Projection-based, frequency-domain estimation of superimposed translational motions

Peyman Milanfar

SRI International, 333 Ravenswood Avenue, Mail Stop 404-69, Menlo Park, California 94025

Received June 19, 1995; revised manuscript received May 16, 1996; accepted June 18, 1996

In this paper a description is given of a computationally efficient algorithm, based on the two-dimensional fast Fourier transform (FFT), for the estimation of multiple translational motions from a sequence of images. The proposed algorithm relies on properties of the projection (Radon) transform to reduce the problem from three to two dimensions and is effective in isolating and reliably estimating several superimposed motions in the presence of moderate levels of noise. Furthermore, the reliance of this algorithm on a novel array processing technique for line detection allows for the efficient estimation of the motion parameters. It is shown that while the technique presented herein is not expected to exhibit the same performance as that of comparable techniques based on the three-dimensional FFT, it is an attractive alternative that makes modest sacrifices in performance for gains in computational complexity. © 1996 Optical Society of America.

1. INTRODUCTION

The estimation of image motion finds applications in a wide variety of areas ranging from machine vision to target detection and tomography. In some machine vision applications the apparent relative motion between a scene and a camera is captured within an image sequence, and the aim may be ultimately to recover the three-dimensional (3-D) motion, and even the shape, of the scene. This process typically involves the computation of estimates of motion vectors on a fine grid from a sequence of images.[1] In particular, the estimation of multiple superimposed motions such as those found in specular and film transparencies is important in a variety of applications.[2–4] Aerial or spaceborne photography and videography are practical examples in which the need for fast, real-time, multiple-motion estimation is great. The application areas are numerous and include meteorological monitoring of clouds and storms from satellite imagery and detection and tracking of dim airborne or ground-based targets from down-looking imaging sensors. In such applications the image sequence is acquired from a moving platform, while the targets of interest may be moving along various directions. Simultaneously, multiple layers of cloud cover, moving in yet other directions, may occlude the target by transparently adding to the image. Furthermore, in such applications, the energy of the motion signal in the cloud component is typically quite different from that corresponding to the targets or the ground because of different texture patterns found in these images. Hence the strongest motion signal may occlude or mask the weaker signals and render these imperceptible. In this paper a fast algorithm for multiple-motion estimation is provided that takes these issues into account, and its effectiveness is demonstrated on simulated and real sequences of aerial imagery.

The wide set of applications encompassing the motion estimation problem has resulted in much research work in the past two decades. Most available algorithms dealing with motion estimation can be roughly categorized as belonging to one of two general classes. The first involves motion estimation directly in the image domain.[5] These algorithms include the celebrated Horn and Schunk[1] algorithm for optical flow and the many variants of this work that have been introduced since. The essence of this algorithm is the assumption that, locally, the motion within an image can be described by one simple translational component if the local neighborhood is sufficiently small.[3] When this single-component local velocity assumption is violated, as is the case in transparent motion of two overlapping patterns in the same scene, other image-domain algorithms have been developed to isolate each of the velocities.[3,6,7]

The second class of motion estimation algorithms consists of frequency-domain approaches. The basic premise of these techniques is the idea that if a sequence of images, thought of as a 3-D function [in two-dimensional (2-D) space and time], contains a uniformly translating pattern or object, then the energy of the 3-D fast Fourier transform (FFT) of this function will be mostly concentrated along a plane through the origin whose orientation is related to the velocity of the moving pattern.[8–10] Hence, by computing the 3-D FFT and then finding one or more planes with strong energy concentration, one can estimate the desired velocity.[9] An important advantage of this technique is that because of the linear superposition of the Fourier transform, the existence of multiple superimposed motions will be manifested in the FFT magnitude simply as energy concentrations along more than one plane through the origin.

Even though the 3-D FFT approach is computationally simpler than most image-domain algorithms, the next step of finding planes with dominant energy, which typically involves matched filtering with a bank of directional filters,[9,10] is not simple, particularly if multiple motions are present. While these 3-D spectral techniques have been shown to be effective, they still involve a significant amount of computation, in addition to heuristic reason-

0740-3232/96/1102151-12$10.00

ing, in deciding on a particular detection criterion. Other techniques have also been developed that combine frequency and spatiotemporal analysis[11] in improving computational complexity and resolution of the existing techniques.

In this paper a novel (2-D) algorithm for projection-based multiple-motion estimation is described. The proposed algorithm can efficiently estimate several velocities in batch or in an ordered (sequential) fashion similar to that of Ref. 6 and is robust in the presence of moderate levels of noise. Beginning with a sequence of frames, the algorithm first projects each frame along two linearly independent directions. Then, 2-D FFT's are applied to the resulting collection of projections. Next, a line detection algorithm estimates the component velocities, and, finally, the component velocities are matched to produce the motion vectors.

As compared with existing techniques for (translational) motion estimation, several aspects of the present approach are novel. In contrast to iterative techniques such as those in Refs. 3 and 6, which can estimate up to two motions from up to three frames, the proposed algorithm is noniterative and, given sufficiently many frames of images, can estimate as many superimposed velocities as the resolution of the data will allow.

Various researchers have studied the use of 2-D FFT's in various forms for estimating motion.[11–16] However, no investigators have, to date, applied their techniques to the estimation of multiple superimposed motions. In a technique[12] related to that of the present paper the authors use projections along the x and y directions only and compute 2-D FFT's, but rather than finding a line in the 2-D FFT's, they estimate velocities from a single peak in one row (spatial frequency bin) of the spectrum. The present 2-D technique follows from the observation that any pair of linearly independent components of a velocity vector [in particular, the horizontal (x) and vertical (y) components] can be estimated independently. This follows from a well-known property of the Radon transform that will be described in Section 2.

While I do not expect the performance of the proposed 2-D algorithm to match that of the corresponding algorithms based on 3-D FFT's and matched filtering, I will demonstrate that it offers an attractive alternative for the estimation of superimposed motions with high quality, but at potentially significant computational savings. To this end, I have also adopted a recently published technique for the estimation of lines in images.[17,18] This technique (called SLIDE, for subspace-based line detection) is based on a clever analogy between lines and propagating wave fronts and employs efficient array processing algorithms. Subsequent to the submission of this paper, I was made aware that in Ref. 19 the authors had independently suggested the use of projection followed by (subspace-based) line fitting for motion estimation.

