NISTIR 5763

A General Motion Model and Spatio-Temporal Filters for 3-D Motion Interpretations

Hongche Liu
Center for Automation Research
Department of Electrical Engineering
University of Maryland
College Park, MD 20742
and Intelligent Systems Division

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Bldg. 220 Rm. B124
Gaithersburg, MD 20899

NIST

NISTIR 5763

A General Motion Model and Spatio-Temporal Filters for 3-D Motion Interpretations

Hongche Liu
Center for Automation Research
Department of Electrical Engineering
University of Maryland
College Park, MD 20742
and Intelligent Systems Division

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Bldg. 220 Rm. B124
Gaithersburg, MD 20899

November 1995

U.S. DEPARTMENT OF COMMERCE
Ronald H. Brown, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Arati Prabhakar, Director

A General Motion Model and Spatio-Temporal Filters for 3-D Motion Interpretations

by Hongche Liu

Abstract

Motion cues are a rich source of visual information. Object boundaries due to motion parallax, perception of collision, and transparency provide crucial information to any mobile vision system, biological or robotic. This report presents the formulation, design, evaluation and implementation of a motion algorithm which accurately and efficiently interprets the above motion cues.

Numerous previous results have suggested that all the relevant motion information is contained in the image motion field, which is qualitatively characterized by optical flow. Optical flow has indeed been applied to many motion tasks such as 3-D scene reconstruction, locating the focus of expansion, image stabilization, scene segmentation, motion detection, obstacle avoidance, image compression, and medical diagnostics. In this report, we take a new approach to the overall motion problem by modeling general 3-D motion (and its projection on the 2-D image) instead of optical flow (2-D image motion). In theory, this results in more comprehensive and unambiguous interpretations of 3-D motion. In fact, the general framework of the motion model based approach has not only generated an elegant optical flow algorithm but also a direct solution to motion boundary extraction, transparent motion segmentation, and time-to-contact estimation. It has also led to more accurate estimations of 2-D motion, mainly as a result of many elegant properties provided by the spatio-temporal Hermite polynomial differentiation filters. These properties include Gaussian derivatives, orthogonality, and a recursive relation that resolves the nonlinearity of the motion model. In addition, the separable filter design has made real-time implementation feasible.

For performance characterization of the optical flow algorithm, we follow the scheme established by Barron [9] to evaluate accuracy; we use another scheme developed by Heyden [46] to evaluate motion boundary extraction; we also develop an accuracy-efficiency

performance plot to evaluate the cost-effectiveness of motion algorithms. These thorough evaluations against existing algorithms show that our algorithms presented in this report have excellent performance characteristics in accuracy, flexibility, noise sensitivity, and efficiency.

Chapter 1
Introduction

1.1 Motivation

Motion cues are a rich source of visual information. Object boundaries due to motion parallax, perception of collision, transparency are all crucial information to any mobile vision systems, biological or mechanical. Numerous previous research efforts have suggested that all the motion information is contained in the image motion field, which is qualitatively characterized by optical flow. Optical flow has indeed been applied to many motion tasks such as 3-D scene reconstruction (Prazdny[90] , Adiv[3] , Young & Chellappa[120] , Bruss & Horn[14] , Negahdaripour & Lee[82]), locating the focus of expansion (FOE) (Prazdny[91] , Guissin & Ullman[39] , Aloimonos & Duric[4]), image stabilization (Burt, et al.[15]), scene segmentation (Hartley[42] , Adiv[3] , Murray & Buxton[73]), motion detection (Duncan & Chou[30]), obstacle detection and avoidance (Prazdny[90] , Nelson & Aloimonos[79] , Young[122] , Young, et al. [123] [124]), video compression (Jain & Jain[51]), and medical diagnostics (Chou & Chen[23]). In this report, we take a new approach to the overall motion problem as we model general 3-D motion (and its projection on the 2-D image) instead of 2-D image motion (optical flow). The traditional approach is depicted in Fig 1. In this approach, all

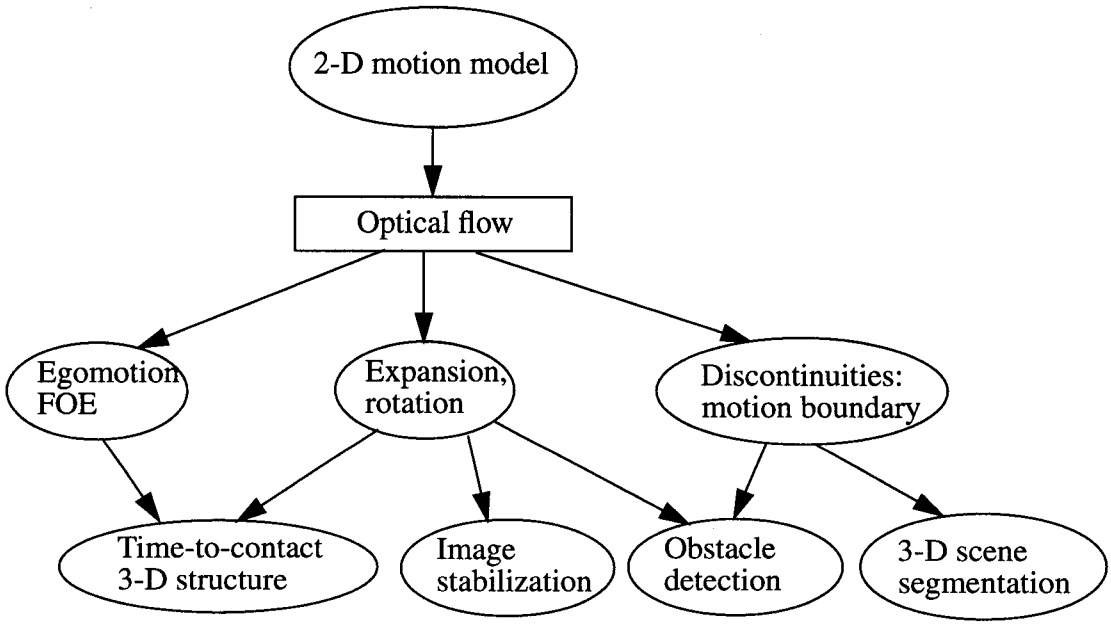


Fig 1. Traditional approach—all interpretations are based on optical flow

the motion interpretations are based on optical flow, computed using a 2-D image motion model. In our approach (Fig 2), all the 3-D motion model parameters can be used

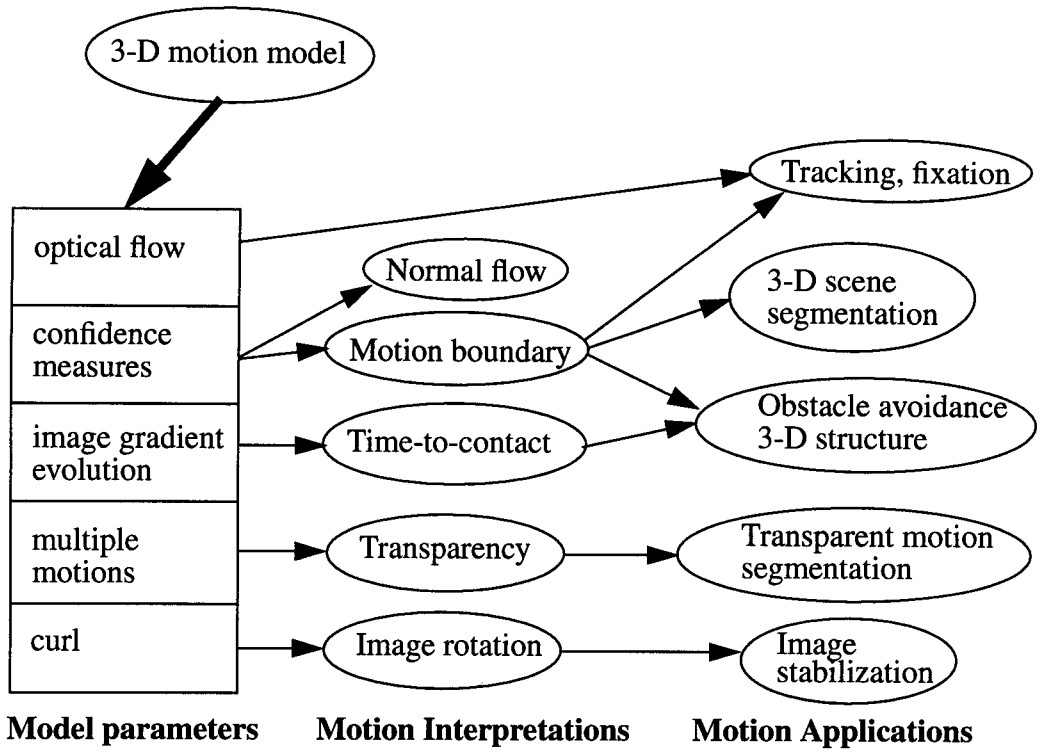


Fig 2. Our approach—model parameters directly used for motion interpretations.

for motion interpretations. Most motion applications can use some combinations of the 3-D motion parameters directly from our model. The model parameters generated in this framework include optical flow, confidence measures, image gradient evolution, a cue for multiple motions, and 2-D motion field curl.

Our primary goal in using the motion model based approach is to interpret and analyze motion in a comprehensive, sound and direct way to resolve many difficulties currently faced in motion research.

The first and foremost difficulty is the computation of optical flow. Image noise and the aperture problem both require the use of a large template (correlation based method) or large filters (gradient and frequency based methods). But a large template or filter covers more image area and therefore the traditional 2-D translational motion model becomes inadequate. For example, when the observer is approaching a scene, there is expansion as well as translation in the 2-D image. This is illustrated in Fig 3.1 and Fig 3.2. Model-

ing the 3-D motion will relax the constraint on the template or filter size.

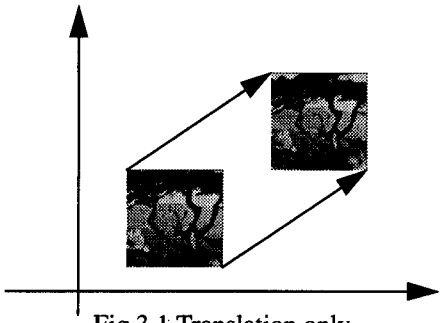


Fig 3.1 Translation only

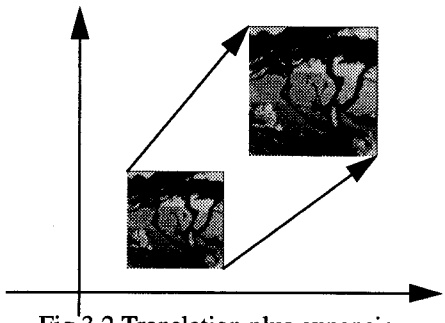


Fig 3.2 Translation plus expansion

Another apparent difficulty lies in motion boundary extraction. To deal with the aperture problem, previous motion estimation algorithms enforce a smoothness constraint or apply filters with appropriate filter support. In either approach, motion estimation on the boundary is not correct. Methods that extract motion boundaries from such flow field by detecting discontinuities will fail. A motion model based approach works similarly to the computational approach proposed by Canny [22] . We both model regularities (motion in our case; edge in Canny's) and irregularities (boundary in our case; noise in Canny's) and observe high response in one of the model parameters to locate boundaries (edges). This resolves the motion boundary conflict.

Another difficulty exists in the computation of flow field divergence. Divergence can be derived from flow field but if the flow field is derived based on the traditional uniform translational motion model, then the model itself is in conflict with the existence of flow field divergence. Our motion model based approach models divergence from the outset and avoids the conflict.

It is clear now that the traditional interpretation of motion based on optical flow has had many problems. Without a sound interpretation of 3-D motion, some motion algorithms and applications using optical flow may be self-contradictory.

In attaining our primary goal, we remain careful not to compromise the algorithms' accuracy and efficiency, which is crucial to real world motion applications. Indeed, current motion research is still far from being utilized to its fullest potential. The key problem is the combination of accuracy (precision and robustness) and efficiency. We realize that the above theoretical innovations are meaningless if the algorithms are not adapted to real world motion tasks. We have aspired to make the implementations as accurate and efficient as possible and at the same time make them applicable to real tasks. In doing so, we have been rewarded not only with an insight into some elegant numerical techniques but also with a new understanding of many issues of "applicability", including accuracy, efficiency, flexibility, etc.

1.2 Contributions

In theory, our motion model based approach results in more comprehensive and sound interpretations of 3-D motion. In fact, the general framework of the motion model based

approach has not only generated an elegant optical flow algorithm but also direct solutions to motion boundary extraction, transparent motion segmentation, and time-to-contact estimation. Motion boundary extraction is based on a model parameter which is analytically related to motion discontinuities. Transparent motions can be modeled and thus segmented. Flow field divergence is shown to be related to image gradient evolution, which is modeled in our formulation.

In practice, our approach has led to more accurate estimations of 2-D motion, mostly due to the 3-D model and the spatio-temporal Hermite polynomial differentiation filters employed. These filters provide many elegant properties including Gaussian derivatives, orthogonality, and a recursive relation that resolves the nonlinearity of the motion model. In addition, the separable filter design has made real-time implementation feasible.

Suppose the image size is S and the maximum motion velocity is V and also suppose that the algorithm is to output dense results. Traditional correlation algorithms performing spatial search or gradient based algorithms using 2-D filters are of complexity $O(V^2S)$. Several recent spatio-temporal filter based methods even have $O(V^3S)$ complexity. Our algorithm has achieved the lower bound of $O(VS)$.

Since there are many existing algorithms, we consider rigorous evaluation and performance characterization as integral parts of this report.

A thorough evaluation has demonstrated our optical flow algorithm's excellent performance. Both synthetic images with ground truth motion and real images are used. The criteria include accuracy, noise sensitivity, algorithm's flexibility, and efficiency. Our motion boundary extraction is also very accurate in extracting and locating boundaries in a quantitative evaluation scheme. Our transparent motion segmentation and collision avoidance algorithms are both evaluated using real images since there are few comparable implementations. Both show good results.

Some of our evaluations, although based on previously reported schemes, contains some more interesting aspects which are fundamental to the understanding of the underlying algorithms. For example, noise sensitivity, while very important in real applications, is not covered in the original optical flow evaluation by Barron [9] . In addition, evaluation of the localization error of motion boundaries is separated from evaluation of the error due to incorrect flow, so that the evaluation points out the algorithm's problem more specifically.

Since our algorithms perform well in many respects, it is natural that we address the issues of real-time implementation. These issues have been neglected because many previous motion algorithms have struggled with either accuracy or efficiency. Once these two conflicting criteria are jointly analyzed, many interesting points arise and an analysis of cost-effectiveness becomes necessary. Our analysis of the accuracy-efficiency trade-off is one of the first such studies and will be very useful in implementations.

All the algorithms presented in this report have computer implementations available for reproducing our results and for users interested in using our algorithms for other motion applications.

Real world tasks that can benefit from the work reported here include unmanned vehicle (obstacle avoidance, range recovery), reconnaissance, surveillance, target acquisition (image stabilization, object tracking, motion detection), video compression (motion estimation, segmentation), etc.

In summary, this report offers a sound theory that interprets 3-D motion unambiguously, a set of algorithms that perform well in terms of accuracy and efficiency, rigorous evaluations that clearly characterize the algorithms' performance, and real-time implementations that are suitable for many real world applications.

1.3 Organization of the report

The remainder of the report is organized follows. Chapter 2 introduces the general motion model and the spatio-temporal filters as a theoretical basis for the entire report; it also presents our optical flow algorithm in this framework. Chapter 3 evaluates the optical flow algorithm's accuracy against several existing ones using Barron's scheme [9] . We also evaluate the algorithms' noise sensitivity, output flexibility and applicability to real world tasks such as obstacle detection. Chapter 4 describes the motion-model-based boundary extraction method and its evaluations. Chapter 5 introduces a visual threat cue—image gradient evolution and its application to obstacle avoidance. Chapter 6 models and formulates transparent motion in our framework and presents a solution to transparent motion segmentation. Chapter 7 presents our real-time implementations and efficiency evaluation. It also addresses many important issues regarding real-time implementations.

Since each motion problem is dealt with separately, relevant previous work is discussed in each chapter, and experiments are also presented separately in each chapter.

Chapter 8 concludes the report with possible future research directions.

Chapter 2

A General Motion Model Approach to Optical Flow

Traditional optical flow algorithms assume local image translational motion and apply simple image filtering techniques. Recent studies have taken two separate approaches toward improving the accuracy of the computed flow: the application of spatio-temporal filtering schemes and the use of advanced motion models such as the affine model. Each has achieved some improvement over traditional algorithms in specialized situations but the computation of accurate optical flow for general motion has been elusive. In this chapter, we exploit the interdependency between these two approaches and propose a unified approach. The general motion model we adopt characterizes arbitrary 3-D steady motion. Under perspective projection, we derive an image motion equation that describes the spatio-temporal relation of gray-scale intensity in an image sequence, thus making the utilization of 3-D filtering possible. However, to accommodate this motion model, we need to extend the filter design to derive additional motion constraint equations. Using Hermite polynomials, we design differentiation filters, whose orthogonality and Gaussian derivative properties insure numerical stability; a recursive relation facilitates application of the general nonlinear motion model while separability promotes efficiency. The resulting algorithm produces accurate optical flow and other useful motion parameters. It is evaluated quantitatively using the scheme established by Barron, et al.[9] and qualitatively with real images.

2.1 Introduction

Research in the field of optical flow, originating from Gibson[36] , has generated many algorithms in the past two decades, and at the same time has led to numerous applications. To name a few, optical flow can be used to compute three-dimensional motion and structure (Prazdny[90] , Adiv[3] , Young & Chellappa[120] , Bruss & Horn[14] , Negahdaripour & Lee[82] , Gupta[40]), to locate the focus of expansion (Prazdny[91] , Guissin & Ullman[39] , Aloimonos & Duric[4]) or a moving observer's direction of heading, to segment independently moving objects (Hartley[42] , Adiv[3] , Murray & Buxton[73]), to extract boundaries (Thompson, et al.[106]), to detect motion (Duncan & Chou[30]), to stabilize images (Burt, et al.[15]), to perform obstacle detection and avoidance (Prazdny[90] , Nelson & Aloimonos[79] , Young[122] , Young, et al. [123] [124] , Coombs, et al.[25]), and to analyze medical video (2D echocardiographic images) for diagnosis (Chou & Chen[23]). All of these applications use optical flow data in a quantitative way. Although it is true that the optical flow field is not necessarily equal to the motion field[110] , relative accuracy in optical flow is very important in obtaining qualitative properties of motion. For example, discontinuities in optical flow

are useful qualitative properties that can be used to locate motion boundaries more precisely if the flow is more accurate.

Evidently, the importance of accurate optical flow cannot be overemphasized. In view of this, Barron, Fleet, and Beauchemin[9] developed a scheme for evaluating optical flow algorithms, highlighting the current endeavor to achieve greater accuracy.

However, attempts to obtain accurate motion estimates are impeded by three sources of error: sensor noise, brightness changes over time, and quantization error. Sensor noise is caused by optical or electronic irregularities. Brightness changes occur due to changing light sources, shadowing, camera aperture adjustments, or shading of a Lambertian or specular surface. Quantization error is inherent in digital images. These factors represent physical challenges that cannot be overcome but can only be mitigated by image processing techniques. In addition to dealing with these physical errors, are there other ways of improving the current best optical flow algorithms? To answer this, we need to review other systematic difficulties that have been facing us.

The first difficulty is the aperture problem or the ill-posed nature of the flow computation problem. Traditional optical flow algorithms have worked on finding reasonable constraints to solve this problem [4,48,68,78,97,109]. Although many ideas have been proposed, the desired accuracy has not been achieved due to two factors: lack of attention to better filtering schemes and the use of the simple assumption of uniform translational motion. A good filtering scheme is essential in dealing with the aforementioned sources of error, and uniform translation is insufficient for describing general 3-D motion.

Recent studies have taken two separate approaches to improving accuracy. The first is the application of spatio-temporal filtering schemes [34,44,58,100,113]. The second is the use of improved motion models [10,19,23,42,77,100,111,116] such as the affine model. The fact that these two approaches are actually complementary to each other will become clear as we analyze their individual advantages and disadvantages.

An intuitive idea for achieving better accuracy when applying a filter is to increase its support. A large support alleviates the aperture problem, smooths out more noise, avoids aliasing, and reduces quantization and truncation errors of the filter. For example, to estimate temporal derivatives, recent research has used temporal filters on multiple frames instead of simple successive frame differencing (Fig 4.1). In fact, more sophisticated spatio-temporal filters[18] [44] [34] , i.e., 3-D filters (Fig 4.2), have been developed to estimate image properties.

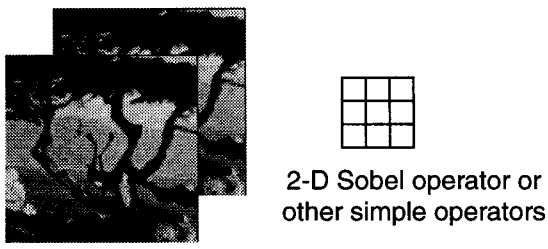


Fig 4.1 Traditional filtering approach

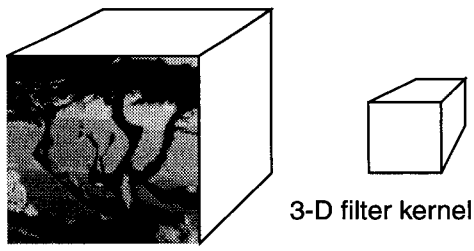


Fig 4.2 Spatio-temporal filtering approach

However, an arbitrary increase in filter support is not adequate due to the second difficulty commonly experienced: the lack of a good motion model. Since traditional algorithms use simple filtering schemes and small filter supports, they can safely assume uniform translational motion described in the following image motion equation:

$$I(x, y, t) = F(x-ut, y-vt) . \quad (1)$$

Once the spatio-temporal filters are applied and the support increases, the motion within the filter region becomes more complicated. Moreover, if we consider perspective projection of the 3-D motion onto the 2-D image plane, the problem gets worse. For example, a forward moving observer sees a diverging scene in which an image patch can undergo both translation and expansion* (Fig 2). Generally, divergence, curl, and deformation[57] as well as translation exist in 2-D image motion. Unless the motion model accommodates all these motion parameters, there is a limit to the useful filter size. Recent research has proposed the affine motion model to cope with this difficulty. However, even if a general motion model is used, it may not necessarily improve accuracy because of the increase in the number of motion parameters. More constraints and sophisticated filtering techniques are required to compute additional image properties; for example, one may use higher order derivatives to compute divergence, curl, and deformation [116] .

The above two approaches (spatio-temporal filtering and improved motion modeling) have achieved a certain degree of improvements over traditional algorithms. Interested readers may refer to Section 2.2 for more details about these approaches and for comparisons. Nonetheless, the interdependencies between them still set a limit to their accuracy. To answer the question posed earlier: Yes, we can improve on the current optical flow algorithms by unifying a general motion model and a spatio-temporal filtering scheme.

A general image motion model based on 3-D relative motion has been developed in[67] . However, we need to extend the *instantaneous* motion model so as to describe *continuous* motion because of the intrinsic requirement of spatio-temporal filtering. The continuous motion model is actually a 4-D model that involves X, Y, Z, t. An image motion equation based on this model is not tractable, especially considering the non-linearity imposed by perspective projection, unless we make a small motion approximation. A pointwise analysis reveals that 2-D motion is quadratic with respect to image coordinates (x, y) . It is then clear that we need a potent spatio-temporal filter design to solve the problem.

The spatio-temporal filtering scheme we use is based on 3-D Hermite polynomial differentiation filters [43] [58] , which possess several advantages: the orthogonality and Gaussian derivative properties of the filters insure numerical stability; the approach is generalizable to higher order derivatives we desire; a recursive relation to be developed below (20) can nicely facilitate the computation based on the quadratic motion equations and linearize with respect to motion parameters; these three properties make possible the coherent application of multiple filters. Furthermore, its separability promotes computing efficiency. Numerous physiological models[38,125] also support the theory that visual receptive fields can be modeled by Gaussian derivatives.

---

*. A patch centered at the focus of expansion has expansion only.

To achieve higher accuracy, we still need to overcome the third difficulty: occlusion or motion boundary effects. It is where accretion or deletion occurs[75] , and the information available to solve the optical flow problem is reduced. This difficulty is also common to other vision problems such as stereo matching. However, this issue is beyond the scope of this chapter and will be investigated in a future study.
The ultimate goal of this research is to develop a flexible set of algorithms that deals with arbitrary 3-D relative motion and computes accurate optical flow for such applications as obstacle detection or motion segmentation. Our method is not only capable of unifying the two approaches attempted by recent research but also results in algorithms whose output is adequate for many motion applications. Its competitive performance is demonstrated using the evaluation framework established by Barron, et al.[9] .

2.2 Previous Work

Recent research in the field of optical flow seems to converge on two ideas to be detailed in this section. They are spatio-temporal filtering and improved motion models.
An earlier method based on these two ideas has been proposed in [100] by Srinivasan. In this approach to an improved gradient method for optical flow, the author concentrated on generalizing spatio-temporal filtering and demonstrated his algorithms on various types of motion. However, the algorithms did not deal with motion that simultaneously contained translation, expansion, and rotation. In fact, it is stated that "erroneous results can occur if a translatory motion is superimposed upon the rotation or expansion". Nonetheless, [100] is one of the first efforts in generalizing the existing optical flow algorithms.
Later, Workhoven and Koenderink [116] introduced the idea of the *affine* flow field (2) to estimate optical flow:

$$\mathbf{V}(\mathbf{x}) = \mathbf{T} + \mathbf{A}\mathbf{x} \text{ where } \mathbf{T} = \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \text{ and } \mathbf{A} = \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix}. \tag{2}$$

A series of algorithms [10,19,23,77,111] using this more comprehensive flow field followed.
Based on an infinitesimal affine flow field, both Workhoven & Koenderink [116] and Nagel [77] used Taylor series expansion and 2-D Gaussian derivative filters to derive motion constraint equations. These equations are organized as a linear system in a similar way in both studies. Their work can be regarded as an extension of the gradient-based method originating from Horn and Schunck's work [48] . Our work is inspired by Workhoven and Koenderink's algorithm because an extensible motion constraint equation similar to (33) was developed in [116] , though only in 2-D. However, the affine model is different from our general motion model, which is based on the pointwise 3-D motion analysis. Also their approach does not offer an algorithm with competitive experimental results. In fact, our implementation of their algorithm shows that it is not reliable due to the high condition number of the linear system.
Campani and Verri [19] , Bergen et al. [10] , and Wang and Adelson [111] used local flow field coherence rather than the Taylor expansion to compute flow. Their work can

be regarded as an extension of the gradient-based method originating from Lucas and Kanade's work [68] . They do not require high-order gradients but need to perform patchwise computation. Patchwise computation is accurate when the motion has been segmented but inaccurate otherwise due to its strong susceptibility to the aperture problem.

Chou [23] modeled the error in the affine flow field as independent Gaussian noise and used Maximum a Priori (MAP) estimation to minimize the error and compute optical flow. We show later that the error modeled in [23] consists of exactly the quadratic terms. It is actually systematic and dependent on motion. Therefore, this noise model is not adequate.

Prior to the affine flow model, Hartley [42] had proposed a quadratic flow field model and used pyramid linking to estimate and segment flow simultaneously. This is the first use of the "correct" motion model in an algorithm. The integration of estimation and segmentation is a promising approach, but the lack of temporal or even spatial filtering to deal with noise is the main weakness in this work.

We realize that modeling a flow field is essentially a 2-D process, whereas modeling motion is a 3-D process, which is relatively difficult. However, we can impose temporal filtering in an integrated theoretical framework based on Hermite polynomials and turn the difficulty into an advantage.

Buxton and Buxton [18] first applied spatio-temporal filtering to the motion estimation problem. Heeger [44] , Fleet and Jepson [34] , and Weber and Malik [113] also achieved success using spatio-temporal filters. However, they used a uniform translational motion model and their improvements are limited by this assumption. Among them, Fleet and Jepson attempted to cope with non-translational motion in [34] . They showed that the phase response, instead of the amplitude response, of the velocity-tuned filters is robust to image affine transformations and photometric deformation. Their algorithm is based on constant phase contours and tends to produce more accurate but sparse flow fields.

If the above methods could take advantage of the general motion model to be developed below (16), which deals with arbitrary 3-D motion, greater improvements might be achieved. However, these methods might have difficulties with spatial nonlinearity (specifically, quadratic) and the number of parameters involved. Hermite polynomial filters, on the other hand, are capable of overcoming these difficulties.

From a theoretical point of view, the image motion corresponding to arbitrary 3-D motion has been studied by Longuet-Higgins and Prazdny[67] and Fang and Huang [33] . We have made progress beyond these efforts not only by integrating temporally continuous analysis but also by exploring robust numerical implementations. Fig 5 summarizes the thread of work leading to our algorithm. An arrow in the figure represents an idea extracted, extended or used similarly.

2.3 The General Motion Model

In this section, we describe how the local optical flow pattern reflects arbitrary 3-D motion and use this knowledge to derive a general motion model and an image motion equation. Rather than considering *instantaneous* velocity[67] , we consider velocity over

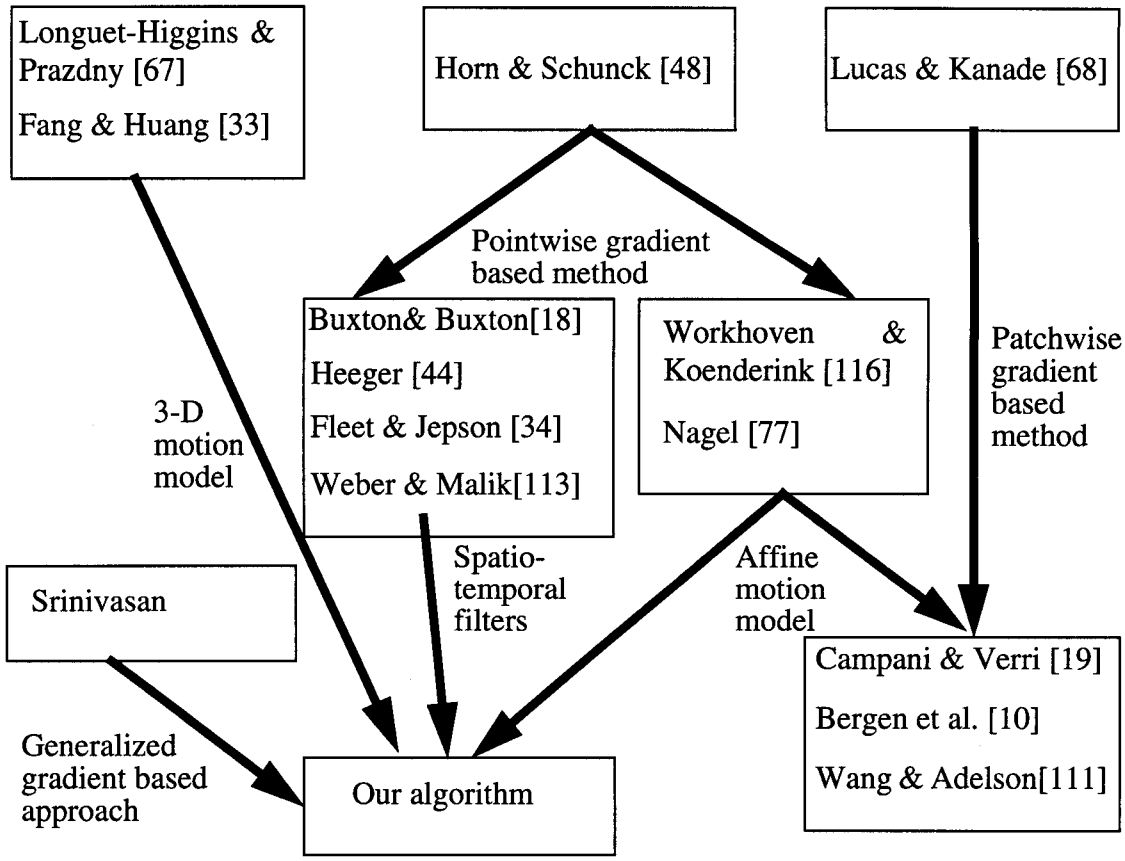


Fig 5.Thread of the optical flow research

time for the sake of spatio-temporal filtering. Let a 3-D point $\grave{P} = (X, Y, Z)$ undergo steady rotation $(\Omega_X, \Omega_Y, \Omega_Z)$ and translation (T_X, T_Y, T_Z) per unit time with respect to an observer at the origin. Previous research that deals with generalizing the motion model has used a two-frame strategy as in [33] , which may be formulated as

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T \text{, where } R = \begin{bmatrix} 1 & -\Omega_Z & \Omega_Y \\ \Omega_Z & 1 & -\Omega_X \\ -\Omega_Y & \Omega_X & 1 \end{bmatrix} \text{ and } T = \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}. \tag{3}$$

Equivalently, we write

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = M \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \text{, where } M = \begin{bmatrix} 1 & -\Omega_Z & \Omega_Y & T_X \\ \Omega_Z & 1 & -\Omega_X & T_Y \\ -\Omega_Y & \Omega_X & 1 & T_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ is the 3-D motion transformation matrix.} \tag{4}$$

The locus of a 3-D point $\vec{P}(t) = (X(t), Y(t), Z(t))^T$ can then be described by

$$\begin{bmatrix} \vec{P}(t) \\ 1 \end{bmatrix} = MM...M \begin{bmatrix} \vec{P}(0) \\ 1 \end{bmatrix} = M^t \begin{bmatrix} \vec{P}(0) \\ 1 \end{bmatrix}. \quad (5)$$

M^t means matrix M raised to the power of t and is a polynomial of matrices. If M were diagonalizable, M^t would be easily computed as $S\Lambda^t S^{-1}$ [49], where Λ is the diagonal matrix composed of the eigenvalues of M and S is the matrix of column eigenvectors. However, M has two identical eigenvalues and is not diagonalizable. Fortunately, M has a *Jordan Canonical Form* SJS^{-1} [102] from which M^t can be computed as SJ^tS^{-1}, where J^t has the analytical form [49]

$$J = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1-\Omega i & 0 \\ 0 & 0 & 0 & 1+\Omega i \end{bmatrix} \text{ where } \Omega = \sqrt{\Omega_X^2 + \Omega_Y^2 + \Omega_Z^2} \text{ and } i = \sqrt{-1}. \quad (6)$$

$$\text{Hence, } J^t = \begin{bmatrix} 1 & t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & (1-\Omega i)^t & 0 \\ 0 & 0 & 0 & (1+\Omega i)^t \end{bmatrix} \approx \begin{bmatrix} 1 & t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1-t\Omega i & 0 \\ 0 & 0 & 0 & 1+t\Omega i \end{bmatrix} \text{ when } \Omega \ll 1. \quad (7)$$

The assumption of small rotation, $\Omega \ll 1$, or equivalently, $\Omega^k \approx 0$ for all $k \geq 2$, is also used in [33] and most other later studies. In other words, we use Taylor's expansion up to order 1. Then,

$$M^t = SJ^tS^{-1} \approx \begin{bmatrix} 1 & -t\Omega_Z & t\Omega_Y & ta_X + b_X \\ t\Omega_Z & 1 & -t\Omega_X & ta_Y + b_Y \\ -t\Omega_Y & t\Omega_X & 1 & ta_Z + b_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -t\Omega_Z & t\Omega_Y & ta_X \\ t\Omega_Z & 1 & -t\Omega_X & ta_Y \\ -t\Omega_Y & t\Omega_X & 1 & ta_Z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (8)$$

where each of a_X, a_Y, a_Z is a function of all of $(\Omega_X, \Omega_Y, \Omega_Z, T_X, T_Y, T_Z)$. In this continuous time motion analysis, the center of 3-D rotation is constantly changing due to 3-D translation. Since, 3-D translation is also complicated by 3-D rotation, all the motion parameters are involved in the expressions of $a_X, a_Y, a_Z, b_X, b_Y, b_Z$. We may regard a_X, a_Y, a_Z as translations in the presence of rotation per frame time. Fortunately, since at $t = 0$, $M^0 = I, b_X, b_Y, b_Z$ vanish in (8), resulting in the last equality.

The locus $\vec{P}(t) = (X(t), Y(t), Z(t))^T$ projects to a point, $(x(t), y(t))$, in the 2-D image plane, where

$$\begin{cases} x(t) = fX(t)/Z(t) \\ y(t) = fY(t)/Z(t) \end{cases}$$, where f is the camera focal length. Let $\begin{cases} x_0 = fX/Z \\ y_0 = fY/Z \end{cases}$. Then (9)

$$\begin{aligned} x(t) &= \frac{f(ta_X + X - t\Omega_Z Y + t\Omega_Y Z)}{ta_Z - t\Omega_Y X + t\Omega_X Y + Z} = \frac{f(ta_X f/Z + x_0 - t\Omega_Z y_0 + t\Omega_Y f)}{ta_Z f/Z - t\Omega_Y x_0 + t\Omega_X y_0 + f} \\ y(t) &= \frac{f(ta_Y + t\Omega_Z X + Y - t\Omega_X Z)}{ta_Z - t\Omega_Y X + t\Omega_X Y + Z} = \frac{f(ta_Y f/Z + t\Omega_Z x_0 + y_0 - t\Omega_X f)}{ta_Z f/Z - t\Omega_Y x_0 + t\Omega_X y_0 + f} \end{aligned} . \quad (10)$$

Note that an instantaneous velocity derived in [67] is a special case of our formulation, namely, the velocity (u, v) at $t = 0$:

$$\begin{aligned} u &= \frac{\partial}{\partial t} x(t)\Big|_{t=0} = f\left(\frac{a_X}{Z} + \Omega_Y\right) - \left(f\frac{a_Z}{Z} x_0 - \Omega_Z y_0\right) + x_0\left(\frac{\Omega_Y}{f} x_0 - \frac{\Omega_X}{f} y_0\right) \\ v &= \frac{\partial}{\partial t} y(t)\Big|_{t=0} = f\left(\frac{a_Y}{Z} - \Omega_X\right) - \left(\Omega_Z x_0 - f\frac{a_Z}{Z} y_0\right) + y_0\left(\frac{\Omega_Y}{f} x_0 - \frac{\Omega_X}{f} y_0\right) \end{aligned} . \quad (11)$$

Note that the flow is generally quadratic in terms of x and y. Computing optical flow based on the uniform translation model is far from adequate, while the affine motion model is only valid when there is no rotation in the X and Y directions.

