are explained, and possible solutions are presented. The computation methods proposed in each section can be applied singly or in combination. The role that each step plays is analyzed based on the experimental results.

5.1 Motion Outlier Inhibited Local Flow Computation

The pipeline starts from local computation, which is based on the brightness constancy assumption,

$$\frac{dE^{(k)}(x,y,t)}{dt} = E_x^{(k)}u^{(k)} + E_y^{(k)}v^{(k)} + E_t^{(k)} = 0, \tag{5.1}$$

where k indexes the kth pixel in the neighbourhood. By taking partial derivatives at both sides of Eq.(5.1) with respect to x, y, and t, an under-determined linear system for

[6] In local flow computation, a pixel's flow recovery is constrained by neighbouring pixels' intensity variation, but not the flow vectors.

Test frame recovery error a zoomed in view

Figure 5.1. The results of the local computation weighted by $\omega^{(k)}_{intensity} \cdot \omega^{(k)}_{occlusion} \cdot \omega^{(k)}_{residual} \cdot \omega^{(k)}_{consistency}$. The motion boundary between the blue car and the background is over-smoothed. Darker intensity indicates larger errors.

8 unknowns is yielded,

$$\begin{bmatrix} E^{(k)}_{xx} & E^{(k)}_{xy} & E^{(k)}_{x} & E^{(k)}_{y} & 0 & 0 & 0 & 0 \\ E^{(k)}_{xy} & E^{(k)}_{yy} & 0 & 0 & E^{(k)}_{x} & E^{(k)}_{y} & 0 & 0 \\ E^{(k)}_{xt} & E^{(k)}_{yt} & 0 & 0 & 0 & 0 & E^{(k)}_{x} & E^{(k)}_{y} \end{bmatrix} \vec{X}^{(k)} = - \begin{bmatrix} E^{(k)}_{xt} \\ E^{(k)}_{yt} \\ E^{(k)}_{tt} \end{bmatrix}. \tag{5.2}$$

where $\vec{X}^{(k)} = \left[u^{(k)}, v^{(k)}, u^{(k)}_x, v^{(k)}_x, u^{(k)}_y, v^{(k)}_y, u^{(k)}_t, v^{(k)}_t\right]^T$.

The local computation model also assumes that pixels in the local patch have similar flow vectors and flow gradient. This enables an over-determined system for the patch center's motion $A\vec{X} = \vec{b}$, which gathers Eq.(5.2) for all neighbouring pixels, with $\vec{X}$ containing the flow and flow gradient of the patch center c.

As discussed in previous chapters, the system is likely to contain motion outliers at the presence of motion boundaries. This entails motion outlier detection and removal. The combination (multiplication) of the weighting functions $\omega_{intensity}$, $\omega_{occlusion}$, $\omega_{residual}$ and $\omega_{consistency}$ suggested in the previous chapter, as they are general to any local system, is applied in this section. Fig. 5.1 demonstrates the results of the weighted local system on the sequence **Long Street**. It can be seen that, although the motion boundary between the red car and the background is reasonably clear, the motion boundary between the blue car and the background is still over-smoothed.

Because of this, the motion inconsistency detection theories introduced in Chapter 3 are developed to two new weighting functions. Particularly, the first motion outlier rejection function ω_{rank} is defined by the rank difference between the spatial Hessian

matrix and the spatio-temporal Hessian matrix of the kth pixel,

$$\omega_{rank}^{(k)} = \begin{cases} 0 & \text{if } \mathrm{rank}(H_{3D}^{(k)}) > \mathrm{rank}(H_{2D}^{(k)}) \\ 1 & \text{otherwise} \end{cases} \tag{5.3}$$

which indicates the contamination of pixel k by the inconsistent motion.

The other rejection function $\omega_{intersection}$ is defined by the lack of non-trivial intersection of the solution manifolds of Eq. 5.2 associated with c and k respectively (see Section 3.2 for the detection strategies),

$$\omega_{intersection}^{(k)} = \begin{cases} 0 & \text{if } \Omega^{(c)} \cap \Omega^{(k)} = \left\{\vec{0}\right\} \\ 1 & \text{otherwise,} \end{cases} \tag{5.4}$$

where $\Omega^{(c)}$ and $\Omega^{(k)}$ denote the solution manifolds of the two pixels.

The final weighting function $\omega^{(k)}$ of the kth pixel is given by

$$\omega^{(k)} = \omega_{intensity}^{(k)} \cdot \omega_{occlusion}^{(k)} \cdot \omega_{residual}^{(k)} \cdot \omega_{consistency}^{(k)} \cdot \omega_{rank}^{(k)} \cdot \omega_{intersection}^{(k)}. \tag{5.5}$$

Fig.5.2 shows the results on **Long Street** using the new weight defined by Eq. 5.5, which validates the improvement by ω_{rank} and $\omega_{intersection}$.

recovery error a zoomed in view

Figure 5.2. The recovery results by the local system weighted by $\omega_{intensity}^{(k)} \cdot \omega_{occlusion}^{(k)} \cdot \omega_{residual}^{(k)} \cdot \omega_{consistency}^{(k)} \cdot \omega_{rank}^{(k)} \cdot \omega_{intersection}^{(k)}$. Compared to Fig. 5.1, the zoomed in view clearly shows that the motion boundary around the car is better preserved.

The local computation is implemented using iterative refinement. If the weighted system is rank-deficient in any refinement stage, the neighborhood is enlarged in the subsequent refinement/warping steps. This is in a similar spirit to (Kanade and Okutomi 1991).

5.2 Flow Field Segmentation

The local flow computation recovers each pixel's displacement independently. On one hand, if the recovery at one pixel is erroneous, the error does not affect the recovery at other pixels. On the other hand, the motion coherence of pixels in the same moving object is not exploited. In (Ohta 1991), it has been shown that a global flow stabilization improves the local flow computation in textured regions. In (Birchfield and Pundlik 2008), the motion coherence has been utilized to track corners and edges jointly, and evident improvement over the traditional Lucas-Kanade independent feature tracking has been reported. In this work, to take advantage of the global regularization and avoid over-smoothing across motion boundaries, an motion segmentation step is implemented. This is to make sure that global constraints are only applied to pixels that *do* have consistent motion pattern. The scene is segmented to independently moving objects based on the flow field obtained by local computation. It involves three steps: measure the confidence of the computed flow vectors; estimate motion models by mean shift clustering of the flow vectors with high fidelity; label all the pixels by model compatibility.

Flow vectors recovered by local methods have different fidelity at different pixels. As discussed above, in smooth regions or near motion boundaries, local flow estimation is error prone. Consequently, it is not robust to estimate the number of motion models and motion parameters if the unreliable flow vectors are involved. Therefore, in this work only flow vectors with high fidelity are picked up for segmentation. In our local computation model, a necessary condition for a pixel's flow to be recovered reliably is that the associated Hessian matrices (both 2D and 3D) have rank 2, which implies that the pixel has sufficient spatial information and consistent motion through consecutive frames. In addition, as it is assumed that pixels in the local patch have similar flow, the deformation and acceleration parameters in Eq. 5.2 should be small. That is, if we rewrite Eq. 5.2 to

$$H_{3D}\vec{V} = -\begin{bmatrix} u_x & v_x \\ u_y & v_y \\ u_t & v_t \end{bmatrix}\begin{bmatrix} E_x \\ E_y \\ E_t \end{bmatrix} = \vec{f}, \tag{5.6}$$

then $\left\|\vec{f}\right\|$ should vanish. Conversely, large magnitude of deformation and acceleration generally signals motion inconsistency. Therefore, pixels with rank 2 Hessian matrices (both 3D and 2D) and $\left\|\vec{f}\right\|$ below a threshold, which are referred to as *flow features*, are selected for segmentation.

Mean-shift clustering algorithm (Comaniciu and Meer 2002), which has been widely applied to image segmentation, is run for the selected flow features. Particularly, the flow features are first clustered according to the angular closeness measured by the normalized inner product of the flow vectors, with the cluster bandwidth limited to $\pi/6$. This yields clusters of flow vectors with similar velocity direction. Within each cluster, the mean shift algorithm is further applied according to the magnitude similarity measured by the Euclidean distance between flow vectors, with the bandwidth of each cluster limited to 0.5; Let N be the number of clusters obtained, and C_i be the ith cluster, the center of each cluster $[u_i, v_i]$ is taken as a motion model.

The flow vector of an arbitrary pixel $[u, v]$ is then labeled according to its closeness to the models. More specifically, motion models whose angular difference from the flow vector is larger than $\pi/6$ are first excluded. Among the remaining M clusters $C_j, (j = 1, \cdots, M)$, the magnitude difference d_j between $[u, v]$ and $[u_j, v_j]$ is computed. The pixel is labeled by cluster j_0, if $j_0 = \arg\min d_j$. To remove the noise effect in each cluster, a smooth filtering is applied to the segmented objects.

Fig.5.3 shows the segmentation result on a benchmark sequence **Long Street**, which depicts a street scene with two cars moving into the image and the camera moving to the left. The segmentation extracts the cars from the rest of the scene with the boundary sharply preserved. Despite the transparency of the right car's front window, the corresponding region is successfully segmented as the background.

5.3 Global Regularization

Global regularization of the flow obtained by local methods can further improve the accuracy. However, as pointed in (Ohta 1991), the significance of the improvement varies from region to region. In textured area, where local computation generally has reliable performance, the accuracy can be further improved by global stablization. However, in smooth areas where local computation suffers the aperture problem, the improvement is less obvious. The reason is due to the intrinsic weakness of the global smoothing, which propagates the error in one flow vector to adjacent ones during the diffusion-reaction process. In (Irani 2002), a global *subspace constraint* is proposed to address the aperture problem. The root of the subspace constraint is that, in case of

Test Frame

Segmented Object 1

Segmented Object 2

Segmented Object 3

Figure 5.3. The three objects segmented from the benchmark sequence **Long Street** by mean shift clustering of the flow obtained from local computation.

static scene and moving camera, the tracked positions (Tomasi and Kanade 1992)/trajectories (Irani 2002) of multiple points across multiple frames reside in low rank subspaces. Compared to the global *smoothness* constraint, the subspace constraint is able to relate the error-prone flow vectors to the reliable ones rather than the spatially proximate ones. However, despite this advantage, the subspace constraint is far less studied and applied in the literature of optical flow computation. This is partly due to the strict requirement of the scene to be static and the sequence to be long-range. In this work, we derive a new subspace constraint for the matrix formed by flow vectors of all points across multiple frames. As the scene has been segmented into regions with consistent motion, the relative motion between each individual object and the camera can be taken as that the object is static and the camera undergoes the compound motion. Therefore, the subspace constraint can be applied to each individual object. With the erroneous flow vectors corrected by the reliable ones, the spatial smoothness constraint is further applied to the low rank subspace to exploit the flow connection between spatially adjacent pixels. The combination of the two global constraints is achieved by minimizing an integrated energy functional in the subspace.

5.3.1 Global Subspace Constraint

Subspace constraint of the trajectory matrix

In (Irani 2002), it is shown that the displacement of N points across F frames reside in a low rank subspace. Particularly, let $\vec{P}_{i,t} = [x_{i,t}\ y_{i,t}]^T$ denote the position of the ith pixel of an object at time t, and $\vec{d}_{i,j} = \left[d^x_{i,j}\ d^y_{i,j}\right]^T$ be the pixel's displacement from the reference frame $(t = 0)$ to the current frame $(t = j)$. Thus $\vec{d}_{i,j}$ can be represented by

$$\vec{d}_{i,j} = \vec{P}_{i,j} - \vec{P}_{i,0},$$

or equivalently

$$\begin{bmatrix} d^x_{i,j} \\ d^y_{i,j} \end{bmatrix} = \begin{bmatrix} x_{i,t} - x_{i,0} \\ y_{i,t} - y_{i,0} \end{bmatrix}.$$

Define U and V by

$$U = \begin{bmatrix} d^x_{0,0} & d^x_{1,0} & \dots & d^x_{N,0} \\ d^x_{0,1} & d^x_{1,1} & \dots & d^x_{N,1} \\ \vdots & \vdots & \vdots & \vdots \\ d^x_{0,F} & d^x_{1,F} & \dots & d^x_{N,F} \end{bmatrix}, V = \begin{bmatrix} d^y_{0,0} & d^y_{1,0} & \dots & d^y_{N,0} \\ d^y_{0,1} & d^y_{1,1} & \dots & d^y_{N,1} \\ \vdots & \vdots & \vdots & \vdots \\ d^y_{0,F} & d^y_{1,F} & \dots & d^y_{N,F} \end{bmatrix}. \tag{5.7}$$

The trajectory matrix $\left[\frac{U}{V}\right]$ is actually of very low rank (≤ 9). The proof is based on a case by case discussion of the 3D motion model and the camera projection model.

New subspace constraint of the flow matrix

This section extends the theory to the long range flow field. We show that, for all points on the same moving object, the cumulative flow field across multiple frames also resides in a low rank space. Given the initial flow recovery for each object by local computation, $\vec{d}_{i,j}$ can be expressed as the sum of the flow vectors obtained from each pair of adjacent frames. Denote the flow at position $\vec{P}_{i,t}$ by $\vec{V}_t = \left[u_t\left(\vec{P}_{i,t}\right)\ v_t\left(\vec{P}_{i,t}\right)\right]^T$.[7] This relates the pixel's positions at time t and $t+1$ by

$$\vec{P}_{i,t+1} = \vec{P}_{i,t} + \vec{V}_t\left(\vec{P}_{i,t}\right). \tag{5.8}$$

Also,

$$\vec{P}_{i,t+1} = \vec{P}_{i,0} + \sum_{k=0}^{t} \vec{V}_k\left(\vec{P}_{i,k}\right). \tag{5.9}$$

[7] Generally $\vec{P}_{i,t}$ are not at integer pixels, where each frame's flow is calculated. We compute the flow vectors at sub-pixel positions by a bilinear interpolation.

Moreover,

$$\vec{d}_{i,t+1} = \vec{P}_{i,t+1} - \vec{P}_{i,0} = \sum_{k=0}^{t} \vec{V}_k \left(\vec{P}_{i,k} \right), \tag{5.10}$$

which represents $\vec{d}_{i,t+1}$ by an accumulation of flow vectors.

Define *flow matrix* F as

$$F = \begin{bmatrix} \vec{V}_0(P_{0,0}) & \vec{V}_0(P_{1,0}) & \dots & \vec{V}_0(P_{N,0}) \\ \vec{V}_1(P_{0,1}) & \vec{V}_1(P_{1,1}) & \dots & \vec{V}_1(P_{N,1}) \\ \vdots & \vdots & \vdots & \vdots \\ \vec{V}_F(P_{0,F}) & \vec{V}_F(P_{1,F}) & \dots & \vec{V}_F(P_{N,F}) \end{bmatrix}_{2F \times N}. \tag{5.11}$$

Each column of F records a pixel's flow "history" across all the frames, while each row of D collects the flow vectors of all pixels at a certain time. By elementary row summation, it can be easily seen that F can be transformed into the following matrix

$$F \sim \begin{bmatrix} \vec{d}_{0,0} & \vec{d}_{1,0} & \dots & \vec{d}_{N,0} \\ \vec{d}_{0,1} & \vec{d}_{1,1} & \dots & \vec{d}_{N,1} \\ \vdots & \vdots & \vdots & \vdots \\ \vec{d}_{0,F} & \vec{d}_{1,F} & \dots & \vec{d}_{N,F} \end{bmatrix}_{2F \times N}.$$

By row permutations, the matrix can be further transformed to

$$F \sim \left[\begin{array}{cccc} d^x_{0,0} & d^x_{1,0} & \dots & d^x_{N,0} \\ d^x_{0,1} & d^x_{1,1} & \dots & d^x_{N,1} \\ \vdots & \vdots & \vdots & \vdots \\ d^x_{0,F} & d^x_{1,F} & \dots & d^x_{N,F} \\ \hline d^y_{0,0} & d^y_{1,0} & \dots & d^y_{N,0} \\ d^y_{0,1} & d^y_{1,1} & \dots & d^y_{N,1} \\ \vdots & \vdots & \vdots & \vdots \\ d^y_{0,F} & d^y_{1,F} & \dots & d^y_{N,F} \end{array}\right]_{2F \times N} = \left[\frac{U}{V}\right].$$

That is, matrix F is *row equivalent* to $\left[\frac{U}{V}\right]$. *Therefore the flow matrix F has the same rank as matrix $\left[\frac{U}{V}\right]$, with an upper bound of 9.* This means that if the flow history vectors of all pixels across all the frames are correctly computed, they are supposed to reside in a low rank subspace, despite the size of the object and the length of the sequence. In other words, every pixel's flow history is highly correlated with the others.

The importance of this theory is that the less reliable flow vectors thus can be regulated by the reliable ones, by constraining the regularized ones to be in the same subspace. In particular, let matrix $\tilde{F}$ be formed by stacking the flow history vectors of m flow features across F frames. Note that if all the flow vectors in F and $\tilde{F}$ are correct, the two matrices should have the same column space and rank. However in practice, $\tilde{F}$ is more reliable than F, therefore the rank and the column space can be found by the singular value decomposition of $\tilde{F} = \tilde{U}\Sigma\tilde{V}^T$, where Σ is a diagonal matrix of the singular values with $\sigma_{1,1} \geq \sigma_{2,2} \geq \ldots \geq \sigma_{m,m} \geq 0$. The decay of the singular values gives the rank r of $\tilde{F}$, and the first r columns of $\tilde{U}$ span the column space of $\tilde{F}$ (and F as well). With the r columns denoted by $\vec{b}_1, \vec{b}_2, \cdots, \vec{b}_r$, the subspace constraint implies that an arbitrary flow history vector $\vec{\tilde{f}}_i$ should reside in the subspace. Hence there exist r scalar coefficients $c_{i,1}, \cdots, c_{i,r}$, such that

$$\vec{\tilde{f}}_i = \sum_{k=1}^{r} c_{i,k}\vec{b}_k, \qquad i = 1, 2, \cdots, N, \tag{5.12}$$

The obtained flow vector $\vec{\tilde{f}}_i$ can be taken as a corrected reconstruction of $\vec{f}_i$, and finishes the global regularization; or it can be further regularized by spatial smoothness constraint for improved accuracy, as explained below.

5.3.2 Spatial Smoothness Constraint

A key functionality of the subspace constraint is to regularize the flow vectors in smooth regions by flow vectors of the corners or textures. The regularization can be further enhanced by the spatial smoothness of the flow field. Given that nearby pixels belong to the same moving object, it is expected that flow vectors of spatially adjacent pixels have strong correlation. A simple heuristic to formulate the regularization is to bound the spatial variation of the reconstructed flow $\vec{\tilde{f}}_i$. However, note that each $\vec{\tilde{f}}_i$ contains the flow vectors across multiple (at least 5) consecutive frames. This means at least 10 energy functionals and diffusion processes have to be implemented. In this work, *the spatial regularization is formulated in the subspace domain* rather than the flow domain, which is more practical. First, Eq.5.12 indicates that, if the spatial variation of the flow vectors are bounded, then the spatial variation of $c_{i,k}$ is also bounded. Therefore the spatial smoothness constraint is applicable to the the subspace domain. Furthermore, the rank of the column space is much smaller than 9 in most motion models (Irani 2002), which means far less iteration processes and hence lower computational cost.

The spatial regularization can be formulated by minimizing the variation of a continuous function $c_k(x,y)$, of which $c_{i,k}$ is the ith sample,

$$\mathfrak{E}_{smooth} = \int\int_{\Omega} \|\nabla c_k\|^2 \, dxdy. \tag{5.13}$$

To enforce the flow smoothness to conform with the image structure, the energy functional is adapted to

$$\mathfrak{E}_{smooth} = \int\int_{\Omega} \left[p(x,y) \left\| \frac{\partial c_k}{\partial x} \right\|^2 + q(x,y) \left\| \frac{\partial c_k}{\partial y} \right\|^2 \right] dxdy. \tag{5.14}$$

where the anisotropic steering functions p and q are defined by

$$p(x,y) = \frac{1}{\left\| \frac{\partial E}{\partial x} \right\| + \epsilon}, q(x,y) = \frac{1}{\left\| \frac{\partial E}{\partial y} \right\| + \epsilon} \tag{5.15}$$

with the small value $\epsilon = 0.001$ to avoid division by zero.