2. PROBLEM FORMULATION AND SOLUTION OUTLINE

Assume that a uniformly sampled sequence of frames $f(x, y, t)$, where all three arguments are discrete and range over $1 \leq x, y \leq N$ and $0 \leq t \leq T-1$, is given. For the moment a single displacement vector $v = (v_x, v_y)^T$ is assumed unknown. Writing $f(x, y, t) = f(x - tv_x, y - tv_y, 0)$, we define the difference frames $d(x, y, t)$ for $1 \leq t \leq T-1$ as

$$d(x, y, t) = f(x, y, t) - f(x - v_x, y - v_y, t - 1). \tag{1}$$

The least-squares (LS) error approach to estimating the displacement[20] vector v is to minimize the following cost function of v_x and v_y:

$$J(v_x, v_y) = \sum_{t,x,y} d^2(x, y, t). \tag{2}$$

Pixel-domain techniques known as optical flow techniques[1] assume that the displacements are sufficiently small across consecutive frames so that $d(x, y, t)$ can be approximated by a truncated Taylor series. Differentiation of the resulting approximate expression for the cost, with respect to the unknown displacements, yields the optical flow equations:

$$v_x \sum \left(\frac{\partial f}{\partial x}\right)^2 + v_y \sum \left(\frac{\partial f}{\partial x}\frac{\partial f}{\partial y}\right) = -\sum \frac{\partial f}{\partial x}\frac{\partial f}{\partial t}, \tag{3}$$

$$v_x \sum \frac{\partial f}{\partial x}\frac{\partial f}{\partial y} + v_y \sum \left(\frac{\partial f}{\partial y}\right)^2 = -\sum \frac{\partial f}{\partial y}\frac{\partial f}{\partial t}. \tag{4}$$

Solving these equations for v_x and v_y gives the estimated displacement vector $\hat{v}$. Iterative improvements to the resulting estimates can be obtained as described in Refs. 3 and 21.

On the other hand, from Parseval's relation we can write the right-hand side of Eq. (2) in the frequency domain as follows:

$$\begin{aligned} J(v_x, v_y) &= c \sum_{\omega_t, \omega_x, \omega_y} |D(\omega_x, \omega_y, \omega_t)|^2 \\ &= 2 \sum_{\omega_t, \omega_x, \omega_y} |F(\omega_x, \omega_y, \omega_t)|^2 \\ &\quad \times [1 - \cos(v_x\omega_x + v_y\omega_y + \omega_t)], \end{aligned} \tag{5}$$

where c is a constant and capital letters refer to the discrete Fourier transform of the corresponding lowercase variables. From Eq. (5) one can immediately see that the minimum of the cost function (=0) is attained when $1 - \cos(v_x\omega_x + v_y\omega_y + \omega_t) = 0$, which yields the fundamental solution

$$v_x\omega_x + v_y\omega_y + \omega_t = 0. \tag{6}$$

This demonstrates that as a result of the displacement $v = (v_x, v_y)^T$, essentially all of the energy in the 3-D spectrum of $f(x, y, t)$ will be concentrated along a plane given by Eq. (6), which passes through the origin of the 3-D frequency domain, and that the orientation of this plane yields the LS estimate of the displacement vector.[22]

An equivalent way of describing the plane (6) is to specify two independent vectors (lines) that span it. These vectors can be given by considering the intersection of Eq. (6) with two nonparallel planes, e.g., $\omega_x = 0$ and $\omega_y = 0$. One attractive way to accomplish this and hence reduce the dimensionality of the problem to two

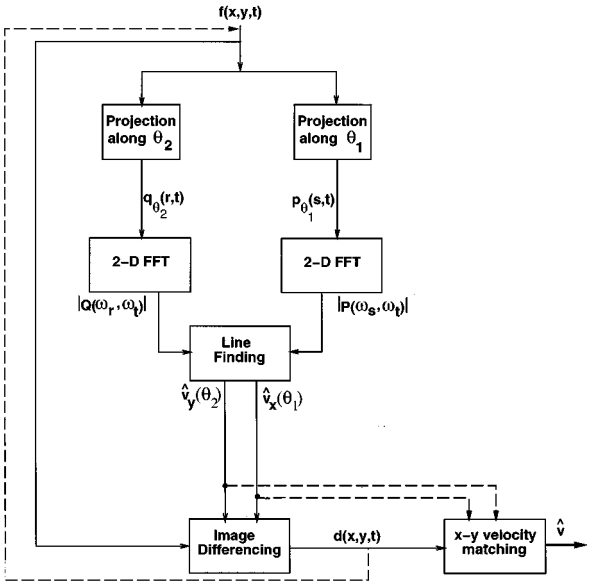

Fig. 1. Overview of the algorithm.

dimensions is to project each frame along a pair of independent directions and then apply 2-D FFT's. The celebrated projection slice theorem[23] implies that the 2-D FFT's of the resulting projections are slices through the 3-D FFT's of the image sequence, where these slices are taken along planes determined by the projection directions. In particular, when the projections are taken along the rows and the columns of the images, the slices are along the planes $\omega_x = 0$ and $\omega_y = 0$. Hence the energy of the resulting 2-D spectra will be concentrated along lines instead of planes in 3-D processing.

Let us define the projections onto the x and y axes of each frame, respectively, as follows:

$$p(x, t) = \sum_y f(x, y, t), \tag{7}$$

$$q(y, t) = \sum_x f(x, y, t). \tag{8}$$

It then follows that $p(x, t)$ undergoes a motion of v_x samples per frame and, similarly, $q(y, t)$ undergoes a motion of v_y samples per frame. This observation has been made elsewhere in past motion estimation literature.[12] A simple fact, not discussed before in the motion estimation literature, is that this is a special case of a more general (shift) property of the Radon (projection) transform.[23] This property states that if $f(x, y)$ is a 2-D image and $p_\theta(s)$ denotes its projection onto an axis (s) forming an angle θ with the x axis, then the projection of a shifted version of the image, $f(x - v_x, y - v_y)$, onto the same axis is given by $p_\theta(s - v_x \cos\theta - v_y \sin\theta)$. This fundamental property forms the basis of the present projection-based algorithm.

The proposed algorithm for motion estimation involves several simple steps, as raised in Section 1. These steps are illustrated in Fig. 1. As mentioned in Section 1, when multiple velocities are to be estimated, the approach allows for the estimation of one velocity at a time. In particular, if this option is exercised, the stronger velocity components (in terms of spectral energy) are estimated and removed first. In this fashion the signal-to-noise ratio (SNR) for weaker velocity components is enhanced, and hence their detection is improved.