To derive an image motion equation in the form of (1), we start with the brightness constancy equation:

$$I(x(t), y(t), t) = I(x_0, y_0, 0). \quad (12)$$

Without loss of generality, let $F(x_0, y_0) = I(x_0, y_0, 0)$. It suffices to find x_0, y_0 in terms of $x(t), y(t)$, which will be denoted by x, y for simplicity. The resulting solution is extremely complicated, but assuming small rotation and small 3-D translation relative to distance, namely, $a_X, a_Y, a_Z \ll Z$, we have

$$x_0 \approx \frac{x + t\left(\frac{a_Z}{Z} x + \Omega_Z y - f\left(\frac{a_X}{Z} + \Omega_Y\right)\right)}{1 + \frac{t}{f}(\Omega_Y x - \Omega_X y)} \qquad y_0 \approx \frac{y + t\left(-\Omega_Z x + \frac{a_Z}{Z} y - f\left(\frac{a_Y}{Z} - \Omega_X\right)\right)}{1 + \frac{t}{f}(\Omega_Y x - \Omega_X y)} . \quad (13)$$

Equation (13) can be further simplified by using the approximation $\frac{t}{f}\Omega_Y x, \frac{t}{f}\Omega_X y \ll 1$, as follows:

$$x_0 \approx \left(x + t\left(\frac{a_Z}{Z}x + \Omega_Z y - f\left(\frac{a_X}{Z} + \Omega_Y\right)\right)\right)\left(1 - \frac{t}{f}(\Omega_Y x - \Omega_X y)\right)$$
$$y_0 \approx \left(y + t\left(-\Omega_Z x + \frac{a_Z}{Z}y - f\left(\frac{a_Y}{Z} - \Omega_X\right)\right)\right)\left(1 - \frac{t}{f}(\Omega_Y x - \Omega_X y)\right) \quad (14)$$

$$x_0 \approx x + t\left(\frac{a_Z}{Z}x + \Omega_Z y - f\left(\frac{a_X}{Z} + \Omega_Y\right)\right) - \frac{t}{f}(\Omega_Y x^2 - \Omega_X xy)$$
$$y_0 \approx y + t\left(-\Omega_Z x + \frac{a_Z}{Z}y - f\left(\frac{a_Y}{Z} - \Omega_X\right)\right) - \frac{t}{f}(\Omega_Y xy - \Omega_X y^2) \quad (15)$$

The above approximation is justified by the following facts. First, the value of f in pixel units is usually large. For example, for a 256x256 image with a field of view of 45 degrees, f is 309 pixels. Second, since we are concerned with motion in a relatively small image spatio-temporal neighborhood (i.e., the so-called pointwise analysis), x, y, t are small. Third, a small rotation in the X and Y direction in 3-D space can be approximated in the 2-D image plane as a simple translation. The error from this approximation will be absorbed by the translation parameters a_X, a_Y, thus offsetting the optical flow error. Inherent 3-D motion ambiguities related to this are described in [2] [121] . We will also use the above arguments for further simplification in our algorithm development (Section 2.5.2).

Now the image motion equation is, from (12) and (15),

$$I(x, y, t) = F(x + t(\alpha + \gamma x + \rho y + \delta x^2 + \varepsilon xy), y + t(\beta - \rho x + \gamma y + \delta xy + \varepsilon y^2)), \quad (16)$$

where

$$\alpha = -f\left(\frac{a_X}{Z} + \Omega_Y\right), \beta = -f\left(\frac{a_Y}{Z} - \Omega_X\right), \gamma = \frac{a_Z}{Z}, \rho = \Omega_Z, \delta = -\frac{1}{f}\Omega_Y, \varepsilon = \frac{1}{f}\Omega_X . \quad (17)$$

We are to develop a filtering scheme to relate all the above motion parameters to the 3-D filter output and then solve them in order to estimate the optical flow, which is $(-\alpha, -\beta)$ at $x = y = t = 0$ (the center of the window) from (11) and (17). Note that these motion parameters are closely related to 3-D motion.

The above two equations, (16) and (17), constitute the general motion model we will utilize throughout the paper. In summary, it is derived from the brightness constancy equation (12) and continuous-time motion analysis using the 3-D motion transformation matrix (4). Since equation (16) is true *pointwise* (under small motion assumptions), it is different from other motion models (affine or quadratic) whose analysis is based on a 3-D planar surface. For the same reason, the motion model is correct for arbitrary 3-D scenes and even non-rigid motion.

To demonstrate the behavior of the image motion equation, consider the following basic motion patterns of a parallel frontal picture:

1. When there is no rotation, and no translation in the Z direction, then $\gamma = \rho = \delta = \varepsilon = 0$ (17), and there is uniform image translational motion, as assumed by traditional algorithms (Figure 3.1 , page 3).
1. When there is no rotation, $\rho = \delta = \varepsilon = 0$, hence the image motion is affine without rotation, i.e. with only expansion and translation (Figure 3.2 , page 3).
1. When there is no translation in the Z direction, $\gamma = 0$ and no rotation in the X and Y direction, $\delta = \varepsilon = 0$, the image undergoes translation and rotation (Fig 6.1).
1. When there is no rotation in the X and Y direction, $\delta = \varepsilon = 0$, the image undergoes affine motion without deformation, i.e. only with translation, expansion and rotation (Fig 6.2).

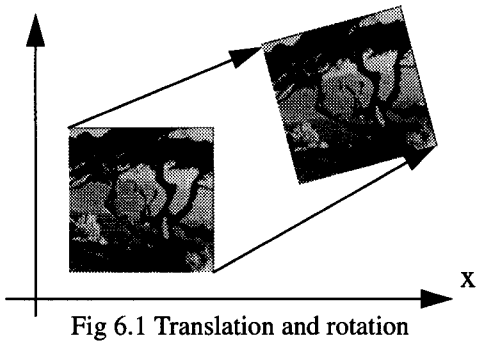


Fig 6.1 Translation and rotation

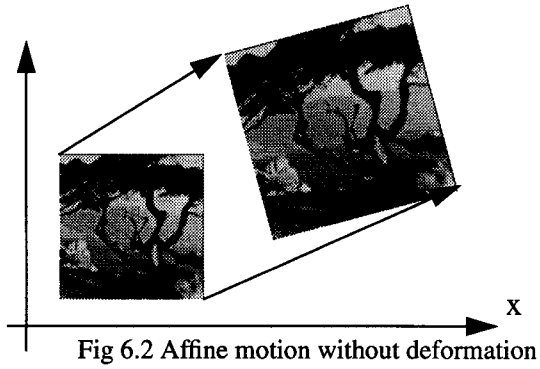


Fig 6.2 Affine motion without deformation

1. An arbitrary 3-D motion generates an image motion like Fig 7.

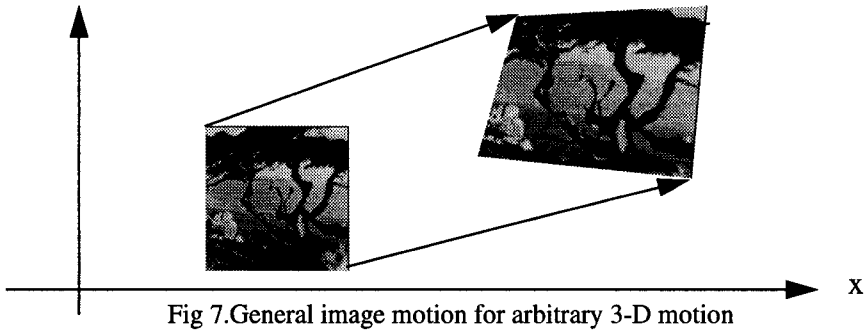


Fig 7.General image motion for arbitrary 3-D motion

Intuitively, (α, β) are related to optical flow. γ and ρ can be interpreted as expansion and rotation parameters respectively. δ and ε are related to perspective effects arising from rotation in both X and Y directions. If the imaging surface is a sphere instead of a plane, there will be no perspective effect.

In summary, the image motion equation is based on expedient and reasonable approximations. It is applicable not only to the algorithm developed here, but also to other motion algorithms, although the extent of improvement depends on the particular algorithm. For the gradient-based method, the filtering process is the decisive factor as far as performance is concerned. We formulate the theory of Hermite polynomial differentiation filters in the next section.

2.4 Hermite Polynomial Filters

2.4.1 Hermite polynomials

The nth Hermite polynomial $H_n(x)$ is a solution of

$$\frac{d^2H_n}{dx^2} - 2x\frac{dH_n}{dx} + 2nH_n = 0. \tag{18}$$

The $H_n(x)$ are derived by Rodrigues' formula [43] :

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n} e^{-x^2} . \tag{19}$$

The computation of $H_n(x)$ is especially easy using the following recursive relations:

$$\begin{aligned} H_{n+1}(x) &= 2xH_n(x) - 2nH_{n-1}(x) \\ H_0(x) &= 1 \\ H_1(x) &= 2x \end{aligned} \tag{20}$$

By substituting $G(x)$ (with variance σ^2) for e^{-x^2} in (19), we generalize to Hermite polynomials with respect to the Gaussian function. Let these Hermite polynomials be denoted by $\overline{H}_n(x)$

$$\overline{H}_n(x) = (-1)^n G^{-1}(x) \frac{d^n}{dx^n}(G(x)) \tag{21}$$

Note that $\overline{H}_n(x)$ differs from $H_n(x)$ by a scaling product:

$$\overline{H}_n(x) = \left(\frac{1}{2^{1/2}\sigma}\right)^n H_n\left(\frac{x}{2^{1/2}\sigma}\right). \tag{22}$$

The scalar product of two functions and the L_2-norm of a function with $G(x)$ as a weight function are defined as follows:

$$\langle a, b\rangle \equiv \int_{-\infty}^{\infty} G(x)a(x)b(x)dx \quad \text{and} \quad \|a\| \equiv \langle a, a\rangle^{1/2}.$$

The orthogonality of $\{\ \overline{H}_n(x)\ \}$ can be expressed in the following way[43] :

$$\langle \overline{H}_m, \overline{H}_n\rangle = \sigma^{-2n} n! \delta_{mn}. \tag{23}$$

The 3D case of Hermite polynomials is especially simple because they are separable.

Thus the polynomial of order $n = i + j + k$ is

$$\overline{H}_{ijk}(x, y, t) = \overline{H}_i(x) \cdot \overline{H}_j(y) \cdot \overline{H}_k(t) \ . \tag{24}$$

2.4.2 Derivation of gradient constraint equations

One of the most important properties of Hermite polynomials is the property of Gaussian derivatives. It is with the aid of this property that we are able to establish gradient constraint equations. This property is manifested in the following theorem.

Theorem 1: A one dimensional signal $I(x)$ can be expanded in terms of Hermite polynomials as

$$I(x) = \sum_{k=0}^{\infty} I_k \frac{\overline{H}_k(x)}{\|\overline{H}_k\|^2} \ .$$

$$\text{Then } I_k = \langle I, \overline{H}_k \rangle = \langle I^{(k)}, \overline{H}_0 \rangle \text{ where } \overline{H}_0(x) = 1 \text{ and } I^{(k)} = \frac{d^k I}{dx^k}. \tag{25}$$

The proof is given in Appendix A. This theorem states that the kth order Gaussian derivative of the image is the inner product of the image and the kth order Hermite polynomial $\overline{H}_k(x)$. Note that the theorem is true only for unnormalized Hermite polynomials. This fact is used when we assign weights to motion constraint equations of different orders in equation (37).
Recall our image motion equation (16),

$$I(x, y, t) = F(x + t(\alpha + \gamma x + \rho y + \delta x^2 + \varepsilon xy), y + t(\beta - \rho x + \gamma y + \delta xy + \varepsilon y^2)) \ .$$

Expand both sides with Hermite polynomials,

$$\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\sum_{k=0}^{\infty} I_{ijk} \frac{\overline{H}_{ijk}}{\|\overline{H}_{ijk}\|^2} = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\sum_{k=0}^{\infty} F_{ijk} \frac{\overline{H}_{ijk}}{\|\overline{H}_{ijk}\|^2} \text{ then}$$

$$I_{ijk} = \langle I, \overline{H}_{ijk} \rangle = F_{ijk} = \langle F, \overline{H}_{ijk} \rangle \tag{26}$$

Equating I_{ij1} to F_{ij1} and using Theorem 1, we derive

$$\begin{aligned} I_{ij1} = F_{ij1} &= \langle F, \overline{H}_{ij1} \rangle = \langle \frac{\partial F}{\partial t}, \overline{H}_{ij0} \rangle \\ &= \langle (\alpha + \gamma x + \rho y + \delta x^2 + \varepsilon xy) \frac{\partial F}{\partial x_0} + (\beta - \rho x + \gamma y + \delta xy + \varepsilon y^2) \frac{\partial F}{\partial y_0} \ , \ \overline{H}_{ij0} \rangle \\ &\approx \langle (\alpha + \gamma x + \rho y + \delta x^2 + \varepsilon xy) \frac{\partial I}{\partial x}, \overline{H}_{ij0} \rangle + \langle (\beta - \rho x + \gamma y + \delta xy + \varepsilon y^2) \frac{\partial I}{\partial y} \ , \ \overline{H}_{ij0} \rangle \end{aligned} \tag{27}$$

We make the above approximation because equations (10) and (17) allow us to derive

$$\frac{\partial F}{\partial x_0} = \frac{\partial I}{\partial x}\frac{\partial x}{\partial x_0} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial x_0} = \frac{\partial I}{\partial x}(1 - \delta fxt) + \frac{\partial I}{\partial y}(\rho t - \delta fyt). \tag{28}$$

Since $\left.\frac{\partial F}{\partial x_0}\right|_{x=0,\,y=0,\,t=0} = \left.\frac{\partial I}{\partial x}\right|_{x=0,\,y=0,\,t=0}$ and the inner product is Gaussian weighted, the error is not significant. Without the approximation, the eventual constraint equation would be nonlinear and very difficult to solve. The analysis is similar for $\frac{\partial F}{\partial y_0}$. Therefore, we arrive at

$$I_{ij1} \approx \langle(\alpha + \gamma x + \rho y + \delta x^2 + \varepsilon xy)\frac{\partial I}{\partial x}, \overline{H}_{ij0}\rangle + ((\beta - \rho x + \gamma y + \delta xy + \varepsilon y^2)\frac{\partial I}{\partial y}, \overline{H}_{ij0}). \tag{29}$$

$$\begin{aligned} I_{ij1} \approx\ & \alpha\langle\frac{\partial I}{\partial x}, \overline{H}_{ij0}\rangle + \gamma\langle\frac{\partial I}{\partial x}, x\overline{H}_{ij0}\rangle + \rho\langle\frac{\partial I}{\partial x}, y\overline{H}_{ij0}\rangle + \delta\langle\frac{\partial I}{\partial x}, x^2\overline{H}_{ij0}\rangle + \varepsilon\langle\frac{\partial I}{\partial x}, xy\overline{H}_{ij0}\rangle \\ & + \beta\langle\frac{\partial I}{\partial y}, \overline{H}_{ij0}\rangle - \rho\langle\frac{\partial I}{\partial y}, x\overline{H}_{ij0}\rangle + \gamma\langle\frac{\partial I}{\partial y}, y\overline{H}_{ij0}\rangle + \delta\langle\frac{\partial I}{\partial y}, xy\overline{H}_{ij0}\rangle + \varepsilon\langle\frac{\partial I}{\partial y}, y^2\overline{H}_{ij0}\rangle \end{aligned} \tag{30}$$

To simplify (30), we derive from (20) and (22) the following equation

$$x\overline{H}_n(x) = \sigma^2\overline{H}_{n+1}(x) + n\overline{H}_{n-1}(x) \text{ . Hence} \tag{31}$$

$$\begin{aligned} I_{ij1} \approx\ & \alpha\langle\frac{\partial I}{\partial x}, \overline{H}_{ij0}\rangle + \gamma\langle\frac{\partial I}{\partial x}, \sigma^2\overline{H}_{i+1\,j0} + i\overline{H}_{i-1\,j0}\rangle + \rho\langle\frac{\partial I}{\partial x}, \sigma^2\overline{H}_{ij+1,\,0} + j\overline{H}_{ij-1,0}\rangle \\ & + \delta\langle\frac{\partial I}{\partial x}, \sigma^4\overline{H}_{i+2\,j0} + (2i+1)\sigma^2\overline{H}_{ij0} + i(i+1)\overline{H}_{i-2\,j0}\rangle \\ & + \varepsilon\langle\frac{\partial I}{\partial x}, \sigma^4\overline{H}_{i+1\,j+1,\,0} + i\sigma^2\overline{H}_{i-1\,j+1,\,0} + j\sigma^2\overline{H}_{i+1\,j-1,\,0} + ij\overline{H}_{i-1\,j-1,\,0}\rangle \\ & + \beta\langle\frac{\partial I}{\partial y}, \overline{H}_{ij0}\rangle - \rho\langle\frac{\partial I}{\partial y}, \sigma^2\overline{H}_{i+1\,j0} + i\overline{H}_{i-1\,j0}\rangle + \gamma\langle\frac{\partial I}{\partial y}, \sigma^2\overline{H}_{ij+1,\,0} + j\overline{H}_{ij-1,\,0}\rangle \\ & + \delta\langle\frac{\partial I}{\partial y}, \sigma^4\overline{H}_{i+1\,j+1,\,0} + i\sigma^2\overline{H}_{i-1\,j+1,\,0} + j\sigma^2\overline{H}_{i+1\,j-1,\,0} + ij\overline{H}_{i-1\,j-1,\,0}\rangle \\ & + \varepsilon\langle\frac{\partial I}{\partial y}, \sigma^4\overline{H}_{ij+2,\,0} + (2j+1)\sigma^2\overline{H}_{ij0} + j(j+1)\overline{H}_{ij-2,\,0}\rangle \end{aligned} \tag{32}$$

Using Theorem 1, we simplify (32) to

$$
\begin{aligned}
I_{ij1} \approx\ & \alpha I_{i+1\,j0} + \gamma[\sigma^2 I_{i+2\,j0} + iI_{ij0}] \\
& + \rho[\sigma^2 I_{i+1\,j+1,\,0} + jI_{i+1\,j-1,\,0}] \\
& + \delta[\sigma^4 I_{i+3\,j0} + (2i+1)\sigma^2 I_{i+1\,j0} + i(i+1)I_{i-1\,j0}] \\
& + \varepsilon[\sigma^4 I_{i+2\,j+1,\,0} + i\sigma^2 I_{ij+1,0} + j\sigma^2 I_{i+2\,j-1,\,0} + ijI_{ij-1,0}] \\
& + \beta I_{ij+1,\,0} - \rho[\sigma^2 I_{i+1\,j+1,\,0} + iI_{i-1\,j+1,\,0}] + \gamma[\sigma^2 I_{ij+2,\,0} + jI_{ij0}] \\
& + \delta[\sigma^4 I_{i+1\,j+2,\,0} + i\sigma^2 I_{i-1\,j+2,\,0} + j\sigma^2 I_{i+1\,j0} + ijI_{i-1\,j0}] \\
& + \varepsilon[\sigma^4 I_{ij+3,\,0} + (2j+1)\sigma^2 I_{ij+1,\,0} + j(j+1)I_{ij-1,\,0}]
\end{aligned}
\tag{33}
$$

For generality, in equation (33), we can substitute ijk for $ij1$ and $ij(k-1)$ for $ij0$. All other terms will remain the same. In deriving equation (33), we assume that the motion parameters $\alpha, \beta, \gamma, \rho, \delta, \varepsilon$ are constant within the filter support. This is true under the assumption that the underlying surface is parallel frontal so that Z is constant (17).

As stated in the introduction, the general motion model is a fundamental element resulting in a coherent spatio-temporal filtering scheme to compute optical flow. This capability stems from two nice properties of (33). The first is the linearity of the equation in terms of the motion parameters as defined in (17); the second is its extensibility to higher orders, i.e., the values of i, j can be as large as required by the number of parameters. Thus to solve for the motion parameters and then for optical flow, we derive a system of linear equations with coefficients computed from spatio-temporal filters I_{ijk}. These result in excellent numerical stability due to the orthogonality of the Hermite polynomial differentiation filters and their inherent Gaussian smoothing. Although (33) appears to be complicated, it in fact suggests a simple, local, and parallel algorithm, which involves only convolutions and solving a linear system, as presented in the next section.

2.5 Algorithms for Computing Optical Flow

Equation (33) gives rise not to a specific algorithm but to a set of algorithms, due to its extensibility. We can derive the same number of equations as unknowns and solve a linear system, or we may incorporate more equations of higher order and solve a linear least square problem. On the other hand, if we possess knowledge about the input image sequence, for example, that there are no rotations around certain directions, the number of unknowns can be reduced. We also make other expedient choices based on numerical considerations and experimental findings. All these options are explored in the following subsections.

2.5.1 The general framework

According to (33), we can derive six equations up to the third order. Within a 3-D local

window, we estimate $\{ I_{ijk} \}$ with the discrete approximation $\{ \hat{I}_{ijk}(x, y, t) \}$, that is, the 3-D convolution of the *sampled* Hermite polynomial filters with the image sequence, and write the equations in matrix vector form:

$M_6 s = c$, where $M_6 = (M_4 \oplus M_2)$ where $\oplus$ means concatenation, and (34)

$$s = \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \rho \\ \delta \\ \varepsilon \end{bmatrix}, c = \begin{bmatrix} \hat{I}_{001} \\ \hat{I}_{101} \\ \hat{I}_{011} \\ \hat{I}_{201} \\ \hat{I}_{111} \\ \hat{I}_{021} \end{bmatrix}, M_4 = \begin{bmatrix} \hat{I}_{100} & \hat{I}_{010} & \sigma^2(\hat{I}_{200} + \hat{I}_{020}) & 0 \\ \hat{I}_{200} & \hat{I}_{110} & \sigma^2(\hat{I}_{300} + \hat{I}_{120}) + \hat{I}_{100} & -\hat{I}_{010} \\ \hat{I}_{110} & \hat{I}_{020} & \sigma^2(\hat{I}_{210} + \hat{I}_{030}) + \hat{I}_{010} & \hat{I}_{100} \\ \hat{I}_{300} & \hat{I}_{210} & \sigma^2(\hat{I}_{400} + \hat{I}_{220}) + 2\hat{I}_{200} & -2\hat{I}_{110} \\ \hat{I}_{210} & \hat{I}_{120} & \sigma^2(\hat{I}_{310} + \hat{I}_{130}) + 2\hat{I}_{110} & \hat{I}_{200} - \hat{I}_{020} \\ \hat{I}_{120} & \hat{I}_{030} & \sigma^2(\hat{I}_{220} + \hat{I}_{040}) + 2\hat{I}_{020} & 2\hat{I}_{110} \end{bmatrix}, \quad (35)$$

$$M_{\bar{I}} \begin{bmatrix} \sigma^4(\hat{I}_{300} + \hat{I}_{120}) + \sigma^2\hat{I}_{100} & \sigma^4(\hat{I}_{210} + \hat{I}_{030}) + \sigma^2\hat{I}_{010} \\ \sigma^4(\hat{I}_{400} + \hat{I}_{220}) + \sigma^2(3\hat{I}_{200} + \hat{I}_{020}) + 2\hat{I}_{000} & \sigma^4(\hat{I}_{310} + \hat{I}_{130}) + \sigma^2 2\hat{I}_{110} \\ \sigma^4(\hat{I}_{310} + \hat{I}_{130}) + \sigma^2 2\hat{I}_{110} & \sigma^4(\hat{I}_{220} + \hat{I}_{040}) + \sigma^2(3\hat{I}_{020} + \hat{I}_{200}) + 2\hat{I}_{000} \\ \sigma^4(\hat{I}_{500} + \hat{I}_{320}) + \sigma^2(5\hat{I}_{300} + 2\hat{I}_{120}) + 6\hat{I}_{100} & \sigma^4(\hat{I}_{410} + \hat{I}_{230}) + \sigma^2 3\hat{I}_{210} \\ \sigma^4(\hat{I}_{410} + \hat{I}_{230}) + \sigma^2(4\hat{I}_{210} + \hat{I}_{030}) + 3\hat{I}_{010} & \sigma^4(\hat{I}_{320} + \hat{I}_{140}) + \sigma^2(4\hat{I}_{120} + \hat{I}_{300}) + 3\hat{I}_{100} \\ \sigma^4(\hat{I}_{320} + \hat{I}_{050}) + \sigma^2 3\hat{I}_{120} & \sigma^4(\hat{I}_{230} + \hat{I}_{050}) + \sigma^2(5\hat{I}_{030} + 2\hat{I}_{210}) + 6\hat{I}_{010} \end{bmatrix} \quad (36)$$

Note that filter outputs of up to fifth order are used. s can be solved exactly by (34) for the center pixel of the 3-D convolution window and optical flow at the center of this window is $(-\alpha, -\beta)$. Note that we can also derive more equations using even higher order derivatives and obtain a linear least square estimate of the parameters and optical flow.

2.5.2 Specialized algorithms

The algorithm presented in the previous subsection, i.e., solving a full-rank linear system, is often not necessary for many image sequences. There are several disadvantages in using it for simple motion: First, it requires the use of higher order filters; their orthogonality is always distorted with limited filter support while the use of a large filter results in greater susceptibility to motion boundary effects. Second, the linear system is often highly ill-conditioned, since the columns of the matrix are of different orders of magnitude. Third, it is computationally expensive.

Let the focal length be large enough and/or the rotation in the X, Y direction small enough for δ and ε to be small (17), i.e. the perspective effect is negligible. Then $M_2 = 0$ and we have a linear least square problem:

$$E = min\|A_4 s_4 - b\| \text{ where } A_4 = WM_4,\ b = Wc, \text{ and}$$
$$W = Diag[w_1, w_2, w_3, w_4, w_5, w_6]\ . \tag{37}$$

W is the diagonal weight matrix for the motion constraint equations. According to (26), we use

$$w_1 = \|\bar{H}_{001}\|^{-2}, w_2 = \|\bar{H}_{101}\|^{-2}, w_3 = \|\bar{H}_{011}\|^{-2}, w_4 = \|\bar{H}_{201}\|^{-2}, w_5 = \|\bar{H}_{111}\|^{-2}, w_6 = \|\bar{H}_{021}\|^{-2}$$

Equation (37) models affine motion without deformation, as depicted in Fig 6.2. Note that this formulation reduces the highest order filter to fourth order. In addition, a least square solution is more stable than a full-rank linear system. The applicability of the weight matrix is another nice feature. We suggest solving (37) by QR decomposition:

$$A_4 = QR, \text{ and } E = min\|QRs + b\| = min\|Rs + Q^H b\|, \text{ where } Q \text{ is unitary.} \tag{38}$$

R can be denoted by $\begin{bmatrix} R_s \\ 0 \end{bmatrix}$, where R_s is an upper triangular matrix; and $Q^H b$ is $\begin{bmatrix} b_s \\ b_r \end{bmatrix}$, correspondingly. Equation (38) then becomes

$$E = min(\|R_s s + b_s\| + \|b_r\|)$$
$$= \|b_r\| = r \qquad \text{if } R_s \text{ is not singular.} \tag{39}$$

$$\text{The solution } s \text{ is computed from } R_s s + b_s = 0 \tag{40}$$

For many practical applications, the above algorithm is adequate for computing optical flow. Nonetheless, one can still simplify the algorithms and improve the stability of the linear system. For example, in many synthetic image or synthesized real image sequences [9] where there is no rotation along the optical axis, i.e., $\Omega_Z = 0$, we get $\rho = 0$ and (37) reduces to

$$E = min\|A_3 s_3 - b\|, \tag{41}$$

where A_3 and s_3 are the first three columns and elements of A_4 and s_4, respectively.

Furthermore, the third column in A_3 involves higher order terms plus a lower order term. Experiments suggest that the lower order term is always dominant and more accurate. Neglecting the higher order terms does not necessarily degrade the accuracy but does save a great deal of computing time. This finding is very crucial especially for real-time implementations.

As far as the implementation is concerned, our algorithm is efficient due to the separability of the 3-D Hermite polynomial filters. Let the 3-D filter size be $W_x \times W_y \times W_t$ and image size be S. The complexity of the computation is $O((W_x + W_y + W_t + C)S)$, instead of $O((W_x W_y W_t + C)S)$, where C is the constant factor associated with solving the linear system.

One of the advantages of using the QR decomposition is the availability of the matrix R_s and the residual. The behavior of R_s and the residual value reveal significant information about the underlying images and motions. There are certain situations where the optical flow cannot be reliably computed from local information due to, for example, the aperture problem. Therefore, R_s and the residual can be investigated for their possible link to the reliability of the computed flow. We devote the next subsection to the discussion of the optic flow errors and confidence measures.

2.5.3 Confidence measures

Our algorithm provides certain information about the behavior of the system equations. This information includes the residual r, the condition number and the determinant of R_s. They can be shown [62] to signify certain image phenomena, e.g., occlusion, the aperture problem, etc., which present difficulties for optical flow computation. Therefore, they can be utilized to locate high error areas and suggest subsequent improvement methods. For the sake of evaluation in Chapter 3, we simply use them as confidence measures or threshold values to extract more accurate data; however, they can also be used for qualitative image analysis such as motion boundary extraction [64] .

2.5.3.1 Residual

The residual of our algorithm is $\|A_n s_n + b\|$ or r (=E) (39). The residual of an overdetermined linear system indicates the degree to which the equations disagree with one another. The reason for the existence of the residual lies in the approximation error of $\{ \hat{I}_{ijk}(x, y, t) \}$. A high approximation error may indicate one of three problems:

1. The assumption of the motion model is violated in the 3-D window: It is possible that the window covers more than one moving object. Occlusion or multiple independently moving objects in a window can cause this problem.
2. The assumption of constant image brightness is violated: It is not unusual for the brightness of an object to change when the viewing angle changes due to relative motion. In addition, the observing camera may adjust the brightness gain as the content of the scene changes, resulting in a change of object brightness. Similar effects can be caused by the shadow of another object.
3. Quantization and truncation errors. Quantization errors result from sampling Hermite polynomial filters: Truncation errors are introduced when we use limited spatial support to compute $\{ \hat{I}_{ijk}(x, y, t) \}$. Within a small window, the Hermite polynomials are no longer orthogonal and the derivatives computed are not accurate. This situation is worse for higher order differentiation filters. For example, $\langle \overline{H}_n, \overline{H}_n \rangle / \sigma^{-2n} n!$ (23) is 0.93 when $n = 5$ and 0.999945 when $n = 1$ for a window size of 21.

We can model the above errors as perturbations to the linear system [62] :

$$\tilde{E} = min\|(A_n + N)\tilde{s} + (b + \Delta b)\| \text{ , where } N \text{ and } \Delta b \text{ denote errors and } n < 6. \quad (42)$$

We prove in Appendix B the following analytical results:

$$\Delta s = \tilde{s} - s \approx -(A_n{}^T A_n)^{-1} A_n^T (Ns + \Delta b) \quad (43)$$

$$\tilde{r} = \tilde{E} \approx \left\|(I - A_n(A_n{}^T A_n)^{-1} A_n{}^T)(Ns + \Delta b)\right\| \quad (44)$$

Note that the expressions for both optical flow error Δs (43) and residual $\tilde{r}$ (44) are proportional to the magnitude of the noise vector $(Ns + \Delta b)$. It is evident that locations with a high residual value reflect large errors and inaccurate optical flow values.
Note that the three problems mentioned above suggest contradictory choices for the window size. With larger windows, problems 1 and 2 becomes worse; with smaller windows, problem 3 becomes worse.

2.5.3.2 Condition Number and Determinant

The condition number of R_s, denoted by $\kappa(R_s)$, is defined as $\|R_s\|\|R_s^{-1}\|$ and can be shown to be $\frac{|\lambda|_{max}}{|\lambda|_{min}}$, where the λ's are eigenvalues or diagonal elements of R_s.
The condition number measures the extent to which a linear system maps an input error into an output error, i.e., the numerical instability of the system. If s contains errors magnified by an ill-conditioned A_n from errors in b, it is not reliable. Since matrix A_n is concerned with the image texture only and not with motion, we find that a high condition number results from the following two image neighborhood situations:

1. When there is a steep edge in the $x(y)$ direction (Fig 8.1), the derivatives are very large for $x(y)$ and small for $y(x)$;
2. When there is a lack of texture in a direction (the $x + y$ direction in Fig 8.2), the derivatives in the x direction are approximately proportional to the derivatives in the y direction, i.e. $I(x, y) \approx I(kx + y)$.

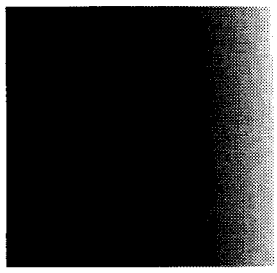


Fig 8.1 Smoothed steep edge

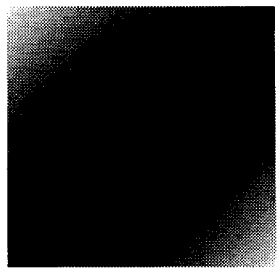


Fig 8.2 Lack of texture in $x+y$ direction

The above two situations can easily be confirmed by inspecting the QR decomposition

process. If there is motion in the area where one of these two situations dominates, then it corresponds to what is known as the aperture problem.
The determinant of R_s is the product of all its eigenvalues. In solving (40) or $s = -R_s^{-1} \cdot b_s$, the determinant plays an important role in the matrix inverse. Since we use the QR decomposition method, Q is unitary (orthonormal projection) so the behavior of R_s is similar to the original A_n. A small determinant of R_s indicates one of the following two situations:

1. The three columns of A_n are close to being linearly dependent. This is the same as the second situation in the above discussion related to the condition number. In fact, a small determinant due to linear dependency also causes a high condition number.
2. All the elements of A_n are very small. This corresponds to a uniform brightness area, e.g. a cloudless sky.

As noted before, the above situations correspond to the general case of the aperture problem. It is interesting to note that Barron, Fleet, and Beauchemin [9] recognize empirically that the determinant is a better confidence measure in the application of the Uras, et al. [109] optical flow algorithm than the condition number used in the original paper. Our analysis agrees with their empirical observations.

2.5.3.3 Integration of confidence measures

Based on the above analysis, we choose a combination of confidence measures according to the nature of a given image sequence.
If the image sequence contains numerous moving objects or the brightness changes significantly, residuals should be used as the confidence measure since they signify the three problems listed in Section 2.5.3.1. No other confidence measure that we know of is effective for these cases.
The condition number and determinant have something in common although they may signify different situations. Together they signify the relationship between numerical instability and the aperture problem. Empirical findings suggest that they be used as a multiplicative combination, or similarly, in the form of $|\lambda|_{min}$ [62] . This has been proposed by Girosi et al.[37] in a similar context and was used in Barron's implementation [9] of Lucas and Kanade's optical flow algorithm. In our algorithm, $|\lambda|_{min}$ simply indicates the whether or not the necessary texture exists. We use it to signify the aperture problem and to avoid numerical instability.
All the above mentioned problems are not unique to our algorithm; they are common to other optical flow algorithms as well.

2.5.3.4 Distinguishing normal flow from full flow

When the aperture problem arises in an image local neighborhood, correct full flow cannot be recovered. In our algorithm, however, the condition number and smallest eigenvalue can assist us in distinguishing different levels of the aperture problem and

extracting accurate normal flow and full flow. In summary, when the condition number $\kappa(R_s)$ is moderately low and the smallest eigenvalue $|\lambda|_{min}$ is reasonably high, which means the image texture is sufficient, then correct full flow can be recovered; when the condition number is very high and the smallest eigenvalue is low, which means the image texture is high in one direction and very low in the perpendicular direction, normal flow can be recovered; when both the condition number and the smallest eigenvalue are very low, which means there is insufficient texture in the local image neighborhood, then no flow can be recovered reliably.

The following example illustrates the concept. In Fig 9.1, the white square is moving to the upper right. Our algorithm successfully segments three areas corresponding to the three situations above: First, at the corners of the square, where these is no aperture problem, the correct optical flow is extracted by thresholding on the smallest eigenvalue, in this example, $|\lambda|_{min} > 1.5$ (Fig 9.2). Second, at the edges of the square, where the one dimensional aperture problem exists, correct normal flow can be extracted by thresholding on the smallest eigenvalue and on the condition number, in this example, $|\lambda|_{min} < 1.5 \wedge \kappa(R_s) > 100$ (Fig 9.3). Third, at all other areas where the two dimensional aperture problem exists, no correct flow can be extracted.