The global regularization is implemented by minimizing $\mathfrak{E}_{smooth}$. The associated Euler-Lagrange equation reads

$$\frac{\partial \left(p \frac{\partial c_k}{\partial x} \right)}{\partial x} + \frac{\partial \left(q \frac{\partial c_k}{\partial y} \right)}{\partial y} = 0; \tag{5.16}$$

Eq.5.16 is discretized by the energy preserving 5-point differencing scheme, e.g.,

$$\begin{aligned} \frac{\partial \left(p \frac{\partial c_k}{\partial x} \right)}{\partial x} &= p \frac{\partial c_k}{\partial x}\Big|_{x+\frac{1}{2},y} - p \frac{\partial c_k}{\partial x}\Big|_{x-\frac{1}{2},y} \\ \frac{\partial c_k}{\partial x}\Big|_{x+\frac{1}{2},y} &= c_k(x+1,y) - c_k(x,y) \\ \frac{\partial c_k}{\partial x}\Big|_{x-\frac{1}{2},y} &= c_k(x,y) - c_k(x-1,y). \end{aligned} \tag{5.17}$$

The differencing leads to

$$c_k(x,y) = \frac{\varpi_1 c_k(x+1,y) + \varpi_2 c_k(x-1,y) + \varpi_3 c_k(x,y+1) + \varpi_4 c_k(x,y-1)}{\sum_{j=1}^{4} \varpi_j} \tag{5.18}$$

where

$$\varpi_1 = p(x+\frac{1}{2},y), \varpi_2 = p(x-\frac{1}{2},y), \varpi_3 = p(x,y+\frac{1}{2}), \varpi_4 = p(x,y-\frac{1}{2}). \tag{5.19}$$

The optimal solution thus can be found by standard iteration scheme, such as Gauss-Seidel. The converged coefficients $\{\bar{c}_{i,k}\}_{i=0,k=1}^{N,r}$ give the final recovery of the optical flow by

$$\vec{f}_i = \sum_{k=1}^{r} \bar{c}_{i,k} \vec{b}_k, \qquad i = 1,2,\cdots,N. \tag{5.20}$$

5.4 Experimental Results

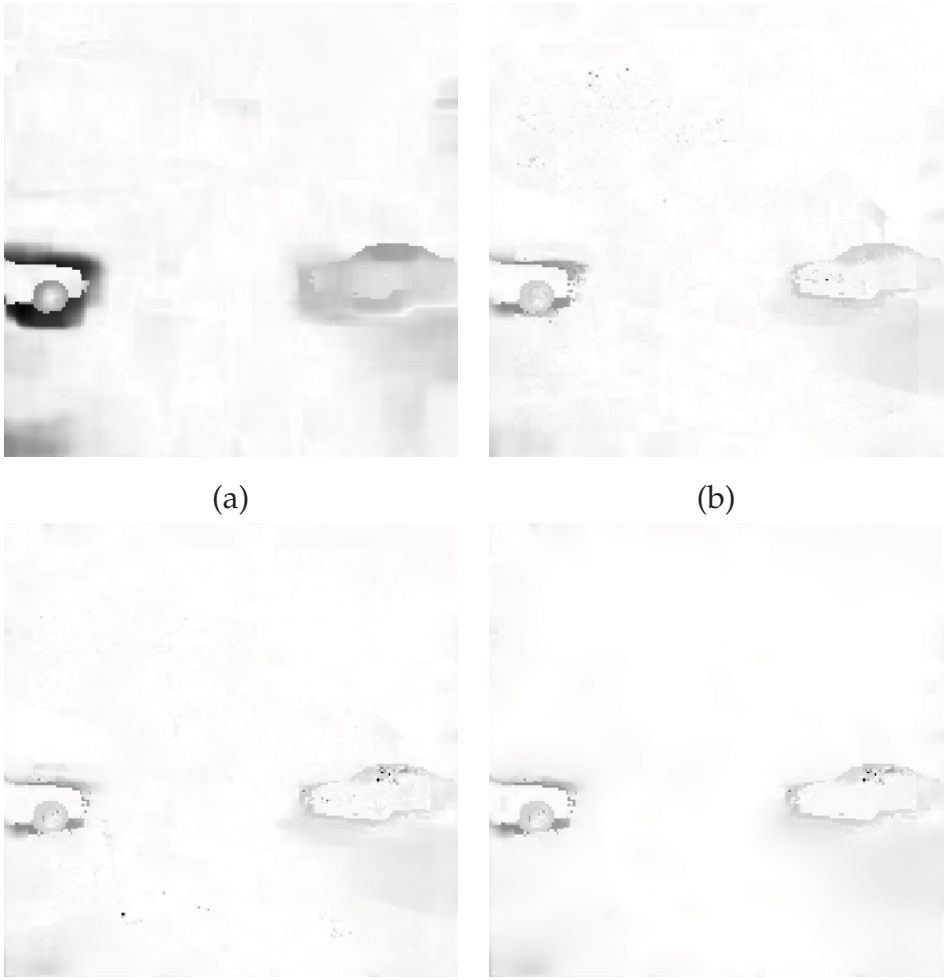
The proposed algorithm is designed for long range sequences. Minimally, the subspace constraint requires local flow across 5 consecutive frames, and local flow requires both 2 preceding and 2 succeeding frames to approximate E_t and E_{tt} by central differencing. Long benchmark sequences created in (McCane *et al.* 2001) are used to test the entire system.

Quantitative performance is measured by average end-point (AEP) error (Baker *et al.* 2007) and average angular error (AAE) (Barron *et al.* 1994). We also report the percentage of pixels that have the error measure above a certain level X, denoted RX. We report $R1.0$ to $R5.0$ for AAE and $R0.1$, $R0.5$, $R1.0$ for AEP. Two pixel wide bands around the image boundaries are discarded, due to the boundary effect in convolving differencing filters with the image.

The first experiment is conducted on sequence **Long Street** (McCane *et al.* 2001), the test frame of which is shown in Fig.5.3. The significant difference around the cars in Fig5.4.a and Fig5.4.b validates the motion boundary preserving capability of the weighted local flow computation. Table 5.1 shows the decrease in error as each step of the proposed method is applied. While each step results in an overall reduction in average error, the application of local (step B) and global methods (C and D) have complementary effects. Step B significantly reduces the number of vectors with an error of 4 degrees or more, but does not improve the overall accuracy of flow vectors with a small error. Conversely, steps C and D significantly improve the accuracy of flow vectors with an error of 3 degrees or less, but do not significantly reduce the number of outliers. Thus the application of local followed by global constraints has the overall effect of first preserving the motion boundary, and then improving the accuracy in each motion segment.

The experiments conducted on sequence **Office** illustrate the effect of applying global constraints to alleviate the aperture problem. This sequence simulates a camera moving toward a static scene. Its 10th frame is shown in Fig5.5.a. On this image, the AAE result obtained in (Bruhn *et al.* 2005) is 3.24°, which is comparable to the proposed method. The large non-textured regions of the images result in the erroneous flow recovery in the local computation step, as shown in Fig5.5.c-Fig5.5.d. However, the application of the global constraints significantly corrected such errors, as shown in Fig5.5.f. Table 5.2 lists the errors as each step is applied to the **Office** sequence.

ground truth flow



(a) (b)

(c) (d)

Figure 5.4. Visual results on sequence **Long Street**. (a) Angular error map of the local computation step without consistency-weighting. Darker intensity indicates larger error. (b) Angular error of the flow field by weighted local computation. (c) Angular error of the dense flow regularized by global subspace constraint. (d) Angular error of the dense flow regularized by spatial smoothness constraint.

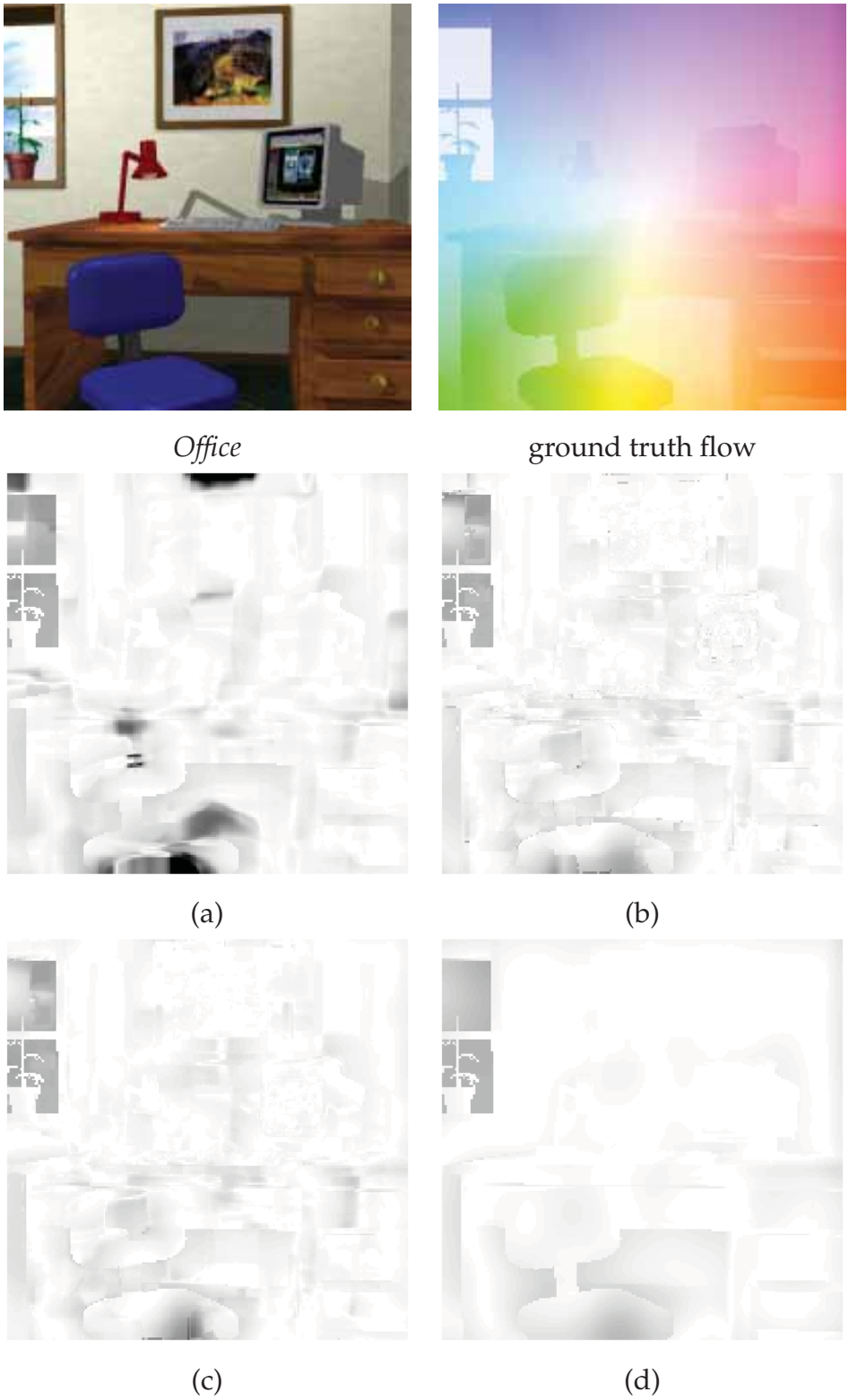


Figure 5.5. Visual results on sequence **Office**. (a)-(d): Average angular maps by steps as described in Fig. 5.4.

	AAE						AEP			
steps	mean	R1	R2	R3	R4	R5	mean	R0.1	R0.5	R1
A.	3.04	49.39	26.36	20.75	17.89	15.27	0.10	19.63	5.57	0.93
B.	2.55	49.39	27.49	19.24	14.40	12.02	0.08	17.68	2.42	0.90
C.	2.12	37.41	20.32	15.11	12.12	10.39	0.07	14.99	2.03	0.72
D.	2.00	31.40	22.14	17.02	13.47	11.39	0.07	16.28	2.73	0.29

Table 5.1. Reduction in error as each step is applied to sequence **Long Street**. Step A is local flow computation without consistency-weighting, B is local flow with consistency-weighting, C is flow with the global subspace constraint, and D is flow with both global subspace and smoothness constraints applied.

	AAE						AEP			
steps	mean	R1	R2	R3	R4	R5	mean	R0.1	R0.5	R1
A.	5.79	92.43	75.43	55.98	40.39	30.76	0.14	35.34	3.84	1.30
B.	4.26	88.63	69.16	49.08	35.40	27.19	0.10	31.14	0.93	0.00
C.	4.10	88.13	66.29	46.62	33.83	25.49	0.09	29.44	0.84	0.00
D.	3.39	75.88	51.42	35.38	24.66	17.85	0.08	24.59	0.83	0.00

Table 5.2. Error statistics as each step is applied to sequence **Office**. Steps are as described in Table 5.1.

Although the experiments show promising results, it is worth noting that, the effectiveness of the subspace constraint relies on the accuracy of the basis vectors detected. The factors that may affect the accuracy include the flow feature selection and the rank decision strategy. For example, if problematic flow vectors obtained in local computation are selected as the flow features in Section 5.3.1, these outliers may lead to distorted estimation of the basis vectors for the subspace. Moreover, even if outliers are not over numbered in this selection, a different rank decision strategy may lead to a different result of the basis vectors. If the rank of the flow subspace is underestimated, important information will be discarded when applying Eq. 5.20; whereas if the rank is overestimated, the contribution of subspace constraints is trivial.

5.5 Summary

This chapter has investigated the complementary effects of local and global optical flow constraints, and proposed improving flow computation accuracy by the combinational use of the local and global constraints. The first section has proposed a local computation system with motion outliers removed or inhibited. With the motion boundary preserved in local computation, the second section has demonstrated that a baseline mean shift clustering algorithm is able to segment multiple moving objects satisfactorily. More advanced motion segmentation techniques can be employed if the scene involves complex motion models, which is out of the scope of this chapter. In the interior of each moving object, the global subspace constraint and spatial smoothness constraint have been used in tandem to further regularize the flow field in the third section. The flow history vectors of all pixels in the same object across multiple frames have proved to reside in a low rank subspace. This subspace constraint enables correcting the less reliable flow vectors by the highly confident ones, and thereby alleviates the aperture problem suffered by the local computation. With the erroneous flow corrected by the subspace constraint, the error propagation problem suffered by the spatial regularization is lessened. The spatial consistency constraint has been formulated as energy minimization in the subspace domain. Experimental results confirm that these constraints complement each other, and address the over-smoothing and aperture problem with better accuracy than previous methods.

Chapter 6

Optical Flow Computation for Fast Rotation

Optical flow computation with fast rotation is a challenging problem in computer vision. This chapter analyzes the aspects of the current flow computation energy functionals that hinder accurate rotational flow recovery, and improves them in three aspects. Firstly, the gradient constancy assumption, which does not apply in the presence of rapid object or camera rotation, is replaced by the constancy of directional derivatives in the isophote and the normal directions. Secondly, the prior assumption of small flow variation, which penalizes the apparent shape change quantities such as curl, divergence and deformation, is inadequate to recover fast rotation. Therefore, it is replaced by the affine motion assumption. Finally, the conventional diffusion in the horizontal-vertical image grid is replaced by directional diffusion in the local isophote-normal grid. Instead of projecting the data terms and the smoothness term to local directions independently, the coordinate system is locally oriented to the intrinsic directions. The global energy functional is posed in this oriented coordinate frame, and a numerical scheme to obtain the global minimizer is derived. With these three modifications to use locally adaptive coordinates, substantial improvement can be achieved over the state-of-the-art techniques.

6.1 Fast Rotation and Large Displacement

The discussion in the preceding chapters focuses on preserving motion discontinuities in the flow computation. To this end, the computation models are based on the brightness constancy assumption (Chapter 3, Chapter 5), gradient invariance assumption (Chapter 4) and minimal flow variation assumption (Chapter 5). Such models have demonstrated promising reliability on sequences that meet the assumptions. In fact, these assumptions can be often found in top performing techniques, for example, (Zimmer *et al.* 2009) and (Wedel *et al.* 2009). [8] However, as the gradient invariance assumption implies translational motion, and the minimal flow variation assumption favors constant flow with minimal change in apparent shape, these models are not suitable for flow recovery with fast rotation. Unfortunately, rotation is one of the most basic motion types and is frequently encountered in practice. Moreover, its consequent large displacement confounds the current optical flow techniques. However, research effort dedicated to rotational flow recovery is still limited.

Solutions to overcome large inter-frame displacement have been proposed in (Brox *et al.* 2009) and (Steinbruecker *et al.* 2009). Both methods employ an additional searching-matching step to compute the displacement. In (Brox *et al.* 2009), the SIFT region matching provides another constraint to steer the flow computation to the right direction; while in (Steinbruecker *et al.* 2009), the data term of the cost function is minimized by a *complete search*. Although the searching-matching enables recovery of large displacement, it is computationally intensive. Moreover, it does not really resolve the intrinsic limitations of the flow estimation models. The most recent attempt for rotational flow recovery is (Ho and Goecke 2008), which suggests using the Fourier-Mellin Transform to deal with rotation and scale change. However, since the spatial information is largely discarded in the Fourier transform, the method does not perform as well as those techniques that exploit flow spatial consistency (e.g. (Ren 2008)) when rotation is not involved.

This chapter elaborates a new approach to generalizing the discontinuity-preserving flow computation to fast rotation, based on an affine motion model in a special coordinate frame that is always aligned with the local isophote and normal directions. Experiments show that, a) when the motion does not involve fast rotation, the proposed flow computation remains the same performance as the traditional ones; b) when the

[8] The performance is according to Middlebury flow evaluation (Baker *et al.* 2009), by the time of writing.

objects undergo fast rotation, substantial improvement can be achieved over the state-of-the-arts techniques.

6.2 Local Adaptive Coordinate System

Previous works tend to formulate flow computation in the horizontal-vertical Cartesian coordinate frame formed by the image grid, except for a few techniques in the log-polar coordinate systems (e.g., (Tistarelli and Sandini 1993) (Daniilidis and Kruger 1995),(Ho and Goecke 2008)). This coordinate system is computationally convenient, but inflexible for flow recovery with rotation. Because fast rotation changes the horizontal and vertical structure between adjacent frames. In other words, a pixel and its horizontal-vertical neighbours have different velocity rather than "moving together". A primary consequence is that the gradient constancy assumption used in previous works becomes invalid, i.e., a pair of true correspondence may have significantly different gradient vectors, as gradient depends on the horizontal-vertical structure. A secondary consequence is that the flow smoothness constraints are inapplicable, since they assume that a pixel has similar velocity to its horizontal-vertical neighbours. As such, it is desirable to construct a coordinate system in which the structure preservation and smoothness constraints can be applied. This section provides a solution based on the local *isophote* and *normal* directions, which are widely used in differential geometry to describe intrinsic properties of a surface. The following subsections discuss how to detect these local directions.

6.2.1 Isophote Direction Detection

In differential geometry, an isophote is defined as the curve of constant intensity. Its tangent and normal directions are intrinsic to the local surface, and form a local reference frame. It is known that derivatives expressed in this reference frame are invariant with rotation and translation. In practice, due to the existence of noise in the image intensity, the isophote/normal direction generally means the direction of least/largest intensity variation. In many applications, the isophote direction is computed as the direction perpendicular to the gradient. This method is simple yet sensitive to noise. A more robust approach is to compute the intrinsic directions as the local structure tensor's eigenvectors. However, this involves structure tensor construction and eigen-decomposition, which is computationally intensive. In this dissertation, the task is

accomplished in a different strategy. Particularly, the 2D plane $[0, 2\pi)$ is evenly quantized to 40 directions. The direction that corresponds to the least directional variation is detected as the isophote direction. The reason for choosing 40 directions is due to the success of the SIFT feature (Lowe 2004), which uses 36 histogram bins to cover the 360° range of orientation. Furthermore, experiments conducted on the image that contains edges of different directions also validate that 40 directions are adequate to present the orientations in the 2D plane (see Fig. 6.3).

The cost of this procedure is dominated by directional derivative computation. A straightforward computation of directional derivative $\frac{\partial E}{\partial \vec{d}}$ in direction $\vec{d}$ (normalized to unit length) is by the "steerable filters" (Freeman and Adelson 1991), which projects the gradient vector $\nabla E = [E_x, E_y]^T$ to $\vec{d}$, i.e.,

$$\frac{\partial E}{\partial \vec{d}} = \left\langle \vec{d}, \nabla E \right\rangle . \tag{6.1}$$

However, such a projection implicitly assumes that the image is everywhere continuously differentiable, which is hard to ensure in practice. This work computes directional derivatives based on its original definition. That is,

$$\frac{\partial E}{\partial \vec{d}} = \lim_{t \to 0^+} \frac{E(X + t\vec{d}) - E(X)}{t}, t \in R^+. \tag{6.2}$$

On the discrete image grid, $\frac{\partial E}{\partial \vec{d}}$ is numerically approximated by the central difference quotient with unit distance, i.e.,

$$\frac{\partial E}{\partial \vec{d}} \approx \frac{E(X + \vec{d}) - E(X - \vec{d})}{2\left\|\vec{d}\right\|} = \frac{E(X + \vec{d}) - E(X - \vec{d})}{2}, \tag{6.3}$$

where $E(X \pm \vec{d})$ are obtained by bilinear interpolation.

Fig.6.1 shows an example with the quantized direction $\vec{d} = [\cos(\theta), \sin(\theta)]^T, \theta \leq \pi/2$. In this example, the bilinear interpolation reads

$$\begin{aligned}
E(X + \vec{d}) &= (1 - \cos(\theta))(1 - \sin(\theta))E(X) + \cos(\theta)(1 - \sin(\theta))E(X + \begin{bmatrix} 1 \\ 0 \end{bmatrix}) \\
&\quad + (1 - \cos(\theta)\sin(\theta)E(X + \begin{bmatrix} 0 \\ 1 \end{bmatrix}) + \cos(\theta)\sin(\theta)E(X + \begin{bmatrix} 1 \\ 1 \end{bmatrix}) \\
E(X - \vec{d}) &= (1 - \cos(\theta))(1 - \sin(\theta))E(X) + \cos(\theta)(1 - \sin(\theta))E(X - \begin{bmatrix} 1 \\ 0 \end{bmatrix}) \\
&\quad + (1 - \cos(\theta))\sin(\theta)E(X - \begin{bmatrix} 0 \\ 1 \end{bmatrix}) + \cos(\theta)\sin(\theta)E(X - \begin{bmatrix} 1 \\ 1 \end{bmatrix}).
\end{aligned}$$

which means that the derivative in direction $\vec{d}$ can be readily obtained by filtering the image with the following mask

$$\frac{1}{2}\begin{bmatrix} -\cos(\theta)\sin(\theta) & -(1-\cos(\theta))\sin(\theta) & 0 \\ -\cos(\theta)(1-\sin(\theta)) & 0 & \cos(\theta)(1-\sin(\theta)) \\ 0 & (1-\cos(\theta))\sin(\theta) & \cos(\theta)\sin(\theta) \end{bmatrix}.$$

The masks with $\theta > \pi/2$ can be obtained similarly. Fig. 6.1 also explains the numerical difference between using steerable derivatives and directional convolution. Steerable filtering is a linear combination of the horizontal and vertical variation; whereas the directional convolution takes the diagonal neighbours into account as well. This extra information leads to better accuracy when the intensity variation is fast.

Filtering the image $E(x,y)$ with this mask can be written as the convolution with a high pass kernel $K_{\frac{\partial}{\partial\vec{d}}}$, i.e.,

$$\frac{\partial E}{\partial \vec{d}} = K_{\frac{\partial}{\partial\vec{d}}} * E, \tag{6.4}$$

which involves one pixel and its nearest neighbours only (see Fig. 6.1), hence we call it pixel-wise directional convolution (PDC). To improve the robustness, a Gaussian convolution with an $N \times N$ Gaussian kernel G_σ, where σ is the standard deviation, can be further applied. Similar to the Derivative of Gaussian (DoG), the directional derivative filtering and Gaussian smoothing can be combined to one filtering process. That is, we need only convolve the image with the one kernel, i.e., the Directional Derivative of Gaussian. This process is expressed by the following derivation

$$\frac{\partial\left(G_\sigma * E\right)}{\partial\vec{d}} = K_{\frac{\partial}{\partial\vec{d}}} * \left(G_\sigma * E\right) = \left(K_{\frac{\partial}{\partial\vec{d}}} * G_\sigma\right) * E = \frac{\partial G_\sigma}{\partial\vec{d}} * E, \tag{6.5}$$

which involves the local area of size $N \times N$ around the pixel, therefore we call it region-wise directional convolution (RDC). As shown in Eq. 6.5, RDC is actually PDC smoothed by a Gaussian convolution. Note that the central differencing is symmetric, therefore only 20 convolutions are needed for the derivatives along 40 directions.