Referring to Fig. 1, the first step in the algorithm is projection along two independent directions θ_1 and θ_2. If we denote the velocity components in these directions as $v_x(\theta_1)$ and $v_y(\theta_2)$, respectively, then we have

$$\begin{pmatrix} v_x(\theta_1) \\ v_y(\theta_2) \end{pmatrix} = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 \\ \cos\theta_2 & \sin\theta_2 \end{bmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix}. \tag{9}$$

For any nontrivial pair (θ_1, θ_2), once all velocity components have been estimated in these directions and matched (i.e., corresponding pairs have been identified), the velocity components v_x and v_y can be uniquely computed from $v_x(\theta_1)$ and $v_y(\theta_2)$ by inverting Eq. (9).[24]

3. VELOCITY ESTIMATION THROUGH LINE DETECTION

Given the magnitude of the 2-D FFT's of p and q, denoted respectively by $I_p(\omega_x, \omega_t) = |P(\omega_x, \omega_t)|$ and $I_q(\omega_y, \omega_t) = |Q(\omega_y, \omega_t)|$, we now effectively have a pair of images in which we seek to detect one or more lines. Numerous techniques have been introduced in the past to accomplish this task. Among these, the most popular and widely used are based on the Hough transform.[25–27] While the Hough transform technique is quite robust to noise, it has significant drawbacks that limit its utility.[17] The first of these drawbacks concerns resolution limitedness arising from the discrete nature of the problem. An additional drawback of the Hough transform is that the search process for multiple peaks is not only computationally burdensome but also hampered by local extrema. Voting schemes aimed at extracting peaks of the Hough transform also suffer from inherent bias as described in Ref. 28. An attractive alternative to the Hough transform is a recently introduced high-resolution technique called SLIDE,[17] which is based on a clever analogy between a line in an image and a planar propagating wave front impinging on an array of sensors. This algorithm provides a natural framework for estimation of multiple motions and requires fewer computations than the Hough transform. In fact, finding one line in an $N \times N$ image by means of the Hough transform requires $O(N^3)$ computations, whereas the same task with the use of SLIDE requires only $O(N^2)$ computations.

SLIDE begins with the assumption that in front of each row (or, equivalently, column) of an image there is a hypothetical receiver as shown in Fig. 2. Then each pixel in a particular row (or column) of the image can be thought of as contributing a received signal at the corresponding sensor. The contribution can be thought of as simply a phase delay if the pixel is regarded as a source of narrowband radiation that travels only toward the sensor. In essence, the problem of detecting a line in an image is then reduced to that of estimating the direction of arrival of the wave front, or the phase delay between consecutive sensors. Rather than describe the detailed derivation of this technique, which is based on the total LS version of ESPRIT,[17,29] I simply state the steps required to arrive at

an estimate of the slope of a line within an image. Without loss of generality, the algorithm is described in terms of an arbitrary $N \times N$ image $I(m, n)$.

The first step in SLIDE is the formation of the array data vector $z = [z(0), z(1), \ldots, z(N-1)]^{\mathrm{T}}$ as follows:

$$z(n) = \sum_{m=0}^{N-1} h(m) I(m, n), \tag{10}$$

where for $m = 0, \ldots, N-1$ we define

$$h(m) = \exp(-j\mu m). \tag{11}$$

The parameter μ (typically set to approximately unity) is a constant (related to the speed of propagation) that can be tuned to produce best performance.

The next step is the formation of the data matrix Z_P as follows:

$$Z_P = \begin{bmatrix} z(0) & z(1) & \cdots & z(N-M) \\ z(1) & z(2) & \cdots & z(N-M+1) \\ \vdots & \cdots & \ddots & \vdots \\ z(M-1) & z(M) & \cdots & z(N-1) \end{bmatrix}, \tag{12}$$

where the parameter M is chosen so as to yield the best spatial smoothing.[17]

The data covariance matrix is then formed as

$$R_{zz} = \frac{1}{P} Z_P Z_P^{\mathrm{H}}, \tag{13}$$

where the superscript H denotes conjugate transpose.

The eigendecomposition of R_{zz} yields

$$R_{zz} = E_s \Lambda_s E_s{}^{\mathrm{H}} + E_w \Lambda_w E_w{}^{\mathrm{H}}, \tag{14}$$

where the first term on the right-hand side corresponds to the signal covariance and the second term captures the noise covariance. Assuming that l signals are present in the data, to apply ESPRIT, we let E_1 denote the submatrix of E_s formed by rows 1 through $M-1$, and define E_2 as the submatrix formed by the rows 2 through M. Next, the $2l \times 2l$ matrix U is formed:

$$U = \begin{pmatrix} E_1{}^{\mathrm{H}} \\ E_2{}^{\mathrm{H}} \end{pmatrix} (E_1 \quad E_2). \tag{15}$$

The eigendecomposition of U yields

$$U = F \Lambda F^{\mathrm{H}}, \tag{16}$$

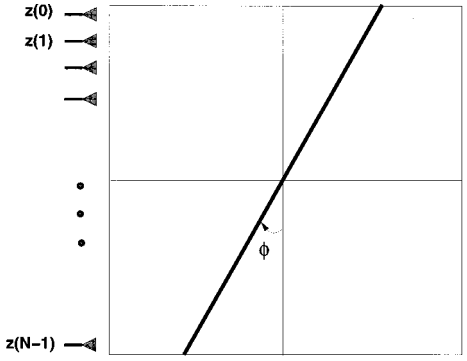

Fig. 2. Interpretation of SLIDE (adopted from Ref. 17).

where F can be partitioned into $l \times l$ submatrices as

$$F = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}. \tag{17}$$

For $k = 1, \ldots, l$ the angles ϕ_k as shown in Fig. 2 are then obtained from

$$\phi_k = \tan^{-1}\left[\frac{1}{\mu} \operatorname{Im}\left(\ln \frac{\zeta_k}{|\zeta_k|}\right)\right], \tag{18}$$

where ζ_k are the eigenvalues of $-F_{12}F_{22}{}^{-1}$. Given the angles ϕ_k, the slopes of the corresponding lines are easily found. Note that when the relative difference in strength between two motion signals is known to be large, the proposed algorithm can estimate one velocity at a time. Hence, if a total of l velocities are being sought, we would use $l = 1$ in each of l passes through the algorithm. In this fashion multiple velocities are estimated sequentially, with the strongest velocities (corresponding essentially to the largest of the eigenvalues of R_{zz}) being extracted first. If the assumption of large relative difference in the energy is not satisfied, then letting $l = 1$ will yield a linear combination of the multiple signals and the sequential approach is not appropriate. But when this relative difference is sufficiently large, then this linear combination is highly biased toward the stronger of the two motion signals. In effect, the sequential approach sacrifices some accuracy in the estimate of the strongest motion signal in exchange for a better estimate of the weaker signal.

4. VELOCITY COMPONENT MATCHING

Let us assume that application of the steps in Section 3, as depicted in Fig. 1, produces l horizontal and l vertical velocity estimates. We can denote these as

$$V_x = \{\hat{v}_x(1), \hat{v}_x(2), \ldots, \hat{v}_x(l)\}, \tag{19}$$

$$V_y = \{\hat{v}_y(1), \hat{v}_y(2), \ldots, \hat{v}_y(l)\}. \tag{20}$$

We shall assume that the true velocity components do not have any x or y components in common[30] (i.e., none of the $\hat{v}_x$'s are equal, and none of the $\hat{v}_y$'s are equal). We need to obtain l displacement vectors $v_1, v_2, \ldots, v_l$ by matching elements of V_x to those of V_y.