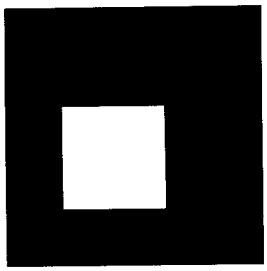


Fig 9.1 Translating square

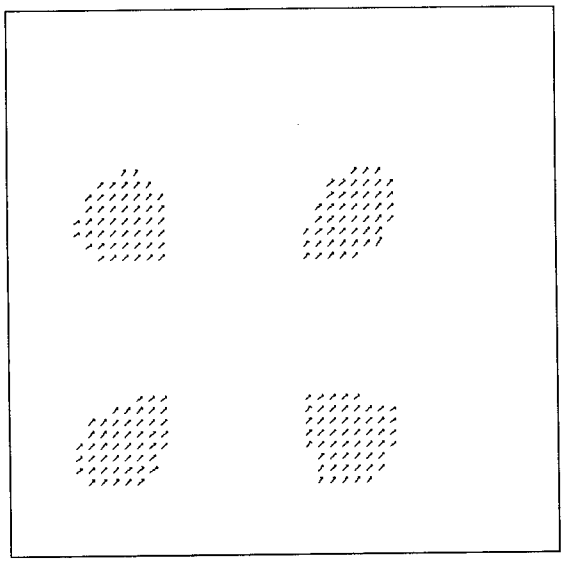


Fig 9.2 Full flow only

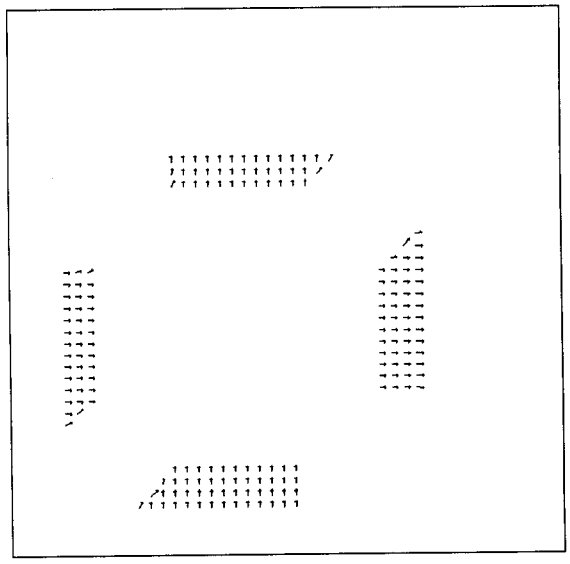


Fig 9.3 Normal flow where there is aperture problem

Chapter 3
Evaluating Optical Flow Algorithms

3.1 Quantitative Evaluation Using Synthetic Images

Following the work of Barron, Fleet, and Beauchemin[9] , we conducted extensive comparisons between our algorithm and existing optical flow algorithms, including those by Horn and Schunck[48] , Lucas and Kanade[68] , Uras et al. [109] , Nagel[78] , Anandan[4] , Singh[97] , Heeger[44] , Waxman et al. [112] , Fleet and Jepson[34] , Bober and Kittler[11] , Weber and Malik[113] . The synthetic image sequences we use for comparison are Sinusoid, Translating tree, Diverging tree, Yosemite fly-by (provided by Barron), and Moon landing.
When the image sequences contain only translational (Sinusoid) and diverging motion (Translating tree, Diverging tree, and Yosemite fly-by), we use the algorithm in (41); when the image motion also contains rotation (Moon landing), we use the algorithm in (37). A simple 3x3 median filter is applied on the flow field to reduce output noise although it may not necessarily improve overall accuracy.
The error statistic utilized is the angle error between the computed optical flow time-space direction $(u_e, v_e, 1)$ and the ground truth flow time-space direction $(u_c, v_c, 1)$ averaged over the whole image. Refer to [9] for more details. In order to make extensive comparisons, we implemented our algorithm in such a way that a certain density of output flow can be extracted by thresholding on a chosen confidence measure. Error statistics in the following subsections are displayed in tables. For a single technique with multiple entities in these tables, different threshold values are used in the algorithm to produce multiple densities of output. The error statistics and associated density for the comparison algorithms were obtained directly from [9] or from the original publications.

3.1.1 Sinusoid

This is a synthetic image sequence (Fig 10) of a spatial sinusoidal wave traversing toward the upper right. For our method we chose a window size large enough (17x17x7 for x,y,t) to prevent aliasing. $|1/r|$ was used as the confidence measure. Fig 11.1 shows the true optical flow, while Fig 11.2 shows the flow computed with our method. Our algorithm performs better than all of the other algorithms except Fleet and Jepson's (Table 1).

3.1.2 Translating tree and Diverging tree

The translating and diverging tree sequences are two realistic synthetic sequences simu-

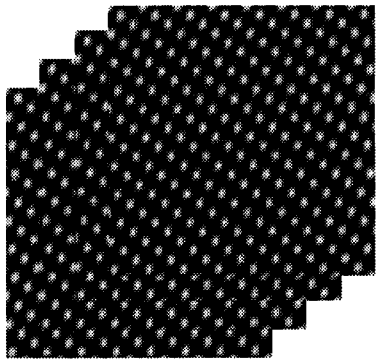


Fig 10.Traversing sinusoid

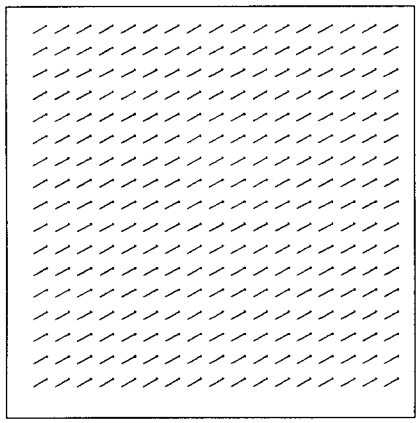


Fig 11.1True optical flow for sinusoid

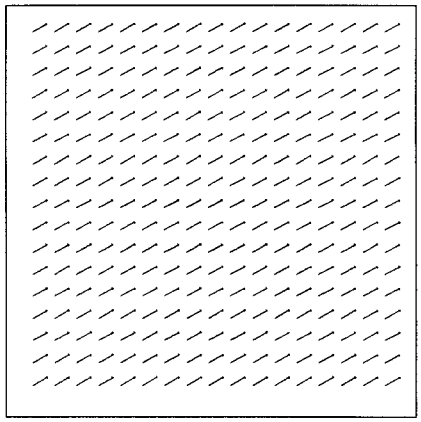


Fig 11.2 Computed optical flow (100% density)

Table 1: Summary of Sinusoid error statistics

Density	Our Algorithm		Other Algorithms		
	Average Error	Standard Deviation	Average Error	Standard Deviation	Technique by
100%	0.63°	0.06°	4.19°	0.50°	Horn & Schunck (original unthresholded)
			2.55°	0.59°	Horn & Schunck (modified unthresholded)
			2.47°	0.16°	Lucas and Kanade (unthresholded)
			2.59°	0.71°	Uras et al. (unthresholded)
			2.55°	0.93°	Nagel
			30.80°	5.45°	Anandan
			2.24°	0.02°	Singh (step 1 unthresholded)
			0.03°	0.01°	Fleet and Jepson
12.8%	0.63°	0.06°	64.26°	26.14°	Waxman et al.

lating the motion of simple translation (Fig 12.1) and expansion (Fig 12.2), respectively, of a poster. The window size used in our method is 19x19x11* for the translating tree and 17x17x11† for the diverging tree. Due to the lack of texture in some background

*. from frame 4 to frame 14.

areas, we used $|\lambda|_{min}$ as the confidence measure. Fig 13 and Fig 14 show the results. For the translating tree sequence, our results are among the best as shown in Table 2. The comparison is also visualized in Fig 15 where we plot all the algorithms' error versus density. In the plot, our algorithm generates a curve while other algorithms generate points. For the diverging tree sequence, our results are second only to Fleet and Jepson's result as shown in Table 3. The comparison is also visualized in Fig 16. Note that the poster is not parallel frontal. This results in errors, especially for the translating tree sequence.

Fig 12.1 Translating tree

Fig 12.2 Diverging tree

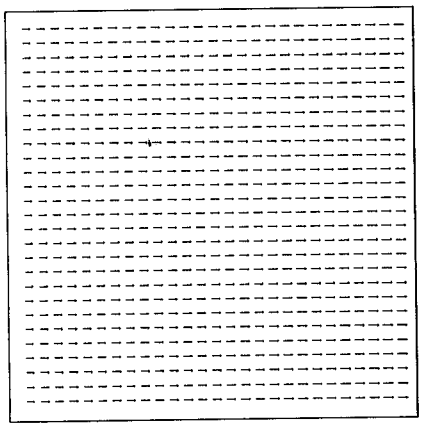


Fig 13.1 True flow for Translating tree

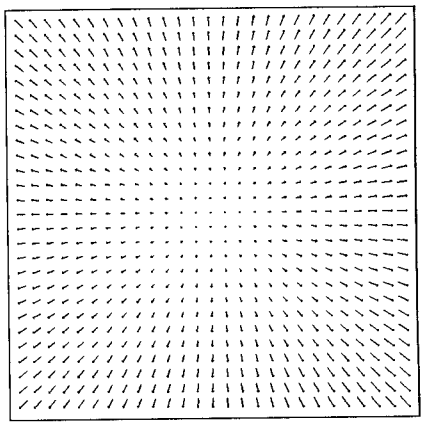


Fig 13.2 True flow for Diverging tree

3.1.3 Yosemite fly-by

The Yosemite fly-by sequence is a realistic synthetic image sequence (Fig 17). The flight scene is simulated using actual aerial photos and digital terrain maps, with artificial sky and clouds. Since the clouds in the sky change brightness over time, it poses difficulties

†. from frame 5 to frame 15.

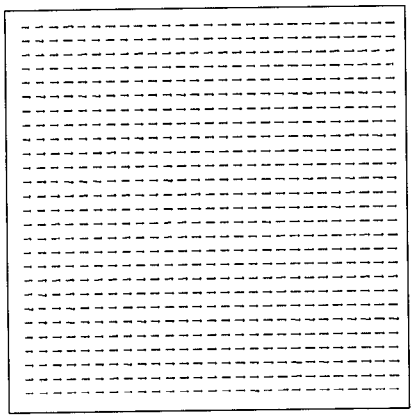


Fig 14.1 Computed flow for Translating tree

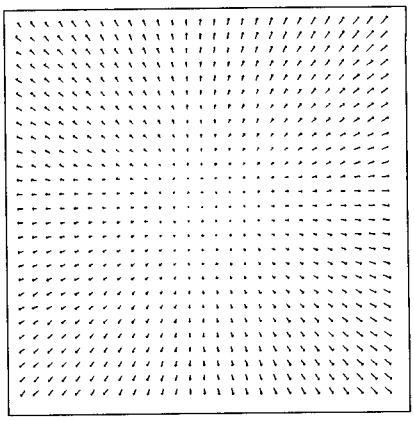


Fig 14.2 Computed flow for Diverging tree

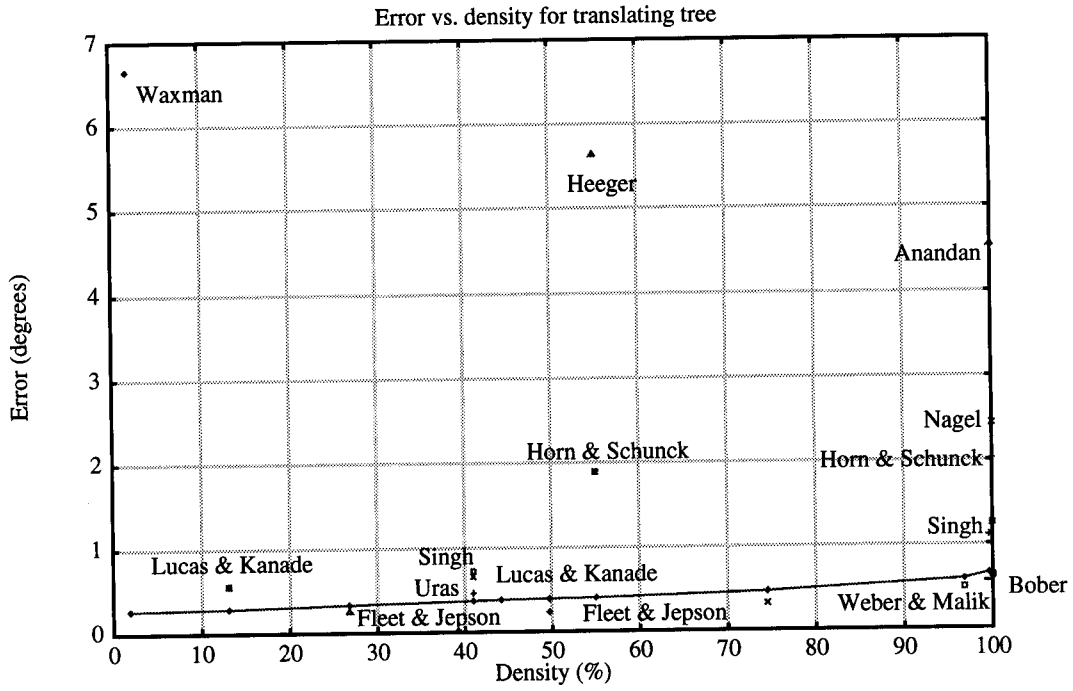


Fig 15. Comparison plot for Translating tree sequence

for all algorithms. Based on our previous analysis, we used $|1/r|$ as the confidence measure to eliminate points that lie in a large blank area in the sky and on motion boundaries. Fig 18.2 shows the results. Since the motion is rather fast in some areas, we used a larger window (21x21x7, from frame 6 to frame 12). Error statistics are shown in Table 4 and visualized in Fig 19. Again, the clouds account for the large magnitude error. Our algorithm performs better than all others.

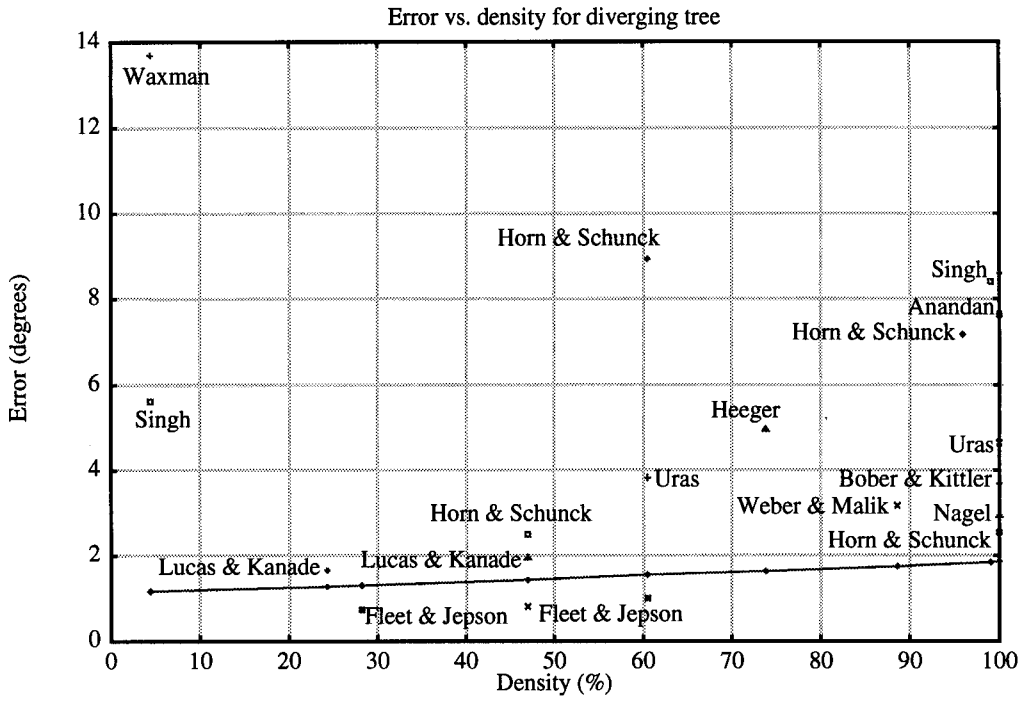


Fig 16. Comparison plot for Diverging tree sequence

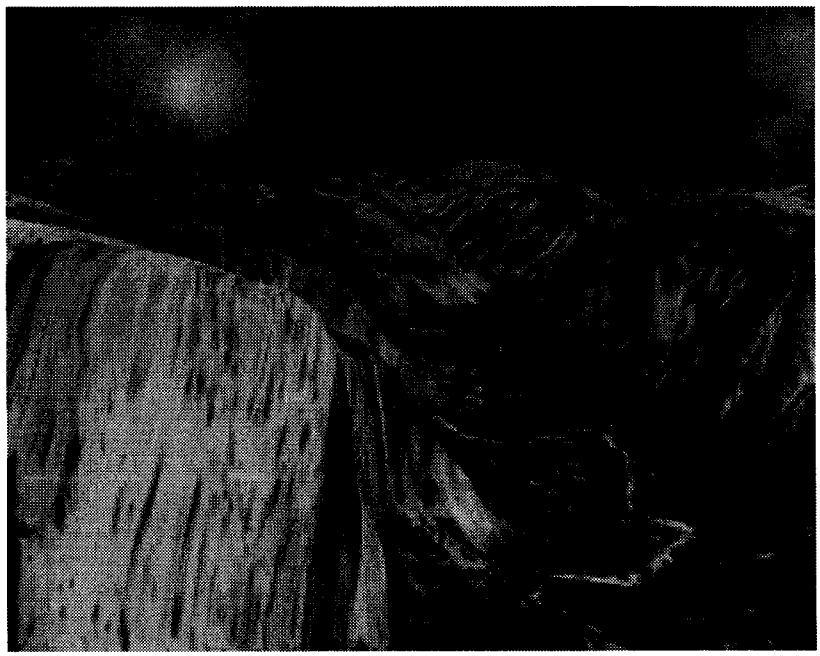


Fig 17. Yosemite fly-by image

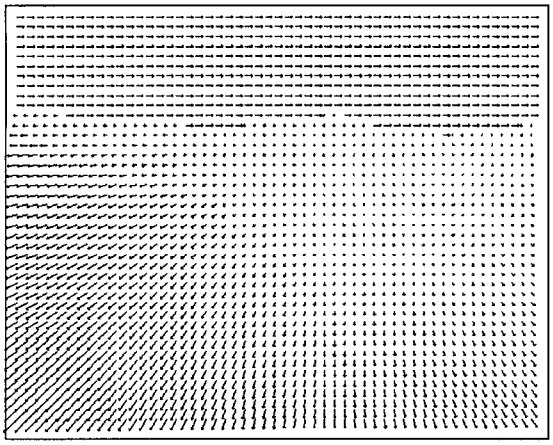


Fig 18.1 True optical flow for Yosemite fly-by

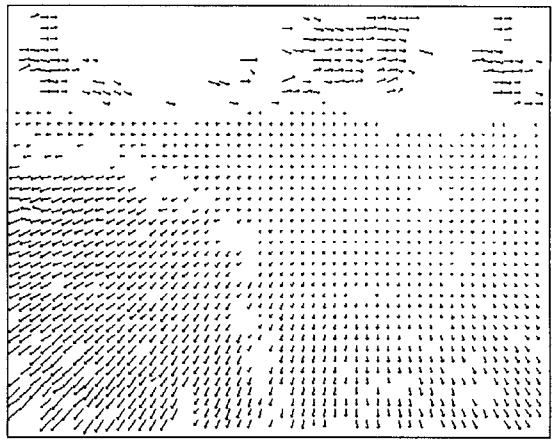


Fig 18.2 Computed optical flow for Yosemite fly-by

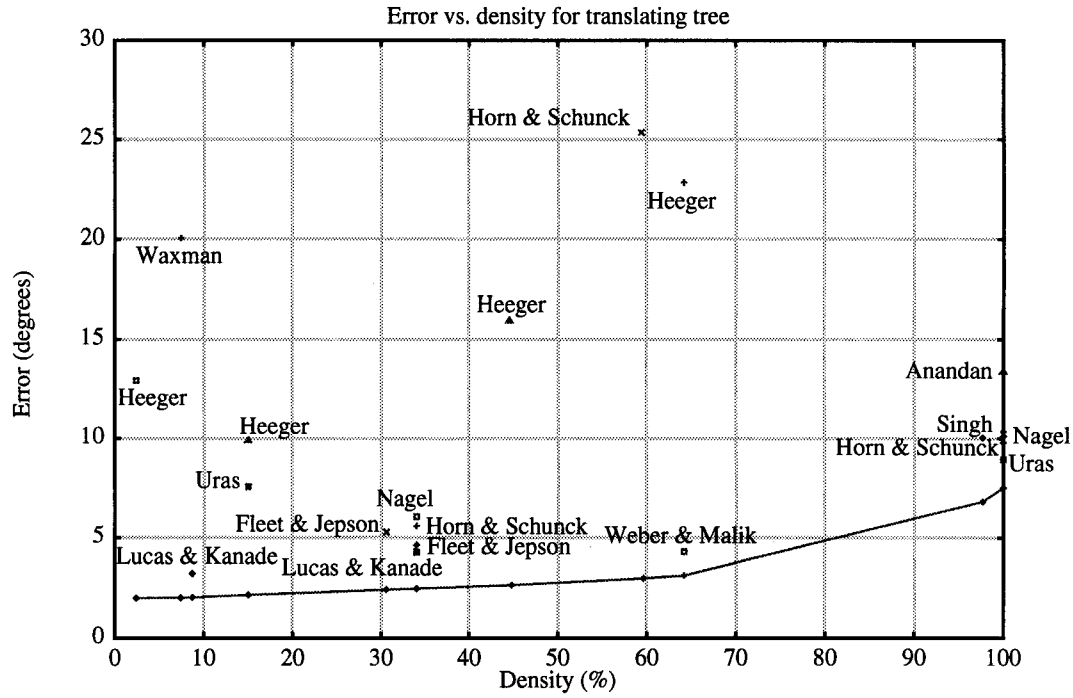


Fig 19. Comparison plot for Yosemite fly-by sequence

3.1.4 Moon landing

The Moon landing sequence (Fig 20) is generated by gradually rotating and expanding* a picture of the surface of the moon. Visually, it is a bird's-eye view of the moon from a spiral landing spaceship. The purpose of this sequence is to demonstrate our algorithm's capability to handle complex motion, specifically, expansion plus rotation. Our algorithm used a 21x21x7 window and $|\lambda|_{min}$ as the confidence measure since there are no

*. Rotation and expansion are done using Khoros 1.5 vrotate and vresize functions, respectively.

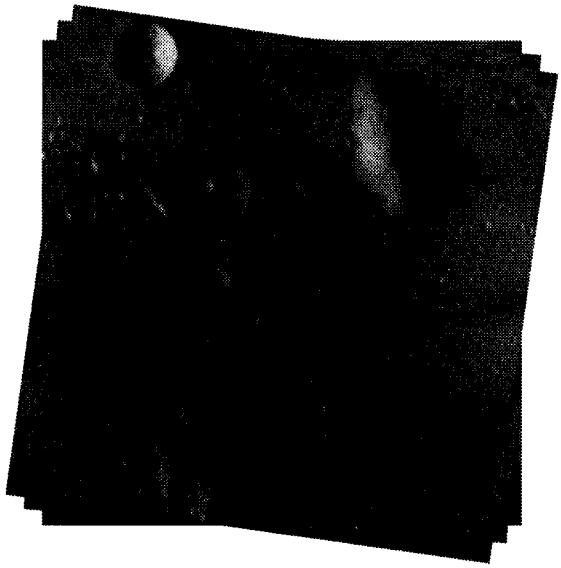


Fig 20. Moon landing sequence

Table 2: Summary of Translating Tree error statistics

Density	Our Algorithm		Other Algorithms		
	Average Error	Standard Deviation	Average Error	Standard Deviation	Technique by
100%	0.66°	0.83°	38.72°	27.67°	Horn & Schunck (original unthresholded)
			2.02°	2.27°	Horn & Schunck (modified unthresholded)
			0.62°	0.52°	Uras et al. (unthresholded)
			2.44°	3.06°	Nagel
			4.54°	3.10°	Anandan
			1.64°	2.44°	Singh (step 1 unthresholded)
			1.25°	3.29°	Singh (step 2 unthresholded)
			0.54°	1.69°	Bober and Kittler (no smoothing)
99.6%	0.66°	0.82°	1.11°	0.89°	Singh (step 2)
96.8%	0.59°	0.66°	0.49°	0.35°	Weber and Malik
74.5%	0.46°	0.45°	0.32°	0.38°	Fleet and Jepson
53-57%	0.40°	0.38°	32.66°	24.50°	Horn & Schunck (original)
			5.63°	2.78°	Heeger (level 1)
			1.89°	2.40°	Horn & Schunck (modified)
49.7%	0.39°	0.36°	0.23°	0.19°	Fleet and Jepson
44.2%	0.38°	0.35°	8.50°	13.50°	Heeger (level 0)
40-42%	0.37°	0.34°	0.46°	0.35°	Uras et al.
			0.72°	0.75°	Singh (step 1)
			0.66°	0.67°	Lucas and Kanade
26.8%	0.33°	0.29°	0.25°	0.21°	Fleet and Jepson
13.1%	0.29°	0.25°	0.56°	0.58°	Lucas and Kanade
1.9%	0.27°	0.26°	6.66°	10.72°	Waxman et al.

motion boundaries. Table 5 shows that our results are better than Fleet and Jepson's and

Table 3: Summary of Diverging tree error statistics

Density	Our Algorithm		Other Algorithm		
	Average Error	Standard Deviation	Average Error	Standard Deviation	Technique by
100%	1.86°	1.35°	12.02°	11.72°	Horn & Schunck (original unthresholded)
			2.55°	3.67°	Horn & Schunck (modified unthresholded)
			4.64°	3.48°	Uras et al. (unthresholded)
			2.94°	3.23°	Nagel
			7.64°	4.96°	Anandan
			17.66°	14.25°	Singh (step 1 unthresholded)
			8.60°	4.78°	Singh (step 2 unthresholded)
			3.69°	4.39°	Bober and Kittler
99%	1.84°	1.32°	8.40°	4.78°	Singh (step 2)
88.6%	1.74°	1.22°	3.18°	2.50°	Weber and Malik
73.8%	1.63°	1.15°	4.95°	3.09°	Heeger (combined)
60-61%	1.54°	1.06°	0.99°	0.78°	Fleet and Jepson
			8.93°	7.79°	Horn & Schunck (original)
			3.83°	2.19°	Uras et al.
46-48%	1.42°	0.95°	2.50°	3.89°	Horn & Schunck (modified)
			0.80°	0.73°	Fleet and Jepson
			1.94°	2.06°	Lucas and Kanade
28.2%	1.29°	0.81°	0.73°	0.46°	Fleet and Jepson
24.3%	1.26°	0.77°	1.65°	1.48°	Lucas and Kanade
3.9-4.9%	1.16°	0.63°	13.69°	11.83°	Waxman et al.
			5.62°	6.16°	Singh (step 1)

Table 4: Summary of Yosemite fly-by error statistics

Density	Our Algorithm		Other Algorithms		
	Average Error	Standard Deviation	Average Error	Standard Deviation	Technique by
100%	7.52°	13.72°	31.69°	31.18°	Horn & Schunck (original unthresholded)
			9.78°	16.19°	Horn & Schunck (modified unthresholded)
			8.94°	15.61°	Uras et al. (unthresholded)
			10.22°	16.51°	Nagel
			13.36°	15.64°	Anandan
			15.28°	19.61°	Singh (step 1 unthresholded)
			10.44°	13.94°	Singh (step 2 unthresholded)
97.7%	6.82°	12.41°	10.03°	13.13°	Singh (step 2)
64.2%	3.13°	4.28°	22.82°	35.28°	Heeger (level 0)
			4.31°	8.66°	Weber and Malik
59.6%	2.99°	3.98°	25.33°	28.51°	Horn & Schunck (original)
44.8%	2.66°	3.43°	15.93°	23.16°	Heeger (combined)

Table 4: Summary of Yosemite fly-by error statistics

Density	Our Algorithm		Other Algorithms		
	Average Error	Standard Deviation	Average Error	Standard Deviation	Technique by
33-35%	2.47°	3.11°	4.28°	11.41°	Lucas and Kanade
			4.63°	13.42°	Fleet and Jepson
			5.59°	11.52°	Horn & Schunck (modified)
			6.06°	12.02°	Nagel
30.6%	2.42°	3.02°	5.28°	14.34°	Fleet and Jepson
15%	2.17°	2.81°	9.87°	14.74°	Heeger (level 1)
			7.55°	19.64°	Uras et al.
8.7%	2.04°	2.76°	3.22°	8.92°	Lucas and Kanade
7.4%	2.03°	2.59°	20.05°	23.23°	Waxman et al.
2.4%	2.01°	2.42°	12.93°	15.36°	Heeger (level 2)

Table 5: Summary of Moon landing error statistics

Density	Algorithms		
	Average Error	Standard Deviation	Technique
100%	1.73°	0.87°	Our algorithm (Translation + Rotation +Expansion model)
	1.91°	0.89°	Our algorithm (Translation model)
33.3%	3.91°	3.80°	Lucas and Kanade
	1.37°	0.71°	Our algorithm (Translation + Rotation +Expansion model)
	1.69°	0.83°	Our algorithm (Translation model)
31.1%	2.47°	1.71°	Fleet and Jepson
	1.36°	0.70°	Our algorithm (Translation + Rotation +Expansion model)
	1.68°	0.82°	Our algorithm (Translation model)

Lucas & Kanade's (also see Fig 21.) It also reveals the amount of improvement (10% to 16%) of a generalized motion model over a uniform translation motion model in our algorithm.

3.1.5 Noise sensitivity

We created noisy images from the synthetic sequences used above and tested the sensitivity of the algorithms to such noise.

The sensitivity analysis is motivated by simple experiments such as the following: On a real-time image processing machine, suppose we run a temporal differencing algorithm at video rate on successive frames while keeping the camera and the scene stationary. Instead of getting a uniform output of zero, the actual output always contains random spots of non-zero values. This kind of sensor noise violates the brightness constancy assumption and degrades the accuracy of any optical flow algorithm.

In Chapter 2, we proved analytically that the magnitude of optical flow error is linear with the magnitude of noise (44), provided that the noise is significantly smaller than the

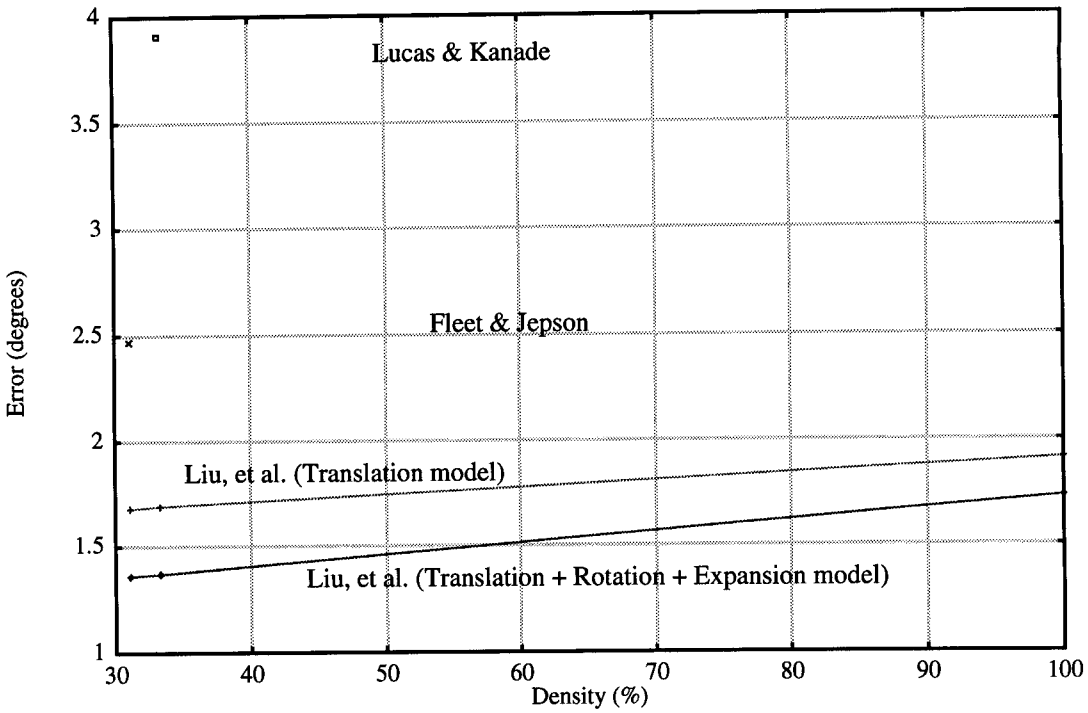


Fig 21. Comparison plot for moon landing sequence

image intensity. Here we report results of experiments that support the claim.
In the following tables, we used additive Gaussian noise of zero mean and increasing variance. In order to conduct a fair comparison, the threshold on the confidence measure is fine-tuned in every single run so that the output density is always 50%. We chose two of the best algorithms in [9] , Lucas & Kanade and Fleet & Jepson, for comparison. For the noisy Diverging tree sequence, the noise sensitivity is summarized in Table 6.

Table 6: Diverging tree noise sensitivity statistics

Noise Standard Deviation	Our Algorithm		Fleet & Jepson		Lucas & Kanade	
	Average Error	Standard Deviation	Average Error	Standard Deviation	Average Error	Standard Deviation
0	1.41°	0.94°	1.09°	0.52°	3.04°	2.53°
3	1.64°	1.08°	1.18°	0.61°	3.28°	2.77°
6	2.03°	1.37°	1.51°	0.93°	3.62°	3.06°
9	2.53°	2.17°	2.15°	1.78°	4.32°	3.79°
12	3.28°	2.78°	3.83°	5.48°	5.17°	4.69°
15	3.87°	3.15°	9.23°	12.04°	5.93°	5.41°

Both our algorithm and Lucas & Kanade's have an approximately linear error increase with noise while Fleet & Jepson's has a quadratic or even exponential error increase (Fig 23). Despite its excellent accuracy for noise-free data, Fleet & Jepson's algorithm performs worse than the other two when the image sequence becomes noisy (see also

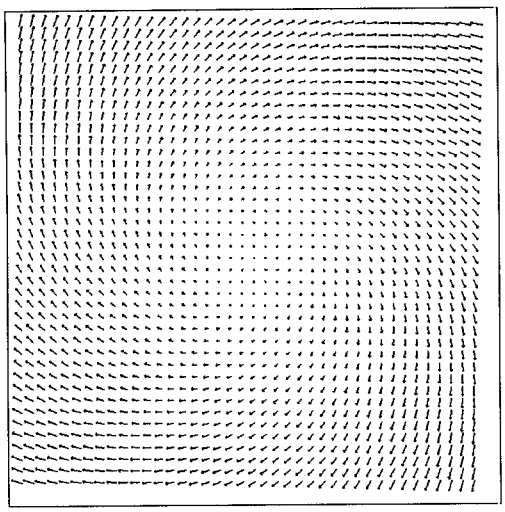


Fig 22.1 Moon landing ground truth flow

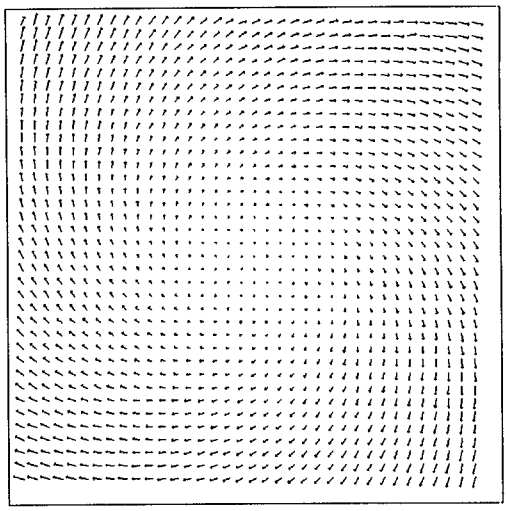


Fig 22.2 Our algorithm's flow field (100%)

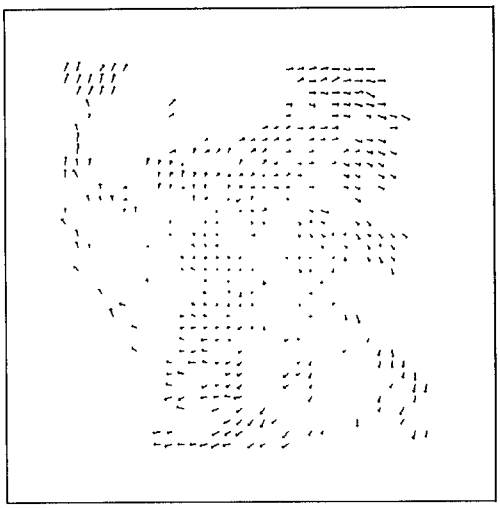


Fig 22.3 Lucas & Kanade's flow field (33.3%)

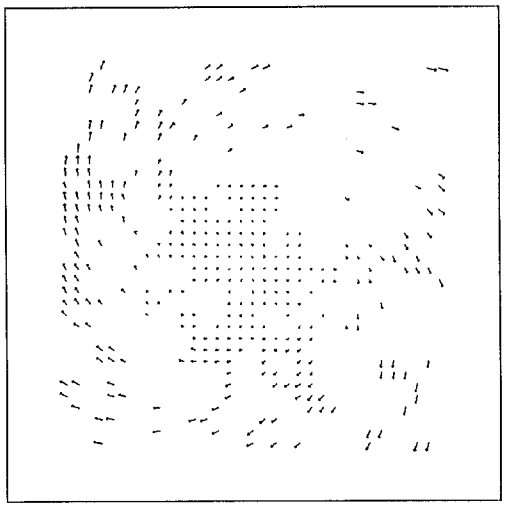


Fig 22.4 Fleet & Jepson's flow field (31.1%)

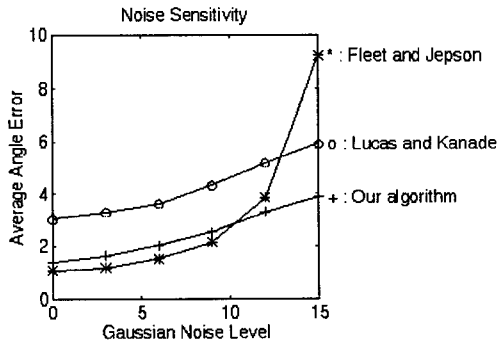


Fig 23. Noise sensitivity plot for Diverging tree

[11]). We also conducted a similar experiment with the Yosemite fly-by sequence. Unfortunately, Fleet & Jepson's algorithm does not generate high enough density data

when the sequence becomes noisy. However, we can once again confirm the linear error with respect to noise for Lucas & Kanade's and our algorithm (Fig 24).

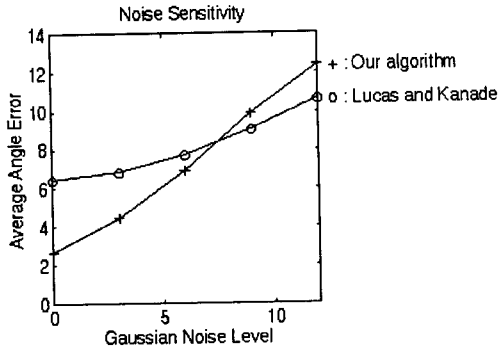


Fig 24. Yosemite fly-by noise sensitivity

Robustness to noise is a very important quality for good optical flow algorithms. Our algorithm achieves robustness by integrating spatio-temporal smoothing with the 3-D Hermite polynomial differentiation filter theory.