By the directional differencing used for directional derivatives (Eq. 6.3), it can be seen that $E_{-\vec{d}} = -E_{\vec{d}}$. Thus the absolute edge strength in direction $\vec{d}$ and $-\vec{d}$ is the same. Therefore, in the work, the two directions are unified to the same direction that falls in the range $[0,\pi]$.

Fig. 6.2 and Fig. 6.3 demonstrate two examples of estimating the edge direction by the gradient's perpendicular, the eigenvector of the structure tensor, PDC and RDC. For

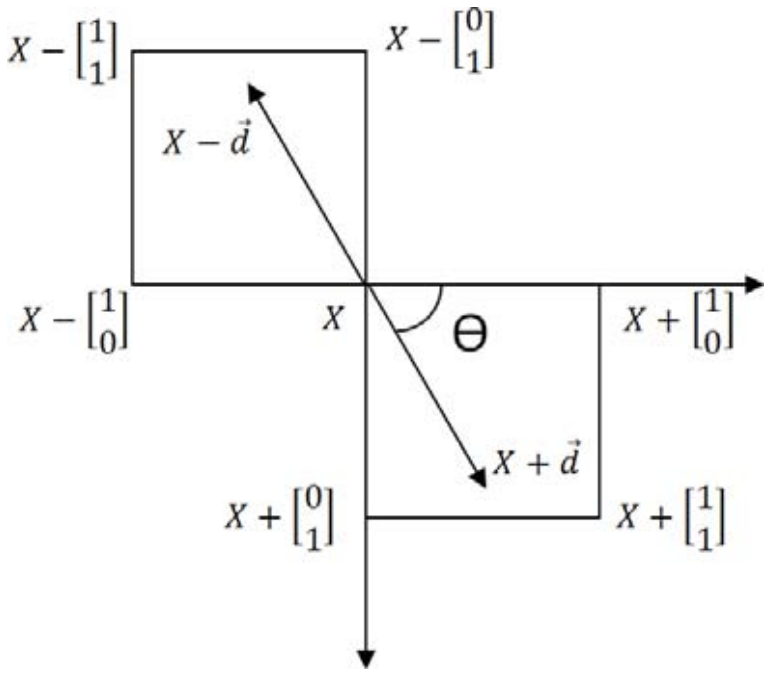


Figure 6.1. An example of pixels used to precompute the directional differencing kernel in direction $\vec{d} = [\cos(\theta), \sin(\theta)]$. The steerable filtering uses only $X, X \pm \begin{bmatrix} 0 \\ 1 \end{bmatrix}, X \pm \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, whereas the proposed method also uses $X \pm \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. The extra information allows more accuracy when the intensity oscillation is fast.

clarity of visualization, in all methods, the edge directions are modulated to the range $[0, \pi]$.

Fig. 6.2 compares these methods on a textured image with noise. The detection is noisy by the gradient method and PDC. The robustness is improved by using the structure tensor and RDC. Moreover, RDC evidentially performs better in the table cloth area than the structure tensor method. Fig. 6.3 presents the performance comparison on a structured image which consists of edges in various directions. In this experiment, PDC and RDC perform comparably to the gradient and structure tensor methods respectively. The comparable performance validates that 40 directions are adequate to quantize the $[0, 2\pi]$ orientation range.

6.2.2 Constructing the Local Reference Frame

The detected isophote direction, its normal direction, and their opposite directions distribute evenly in the 2D plane, with each direction's unit vector lying in one quadrant.

test image *Barbara*

Middlebury Colorwheel

gradient perpendicular

eigendecomposition of structure tensor

pixel-wise directional convolution

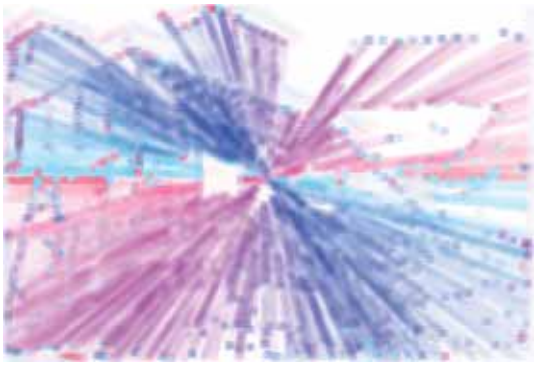
region-wise directional convolution

Figure 6.2. Detect isophote direction by different methods on a noisy textured image. For the visualization clarity, the direction vectors and edge strength are coded by the Middlebury colorwheel. The edge strength is computed as (1) the gradient magnitude in the gradient method; (2) the largest eigenvalue in structure tensor method; (3) the largest directional intensity variation in PDC and RDC. In all tests, the edge strength is mapped to [0,20]. Attention should be drawn to the two pieces of triangle table cloth. The lines on the piece of table cloth that is closer to the camera have positive diagonal direction. The lines on the piece of table cloth that is further away from the camera have negative diagonal direction. The RDC detect the directions in these two regions more accurately.

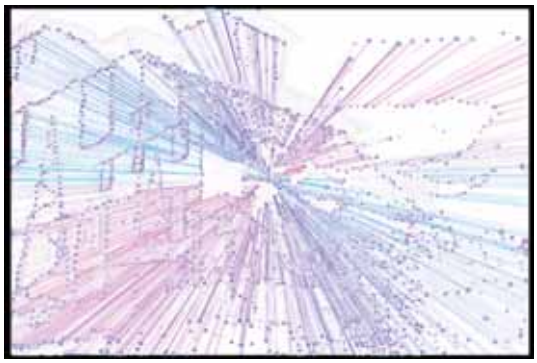
test image

Middlebury Colorwheel

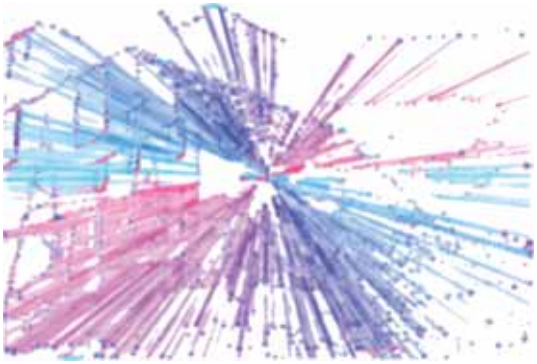
gradient perpendicular



eigendecomposition of structure tensor



pixel-wise directional convolution



region-wise directional convolution

Figure 6.3. Detect isophote direction by different methods on a image that contains edges in various directions. The scaling and coding are as same as in Fig. 6.2. The comparable results confirm that the 40 quantization directions represent various directions in the 2D plane with adequate approximation accuracy.

The local reference frame is constructed in such a way that its axes are always aligned with the direction vectors in the first and second quadrant. In the rest of the dissertation, the local reference system is called the "$\xi\eta$" frame; notations $\vec{d}$ and $\vec{n}$ denote the unit vectors on the isophote and normal axes. The new coordinate system thus is rotated to the local reference system at each pixel. This system is an extension of the traditional one. If $\vec{d}$ and $\vec{n}$ are unit vectors in the horizontal and vertical directions at each pixel, then the "$\xi\eta$" frame degenerates to the image grid. The next section introduces the formulation of affine flow computation in the rotated coordinate system.

6.3 Flow Formulation in the Local Coordinates

Since the coordinate frame is always locally aligned with the intrinsic directions, the flow computation in this coordinate frame is based on local intrinsic structure. In previous works, the energy functionals for flow computation are generally based on the brightness constancy constraint, gradient constancy constraint and smoothness constraint. This section discusses the new formulation of energy functional in the rotated coordinate system, assuming a local affine motion model, i,e,

$$\begin{aligned} u &= \alpha_1 + \alpha_2\xi + \alpha_3\eta \\ v &= \alpha_4 + \alpha_5\xi + \alpha_6\eta, \end{aligned} \tag{6.6}$$

where motion parameters α_i's are locally constant.

6.3.1 The Brightness Constancy Constraint

The brightness constancy constraint in the oriented coordinates is the same as in the conventional coordinates. That is, the flow vector (u, v) is constrained to minimize the brightness mismatching error

$$\begin{aligned} \mathfrak{E}_{\dot{E}} &= |E(x,y,t) - E(x+u, y+v, t+1)| \\ &\approx \left|E_x u + E_y v + E_t\right|; \end{aligned} \tag{6.7}$$

Here we use $\dot{E}$ as a compact notation for the total temporal derivative of E w.r.t. t. To tackle the non-differentiability of the function of absolute value, function $\Psi(\bullet) = \sqrt{\bullet^2 + \epsilon}$ is used instead of $|\bullet|$, where ϵ is a small positive number. Therefore, the first energy term for the flow vector is

$$\mathfrak{E}_{\dot{E}} = \Psi\left(E_x u + E_y v + E_t\right). \tag{6.8}$$

6.3.2 The Edge-Normal Derivative Invariance Constraint

Based on the brightness constancy constraint $\dot{E} = 0$, many existing works assume that $\frac{\partial \dot{E}}{\partial x} = 0$ and $\frac{\partial \dot{E}}{\partial y} = 0$. In the oriented coordinate system, they are replaced by $\frac{\partial \dot{E}}{\partial \vec{d}} = 0$ and $\frac{\partial \dot{E}}{\partial \vec{n}} = 0$. That is, the flow vector (u, v) is constrained to minimize the bias

$$\begin{aligned} \mathfrak{E}_{\dot{E}_d} &= \Psi\left(\frac{\partial \dot{E}}{\partial \vec{d}}\right) \\ &= \Psi\left(E_{x\vec{d}}u + E_x u_{\vec{d}} + E_{y\vec{d}}v + E_y v_{\vec{d}} + E_{\vec{d}t}\right) \\ \mathfrak{E}_{\dot{E}_n} &= \Psi\left(\frac{\partial \dot{E}}{\partial \vec{n}}\right) \\ &= \Psi\left(E_{x\vec{n}}u + E_x u_{\vec{n}} + E_{y\vec{n}}v + E_y v_{\vec{n}} + E_{\vec{n}t}\right), \end{aligned} \tag{6.9}$$

where subscripts $\vec{d}$ and $\vec{n}$ denote the directional derivatives in the two directions.

6.3.3 The Regularity Constraints in Edge and Normal Directions

The three constraints given by Eq.6.8 and Eq.6.9 all originate from the brightness constancy assumption, and therefore are always linearly dependent. As a consequence, there are infinite number of flow vectors that may fulfill the three constraints, and they can be combined with infinite possibilities for the whole dense flow field. A typical solution is to find the one which has the most desirable variation over the whole image. The difficulty is to quantify the "desirable variation", which means simultaneously preserving the flow regularity in smooth motion area and the sharpness at motion boundaries. As introduced in Chapter 2, one approach is to limit the flow smoothness by the intensity smoothness (e.g., (Nagel 1983b), (Brox *et al.* 2009)). An alternative approach is to regularize the flow field by the motion discontinuities successively (e.g., (Weickert and Schnörr 2001)). More recently, researchers have proposed using Joint Image- and Flow- driven regularization (e.g., (Zimmer *et al.* 2009)), where the flow gradient is projected to the intensity normal and edge direction. In this work, the regularization is also controlled by both intensity and motion discontinuities, yet by a different method. The regularity constraints of the flow are imposed in the edge and normal directions, which are the most and least variation of intensity directions. To tune the smoothness anisotropically in different directions, two affine prior terms have been tried in this work. The first term, as shown in Eq.6.10, aims at penalizing the variation of flow gradient based on the affine parameters.

$$\mathfrak{E}_s = \int\int \left(\left(u_{\vec{d}} - \alpha_2\right)^2 + \left(u_{\vec{n}} - \alpha_3\right)^2 + \left(v_{\vec{d}} - \alpha_5\right)^2 + \left(v_{\vec{n}} - \alpha_6\right)^2 \right) d\xi d\eta. \tag{6.10}$$

This is in a similar spirit to (Ju *et al.* 1996), where the parameter α_i's are estimated from the neighbouring patch. This approach suffers the error propagation at motion boundary. Because the motion estimation is generally error-prone around motion boundaries, therefore the subsequent estimation of the α_i's is usually unreliable. The significant errors in the α_i's are propagated to further iterations in the numerical scheme. To avoid this issue, another affine prior term is proposed. Similar to (Nir *et al.* 2008), the proposed regularization term penalizes the gradient of motion parameters. More specifically, let

$$\vec{m} = \begin{bmatrix} u & v & u_{\vec{d}} & v_{\vec{d}} & u_{\vec{n}} & v_{\vec{n}} \end{bmatrix}^T.$$

The regularization term penalizes the variation of $\vec{m}$ by minimizing

$$\mathfrak{E}_s = \int\int_{\Omega} \left(\|\nabla_d \vec{m}\|_2^2 \right) d\xi d\eta. \tag{6.11}$$

To preserve the flow smoothness in the edge direction and impede oversmoothing in the normal direction, the energy functional is adjusted to

$$\mathfrak{E}_s = \int\int_{\Omega} \left(p \cdot \|\vec{m}_{\vec{d}}\|_2^2 + q \cdot \|\vec{m}_{\vec{n}}\|_2^2 \right) d\xi d\eta. \tag{6.12}$$

where p and q are nonnegative monotonically decreasing functions of the flow variation. Particularly in this work, they are defined by

$$\begin{aligned} p &= \left(\|\vec{m}_{\vec{d}}\|_2^2 + \epsilon \right)^{-\frac{1}{2}} \\ q &= \left(\|\vec{m}_{\vec{n}}\|_2^2 + \epsilon \right)^{-\frac{1}{2}}. \end{aligned} \tag{6.13}$$

The advantage of such definition is two-fold. First, function p and q steer the flow smoothness based on the flow itself, and hence avoid over-segmenting at intensity edges. Second, the energy functional is in the same order of an $L1$ regularizer, which is more robust than its $L2$ counterpart; Although the authors of (Trobin *et al.* 2008) pointed out that bias might be induced by the regularization term proposed in (Nir *et al.* 2008), the empirical study shows that using Eq.6.12 is uniformly more robust than Eq.6.10. Therefore this work adopts Eq.6.12 as the smoothness term.

6.3.4 The Balanced Combination of the Constraints

The ultimate energy functional of the flow computation is a balanced combination of the above constraints

$$\mathfrak{E} = \lambda_1 \mathfrak{E}_{\dot{E}} + \lambda_2 \left(\mathfrak{E}_{\dot{E}_d} + \mathfrak{E}_{\dot{E}_n} \right) + \lambda_3 \mathfrak{E}_s, \tag{6.14}$$

where λ_i's are Lagrange multipliers balancing the importance of each error term. By calculus of variation, the minimizer of the energy functional fulfills the associated Euler-Lagrange equation pair in the oriented coordinate system

$$\begin{aligned}\frac{\partial \mathfrak{E}}{\partial u} - \frac{\partial}{\partial \vec{d}}\left(\frac{\partial \mathfrak{E}}{\partial u_{\vec{d}}}\right) - \frac{\partial}{\partial \vec{n}}\left(\frac{\partial \mathfrak{E}}{\partial u_{\vec{n}}}\right) &= 0 \\ \frac{\partial \mathfrak{E}}{\partial v} - \frac{\partial}{\partial \vec{d}}\left(\frac{\partial \mathfrak{E}}{\partial v_{\vec{d}}}\right) - \frac{\partial}{\partial \vec{n}}\left(\frac{\partial \mathfrak{E}}{\partial v_{\vec{n}}}\right) &= 0. \end{aligned} \tag{6.15}$$

By substituting Eq. 6.14 into the Euler-Lagrange pair, and introducing the following notation for the directional divergence

$$\mathrm{div}_d() = \frac{\partial}{\partial \vec{d}}() + \frac{\partial}{\partial \vec{n}}()$$

one has

$$\begin{aligned} 0 &= 2\lambda_1 \Psi'\left(\dot{E}\right) E_x \left(E_x u + E_y v + E_t\right) \\ &+ 2\lambda_2 \left[\Psi'\left(\dot{E}_d\right) E_{x\vec{d}} \left(E_{x\vec{d}} u + E_x u_{\vec{d}} + E_{y\vec{d}} v + E_y v_{\vec{d}} + E_{\vec{d}t}\right)\right] \\ &+ 2\lambda_2 \left[\Psi'\left(\dot{E}_n\right) E_{x\vec{n}} \left(E_{x\vec{n}} u + E_x u_{\vec{n}} + E_{y\vec{n}} v + E_y v_{\vec{n}} + E_{\vec{n}t}\right)\right] \\ &- 2\lambda_2 \mathrm{div}_d \begin{pmatrix} \Psi'\left(\dot{E}_d\right) E_x \left(E_{x\vec{d}} u + E_x u_{\vec{d}} + E_{y\vec{d}} v + E_y v_{\vec{d}} + E_{\vec{d}t}\right) \\ \Psi'\left(\dot{E}_n\right) E_x \left(E_{x\vec{n}} u + E_x u_{\vec{n}} + E_{y\vec{n}} v + E_y v_{\vec{n}} + E_{\vec{n}t}\right) \end{pmatrix} \\ &- \lambda_3 \mathrm{div}_d \begin{pmatrix} \frac{\partial p}{\partial u_{\vec{d}}} \cdot \left\|\vec{m}_{\vec{d}}\right\|_2^2 + 2p \cdot u_{\vec{d}} \\ \frac{\partial q}{\partial u_{\vec{n}}} \cdot \left\|\vec{m}_{\vec{n}}\right\|_2^2 + 2q \cdot u_{\vec{n}} \end{pmatrix} \end{aligned} \tag{6.16}$$

$$\begin{aligned} 0 &= 2\lambda_1 \Psi'\left(\dot{E}\right) E_y \left(E_x u + E_y v + E_t\right) \\ &+ 2\lambda_2 \left[\Psi'\left(\dot{E}_d\right) E_{y\vec{d}} \left(E_{x\vec{d}} u + E_x u_{\vec{d}} + E_{y\vec{d}} v + E_y u_{\vec{d}} + E_{\vec{d}t}\right)\right] \\ &+ 2\lambda_2 \left[\Psi'\left(\dot{E}_n\right) E_{y\vec{n}} \left(E_{x\vec{n}} u + E_x u_{\vec{n}} + E_{y\vec{n}} v + E_y u_{\vec{n}} + E_{\vec{n}t}\right)\right] \\ &- 2\lambda_2 \mathrm{div}_d \begin{pmatrix} \Psi'\left(\dot{E}_d\right) E_y \left(E_{x\vec{d}} u + E_x u_{\vec{d}} + E_{y\vec{d}} v + E_y v_{\vec{d}} + E_{\vec{d}t}\right) \\ \Psi'\left(\dot{E}_n\right) E_y \left(E_{x\vec{n}} u + E_x u_{\vec{n}} + E_{y\vec{n}} v + E_y v_{\vec{n}} + E_{\vec{n}t}\right) \end{pmatrix} \\ &- \lambda_3 \mathrm{div}_d \begin{pmatrix} \frac{\partial p}{\partial v_{\vec{d}}} \cdot \left\|\vec{m}_{\vec{d}}\right\|_2^2 + 2p \cdot v_{\vec{d}} \\ \frac{\partial q}{\partial v_{\vec{n}}} \cdot \left\|\vec{m}_{\vec{n}}\right\|_2^2 + 2q \cdot v_{\vec{n}} \end{pmatrix}. \end{aligned} \tag{6.17}$$

As the equation pair have the same form for u and v, the following discussion focuses on u and can be extended to v straightforwardly. Given the motion model is affine, the second order derivatives of the flow vector vanish. Thus it can be easily verified that

$$\mathrm{div}_d \begin{pmatrix} \frac{\partial p}{\partial u_{\vec{d}}} \cdot \left\|\vec{m}_{\vec{d}}\right\|_2^2 \\ \frac{\partial q}{\partial u_{\vec{n}}} \cdot \left\|\vec{m}_{\vec{n}}\right\|_2^2 \end{pmatrix} = 0.$$

Consequently the Euler-Lagrange equation is simplified to

$$
\begin{aligned}
0 &= \lambda_1 \Psi'\left(\dot{E}\right) E_x \left(E_x u + E_y v + E_t\right) \\
&+ \lambda_2 \left[\Psi'\left(\dot{E}_d\right) E_{x\vec{d}} \left(E_{x\vec{d}} u + E_x u_{\vec{d}} + E_{y\vec{d}} v + E_y v_{\vec{d}} + E_{\vec{d}t}\right)\right] \\
&+ \lambda_2 \left[\Psi'\left(\dot{E}_n\right) E_{x\vec{n}} \left(E_{x\vec{n}} u + E_x u_{\vec{n}} + E_{y\vec{n}} v + E_y v_{\vec{n}} + E_{\vec{n}t}\right)\right] \\
&- \lambda_2 \mathrm{div}_d \begin{pmatrix} \Psi'\left(\dot{E}_d\right) E_x \left(E_{x\vec{d}} u + E_x u_{\vec{d}} + E_{y\vec{d}} v + E_y v_{\vec{d}} + E_{\vec{d}t}\right) \\ \Psi'\left(\dot{E}_n\right) E_x \left(E_{x\vec{n}} u + E_x u_{\vec{n}} + E_{y\vec{n}} v + E_y v_{\vec{n}} + E_{\vec{n}t}\right) \end{pmatrix} \\
&- \lambda_3 \mathrm{div}_d \begin{pmatrix} p \cdot u_{\vec{d}} \\ q \cdot u_{\vec{n}} \end{pmatrix}
\end{aligned} \tag{6.18}
$$

The optimal flow field (u^o, v^o) is obtained by solving Eq. 6.18 and its counterpart for v. In the next section, we derive the numerical scheme to obtain the solution.