Assuming a given measure $L(i, j)$ for how well a pair $[\hat{v}_x(i), \hat{v}_y(j)]$ match, we can proceed by first finding the best match between every possible pair of candidate component velocities. That is to say,

$$\hat{v}_1 = [\hat{v}_x(i_1), \hat{v}_y(j_1)]^{\mathrm{T}}, \tag{21}$$

where for all $1 \leq i, j \leq l$

$$L(i_1, j_1) \leq (i, j). \tag{22}$$

This requires that we compute exactly l^2 values of L and pick the minimum. Once this has been accomplished, $\hat{v}_x(i_1)$ and $\hat{v}_y(j_1)$ are removed from the sets V_x and V_y and the next displacement vector $\hat{v}_2$ is obtained as

$$\hat{v}_2 = [\hat{v}_x(i_2), \hat{v}_y(j_2)]^{\mathrm{T}}, \tag{23}$$

where

$$L(i_2, j_2) \leq L(i, j) \tag{24}$$

for all $1 \leq i, j \leq l$ with $i \neq i_1$ and $j \neq j_1$. Note that this second optimization requires no more computations of L, since the same values computed in arriving at $\hat{v}_1$ are used again. Proceeding in this fashion, we find that the process of matching the horizontal and vertical velocities involves only l^2 computations of the measure L. Once these values are found, to obtain $\hat{v}_k$, one picks the minimum value among the remaining $(l - k + 1)^2$ values of L.

By appealing to the LS formulation introduced in Eq. (2), we choose L to be a quadratic error function. Akin to the definition of $J(v_x, v_y)$ in Eq. (2) the average per pixel error L is defined as

$$L(i, j) = \frac{1}{K} \sum_{t=1}^{K} d^2[x, y, t, \hat{v}_x(i), \hat{v}_y(j)]/N^2, \quad (25)$$

where d is defined in Eq. (1) and $1 \leq K \leq T - 1$. Note that K is allowed to be smaller than the total number of given difference frames. I have observed that the above criterion works quite well in matching velocities, even if K is only a fraction of the available total number of difference frames $(T - 1)$, while incurring a much smaller computational cost than that if all the available frames are used.

5. PERFORMANCE

In this section the performance of the proposed algorithm is assessed by way of computing approximations for the error variances of the estimates under high-SNR and small-motion assumptions.

First, without loss of generality, let us consider the case in which Gaussian white noise corrupts the frames. The measured image frames are

$$u(x, y, t) = f(x, y, t; v) + w(x, y, t), \quad (26)$$

where $v = (v_x, v_y)^{\mathrm{T}}$ and $w(x, y, t)$ denotes joint spatiotemporally white noise with zero mean and finite variance σ^2. The resulting projections calculated from the noisy frames are given by

$$p_u(x, t) = p(x, t; v_x) + w_1(x, t), \quad (27)$$

$$q_u(y, t) = q(y, t; v_y) + w_2(y, t), \quad (28)$$

where $p(x, t)$ and $q(y, t)$ are defined in Eqs. (7) and (8).

The variables w_1 and w_2 are then Gaussian with zero mean and variance $N\sigma^2$. Furthermore, $w_1(x, t)$ and $w_2(y, t)$ are both white in the spatial and temporal variables on account of the whiteness of $w(x, y, t)$. In fact, for a fixed t_0, $w_1(x, t_0)$ and $w_2(y, t_0)$ are essentially uncorrelated in the spatial variables. This follows from the observation that the sums $w_1(x, t_0) = \Sigma_{y=0}^{N-1} w(x, y, t_0)$ and $w_2(y, t_0) = \Sigma_{x=0}^{N-1} w(x, y, t_0)$ have only (exactly) one term in common for each pair (x, y). A simple calculation shows that the (spatial) correlation of w_1 and w_2 is $1/N$, which diminishes with large N.[31] This is, in fact, a special case of another interesting, deeper property, which states that the Radon transform is a whitening filter in the angular parameter when applied to stationary random fields.[26] That is to say, for a given stationary random field, in the limiting case ($N = \infty$), every pair of projections from nontrivial angles is statistically uncorrelated.

Since the temporal correlation of w_1 and w_2 is zero by assumption, in general, we can effectively state that when N is sufficiently large, the measurement equations (27) and (28) are essentially statistically uncorrelated measurements of functions of the corresponding components of the velocities. Therefore we can view the optimal estimation of the velocities in these two directions as independent estimation problems that can be solved separately. Having stated this, we investigate the relative performance of the optimum 3-D approach compared with the projection-based approach.

The problem of estimating the motion vector $v = (v_x, v_y)^{\mathrm{T}}$ from a sequence of noisy frames given in Eq. (26) is, in general, a nonlinear estimation problem. If we denote the probability-density function of the data u by $\mathbf{P}[u(x, y, t); v]$, then the Cramér–Rao lower bound (CRLB) for the variance of the estimate $\hat{v}$ is given by[32]

$$\mathrm{cov}(\hat{v}) \geq I^{-1}(v), \quad (29)$$

where $I(v)$ is the Fisher information matrix for the data:

$$I(v) = \mathrm{E}\left(\left\{ \frac{\partial \log[\mathbf{P}(u; v)]}{\partial v} \right\}^{\mathrm{T}} \left\{ \frac{\partial \log[\mathbf{P}(u; v)]}{\partial v} \right\}\right). \quad (30)$$

Noting that $\mathbf{P}$ is a Gaussian density with mean f and variance σ^2, and after some simplification, we have

$$I(v) = \frac{1}{\sigma^2} \begin{bmatrix} \sum_{x, y, t} \left(\frac{\partial f}{\partial v_x}\right)^2 & \sum_{x, y, t} \frac{\partial f}{\partial v_x} \frac{\partial f}{\partial v_y} \\ \sum_{x, y, t} \frac{\partial f}{\partial v_x} \frac{\partial f}{\partial v_y} & \sum_{x, y, t} \left(\frac{\partial f}{\partial v_y}\right)^2 \end{bmatrix}. \quad (31)$$

For high SNR's the CRLB is an accurate approximation of the actual error covariance matrix. Unfortunately, the above expression for the information matrix is not particularly illuminating, as we generally cannot compute the derivatives. We can, however, approximate the CRLB for small-motion vectors.