3.2 Qualitative Evaluation Using Real Images

Many existing optical flow algorithms often have difficulty with real image sequences. Our algorithm performs best on the Yosemite and Moon landing sequences because these sequences model real 3-D motion and are thus complicated enough to reveal the strengths of our algorithm. Here we demonstrate our algorithm with the following real image sequences: SRI trees, Hamburg taxi, Rubik Cube, NASA and HMMWV. The HMMWV sequence was taken in an outdoor environment with a camera mounted on a forward moving HMMWV (High Mobility Multipurpose Wheeled Vehicle). It was later stabilized[126] to eliminate unsteady motion and then all algorithms were applied to the stabilized images. All the other sequences were obtained from Barron [9] . For the first three sequences, we display our flow results only; readers may refer to Barron, et al. [9] for the results of other algorithms. For the HMMWV sequence, the flow outputs for our algorithm as well as Lucas & Kanade's, Fleet & Jepson's, and Anandan's are displayed in Fig 28. Finally, we use the NASA sequence to demonstrate an obstacle detection capability based on optical flow and make a comparison among the algorithms. In our algorithm's implementation, the output has undergone thresholding based on two confidence measures, $|1/r|$ and $|\lambda|_{min}$.

3.2.1 SRI trees, Hamburg taxi, and Rubik cube sequences

In the SRI trees sequence (Fig 25.1), the camera translates parallel to the ground plane, perpendicular to its line of sight, in front of clusters of trees. This image sequence contains many occluding boundaries and is very noisy. It is particularly challenging because of the presence of thin tree branches and small leaves, which requires the use of a small window (13x13x5). Our flow result (Fig 25.2) displays the scene structure clearly (tree

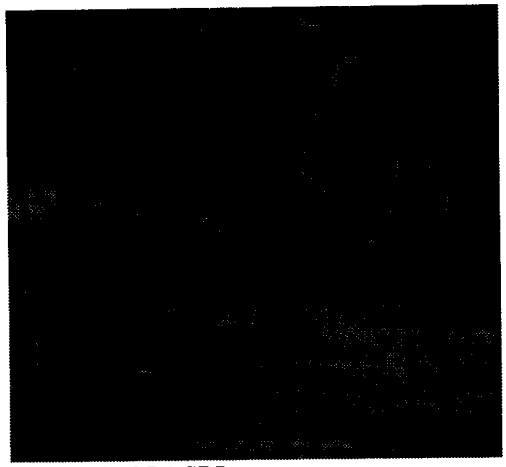


Fig 25.1 SRI trees sequence

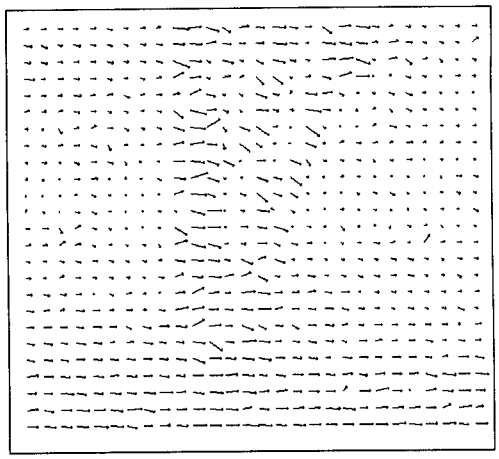


Fig 25.2 Our algorithm's flow output (100%)

in the center, depth of the ground). In the Rubik cube sequence (Fig 26.1), a Rubik cube

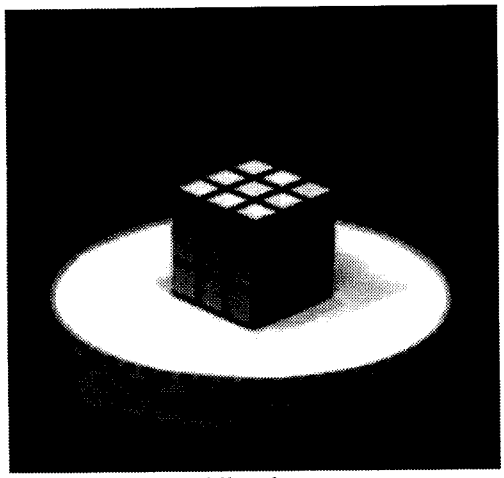


Fig 26.1 Rubik cube sequence

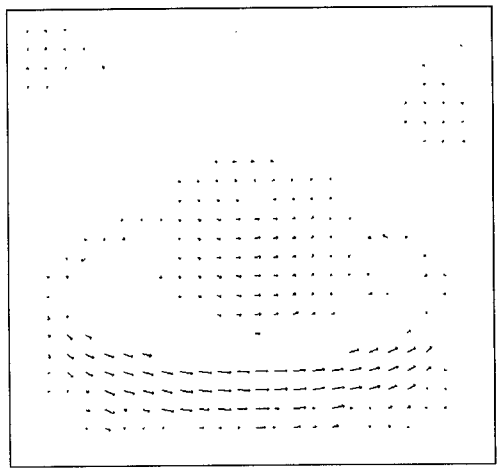


Fig 26.2 Our algorithm's flow output (44.2%)

is rotating counter-clockwise on a turntable. This sequence contains many regions with 1-D (side patterns on the turntable) or 2-D (top of the turntable) aperture problems and the cube's motion is very small. The flow result (Fig 26.2) demonstrates our algorithm's capability to extract 2-D shape (cube and turntable) from confidence measures, handle aperture problems, and estimate subpixel motions. In the Hamburg Taxi sequence (Fig 27.1), there are three vehicles and a pedestrian moving independently. The flow result (Fig 27.2) demonstrates our algorithm's capability of handling widely varying motion velocities.

Not all the capabilities that our algorithm possesses are common in most of the optical flow algorithms reported in [9] . We observe from the flow results in [9] that those algorithms tend to be adapted to a limited set of situations but do not do well in other situations. In this qualitative evaluation, our algorithm appears to be more versatile than other existing algorithms. This can be attributed to the good filter design and the confidence measures. In the following sections, we demonstrate the advantage of the general motion model.

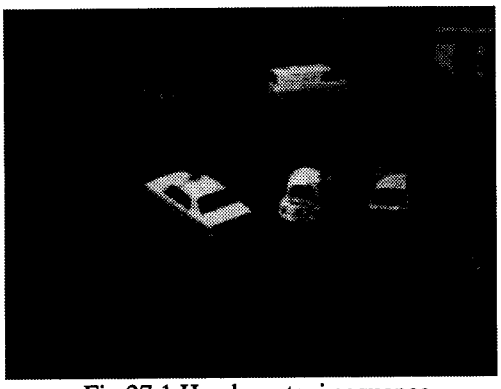


Fig 27.1 Hamburg taxi sequence

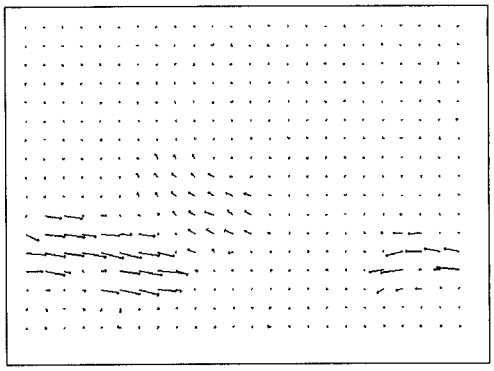


Fig 27.2 Our algorithm's flow output (100%)

3.2.2 HMMWV sequence

In the HMMWV (High Mobility Multi-Purpose Vehicle) sequence, visual inspection shows that Lucas & Kanade's flow* output (Fig 28.4) is very noisy, as evidenced by random velocity changes in small neighborhoods. Fleet & Jepson's flow output (Fig 28.5) shows no indication of the flow field divergence. It is probable that the particular implementations (provided by Barron) of these two algorithms are not tuned to the relatively large flow existing in this sequence. The implementations of both Anandan's and our algorithm are suitable for large flow and the output flow fields of both are quite good. Anandan's flow field (Fig 28.2) appears to be the smoothest due to the use of neighborhood smoothness constraint. But a close inspection on the image border reveals an apparent inconsistency since the rightmost columns of the flow field are converging. Our algorithm, on the other hand, produces a coherently diverging flow field (Fig 28.3) except in the area of the sky.

3.2.3 NASA sequence and obstacle detection demonstration

In the NASA sequence, both our flow (Fig 29.2) and Fleet & Jepson's flow† outputs (Fig 29.4) are very good, while Lucas & Kanade's algorithm produces a noisy flow field (Fig 29.3). Note that our output density is twice that of Fleet & Jepson's but it achieves approximately the same accuracy visually. If Fleet & Jepson were to generate the same density, it might not be as accurate.

Finally we apply the NASA flow field outputs from these three algorithms to an obstacle detection algorithm developed by Young, et al.[122] [123] [124] . This algorithm discriminates between obstacle and non-obstacle regions in the image using only the component of flow perpendicular to arbitrarily chosen image lines. In the following figures, a

---

*. $\sigma = 0.8$ is used for both Lucas & Kanade's and Fleet & Jepson's algorithms because only 10 image frames are available. The filter size of our algorithm is 21x21x7.

†. $\sigma = 2.0$ is used for both Lucas & Kanade's and Fleet & Jepson's algorithms. The filter size of our algorithm is 21x21x7.

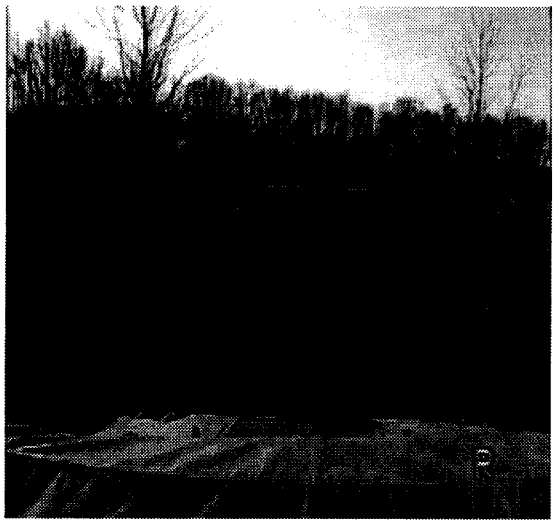


Fig 28.1 HMMWV sequence

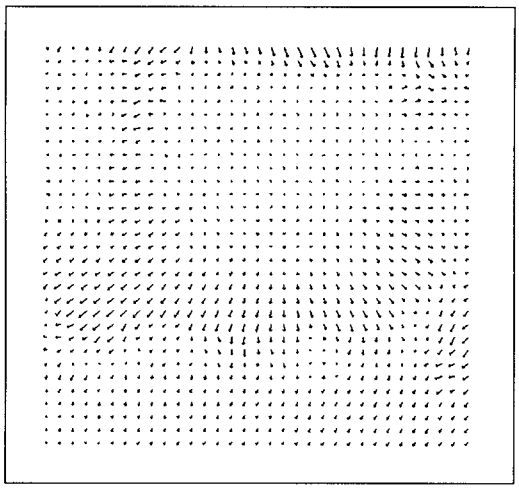


Fig 28.2 Anadan's flow field (100%)

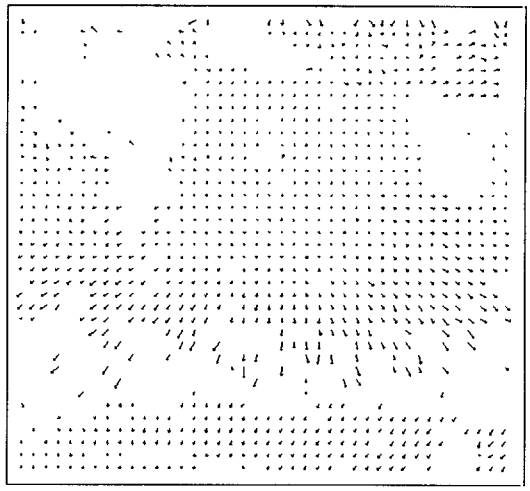


Fig 28.3 Our algorithm's flow field (64% density)

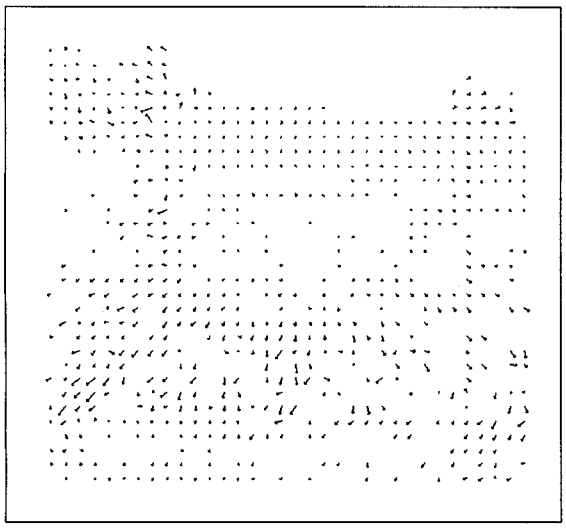


Fig 28.4 Lucas & Kanade's flow field (48%)

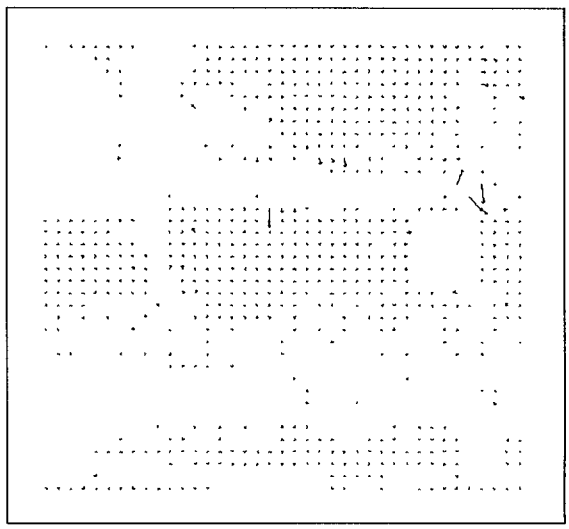


Fig 28.5 Fleet & Jepson's flow field (34%)

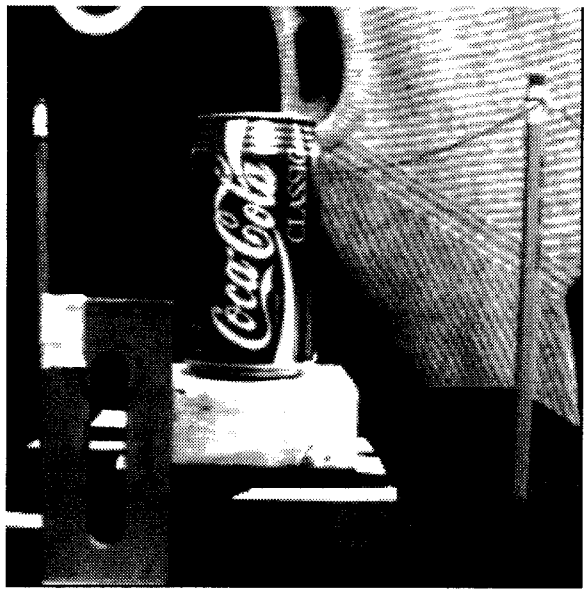


Fig 29.1 NASA sequence

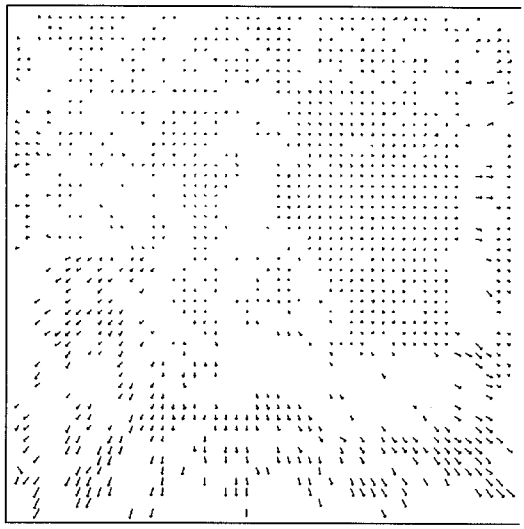


Fig 29.2 Our algorithm's flow field (75% density)

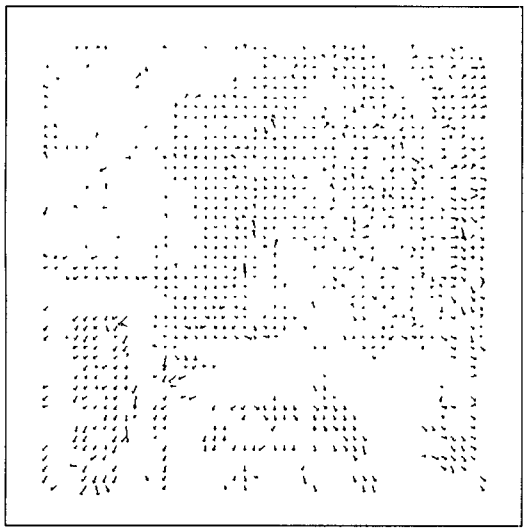


Fig 29.3 Lucas & Kanade's flow field (48% density)

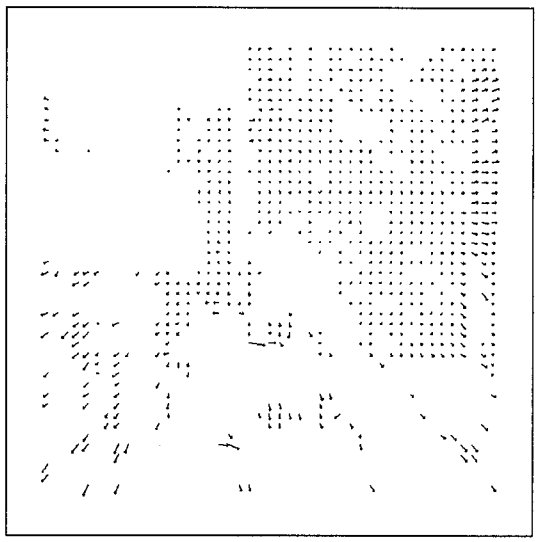


Fig 29.4 Fleet & Jepson's flow field (37% density)

protrusion or a depression represents an obstacle detected by the algorithm.

For the first set of data, we select the horizontal lines from 220 to 260 (Fig 30.2). Over these lines, there is a vertical long metal plate with a hole in the left end of the image strip. We should expect two depressions at the locations of the plate. The detection results from all lines are averaged and then displayed in Fig 31. In Fig 31.1, Lucas & Kanade's flow does not detect the metal plate clearly; in Fig 31.2, Fleet & Jepson's flow detects the metal plate but its shape is hardly recognizable; in Fig 31..3, our algorithm not only detects the plate but also shows its shape as should be expected.

For the second set of data, we select horizontal lines 45 to 90 (Fig 30.1). Over these lines, there is a pole on each end of the image strip and a coke can at the center. In Fig 32.1, Lucas & Kanade's flow does not make out meaningful objects. In Fig 32.2,

Fig 30.1 NASA image lines 45 to 90

Fig 30.2 NASA image lines 220 to 260

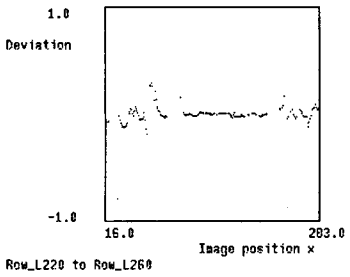


Fig 31.1 Lucas & Kanade's results

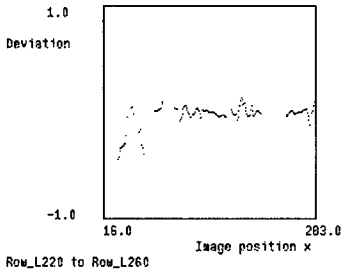


Fig 31.2 Fleet & Jepson's results

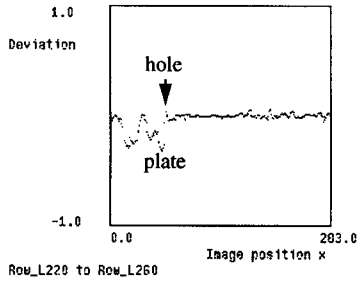


Fig 31.3 Our algorithm's results

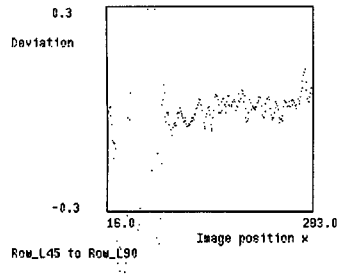


Fig 32.1 Lucas & Kanade's results

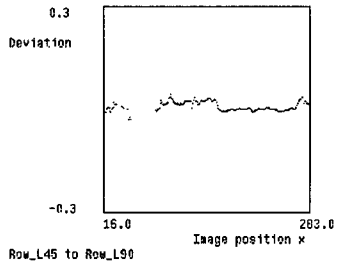


Fig 32.2 Fleet & Jepson's results

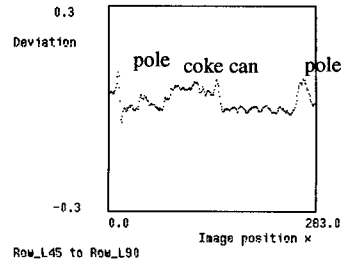


Fig 32.3 Our algorithm's results

Fleet & Jepson's flow barely detects the coke can and the right pole, and does not detect the left pole. In Fig 32.3, our flow clearly detects all three objects. Note that for this particular image strip we used dense (100%) flow for our algorithm.

In summary, compared with other existing optical flow algorithms, our algorithm offers the qualities of accuracy, flexibility, and adequate noise sensitivity, which are very crucial for real world tasks.

Chapter 4
Motion-Model-Based Boundary Extraction

Motion boundary extraction and optical flow computation are two subproblems of the motion recovery problem that cannot be solved independent of one another. We present a local, non-iterative algorithm that extracts motion boundaries and computes optical flow simultaneously. This is achieved by modeling a 3-D image intensity block with the general motion model formulated in the previous chapter that presumes locally coherent motion. Local motion coherence, which is measured by the fitness of the motion model, is the criterion we use to determine whether motion should be estimated. If not, then motion boundaries should be located. The motion boundary extraction algorithm is evaluated quantitatively and qualitatively against other existing algorithms using a scheme originally developed for edge detection.

4.1 Introduction

This chapter studies the strengths and weaknesses of recent motion boundary detection and motion segmentation algorithms and proposes a local, non-iterative algorithm for motion boundary detection with potential for real-time implementation. This algorithm extracts motion boundaries and computes optical flow at the same time. We apply a quantitative evaluation scheme for boundary detection to show that our algorithm is accurate in locating motion boundaries.

In this chapter, the problem of *motion recovery* is referred to as involving two major subproblems: *optical flow computation* and *motion segmentation*. Optical flow computation quantitatively measures the motion associated with the perceived objects; motion segmentation, on the other hand, qualitatively distinguishes different moving objects. The fact that they are dependent on each other has complicated the general motion recovery problem.

Due to the aperture problem, motion estimation algorithms [47] [48] developed earlier usually enforced a smooth flow field as an additional constraint. Recent approaches use spatio-temporal filters [34] [44] [65] , often with large support, to estimate image properties and then solve for optical flow. In either case, on or near motion boundaries, this smoothing or filtering renders the estimation incorrect. In other words, motion estimation is not accurate until we know where the boundaries are. On the other hand, motion boundaries are defined as motion field discontinuities. (The motion field is qualitatively equivalent to the optical flow field [110] .) Due to the aforementioned optical flow error around motion boundaries, the requirement of a dense flow field, and noise in the optical flow field, the motion boundaries are very difficult to extract and/or locate from optical flow. Researchers have used other image cues, for example, accretion and deletion[75] ,

or normal flow [47] , to detect motion discontinuities, but they are not always correct for all situations because they provide only partial information about the motion. In other words, motion boundaries cannot be located accurately without a dense and accurate optical flow field.

Even though they are two aspects of the same problem, optical flow computation has received much more attention in the literature than motion boundary extraction. Existing methods for motion boundary extraction are approached through optical flow algorithms. A popular technique is to use an iterative scheme that consists of two components: optical flow estimation and motion boundary extraction. The basic idea is to refine both components' results through iteration. This approach is time-consuming and sometimes does not converge. We believe that optical flow and motion boundaries are of equal importance and we present an algorithm that produces both outputs at the same time.

Although "global" motion segmentation may be more convenient for other motion applications, we realize, from the above analysis, that "local" motion boundary detection/ extraction is sufficient for combining with optical flow an algorithm for motion recovery. In fact, our view is that local image properties provide abundant information and motion estimation should be performed pointwise [63] .

The local properties that we use are image spatial and temporal derivatives up to third order. It has been shown in [63] that Hermite polynomial differentiation filters are very stable and insensitive to noise even up to this high order. With the aid of sufficient and accurate image properties, a motion-model-based approach to boundary extraction becomes possible. In this approach, we fit the image properties with a single coherent motion model, which leads to a linear system of multiple motion constraint equations. Pixels that fit the model are locations where the motion is coherent, so the motion can be estimated using the linear system. Those pixels that do not fit the model represent failures of the motion model in describing the local motion. A failure of the model can only be attributed to multiple motions existing in the local window used to estimate the image properties, assuming that brightness constancy is maintained. Using a least square error method on the overdetermined linear system, a failure of the model is measured by the residual. An analysis of the residual is shown to reflect the likelihood of a motion boundary. Using an optimal filter on the residual, we can easily locate motion boundaries.

Using the residual for motion boundary extraction offers several advantages over using flow. First, the residual is a scalar, so it avoids the difficulty of handling vector field discontinuities while providing equivalent information about motion boundaries, e.g., whenever one component of the flow is discontinuous, the residual is high. Second, flow values on the boundaries are not accurate and are very noisy, and thus require smoothing for boundary extraction. This extra smoothing may cause localization error. Third, the residual is computed using a 3-D motion model so that it corresponds to real motion boundaries and it is not susceptible to nonuniform flow within an object; whereas nonuniform flow can induce false detections of discontinuities in flow-based methods.

The appeal of a local, non-iterative approach lies in its potential speed. However, its accuracy should not be compromised. To measure the accuracy, we need an evaluation scheme to compare different motion boundary extraction algorithms. Since recent approaches combine optical flow and motion boundary detection, evaluation has often

been performed based on the final optical flow. This has the disadvantage of not distinguishing the source of error, which may be due to inaccurate optical flow or inaccurate motion boundary location. In other words, evaluation based on segmented optical flow does not suggest a direction for improvement. Hence, we employ here a quantitative evaluation scheme applied only to motion boundary extraction. This scheme takes into account not only the probabilities of detection and miss but also localization error. This scheme was originally developed for edge detection [46] .

4.2 Previous Work

Braddick's psychological experiments on random dot motion [12] set the stage for vision research on motion boundaries. It verified the human visual capability of perceiving motion boundaries clearly without any other visual cues such as texture. Table 7 summarizes the existing work on motion boundary extraction or segmentation. This survey is not exhaustive but represents typical work in this area, which will be elaborated in the following subsections.

Table 7: Recent motion boundary extraction algorithms

Non-iterative schemes		**Iterative schemes**	
Motion boundary extraction vs. flow estimation	**Algorithm by**	**Techniques**	**Algorithm by**
Prior to	Hildreth [47]	Pyramid linking	Hartley [42]
	Spoerri & Ullman [101]	Markov random field with binary line processes	Koch, Marroquin & Yuille [56] , Murray & Buxton [74] , Heitz & Bouthemy [45]
Simultaneous with	Mutch & Thompson [75]		
	Schunck [93]		
	Shizawa & Mase [95]	Tracking & nulling	Bergen et al. [10]
After	Potter [89] , Nakayama & Loomis [79] , Adiv [2] , Thompson, Mutch, & Berzins [106] , Dengler [28]	Robust estimation	Darrel & Pentland [26] Jepson & Black [53]

4.2.1 Non-iterative algorithms

Early research on motion boundary extraction or motion segmentation can be roughly characterized as based on a non-iterative approach. These algorithms can also be put into three categories [28] [101] based on whether the motion boundary extraction is performed prior to, simultaneously with, or after the flow field estimation (refer to Table 7). The approaches that extract motion boundaries prior to flow field estimation employ "motion primitives"[101] , usually normal flow [47] , as a basis. Hildreth's method [47] is based on the intensity zero-crossing contours, which are different from motion

contours, and may well cross motion boundaries. Traveling along a contour, the algorithm detects a boundary point as the sign of the normal flow changes. The method has two limitations [47] : first, it does not detect boundaries when the neighboring moving objects are traveling in about the same direction; second, it requires that there be two edge points with the same orientation in the contour. In addition, due to the use of contours, the boundaries detected are sparse, which is very restrictive for general applications. Spoerri & Ullman [101] use a local histogram on motion primitives and statistical tests to infer motion boundaries, or tracking of "thin-bars" to find occlusion. This method is quite appealing for the diversity of statistical tests offered, but the motion primitives do not always provide sufficient information about the boundaries. The experiments show detection capabilities but the localization errors are significant even in synthetic images.

The approaches that extract motion boundaries simultaneously with flow field estimation include [75] ,[93] ,and [95] . Mutch & Thompson [75] use the fact that motion around occlusion boundaries induces accretion and deletion so that a local no-match between successive frames can signal occlusion boundaries, whereas a match can be used for estimating optical flow. This algorithm detects only occlusion boundaries but not all motion boundaries, for instance, two neighboring objects moving in parallel directions or a rotating object where no accretion or deletion occurs, will not be detected. Schunck's algorithm [93] pays special attention to avoiding optical flow ambiguity at motion boundaries by constraint line clustering. The motion boundaries are actually detected from the flow field discontinuities. We categorize this algorithm as performing simultaneous estimation and segmentation because of its special treatment of boundary flow. The basic idea of the algorithm is to use local consensus to assign flow values instead of smoothing. The algorithm produces high localization error on motion boundary "corners". The clustering technique is heuristic and is prone to numerical instability. Shizawa & Mase [95] use "multiple-flow constraint equations", a generalization of the common optical flow constraint equation, to deal with motion boundaries and/or transparent motion. The algorithm generates not only flow but also a measure of the degree of multiplicity of motion. When a pixel's associated multiplicity is determined to be greater than one, it is on a motion boundary or where transparent motion occurs. This method unifies rather than distinguishes motion boundaries and transparent motion. However, it is more suitable for transparent motion than motion boundary extraction because of the assumption of additive multiple flows, which is less valid around motion boundaries.

The approach that extracts motion boundaries after flow field estimation is the most popular one. Global techniques such as the Hough transform (Adiv [2]), region growing (Potter [89]), and pyramid linking (Dengler [28]) have been proposed. Local techniques include center-surround filters (Nakayama & Loomis [79]), and direction reversals of the Laplacian operator on the flow vector field (Thompson, Mutch, & Berzins [106]), etc. This approach offers only a partial solution to the motion estimation problem because boundary detection depends heavily on the accuracy of the optical flow. However, this approach offers an algorithm suitable for one component of an iterative scheme. Such schemes are described next.

4.2.2 Iterative approach

The iterative method of motion estimation is an approach developed more recently. It has both optical flow estimation and segmentation components. These components interact with each other and improve their individual results during the course of the iteration. Pyramid linking, Markov random fields with line processes, robust estimation, and tracking plus nulling techniques have been proposed. Iterative methods tend to be more accurate than non-iterative methods but are time-consuming. Note that there are algorithms that use an iterative scheme to compute optical flow only. However, we do not label them as iterative methods here since they do not include the segmentation component.

Hartley's algorithm [42] uses an iterative pyramid linking technique for flow field segmentation. Segmentation is done by hierarchical linking and the flow field is computed and smoothed by fitting a linear or quadratic flow field model to the current flow. The algorithm is efficient and always converges but its overall accuracy depends heavily on the initial flow values, which the author does not address.

The use of a Markov random field model for flow has been proposed by Koch, Marroquin & Yuille [56] , Murray & Buxton [73] , and Heitz & Bouthemy [45] . They handle flow discontinuities by introducing a binary line process to discourage smoothing across boundaries. Although the reason for modeling the flow fields as Markov random fields is not clear, the results of these algorithms are generally good. The computational cost, however, is formidable (usually hundreds of iterations, or image sweeps).

Robust estimation techniques have been proposed by Darrel & Pentland [26] , and Jepson & Black [53] . They use a multi-layered motion model ("mixture model"[53]) and thus are capable of handling loosely occluded scenes (e.g., tree leaves) or transparent motion. The main idea is to estimate the dominant motion(s) in a window while rejecting inconsistent constraints as outliers so as to minimize their influence on the results. The results are promising when the algorithms converge.

Instead of using a layered motion model, Bergen et al. [10] model the addition of motions of differently moving image patterns (not necessarily square, as dictated by the window). A simple tracking and a 'nulling' mechanism is used to separate and estimate individual motions. In other words, image registration and residual motion estimation are iterated. This algorithm has potential for high speed implementation on a system with warping hardware. The results are reasonably good but the algorithm may not always converge, depending on the noise level.

The results of the iterative methods seem good, but they have two major problems. The first is the computational load. the second is that the convergence rate depends on the scene, noise, and motion. Moreover, some of these algorithms may not converge at all.

4.3 Motion-Model-Based Boundary Extraction

The basic idea of our motion-model-based boundary extraction method is to fit the local image properties with a general motion model. The necessary elements of the scheme are a general motion model which is based on arbitrary 3-D motion; an accurate estimate of image properties, for which we use image spatial and temporal derivatives; and a pro-

cedure to measure the goodness of the image properties to the motion model. The following subsections briefly present the derivations of these three elements; the details can be found in [63] and [65] .

4.3.1 Spatial and temporal image derivatives

We estimate image spatial and temporal derivatives with Hermite polynomial differentiation filters described in previous chapters. These filters are orthogonal and their Gaussian derivative properties provide the numerical stability required. We use this filtering scheme and the motion equation (37) to derive a linear system of equations. In linear least square form,

$$E = min\|As + b\|, \text{ where} \tag{45}$$

$$s = \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \rho \end{bmatrix}, b = \begin{bmatrix} \hat{I}_t \\ \hat{I}_{xt} \\ \hat{I}_{yt} \\ \hat{I}_{xxt} \\ \hat{I}_{xyt} \\ \hat{I}_{yyt} \end{bmatrix}, A = \begin{bmatrix} \hat{I}_x & \hat{I}_y & \sigma^2(\hat{I}_{xx}+\hat{I}_{yy}) & 0 \\ \hat{I}_{xx} & \hat{I}_{xy} & \sigma^2(\hat{I}_{xxx}+\hat{I}_{xyy})+\hat{I}_x & -\hat{I}_y \\ \hat{I}_{xy} & \hat{I}_{yy} & \sigma^2(\hat{I}_{xxy}+\hat{I}_{xxx})+\hat{I}_y & \hat{I}_x \\ \hat{I}_{xxx} & \hat{I}_{xxy} & \sigma^2(\hat{I}_{xxxx}+\hat{I}_{xxyy})+2\hat{I}_{yy} & -2\hat{I}_{xy} \\ \hat{I}_{xxy} & \hat{I}_{xyy} & \sigma^2(\hat{I}_{xxxy}+\hat{I}_{xyyy})+2\hat{I}_{xy} & \hat{I}_{xx}-\hat{I}_{yy} \\ \hat{I}_{xyy} & \hat{I}_{yyy} & \sigma^2(\hat{I}_{xxyy}+\hat{I}_{yyyy})+2\hat{I}_{yy} & 2\hat{I}_{xy} \end{bmatrix}, \tag{46}$$

where the $\hat{I}$'s are derivatives computed by Hermite polynomial filters and σ is the standard deviation of the $G(x)$ used in defining the Hermite polynomials. Note that the higher order Hermite filter outputs are relatively small [65] , the above matrix A can be simplified to

$$A = \begin{bmatrix} \hat{I}_x & \hat{I}_y & 0 & 0 \\ \hat{I}_{xx} & \hat{I}_{xy} & \hat{I}_x & -\hat{I}_y \\ \hat{I}_{xy} & \hat{I}_{yy} & \hat{I}_y & \hat{I}_x \\ \hat{I}_{xxx} & \hat{I}_{xxy} & 2\hat{I}_{yy} & -2\hat{I}_{xy} \\ \hat{I}_{xxy} & \hat{I}_{xyy} & 2\hat{I}_{xy} & \hat{I}_{xx}-\hat{I}_{yy} \\ \hat{I}_{xyy} & \hat{I}_{yyy} & 2\hat{I}_{yy} & 2\hat{I}_{xy} \end{bmatrix}. \tag{47}$$

The above simplification is also supported by the fact that the higher order Gaussian derivatives are usually smaller and less accurate. In other words, such an approximation does not induce much error.

It is necessary that we derive the motion constraint equations up to the third order for the purpose of motion boundary extraction because we need to have more constraints than unknowns to obtain a least square formulation and compute the residual, E in (45). The

residual measures the amount of disagreement among the equations in a linear system. In other words, if these equations are derived from a mathematical model, then the residual reflects the deviation from the underlying assumptions of the model. In our case, the assumptions are brightness constancy and local coherent motion. In the following subsection, we analyze the relationship between the residual and the motion boundary.

4.3.2 Analytical relations between the residual and the motion boundary

The residual of our algorithm is $E = min\|As + b\|$. The residual error can result from the approximation errors of our computational model in describing the physical world. Specifically, these errors are:

1. The assumption of the motion model is violated in the local window
2. The assumption of constant image brightness is violated.
3. Quantization or truncation error.

The following figures show the typical effects and magnitudes of errors due to motion boundaries, brightness changes, and quantization in a small local neighborhood. Fig 33.1 shows the image at time 0. Fig 33.2.1 shows the same patch occluded by another patch. Fig 33.2.2 shows the patch undergoing random brightness change due to sensor noise. Fig 33.2.23 shows the intensity of the same patch quantized coarsely. Fig 33.3.1-Fig 33.3.3 show the magnitudes of the errors associated with these situations, where mid-gray represents 0, brighter levels mean positive and darker levels mean negative. It can be seen that motion boundaries usually induce the largest errors among the three.