6.4 Multi-Scale Multi-Stage Numerical Solution

To deal with the large displacement caused by fast rotation, the numerical implementation of the algorithm follows the multi-scale multi-stage refinement routine. The optimization thus requires nested loops of iterations to achieve the solution. Briefly (as shown in the pseudo-code below), in each level, the flow is computed by several iterations of image warping and refinement; in each refinement stage, the optimal solution of the non-linear equations is approximated by several iterations of solving the

linearized functions; in each step of solving linearized equations, standard Jacobi iterations are employed. Details are explained in the following subsections and the flow chart.

Input: $I(x,y,t)$,$I(x,y,t+1)$

Output: u,v

foreach *Level l (Sec. 6.4.1)* **do**
    initialize the flow of level l;
    warp the second image and compute the new partial derivatives;
    foreach *Stage s (Sec. 6.4.2)* **do**
        foreach *Re-weight step r (Sec. 6.4.3)* **do**
            compute $\Psi'()$, $p()$ and $q()$;
            foreach *Jacobi iteration τ (Sec. 6.4.4)* **do**
                compute h_1, h_2, g_1, g_2;
                compute $\delta u^{l,s,r,\tau+1}$, $\delta v^{l,s,r,\tau+1}$ by Eq. 6.25;
            end
        end
        warp the second image and compute the new partial derivatives;
    end
end

6.4.1 Multi-Scale Pyramidal Implementation

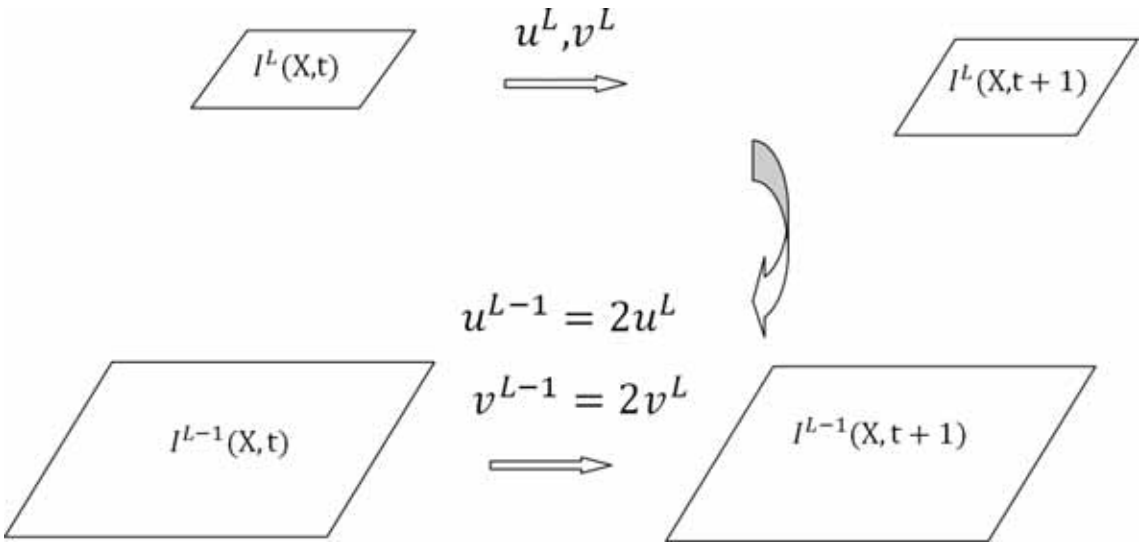


Figure 6.4. An illustration of the pyramidal implementation. The flow computation starts from the coarsest level of the pyramid, and is propagated to the next level as an initialization.

In the implementation, a Gaussian pyramid is constructed by down-sampling the original image pair by a factor of 2. The flow computation starts from the coarsest level L with the initialization $(u^L, v^L) = (\underline{0}, \underline{0})$ and is then refined. The computed flow on level l is propagated to the next level by $(u^{l-1}, v^{l-1}) = (2u^l, 2v^l)$ until the bottom level of the pyramid is reached.

6.4.2 Multi-stage Refinement

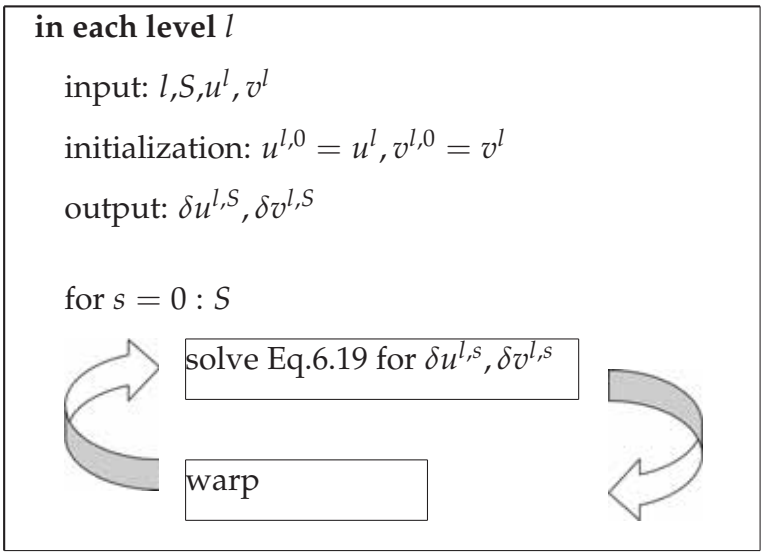


Figure 6.5. The flow chart for multi-stage image warping and flow refinement in the level l.

In each level, the optical flow can be estimated from the down-sampled image pair according to Eq.6.18. The flow accuracy can be further refined by successive image warping. More particularly, the flow field is initialized by $(u^{l,0}, v^{l,0}) = (u^l, v^l)$. In each stage s, the second image is warped according to $(u^{l,s}, v^{l,s})$. The "residual" mismatch between the warped image $I^{l,s}(X, t+1)$ and the first image $I^l(X, t)$ generates the new partial derivatives in Eq.6.18, which yields a flow increment $(\delta u^{l,s}, \delta v^{l,s})$. The flow is then updated by $u^{l,s+1} = u^{l,s} + \delta u^{l,s}$ and $v^{l,s+1} = v^l + \delta v^{l,s}$. The refinement procedure can be repeated for several stages until the flow increment is smaller than a predefined

threshold. In this setting, Eq.6.18 becomes

$$
\begin{aligned}
0 \;=\;& \lambda_1 \Psi'\left(\dot{E}^{l,s}\right) E_x^{l,s}\left(E_x^{l,s}\delta u^{l,s} + E_y^{l,s}\delta v^{l,s} + E_t^{l,s}\right) \\
+\;& \lambda_2 \Psi'\left(\dot{E}_{\vec{d}}^{l,s}\right) E_{x\vec{d}}^{l,s}\left(E_{x\vec{d}}^{l,s}\delta u^{l,s} + E_x^{l,s}\delta u_{\vec{d}}^{l,s} + E_{y\vec{d}}^{l,s}\delta v^{l,s} + E_y^{l,s}\delta v_{\vec{d}}^{l,s} + E_{t\vec{d}}^{l,s}\right) \\
+\;& \lambda_2 \Psi'\left(\dot{E}_{\vec{n}}^{l,s}\right) E_{x\vec{n}}^{l,s}\left(E_{x\vec{n}}^{l,s}\delta u^{l,s} + E_x^{l,s}\delta u_{\vec{n}}^{l,s} + E_{y\vec{n}}^{l,s}\delta v^{l,s} + E_y^{l,s}\delta v_{\vec{n}}^{l,s} + E_{t\vec{n}}^{l,s}\right) \\
-\;& \lambda_2 \mathrm{div}\begin{pmatrix} \Psi'\left(\dot{E}_{\vec{d}}^{l,s}\right) E_x^{l,s}\left(E_{x\vec{d}}^{l,s}\delta u^{l,s} + E_x^{l,s}\delta u_{\vec{d}}^{l,s} + E_{y\vec{d}}^{l,s}\delta v^{l,s} + E_y^{l,s}\delta v_{\vec{d}}^{l,s} + E_{t\vec{d}}^{l,s}\right) \\ \Psi'\left(\dot{E}_{\vec{n}}^{l,s}\right) E_x^{l,s}\left(E_{x\vec{n}}^{l,s}\delta u^{l,s} + E_x^{l,s}\delta u_{\vec{n}}^{l,s} + E_{y\vec{n}}^{l,s}\delta v^{l,s} + E_y^{l,s}\delta v_{\vec{n}}^{l,s} + E_{t\vec{n}}^{l,s}\right) \end{pmatrix} \\
-\;& \lambda_3 \mathrm{div}\begin{pmatrix} p(\vec{m}_{\vec{d}}^{l,s} + \delta\vec{m}_{\vec{d}}^{l,s}) \cdot (u_{\vec{d}}^{l,s} + \delta u_{\vec{d}}^{l,s}) \\ q(\vec{m}_{\vec{n}}^{l,s} + \delta\vec{m}_{\vec{n}}^{l,s}) \cdot (u_{\vec{n}}^{l,s} + \delta u_{\vec{n}}^{l,s}) \end{pmatrix}
\end{aligned}
\tag{6.19}
$$

6.4.3 Removal of the Nonlinearity and Its Relation to Iteratively Reweighted Least Squares

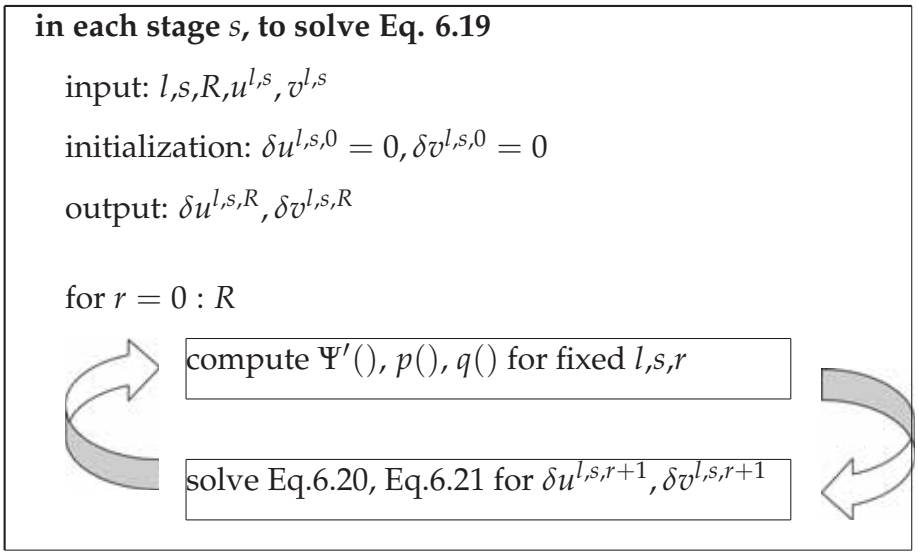


Figure 6.6. The flow chart for approximating the optimization of nonlinear functions by multiple steps of linear optimization.

A significant problem with using Eq.6.19 is its nonlinearity, because Ψ', p and q involve δu and δv. This is a side effect of replacing the L_2 regularization by L_1 regularization. In (Brox *et al.* 2004), the nonlinearity is removed by a fixed-point iteration, which is applied here. The process begins by initializing $(\delta u^{l,s,0}, \delta v^{l,s,0})$ to zeros, in each iteration

step r, the values of Ψ', p and q are computed as functions of $(\delta u^{l,s,r}, \delta v^{l,s,r})$. Subsequently, Eq.6.19 becomes a linear equation of $(\delta u^{l,s,r+1}, \delta v^{l,s,r+1})$,

$$
\begin{aligned}
0 &= \lambda_1 \Psi'\left(\dot{E}^{l,s,r}\right) E_x^{l,s}\left(E_x^{l,s}\delta u^{l,s,r+1} + E_y^{l,s}\delta v^{l,s,r+1} + E_t^{l,s}\right) \\
&+ \lambda_2 \Psi'\left(\dot{E}_{\vec{d}}^{l,s,r}\right) E_{x\vec{d}}^{l,s}\left(E_{x\vec{d}}^{l,s}\delta u^{l,s,r+1} + E_x^{l,s}\delta u_{\vec{d}}^{l,s,r+1} + E_{y\vec{d}}^{l,s}\delta v^{l,s,r+1} + E_y^{l,s}\delta v_{\vec{d}}^{l,s,r+1} + E_{t\vec{d}}^{l,s}\right) \\
&+ \lambda_2 \Psi'\left(\dot{E}_{\vec{n}}^{l,s,r}\right) E_{x\vec{n}}^{l,s}\left(E_{x\vec{n}}^{l,s}\delta u^{l,s,r+1} + E_x^{l,s}\delta u_{\vec{n}}^{l,s,r+1} + E_{y\vec{n}}^{l,s}\delta v^{l,s,r+1} + E_y^{l,s}\delta v_{\vec{n}}^{l,s,r+1} + E_{t\vec{n}}^{l,s}\right) \\
&- \lambda_2 \mathrm{div}\begin{pmatrix} \Psi'\left(\dot{E}_{\vec{d}}^{l,s,r}\right) E_x^{l,s}\left(E_{x\vec{d}}^{l,s}\delta u^{l,s,r+1} + E_x^{l,s}\delta u_{\vec{d}}^{l,s,r+1} + E_{y\vec{d}}^{l,s}\delta v^{l,s,r+1} + E_y^{l,s}\delta v_{\vec{d}}^{l,s,r+1} + E_{t\vec{d}}^{l,s}\right) \\ \Psi'\left(\dot{E}_{\vec{n}}^{l,s,r}\right) E_x^{l,s}\left(E_{x\vec{n}}^{l,s}\delta u^{l,s,r+1} + E_x^{l,s}\delta u_{\vec{n}}^{l,s,r+1} + E_{y\vec{n}}^{l,s}\delta v^{l,s,r+1} + E_y^{l,s}\delta v_{\vec{n}}^{l,s,r+1} + E_{t\vec{n}}^{l,s}\right) \end{pmatrix} \\
&- \lambda_3 \mathrm{div}\begin{pmatrix} p(\vec{m}_{\vec{d}}^{l,s} + \delta\vec{m}_{\vec{d}}^{l,s,r}) \cdot (u_{\vec{d}}^{l,s} + \delta u_{\vec{d}}^{l,s,r+1}) \\ q(\vec{m}_{\vec{n}}^{l,s} + \delta\vec{m}_{\vec{n}}^{l,s,r}) \cdot (u_{\vec{n}}^{l,s} + \delta u_{\vec{n}}^{l,s,r+1}) \end{pmatrix}
\end{aligned}
\tag{6.20}
$$

Similarly, Eq.6.20's counterpart for v is

$$
\begin{aligned}
0 &= \lambda_1 \Psi'\left(\dot{E}^{l,s,r}\right) E_y^{l,s}\left(E_x^{l,s}\delta u^{l,s,r+1} + E_y^{l,s}\delta v^{l,s,r+1} + E_t^{l,s}\right) \\
&+ \lambda_2 \Psi'\left(\dot{E}_{\vec{d}}^{l,s,r}\right) E_{y\vec{d}}^{l,s}\left(E_{x\vec{d}}^{l,s}\delta u^{l,s,r+1} + E_x^{l,s}\delta u_{\vec{d}}^{l,s,r+1} + E_{y\vec{d}}^{l,s}\delta v^{l,s,r+1} + E_y^{l,s}\delta v_{\vec{d}}^{l,s,r+1} + E_{t\vec{d}}^{l,s}\right) \\
&+ \lambda_2 \Psi'\left(\dot{E}_{\vec{n}}^{l,s,r}\right) E_{y\vec{n}}^{l,s}\left(E_{x\vec{n}}^{l,s}\delta u^{l,s,r+1} + E_x^{l,s}\delta u_{\vec{n}}^{l,s,r+1} + E_{y\vec{n}}^{l,s}\delta v^{l,s,r+1} + E_y^{l,s}\delta v_{\vec{n}}^{l,s,r+1} + E_{t\vec{n}}^{l,s}\right) \\
&- \lambda_2 \mathrm{div}\begin{pmatrix} \Psi'\left(\dot{E}_{\vec{d}}^{l,s,r}\right) E_y^{l,s}\left(E_{x\vec{d}}^{l,s}\delta u^{l,s,r+1} + E_x^{l,s}\delta u_{\vec{d}}^{l,s,r+1} + E_{y\vec{d}}^{l,s}\delta v^{l,s,r+1} + E_y^{l,s}\delta v_{\vec{d}}^{l,s,r+1} + E_{t\vec{d}}^{l,s}\right) \\ \Psi'\left(\dot{E}_{\vec{n}}^{l,s,r}\right) E_y^{l,s}\left(E_{x\vec{n}}^{l,s}\delta u^{l,s,r+1} + E_x^{l,s}\delta u_{\vec{n}}^{l,s,r+1} + E_{y\vec{n}}^{l,s}\delta v^{l,s,r+1} + E_y^{l,s}\delta v_{\vec{n}}^{l,s,r+1} + E_{t\vec{n}}^{l,s}\right) \end{pmatrix} \\
&- \lambda_3 \mathrm{div}\begin{pmatrix} p(\vec{m}_{\vec{d}}^{l,s} + \delta\vec{m}_{\vec{d}}^{l,s,r}) \cdot (v_{\vec{d}}^{l,s} + \delta v_{\vec{d}}^{l,s,r+1}) \\ q(\vec{m}_{\vec{n}}^{l,s} + \delta\vec{m}_{\vec{n}}^{l,s,r}) \cdot (v_{\vec{n}}^{l,s} + \delta v_{\vec{n}}^{l,s,r+1}) \end{pmatrix}
\end{aligned}
\tag{6.21}
$$

Recall the definitions of functions Ψ', p and q, which are monotonically decreasing functions of the deviation of the constraints. Therefore, in Eq.6.20 and Eq.6.21, they measure "how well" the corresponding constraint is fulfilled by the flow recovered in step r. If the residual is small, the corresponding constraint is emphasized with larger weights in the iteration step $r+1$. Thus the nonlinearity is removed by iteratively re-weighting each constraint, where the weights are defined by the fulfillment of the constraint in the previous step. This is in a very similar fashion to achieving least absolute deviation regression by iteratively re-weighted least squares (IRLS) (Chapter 4, (Björck 1996)).

6.4.4 Jacobi Iteration

To derive the numerical iteration scheme at each pixel, finite differencing is needed to approximate the partial derivatives in Eq.6.20 and Eq.6.21. Standard central differencing is used to approximate the first order partials in the non-divergence terms. For example

$$\begin{aligned} \delta u_{\vec{d}} &= \frac{\delta u|_{X+\vec{d}} - \delta u|_{X-\vec{d}}}{2} \\ \delta u_{\vec{n}} &= \frac{\delta u|_{X+\vec{n}} - \delta u|_{X-\vec{n}}}{2}. \end{aligned} \tag{6.22}$$

$\delta v_{\vec{d}}$, $u_{\vec{d}}$ and $v_{\vec{d}}$ are approximately similarly.

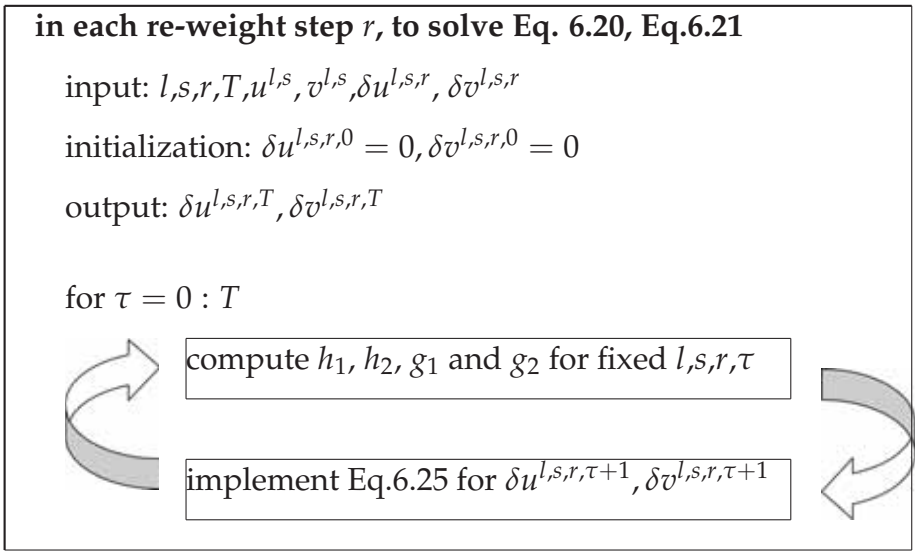


Figure 6.7. The flow chart for each step of linear optimization by Jacobi iteration.