To obtain an approximation, we proceed by linearizing the measurements about $v = \mathbf{0}$. To this end, we first write

$$f(x, y, t; v) = f(x - v_x t, y - v_y t, 0). \quad (32)$$

The right-hand side can be expanded in a Taylor series about $v = \mathbf{0}$ to produce

$$f(x, y, t; v) \approx f(x, y, 0) - v_x t \frac{\partial f}{\partial x} - v_y t \frac{\partial f}{\partial y} + \text{higher-order terms}. \quad (33)$$

Ignoring the higher-order terms and redefining the data, we have

$$d = \frac{u(x, y, t) - f(x, y, 0)}{t} = -\begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix} + w(x, y, t). \quad (34)$$

It is noteworthy here that as t approaches zero, the left-hand side of Eq. (34) is, in essence, a noisy measurement of the partial derivative, with respect to time, of the mo-

tion sequence, i.e., $\partial f/\partial t$. With this in mind, we recognize Eq. (34) as the measurement equation that leads to the optical flow formulation. In fact, what we have now is a linear estimation problem for v. The error covariance matrix for this linear problem is given by

$$Q = \sigma^2 \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}^{-1}, \tag{35}$$

where

$$D_{xx} = \sum_{x,y,t} \left(\frac{\partial f}{\partial x} \right)^2, \tag{36}$$

$$D_{xy} = \sum_{x,y,t} \frac{\partial f}{\partial x} \frac{\partial f}{\partial y}, \tag{37}$$

$$D_{yy} = \sum_{x,y,t} \left(\frac{\partial f}{\partial y} \right)^2. \tag{38}$$

Not surprisingly, these entries coincide with the coefficients found on the left-hand side of the optical flow equations (3) and (4).

For small v, Q is an adequate approximation of the lower bound in relation (29). In turn, the CRLB itself is an accurate approximation of (or a tight bound on) the actual covariance for small noise variance σ^2. Hence we can use Q as a measure of performance for high-SNR scenarios and around $v = \mathbf{0}$. For comparison with the 2-D case the trace of Q (a scalar) is a useful measure. This is given by

$$C_3 = \mathrm{Tr}(Q) = \sigma^2 \left(\frac{D_{xx} + D_{yy}}{D_{xx} D_{yy} - D_{xy}^2} \right). \tag{39}$$

The derivation of the approximate variances for motion estimates from projections is essentially similar, leading to

$$\mathrm{var}(\hat{v}_x) \approx \frac{N\sigma^2}{d_{xx}}, \tag{40}$$

$$\mathrm{var}(\hat{v}_y) \approx \frac{N\sigma^2}{d_{yy}}, \tag{41}$$

where

$$d_{xx} = \sum_{x,t} \left(\frac{\partial p}{\partial x} \right)^2, \tag{42}$$

$$d_{yy} = \sum_{y,t} \left(\frac{\partial q}{\partial y} \right)^2. \tag{43}$$

We define the sum of these variances as an aggregate scalar measure of 2-D performance (for high-SNR cases):

$$C_2 = N\sigma^2 \left(\frac{1}{d_{xx}} + \frac{1}{d_{yy}} \right). \tag{44}$$

To bound the relative difference between C_2 and C_3, we begin by observing that

$$d_{xx} \leqslant N D_{xx}, \tag{45}$$

$$d_{yy} \leqslant N D_{yy}. \tag{46}$$

To see this, recall the definition of $p(x, t)$ in Eq. (7). From this it follows that[33]

$$\frac{\partial p}{\partial x} = \sum_y \frac{\partial f}{\partial x}. \tag{47}$$

Using this, we can write

$$d_{xx} = \sum_{x,t} \left(\frac{\partial p}{\partial x} \right)^2 = \sum_{x,t} \left(\sum_y \frac{\partial f}{\partial x} \right)^2 \tag{48}$$

$$\leqslant \sum_{x,t} \left[N \sum_y \left(\frac{\partial f}{\partial x} \right)^2 \right] \tag{49}$$

$$= N \sum_{x,y,t} \left(\frac{\partial f}{\partial x} \right)^2 = N D_{xx}, \tag{50}$$

where relation (49) is a consequence of the Cauchy–Schwarz inequality.[34] We similarly have $d_{yy} \leqslant N D_{yy}$. Rewriting C_3 and incorporating the above bounds, we obtain

$$C_3 = \sigma^2 \left(\frac{1}{D_{xx}} + \frac{1}{D_{yy}} \right) \left(\frac{1}{1 - D_{xy}^2 / D_{xx} D_{yy}} \right) \tag{51}$$

$$\leqslant N\sigma^2 \left(\frac{1}{d_{xx}} + \frac{1}{d_{yy}} \right) \left(\frac{1}{1 - D_{xy}^2 / D_{xx} D_{yy}} \right) \tag{52}$$

$$= C_2 \left(\frac{1}{1 - D_{xy}^2 / D_{xx} D_{yy}} \right), \tag{53}$$

which readily implies that

$$\frac{C_3 - C_2}{C_3} \leqslant \frac{D_{xy}^2}{D_{xx} D_{yy}}. \tag{54}$$

This indicates that, at least for small motions and large SNR's, the relative performance loss suffered in using a projection-based approach as opposed to a 3-D approach can be quite small if, for a given D_{xy}, there are sufficiently large spatial gradients in the directions of projection, i.e., if the right-hand side of relation (54) is sufficiently small. In particular, since the covariance matrix Q is positive definite, the right-hand side of relation (54) is always strictly less than unity. Similar statements can be made for projections made along any pair of nontrivial directions.

By way of reference, I mention here that CRLB's for translational motion estimates in highly textured images have been derived in Ref. 35, where the author has taken the measurement model for the motion signal to be the output of spatiotemporal Gabor and difference-of-Gaussian energy filters. The assumption of highly textured images in Ref. 35 is motivated by the fact that the estimation error is smaller when higher (spatial) frequency information is available in more of the image. The present estimates of the error covariance matrices for both the 3-D and 2-D cases support this assertion.

6. COMPUTATIONAL COMPLEXITY

In this section a computational budget is developed for the proposed algorithm and compared with the computational complexity of a comparable algorithm based on the use of 3-D FFT's. To make a fair comparison with the proposed algorithm, we consider the 3-D equivalent of

Table 1. Computational Budget

Operation	2-D	3-D
Projection	$2N^3$	0
FFT	$2N^2 \log N$	$N^3 \log N$
FFT Magnitude	$6N^2$	$3N^3$
SLIDE	$O(N^2)$	$O(N^3)$
Image Differencing	$O(N^3)$	$O(N^3)$
Matching	$O(N^2K)$	0
Total (l velocities)	$\max\{O(l^2N^2K), O(lN^3)\}$	$O(lN^3 \log N)$

this algorithm, where after the 3-D FFT is computed, velocity vectors are computed by plane fitting with the use of SLIDE. Note that this comparison will be a rather conservative one, since this aforementioned (hypothetical) 3-D approach is apparently less computationally costly than matched filtering in three dimensions with a bank of directional filters [which, incidentally, requires $O(N^5)$ operations].

Now suppose that a sequence of N frames of $N \times N$ images is given that contains exactly l distinct translational motions. Then (the worst-case, sequential versions of) the 2-D and 3-D algorithms will require the number of computations outlined in Table 1.