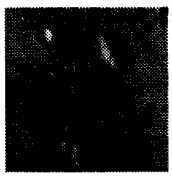


Fig 33.1 Original image

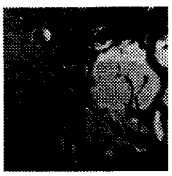


Fig 33.2.1 Boundary

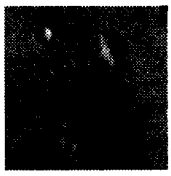


Fig 33.2.2 Brightness change

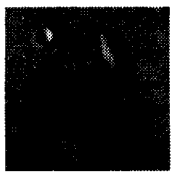


Fig 33.2.3 Quantization effect

Fig 33.3.1 Boundary error

Fig 33.3.2 Brightness change error

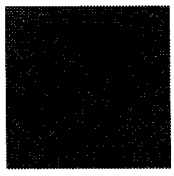


Fig 33.3.3 Quantization error

Note that the figures only show the errors in the image sequence. After we apply the differentiation filters, the errors are transformed into errors in the linear system.
Hence we can model the above errors as perturbations or noise to the linear system [58] :

$$\tilde{E} = min\|(A + N)\tilde{s} + (b + \Delta b)\| \quad \text{, where } N \text{ and } \Delta b \text{ denote errors.} \qquad (48)$$

We derive the analytical relationship between the residual and the errors as follows. Let

A and b, defined in (46), contain no noise. Then

$$E = As + b = 0 \text{ and } s = -(A^T A)^{-1} A^T b. \tag{49}$$

Let the noise contaminated optical flow be $\tilde{s}$ and the new residual be $\tilde{E}$, and assume that $N \ll A$ and $\Delta b \ll b$ elementwise, i.e., $NN^T = 0$ and $N\Delta b = 0$. Then

$$\tilde{s} = -[(A+N)^T(A+N)]^{-1}(A+N)^T(b+\Delta b), \text{ and} \tag{50}$$

$$[(A+N)^T(A+N)]^{-1} \approx (A^T A[I + (A^T A)^{-1}(A^T N + N^T A)])^{-1}$$

$$\approx [I - (A^T A)^{-1}(A^T N + N^T A)](A^T A)^{-1} \quad \text{, so} \tag{51}$$

$$\tilde{s} \approx -(A^T A)^{-1} A^T b + (A^T A)^{-1}(A^T N + N^T A)(A^T A)^{-1} A^T b - (A^T A)^{-1} N^T b - (A^T A)^{-1} A^T \Delta b$$

Using (49), this can be simplified as follows:

$$\tilde{s} \approx s - (A^T A)^{-1} A^T N s - (A^T A)^{-1} A^T \Delta b \text{ and } \Delta s \approx -(A^T A)^{-1} A^T N s - (A^T A)^{-1} A^T \Delta b.$$

For the residual, substituting $\tilde{s}$ into (48), and using (49), we derive

$$\tilde{E} \approx \|(A+N)s - A(A^T A)^{-1} A^T N s - A(A^T A)^{-1} A^T \Delta b + b + \Delta b\|$$

$$\approx \|(I - A(A^T A)^{-1} A^T)(Ns + \Delta b)\| \quad . \tag{52}$$

Further analysis shows that expression $I - A(A^T A)^{-1} A^T$, denoted by T, has only two nontrivial eigenvalues, both 1. We thus conclude that $\tilde{E}$ is proportional to the noise magnitude and noise orientation with respect to the matrix T. And since T is dependent on the image intensity pattern, which we cannot separate from the noise, we will only use the fact that residual error is proportional to the noise magnitude. But in order to use the residual to extract boundaries we still need to separate the residual error induced by motion boundaries from that by other sources. Therefore we analyze the residual profile in the spatial domain in the following subsection.

4.3.3 Residual profile

We now show that the residual profile across a motion boundary follows a specific pattern and is very different from the residual profiles arising from brightness changes or quantization errors. We can then use a spatial filter that matches this profile to extract motion boundaries.

Fig 34.1 shows a motion boundary neighborhood. A dotted square represents a local window used to estimate image derivatives and the residual for the center pixel. By sliding the window across the boundary, we can compute and plot the residual profile. A typical residual profile is shown in Fig 34.2. It has a big plateau centered on the motion boundary. The width of the plateau is about the same as the local window size. This is because only in that region does the local window cover the boundary.

Brightness changes and quantization errors, on the other hand, are usually scattered in

Fig 34.1 Sliding window across boundary

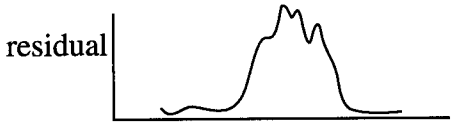


Fig 34.2 Typical residual profile across boundary

the image; we do not expect their residual profiles to look like the residual profiles due to motion boundaries. Also, since residuals arising from motion boundaries are larger than those arising from the other two sources, their profiles should be very prominent.

4.3.4 Motion boundary extraction based on residual profile

Based on the above findings, we can extract motion boundaries using two spatial filter (for different directions) designed according to Canny's criteria [22] for wide ridge edge detection. The maxima of the two responses are thresholded to form thick boundaries. On the thick boundaries, we perform a morphological medial axis operation or skeletonization*[35] to extract the center loci of the boundaries. Some simple pruning and contour following are then done to prevent streaking since the medial axis does not guarantee connectivity. Note that we do not use nonmaximum suppression and hysteresis as in [22] . This is because the residual is not proportional to noise† and there may be multiple peaks in the residual profile that will cause the maximum to drift away from the actual center of the ridge. We have verified in experiments that the medial axis provides better localization than nonmaximum suppression and hysteresis. The algorithm is summarized in Fig 35.

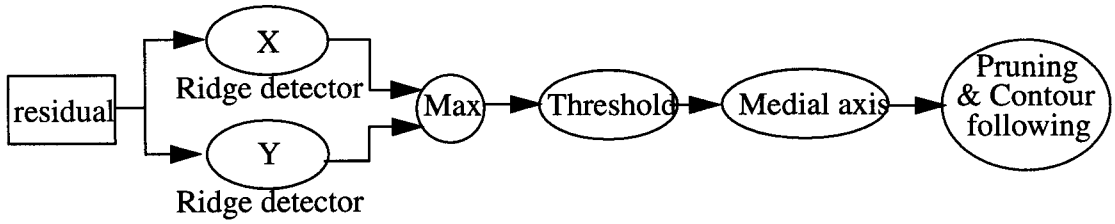


Fig 35. Summary of our boundary detection algorithm.

4.4 Evaluation and Experiments

It is very important to evaluate motion boundary extraction separately from optical flow. This makes clear what component of the motion estimation algorithm needs to be improved.

---

*. It is performed using Khoros 1.5 vmskel.

†. It depends on the noise vector direction (Section 4.3.2).

The evaluation of motion boundary extraction is similar to that of edge detection except that the ground truth is less clearly defined in the former case. For example, given only two successive images with motion boundaries (Fig 36.), the motion boundary can be

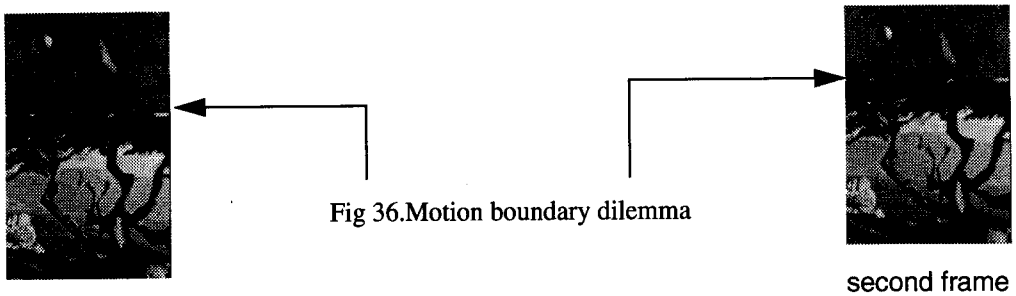


Fig 36.Motion boundary dilemma

designated at the location before or after the motion. We therefore use an odd number of frames to extract boundaries. The motion boundary will be defined as the location of the boundary in the center frame.

A good quantitative evaluation scheme for motion boundary extraction should account for the probabilities of detection and miss as well as the localization error. We reviewed several existing schemes and found that Heyden's method of evaluation [46] is best suited for motion boundary extraction purposes. It offers the following advantages over Abdou and Pratt's method[1] : first, it penalizes long streaking, i.e., large gaps of missed boundaries; second, it penalizes thick edges; third, there is no need to perform a search for correspondences between detected and ground truth motion boundaries. This evaluation scheme is sketched in Fig 37. It involves only binary image subtraction, Gaussian

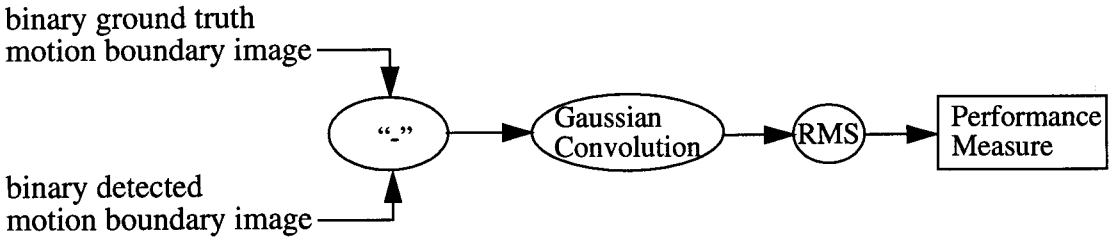


Fig 37. Heyden's quantitative evaluation scheme.

convolution, and computing the root mean square (RMS) of all pixels in the resulting image. The performance measure is the RMS of the Gaussian smoothed difference image. The Gaussian convolution is actually the crucial step that results in the above advantages.

Note that in this scheme, a better algorithm will yield a smaller output quantity, with zero as its minimum.

In order to make comparisons, we also implemented algorithms developed by Schunck [93] and Thompson et al. [106] . In implementing Schunck's algorithm, we used 3-D Hermite polynomial filters to compute first order derivatives and perform constraint line clustering to estimate optical flow. In fact, we had originally used Gaussian smoothing and Sobel operators to compute the derivatives and found the results too noisy to use. The Canny edge detector is applied to the flow components to find motion boundaries. In implementing Thompson's algorithm, we used Lucas & Kanade's algorithm imple-

mented by Barron et al. [9] . The initial flow output is not dense, so we implemented a propagation and smoothing technique to fill the field. After the flow is estimated, we use the vector field discontinuity detector suggested in [106] , using direction reversal of the Laplacian response of the flow vector field, to locate motion boundaries.
The first image we used is shown in Fig 38.1. It is a sequence composed of a baby face

Fig 38.1 Moving face on random dots

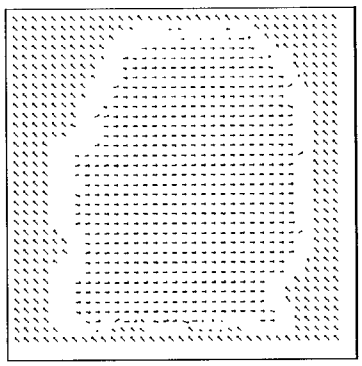


Fig 38.2 Approximate flow field

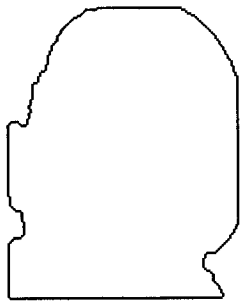


Fig 38.3 Motion boundary

traversing laterally in front of a moving random dot background. The approximate flow map and the motion boundary ground truth are shown in Fig 38.2 and Fig 38.3, respectively. This image sequence is synthesized so as to contain curved motion boundaries, which are common to real world scenes but present difficulties for most motion boundary extraction algorithms. This is because motion boundaries are often wider than intensity edges due to the nature of the motion estimation algorithm and it is very difficult to capture wide as well as high curvature features.
In Fig 38.1-Fig 38.3, we show our algorithm's residual map and Schunck's and Thompson's flow fields. They represent the bases upon which these algorithms extract boundaries. Thompson's flow field (Fig 39.3) is smooth across boundaries as expected, while

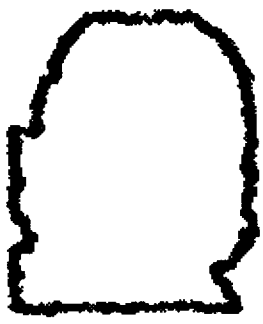


Fig 39.1 Residual map

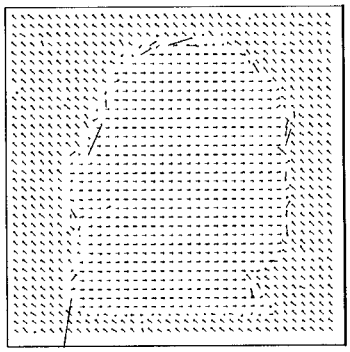


Fig 39.2 Schunck's flow field

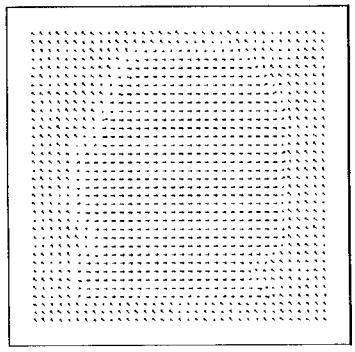


Fig 39.3 Thompson's flow field

Schunck's flow field (Fig 39.2) is noisier right on boundaries but more accurate near boundaries.
Next we show the detected and true motion boundaries for the three algorithms. In Fig 40.1-Fig 40.3, the dark edge represents the true motion boundary while the white edge represents the boundary detected. Note that when the true boundary is detected, the

color of the edge becomes gray. These images are obtained by subtracting the ground truth boundary image from the detected boundary image as dictated by Heyden's evaluation scheme.

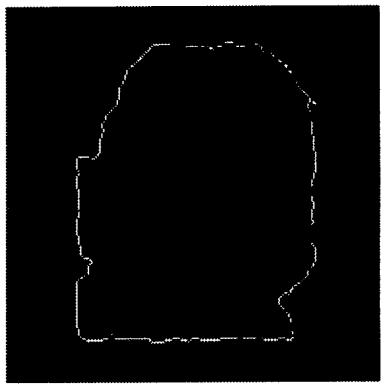


Fig 40.1 Our algorithm's boundary

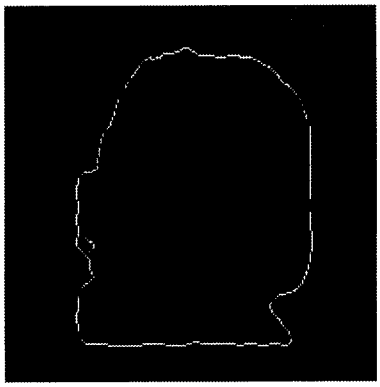


Fig 40.2 Schunck's boundary

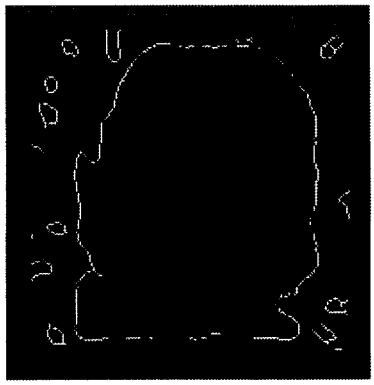


Fig 40.3 Thompson's boundary

In Fig 40.2, it can be seen that Schunck's algorithm suffers from boundary drift caused by noise on the boundary as well as localization errors in the corners, as mentioned in [93] . On the other hand, when the motion boundary is a straight line, Schunck's algorithm performs better than the other two. In Fig 40.3, it can be seen that Thompson's algorithm suffers from flow noise away from boundaries. Since it uses a direction reversal technique similar to zero crossings, spurious edges are detected. Otherwise, the localization is very good. Our algorithm's boundary is better at corners and essentially free of the major problems of the other two. The following table summarizes the quantitative performance measure computed by Heyden's evaluation scheme. It can be seen that our algorithm is better than the other two.

Table 8: Evaluation of the motion boundary extraction algorithms

Comparison Algorithms	**Our Algorithm**	**Schunck's Algorithm**	**Thompson et al.'s Algorithm**
Performance measure	5.85	7.86	10.32

The next image we use is the Yosemite fly-by sequence shown in Fig 41.1. This is a synthesized sequence in which the observer is approaching the scene and motion boundaries exist between objects of different depths. As can be seen by the flow field (Fig 41.2), two prominent motion boundary curves exist. One separates the sky from the mountains, and the other separates the domed mountain in the lower left corner from the other mountains. The boundary ground truth is not available. In Fig 42.1-Fig 42.3, we show the results of the three boundary extraction algorithms overlaid on the original image. The white edge points represent the extracted boundaries.

Note that the boundaries that separate sky and mountains are easier to extract because the motion directions are different on the two sides. All three algorithms indeed extract these boundaries. However, the other boundary that separates the domed mountain from the other background mountains is not as easy to extract because the motions on the two sides are in the same direction but have different magnitudes. Note that this kind of

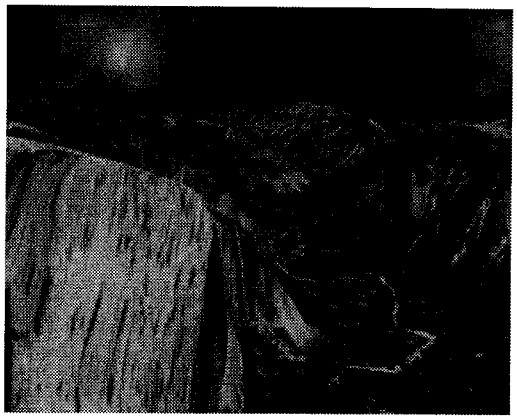


Fig 41.1 Yosemite fly-by

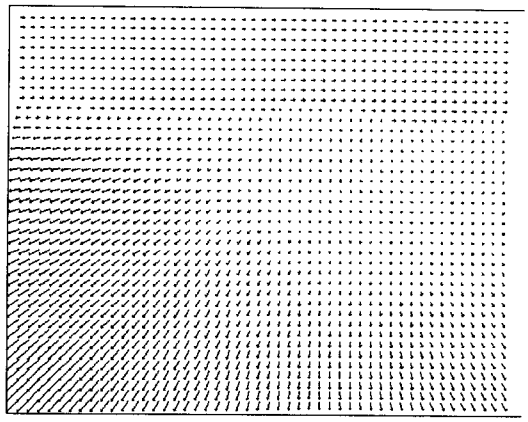


Fig 41.2 Yosemite fly-by flow field

motion field is typical in the image sequences captured by a forward moving observer. In Fig 42.2 Schunck's algorithm fails to extract these boundaries because the noise on both sides overwhelms the small variation in flow. In Fig 42.3 Thompson's algorithm fails to extract these boundaries because the presmoothing and filling of the sparse field smooths out the small flow variation. On the other hand, our algorithm extracts a large part of this boundary curve (Fig 42.1).

The motion-model-based method used here offers the capability of segmenting moving objects with different flows, divergences, or curls. The Yosemite fly-by sequence, for example, contains different divergences. The residual values indeed account for incoherence of the above three motion parameters in the local window. This is why our algorithm is capable of extracting these boundaries.

4.5 Conclusion

Motion boundary extraction algorithms are as important as motion estimation algorithms for the complete motion recovery problem. However, their interdependency poses a computational dilemma that renders any partial solution inaccurate. Indeed, the only way to solve the motion recovery problem is to simultaneously address both motion segmentation and estimation. While recent research has focused on iterative methods, we propose a method based on a general motion model. This method is local, non-iterative, and simultaneously deals with both motion estimation and boundary extraction.

The motion-model-based approach fits the local 3-D image pattern to a motion model and outputs a boundary likelihood measure, the residual, which may be used to extract motion boundaries. Compared with motion boundary extraction from flow, it offers several advantages: first, it is a scalar and thus avoids handling vector field discontinuities; second, the residual is less noisy on the boundary than the flow; third, the residual corresponds to true 3-D motion discontinuities instead of high variations caused by flow field nonuniformity within an object. In fact, the residual accounts for discontinuities in flow, divergence, and curl.

The evaluation of motion boundary extraction should be separated from the evaluation of optical flow to truly understand the performance of the individual motion algorithm

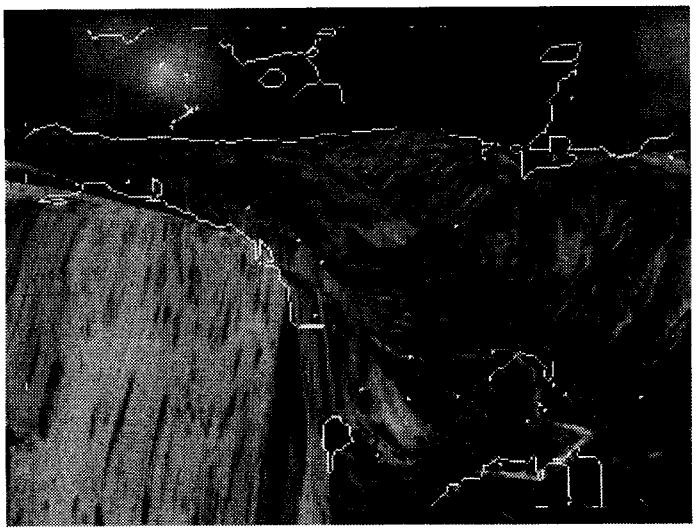


Fig 42.1 Our algorithm's result

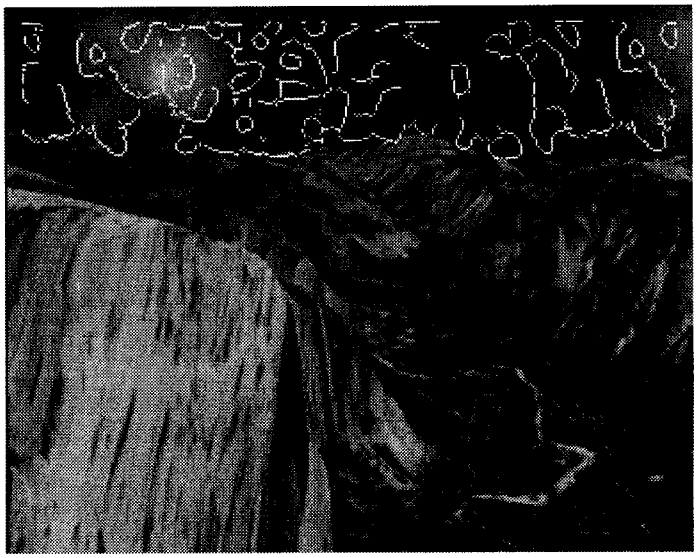


Fig 42.2 Schunck's result

Fig 42.3 Thompson's result

components. We employed a simple but elegant evaluation scheme and synthesized a

difficult motion sequence for comparison. We then demonstrated that our algorithm performs better than existing algorithms.

In the future, we will integrate the motion estimation algorithm described in [63] and the boundary extraction algorithm developed here. This will involve not only updating the flow around the boundaries but also finding important motion information from boundary properties such as the local relative depth. Other future topics include finding occluding surfaces, and global topological sorting of moving objects. These are important for obstacle avoidance and navigation.

Chapter 5

Image Gradient Evolution— A Visual Cue for Threat

This chapter is concerned with the task of visual motion-based navigation. A critical requirement of the task is the ability to estimate 3-D depth and motion from visual information. Recent studies have demonstrated that the relevant cues consist in motion parallax or optical flow and that flow field divergence and hence time-to-contact can be extracted. We present a new concept called image gradient evolution, which utilizes the change of image spatial gradients over time as a threat cue: an approaching object induces 2-D expanding motion and causes the image spatial structure to stretch so the image gradients decrease. Based on this idea, our method offers a one-step solution directly from image gradients, instead of from optical flow and its derived properties. We use a technique that is local and linear so the implementation can be very fast. The threat map is expectedly noisy but sufficiently informative, as is seen in demonstrations on several real images. These two aspects, fast implementation and useful qualitative information, provide a viable solution to navigational tasks.

5.1 Motivation

We start with a simple illustration to introduce the idea of image gradient evolution and emphasize its difference from the optical flow approach.

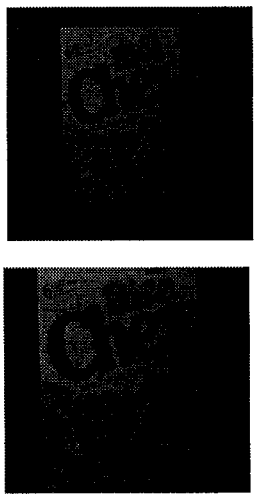


Fig 43.1 Approaching box

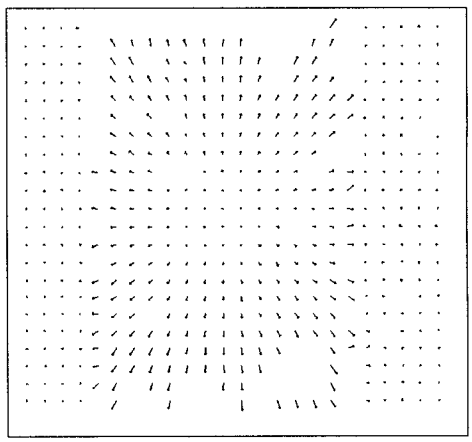


Fig 43.2 Flow field

Fig 43.1 shows a diverging object in the image. Our goal is to obtain an algorithm that

can warn the observer of a potential collision.
Conventionally, optical flow (Fig 43.2) is computed first and then the first order flow field properties (diverging or converging) is used to characterize the underlying objects' 3-D motion. In our approach, however, the change of the image gradients over time is computed. As illustrated in Fig 44, a decreasing image intensity gradient(slope) at the

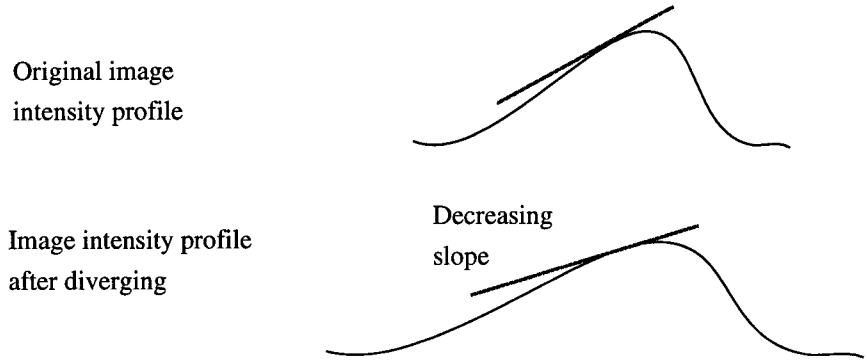


Fig 44 1-D image gradient evolution

corresponding point signifies a diverging object. As shown in Fig 45, we wish to avoid processing the noisy flow data when it is evident that we can achieve the same result from image gradient evolution.

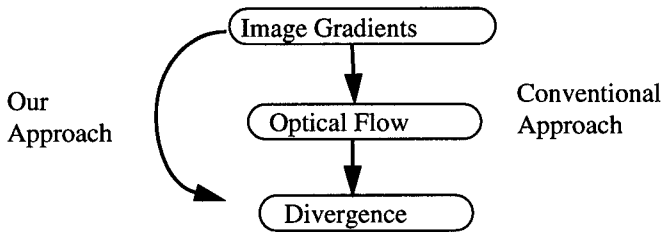


Fig 45 Different approaches to divergence

The following sections of this chapter are organized as follows. Section 5.2 discusses our work in the context of previous research. The image gradient evolution is formulated in the context of a generalized motion model in Section 5.3. Section 5.4 details our filtering scheme and implementation issues. Section 5.5 demonstrates experimental results on real images. Section 5.6 concludes the chapter.

5.2 Previous Work

The looming effect is a major cue in biological vision systems used to sense danger (Schiff, et al.[93]). Local motion parallax is in turn a cue for looming or divergence (Werkhoven & Koenderink[116]). Since time-to-contact is related to both divergence and 3-D scene structure, to solve for time-to-contact is, in an exact sense, equivalent to recovering 3-D motion and structures. There is a plethora of literature about recovering 3-D motion and structure from optical flow (Adiv[3] , Bruss & Horn[14] , Negahdaripour & Lee[82]), image sequence (Broida & Chellappa[13]), or features (Negahdaripour & Horn[81] , Tsai & Huang[108]). It is basically an ill-posed and nonlinear problem. These

methods usually use iterative optimization techniques, which is often time-consuming. However, these reports established the feasibility of imposing extra constraints and/or using derivatives of flow to solve the problem in theory. But the intended precision of these quantitative methods is often an illusion due to the limit on the accuracy with which the input measures can be obtained (Thompson & Kearney[107]). Accurate optical flow estimation is still a difficult problem.

The avoidance of differentiating optical flow has been approached in different ways. Integration theorems such as Green's theorem (Poggio, et al.[88] , Cipolla, et al.[24]), Stoke theorem [88] , and Gauss theorem (Gupta[40]) are used to estimate first order flow parameters directly from image intensity integrals[40] , image moments (and their temporal derivatives) [24] , or flow integrals[88] . The integration techniques basically trade off noise sensitivity for smoothness, which arises from a single motion assumption within the integration contour. To prevent the integration contour from going across boundaries, it requires additional mechanisms, probably global, to segment images.

Another approach models the local motion with an affine model. It uses higher order pointwise image derivatives (Nagel[77] , Werkhoven & Koenderink[116]) or patchwise motion coherence (Campani & Verri [19] , Bergen, et al.[10]) techniques to solve for first order motion parameters. This approach is actually aimed at accurate flow estimation and the reports make little mention of 3-D motion estimation.

Since the above two approaches do not model the 3-D structure, they do not offer sufficient information to solve for time-to-contact without additional constraints or differentiations in the general case. It has been proved that only the upper and lower bounds on time-to-contact can be derived (Subbarao[103] , Cipolla, et al.[24]). Meyer and Bouthemy[59] used temporal derivatives on the first order parameters to circumvent the problem. Essentially this is equivalent to using second order derivatives, but Kalman filtering on the temporal derivatives makes the result much smoother and more practical. However, a fast implementation is relatively difficult.

Nelson & Aloimonos[79] were the first to attempt a realistic approach to navigation. Their algorithm computes directional divergence, which is a second order flow parameter, and can be very noisy. Coombs, et al.[25] employed flow divergence for real-time obstacle avoidance. Their obstacle avoidance system currently appears to be the fastest one using flow divergence. In their system, time-to-contact is equivalent to divergence when carefully controlling the camera so as to approximately translate in the direction of the optical axis. The computation of divergence, however, is based on noisy flow and requires temporal integration in order to interpret the result. Camus[20] implemented a real-time algorithm for time-to-contact which is quite reliable. However, the algorithm is limited by its restrictive assumptions about the motion and the surfaces. Kundur & Raviv[59] proposed the use of image quality measure for the visual threat cue. This method exploits the camera defocusing effect for navigation.

The major contribution of this chapter is to pioneer the idea of image gradient evolution and use it as a cue for threat during navigation. Such a capability is embedded in a generalized 2-D motion equation that also models expansion. An integrated spatio-temporal filtering scheme is designed to estimate image derivatives in a numerically coherent manner. From these derivatives, image gradient evolution and optical flow can be estimated at the same time in a fast manner.

5.3 Generalized Motion Model

The brightness constancy equation is often interpreted in the following way.

$$\nabla I = 0 \Rightarrow I(x, y, t) = F(x-ut, y-vt) \tag{53}$$

To measure image gradient evolution, the first step is to extend the image motion model from simple 2-D translation to translation plus expansion. A 3-D point at position $P = (X, Y, Z)$, under perspective projection, projects to a point in the 2-D image plane, (x, y),

$$\begin{aligned} x &= fX/Z \\ y &= fY/Z \end{aligned}, \text{ where } f \text{ is the focal length of the projection.} \tag{54}$$

Let there be relative 3-D translational motion $P(t) = (X + U_X t, Y + U_Y t, Z - U_Z t)$. Hence,

$$\begin{aligned} x(t) &= f(X + U_X t)/(Z - U_Z t) \\ y(t) &= f(Y + U_Y t)/(Z - U_Z t) \end{aligned}. \tag{55}$$

Brightness constancy and (55) yield

$$I(x, y, t) = F\left(x\left(1 - \frac{U_Z}{Z}t\right) - \frac{fU_X}{Z}t,\ y\left(1 - \frac{U_Z}{Z}t\right) - \frac{fU_Y}{Z}t\right). \tag{56}$$

3-D translation only is assumed for its simplicity since 3-D rotation has no bearing on expansion (Koenderink[58]). A general motion model that includes rotation parameters can be found in (Liu, et al.[63]). The following derivations can also be derived from the general motion model.

To understand the generalized motion equation better, we describe equation (56) in the context of optical flow.

$$\begin{aligned} (u, v) &= \left(\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}\right) = \\ &\left(\frac{fU_X}{Z - U_Z t} + \frac{U_Z x}{Z - U_Z t}, \frac{fU_Y}{Z - U_Z t} + \frac{U_Z y}{Z - U_Z t}\right) \approx \\ &\left(\frac{fU_X}{Z} + \frac{U_Z x}{Z}, \frac{fU_Y}{Z} + \frac{U_Z y}{Z}\right) \end{aligned} \tag{57}$$

Let $\frac{U_Z}{Z}$ be denoted by s, and $\left(\frac{fU_X}{Z}, \frac{fU_Y}{Z}\right)$ by (p, q).

Rewriting (56) and (57) as

$$I(x, y, t) = F(x(1 - st) - pt, y(1 - st) - qt) \tag{58}$$

$$(u, v) = (p + sx, q + sy) \tag{59}$$

Equation (59) lays out the two components of optical flow: (p, q), and (sx, sy). From the linear dependency of (sx, sy) on (x, y), s is interpreted as expansion. The translation component (p, q) is induced by (U_X, U_Y) only, and the expansion component s by U_Z only. When $s = 0$, then $(u, v) = (p, q)$ and (58) becomes (53).

The image gradient evolution is characterized by the following equation (derived in Appendix C),

$$\frac{\partial}{\partial x}I(x, y, t)=(1-st)\frac{\partial}{\partial x}I(x', y', 0), \begin{cases} x' = x(1-st)-pt \\ y' = y(1-st)-qt \end{cases} \tag{60}$$

An image point (x', y') at time 0 moves to (x, y) at time t. The two image gradients are related by (60). When s is positive, which means the object is approaching ($U_Z > 0$, in (56)), the slope is decreasing ($1 - st < 1$). This coincides with our previous observation in Fig 44. In the extreme case, when $st = 1$, we derive the following equation

$$t = \frac{1}{s} = \frac{Z}{U_Z} \text{ and } \frac{\partial}{\partial x}I(x, y, t) = 0. \tag{61}$$

In equation (61), t is interpreted as the time-of-contact and at that instant, the entire image texture disappears, which is what we observe when the object is too close. It is clear that s measures not only 2-D expansion, but also image gradient evolution and time-to-contact. We can use it as a cue for collision avoidance.

Note that the focus of expansion (FOE) is predefined to be at $(0, 0)$ in (56). To complete the formulation, we modify (58) and (59) to allow the FOE to be at an arbitrary location (x_0, y_0).

$$I(x, y, t) = F((x - x_0)(1 - st) - pt + x_0, (y - y_0)(1 - st) - qt + y_0) \tag{62}$$

$$(u, v) = (p + s(x - x_0), q + s(y - y_0)). \tag{63}$$

Note that equation (60) is derived based on the assumption of a parallel frontal surface, i.e., the surface normal is parallel to the optical axis. When the surface is not parallel frontal, the image gradient evolution cannot be reliably interpreted for 3-D motion. However, to use it as a qualitative cue for threat, our algorithm remedies this problem by identifying other types of surfaces and potential boundaries as outliers. A post-smoothing stage then overwhelms the errors induced by the outliers. This technique is reasonable because "divergence due to a relatively distant object can be large, but only over a short distance in the image" (Nelson & Aloimonos[79]).

Compared with other approaches that attempt to compute absolute time-to-contact (Burlina & Chellappa [15]), our method appears inexact. In fact, we use a simplified model of the affine motion with reasonable assumptions. However, it is a sound approach as we realize that humans can navigate in complex environments without estimating time-to-contact exactly. In addition, it is important to navigate in an unknown environment where the scene and objects change dynamically, therefore there is little point in spending time computing exact time-to-contact for the underlying scene [107] . For navigational tasks, a practical approach is to build a "qualitative" threat map quickly instead of an accurate time-to-contact map slowly.

5.4 Algorithm and Implementation

To facilitate the estimation of image gradient evolution in the framework of the general motion model, a potent and stable image differentiation filtering scheme is needed. The set of orthogonal 3-D Hermite polynomial filters is excellent for the task.

$$I_{ij1} \approx -uI_{(i+1)j0} - vI_{i(j+1)0} - (i+j)sI_{ij0} \text{ where } (u, v) \text{ are defined in (63).} \quad (64)$$

Equation (64) is linear in terms of optical flow (u, v) and image gradient evolution s. Using 00, 01, 10 for ij, we can derive three equations up to the second order and solve the linear system. In our implementation, we use six equations up to the third order to form a least square formulation. The reason is that the residual is an excellent reliability measure. In fact, if we consider the residual as the extent to which the motion model is violated, it can be used to indicate noise, non-frontal surfaces, and boundaries (Liu, et al.[64]). Since s is noisy, our implementation exploits smoothing with confidence weighting, i.e., extra steps of smoothing on s with the reciprocal of the residual as weighting. This will smooth out noise but prevent smoothing across boundaries. It will also overwhelm the errors due to non-frontal surfaces as long as there is a portion within the object that is parallel frontal or nearly parallel frontal. If the objects' surfaces are nowhere frontal, we shall direct the robot to look and move in the same direction in a piecewise manner using pan/tilt camera control, thus eliminating lateral motion that may confuse the algorithm.