Partial derivatives involved in the divergence terms are approximated by *energy preserving* differencing scheme, which is necessary for the numerical iteration to converge. For example

$$\begin{aligned} \frac{\partial}{\partial \vec{d}} \left(C(\bullet) \cdot \delta u_{\vec{d}} \right) &= C(\bullet)\, \delta u_{\vec{d}}\big|_{X+\frac{\vec{d}}{2}} - C(\bullet)\, \delta u_{\vec{d}}\big|_{X-\frac{\vec{d}}{2}}, \\ \delta u_{\vec{d}}\big|_{X+\frac{\vec{d}}{2}} &= \delta u|_{X+\vec{d}} - \delta u|_{X}, \\ \delta u_{\vec{d}}\big|_{X-\frac{\vec{d}}{2}} &= \delta u|_{X} - \delta u|_{X-\vec{d}}, \end{aligned} \tag{6.23}$$

where $C(\bullet)$ is a general function representing any coefficient function that may appear before $\delta u_{\vec{d}}$ in the divergence terms. Bilinear interpolation is employed to estimate

$C(\bullet)|_{X\pm\frac{\vec{d}}{2}}$. Approximations to $\frac{\partial}{\partial\vec{n}}\left(C(\bullet)\,\delta u_{\vec{n}}\right)$, $\frac{\partial}{\partial\vec{d}}\left(C(\bullet)\,\delta v_{\vec{d}}\right)$ and $\frac{\partial}{\partial\vec{n}}\left(C(\bullet)\,\delta v_{\vec{n}}\right)$ are obtained in the same way. Let $\mathcal{N}(X)=\left\{X\pm\vec{d},X\pm\vec{n}\right\}$ denote the set of the four neighbours of pixel X. Eq.6.20 and Eq.6.21 then can be re-written using these neighbours to the form of

$$\begin{aligned}\delta u^{l,s,r+1}(X) &= h_1^{l,s,r}+g_1\left(\delta v^{l,s,r+1}(X),\delta u_j^{l,s,r+1},\delta v_j^{l,s,r+1}\right)\\ \delta v^{l,s,r+1}(X) &= h_2^{l,s,r}+g_2\left(\delta u^{l,s,r+1}(X),\delta u_j^{l,s,r+1},\delta v_j^{l,s,r+1}\right),\end{aligned}\tag{6.24}$$

where $h_1^{l,s,r}$ and $h_2^{l,s,r}$ are the sum of constant terms with fixed l,s,r; and $g_1()$,$g_2()$ are linear functions; and $j\in\mathcal{N}(X)$. The Jacobi iteration, which starts from initializing $(\delta u^{l,s,r+1,0},\delta v^{l,s,r+1,0})$ by zeros, is updated at iteration step $\tau+1$ by

$$\begin{aligned}\delta u^{l,s,r+1,\tau+1}(X) &= h_1+g_1\left(\delta v^{l,s,r+1,\tau}(X),\delta u_j^{l,s,r+1,\tau},\delta v_j^{l,s,r+1,\tau}\right)\\ \delta v^{l,s,r+1,\tau+1}(X) &= h_2+g_2\left(\delta u^{l,s,r+1,\tau}(X),\delta u_j^{l,s,r+1,\tau},\delta v_j^{l,s,r+1,\tau}\right),\end{aligned}\tag{6.25}$$

6.5 Experimental Results

6.5.1 Comparison with Existing Methods

Table 6.1 presents the algorithm's numerical performance on 5 Middlebury training sequences. Similar to (Brox *et al.* 2004), the parameters are set to $\lambda_1=1$, $\lambda_1=2$, $\lambda_3=600$ for synthesized images; and following (Brox *et al.* 2009), $\lambda_1=1$, $\lambda_1=5$, $\lambda_3=100$ are used for real images; $\epsilon=1.0^{-3}$ in all the experiments.

Table 6.1 also compares the performance of the proposed method to a contemporary technique (Ho and Goecke 2008), which deals with rotation using the Fourier-Mellin Transform (FMT). It can be seen that except on **Venus**, the proposed method significantly outperforms FMT on all the other sequences. Moreover, to illustrate the performance of the proposed method relative to other methods, Table 6.1 also lists the numerical results on the same sequences reported in a recent work (Ren 2008) and the reference therein. The visual results of these experiments are presented in Fig. 6.8.

6.5.2 Numerical Evaluation of Each Step

This section experimentally evaluates the three features of the proposed technique. These are the oriented coordinate system, the affine model and the proposed directional derivatives computation method. Experiments are conducted on benchmark

NOTE:
This table is included on page 90
of the print copy of the thesis held in
the University of Adelaide Library.

Table 6.1. Comparison of the proposed directional flow recovery technique with traditional and contemporary methods. Bold numbers indicate the smallest errors on the test sequences. The results by Black-Anandan's method (Black and Anandan 1996) are taken from (Ren 2008)

.

sequences that undergo a variety of motion, such as rigid/non-rigid fast/slow translation or rotation. In all the experiments, two frames are extracted from the sequence and converted to gray-scale to compute the dense flow field.

The three features are tested separately, by evaluating the difference with a given feature is "on" and "off" with all other components of the experiment remain unchanged. In the numerical evaluation, both average angular error (AAE) and average endpoint error (AEP) metrics are applied. The two metrics are found to be highly consistent with each other in all experiments, hence only AAE values are reported in Table 6.2. The AEP values support the same analysis and conclusion.

The Oriented and Horizontal-Vertical Coordinate System

Table 6.2 compares the flow computation in the oriented and traditional coordinate system. On sequences that are dominated by rigid translation, such as the **Hydrangea** and **Venus** sequences, computation in both systems obtain similar results. The benefit of the oriented coordinate system is clearly demonstrated by tests on **Dimetrodon**, which contains non-rigid motion; and **Grove2**, which is viewed by a moving and tilting down camera. On sequences **RubberWhale** and **Urban2**, the error using the oriented coordinate system is slightly greater than the traditional coordinate system. This is most likely because the motion boundaries in both sequences are mainly in the horizontal, vertical or diagonal directions, which coincide with the axes of the traditional coordinate frame. In contrast, **Hydrangea**, **Dimetrodon** and **Grove2** contain motion

boundaries of various directions, and hence the oriented coordinate frame adapts to the direction change more flexibly.

The Steerable Derivatives and Directional Convolution

Directional derivatives can be obtained by either steerable filtering or using directional convolution as proposed in Section 6.2.1. In all experiments (Table 6.2), directional convolution either improves the computed flow field or achieves similar results to using the steerable filtering. On sequences **Dimetrodon** and **Grove2** this benefit is more obvious. This confirms that directional convolution is more flexible when objects undergo nonrigid motion or shape change, whilst still performing well under translation.

Affine and Constant Motion Model

On all the test sequences except **Urban2**, affine model demonstrates better performance than the constant motion model (see Table 6.2); however the improvement is not as substantial (generally around 0.1 in AAE metric) as in the case of changing coordinates. The only exception is **Grove2**, where the difference is 0.27°. This not surprising because the tilting camera in **Grove2** causes clear change in the apparent shape, which is better described by an affine motion model.

	Dimetrodon	Grove2	Hydrangea	Venus	Rubber Whale	Urban2
proposed	2.64	3.54	2.68	7.09	5.03	4.05
coordinate	3.22	3.94	2.74	7.22	4.81	3.72
derivative	3.08	3.89	2.69	7.24	4.96	3.97
motion model	2.76	3.81	2.77	7.03	4.93	3.91

Table 6.2. Numerical comparison of each feature's contribution based on the AAE metric. 1st row: the AAE performance of the proposed method; 2nd row: the AAE performance if the proposed coordinate system is replaced by the traditional coordinate system; 3rd row: the AAE performance if the directional convolution is replace by steerable filtering; 4th row: the AAE performance if the 4 deformation parameters are replaced by 0s.

6.5.3 Real Sequences

Both (Brox *et al.* 2009) and (Steinbruecker *et al.* 2009) report experimental results on the **HumanEva-II** sequence as shown in Fig 6.10. To simulate large displacement, both works extract frame 546 and 550 for experimental testing. For direct comparison, we have also used the same frames for evaluation. The most challenging region in the scene occurs where the running person's right foot has a large 2D displacement (about 20 pixels) due to translation and rotation. To examine the role that each feature plays on the real sequence, the interpolation error is computed. The second frame is warped according to the flow computed. The difference between the first frame and the warped second frame is coded by 8-bit grayscale, as shown in Fig.6.9.

From the interpolation errors before and after each feature is applied, noticeable improvement by the affine motion model can be observed in the foot area. However, computation for the foot's motion in the oriented coordinate frame is less accurate than in the traditional frame. A closer inspection shows that the rotated coordinate system slightly improves the computation performance in the area of the right hand and around some segments of the motion boundaries, but not substantial.

Fig.6.10 compares the colorcoded flow field obtained by different methods. Compared to the patch matching guided technique of (Brox *et al.* 2009), the proposed method recovers the motion of the right foot with more success; compared to the complete search technique of (Steinbruecker *et al.* 2009), the motion estimation of the right foot by the proposed method is more blurry. However, the proposed method does more faithfully recover the motion between the two legs.

6.6 Summary

This chapter has presented a method to generalize the flow computation from translational motion to fast rotation. This is done by rotating the coordinate system to the local isophote-normal directions, which are detected by directional convolution with pre-computed high pass kernels. Optical flow computation in the locally oriented coordinate system is formulated based on the rotational invariance of the directional derivatives, directional smoothness constraints, and the affine motion model. The numerical scheme to find the optimal flow field is derived. Experimental results have confirmed the contribution of the intrinsic direction detection approach, the use of the oriented coordinate system and the affine motion prior.

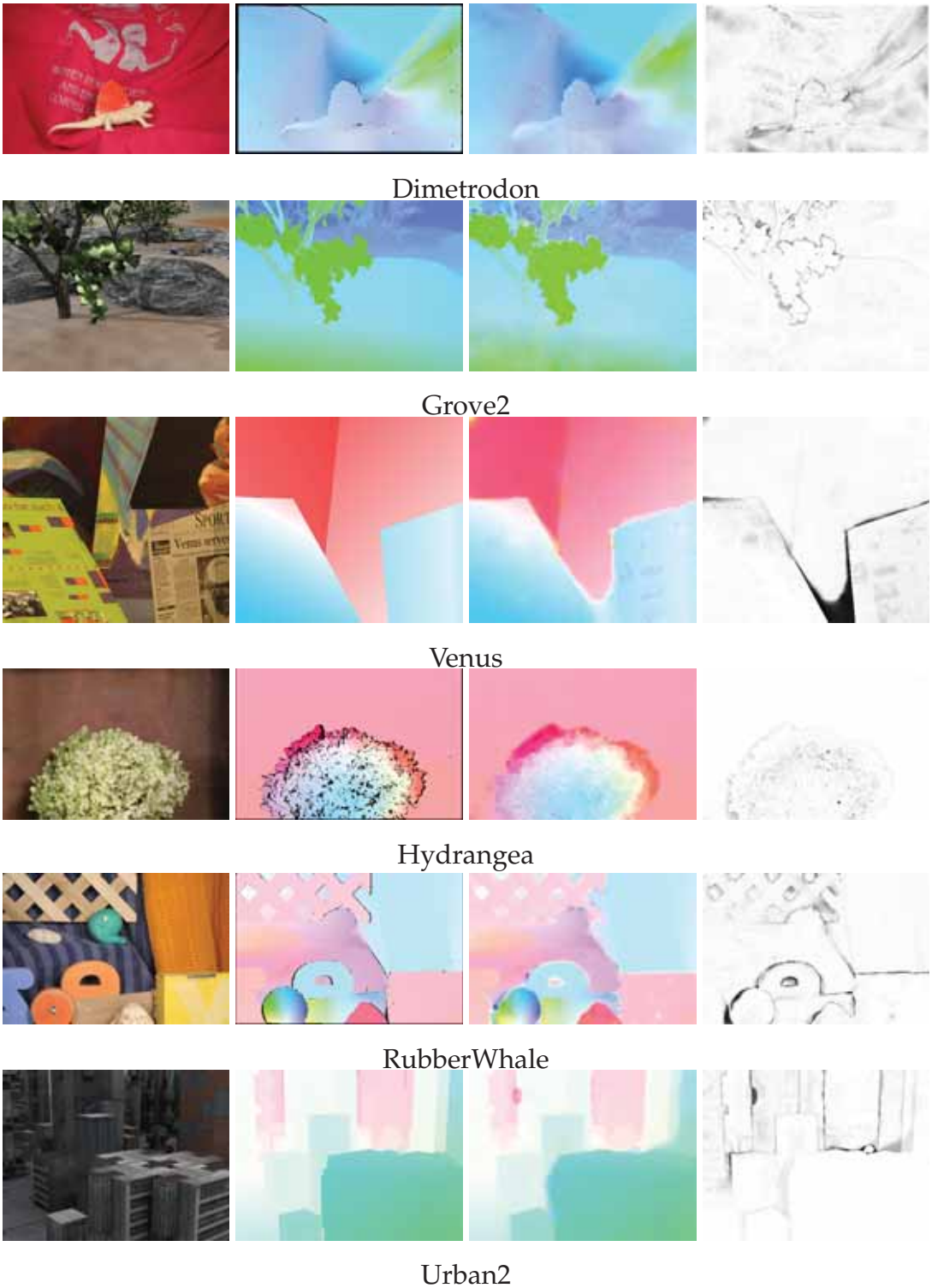


Figure 6.8. Visual results of the flow estimation by the proposed method on Middlebury sequences. From left to right, the test image, the ground truth flow, the computed flow, and the AAE images. In the grayscale error maps, darker intensity indicates larger errors.

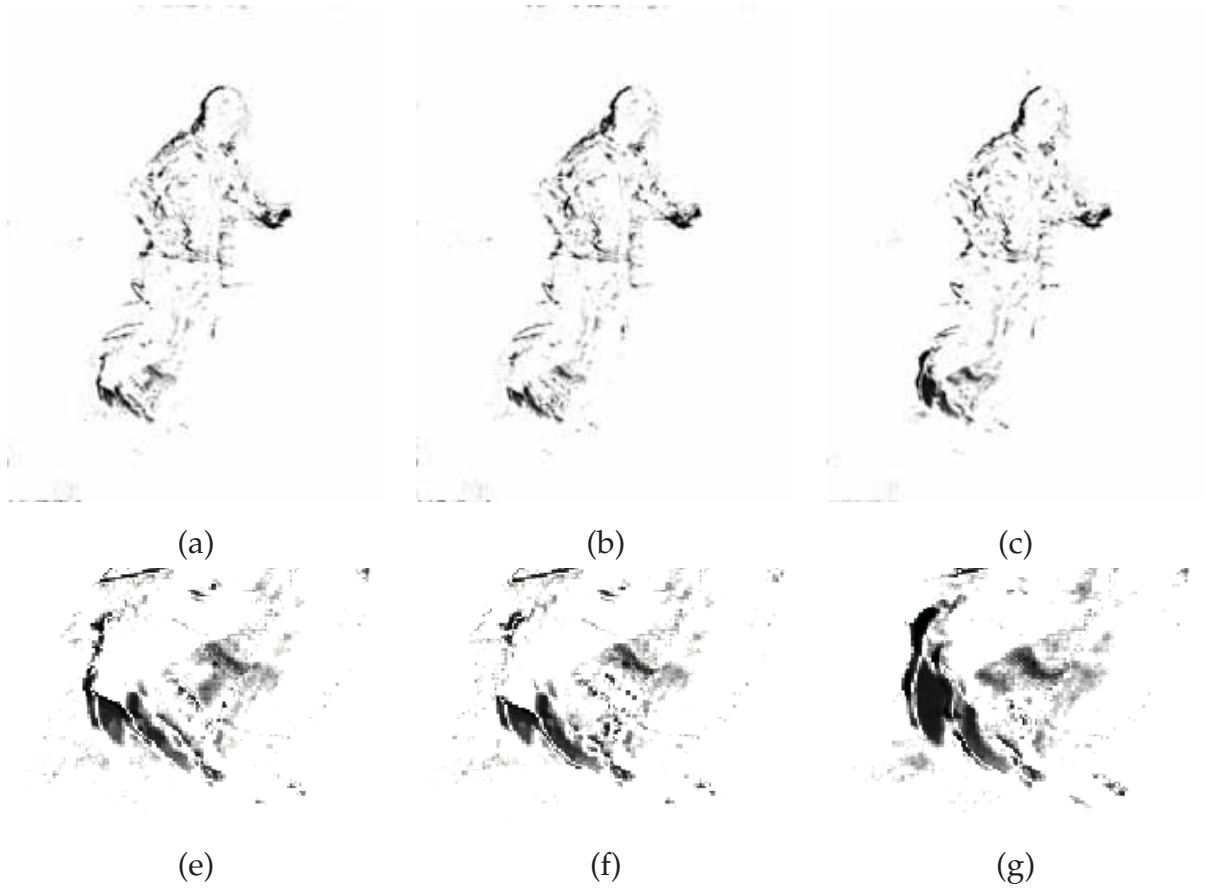


Figure 6.9. The interpolation errors between the first frame and the warped second frame according to the flow computed by a) the proposed method; b) replacing the oriented coordinate system by the traditional one; c) replacing the deformation parameters in the motion model by zeros; (e)-(f) zoomed-in view of the feet area of (a)-(c). Darker intensity indicates larger error at the position. Errors are uniformly scaled for clarity. The comparison between (e) and (f) shows that the traditional coordinate frame works better near the boundary between the right foot and the background; the comparison between (e) and (g) demonstrates that the affine morel significantly improves the recovery accuracy for the right foot in this example.

Frame 546

Frame 550

overlaid

NOTE:
These images are included on page 95 of the print copy of the thesis held in the University of Adelaide Library.

colorwheel (Brox *et al.* 2009) (Brox *et al.* 2004)

warping (Brox *et al.* 2004)

patch matching (Brox *et al.* 2009)

NOTE:
These images are included on page 95 of the print copy of the thesis held in the University of Adelaide Library.

colorwheel (Steinbruecker *et al.* 2009)

complete matching (Steinbruecker *et al.* 2009)

Middlebury colorwheel

proposed method

Figure 6.10. Performance comparison on **HumanEva-II**. Results by previous works are taken from the corresponding publications. Different work uses different colourcoding schemes to visualize the flow field. This works adopts the standard Middlebury colorwheel. The patch matching method does not recover the flow of the right foot, where the complete matching method shows success. The proposed method recovers the flow in this area, but with blurry effects. The complete matching method estimates the motion between the two legs inaccurately; whereas the proposed method performs better in this area.

Chapter 7

Optical Flow Computation by Expectation-Maximization

The first part of this chapter presents an alternative framework to combine local and global optical flow computation, in a similar fashion to Expectation-Maximization. The expectation step predicts the flow vector subject to the global smoothness assumption, while the maximization step estimates the optimal flow refinement by solving a local system. The second part of the chapter generalizes this combinational computation to fast rotation. To this end, both steps are adapted to the intrinsic directions. Different from the quantization vectors used in the previous chapter to represent the isophote direction, a new set of integer vectors are proposed. The convenience and demands of applying these quantization vectors are discussed. Experimental results show that the proposed combination framework preserves sharp motion boundaries better than using either the local system or global computation of the previous chapters. Moreover, it also shows better performance than some contemporary flow techniques that are designed to be robust to rotation.

In the previous chapter, a global flow computation scheme is generalized to rotation by locally rotating the coordinate frame to the intrinsic directions. In this chapter, a local computation system is introduced to improve the robustness of global computation for fast rotation. Different from the combinational computation proposed in Chapter 5, the method presented in this chapter does not involve segmentation, because the flow caused by rotation is difficult to segment reliably. To allow more flexibility, the optimization is realized by two sub- optimization steps. One step minimizes the deviation from the global smoothness, while the other one minimizes the deviation from local structure preservation, which is in a similar fashion of expectation-maximization (Dempster *et al.* 1977). Due to the separation of the two steps, the global and local constraints are adapted to rotation separately, rather than formulating all the components in a unified framework as in the previous chapter.

7.1 Combine Local and Global Computation by EM

Variational flow computation is typically formulated as an energy minimization problem, which minimizes the overall error composed of the data term $D(x,y)$ and the smoothness term $S(x,y)$. The global flow computation of the previous chapter falls in this category. As global computation is more sensitive to noise than local methods (Barron *et al.* 1994), it is suggested in (Bruhn *et al.* 2005) to replace the data term $D(x,y)$ of the global energy functional by a Lucas-Kanade local computation system. For instance, in Horn-Schunck global formulation, the original data term

$$D(x,y) = \int\int \left(E_x u + E_y v + E_t\right)^2 dxdy$$

is replaced by

$$D(x,y) = \int\int \left\| \begin{bmatrix} G_\sigma * E_x^2 & G_\sigma * E_x E_y \\ G_\sigma * E_x E_y & G_\sigma * E_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} G_\sigma * E_x E_t \\ G_\sigma * E_y E_t \end{bmatrix} \right\|_2^2 dxdy,$$

where G_σ is a Gaussian convolution kernel of standard deviation σ. This combination has been demonstrated to be more powerful than using either the local or global component individually. Furthermore, the improved robustness to noise has been validated by experimental results reported in (Bruhn *et al.* 2005).

This chapter applies this idea of using a local system for improved robustness. Thus it seems to be an easy solution to formulate the computation in the same way as in

(Bruhn *et al.* 2005), but on the locally oriented coordinate system. However, there are two potential drawbacks to this strategy. First, the combination is still posed as a global computation problem, which, although it has improved robustness to noise, still suffers error propagation. Second, both the local and global components in the combination can estimate the flow independently, which can be utilized to guide each other; however, in the standard combination scheme, the interaction between the local and global computation is limited. In (Xiao *et al.* 2006), it is shown that the minimization process of a global energy function can be separated into two steps, with each step updating the flow subject to the data and smoothness terms respectively. Furthermore, as the latter step is equivalent to a bilateral filtering operation, more flexibility of customization (e.g., weight by occlusion labelling) is allowed. Inspired by this idea, this work structures the optimization by two sub-steps rather than integrating them into one global framework.

More particularly, with the flow field initialized by $u^0 = \underline{0}$ and $v^0 = \underline{0}$, in each iteration step (indexed by τ), the algorithm computes the intermediate flow $\begin{bmatrix} u^{\tau+\frac{1}{2}} & v^{\tau+\frac{1}{2}} \end{bmatrix}$ by minimizing the smoothness term S^τ, i.e.,

$$\begin{bmatrix} u^{\tau+\frac{1}{2}} \\ v^{\tau+\frac{1}{2}} \end{bmatrix} = \underset{u,v}{\operatorname{argmin}} S^\tau. \tag{7.1}$$

The local image patch is then warped based on the intermediate flow, and the data term $D^{\tau+\frac{1}{2}}$ is computed accordingly. The flow $\begin{bmatrix} u^{\tau+1} & v^{\tau+1} \end{bmatrix}$ is obtained by minimizing the data term, i.e.,

$$\begin{bmatrix} u^{\tau+1} \\ v^{\tau+1} \end{bmatrix} = \underset{u,v}{\operatorname{argmin}} D^{\tau+\frac{1}{2}}. \tag{7.2}$$

The two steps are implemented alternatively for several steps until convergence or reaching the predefined number of iterations.

In this algorithm, the fist step (Eq. 7.1) explores the underlying spatial smoothness of the motion field. Its numerical scheme of optimization is equivalent to a low pass convolution (Xiao *et al.* 2006). In other words, this step evaluates the expectation of the flow from the surrounding flow vectors. The second step (Eq. 7.2) optimizes the flow refinement given the warped image patch. Therefore, this two step optimization process is in the same fashion of the Expectation-Maximization algorithm. This framework of combining the local and global computation is general, in the sense that $S(x, y)$ and $D(x, y)$ can be designed freely according to the application needs. In the

next section, we extend this formulation to flow recovery for rotation, and show how to construct $D(x,y)$ and $S(x,y)$ for this purpose.