The 3-D approach will require $O(lN^3 \log_2 N)$ computations, since one 3-D FFT will be required for each velocity. For the 2-D approach, in the worst case, the process of estimating the velocity components is repeated l times; also, matching of the velocity components, as described in Section 4, requires $O(l^2N^2K)$ operations, where $K \leqslant N - 1$ is the number of frames used for matching, defined in Eq. (25).

Note that the number of operations that we just computed for the 2-D approach assumes that no reprojections are necessary. As described in Section 4, we assumed that no x components were identical in the set V_x (and similarly for V_y). If this assumption is violated, then it is necessary to choose new projection angles and repeat at least some of the steps of the algorithm to resolve the ambiguity. If the resolution of this ambiguity requires r reprojections along pairs of independent directions, then the number of computations for the 2-D approach will be at most $\max[O(rl^2N^2K), O(rlN^3)]$. Even in this case, the 2-D algorithm is more efficient for sufficiently large N.

7. EXPERIMENTAL RESULTS

In this section the results are presented of two experiments that examine the potential of the proposed algorithm. The first experiment will demonstrate the estimation of two superimposed (added) subpixel motions in a simulated image sequence. The second will estimate the dominant translational motion in a real aerial image sequence.

A. Simulated Example

To simulate the measured frames in this experiment, I used a 475×720 aerial (ortho-) photograph of Washington, D.C., shown in Fig. 3. From this image 40 subimages of dimension 256×256, each pair separated by 2 pixels, were collected, beginning with the lowermost square, shown in Fig. 3, and ending in the uppermost square. Next, the synthetic 512×512 cloud image shown in Fig. 4 was generated, and from this image 40 subimages of dimension 256×256, each pair separated by 1 pixel, were collected, this time beginning with the uppermost square and ending in the lowermost square.[36] Denote by $I_{\text{D.C.}}(t)$ and $I_{\text{Cloud}}(t)$ the t-th subimages of the D.C. and the cloud sequence, respectively. Then, the 256×256 images $I(t)$ were generated as

$$I(t) = 0.3I_{\text{D.C.}}(t) + I_{\text{Cloud}}(t), \tag{55}$$

so that the power in the cloud component of the above sum is roughly 3 dB higher than that of the ground, corresponding to a heavy (thick) cloud cover scenario.

The frames thus generated now contain two displacement components, namely $(2, -2)^{\text{T}}$ and $(-1, 1)^{\text{T}}$ pixels per frame. To simulate subpixel motion, I generated the frames $f(x, y, t)$ from $I(t)$ by low-pass filtering followed by downsampling. The low-pass filtering step is necessary to avoid aliasing and, in fact, can represent the

Fig. 3. Aerial photograph of Washington, D.C. (courtesy U.S. Geological Survey).

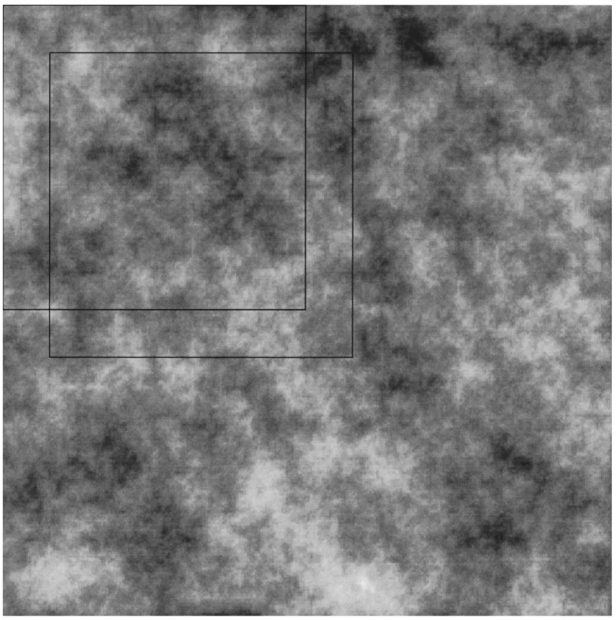

Fig. 4. Synthetic cloud image.

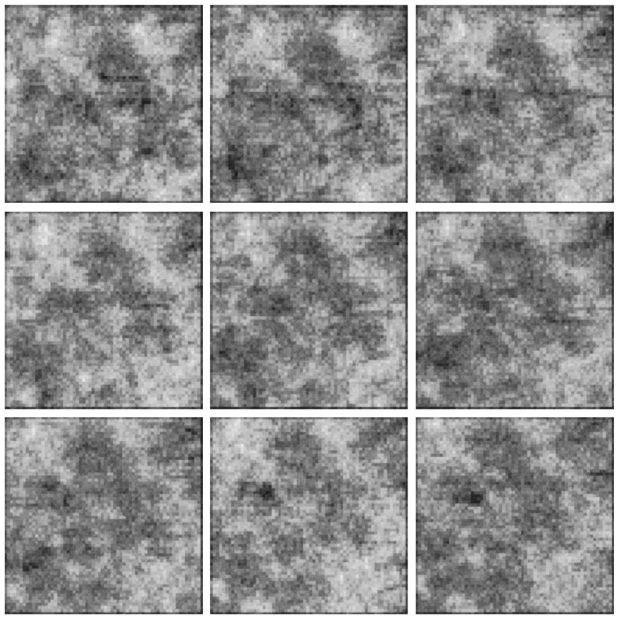

Fig. 5. From left to right and top to bottom: 1, 5, 10, 15, 20, 25, 30, 35, and 40 of the sequence.

point-spread function of the imaging system. In this case this point-spread function was (somewhat arbitrarily) chosen as an 8×8 Gaussian with unit variance. The images $I(t)$ were filtered and downsampled by a factor of 4 to yield a sequence of 40 64×64 images $f(x, y, t)$. This sequence thereby contains two superimposed subpixel motion components given by $v_1 = (1/2, -1/2)^{\mathrm{T}}$ and $v_2 = (-1/4, 1/4)^{\mathrm{T}}$, corresponding to the ground and the cloud motion, respectively. Gaussian white noise is then added to each frame (corresponding to SNR = 16 dB),[37] and they are processed through the algorithm. Figure 5 shows selected frames from the sequence thus generated.