5.5 Experiments

The flow portion of the algorithm has been shown to be very accurate (Liu, et al.[63]) compared with other current algorithms[9] . The following figures(Fig 46-Fig 48) show one image of the sequence and its 3-D perspective threat map based on image gradient evolution s. In the 3-D threat maps, elevation is used to depict threat. So the elevated area represents closer objects. The threat value is thresholded to enhance 3-D structure perception. For example, in Fig 46, the cereal box stands out in the center because most of the threat values within the box are above the threshold; a navigational algorithm can then use this threat map and veer aside to avoid the dangerous area in the center. In Fig 47, the threat map is gradually elevated from the valley to the mountain in the lower left corner. And the sky is perceived as safe except in some areas where the clouds change brightness irregularly and deceive the algorithm; a navigational algorithm can use this threat map to avoid heading for the lower left corner. In Fig 48, the metal plate in the lower left corner is extracted; the Coke can and the platform are also partially visible; the pole on the right side is visible at both of its ends. The fact that the metal plate is detected and the hole remains intact demonstrates the effect of smoothing with confidence weighting. We are currently working on the real-time implementation of the algorithm. On 64x64 images, image gradient evolution without smoothing can be expected at the rate of 3 to 4 frames per second on a Hyper Sparc 10 MP board. The amount of smoothing is dependent on the image noise and may take a little more time than required by computing image gradient

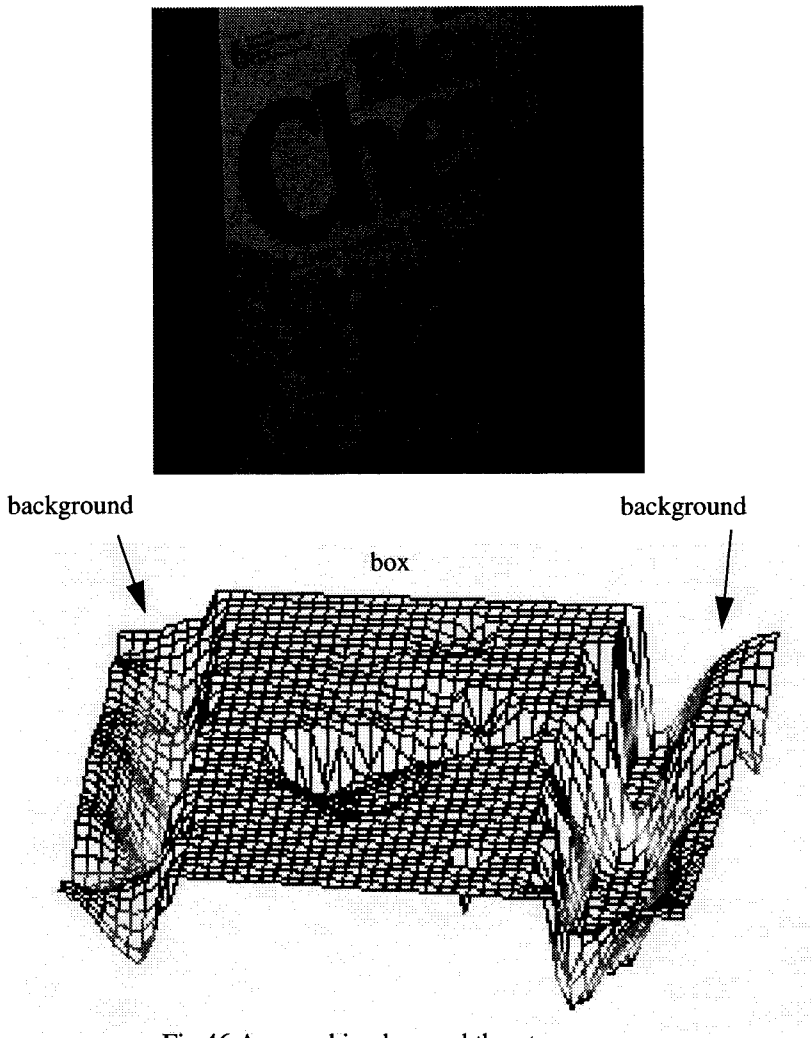
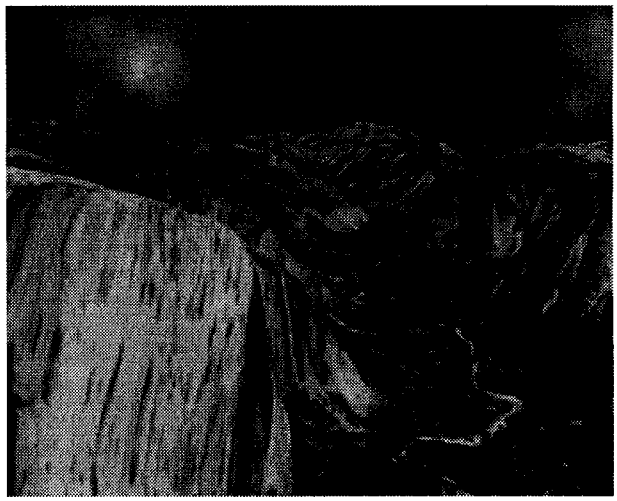


Fig 46 Approaching box and threat map

evolution. Although the threat map is noisy and currently the resolution is limited by the noise and irregular brightness change, it already provides useful information for navigation.

5.6 Conclusion

Image gradient evolution is shown to be a useful cue for threat. The use of the image gradient evolution eliminates the need to process noisy optical flow. Our algorithm builds a dense qualitative threat map based on image gradient evolution. For navigation, we would rather compute useful, probably inexact, information quickly than exact information slowly. That is what our algorithm achieves.



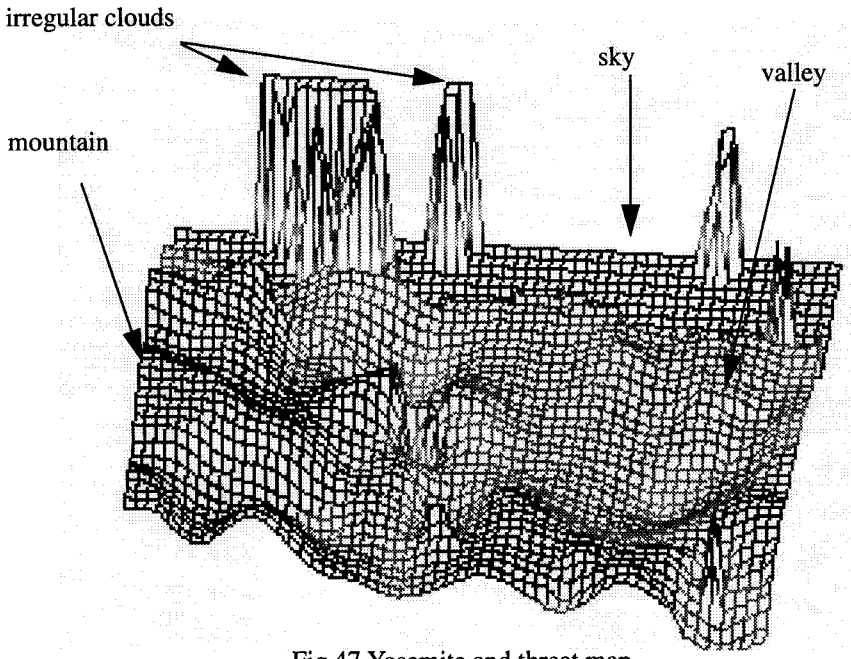


Fig 47 Yosemite and threat map

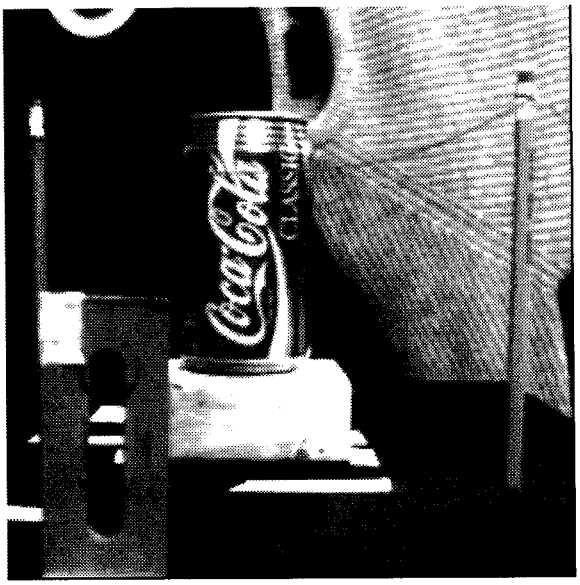


Fig 48 NASA sequence and threat map

Chapter 6
Transparent Motion Segmentation

An image is ideally a projection of the 3-D scene. However, the imaging process is always imperfect and constrained by the physical environment, for example, viewing through a window with reflections. This chapter is concerned with image sequences acquired in such situations, the so-called transparency. When this occurs, the image sequence contains undesirable transparent motion, for example, of the window reflections. This complicates the already difficult motion estimation problem. We present an algorithm to segment transparent motion based on a spatio-temporal filtering technique—3-D Hermite polynomial differentiation filters. With motion segmentation accomplished, we can then focus on the analysis of the scene. The implementation of our algorithm is fast and accurate.

6.1 Introduction

Transparent motion refers to two superimposed intensity patterns with different motions present in a single image. The resulting image can be hypothesized to be either the addition or multiplication of the two moving patterns depending on the types of transparency. Specular and diaphanous transparencies are additive; film and shadow transparencies are multiplicative [55] .
Fig 49.1-Fig 49.3 give an example of specular transparent motion. Since the moon is rotating, the surface is translating (to the right in Fig 49.3). The moon also revolves around the earth causing the specular reflection of the sun to translate (to the upper left) at the same time.
An image I with transparent motion is formulated as follows:

$$I(x, y, t) = I_1(x, y, t) \oplus I_2(x, y, t), \tag{65}$$

where $\oplus$ can be addition or multiplication, and

$$I_i(x, y, t) = I_i(x - u_i t, y - v_i t, 0), \; i = 1, 2, \tag{66}$$

using the translational motion model.
It can easily be seen that any single motion estimation algorithm would fail to compute either one of the motions unless there is one dominant texture. Transparent motion segmentation needs to be performed before an analysis of the individual motions is possible. Although it may seem that the transparent motion scenario is rarely present, it is actually very common in the real world. The specular reflection depicted in Fig 49.2 is a good example. Also common is the dirty lens scenario where the dirt pattern on the lens is motionless but contaminates the intensity pattern of the scene.
Transparent motion segmentation is difficult to solve. However, it is encouraging to note

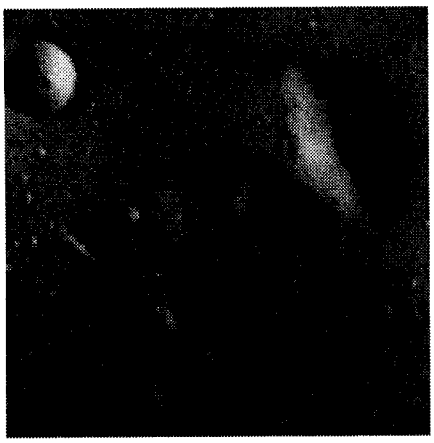


Fig 49.1 Diffuse reflection

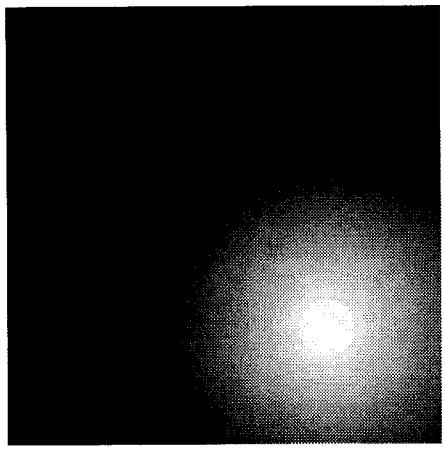


Fig 49.2 Specular reflection

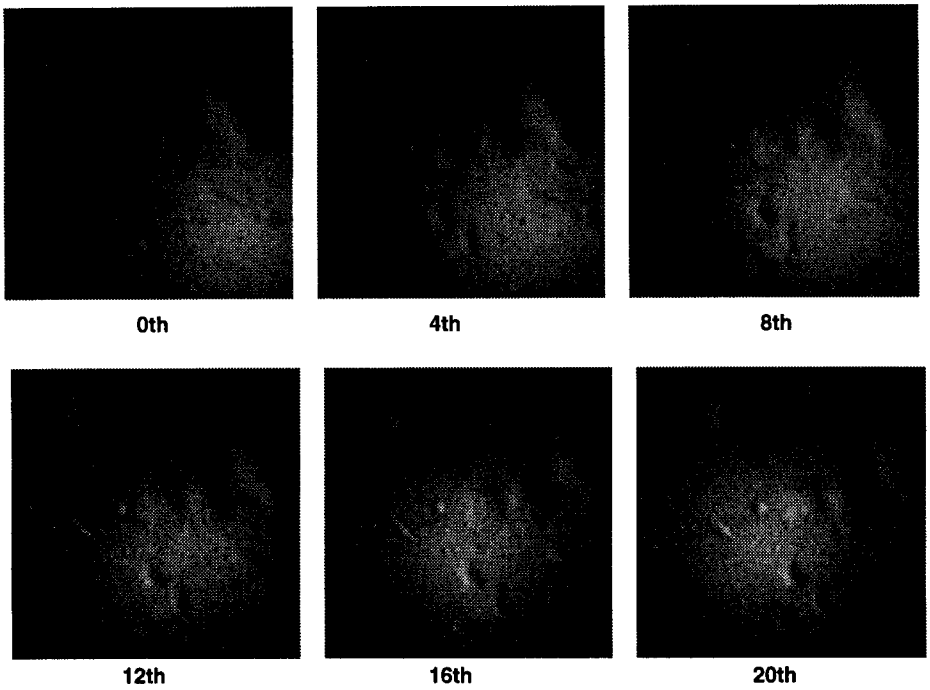


Fig 49.3 Sequential display of image frames

that when humans view images with transparent motion, they have no problem segmenting different motions.

There are three steps involved in transparent motion segmentation:

1. Determining whether multiple motions exist.
2. Extracting transparent motions.
3. Segmenting transparent motions.

Unlike most previous studies which are primarily focused on 2. and sometimes 1., this chapter address each of the three steps.

6.2 Previous Work

In this chapter, we expand on the multiple motion model proposed by Shizawa and Maze[95] [96] . Their basic idea is to apply consecutive linear operators, which are equivalent to single motion constraint equations. For two motions, it is formulated as:

$$[(u_1, v_1, 1) \bullet \nabla][(u_2, v_2, 1) \bullet \nabla]I = 0 \tag{67}$$

where $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial t}\right)$ and $\bullet$ is the inner product.

With this model, Langley, et al. [60] used a phase-based method to extract multiple motion.

Our expansion of the model is based on the gradient constancy assumption, which is true when brightness constancy is true. It is formulated as:

$$[(u_1, v_1, 1) \bullet \nabla^i][(u_2, v_2, 1) \bullet \nabla^j]I = 0 \tag{68}$$

where $\nabla^i = \left(\frac{\partial^i}{\partial x^i}, \frac{\partial^i}{\partial y^i}, \frac{\partial^i}{\partial t^i}\right)$.

With different combinations of the i and j, we get multiple motion constraint equations to solve the problem.

We adopt the conventional gradient-based method instead of using frequency information because our previous work [63] [66] has shown that spatio-temporal filtering based on 3-D Hermite polynomial differentiation filters is excellent in estimating image gradients. The separability of the filters offers a tremendous speed advantage over the frequency domain filters of Langley, et al. and Shizawa & Mase.

Bergen et al. [10] proposed an iterative scheme and image registration to minimize their multiple motions estimation error. Darrell and Pentland [26] used robust estimation techniques and a layered representation of the images to segment multiple flows. Although these algorithms are better at handling occluding boundaries, their iterative nature does not guarantee convergence to a global minimum and their computational cost is very high.

6.3 The Algorithm

To facilitate the estimation of accurate image gradients in the framework of the expanded transparent motion model depicted in (68), an accurate and stable image differentiation filtering scheme is needed. The set of orthogonal 3-D Hermite polynomial filters is excellent for the task.

6.3.1 Extracting transparent motions

If the transparency is multiplicative, we simply take the logarithm of the image intensity and follow the same procedure as for additive transparency.

From equation (68), we derive enough equations to solve for the unknowns (x in (70)). The unknowns are nonlinear functions of (u_1, v_1) and (u_2, v_2). But we solve the linear system for the unknowns first and use the relations (x in (70)) to solve for the flows. For double transparent motion, we have

$$min\|Ax - b\| \text{ where} \tag{69}$$

$$A = \begin{bmatrix} I_{xx} & I_{xy} & I_{yy} & I_{xt} & I_{yt} \\ I_{xxx} & I_{xxy} & I_{xyy} & I_{xxt} & I_{xyt} \\ I_{xxy} & I_{xyy} & I_{yyy} & I_{xyt} & I_{yyt} \\ I_{xxxx} & I_{xxxy} & I_{xxyy} & I_{xxxt} & I_{xxyt} \\ I_{xxxy} & I_{xxyy} & I_{xyyy} & I_{xxyt} & I_{xyyt} \\ I_{xxyy} & I_{xyyy} & I_{yyyy} & I_{xyyt} & I_{yyyt} \end{bmatrix}, b = \begin{bmatrix} I_{tt} \\ I_{xtt} \\ I_{ytt} \\ I_{xxtt} \\ I_{xytt} \\ I_{yytt} \end{bmatrix} \text{ and } x = \begin{bmatrix} u_1 u_2 \\ u_1 v_2 + u_2 v_1 \\ v_1 v_2 \\ u_1 + u_2 \\ v_1 + v_2 \end{bmatrix}. \tag{70}$$

Once the value of x is computed from the least square system, (u_1, v_1) and (u_2, v_2) can be easily computed and properly aligned according to [95] .

6.3.2 Segmenting transparent motions

Now that every image point has two flow vectors, we need to separate them into two coherent flow fields. For this, we assume that the foreground flow is constant. Thus we compute a 2-D image flow histogram and find a peak in the histogram, which corresponds to the constant motion. Then at every point, the flow closer to this peak is assigned to the foreground motion, while the other flow is assigned to the background motion. This assumption is often true when the foreground motion is induced by window reflections or dirt patterns on the lens.

6.4 Experiments

We have tested our algorithm on a synthesized image sequence (Fig 49) and on real image sequence (Fig 52). The 3-D window size used in both instances is 25x25x17.
In Fig 50, we show the true and computed flow for the moon surface and the specular reflection. As can be seen, the computed background motion corresponding to the moon surface is very accurate. The computed specular motion is less accurate because the specular component has significantly less texture. This is the general aperture problem present in motion algorithms. In the extreme case where one moving pattern has no texture, only the other motion can be detected.
We then separate the two intensity patterns by the following technique. In Fig 51.1, we warp the first frame with flows from the flow field associated with specular reflection motion and then subtract the second frame, resulting in the cancellation of specular reflection pattern. The output is the temporal difference of the moon surface pattern. Fig 51.2 is derived conversely.

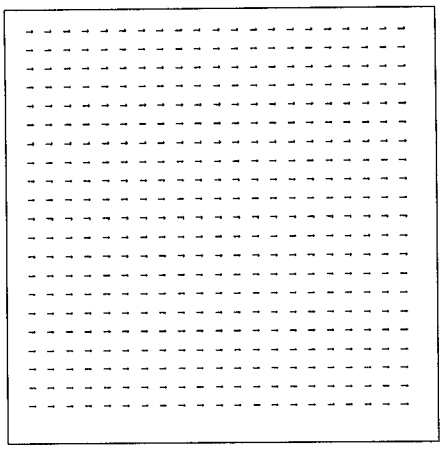


Fig 50.1 True moon surface flow

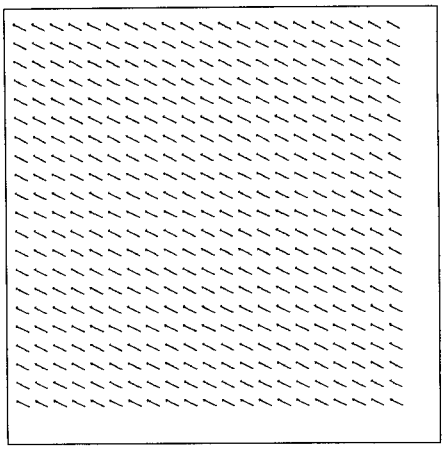


Fig 50.2 True specular flow

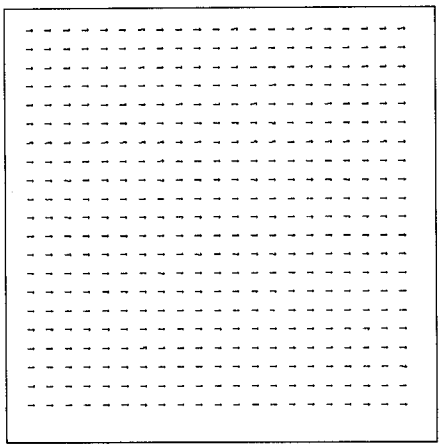


Fig 50.3 Computed moon surface flow

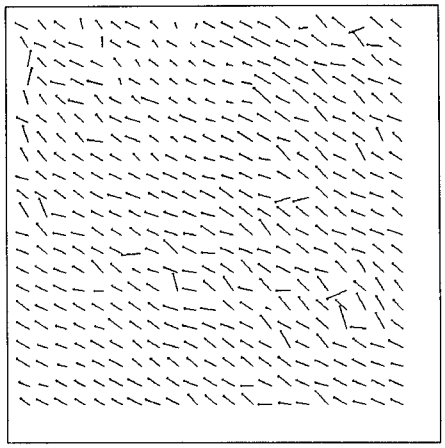


Fig 50.4 Computed specular flow

Fig 51.1 Temporal difference of the moon surface pattern

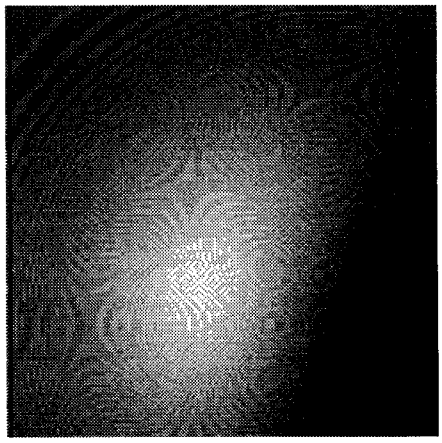


Fig 51.2 Temporal difference of the specular reflection pattern

The real image sequence used in our experiments is presented in Fig 52. In this sequence, the scene is composed of a picture in a frame which reflects the ceiling light. Note that

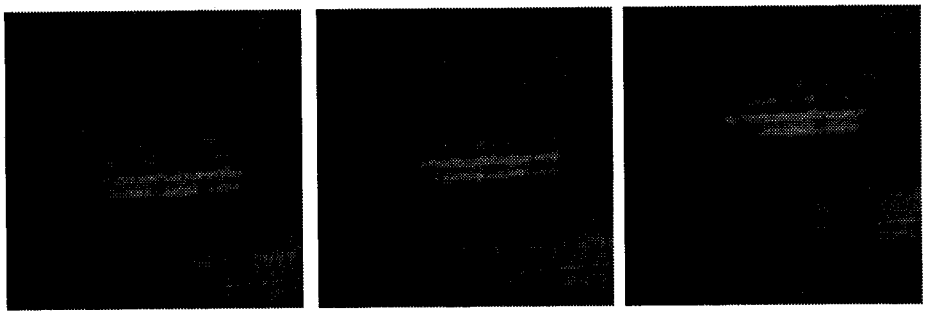


Fig 52.Sequential display of real images

the picture is moving to the upper right corner and the reflection is moving upward. From the results in Fig 53.1, we see that the motion of the picture is extracted accurately

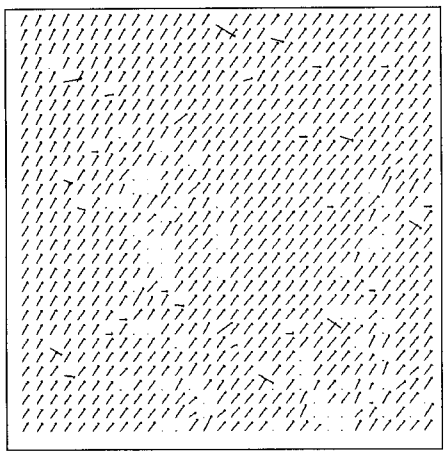


Fig 53.1 Computed picture flow

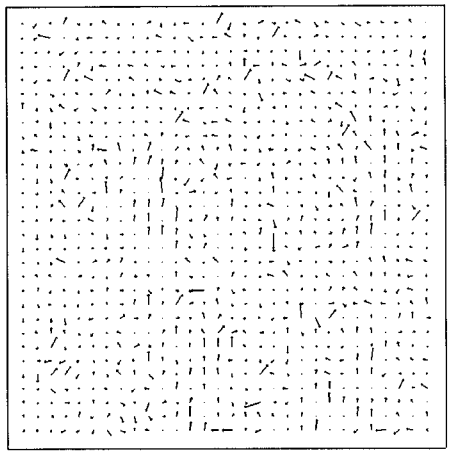


Fig 53.2 Computed reflection flow

in most areas. However, the motion of the light reflection is noisy because of the lack of texture in this pattern. Nonetheless, a close inspection of the flow field in Fig 53.2 reveals that it is indeed moving upward. It should be noted that the object motion, rather than reflection motion, is usually the desirable information to extract and the accuracy associated with the former is more important

Our transparent motion segmentation algorithm is implemented on a Sun Hyper Sparc 10 board. For the window size specified above, it ran on a 150x150 image sequence in less than 1 minute. The code is available through our ftp site at giskard.cme.nist.gov, in directory ftp/pub/motion/transparent.

6.5 Conclusion

Transparent motion segmentation has received relatively little attention due to its numerical demands. The same reason appears to have led recent approaches away from gradient-based methods. In this chapter, however, we present a gradient-based method for transparent motion segmentation which is facilitated by a spatio-temporal filtering tech-

nique based on 3-D Hermite polynomials. We achieve a fast and accurate algorithm. This work offers a comprehensive solution to transparent motion segmentation by addressing all three issues involved in transparent motion segmentation: determining the existence of multiple motions, extracting transparent motions and segmenting transparent motions. In the future, a more sophisticated assumption about the motion will be developed to adapt the algorithm to the general case.

Chapter 7

Accuracy vs. Efficiency Trade-offs in Optical Flow Algorithms

There have been two thrusts in the development of optical flow algorithms. One has emphasized higher accuracy; the other faster implementation. These two thrusts, however, have been independently pursued, without addressing the accuracy vs. efficiency trade-offs. Although the accuracy-efficiency characteristic is algorithm dependent, an understanding of a general pattern is crucial in evaluating an algorithm as far as real world tasks are concerned. To meet various performance requirements of these tasks, this chapter addresses many implementation issues relevant to the accuracy vs. efficiency trade-offs that have often been neglected in previous research, for example, subsampling, temporal filtering of the output stream, algorithms' flexibility and robustness, etc. We present a critical survey of different approaches toward the goal of higher performance and conduct experimental studies of accuracy vs. efficiency trade-offs. A detailed analysis of how this trade-off affects algorithm design is manifested in a case study involving two state-of-the-art optical flow algorithms: one is gradient-based [65] [66] and the other correlation-based [21] . The goal of this chapter is to bridge the gap between the accuracy and the efficiency-oriented approaches.

7.1 Introduction

Whether the results of motion estimation are used in robot navigation, object tracking, or other applications, one of the most compelling requirements for an algorithm to be effective is the speed. No matter how accurate an algorithm may be, it is not useful unless it can output its results within the necessary response time for a given task. On the other hand, no matter how fast an algorithm runs, it is useless unless it computes motion sufficiently accurately and precisely for subsequent interpretations.

Both accuracy and efficiency are important as far as real world applications are concerned. However, recent motion research has taken two approaches in opposite directions. One neglects all considerations of efficiency to achieve the highest accuracy possible. The other trades off accuracy for speed as required by a task. These two criteria could span a whole spectrum of different algorithms, ranging from very accurate but slow to very fast but inaccurate. Most motion algorithms are clustered at the ends of the spectrum. Applications which need a certain combination of speed and accuracy may not find a good solution among these motion algorithms. Therefore, to evaluate an algorithm for practical applications, we propose a 2-dimensional scale where one of the coordinates is accuracy and the other is time efficiency. In this scale, an algorithm with different parameter settings generates an accuracy-efficiency (AE) curve, which will assist users in understanding its operating range (accuracy-efficiency trade-offs) and in

optimizing its performance.
In evaluating accuracy-efficiency trade-offs, we also consider implementation issues such as subsampling, temporal filtering and their effects on both accuracy and speed. The above issues are highlighted in a case study, in which we compare a gradient-based method [66] and a correlation-based method [21] . Both algorithms achieve about the same speed but work very differently. They represent state-of-the-art optical flow algorithms because few gradient-based methods achieve such high accuracy as that of [66] and few correlation-based methods achieve such high speed as that of [21] . We analyze the characteristics and state the strengths and weaknesses of each algorithm. The understanding of the subtle differences will assist a user in selecting a good algorithm for real-time applications.
Since the focus of the previous chapters was on accuracy and the relevant previous work has been discussed, we start with a survey of real-time implementations here.

7.2 Previous Work on Real-Time Implementations

Regarding the issue of speed, there is a prevailing argument in most motion estimation literature that with more advanced hardware that will be available in the near future, the techniques could be implemented to run at frame rate [9] . In a recent report, many existing algorithms' speeds (computing optical flow for the diverging trees sequence) are compared and compiled in a tabular form[11] . We use the data from this table and calculate the time (in years) it may take for these algorithms to achieve frame rate, assuming computing power doubles every year [87] . This result is displayed in Table 9. Note

Table 9: Execution time and expected time to achieve frame rate for diverging tree

Techniques	Horn	Uras	Anandan	Lucas	Fleet	Bober
Execution time (mins:secs)	8:00	0:38	8:12	0:23	30:02	8:10
Approximate execution time on HyperSparc 10	2:00	0:10	2:03	0:06	6:00	2:03
Expected time to frame rate	12 years	8 years	12 years	7 years	14 years	12 years

that some algorithm may take up to 14 years (from when the table was compiled) to achieve frame rate. This would drastically limit the potential of such algorithms for many practical applications over the next decade.
There have been numerous attempts to realize fast motion algorithms. There are two major approaches: the hardware approach and the algorithmic approach. They are sum-

marized in Table 10 and Table 11 given below and elaborated in the following para-

Table 10: Real-time motion estimation algorithms—hardware approach

Category	Type	Difficulties
Parallel computers	Connection machine [17] [85] [114] [117] Parsytec transputer [99] and hybrid pyramidal vision machine (AIS-4000 and CSA transputer)[31]	high cost, weight and power consumption
Special image processing hardware	PIPE [4] [25] [112] , Datacube [83] and PRISM-3 [86]	low precision
Dedicated VLSI chips	Vision Chips: gradient method [73] [105] , correspondence method [28] [104] and biological receptive field design [32] [72]	low resolution
	Non-Vision Chips: analog neural networks [61] digital block matching technique [8] [50] [115]	quantized results

Table 11: Real-time motion estimation algorithms—software approach

Technique	Algorithms	Difficulties
Sparse feature motion	tracking [4] [69] , computing time-to-contact (and hence obstacle avoidance)[25] and segmentation [99]	requirement of temporal filtering
Special constraints	constraint on motion velocity [21] , constraint on projection [118]	constraint on input images
Efficient algorithm	1-D spatial search [7] , separable filter design (Liu's algorithm[66])	requirement of careful algorithm design

graphs.

The hardware approach uses specialized hardware[34] to achieve real-time performance. There have been three categories of specialized hardware employed for motion estimation: parallel computers, specialized image processing hardware and dedicated VLSI chips.

Parallel computers such as the Connection Machine [17] [85] [114] [117] , Parsytec Transputer [99] , and hybrid pyramidal vision machine (AIS-4000 and CSA transputer)[31] have been used for optical flow estimation. These machines tend to be bulky and consume huge power. For applications that use motion cue for mobility such as robot navigation, such machines would significantly hamper the practicality of the motion algorithms in two ways: if the computer system is to be mounted on the robot platform, the power requirement is a major challenge for current battery technology; if the computer system is off the platform, the robot would have to be tethered in order for the computer to collect and process huge amount of sensor data. Noise and fluctuation in battery power often adversely affect the sensor signals (e.g. corrupt video synchronization). On the other hand, a tethered robot's mobility is limited. At the Perception Systems Laboratory at the National Institute of Standards and Technology, we have experienced these problems.

Specialized image processing hardware such as PIPE [4] [25] [92] [112] , Datacube [83] , and PRISM-3 [86] have been used for flow estimation. These machines achieve high speed with simplified arithmetic circuitry. Thus the algorithms' accuracies are compromised. This approach can only be used in motion applications which do not require precise estimates.

Finally, there is the use of dedicated VLSI chips to estimate flow. In terms of applications, there is significant potential in this approach because of its small size and power consumption. One method uses analog computing circuitry with built-in photoreceptors. These estimation chips represent a major class of the so-called "Vision Chips" or "Seeing Silicons"[71] . Gradient methods [73] [105] , correspondence methods [28] [104] , and biological receptive field designs [32] [72] have all been implemented. Although a great deal of progress has been made in the past decade, there are still several barriers to this approach. First, there is a lack of VLSI-friendly algorithms. Most flow estimation algorithms are simply too complex to implement efficiently. Second, there is a requirement for huge routing areas in the chips between adjacent photoreceptors. This requirement arises in any flow algorithm which performs even the simplest local operations (e.g. differentiation). This puts a limit on the number of cells that can be put on a chip. The third barrier is the insufficient understanding of biological visual systems, which usually inspire vision chip design. Hence, the limitations of the current chips include very low resolution (usually less than a hundred cells [32])and simplified applications (e.g. only 1-D motion estimation, or even only motion detection). Another method uses digitized images as input to the VLSI chips to perform motion estimation, i.e., there are no on-chip photosensors. One of the two major approaches uses analog neural networks [61] while the other uses digital block matching techniques [8] [50] [115] . This method achieves higher speed and marginally higher resolution than using vision chips. The precision of motion estimation, however, is limited because the motion vector is coarsely quantized.

It should be noted that any progress made in basic motion research can be reevaluated in the context of hardware implementations. It is hoped that such understanding can bridge the gap between algorithm development and its applications. It is clear now that the use of specialized hardware has had many problems, compromising its applicability to real world tasks. While the difficulties are being coped with, other research efforts have taken a different approach to real-time motion estimation—focusing on algorithm design.

The most popular method is to compute sparse feature motion. Sparse feature motion has been used for tracking [4] [69] [80] , computing time-to-contact (and hence obstacle avoidance)[25] and segmentation [99] . However, with sparse feature motion, to interpret the scenes in a sufficiently comprehensive way for these applications, these algorithms need to use extensive temporal filtering (e.g., recursive least square, Kalman filtering), which is time-consuming. Therefore, either the speed is not satisfactory[4] or they need to run on special hardware to achieve high speed[25] [69] [99] .

Another method is to constrain the motion estimation to a more tractable problem. For example, images can be subsampled so that the maximum velocity is constrained to be less than 1 pixel per frame. Therefore, a correlation method [21] can simply perform temporal matching in linear time instead of spatial search in quadratic time. Another

technique is to use a different projection. For example, any 3-D vertical line appears as a radial line in a conic projected image. This fact has been exploited for real-time navigation[118] .
Another elegant idea is to work on efficient design and implementation of the flow estimation algorithm. The main goal of this approach is to reduce the computational complexity. Suppose the image size is S and the maximum motion velocity is V. Traditional correlation algorithms performing spatial search or gradient-based algorithms using 2-D filters have the complexity $O(V^2S)$. Several recent spatio-temporal filter based methods even have $O(V^3S)$ complexity. Recent correlation-based algorithms use 1-D spatial search [7] while another gradient-based algorithm [66] exploits filter separability. They have achieved $O(VS)$ complexity,. This is a breakthrough because to output dense results, this complexity is the lower bound. The discussion of resolution, accuracy and robustness will be presented in the following sections. A nice characteristic of this design and implementation is that it runs very efficiently on a general purpose workstation or a microcomputer. It can easily be integrated with high-level processes (e.g., obstacle avoidance, robot control) because they often run on the same host computer.

7.3 Accuracy vs. Efficiency Trade-offs

Although real time is often used to mean video frame rate, in this chapter, real time is loosely defined as sufficiently fast for interactions with humans, robots or imaging hardware. The following subsections discuss the issues that are only of interest when one is concerned about both accuracy and speed. All the experiments illustrating our discussions are done on the diverging tree sequence.

7.3.1 Accuracy-efficiency curve

If a motion algorithm is intended to be applied in a real world task, its overall performance, including accuracy and efficiency, should be evaluated. Analogous to the use of electronic devices, without the knowledge of an algorithm's full operating range and characteristics, one may fail to optimize its performance. Using accuracy (or error) as one coordinate and efficiency (or execution time) as the other, we propose the use of a 2-D accuracy-efficiency (AE) curve to characterize an algorithm's performance. This curve is generated by setting parameters in the algorithm to different values.
For correlation methods, the template window size and the search window size are common parameters. For gradient methods, the (smoothing or differentiation) filter size is a common parameter. More complex algorithms may have other parameters to consider. The important thing is to understand their characteristics in a quantitative way.
For optical flow, accuracy has been extensively researched in Barron[9] . We will use the error measure in [9] , that is, the angle error between the computed flow $(u_c, v_c, 1)$ and the ground truth flow $(u, v, 1)$, as one quantitative criterion. For efficiency, there are two indicators: throughput and latency. An algorithm should attain high throughput (number of output frames per unit time) without long latency. However, throughput is more rele-

vant to efficiency than latency. We will use it as the other quantitative criterion.