7.2 Generalization to Fast Rotation

To extend the computation to rotation, the bilateral filtering in the E-step should be attached to the isophote and normal directions; while the data term in the M-step should be based on the preservation of a rotational invariant feature. Thus the isophote direction detection and directional derivative computation discussed in the previous chapter can be directly applied for these purposes. However, this chapter proposes another set of 40 directions to quantize the 2D plane. The new set of directions are special as they are represented by integer vectors. As will be shown soon, this allows computational convenience in computing directional derivatives and structuring the EM steps, but also requires sufficiently high image resolution. We start from introducing these quantization directions.

7.2.1 The Quantization of the 2D plane by Integer Vectors

Within a local 11×11 patch there exist 40 vectors, which originate from the patch center and end at 40 integer positions. More importantly, they cover the 2D plane quite evenly. In particular, the 10 vectors that cover the first quadrant are

$$\vec{e}_0 = [0,1]; \quad \vec{e}_1 = [1,5]; \quad \vec{e}_2 = [1,3]; \quad \vec{e}_3 = [1,2]; \quad \vec{e}_4 = [2,3];$$
$$\vec{e}_5 = [1,1]; \quad \vec{e}_6 = [3,2]; \quad \vec{e}_7 = [2,1]; \quad \vec{e}_8 = [3,1]; \quad \vec{e}_9 = [5,1].$$

Their orthogonal vectors cover the second quadrant, as shown in Fig.7.2.1.

Among these vectors, the largest angle between two neighbouring vectors is 11.31° (e.g., between $\vec{e}_0$ and $\vec{e}_1$) and the smallest is 7.13° (e.g., between $\vec{e}_3$ and $\vec{e}_4$). Thus the quantization distortion is smaller than $5.7^\circ = 11.31^\circ/2$. This is comparable to using the 40 evenly distributed quantization directions in Section 6.2.1, where the quantization distortion upper limit is 4.5°.

Together with the opposite vectors, the quantization vectors in Fig. 7.2.1 can be grouped into 5 sets, as shown in Fig. 7.2.1. Each set contains 8 vectors that constitute 4 cardinal and 4 ordinal directions. Therefore, no matter how the local patch is rotated, the 8 intrinsic directions fall into one of the five sets.

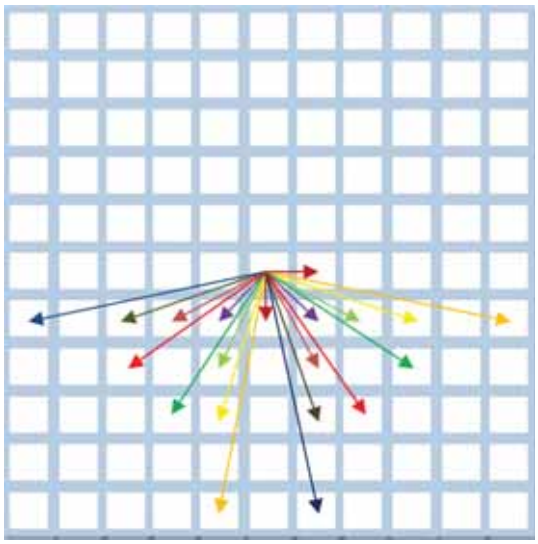


Figure 7.1. An illustration of the 20 vectors that fall on the integer positions of the image grid, covering the first and second quadrant. The pair of vectors that are of the same colour are orthogonal.

In each of these quantization direction $\vec{d}$, the directional derivative computation is then adapted to

$$\frac{\partial E}{\partial \vec{d}} \approx \frac{E(X+\vec{d}) - E(X)}{\left\|\vec{d}\right\|}. \tag{7.3}$$

With $\vec{d}$ represented by integers, pixel $X+\vec{d}$ falls on the image grid. Thus Eq.7.3 utilizes intensity values that are immediately available from the image. For example, the first order derivative in the direction of $\vec{e}_2$ can be obtained by simply filtering the image with the high pass filter

$$K_{\frac{\partial}{\partial \vec{e}_2}} = 1/\sqrt{3} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Compared to the differencing scheme in Section 6.2.1, which needs bilinear interpolation to approximate the intensity values at subpixels, Eq.7.3 avoids blurring sharp edges resulted from smooth interpolation. However, as these directions are not at the same scale, sample aliasing may lead to biased approximation. Therefore, this scheme suits the applications that have high image resolution.

If this requirement is satisfied, the local isophote direction is computed by the following steps. First, the partial derivatives in the 40 directions are computed by convolving the image with the directional high pass filters, and grouped into 5 sets. Next, the local

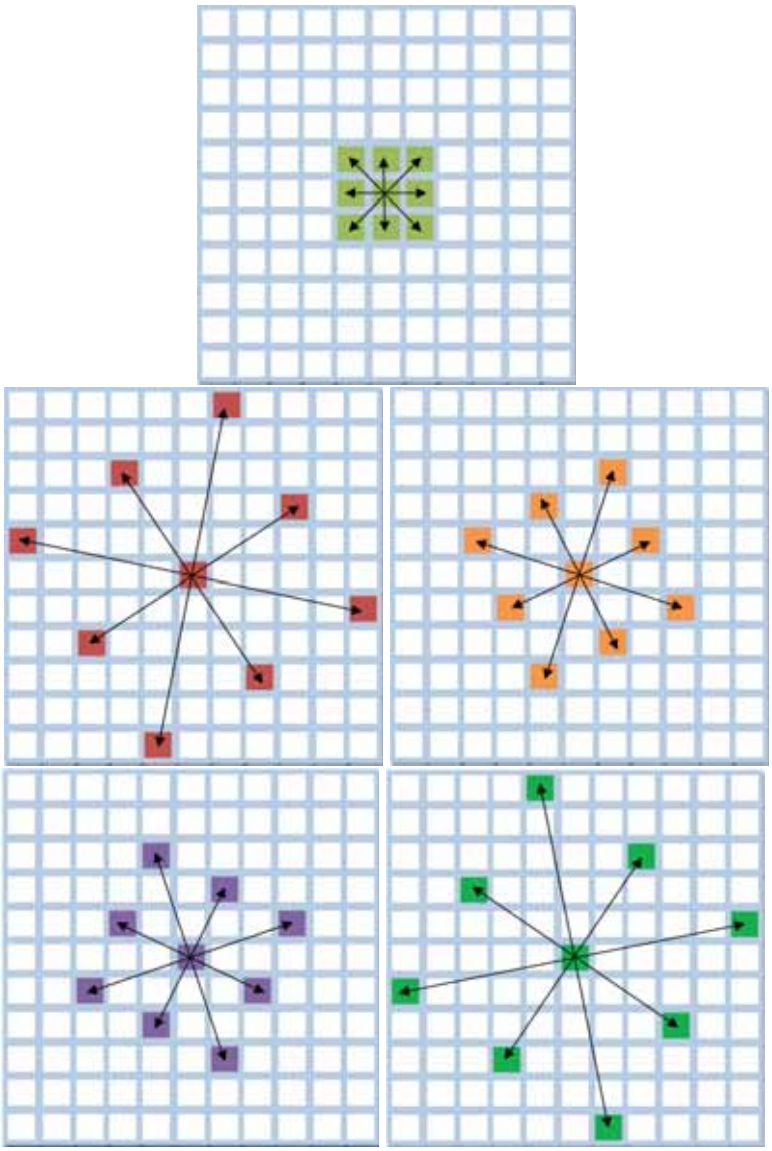


Figure 7.2. The quantization vectors are grouped into 5 sets. Each set contains the cardinal and ordinal direction of a rotated compass rose.

dominant direction is detected by the eigen-decomposition of structure tensors (Section 6.2.1), and modulated to the range of $[0, \pi]$. Finally, the pixel is assigned one of the basis sets in Fig. 7.2.1 that the dominant direction belongs to.

7.2.2 E-Step for Rotation

The expectation step of the algorithm predicts the flow of a pixel from the nearby pixels, subject to the assumed smoothness correlation. Note that, as explained in the previous section and shown in Fig 7.2.1, whichever the isophote direction is, its 8 neighbours in the 4 cardinal and ordinal directions also fall into the same set. Therefore in this section the prediction is given by the weighted average of the flow at these neighbouring pixels. Denote the set of 8 neighbours by $\mathcal{N}$, the prediction step is performed by a low-pass filtering

$$\begin{aligned} u^{\tau+\frac{1}{2}} &= \frac{\sum_{i\in\mathcal{N}} \omega_i u_i^\tau}{\sum_{i\in\mathcal{N}} \omega_i} \\ v^{\tau+\frac{1}{2}} &= \frac{\sum_{i\in\mathcal{N}} \omega_i v_i^\tau}{\sum_{i\in\mathcal{N}} \omega_i} \end{aligned} \tag{7.4}$$

The weights $\omega_i, i = 1, \cdots, 8$ is to steer the flow smoothness anisotropically. It can be given by intensity contrast, occlusion labeling or spatial distance, as in (Xiao *et al.* 2006). In this work, we use two weight functions, which are the simplified $\omega_{intensity}$ and $\omega_{consistency}$ defined in Chapter 4,

$$\omega_i = \frac{1}{\sqrt{\left(u^\tau - u_i^\tau\right)^2 + \left(v^\tau - v_i^\tau\right)^2 + \epsilon}} \cdot \frac{1}{\sqrt{\left(E - E_i\right)^2 + \epsilon}}. \tag{7.5}$$

where $\epsilon = 0.001$ to prevent the denominators from being zero. It can be seen that if the neighbouring pixel has large intensity inconsistency or motion inconsistency with the current pixel, it is weighted down. Therefore the low pass filtering Eq. 7.4 steers the regularization toward pixels that have consistent pattern, and alleviates over-smoothing across the motion boundaries.

7.2.3 M-Step for Rotation

The flow vector obtained from the expectation step is used to warp the local patch. The M-step estimates the optimal flow increase $[\ \delta u \quad \delta v\ ]$ given the intensity variation of warped patch, by minimizing the deviation from the preservation of a feature. To adapt the M-step to rotation, this feature should be rotational invariant. This section proposes a new feature descriptor for this purpose.

Let $\vec{d}(t)$ be a pixel's quantized dominant direction at time t. As discussed before, $\vec{d}(t)$ falls into one of the 5 sets illustrated in Fig.7.2.1. Denote the 8 directions, in clockwise

order, by $\vec{d}_0(t), \cdots, \vec{d}_7(t)$ with $\vec{d}_0(t) = \vec{d}(t)$. The first order partial derivatives in these directions form a vector $\vec{f}(t)$,

$$\vec{f}(x,y,t) = \left[\frac{\partial E(x,y)}{\partial \vec{d}_0(t)}, \cdots, \frac{\partial E(x,y)}{\partial \vec{d}_7(t)}\right]. \tag{7.6}$$

Because each component of this descriptor hinges on the intrinsic directions, it is rotational invariant. It should be noted that, however, approximating the partial derivatives by the differencing scheme Eq. 7.3 may weaken the invariance, if the image resolution is low.

The invariance of $\vec{f}(x,y,t)$ provides a constraint for the pixel's displacement between the first image and the warped second image $(\delta u^{\tau+\frac{1}{2}}, \delta v^{\tau+\frac{1}{2}})$, which is in the same form of the brightness constancy assumption

$$\frac{d\vec{f}}{dt} = \vec{f}_x \delta u^{\tau+\frac{1}{2}} + \vec{f}_y \delta v^{\tau+\frac{1}{2}} + \vec{f}_t^{\tau+\frac{1}{2}} = 0, \tag{7.7}$$

where $\vec{f}_t^{\tau+\frac{1}{2}}$ is computed on the first image and the warped second image. Furthermore, we assume the local motion model to be affine,

$$\begin{aligned} \delta u^{(k,\tau+\frac{1}{2})} &= \alpha_1 x^{(k)} + \alpha_2 y^{(k)} + \alpha_3 \\ \delta v^{(k,\tau+\frac{1}{2})} &= \alpha_4 x^{(k)} + \alpha_5 y^{(k)} + \alpha_6, \end{aligned} \tag{7.8}$$

where the superscript k is the index of a generic pixel in the neighbourhood, and $\alpha_1, \cdots, \alpha_6$ are the motion parameters. Combining Eq.7.7 and Eq. 7.8, the following equation can be easily verified:

$$\begin{aligned} -\vec{f}_t^{(k,\tau+\frac{1}{2})} &= \vec{f}_x^{(k)}(\delta u^{\tau+\frac{1}{2}} + \alpha_1 \Delta x^{(k)} + \alpha_2 \Delta y^{(k)}) + \\ &+ \vec{f}_y^{(k)}(\delta v^{\tau+\frac{1}{2}} + \alpha_4 \Delta x^{(k)} + \alpha_5 \Delta y^{(k)}), \end{aligned} \tag{7.9}$$

where $\Delta x^{(k)} = x^{(k)} - x$ and $\Delta y^{(k)} = y^{(k)} - y$. To alleviate the distortion caused by outliers, the equations are weighted by the motion consistency measured from the flow obtained from the previous step, i.e.,

$$\begin{aligned} -\omega^{(k,\tau+\frac{1}{2})}\vec{f}_t^{(k,\tau+\frac{1}{2})} &= \omega^{(k,\tau+\frac{1}{2})}\vec{f}_x^{(k)}(\delta u^{\tau+\frac{1}{2}} + \alpha_1 \Delta x^{(k)} + \alpha_2 \Delta y^{(k)}) + \\ &+ \omega^{(k,\tau+\frac{1}{2})}\vec{f}_y^{(k)}(\delta v^{\tau+\frac{1}{2}} + \alpha_4 \Delta x^{(k)} + \alpha_5 \Delta y^{(k)}), \end{aligned} \tag{7.10}$$

with

$$\omega^{(k,\tau+\frac{1}{2})} = \frac{1}{\sqrt{\left(u^{\tau+\frac{1}{2}} - u^{(k,\tau+\frac{1}{2})}\right)^2 + \left(v^{\tau+\frac{1}{2}} - v^{(k,\tau+\frac{1}{2})}\right)^2 + \epsilon}}. \tag{7.11}$$

Note that other weighting functions discussed in Chapter 4 can be applied to remove or inhibit outliers. The collection of all pixels' Eq.7.10 form an overdetermined system $A\vec{X} = \vec{b}^{\tau+\frac{1}{2}}$, where $\vec{X} = [\delta u^{\tau+\frac{1}{2}}, \delta v^{\tau+\frac{1}{2}}, \alpha_1, \alpha_2, \alpha_4, \alpha_5]^T$.

Although the weighting functions can weight down pixels with inconsistent patterns, there may still be outliers in the local system. As discussed in Chapter 2, to achieve robustness to outliers, $L1$ norm penalizing terms may be used; however, the associated side effects such as non-differentiability have to be handled with extra effort. Thus it is preferable to use $L1$ penalization if the presence of outliers is detected in the system and $L2$ penalization otherwise. To choose the penalizing functions adaptively, the COIN measure introduced in Section 3.3 is employed to detect occurrences of inconsistent motion. In particular, the measure values are initialized to be zeros. For the measure computed at each stage τ of the optimization, the mean and standard deviation are computed over the whole image. A threshold T is determined as the mean value plus one standard deviation. If the measure at a pixel is smaller than the threshold, the motion in the local surrounding area is considered to be consistent; In this case, the flow refinement at stage $\tau + 1$ is estimated by the LSE solution, i.e.,

$$\vec{X}^{\tau+1} = \underset{\vec{X}}{\operatorname{argmin}} \left\| A\vec{X} - \vec{b}^{\tau+\frac{1}{2}} \right\|_2 .$$

Elsewhere, the local area is partitioned to two segments along the isophote direction, and a sub-system $A_i\vec{X}_i = \vec{b}_i^{\tau+\frac{1}{2}}, i \in [1,2]$ is formed for each segment. The least absolute error solution is computed for each segment, i.e.,

$$\vec{X}_i^{\tau+1} = \underset{\vec{X}}{\operatorname{argmin}} \left\| A_i\vec{X} - \vec{b}_i^{\tau+\frac{1}{2}} \right\|_1 , \quad i = 1,2$$

which is robust to outliers, by linear programming. The solution that best satisfies the brightness constancy assumption recovers the flow increase at the current pixel. Therefore, the flow obtained by the M-step is given by

$$\begin{aligned} u^{\tau+1} &= u^{\tau+\frac{1}{2}} + \delta u^{\tau+\frac{1}{2}} \\ v^{\tau+1} &= v^{\tau+\frac{1}{2}} + \delta v^{\tau+\frac{1}{2}} \end{aligned} \tag{7.12}$$

7.3 Experimental Results

The proposed flow computation method aims at providing a reliable initial point for motion analysis tasks that involve fast rotation. One particular example is human activity recognition (e.g., (Wang *et al.* 2009),). Therefore, the method is tested on benchmark human motion sequences depicting a variety of scenarios. Moreover, to evaluate its generality in applications where rotation is not involved, it is also tested on Middlebury sequences quantitatively.

7.3.1 Experiments on Real Sequences

The first experiment evaluates the performance of the proposed method for large displacement caused by fast rotation; and compares to recent works that aim at large displacement recovery. The experiment is conducted on a pair of frames taken from the **HumanEva-II** dataset, which have also been used for the experiments in Section 6.5, as shown in Fig 7.3. Despite the large displacement and fast rotation presented in area of the right leg and foot, our technique recovers the flow reliably. The colorcoded result shows that different moving parts are well recovered with consistent motion, and hence one can clearly tell the motion type of each body segment. Compared to the results reported on the same test frames (Brox *et al.* 2009) (and the results by (Brox *et al.* 2004) reported therein), where gradient constancy is assumed, the presented method faithfully recovers the leg and foot motion that is too large to be captured by (Brox *et al.* 2004) and (Brox *et al.* 2009). This confirms the advantage of using the proposed rotation invariant feature over the gradient feature. Compared to a recent technique (Steinbruecker *et al.* 2009), the presented flow computation shows comparable performance, but preserves the motion boundary between the two legs more sharply. Fig.7.4 provides the comparison of the zoomed-in views in this area.

The experiments on two outdoor sequences **Minicooper** and **Walking**, which are taken from the Middlebury and Brown University Datasets, demonstrate the improvement by the motion boundary handling. Moreover, the results are compared to SIFT-Flow (Liu *et al.* 2008), which computes the flow based on the preservation of a rotational invariant feature SIFT and a global regularization. This method, as it is designed for scene matching rather than accurate flow computation, does not take motion boundaries into consideration. In the **Minicooper** sequence (Fig. 7.6), the shading on the right arm causes brightness change and some moving structures are of very small scale (e.g.,

left arm, hair, and nose tip). In the **Walking** sequence (Fig. 7.5), self occlusion occurs at both arms. Nevertheless, motion boundary in both sequences are well preserved by the proposed method, whereas SIFT-Flow shows blocky effects. This example illustrates that motion boundary handling is necessary if the feature on which flow computation is based has a large image footprint.

Fig 7.7 demonstrates the importance of bilateral filtering in the intrinsic directions (the E-step adapted to rotation). In this example, the camera is moving to the right of the scene. The results are compared to the bilateral filtering on horizontal, vertical and diagonal neighbours. The prediction by the nearest neighbours shows inferior performance in area of the quickly rotating arm as well as non-textured regions. This is because the oriented E-step "fills in" the edgy region by the flow of the "corners" along the isophote direction, and thereby preserves the regularity adaptively.

7.3.2 Quantitative Evaluation

Experiments are also conducted on benchmark sequences with ground truth **RubberWhale**, **Grove2**, **Venus**, **Hydrangea** and **Dimetrodon** to evaluate the generality of the proposed method, because these sequences contain a variety of motion patterns. The numerical results are presented in Table 7.1, which also lists the results by the local computation proposed in Chapter 4, and by the global computation proposed in Chapter 6, as well as by the Local Grouping method (Ren 2008) and Fourier-Mellin Transform (FMT) (Ho and Goecke 2008).

On comparison, the proposed scheme performs better than the local or global computation presented in other chapters on **Venus**, **Grove2** and **RubberWhale**. These sequences contain substantial motion boundaries which coincide with object boundaries. Therefore, the combination with the local system effectively alleviates the oversmoothing caused by the global computation. Fig. 7.8 shows an example on **Grove2**, which illustrates the regions where the proposed computation has better accuracy than the global computation. It can be seen that such regions are around motion boundaries. Moreover, these boundaries are sharp, i.e., the two motion patterns at each side of the boundary have large contrast.

This is not only observed on **Venus** and **RubberWhale**, but also on **Hydrangea**, where the proposed computation performs worse on average than the global computation. Fig. 7.9 shows the regions where either method achieves better accuracy. It confirms

that the proposed method preserves sharp boundaries more faithfully. But the performance is weakened around boundaries which have low intensity or motion contrast. Presumably this is because of the weighting functions defined in Eq.7.5 and Eq.7.11, which detect outliers by the motion and intensity inconsistency. This suggests that more sophisticated weighting functions may be defined based on the knowledge on the scene obtained along with the process. Fig. 7.9 presents another example on **Dimetrodon**, where the integration of local computation seems to weaken the performance of the global computation in the area with low intensity contrast and shadows.

The comparison with other contemporary works shows significant improvement on most of the sequences to the FMT technique (Ho and Goecke 2008), which is designed for rotation; and shows comparable performance to the semi-local method of (Ren 2008).

Method	Venus		Grove2		RubberWhale		Dimetrodon		Hydrangea	
	AAE	AEP	AAE	AEP	AAE	AEP	AAE	AEP	AAE	AEP
proposed	5.57	0.35	**2.89**	**0.20**	**4.69**	**0.15**	3.18	0.16	4.24	0.34
Chap.4	7.86	0.58	3.30	0.24	5.01	0.16	2.80	0.14	4.56	0.40
Chap.6	7.09	0.40	3.54	0.26	5.03	0.16	**2.64**	**0.14**	**2.68**	**0.25**
Grouping	**3.93**	0.26	-	-	5.32	0.17	3.34	0.17	2.77	0.26
FMT	5.51	**0.14**	-	-	10.07	0.26	7.33	0.18	11.83	0.56

Table 7.1. Comparison of the proposed techniques with two contemporary methods. The results by other works are taken from the original publications. The bold number in each column indicates the best performance on the corresponding sequence.