Figure 6 shows the magnitude of the 2-D FFT's of the (de-meaned) rowwise projections on the right and that of the columnwise projections on the left. The distinct lines corresponding to the displacements are seen.[38]

SLIDE is then used to extract one velocity out of each image. In this case the sequential extraction of velocities is appropriate, as the ground image is much more highly textured than the superimposed clouds. The extracted velocities were $\hat{v}_x(1) = 0.5196$ and $\hat{v}_y(1) = -0.4959$. After storage of these estimated velocities the difference frames were formed and fed back into the projection process. The 2-D FFT magnitudes of the results are shown

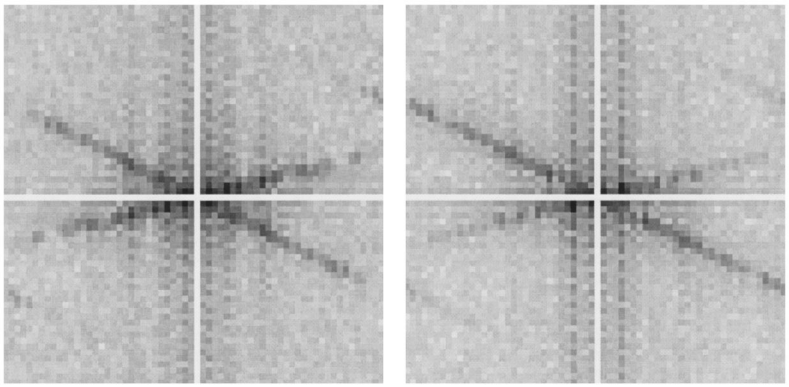

Fig. 6. (a) $|P(\omega_x, \omega_t)|$, (b) $|Q(\omega_y, \omega_t)|$.

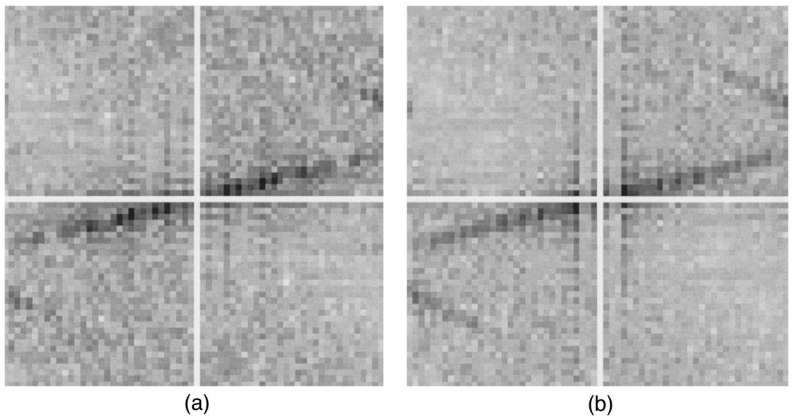

Fig. 7. (a) $|D_p(\omega_x, \omega_t)|$ and (b) $|D_q(\omega_y, \omega_t)|$ Fourier magnitudes of projections after removal of one velocity.

in Fig. 7, and these clearly display the remaining velocities to be estimated. Another application of SLIDE yields $\hat{v}_x(2) = -0.2603$ and $\hat{v}_y(2) = 0.2522$.

To match the velocities $V_x = \{\hat{v}_x(1), \hat{v}_x(2)\}$ and $V_y = \{\hat{v}_y(1), \hat{v}_y(2)\}$, we compute the difference measure L for all possible pairs over only the first $K = 5$ difference frames. This gives $L(1, 1) = 248.6188$, $L(1, 2) = 235.5621$, $L(2, 1) = 212.0420$, and $L(2, 2) = 183.4123$. Hence, since $L(2, 2)$ is the smallest, we have $\hat{v}_2 = [\hat{v}_x(2), \hat{v}_y(2)]^T = (-0.2603, 0.2522)^T$, which also yields $\hat{v}_1 = [\hat{v}_x(1), \hat{v}_y(1)]^T = (0.5196, -0.4959)^T$. Each of these velocity estimates is seen to be within roughly 4% or 1/100 ppf of the true values.

To confirm that the performance obtained in this example was typical and to gain more insight about the performance as a function of noise, I performed a Monte Carlo simulation in which the above experiment was repeated for 50 different realizations of the noise at each of various SNR's. The mean performance curves for the x and y velocity estimates are shown in Figs. 8 and 9. It is seen that the performance is consistently good for SNR's above 10 dB. For higher-intensity noise the performance degrades quickly and severely. To see how the 2-D performance would compare with the direct 3-D spectral approach, we compute the bound in relation (54) for this example. The upper bound on the relative difference between the approximate error variances for the two approaches turns out to be $\approx 5 \times 10^{-4}$, or 0.05%.

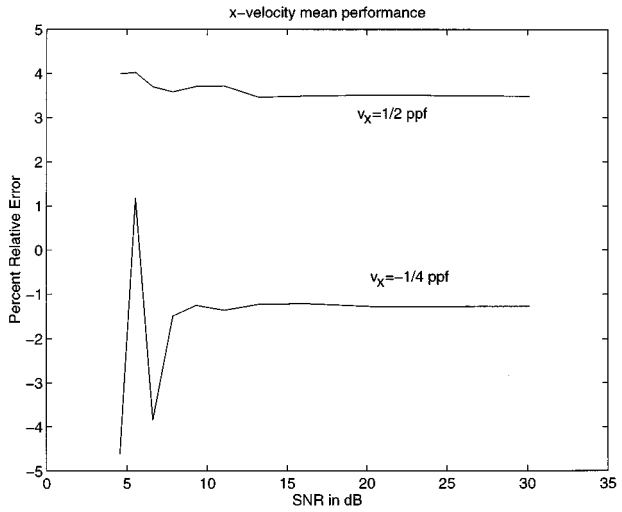

Fig. 8. Mean performance for x velocity estimates.

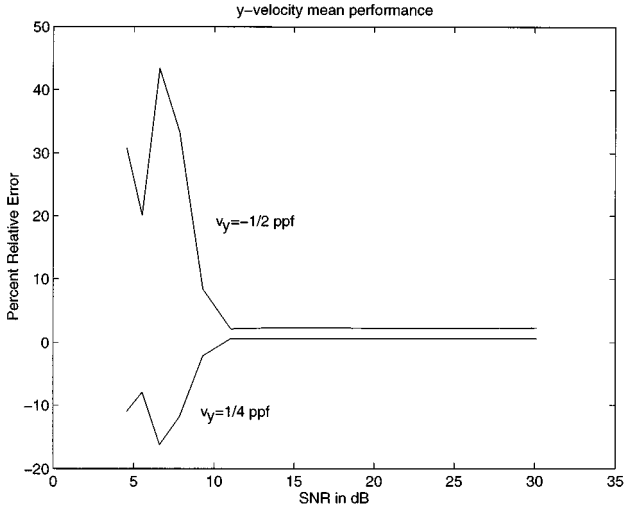

Fig. 9. Mean performance for y velocity estimates.

B. Real Example

The real image sequence processed in this example was obtained from the VASC image database at Carnegie Mellon University. The sequence consists of 50 frames (of a longer sequence) of aerial images of the Pittsburgh area taken from an aircraft. Low-resolution frames (60×64) were chosen for processing, and selected frames of this sequence are shown in Fig. 10. The projection functions $p(x, t)$ and $q(y, t)$ are shown in Fig. 11. By looking at these images, we can immediately note that the

Fig. 10. From left to right and top to bottom: frames 1, 5, 9, . . ., 41, and 45 of the Pittsburgh sequence.

motion occurring in this image sequence is not purely translational (this would correspond to linear strips in both projection images); this is particularly clear in $q(y, t)$. However, we can see that a strong translational component is present in both projection images (at least for times $t > 25$). This is further supported by the shape of the magnitude spectra of these projection functions shown in Fig. 12, which exhibit linear features. Under the assumption that only one translational motion is present in each direction the algorithm produces a motion vector estimate $\hat{v} = (-0.3334, 0.3540)^{\mathrm{T}}$.