In the 2-D performance diagram depicted below (Fig 54.), the x axis represents the angle error; the y axis represents the execution time. A point in the performance diagram corresponds to a certain parameter setting. The closer the performance point is to the origin (small error and low execution time), the better the algorithm is. An algorithm with different parameter settings spans a curve, usually of negative slope. Analogously, the distance from the origin to the AE curve represents the algorithm's AE performance. In Fig 54, there are two AE curves and several points*. It can be seen that some algo-

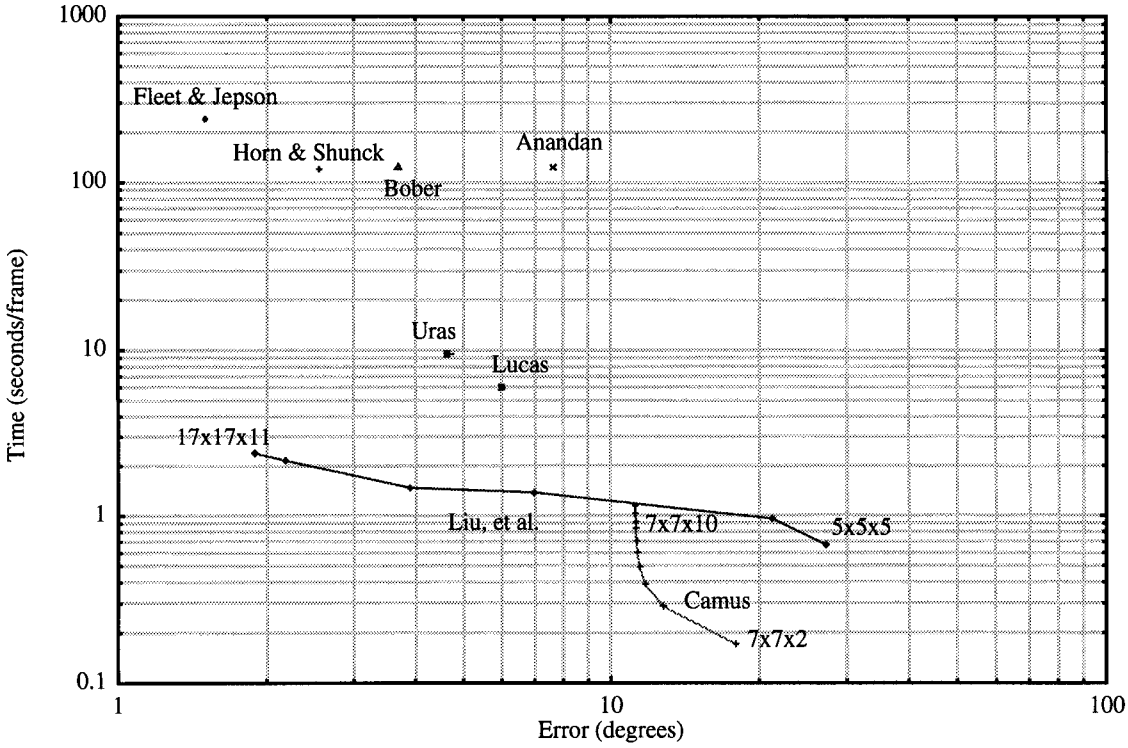


Fig 54. 2-D performance diagram

rithms (e.g., Fleet & Jepson[34]) may be very accurate but very slow while some algorithms (e.g., Camus[21]) may be very fast but not very accurate. In terms of AE performance, Liu, et al.'s algorithm in [66] is most flexible and effective because the curve is closest to the origin. It is interesting to find that the curve corresponding to the gradient-based method in [66] coincides with Camus's [21] curve at one point. This should not be a surprise because these two algorithms are approaching the problem from opposite starting points. Liu, et al's [66] algorithm is more focused on accuracy so the AE curve is close to horizontal, while Camus's algorithm is concerned more with speed so the AE curve is close to vertical. These curves are bound to intersect each other.

*. The implementations of other algorithms are provided by Barron [9] . Some of the algorithms produce different density, we simply project the error by extrapolation. In Liu's curve, the filter size used range from 5x5x5 to 17x17x11. In Camus's curve the template size ranges from 7x7x2 to 7x7x10. The execution time for all algorithms is the approximate elapsed time running on the same machine (80MHz HyperSparc 10 board). They are faster than those reported in Table 9.

The AE curve is also useful in understanding the effect and cost of certain types of processing. For example, Fig 55 depicts the trade-off of using a median filter in Liu, et al.'s

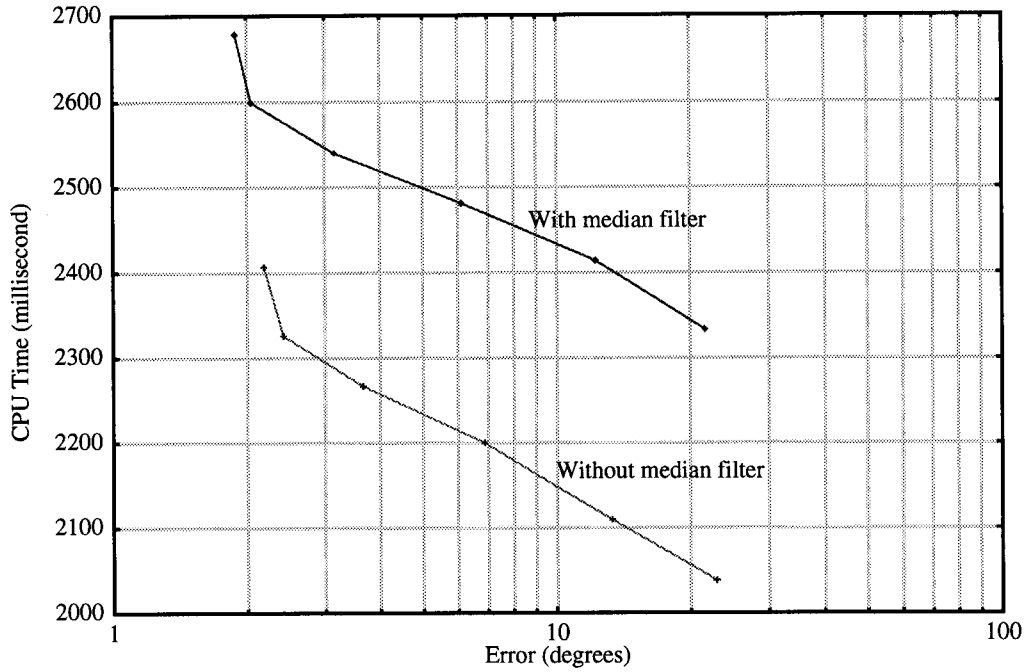


Fig 55. The effect and cost of using a median filter

algorithm on the flow field output. The two curves are generated using different filter sizes. Note that the curve using the median filter has less error (~14%) but the computation cost is higher (~12%). Since the curve using the median filter is generally farther away from the origin, it is not advisable to use a median filter for AE critical tasks. However, to achieve maximum accuracy, a median filter can be used. For the same speed, using no median filter (with a larger filter size) results in higher accuracy than using the median filter (with a smaller filter size). This means that the selection of the right filter size is more important than the application of a median filter. Fig 56 shows the effect and cost of using different orders of image derivatives in Liu, et al.'s algorithm. The trade-off is even clearer in this example. Using only up to second order derivatives saves 50% in time while sacrificing 85% in accuracy. Parameters can be adjusted for different applications according to their specific needs.

7.3.2 Subsampling effect

The computational complexity of optical flow algorithms is usually proportional to the image size. However, an application may not need the full resolution of the digitized image. An intuitive idea to improve the speed is to subsample the images. Subsampling an image runs the risk of undersampling below the Nyquist frequency, resulting in aliasing, which can confuse any motion algorithms. To avoid aliasing, the spatial sampling distance must be smaller than the scale of image texture and the temporal sampling period must be shorter than the scale of time. That is, the image intensity pattern must evidence a phase shift that is small enough to avoid phase ambiguity[94] .

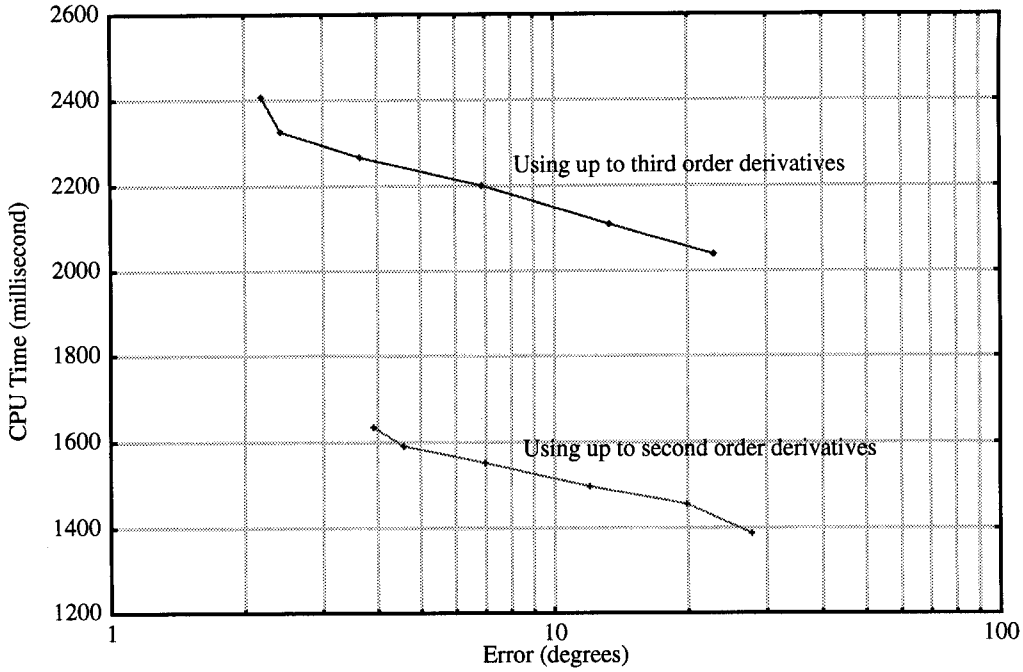


Fig 56. The effect and cost of using different orders of derivatives.

Temporal subsampling resulting in phase ambiguity and hence false motion perception is easier to understand. In fact, it is often seen in motion pictures, e.g. reverse spinning of vehicle wheels. However, spatial subsampling causing spatial aliasing and hence false motion perception is illustrated in Fig 57. The left column shows the original curve and the right column shows only the sample points. The top two rows show the original sample pattern and the bottom two show the subsampled pattern. It is easily seen that the original image is perceived to move one pixel to the right while the subsampled image is perceived to move one pixel to the left.

In summary, subsampling should be avoided on an image sequence with high spatial frequency and large motion. This aliasing problem can be dealt with by smoothing (low-pass filtering) before subsampling. However, since smoothing is done on the original images and the computational cost is proportional to the original image size, the advantage of subsampling is lost.

Aliasing is not the only problem in subsampling. Object size in subsampled images is reduced quadratically but the image boundaries are reduced linearly (in terms of number of pixels). Hence the density of motion boundaries is higher. This is detrimental to optical flow algorithms in general.

In short, subsampling can improve efficiency but needs careful treatment.

7.3.3 Temporal processing of the output

Since most general purpose optical flow algorithms are still very inefficient and often operate on short "canned" sequences of images, they tend to be oblivious of the abundant past output information. For long image sequences, it is natural to consider the possibility of temporally integrating the output stream to achieve better accuracy.

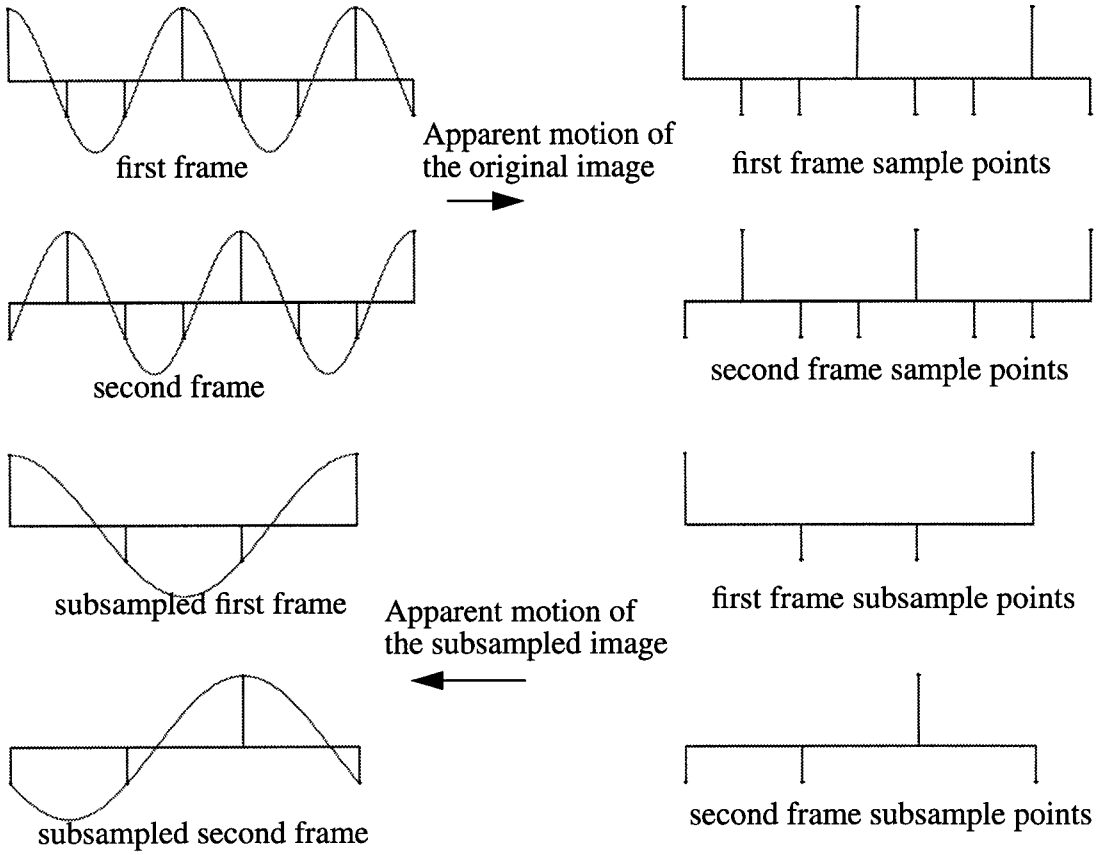


Fig 57. Subsampling effect.

Temporal filtering is often used in situations where noisy output from the previous stage cannot be reliably interpreted. Successful applications of temporal filtering on the output requires a model (e.g., Gaussian with known variance) for noise (Kalman filtering) or a model (e.g. quadratic function) for the underlying parameters (recursive least squares). Therefore, these methods are often task specific.

A general purpose Kalman filter has been proposed in [98] where the noise model is tied closely to the framework of the method. In this scheme, an update stage requires point-to-point local warping using the previous optical flow field (as opposed to global warping using a polynomial function) in order to perform predictive filtering. It is computationally very expensive and therefore has little prospect of real-time implementation in the near future (note that for example, the current Datacube warper provides only global polynomial warping function but not local warping). So far, Kalman filters and recursive least square filters implemented in real-time are limited to sparse data point [69] [25] .

We have experimented with a simple, inexpensive temporal processing approach—exponential filtering. Suppose $b(t)$ is the original optical flow estimate. We derive the exponential filtered output $a(t)$ based on the following equation:

$$a(t) = \sum_{k=0}^{\infty} f(1-f)^k b(t-k) = f\,b(t) + (1-f)\,a(t-1). \qquad (71)$$

where f is the forgetting factor. When the forgetting factor is 1, which means the algorithm forgets everything in the past, there is no exponential filtering. On the other hand, when the forgetting factor is 0, the filtering always outputs the previous result and completely ignores the current estimate. With an appropriate forgetting factor, this filtering is excellent for estimating a flow field that is constant over time. It is also very helpful for flow outputs that are very noisy. Applying the filtering to Liu, et al's algorithm [66] running on the diverging tree sequence, our experiments show that with parameter settings that already achieve high accuracy, exponential filtering may increase accuracy, although oversmoothing may reduce accuracy. This is illustrated in Fig 58.1: the average angle

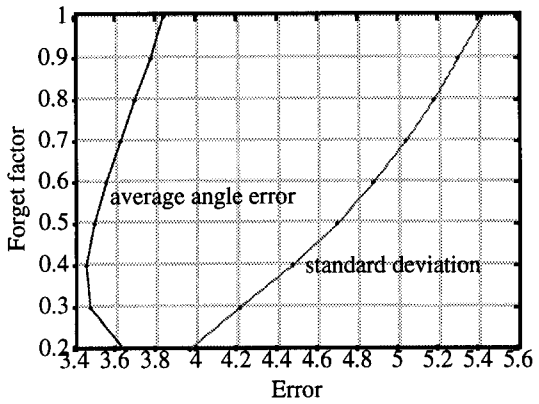


Fig 58.1 Temporal smoothing on high accuracy output

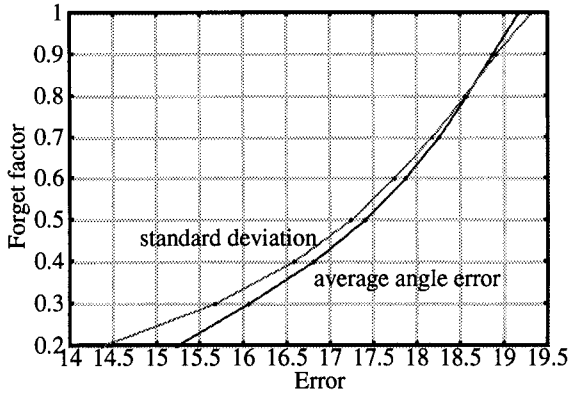


Fig 58.2 Temporal smoothing on low accuracy output

error improves with some filtering (i.e., as the forgetting factor decreases from 1) but worsens when filtering is applied beyond some point (forgetting factor < 0.4). On the other hand, in Fig 58.2, with parameter settings that generate low accuracy original output, the filtering can greatly improve accuracy (up to 25%, from 19.17 to 15.23) very much while its computational cost increases minimally (less than 2%, from 0.975 to 0.989 CPU second/frame). As expected, noise in the filtered output, represented by the standard deviation of the angle error, is uniformly proportional to the forgetting factor (see Fig 58.1 and Fig 58.2).

In summary, when the noise in the output is high or the output is expected to remain roughly constant, the exponential filtering improves accuracy with little computational overhead, but when the scene or motion is very complex or contains numerous objects, exponential filtering is less likely to improve accuracy.

7.3.4 Flexibility and robustness

It has been pointed out that some motion algorithms achieve higher speed by constraining the input data, e.g., limiting the motion velocity to be less than a certain value, thus

sacrificing some flexibility and robustness. Some algorithms optimize the performance for a specific situation. For good performance in other situations, users may need to retune an array of parameters. It is thus important to understand how these constraints or parameter tunings affect the accuracy. Flexibility refers to an algorithm's capability to handle widely varying scenes and motions. Robustness refers to resistance to noise. These two criteria prescribe an algorithm's applicability to general tasks.

To evaluate algorithm flexibility, we conducted a simple experiment as follows. We generated a sequence by taking every other frame of the Diverging trees sequence. The motion in the new sequence should be twice as large. We then ran algorithms on this sequence using the same parameters as on the original sequence and compared the errors in the two outputs. A flexible algorithm should yield similarly accurate results, so we examine performance variation rather than absolute accuracy here. The following figure

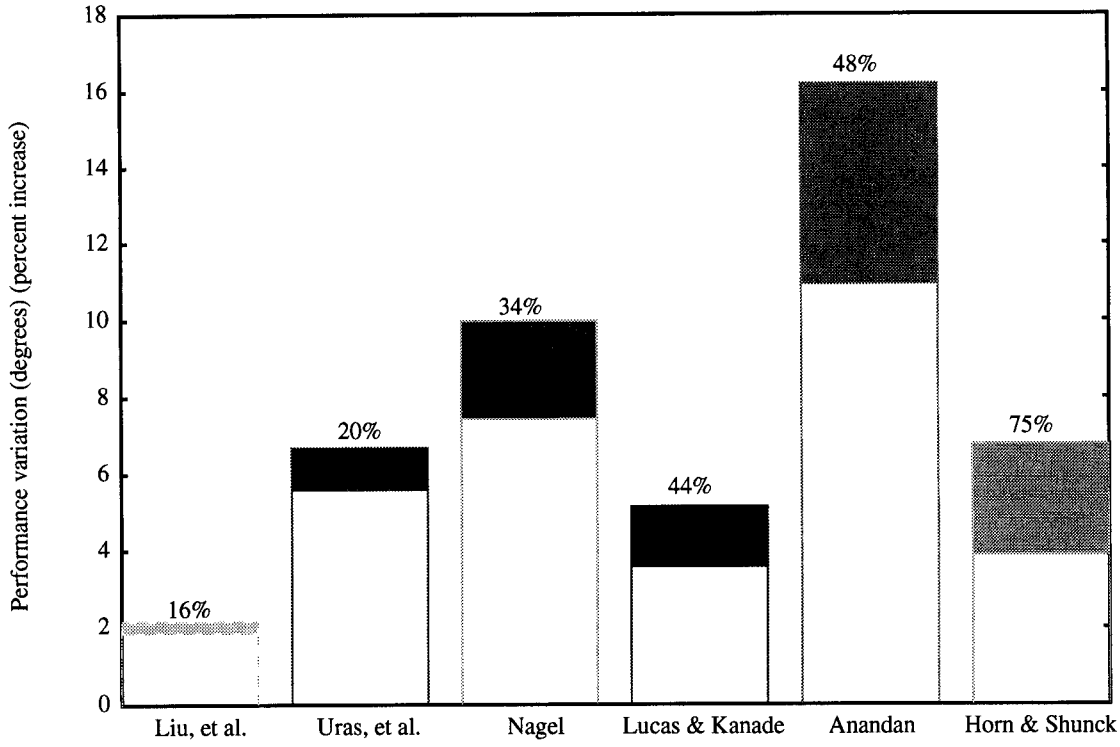


Fig 59.Algorithm flexibility in handling different motion.

illustrates the results. The algorithms' performance variations for these two sequences ranges from 16% (Liu, et al.) to 75% (Horn & Shunck). We had hoped to include the results of Fleet and Jepson's algorithm, which has been very accurate, but we were unable to generate other than 0% density on the new sequence. Their algorithm seems to be perfectly tuned for the original sequence.

To evaluate the algorithms' noise sensitivity, we generated a new diverging tree sequence by adding Gaussian noise of increasing variance and observed the algorithms' performance degradation. The following figure illustrates the algorithms' noise sensitivity. Some algorithms (Lucas & Kanade, Liu, et al., Anandan, Camus) have linear noise sensitivity with respect to noise variance, some (Fleet and Jepson) show quadratic noise

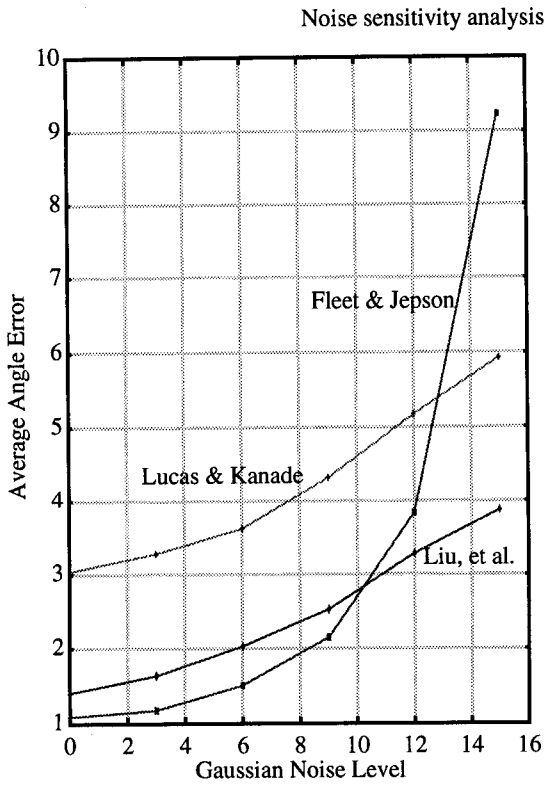


Fig 60.1 Noise sensitivity for 50% density data

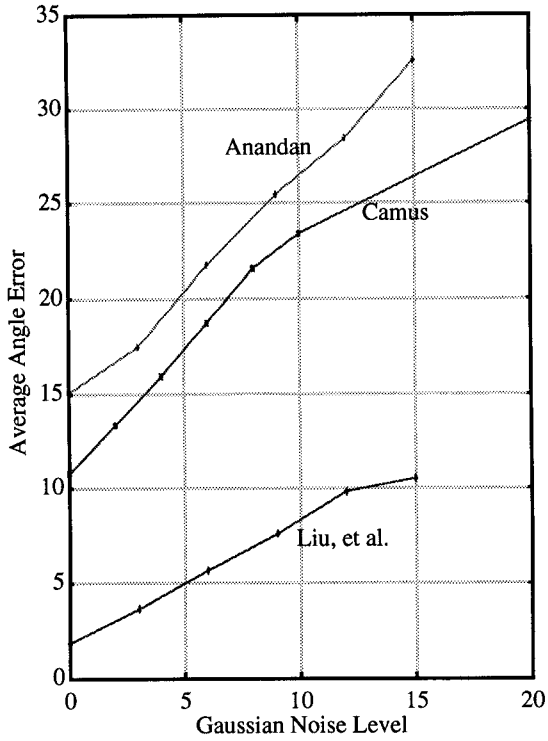


Fig 60.2 Noise sensitivity for 100% density data.

sensitivity.

7.3.5 Output density

Most algorithms do some thresholding to eliminate unreliable data and hope that the density is adequate for the subsequent applications. In addition, the threshold value is often chosen arbitrarily (by users who are not experts in the algorithms) without regard to the characteristics of the algorithms. The important characteristics that should be considered are flexibility and robustness to noise. If an algorithm is accurate but not flexible and not robust to noise, then it is better off generating a sparse field because the more data it outputs the more likely it will contain noisy data. However, the applications should have a final say on the output density. A dense flow field is always ideal, but selecting the right density of sufficiently accurate output is a more practical approach.
In section 3.2 on page 38, we showed that density is very important in real-world applications. For example, although Fleet and Jepson's algorithm [34] is claimed to be very accurate, its sparse flow field on the NASA sequence can barely be used for obstacle detection. It should be noted that sometimes a less accurate dense flow field is better than an accurate sparse flow field in real world applications. This issue is unfortunately neglected in Barron's performance evaluation [9] .

7.3.6 Hardware constraints

Many researchers, after developing an accurate but slow algorithm, often argue that specialized hardware can make a real-time implementation feasible. It should be noted that many potential problems may arise when attempting such an implementation. For example, limitations in hardware precision, memory space, data path and data routing capacity, unsupported operations (e.g., division is not supported in the Datacube), and synchronization overhead (among parallel or pipelined units) can all set a limit to the accuracy and/or speed of the implementation. For example, we have implemented an optical flow algorithm[62] running on a Datacube MV200 at about 10 frames per second. Its accuracy, due to low precision convolution, is poor compared to the implementation on a general purpose machine where floating point computation is performed.

7.4 A Case Study

In this section, we describe similarities and distinctions between gradient-based methods and correlation-based methods for optical flow estimation.
The traditional intuitive distinctions between these two methods are the following: the correlation-based methods perform search, so they are very slow; the gradient-based methods perform numerical differentiations, so they are very inaccurate. However, with recent advances in optical flow research, the above intuition is no longer true. For example, efficient correlation algorithms have been developed [7] [21] . Some accurate gradient algorithms have also been developed [66] [114] .
In fact, a certain correlation algorithm is equivalent to a certain gradient algorithm. In this section, we investigate their similarities and distinctions. We present a comparison study between Liu, et al.'s gradient algorithm [66] and Camus's correlation algorithm [21] . They represent state-of-the-art optical flow algorithms because few correlation algorithms have been as efficient as [21] and few gradient algorithms have been as accurate as [66] . This comparison serves two purposes. First, it puts all the issues discussed in the previous section in the context of two specific algorithms. Second, it points out the strengths and weaknesses between a general gradient method and a general correlation method. This understanding is important in selecting algorithms for particular real-time applications.

7.4.1 Equivalency

Although correlation methods and gradient methods are conceptually different, we show here that under some situations, they are equivalent.
A correlation algorithm that minimizes Sum of Squared Differences (SSD) is formulated as the following:

$$Min_{\Delta x,\,\Delta y}\iint (E)^2 dxdy \quad \text{, where } E = I(x, y, t) - I(x + \Delta x, y + \Delta y, t + \Delta t). \qquad (72)$$

If the image intensity pattern is smooth, then

$$E \approx -\frac{\partial I}{\partial x}\Delta x - \frac{\partial I}{\partial y}\Delta y - \frac{\partial I}{\partial t}\Delta t = -(uI_x + vI_y + I_t)\Delta t \ . \tag{73}$$

$$\text{so} \iint (E)^2 dxdy \approx \iint (uI_x + vI_y + I_t)^2 \Delta t^2 dxdy \tag{74}$$

Equation (74) is used in the least squares formulation of the patchwise gradient algorithm[68] . So

$$Min_{\Delta x,\, \Delta y} \iint (E)^2 dxdy \approx Min_{u,\, v} \iint (uI_x + vI_y + I_t)^2 \Delta t^2 dxdy . \tag{75}$$

In addition to the above analysis, Anandan[6] has used another weighting function to analyze the relation between his correlation algorithm and Nagel's second order gradient method.

We see from this analysis that many motion algorithms are equivalent or related. It is the implementations that distinguish them. And this is the reason why we consider implementation issues to be very important in this chapter.

7.4.2 Comparing a gradient algorithm with a correlation algorithm

Camus [21] has developed a real-time algorithm using the correlation method. Its speed is about 5 frames per second on a general purpose computing machine (50 MHz Sparc 20) on 64x64 images. It is currently one of the most efficient general purpose optical flow algorithms. The basic idea of this algorithm is to subsample the image and thus constrain the motion velocity so that a quadratic search in space can be reduced to a linear search in time. This temporal matching concept is depicted in Fig 61.1-2. This algo-

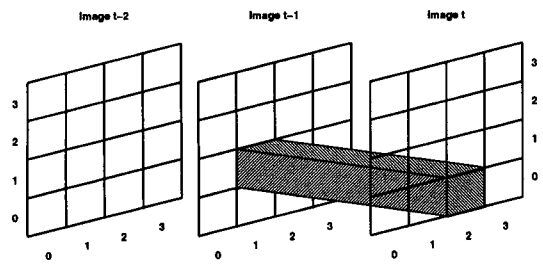


Fig 61.1 Visualization of pixel (1,1) in image T-1 moving to pixel (2,0) in image T, an optical flow of (1,-1) pixels per frame.

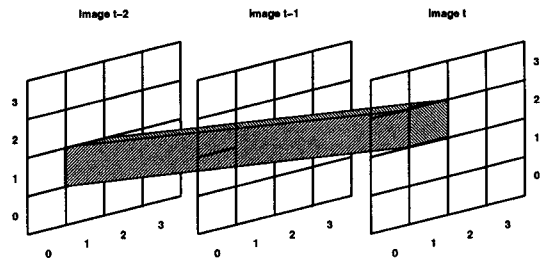


Fig 61.2 Visualization of pixel (1,1) in image T-2 moving to pixel (1,2) in image T, an optical flow of (0,1/2) pixels per frame.

rithm outputs quantized flow. The number of quantization levels is proportional to the range of the temporal search. For the performance mentioned above, the temporal search range is 10 frames and the template window size is 7x7. Despite the algorithm's simplicity and quantization, it is sufficiently accurate for computing time-to-contact robustly [20] .

On the other hand, Liu, et al.'s algorithm [66] (denoted as the gradient algorithm) uses up to third order spatio-temporal derivatives and a generalized motion model that accommodates expansion and translation. The speed is roughly 3 to 10 frames per sec-

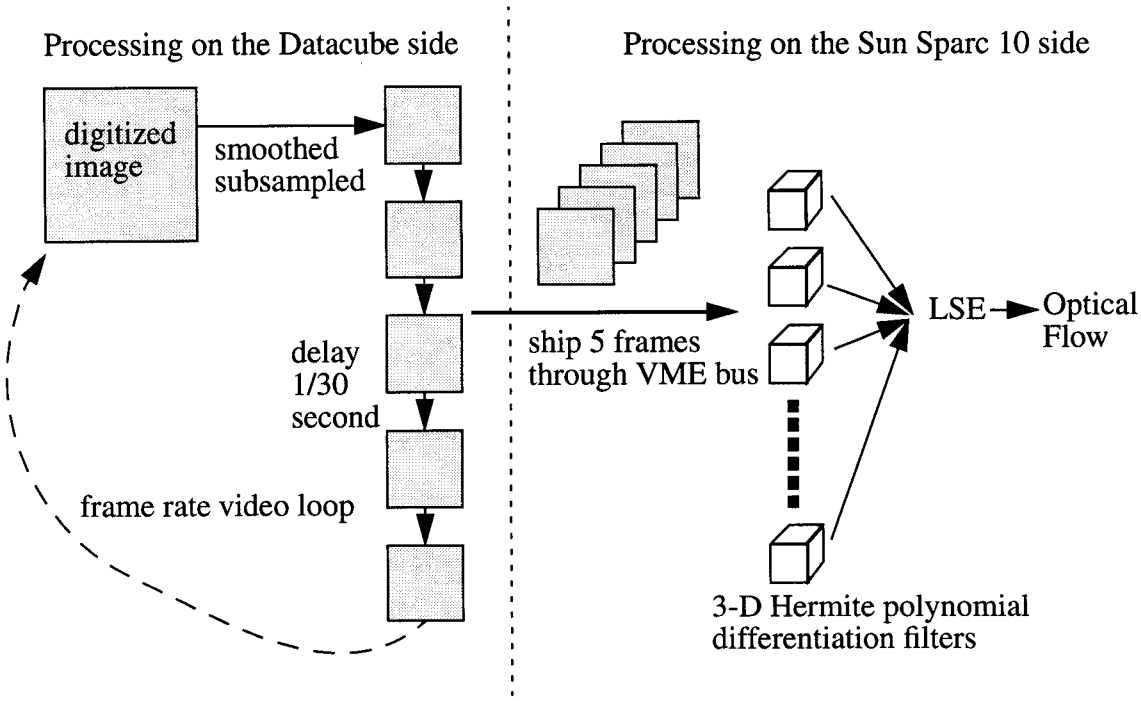


Fig 62.Real-time implementation of Liu, et al.'s optical flow algorithm

ond depending on the window size, median filtering, order of derivatives used, density of output, etc. Generally speaking, the efficiency is about equal to that of Camus's algorithm [21] (denoted the correlation algorithm).

Efficiency is the only thing that is common in these two algorithms. Their differences are listed in Table 12 and elaborated in the following paragraphs.

Table 12: Comparing Liu's gradient algorithm with Camus's correlation algorithm

Liu's gradient algorithm		**Camus's correlation algorithm**	
strengths	accurate flow	weaknesses	quantized flow
	handles various motion velocity		constrains motion velocity
	models expansion explicitly		models translation only
	multiple confidence measures		no confidence measure
	handles occluding boundaries		distorts occluded object
	robust to aperture problem		less robust to aperture problem
weaknesses	sensitive to noise (brightness change)	strengths	more robust to noise
	susceptible to subsampling effect		less susceptible to subsampling effect
	latency due to frame delays		no latency due to frame delays
	requires floating point computation		uses only integer computation

The gradient algorithm outputs very accurate flow, as evaluated previously, while the correlation algorithm outputs less accurate quantized flow. It would be very difficult to extract from quantized flow the first order optical flow information (e.g., divergence), which is often used for subsequent applications. On the other hand, since the gradient method is based on the brightness constancy assumption, it is sensitive to noise or to brightness changes of any scale. As analyzed earlier, the error magnitude is proportional to the noise magnitude. In many real world image sequences, the correlation algorithm shows more robustness to brightness changes and the flow field seems smoother in space and more stable over time.

The gradient algorithm is capable of handling motion velocities up to about a quarter of the filter size. It is also very good at extracting very small motions, like that in the NASA sequence. Generally, it can handle 0.05 to 4 pixels per frame of motion. The correlation algorithm is only capable of handling 1/10 to 1 pixel per frame of motion.

The gradient algorithm can use two confidence measures (residuals and smallest eigenvalue) to output more reliable sparse flow. These confidence measures, as discussed earlier, reflect not only reliability of the output data but also image properties. The correlation algorithm does not generate confidence measures.

Occluding boundaries are difficult for all motion estimation algorithms. The gradient algorithm is better at handling occluding boundaries because symmetric (noncausal) temporal filters are used, i.e., future frames as well as past frames are used to compute flow. Although the motion within the local 3-D filter window contains multiple motions,

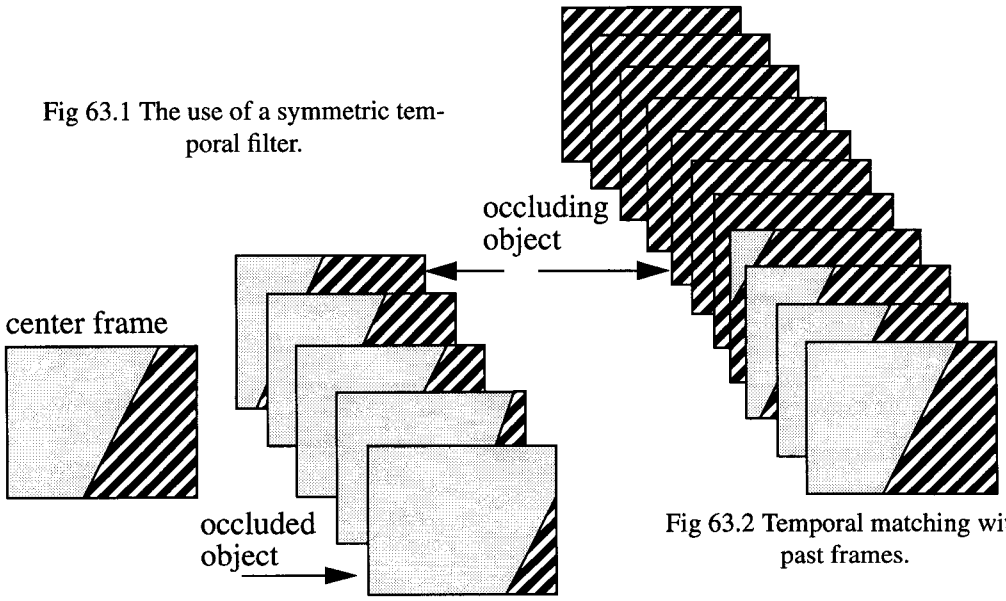


Fig 63.1 The use of a symmetric temporal filter.