7.4 Summary

This chapter has described another approach to combine local and global computation, which has also been generalized to recover fast rotation. First, the flow computation is formulated as an iterative Expectation-Maximization process, which allows explicit interaction between local and global flow computation and flexibility for customization. Second, a new feature descriptor is introduced that is robust to significant image or object rotation, which facilitates adjusting the expectation and maximization steps to fast rotation. The experimental results have demonstrated the benefit of employing the proposed rotational robust feature and directional bilateral filtering in flow

computation for fast rotation. The advantage of the proposed scheme has also been confirmed over using Fourier-Mellin transform in the log-polar coordinates on many test sequences.


Frame 546


Frame 550


overlaid

NOTE:
These images are included on page 110 of the print copy of the thesis held in the University of Adelaide Library.

colorwheel
(Brox *et al.* 2009)
(Brox *et al.* 2004)

warping
(Brox *et al.* 2004)

patch matching
(Brox *et al.* 2009)

NOTE:
These images are included on page 110 of the print copy of the thesis held in the University of Adelaide Library.

colorwheel
(Steinbruecker *et al.* 2009)

complete matching
(Steinbruecker *et al.* 2009)


Middlebury colorwheel


proposed method

Figure 7.3. Test results on the frame 546 and 550 from a benchmark **HumanEva-II** sequence by previous methods and the proposed methods. Results by previous methods are taken from the corresponding publications.

Zoom-In view

NOTE:
This image is included on page 111 of the print copy of the thesis held in the University of Adelaide Library.

(Brox *et al.* 2009)

NOTE:
This image is included on page 111 of the print copy of the thesis held in the University of Adelaide Library.

(Steinbruecker *et al.* 2009)

the proposed method

Figure 7.4. A Zoom-In comparison of the results reported in Fig. 7.3

(a)

(b)

(c)

(d)

(e)

Figure 7.5. Experimental results on sequence Walking. (a): The overlaid image of the two test frames. (b): the motion boundary probability measured by the COIN measure (Chapter 3) of the image in an intermediate stage. Dark colour indicates higher probability. (c): The flow computed by SIFT-Flow (Liu *et al.* 2008). (d): The flow computed by the proposed method without boundary handling. (e): the flow computed with motion boundary handling.

(a)

(b)

(c)

(d)

(e)

Figure 7.6. Experimental results on sequence **MiniCooper**. (a): The overlaid image of the two test frames. (b): the motion boundary probability measured by the COIN measure (Chapter 3) of the image in an intermediate stage. Dark colour indicates higher probability. (c): The flow computed by SIFT-Flow (Liu *et al.* 2008). (d): The flow computed by the proposed method without boundary handling. (e): the flow computed with motion boundary handling.

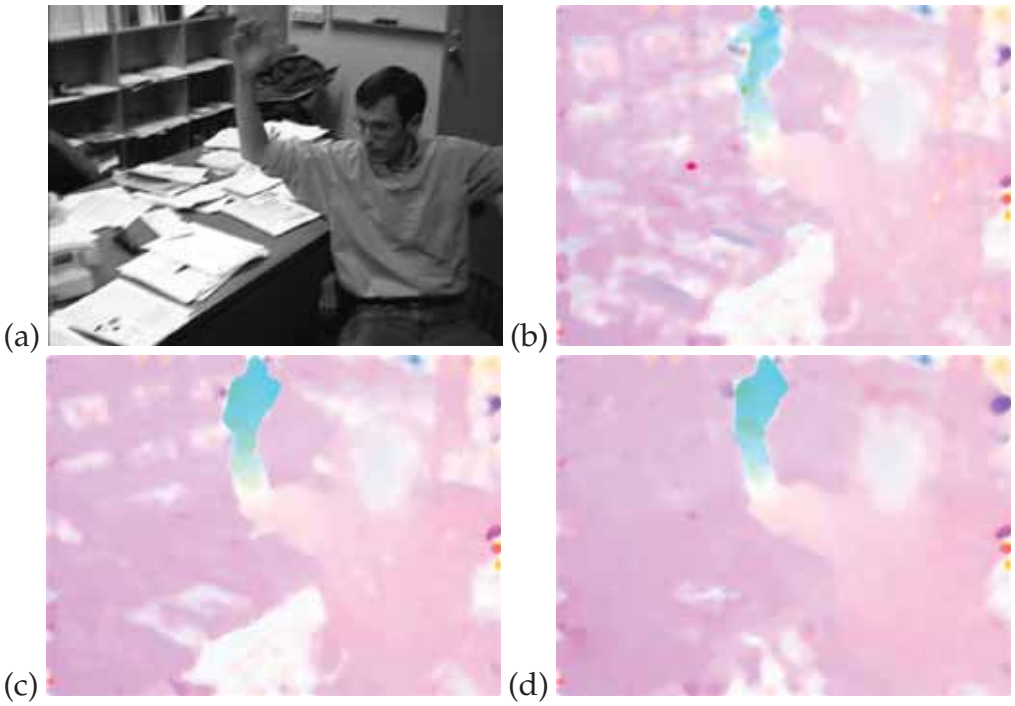


Figure 7.7. Experimental results on sequence **Moving**, taken from the Brown University dataset. (a): The overlaid image of the two test frames. (b): Flow computation by the expectation-maximization scheme without adaption to rotation or motion boundary handling. (c): Flow computation by the expectation-maximization adapted to rotation, but without motion boundary handling. (d): Flow computation by the proposed method. This improves the estimate on the outside edge of the arm, and in background regions including the pigeonholes and the top of the head.

Grove2 flow ground truth performance comparison

Figure 7.8. An illustration of regions where the proposed method performs better than the global method on Grove2. Bright pixels in the right image indicate regions where the angular error of the proposed method is at least 10° smaller than the global method.

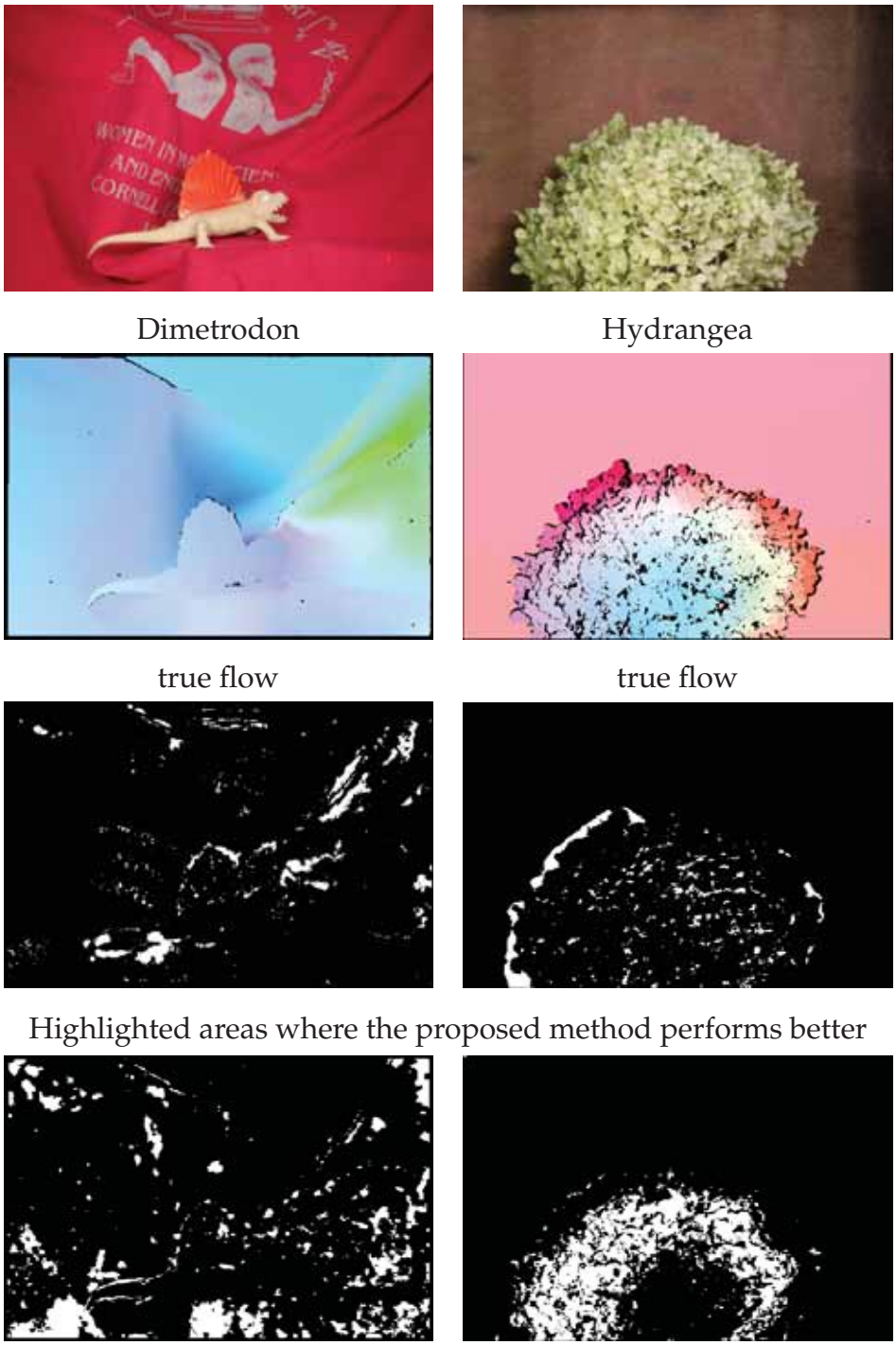


Figure 7.9. A comparison of the strength and weakness of the proposed and global methods. In the images of the 3rd row, highlighted regions indicate that the proposed method is at least 3° more accurate than the global method; whereas images of the 4th row illustrates the regions where the global method is more accurate.

Chapter 8

Summary and Discussion

8.1 Summary

This dissertation has presented several techniques for optical flow computation in the presence of discontinuous and fast rotation, which are two of the most challenging problems in this area.

The thesis started with detecting the presence of multiple motions in a local area, which violates most prior assumptions for flow computation. Chapter 3 has proposed motion inconsistency detection schemes, based on the evidence found from the pixels' 2D and 3D Hessian matrices. Moreover, it has derived a new motion inconsistency measure, based on evaluating the constraint inconsistency in a general linear system. This measure has also been shown to be a good confidence measure, which estimates the flow recovery reliability without having to compute the flow.

With the motion discontinuity detection investigated in Chapter 3, Chapter 4 proceeded to address the motion outliers in the local computation system. Different possible weighting functions have been studied to remove or inhibit the outliers. Empirical study has demonstrated promising improvement by the weighting functions based on intensity contrast, occlusion likelihood and dynamic motion inconsistency detection, especially around motion boundaries.

With the motion discontinuity preserving local computation described in Chapter 4, Chapter 5 has further improved the local computation by the combinational use of global subspace and smoothness constraints. The benefit of linking the local and global computation by motion segmentation has been discussed. The combination addresses two common problems in optical flow: the aperture problem and motion boundaries. Experimental results show that they reduce the error of calculated flow in these cases, to lower values than previously reported for these test sequences. The analysis has

shown the complementary effects of local and global constraints in correcting large (outlier) and small (inlier with noise) errors respectively.

Chapter 6 has described an approach to generalize flow computation from translational motion with small deformation to fast rotation, by formulating the flow computation on a special coordinate system which is locally oriented to the intrinsic directions. A numerical scheme to recover the optimal flow using these coordinates has been derived. Quantitative evaluation shows superior performance to a contemporary technique that utilizes polar coordinates for flow computation with fast rotation.

Chapter 7 has investigated a new combination scheme for local and global computation by Expectation-Maximization, which is also generalized to rotation by adjusting the E-step and M-step individually. The E-step regularizes the flow along the oriented sub-grid determined by the local structure; and the M-step updates the flow by introducing a new feature descriptor, which is robust to rotation, noise and additive illumination change. The method has been demonstrated to preserve sharp motion boundaries more faithfully than the local or global computation proposed in other chapters. Moreover, it also outperforms contemporary techniques for large displacement or rotation on both synthetic and real sequences.

8.2 Future Research

Although this thesis has presented new ideas for optical flow computation, and has shown advantages over previous methods, the performance can be further enhanced. Based on the investigation presented in this thesis, future research on the following topics are believed promising.

Spatially adaptive Lagrange multiplier In Chapter 6, the data terms and the regularization term are balanced by Lagrange Multipliers in the global energy function. These Lagrange multipliers play the role of adjusting the dependency of the optimization on each term. In the current literature, as well as in this work, these multipliers are predefined heuristically based on empirical experience, and they remain fixed for all the pixels. This predefinition lacks flexibility - in smooth regions, the optimization should count more on the regularization term; whereas around motion boundaries, the data terms should have the leading position in the optimization. Therefore, it is preferable to adaptively define the Lagrange

multipliers based on the nature of each pixel. So far to the best of our knowledge, there has not been any particular work about the optimal decision of these parameters in the literature of flow computation. This topic is believed worth investigating.

Comparison of different solution methods Variational flow techniques typically formulate flow computation by integrating the data terms and smoothness terms into an energy functional, the minimizer of which gives the flow field. There are two main branches of optimization methods in the literature: one is to use an auxiliary variable to link the data terms and the smoothness terms, and divide the optimization to two sub-processes. The other one keeps the terms combined in the associated Euler-Lagrange equation pair, and finds the optimal solution by numerical iterations (see Chapter 2). Both approaches have been used to obtain numerical results on benchmark sequences. However, they use different formulation of the energy functionals and different experimental settings. Chapter 7 qualitatively compares the proposed Expectation-Maximization framework to two state-of-the-art methods, which fall in these two categories, on a real sequence **HumanEva-II**. However, it would be interesting to quantitatively compare their performance for the same energy functional in the same experimental settings, which would provide important information for future research.

Generalization to scale change Optical flow for sequences that contain significant scale change has not been investigated much. However, scale change is rather common in real sequences, for example, when the camera changes zoom. A scale invariant feature is thus in need. Therefore, a possible direction for the future research based on this work is to develop the feature descriptor in Chapter 7 to become scale invariant.

More sophisticated motion segmentation In Chapter 5, the local and global computation are linked by motion segmentation, which is performed by a basic mean-shift clustering on the flow vectors. This baseline segmentation is limited to scenes that contain distinguishable motion patterns. A more sophisticated mean-shift clustering based on both motion, intensity and spatial proximity may further improve the performance. Moreover, in the current stage, the local flow computation is used as an input to the segmentation algorithm. Indeed, a pyramidal segmentation algorithm that can interact with the local computation is preferable.

8.3 Conclusion

With the increased demand for automatic processing of digital videos, computer vision systems that are able to infer motion from images with high accuracy are more and more in need.

The optical flow techniques presented in this work, although they are not yet of the precision and generality required of a fully automatic machine vision system, have demonstrated the ability to improve the performance of contemporary techniques in several challenging situations. Adaptively selecting and combining different optical flow constraints has proved useful in enhancing the accuracy and robustness of the flow computation, in the presence of motion discontinuity and fast rotation.

Appendix A

List of Test Sequences

?? This appendix lists the data sets used for evaluation throughout the thesis. For each data set, the test frame and the ground truth flow are shown and the property is specified.

A.1 MiddleBury Training Dataset

This data base was created to compare state-of-the-art flow computation techniques (Baker *et al.* 2007). Details about the creation of the datasets can be found in the authors' extended Technical Report (Baker *et al.* 2009). The training data sets and their ground truth are publicly available at `http://vision.middlebury.edu/flow/data/`. The authors also proposed using the colorwheel (shown in Fig. A.1) to densely code the flow vector recovered at each pixel. The colorwheel is used throughout the thesis to visualize the dense flow field. As the Middlebury datasets cover a variety of motion patterns, they are used in Chapter 3 to test the generality of the COIN measure, in Chapter 4 to evaluate the performance of different weighting functions, and in Chapter 6 to validate that the flow formulation designed for fast rotation have stable performance in different scenarios.

Figure A.1. The Middlebury colorwheel used to code the flow vectors.

A.1.1 Hidden Fluorescent Texture Sequences

RubberWhale This training sequence depicts several objects moving independently in the scene, as shown in Fig. A.2. It contains motion boundaries, occlusion, disocclusion and non-rigid motion. The largest displacement is about 4.6 pixels/frame. The sequence has 8 frames.

Hydrangea This sequence depicts a rotating Hydrangea against a moving background, as shown in Fig. A.2. It contains motion boundaries, occlusion, disocclusion and non-rigid motion, two moving objects. Low contrast background. The maximum motion is about 11 pixels/frame. The sequence has 8 frames.

Dimetrodon This sequence has only two frames. The first frame is demonstrated in Fig. A.2. The motion is non-rigid and a large area of the scene has minimal texture. The maximum flow is about 4.6 pixels/frame.

A.1.2 Computer Graphics Synthetic Sequences

Grove2 In this 8-frame synthetic sequence, the rising camera creates an elevated view. Meanwhile the camera is tilting down a little bit. The sequence contains motion discontinuities caused by the depth discontinuities. The maximum motion is about 5 pixels/frame. The test frame is shown in Fig. A.3.

Grove3 In this 8-frame sequence, the camera is rising to the left and tilting down. The sequences contains substantial parallax and motion discontinuities. The maximum motion is about 18 pixels/frame. The test frame is shown in Fig. A.3.

Urban2 This 8-frame synthetic sequence depicts an urban scene, which contains independently moving objects, including the camera. The camera is rising up to the right and panning left. Fig. A.3 shows the test frames. The sequence is also characterized by motion discontinuity, occlusion, disocclusion. It has large motion range, with the maximum motion up to 22 pixels/frame.

A.1.3 Modified Stereo Sequence

Venus This sequence has been widely used for stereo matching evaluation. In the Middlebury database, this stereo pair is modified to evaluate optical flow techniques. The scene undergoes rigid motion, but contains motion boundaries. It

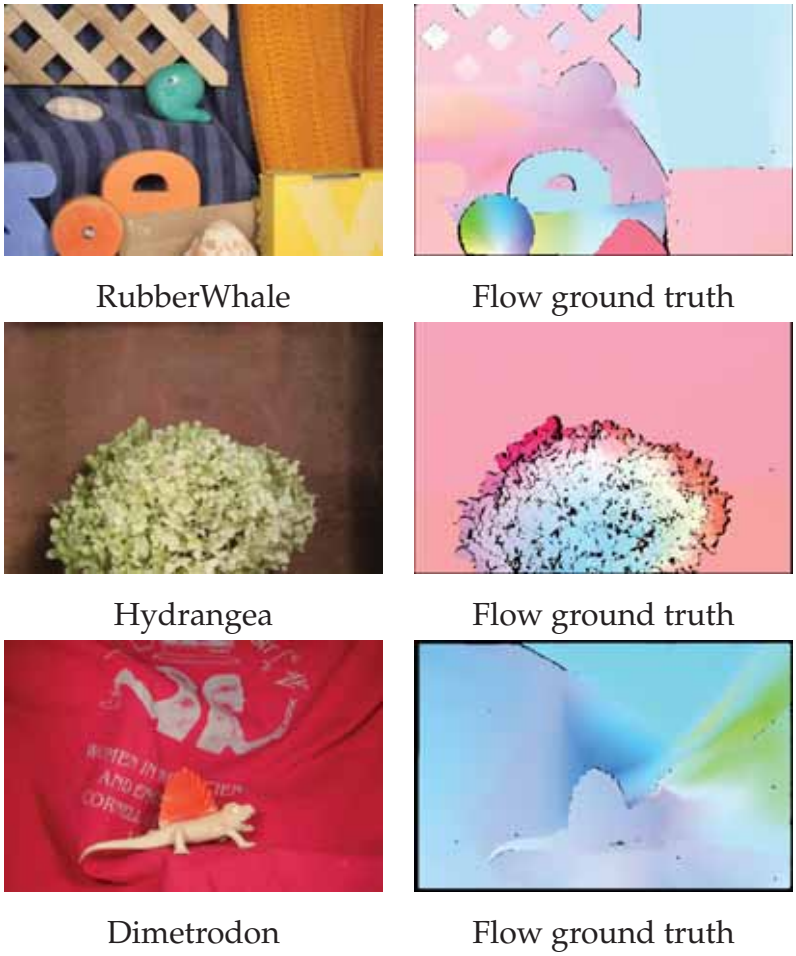


Figure A.2. Test frames and their color-coded ground truth flow of the Middlebudy hidden texture sequences RubberWhale, Hydrangea and Dimetrodon

has 2 frames only. The first frame and the ground truth flow are presented by Fig. A.4.

A.1.4 Real Sequence

MiniCooper A real sequence captured by a high speed camera (60 frames per second). Every other frame is provided. This sequence depicts a man pushing the back door of a minicooper. Fig. A.5 shows the test frame. It contains non-rigid human motion, with a large area affected reflection and shadows. As this is a real sequence, the ground truth is the intermediate frame (the apparent motion), rather than the 2D motion field. Algorithms tested on this sequence are evaluated by the interpolation error, which is obtained by warping the test frames by

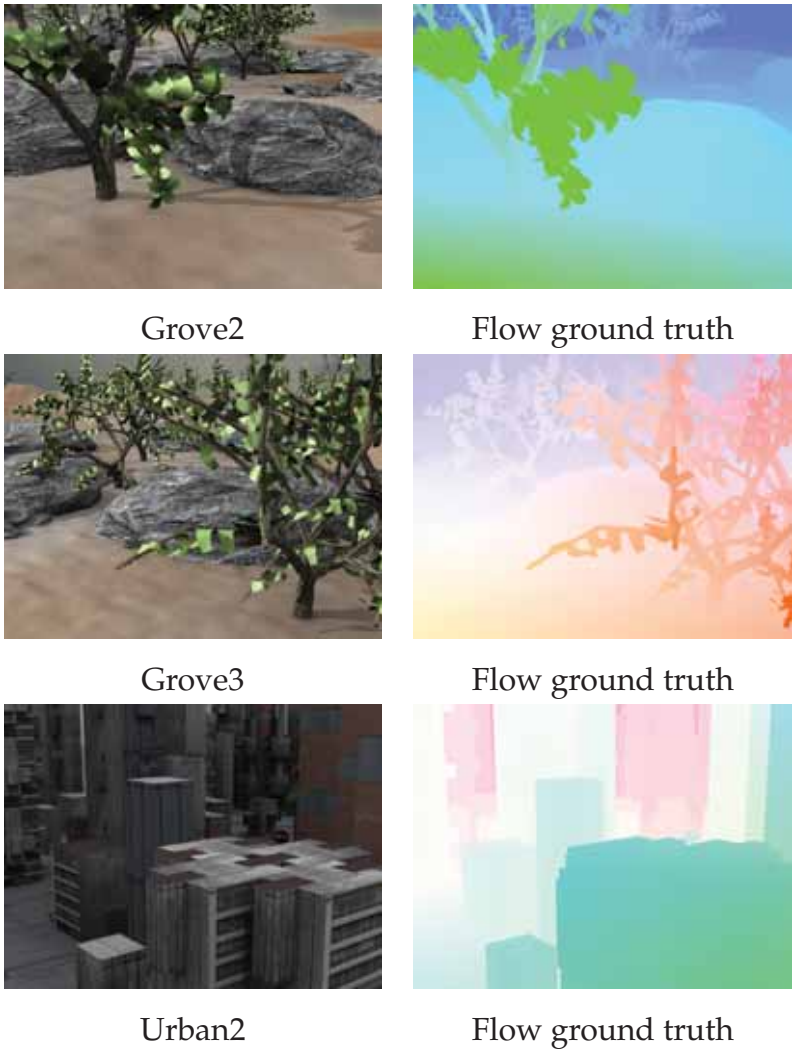


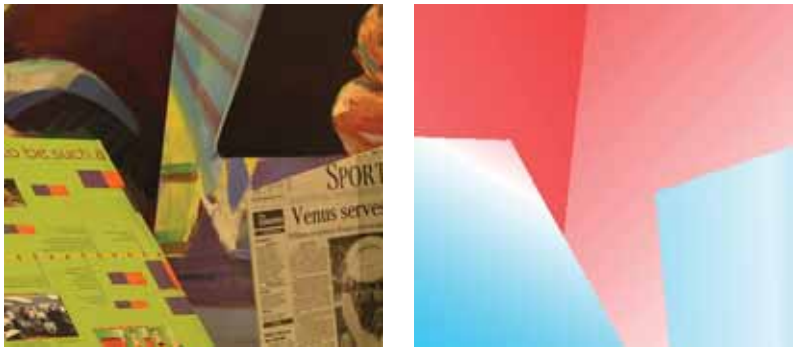
Figure A.3. Test frames and their color-coded ground truth flow of the computer graphics synthetic sequences Grove2, Grove3, Urban2.

the computed flow and taking the difference between the warped image and the true intermediate frame. The sequence has 8 frames.

A.2 McCane et al. Dataset

A.2.1 Computer Graphics Synthetic Sequences

This database created by McCane et al. by computer graphics rendering in their survey (McCane *et al.* 2001) for algorithm evaluation. Therefore the ground truth for the optical flow is the 2D motion field. Compared to the Middlebury datasets, these sequences are longer, the size of the images is smaller. However, the temporal aliasing



Venus Flow ground truth

Figure A.4. Stereo sequence Venus and its color-coded flow.

in this dataset is higher. These sequences have been used in Chapter 3, Chapter 4 to test the generality of the proposed COIN measure and the weighting functions for local computation. They are also used in Chapter 5 to test the proposed combination scheme of local and global computation, as the algorithm requires at least 9 frames of the sequence.

Short Street This sequence depicts a street scene with a car moving to the right of the scene, and the camera is moving to the left. Fig. A.6 illustrates the test frame. It contains independently moving objects. The motion is rigid. The sequence contains shadows and transparency (in the area of the car window). Compared to the Middlebury sequences, this sequence has small motion range. The maximum motion is about 4.5 pixels/frame. The data sets and the flow ground truth are available at `http://of-eval.sourceforge.net/`.

Long Street Similar to Short Street, this sequence depicts a street scene with two cars moving to each other, and the camera is moving to the left. See Fig. A.6 for the test frame. The maximum motion is about 2.6 pixels/frame. The data sets and the flow ground truth are available at `http://of-eval.sourceforge.net/`. Compared to **Short Street**, this sequence is of longer range.

Office In this sequence, the camera is moving into the office scene, as shown in Fig. A.6. This sequence contains substantial motion discontinuity caused by depth discontinuity, large area lack of texture, and a transparent window. The maximum motion is about 1.4 pixels/frame. The data sets and the flow ground truth are available at `http://sourceforge.net/projects/of-eval/files/of-eval/1.0/`.

Frame 10

Frame 11

The intermediate frame

Figure A.5. Real sequence MiniCooper. The groundtruth is given by the intermediate frame.

A.3 Miscellaneous

Yosemite A sequence synthesized by texture mapping the image onto the depth map. The original version with flying clouds is used in Barron et al.'s seminal survey paper (Barron *et al.* 1994), and has been widely used for evaluation since then. Black argued that the cloud is doing Brownian motion, and hence the ground truth flow for the clouds does not make sense (see FAQ, `http://www.cs.brown.edu/~black/`). Therefore contemporary techniques generally report results on the **Yosemite without cloud**. Compared to other sequences used in this thesis, Yosemite has relatively low difficulty, although it contains occlusion between the mountains and aliasing in the lower portion of the image. The displacement is as large as about 5 pixels/frame. It is interesting to note that **Yosemite** used to be the most challenging synthesized sequence when (Barron *et al.* 1994) was published.

Furthermore, the publicly available ground truth is quantized to 8 bits, which is a limit for accurate evaluation. This sequence is used in Chapter 3 to test the generality of the COIN measure. Fig. A.7 presents the test frame.

Walking A real sequence created by Black, available at `http://www.cs.brown.edu/~black/`. It depicts a person walking parallel to the camera. It contains self occlusion and large area of reflection. See Fig. A.8 for the test frame. This sequence is used in Chapter 7 to evaluate the optical flow computation with fast rotation, especially the motion boundary handling.

Moving A human motion sequence created by Black, available at `http://www.cs.brown.edu/~black/`. In the scene, a person is stretching the right arm to the phone, and the camera is moving too. The sequence has low resolution, fast human motion, nonrigid motion, independently moving objects and large area with little textures. See Fig. A.8 for the test frame. Chapter 7 conducts experiments on this sequence to test the performance of oriented diffusion in the textureless area.

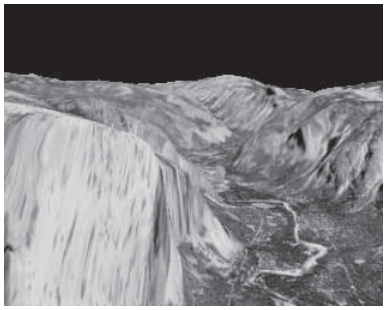
HumanEva-II This dataset is created by Brown University to evaluate techniques for articulated human motion estimation. It is available at `http://vision.cs.brown.edu/humaneva/`. The test frames used in this thesis depict a running person, whose right leg and foot undergo fast motion. The dataset is captured by high speed camera. To simulate large displacement, in the thesis test frame 546 and 550 are extracted as a frame pair. See Fig. A.8 for the test frame. As this sequence is used in state-of-the-art flow techniques for large displacement, it is used in both Chapter 6 and Chapter 7 for comparison.

Short Street color-coded trueflow

Long Street color-coded trueflow

Office color-coded trueflow

Figure A.6. Computer graphics rendered sequences Short Street, Long Street and Office. The ground truth flow is coded by the Middlebury colorwheel.



Yosemite

ground truth flow

Figure A.7. Yosemite without cloud. The ground truth flow is color-coded by Middlebury color wheel.

Walking

Moving

HumanEva-II

Figure A.8. Test sequences Walking, Moving and HumanEva-II. The pair of test frames are overlaid in each image.

Bibliography

ANANDAN-P. (1989). A computational framework and an algorithm for the measurement of visual motion, *International Journal of Computer Vision*, **2**(3), pp. 283–310.

BAKER-S., AND MATTHEWS-I. (2004). Lucas-kanade 20 years on: A unifying framework, *International Journal of Computer Vision*, **56**(3), pp. 221–255.

BAKER-S., ROTH-S., SCHARSTEIN-D., BLACK-M., LEWIS-J., AND SZELISKI-R. (2007). A database and evaluation methodology for optical flow, *Proc. International Conference on Computer Vision 2007*.

BAKER-S., SCHARSTEIN-D., LEWIS-J. P., ROTH-S., BLACK-M. J., AND SZELISKI-R. (2009). A database and evaluation methodology for optical flow, *Technical Report MSR-TR-2009-179*, Microsoft Technical Report.

BARRON-J., FLEET-D. J., AND BEAUCHEMIN-S. (1994). Performance of optical flow techniques, *International Journal of Computer Vision*, **12**(1), pp. 43–77.

BERTERO-M., POGGIO-T. A., AND TORRE-V. (1988). Ill-posed problems in early vision, *Proceedings of the IEEE 1988*, pp. 869–889.

BIRCHFIELD-S., AND PUNDLIK-S. (2008). Joint tracking of features and edges, *Proc. Computer Vision and Pattern Recognition 2008*, pp. 1–6.

BJÖRCK-Å. (1996). *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia.

BLACK-M., AND ANANDAN-P. (1996). The robust estimation of multiple motions: Parametric and piecewise-smooth flow-fields, *Computer Vision and Image Understanding*, **63**(1), pp. 75–104.

BROX-T., BREGLER-C., AND MALIK-J. (2009). Large Displacement Optical Flow, *Proc. Computer Vision and Pattern Recognition 2009*.

BROX-T., BRUHN-A., PAPENBERG-N., AND WEICKERT-J. (2004). High accuracy optical flow estimation based on a theory for warping, *Proc. European Conference on Computer Vision 2004 (LNCS)*, Vol. 3024, pp. 25–36.

BRUHN-A., AND WEICKERT-J. (2006). A confidence measure for variational optic flow methods.

BRUHN-A., WEICKERT-J., AND SCHNÖRR-C. (2005). Lucas/kanade meets horn/schunck: Combining local and global optic flow methods, *International Journal of Computer Vision*, **61**(3), pp. 211–231.

CAMPANI-M., AND VERRI-A. (1990). Computing optical flow from an overconstrained system of linear algebraic equations, *International Conference on Computer Vision 1990*, pp. 22–26.

CIPOLLA-R., AND BLAKE-A. (1997). Image divergence and deformation from closed curves, *International Journal of Robotics Research*, **16**(1), pp. 77–96.

COMANICIU-D., AND MEER-P. (2002). Mean shift: A robust approach toward feature space analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**, pp. 603–619.

DANIILIDIS-K., AND KRUGER-V. (1995). Optical flow computation in the log-polar plane, *Computer Analysis of Images and Patterns*, pp. 65–72.

DEMPSTER-A. P., LAIRD-N. M., AND RUBIN-D. B. (1977). Maximum likelihood from incomplete data via the em algorithm, *Journal of The Royal Statistical Society, Series B*, **39**(1), pp. 1–38.

FLEET-D., AND JEPSON-A. (1990). Computation of component image velocity from local phase information, *International Journal of Computer Vision*, **5**(1), pp. 77–104.

FREEMAN-W. T., AND ADELSON-E. H. (1991). The design and use of steerable filters, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **13**, pp. 891–906.

GALVIN-B., MCCANE-B., NOVINS-K., MASON-D., AND MILLS-S. (1998). Recovering motion fields: An evaluation of eight optical flow algorithms, *British Machine Vision Conference 1998*, pp. 195–204.

GOLUB-G. H., AND VAN LOAN-C. F. (1996). *Matrix Computations*, 3 edn, Johns Hopkins University Press.

HADAMARD-J. (1902). Sur les problmes aux drives partielles et leur signification physique, *Princeton University Bulletin*, pp. 49–52.

HAUSSECKER-H., SPIES-H., AND JÄHNE-B. (1998). Tensor-based image sequence processing techniques for the study of dynamical processes, *International Symposium on Real-time Imaging and Dynamic Analysis 1998*, pp. 704–711.

HEEGER-D. (1988). Optical flow using spatiotemporal filters, *International Journal of Computer Vision*, **1**, pp. 279–302.

HO-H. T. H., AND GOECKE-R. (2008). Optical flow estimation using fourier mellin transform, *Proc. Computer Vision and Pattern Recognition 2008*.

HORN-B. K. P., AND SCHUNCK-B. G. (1981). Determining optical flow, *Artificial Intelligence*, **17**, pp. 185–203.

HUBER-P. (1973). Robust regression: Asymptotics, conjectures and monte carlo, *Annnals of Statistics*, **1**(5), pp. 799–821.

IRANI-M. (2002). Multi-frame optical flow estimation using subspace constraints.

JEPSON-A., AND BLACK-M. (1993). Mixture models for optical flow computation, *Proc. Computer Vision and Pattern Recognition 1993*, pp. 760–761.

JU-X., BLACK-M. J., AND JEPSON-A. D. (1996). Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency, *Proc. Computer Vision and Pattern Recognition 1996*, pp. 307–314.

KANADE-T., AND OKUTOMI-M. (1991). A stereo matching algorithm with an adaptive window: Theory and experiment, *Proceedings of the 1991 IEEE International Conference on Robotics and Automation (ICRA '91)*, Vol. 2, pp. 1088–1095.

KONDERMANN-C., KONDERMANN-D., JÄHNE-B., AND GARBE-C. (2007). An adaptive confidence measure for optical flows based on linear subsapce projections, *German Pattern Recognition Symposium 2007*, pp. 132–141.

KONDERMANN-C., RUDOLF-M., AND GARBE-C. (2008). A statistical confidence measure for optical flows, *Proc. European Conference of Computer Vision 2008*, pp. 290–301.

LASKOV-P., AND KAMBHAMETTU-C. (2003). Curvature-based algorithms for nonrigid motion and correspondence estimation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **25**(10), pp. 1349–1354.

LEI-C., AND YANG-Y. H. (2009). Optical flow estimation on coarse-to-fine region-trees using discrete optimization, *Proc. International Conference on Computer Vision 2009*.

LINDEBERG-T., AKBARZADEH-A., AND LAPTEV-I. (2004). Galilean-diagonalized spatio-temporal interest operators, *Proc. International Conference on Pattern Recognition*, pp. 57–62.

LIU-C., YUEN-J., TORRALBA-A., SIVIC-J., AND FREEMAN-W. (2008). Sift flow: Dense correspondence across different scenes, *Proc. European Conference on Computer Vision 2008*, pp. III: 28–42.

LI-Y., AND HUTTENLOCHER-D. (2008). Learning for optical flow using stochastic optimization, *Proc. European Conference on Computer Vision 2008*, pp. 379–391.

LOWE-D. (2004). Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision*, **60**(2), pp. 91–110.

LUCAS-B., AND KANADE-T. (1981). An iterative image registration technique with an application to stereo vision, *DARPA Image Understanding Workshop 1981*, pp. 121–130.

MARTIN-D., FOWLKES-C., AND MALIK-J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues, *IEEE Transactions on Pattern Recognition and Machine Intelligence*, **26**(5), pp. 530–549.

MCCANE-B., NOVINS-K., CRANNITCH-D., AND GALVIN-B. (2001). On benchmarking optical flow, *Computer Vision and Image Understanding*, **84(1)**, pp. 126–143.

NAGEL-H. (1983a). Displacement vectors derived from second-order intensity variations in image sequences, *Computer Vision, Graphics and Image Processing*, **21**(1), pp. 85–117.

NAGEL-H.-H. (1983b). Constraints for the estimation of displacement vector fields from image sequences, *In International Joint Conference on Artificial Intelligence 1983*, pp. 945–951.

NIR-T., BRUCKSTEIN-A., AND KIMMEL-R. (2008). Over-parameterized variational optical flow, *International Journal of Computer Vision*, **76**(2), pp. 205–216.

NIU-Y., DICK-A., AND BROOKS-M. (2007). Discontinuity-preserving optical flow computation by a dynamic overdetermined system, *Proc. Digital Image Computing Techniques and Applications 2007*, pp. 352–359.

NIU-Y., DICK-A., AND BROOKS-M. (2008). A new combination of local and global constraints for optical flow computation, *Proc. Image and Vision Computing New Zealand 2008*.

OHTA-N. (1990). Movement vector detection with reliability indices, *IAPR Workshop on Machine Vision Applications 1990*, pp. 177–180.

OHTA-N. (1991). Image movement detection with reliability indices, *IEICE Transactions on Information and Systems*, **E74**, pp. 3379–3388.

OLSON, D. L.; DELEN-D. (2008). *Advanced Data Mining Techniques*, 1st edn, Springer.

OTTE-M., AND NAGEL-H.-H. (1995). Estimation of optical flow based on higher-order spatiotemporal derivatives in interlaced and non-interlaced image sequences, *Artificial Intelligence*, **78**, pp. 5–43.

REN-X. (2008). Local grouping for optical flow, *Proc. of Computer Vision and Pattern Recognition 2008*, pp. 1–8.

ROTH-S., AND BLACK-M. (2007). On the spatial statistics of optical flow, *International Journal of Computer Vision*, **74**(1), pp. 33–50.

SAND-P., AND TELLER-S. (2006). Particle video: Long-range motion estimation using point trajectories, *Proc. Computer Vision and Pattern Recognition 2006*, pp. 2195–2202.

SHECHTMAN-E., AND IRANI-M. (2007). Space-time behavior based correlation or how to tell if two underlying motion fields are similar without computing them?, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**, pp. 2045–2056.

SHI-J., AND TOMASI-C. (1994). Good features to track, *Proc. Computer Vision and Pattern Recognition 1994*, pp. 593–600.

SHULMAN-D., AND HERVE-J. (1989). Regularization of discontinuous flow fields, *Workshop on Visual Motion 1989*, pp. 81–86.

SIMONCELLI-E. (1993a). Bayesian multi-scale differential optical flow, *IEEE Workshop on Image and Multidimensional Signal Processing*, pp. 128–129.

SIMONCELLI-E. P. (1993b). Distributed representation and analysis of visual motion, Ph.D dissertation.

SRINIVASAN-S. V. A. G. R. L. (1992). How bees exploit optic flow: Behavioural experiments and neural models [and discussion], *Journal of Philosophical Transactions: Biological Sciences*, **337**(1281), pp. 253–259.

STEINBRUECKER-F., POCK-T., AND CREMERS-D. (2009). Large displacement optical flow computation without warping, *Proc. International Conference on Computer Vision 2009*.

STILLER-C., AND KONRAD-J. (1999). Estimating motion in image sequences - a tutorial on modeling and computation of 2d motion, *IEEE Signal Processing Magazine*, **16**, pp. 70–91.

SUN-D., ROTH-S., LEWIS-J., AND BLACK-M. (2008). Learning optical flow, *Proc. European Conference on Computer Vision 2008*, pp. III: 83–97.

TISTARELLI-M. (1996). Multiple constraints to compute optical-flow, *Pattern Recognition and Machine Intelligence*, **18**(12), pp. 1243–1250.

TISTARELLI-M., AND SANDINI-G. (1993). On the advantage of polar and log-polar mapping for direct estimation of time-to-impact from optical flow, *Pattern Analysis and Machine Intelligence*, **15**(4), pp. 401–410.

TOMASI-C., AND KANADE-T. (1991). Detection and tracking of point features, *Technical Report CMU-CS-91-132*, Carnegie Mellon University.

TOMASI-C., AND KANADE-T. (1992). Shape and motion from image streams under orthography: A factorization method, *International Journal of Computer Vision*, **9**, pp. 137–154.

TROBIN-W., POCK-T., CREMERS-D., AND BISCHOF-H. (2008). An unbiased second-order prior for high-accuracy motion estimation, *Proc. of the 30th German symposium on Pattern Recognition (LNCS)*, Vol. 5096, pp. 396–405.

TRUCCO-E., AND VERRI-A. (2006). *Introductory Techniques for 3-D Computer Vision*, Prentice Hall.

URAS-S., GIROSI-F., VERRI-A., AND TORRE-V. (1988). A computational approach to motion perception, *Biol. Cybern*, pp. 79–87.

VALGAERTS-L., BRUHN-A., AND WEICKERT-J. (2008). A variational model for the joint recovery of the fundamental matrix and the optical flow, *German Pattern Recognition Symposium 2008*.

WANG-S., HUANG-K., AND TAN-T. (2009). A compact optical flow based motion representation for real-time action recognition in surveillance scenes, *International Conference on Image Processing 2009*, pp. 1121–1124.

WEDEL-A., CREMERS-D., POCK-T., AND BISCHOF-H. (2009). Structure- and motion-adaptive regularization for high accuracy optic flow, *Proc. International Conference on Computer Vision 2009*.

WEDEL-A., POCK-T., BRAUN-J., FRANKE-U., AND CREMERS-D. (2008). Duality tv-l1 flow with fundamental matrix prior, *Proc. Image and Vision Computing New Zealand 2008*.

WEICKERT-J., AND SCHNÖRR-C. (2001). Variational optic flow computation with a spatio-temporal smoothness constraint, *Journal of Mathematical Imaging and Vision*, **14**(3), pp. 245–255.

WEICKERT-J., BRUHN-A., BROX-T., AND PAPENBERG-N. (2006). A survey on variational optic flow methods for small displacements, *Mathematical Models for Registration and Applications to Medical Imaging*, **10**, pp. 103–136.

WERLBERGER-M., TROBIN-W., POCK-T., WEDEL-A., CREMERS-D., AND BISCHOF-H. (2009). Anisotropic Huber-L1 optical flow, *Proc. British Machine and Vision Conference 2009*.

XIAO-J., CHENG-H., SAWHNEY-H., RAO-C., AND ISNARDI-M. (2006). Bilateral filtering-based optical flow estimation with occlusion detection, *Proc. of European Conference on Computer Vision 2006*, pp. 211–224.

XU-L., CHEN-J., AND JIA-J. (2008). A segmentation based variational model for accurate optical flow estimation, *Proc. European Conference on Computer Vision 2008*, pp. 671–684.

ZIMMER-H., BRUHN-A., WEICKERT-J., VALGAERTS-L., SALGADO-A., ROSENHAHN-B., AND H.-P.-S. (2009). Complementary optic flow, *Energy Minimization Methods in Computer Vision and Pattern Recognition 2009 (LNCS)*, Vol. 5681, pp. 207–220.