Fig 63.2 Temporal matching with past frames.

as long as the object under consideration occupies more than half of the window, the estimation is still closer to true motion(Fig 63.1). On the other hand, matching the current frame with only previous frames may produce erroneous estimates. In fact, it has a tendency to find the best match in the most recent frame because the object under consideration is occluded in most of the past frames(Fig 63.2). This is a systematic error for any algorithm using noncausal filtering or matching. For the same reason, the gradient

algorithm introduces an additional latency in the output proportional to half the temporal filter size. The latency of the correlation algorithm, on the other hand, is only due to computation. Specifically, for a throughput of 5 frames per second, the latency of the gradient algorithm is $2 \times 33ms + 200ms = 266ms$ (temporal filter support is 5, resulting in a 2 frames latency, and images are captured at frame rate, 33ms per frame); The correlation latency is $200ms$.

The gradient algorithm is less susceptible to the aperture problem because a large filter support is used.

When the image is subsampled, the gradient algorithm is more likely to suffer from aliasing than the correlation algorithm. Consider as an example an extreme case where a random dot pattern is moving one pixel to the right. In the subsampled image sequence, adjacent frames become completely uncorrelated, so the gradient algorithm cannot estimate motion correctly. The correlation algorithm, on the other hand, can find a perfect match in one particular frame in the past. The distance between the current frame and the matched frame is equal to the subsampling period. Note that the gradient method is operating similarly to biological receptive fields, which also have difficulties perceiving motion in a subsampled random dot pattern, especially when the subsampling distance is high.

Finally, the gradient algorithm requires extensive use of floating point operations. Using only integer computations may significantly compromise the gradient algorithm's accuracy. The correlation algorithm uses only integer computations and thus may be easily and efficiently implemented in special hardware.

Although this comparison is specific to the two algorithms under consideration, the major strengths and weaknesses mentioned are mostly general to gradient algorithms and correlation algorithms. The understanding of these strengths and weaknesses offers new insight beyond the traditional intuitive distinctions between these two approaches.

7.5 Conclusion

Motion research has typically focused on only accuracy or only speed. We have reviewed many different approaches to achieving higher accuracy or speed and pointed out their difficulties in real world applications. We also have raised the issues of accuracy-efficiency trade-offs resulting from subsampling effects, temporal processing of the output, algorithm flexibility and robustness, output density, and hardware implementation constraints. It is only through consideration of these issues that we can address a particular algorithm's applicability to real-world tasks. We highlight these issues by doing a case study of comparisons between Liu, et al.'s gradient algorithm [66] and Camus's correlation algorithm [21] . Due to recent advances in motion research, the distinctions between these two approaches are becoming less clear. In theory, some gradient methods and some correlation methods are equivalent. However, the implementations are very different. The understanding of relative strengths and weaknesses is important in selecting an algorithm for a real-time application. The accuracy-efficiency trade-off issues discussed here are by no means exhaustive. We hope that this initial study can generate more interesting discussions and shed some light on the use of

motion algorithms in real world tasks.

Chapter 8
Conclusions and Future Research

8.1 Conclusions

Optical flow estimation is difficult and ill-posed. The past two decades of research have led to spatio-temporal filtering techniques to overcome sensor noise, brightness change over time, and quantization error. The aperture problem is mitigated by increased filter support or other global techniques, while other approaches attempt to use an affine motion model to pursue better accuracy in the optical flow. We have learned from these results and have developed an integrated approach that combines a general motion model and 3-D Hermite polynomial differentiation filters.

The general motion model we use avoids many dilemmas in motion algorithms including filter (template) size selection, motion boundary extraction and flow field divergence computation. Due to these dilemmas, traditional approaches that interpret motion using optical flow are often self-conflicting.

What distinguishes the general motion model we use is that it is based on the *continuous-time* analysis of arbitrary 3-D motion. The motion equation depicts *pointwise* relation in the image sequence and hence is correct for arbitrary 3-D scene and not restricted to rigid motion. It is useful for all motion algorithms, but better numerical techniques are required to make good use of the model. We have found that Hermite polynomial theory provides necessary advantages for this purpose. It possesses many elegant properties, including orthogonality, extensibility, Gaussian smoothing, and the recursive relation (20) that facilitates the quadratic motion equation, etc. Contrary to general belief, the behaviors of these high order differentiation filters are quite insensitive to noise. This observation is supported by the good results in our noise sensitivity analysis. Simplicity adds yet another dimension to the strength of this algorithm, making real-time implementation feasible.

Combining the general motion model and spatio-temporal Hermite polynomial filters, we have solved for optical flow computation, motion boundary extraction, time-to-contact estimation, and transparent motion segmentation.

At the application level, our optical flow algorithm generates a set of confidence measures that we prove reflect physical phenomena about the image and motion. These measures can then be used for subsequent qualitative processing, for example, separating normal flow from accurate optical flow (Section 2.5.3.4). In experiments, our algorithm generates accurate and *dense* results, which are very useful for such tasks as motion segmentation and obstacle detection (Section 3.2.3). Our motion boundary extraction algorithm accurately localizes boundaries and the separate evaluation scheme also helps distinguishes this characteristic of our algorithm. The time-to-contact algorithm successful builds a dense threat map of the environment from real images. Transparent motion is no longer an outlying process that disturbs normal processing because it is correctly

modeled so that the foreground motion can be reliably segmented.
Our algorithms' accuracy is rigorously evaluated using not only well established standard schemes (e.g., [9]) but also real world applications (e.g., obstacle detection in Section 3.2.3). Their accuracy, flexibility, and robustness is the justification of our sound theory and excellent numerical techniques.
Above all, our algorithms' complexity achieves the lower bound and the implementations are so efficient that we are able to address a whole new array of real-time issues that are certainly of interest to speed conscious researchers.

In summary, the major contributions of this report are a sound theory that interprets 3-D motion unambiguously, a solid set of algorithms that perform well in terms of accuracy and efficiency, rigorous evaluations that clearly distinguishes algorithms' characteristics, and real-time implementations that are suitable for many real world applications.

8.2 Future Research

Real world tasks that can benefit from the theory and implementations presented here include unmanned vehicle (obstacle avoidance, range recovery), reconnaissance, surveillance, target acquisition (image stabilization, object tracking, motion detection), video compression (motion estimation, segmentation), etc.
Future directions for study include integration of the algorithms in an adaptive way, for example, when there is transparent motion, we segment the foreground motion; and when there are independently moving objects, we segment them using boundary information and estimate their motions using 3-D motion parameters; when an object poses a threat to the observer, the algorithm issues certain warnings to a (robot) control system that navigates to avoid the threat. We hope that our evaluation and real-time discussion generates more rigorous study on the applicability of motion algorithms to real world tasks.

Appendix. A

We prove Theorem 1 as follows:

Proof: The first equality comes from the orthogonality of $\{\overline{H}_n(x)\}$. We now prove the second equality, which claims that the scalar product of a function and the kth Hermite polynomial is equal to the scalar product of the kth derivative of the function and 1.

$$\begin{aligned}
\langle I, \overline{H}_k \rangle &= \int_{-\infty}^{\infty} G(x)I(x)\overline{H}_k(x)dx \\
&= \int_{-\infty}^{\infty} G(x)I(x)(-1)^k G^{-1}(x)\frac{d^k G(x)}{dx^k}dx \\
&= (-1)^k \int_{-\infty}^{\infty} I(x)\frac{d^k G(x)}{dx^k}dx \\
&= (-1)^k I(x)\frac{d^{k-1}G(x)}{dx^{k-1}}\Big|_{-\infty}^{\infty} + (-1)^{k-1}\int_{-\infty}^{\infty}\frac{dI(x)}{dx}\frac{d^{k-1}G(x)}{dx^{k-1}}dx \\
&= \int_{-\infty}^{\infty} I^{(1)}(x)(-1)^{k-1}\frac{d^{k-1}G(x)}{dx^{k-1}}dx \\
&= \langle I^{(1)}, \overline{H}_{k-1}\rangle \\
&\vdots \\
&= \langle I^{(k)}, \overline{H}_0\rangle
\end{aligned}$$

❑

Appendix. B

Let A_n and b, defined in (35), contain no noise and let the noise be modelled as in (42). Then

$$E = A_n s + b = 0 \text{ and } s = -(A_n^T A_n)^{-1} A_n^T b. \tag{76}$$

Let the noise contaminated optical flow be $\tilde{s}$ and the new residual be $\tilde{E}$, and assume that $N \ll A_n$ and $\Delta b \ll b$ elementwise, i.e., $NN^T \approx 0$ and $N\Delta b \approx 0$. Then

$$\tilde{s} = -[(A_n + N)^T(A_n + N)]^{-1}(A_n + N)^T(b + \Delta b), \text{ and} \tag{77}$$

$$[(A_n+N)^T(A_n+N)]^{-1} \approx (A_n^TA_n[I+(A_n^TA_n)^{-1}(A_n^TN+N^TA_n)])^{-1}$$

$$\approx [I-(A_n^TA_n)^{-1}(A_n^TN+N^TA_n)](A_n^TA_n)^{-1} \quad \text{, so} \qquad (78)$$

$$\tilde{s} \approx -(A_n^TA_n)^{-1}A_n^Tb+(A_n^TA)^{-1}(A_n^TN+N^TA_n)(A_n^TA_n)^{-1}A_n^Tb+(A_n^TA_n)^{-1}N^Tb-(A_n^TA$$

Using (76), this can be simplified as follows:

$\tilde{s} \approx s-(A_n^TA_n)^{-1}A_n^TNs-(A_n^TA_n)^{-1}A_n^T\Delta b$ and $\Delta s \approx (A_n^TA_n)^{-1}A_n^TNs-(A_n^TA_n)^{-1}A_n^T\Delta b$.

For the residual, substituting $\tilde{s}$ into (42), and using (76), we have

$$\tilde{E} \approx \left\|(A_n+N)s-A_n(A_n^TA_n)^{-1}A_n^TNs-A_n(A_n^TA_n)^{-1}A_n^T\Delta b+b+\Delta b\right\|$$

$$\approx \left\|(I-A_n(A_n^TA_n)^{-1}A_n^T)(Ns+\Delta b)\right\| \qquad \text{as in (44).}$$

To understand $\tilde{E}$ better, we analyze the matrix $I-A_n(A_n^TA_n)^{-1}A_n^T$, denoted by T.

It is easy to verify that the only nontrivial eigenvalues of matrix T is/are 1, which means that it maps any vector $(Ns+\Delta b)$ to only the directions specified by the eigenvectors corresponding to the nontrivial eigenvalues.

Appendix. C

We prove equation (60) here.

$$\frac{\partial}{\partial x}I(x,y,t) = \frac{\partial}{\partial x}F(x',y') \text{ where } \begin{matrix} x' = x(1-st)-pt \\ y' = y(1-st)-qt \end{matrix}. \qquad (79)$$

Assume the surface is parallel frontal, so $\frac{\partial s}{\partial x} = 0$

$$\text{so } \frac{\partial}{\partial x}I(x, y, t) = \frac{\partial}{\partial x'}F(x', y')\frac{\partial}{\partial x}x' \tag{80}$$

$$= (1-st)\left(\frac{\partial}{\partial x'}F(x', y')\right)\Big|_{\substack{x' = x(1-st)-pt \\ y' = y(1-st)-qt}}$$

$$= (1-st)\left(\frac{\partial}{\partial x'}I(x', y', 0)\right)\Big|_{\substack{x' = x(1-st)-pt \\ y' = y(1-st)-qt}}$$

$$= (1-st)\frac{\partial}{\partial x}I(x(1-st)-pt, y(1-st)-qt, 0)$$

$$= (1-st)\frac{\partial}{\partial x}I(x', y', 0)$$

Appendix. D

We prove equation (64) here
First, from (18) and (22), we establish

$$x\overline{H}_n(x) = \sigma^2\overline{H}_{n+1}(x) + n\overline{H}_{n-1}(x). \tag{81}$$

Equating I_{ij1} to F_{ij1} and using Theorem 1,

$$\begin{aligned} I_{ij1} = F_{ij1} &= \langle F, \overline{H}_{ij1}\rangle = \langle\frac{\partial F}{\partial t}, \overline{H}_{ij0}\rangle \\ &= -\langle\left(\frac{s(x-x_0)+p}{1-st}\right)\frac{\partial I}{\partial x} + \left(\frac{s(y-y_0)+q}{1-st}\right)\frac{\partial I}{\partial y}, \overline{H}_{ij0}\rangle \end{aligned} \tag{82}$$

Practically, $s \ll 1$ so $1-st \approx 1$. Equation (82) can be approximated by

$$-\langle(s(x-x_0)+p)\frac{\partial I}{\partial x} + (s(y-y_0)+q)\frac{\partial I}{\partial y}, \overline{H}_{ij0}\rangle \text{ or} -\langle(sx+u)\frac{\partial I}{\partial x} + (sy+v)\frac{\partial I}{\partial y}, \overline{H}_{ij0}\rangle \tag{83}$$

Using Theorem 1 again, we derive

$$-uI_{(i+1)j0} - vI_{i(j+1)0} - s\langle\frac{\partial I}{\partial x}, x\overline{H}_{ij0}\rangle - s\langle\frac{\partial I}{\partial y}, y\overline{H}_{ij0}\rangle. \tag{84}$$

Equation (84) and (81) yield

$$-uI_{(i+1)j0} - vI_{i(j+1)0} - (i+j)sI_{ij0} - s\sigma^2(I_{(i+2)j0} + I_{i(j+2)0}) \tag{85}$$

The last term in (85) involves higher order differentiation, which often suffers from quantization error due to limited filter support. Furthermore, it is very small in smooth images. We choose to ignore it in practice. Hence, we have proved equation (64).

References

[1] Abdou, I.E. and Pratt, W.K., "Quantitative Design and Evaluation of Enhancement Thresholding Edge Detectors", Proceedings of IEEE, vol. 67, no. 5, pp. 753-763, 1979.

[2] Adiv, G., "Inherent Ambiguities in Recovering 3-D Motion and Structure from a Noisy Flow Field", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 5, pp. 477-489, 1989.

[3] Adiv, G., "Determining Three-Dimensional Motion and Structure from Optical Flow Generated by Several Moving Objects", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 7, no. 4, pp. 384-401, 1985.

[4] Allen, P., Timcenko, A., Yoshimi, B. and Michelman, P., "Automated Tracking and Grasping of a Moving Object with a Robotic Hand-Eye System", IEEE Transactions on Robotics and Automations, vol. 9, no. 2, pp. 152-165, 1993.

[5] Aloimonos, Y. and Duric, Z., "Estimating the Heading Direction Using Normal Flow", International Journal of Computer Vision, vol. 13, no. 1, pp. 33-56, 1994.

[6] Anandan, P., "Measuring Visual Motion from Image Sequences", Ph.D. Thesis, COINS TR 87-21, University of Massachusetts, Amherst MA, 1987.

[7] Ancona, N. and Poggio, T., "Optical Flow from 1-D Correlation: Applications to a Simple Time-to-Crash Detector", Proceedings of IEEE International Conference on Computer Vision, Berlin, Germany, pp. 209-214, 1993.

[8] Artieri, A. and Jutand, F., "A Versatile and Powerful Chip for Real-Time Motion Estimation", Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, Glasgow, UK, pp. 2453-2456, 1989.

[9] Barron, J. L., Fleet, D. J. and Beauchemin, S. S., "Performance of Optical Flow Techniques", International Journal of Computer Vision, vol. 12, no. 1, pp. 43-77, 1994.

[10] Bergen, J.R., Burt, P.J., Hingorani, R. and Peleg, S., "A Three-Frame Algorithm for Estimating Two-Component Image Motion", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 9, pp. 886-896, 1992.

[11] Bober, M. and Kittler, J., "Robust Motion Analysis", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Seattle, WA, pp. 947-952, 1994.

[12] Braddick, O.J., "A Short-Range Process in Apparent Motion", Vision Research, vol. 14, pp. 519-527, 1974.

[13] Broida, T.J. and Chellappa, R., "Estimation of Object Motion Parameters from Noisy Images", IEEE Transactions on Pattern Analysis and Machine Intelligence,

vol. 8, no. 1, pp. 90-99, 1986.

[14] Bruss A.R. and Horn, B.K., "Passive Navigation", Computer Vision, Graphics and Image Processing: Image Understanding, vol. 21, no. 1, pp. 3-20, 1983.

[15] Burlina, P. and Chellappa, R., "Time-to-X: Analysis of Motion Through Temporal Parameters", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 461-468, Seattle, WA, 1994.

[16] Burt, P., Hingorani, R. and Kolczynski, R., "Mechanisms for Isolating Component Patterns in the Sequential Analysis of Multiple Motion", Proceedings of the IEEE Workshop on Visual Motion, Princeton, NJ, pp.187-195, 1991.

[17] Bülthoff, H., Little, J. and Poggio, T., "A Parallel Algorithm for Real-Time Computation of Optical Flow", Nature, vol. 337, pp. 549-553, 1989.

[18] Buxton, B.F. and Buxton, H., "Monocular Depth Perception from Optical Flow by Space Time Signal Processing", Proceedings of the Royal Society of London, vol. B 218, pp. 27-47, 1983

[19] Campani, M. and Verri, A., "Motion Analysis from First-Order Properties of Optical Flow", CVGIP: Image Understanding, vol. 50, no. 1, pp. 90-107, 1992.

[20] Camus, T., "Calculating Time-to-Contact Using Real-Time Quantized Optical Flow", NISTIR-5609, Gaithersburg, MD, 1995.

[21] Camus, T., "Real-Time Quantized Optical Flow", Proceedings of IEEE conference on Computer Architectures for Machine Perception, Como, Italy, 1995.

[22] Canny, J., "A Computational Approach to Edge Detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, no.11, pp. 679-698, 1986.

[23] Chou, W-S. and Chen, Y-C., "Estimation of the Velocity Field of Two-Dimensional Deformable Motion", Pattern Recognition, vol. 26, no. 2, pp. 351-364, 1993.

[24] Cipolla, R. and Blake, A., "Surface Orientation and Time to Contact from Image Divergence and Deformation", Proceedings of IEEE European Conference on Computer Vision, Italy, pp. 187-202, 1992.

[25] Coombs, D., Herman M., Hong T. and Nashman, M., "Real-time Obstacle Avoidance Using Central Flow Divergence and Peripheral Flow", Proceedings of IEEE International Conference on Computer Vision, Cambridge, MA, 1995.

[26] Darrel, T. and Pentland, A., "Robust Estimation of a Multi-Layered Motion Representation", Proceedings of IEEE Workshop on Visual Motion, Princeton, NJ, pp. 173-178, 1991.

[27] Davis, L., Wu, Z. and Sun, H., "Contour-Based Motion Estimation", University of Maryland TR-1179, 1982.

[28] Delbruck, T., "Silicon Retina with Correlation-Based Velocity-Tuned Pixels", IEEE Transactions on Neural Networks, vol. 4, no. 3, pp. 529-541, 1993.

[29] Dengler, J., "Estimation of Discontinuous Displacement Vector Fields with the

Minimum Description Length Criterion", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Lahaina, HI, pp. 276-282, 1991.

[30] Duncan, J. and Chou, T., "On the Detection of Motion and the Computation of Optical Flow", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 3, pp. 346-352, 1989.

[31] Ens, J. and Li, Z.N., "Real-Time Motion Stereo", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp.130-135, New York, NY, 1993.

[32] Etienne-Cummings, R., Fernando, S.A., Van der Spiegel, J. and Mueller, P., "Real-Time 2D Analog Motion Detector VLSI Circuit", Proceedings of IEEE International Joint Conference on Neural Networks, vol. 4, pp. 426-431, New York, NY, 1992.

[33] Fang, J.Q. and Huang, T.S., "Solving Three Dimensional Small-Rotation Motion Equation", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC., pp. 253-258, 1983.

[34] Fleet, D.J. and Jepson, A.L., "Computation of Component Image Velocity from Local Phase Information", International Journal of Computer Vision, vol. 5, no.1, pp. 77-104, 1990.

[35] Giardina, C. and Dougherty, E.R., "Morphological Methods in Image and Signal Processing", Prentice Hall, Englewood Cliffs, NJ, 1988.

[36] Gibson, J., "The Senses Considered as Perceptual Systems", Houghton-Mifflin, Boston, MA, 1966.

[37] Girosi, F., Verri, A. and Torre, V., "Constraints for the Computation of Optical Flow", Proceedings of IEEE Workshop on Visual Motion, Irvine, CA, pp. 116-124, 1989.

[38] Grzywacz, N.M. and Yuille A.L., "A Model for Estimate of Local Image Velocity by Cells in the Visual Cortex", Proceedings of the Royal Society of London, vol. A 239, pp. 129-161, 1990.

[39] Guissin, R. and Ullman, S., "Direct Computation of Focus of Expansion from Velocity Field Measurements", Proceedings of the IEEE Workshop on Visual Motion, Princeton, NJ, pp. 146-155, 1991.

[40] Gupta, N., "Recovering Shape and Motion from a Sequence of Images", PH.D. Dissertation, University of Maryland, 1993.

[41] Haralick, R. M. and Lee, J. S., "The Facet Approach to Optic Flow", Proceedings of DARPA Image Understanding Workshop, Arlington VA, pp. 84-94, 1983.

[42] Hartley, R., "Segmentation of Optical Flow Fields by Pyramid Linking", Pattern Recognition Letters, vol.3, no. 5, pp.253-262, 1985.

[43] Hashimoto, M. and Sklansky, J., "Multiple-Order Derivatives for Detecting Local Image Characteristics", Computer Vision, Graphics and Image Processing, vol. 39,

no. 1, pp. 28-55, 1987.

[44] Heeger, D. J., "Optical Flow Using Spatiotemporal Filters", International Journal of Computer Vision, vol. 1, no. 4, pp. 279-302, 1988.

[45] Heitz, F. and Bouthemy, P., "Multimodal Estimation of Discontinuous Optical Flow Using Markov Ransom Fields", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, no.12, pp. 1217-1232, 1993.

[46] Heyden, F., "Evaluation of Edge Detection Algorithms", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.618-622, 1992.

[47] Hildreth, E., "The Measurement of Visual Motion", MIT Press 1984.

[48] Horn, B. K. P. and Schunck, B. G., "Determining Optical Flow", Artificial Intelligence, vol. 17, pp. 185-204, 1981.

[49] Horn, R.A. and Johnson, C.R., "Topics in Matrix Analysis", Cambridge University Press, Cambridge, UK, Section 6.1, pp. 382-397, 1991.

[50] Inoue, H., Tachikawa, T. and Inaba, M., "Robot Vision System with a Correlation Chip for Real-Time Tracking, Optical Flow and Depth Map Generation", Proceedings IEEE Conference on Robotics and Automation, Nice, France, vol. 2, pp. 1621-1626, 1992.

[51] Jain, J.R. and Jain, A.K., "Displacement Measurement and Its Applications in Interframe Image Coding", IEEE Transactions on Communications, vol. 29, no. 12, pp. 1799-1808, 1981.

[52] Ja'Ja', J., "An Introduction to Parallel Algorithms", Addison-Wesley, Reading, MA, 1991.

[53] Jepson, A. and Black, M., "Mixture Model for Optical Flow Computation", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, New York, NY, pp.760-761, 1993.

[54] Kearney, J. K., Thompson, W. B. and Boley, D. L., "Optical Flow Estimation: An Error Analysis of Gradient Based Methods With Local Optimization", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, no. 2, pp. 229-244, 1987.

[55] Kersten, D. "Transparency and the Cooperative Computation of Scene Attributes", Computational Models of Visual Processing, ed. by Landy, M., Movshon, J.A., pp. 209-228, MIT Press, 1991.

[56] Koch, C., Marroquin, J. and Yuille, A., "Analog 'Neural' Networks in Early Vision", Proceedings of the National Academy of Sciences, vol. 83, pp. 4263-4267, 1986.

[57] Koenderink, J.J. and Van Doorn, A.J., "Representation of Local Geometry in the Visual System", Biological Cybernetics, vol. 55, pp. 367-375, 1987.

[58] Koenderink, J.J., "Optic Flow", Vision Research, vol. 26, no. 1, pp. 161-180, 1986.

[59] Kundur, S.R. and Raviv, D., "An Image-Based Texture-Independent Visual Motion Cue for Autonomous Navigation", NISTIR 5567, April 1995.

[60] Langley, K., Fleet, D. and Atherton, T., "Multiple Motion from Instantaneous Frequency", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 846-849, 1992.

[61] Lee, J.C., Sheu, B. J., Fang, W.C. and Chellappa, R., "VLSI neuroprocessors for Video Motion Detection", IEEE Transactions on Neural Networks, vol. 4, no. 2, pp. 178-191, 1993.

[62] Liu, H., Hong, T., Herman, M. and Chellappa, R., "A Reliable Optical Flow Algorithm Using 3-D Hermite Polynomials", NIST-IR 5333, December, 1993.

[63] Liu, H., Hong, T., Herman, M. and Chellappa, R., "A Generalized Motion Model for Estimating Optical Flow Using 3-D Hermite Polynomials", Proceedings of the IEEE International Conference on Pattern Recognition, Jerusalem, Israel, pp. 360-366, 1994.

[64] Liu, H., Hong, T., Herman, M. and Chellappa, R., "Motion-Model-Based Boundary Extraction", To appear in the Proceedings of the IEEE International Symposium on Computer Vision, Coral Gable, FL, 1995.

[65] Liu, H., Hong, T., Herman, M. and Chellappa, R., "A General Motion Model and Spatio-temporal Filters for Computing Optical Flow", University of Maryland TR - 3365, November, 1994; NIST-IR 5539, Gaithersburg MD, January, 1995

[66] Liu, H., Hong, T., Herman, M. and Chellappa, R., "A General Motion Model and Spatio-temporal Filters for Computing Optical Flow", to appear in International Journal of Computer Vision.

[67] Longuet-Higgins, H.C. and Prazdny, K., "The Interpretation of a Moving Retinal Image", Proceedings of the Royal Society of London, vol. B 208, pp. 385-397, 1980.

[68] Lucas, B. D. and Kanade, T., "An Iterative Image Registration Technique with an Application to Stereo Vision", Proceedings of the DARPA Image Understanding Workshop, pp.121-130, 1981.

[69] Matteucci, P., Regazzani, C.S. and Foresti, G.L., "Real-Time Approach to 3-D Object Tracking in Complex Scenes", Electronics Letters, vol. 30, no. 6, pp. 475-477, 1994.

[70] Meyer, F. and Bouthemy, P., "Estimation of Time-to-Collision Maps from First Order Motion Model and Normal Flows", Proceedings of IEEE International Conference on Pattern Recognition, pp. 78-82, 1992.

[71] Mioni, A., "VLSI Vision Chips Homepage", http://www.eleceng.adelaide.edu.au/Groups/GAAS/Bugeye/visionchips.

[72] Mioni, A., Bouzerdoum, A., Yakovleff, A., Abbott, D., Kim, O., Eshraghian, K. and Bogner, R.E., "An Analog Implementation of Early Visual Processing in Insects", Proceedings International Symposium on VLSI Technology, Systems, and Applications, pp. 283-287, 1993.

[73] Moore, A. and Koch C., "A Multiplication Based Analog Motion Detection Chip", Proceedings of the SPIE, vol. 1473, Visual Information Processing: From Neurons to Chips, pp. 66--75, 1991.

[74] Murray, D. and Buxton, B., "Scene Segmentation from Visual Motion Using Global Optimization", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, no. 2, pp. 220-228, 1987.

[75] Mutch, K. and Thompson, W., "Analysis of Accretion and Deletion at Boundaries in Dynamic Scenes", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 7, no.2, pp. 133-138, 1985.

[76] Nagel, H. H., "Constraints for the Estimation of Displacement Vector Fields from Image Sequences", Proceedings of International Joint Conference on Artificial Intelligence, pp. 945-951, 1983.

[77] Nagel, H. H., "Direct Estimation of Optical Flow and of its Derivatives", Artificial and Biological Vision Systems, Ed. Orban, G.A., Nagel, H. H., pp. 193-224, 1992.

[78] Nagel, H. H., "Displacement Vectors Derived from Second-order Intensity Variations in Image Sequences", Computer Vision, Graphics and Image Processing, vol. 21, no. 1, pp. 85-117, 1983.

[79] Nakayama, K. and Loomis, J.M., "Optical Velocity Patterns, Velocity Sensitive Neurons and Space Perception: a Hypothesis", Perception, vol. 3, 1974.

[80] Nashman, M., Rippey, W., Hong, T.-H. and Herman, M., "An Integrated Vision Touch-Probe System for Dimensional Inspection Tasks", NISTIR 5678, National Institute of Standards and Technology, 1995.

[81] Negahdaripour, S. and Horn, B.K., "Direct Passive Navigation", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, no. 1, pp. 168-176, 1987.

[82] Negahdaripour, S. and Lee, S., "Motion Recovery from Image Sequences using First-Order Optical Flow Information", Proceedings of IEEE Workshop on Visual Motion, Princeton, pp. 132-139, 1991.

[83] Nelson, R., "Qualitative Detection of Motion by a Moving Observer", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Lahaina, HI, pp. 173-178, 1991.

[84] Nelson, R. and Aloimonos, J., "Obstacle Avoidance Using Flow Field Divergence", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 10, pp. 1102-1106, 1989.

[85] Nesi, P., DelBimbo, A.D. and Ben-Tzvi, D. "A Robust Algorithm for Optical Flow Estimation", Computer Vision and Image Understanding, vol. 62, no. 1, pp. 59-68, 1995.

[86] Nishihara, H.K., "Real-Time Stereo- and Motion-Based Figure Ground Discrimination and Tracking Using LOG Sign Correlation", Conference Record of the Twenty-Seventh Asilomar Conference on Signals, Systems and Computers, vol. 1,

pp. 95-100, 1993.

[87] Patterson, D. and Hennessy, J., "Computer Organization and Design: the Hardware/Software Interface", Morgan Kaufman, San Mateo, CA 1994.

[88] Poggio, T., Verri, A. and Torre, V., "Green Theorems and Qualitative Properties of the Optical Flow", AI Memo 1289, MIT AI Laboratory, 1991.

[89] Potter, J. L., "Velocity as a Cue to Segmentation Using Motion Information", IEEE Transactions on Systems, Men, Cybernetics, vol. 5, pp. 390-395, 1975.

[90] Prazdny, K., "Egomotion and Relative Depth Map From Optical Flow", Biological Cybernetics, vol. 36, pp. 87-102, 1980.

[91] Prazdny, K., "Determining the Instantaneous Direction of Motion from Optical Flow Generated by Curvilinearly Moving Observer", Proceedings of IEEE Conference on Pattern Recognition and Image Processing, pp. 82-87, Dallas, TX, 1981.

[92] Rangachar, R., Hong, T.-H., Herman, M., and Luck, R., "Three Dimensional Reconstruction from Optical Flow Using Temporal Integration", Proceedings of SPIE Advances in Intelligent Robotic Systems: Intelligent Robots and Computer Vision, Boston, MA, 1990.

[93] Schiff, W. and Caviness, J.A. and Gibson, J.J., "Persistent Fear Response in Rhesus Monkeys in Response to the Optical Stimulus of Looming", Science, vol. 136, pp. 982-983, 1962.

[94] Schunck, B., "Image Flow Segmentation and Estimation by Constraint Line Clustering", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no.10, pp. 1010-1027, 1989.

[95] Shizawa, M. and Mase, K., "Principles of Superposition: A Common Computational Framework for Analysis of Multiple Motion", Proceedings of the IEEE Workshop on Visual Motion, Princeton, NJ, pp.164-172, 199.

[96] Shizawa, M. and Mase, K., "A Unified Computational Theory for Motion Transparency and Motion Boundaries Based on Eigenenergy Analysis", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Lahaina, HI, pp.289-295, 1991.

[97] Singh, A., "An Estimation-Theoretic Framework for Image-Flow Computation", Proceedings of the International Conference on Computer Vision, pp. 169-177, Osaka, Japan, 1990.

[98] Singh, A., "Optical Flow Computation: A Unified Perspective", IEEE Computer Society Press, 1991.

[99] Smith, S.M., "ASSET-2 Real-Time Motion Segmentation and Object Tracking", Proceedings of the Fifth International Conference on Computer Vision, pp. 237-244, Cambridge, MA, 1995.

[100] Srinivasan, M.V., "Generalized Gradient Schemes for the Measurement of Two-Dimensional Image Motion", Biological Cybernetics, vol. 63, pp. 421-431, 1990.

[101] Spoerri, A. and Ullman, S., "The Early Detection of Motion Boundaries", A.I. Memo no. 935, AI Lab, MIT, 1987.

[102] Stewart, G.W., "Introduction to Matrix Computation", Academic Press, New York, 1973.

[103] Subbarao, M., "Bounds on Time-to-Collision and Rotational Component from First Order Derivatives of Image Flow", Computer Vision, Graphics and Image Processing: Image Understanding, vol. 50, no. 1, pp. 329-341, 1990.

[104] Tanner J. and Mead C., "A Correlating Optical Motion Detector", MIT Advanced Research in VLSI, pp. 57--64, 1984.

[105] Tanner, J. and Mead, C., "An Integrated Analog Optical Motion Sensor", VLSI Signal Processing II, R.W. Brodersen and H.S. Moscovitz, Eds., pp. 59-87, New York, 1988.

[106] Thompson, W., Mutch, K. and Berzins, V., "Dynamic Occlusion Analysis in Optical Flow Fields", IEEE Transactions on Pattern Recognition and Machine Intelligence, vol. 7, no.4, pp. 374-383, 1985.

[107] Thompson, W.B. and Kearney, J.K., "Inexact Vision", Proceedings of the Workshop on Motion, Representation, and Analysis, pp.15-21, May 1986.

[108] Tsai, R.Y., Huang, T.S. and Zhu, W.L., "Estimating Three-Dimensional Motion Parameters of a Rigid Planar Patch II: Singular Value Decomposition", IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 30, no. 4, pp. 525-534, 1982.

[109] Uras, S., Girosi, F., Verri, A. and Torre, V. "A Computational Approach to Motion Perception", Biological Cybernetics, vol. 60, pp. 79-97, 1988.

[110] Verri, A. and Poggio, T. "Motion Field and Optical Flow: Qualitative Properties", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 5, pp. 490-498, 1989.

[111] Wang, J. and Adelson, E., "Layered Representation for Motion Analysis", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York, NY, pp. 361-366, 1993.

[112] Waxman, A.M., Wu J. and Bergholm F. "Convected Activation Profiles and Receptive Fields for Real Time Measurement of Short Range Visual Motion", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Ann Arbor, MI, pp. 717-723, 1988.

[113] Weber, J. and Malik, J., "Robust Computation of Optical Flow in a Multi-Scale Differential Framework", Proceedings of the Fourth International Conference on Computer Vision, Berlin, Germany, 1993.

[114] Weber, J. and Malik, J., "Robust Computation of Optical Flow in a Multi-Scale Differential Framework", International Journal of Computer Vision, vol. 14, pp. 67-81, 1995.

[115] Weiss, P. and Christensson, B., "Real Time Implementation of Subpixel Motion Estimation for Broadcast Applications", IEE Colloquium on Applications of Motion Compensation, London, UK, no. 128, pp. 7/1-3, 1990.

[116] Werkhoven, P. and Koenderink, J.J., "Extraction of Motion Parallax Structure in the Visual System I", Biological Cybernetics, vol. 63, pp. 185-191, 1990.

[117] Woodfill, J. and Zabin, R., "An Algorithm for Real-Time Tracking of Non-Rigid Objects", Proceedings of the Ninth National Conference on Artificial Intelligence, pp. 718-723, 1991.

[118] Yagi, Y., Kawato, S. and Tsuji, S., "Real-Time Omnidirectional Image Sensor (COPIS) for Vision-Guided Navigation", IEEE Transactions on Robotics and Automation, vol. 10, no. 1, pp. 11-22, 1994.

[119] Yamamoto, M., "A Segmentation Method Based on Motion from Image Sequence and Depth", Proceedings of IEEE International Conference on Pattern Recognition, Atlantic City, pp. 230-232, 1990.

[120] Young, G.J. and Chellappa, R., "3-D Motion Estimation Using a Sequence of Noisy Stereo Images: Models, Estimation and Uniqueness Results", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 8, pp. 735-759, 1990.

[121] Young, G.J. and Chellappa, R., "Statistical Analysis of Inherent Ambiguities in Recovering 3-D Motion from a Noisy Flow Field", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 10, pp. 995-1013, 1992.

[122] Young, G-S., "Safe Navigation and Active Vision for Autonomous Vehicles: A Purposive and Direct Solution" Ph.D. Dissertation, University of Maryland, May 1993.

[123] Young, G-S., Hong, T., Herman, M. and Yang, J., "New Visual Invariant for Obstacle Detection Using Optical Flow Induced from General Motion", Proceedings of the IEEE Workshop on Applications of Computer Vision, Palm Springs, CA, pp. 100-109, 1992.

[124] Young, G-S., Hong, T., Herman, M. and Yang, J., "Safe Navigation and Active Vision for Autonomous Vehicles: A Purposive and Direct Solution", Proceedings of the SPIE, vol. 2056: Active Vision and 3D Methods, Boston, MA, pp. 31-42, 1993.

[125] Young, R. A.,"Simulation of Human Retinal Function with the Gaussian Derivative Model", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, FL, pp. 564-569, 1986.

[126] Zheng, Q. and Chellappa, R., "A Computational Vision Approach to Image Registration", IEEE Transactions on Image Processing, vol. 2, no. 7, pp. 311-326, 1993.