Since no ground truth is available for this example, we rely on a heuristic technique to show that a strong component of the motion in this image sequence is captured by the estimated translational motion vector. To this end, we follow[7] and use the internal representation image sequence $g(x, y, t)$. This sequence is defined by

$$g(x, y, 0) = f(x, y, t), \tag{56}$$

$$g(x, y, t + 1) = (1 - \gamma) f(x, y, t + 1) + \gamma \,\text{Register}\,[g(x, y, t), f(x, y, t + 1)], \tag{57}$$

where Register $(\cdot, \cdot)$ denotes the difference image obtained by registering the arguments with the use of the estimated motion vector $\hat{v}$, and $0 < \gamma < 1$ is a smoothing parameter. The effect of this iteratively defined tempo-

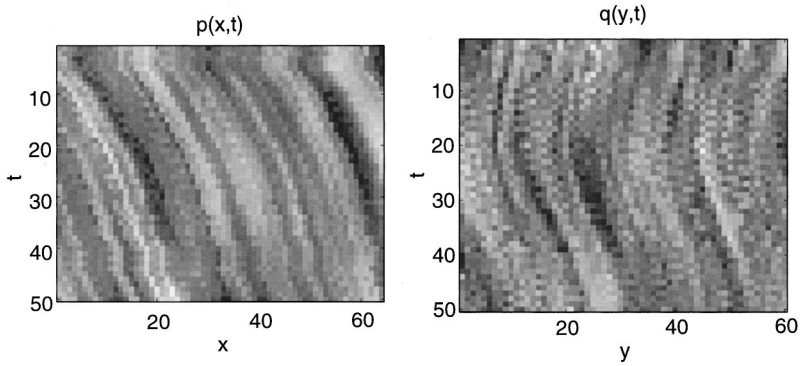

Fig. 11. Projection images of the Pittsburgh sequence.

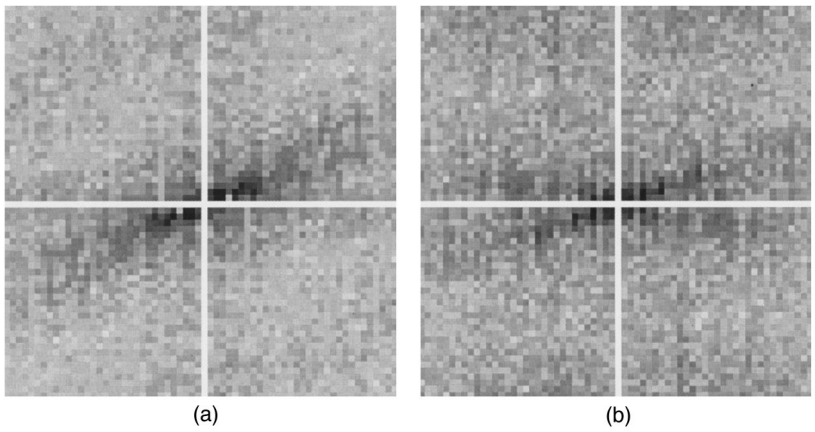

Fig. 12. Magnitude spectra of (a) p and (b) q for the Pittsburgh sequence.

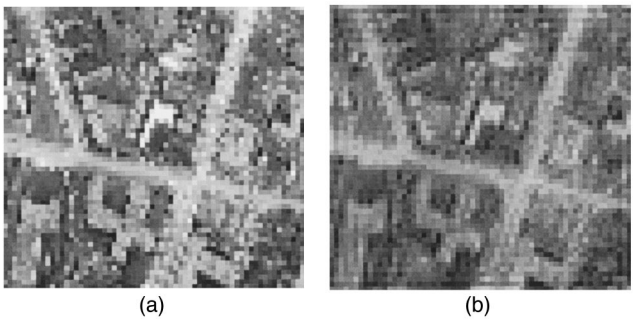

Fig. 13. (a) 50th frame of the original sequence, (b) corresponding frame of the internal representation.

ral smoothing operation is that $g(x, y, t)$ maintains sharpness on parts of the image where the motion vector is an accurate representation of the true motion, while other parts of the image are blurred. To see the effect of this operation on the current example, I display the 50th frames of both $f(x, y, t)$ and $g(x, y, t)$ in Fig. 13, where $\gamma = 0.75$ was used. As we can see, most larger-scale features of the image have remained in focus, while the smaller-scale features have been blurred out. This indicates, at least qualitatively, that the dominant motion component has been captured by the estimated motion vector.

8. DISCUSSION AND CONCLUSIONS

In this paper an efficient algorithm has been described for the estimation of superimposed translational motions from multiple image frames. The algorithm has been shown to be effective for moderate noise levels. As a result of using projections, followed by 2-D FFT's, and an efficient array processing technique for line detection, the algorithm incurs lower computational cost when compared with alternative techniques based on the 3-D FFT. The effectiveness of the proposed algorithm has been further demonstrated on both a simulated and a real sequence of aerial images. It is noteworthy that after the submission of this paper I became aware of Ref. 19, in which the idea of projections followed by (subspace-based) line detection was independently suggested.

While the present algorithm offers potentially significant computational savings, it is not expected to perform as well as comparable 3-D FFT-based algorithms, since the 2-D processing deals with only two slices of the complete 3-D spectrum of the image sequence. In Section 5 we used approximate error variances to demonstrate that when the motions are small and the SNR is high, the relative degradation in performance suffered when using the proposed approach rather than a direct 3-D approach can be quite small if the spatial gradients of the images are sufficiently large in the directions where projections are taken.[39]

A potential difficulty that can arise in the 2-D approach is that if two motions in the image frames have identical velocity components along a direction of projection, then the 2-D FFT of the projection along that direction will show energy concentration along only one line. In general, the algorithm is not designed to distinguish the number of multiple motions present, although this can be done.[2] However, as was pointed out in Section 6, the ambiguity can be remedied by reprojecting the frames along new directions. The new directions should be chosen to be maximally apart from the original directions. This can be accomplished by selecting the bisector of the angle between the original pair of directions. In any case the process of reprojection will increase the computational load of the algorithm. However, if the number of reprojections needed to resolve such ambiguities can be kept low—for instance, by choosing new directions as described above—the proposed approach is still useful, particularly when relatively few velocities are being sought.

The experimental results of Section 7 demonstrate that the algorithm proposed herein is capable of efficiently estimating multiple velocities to within a few percent of truth given a moderate number of frames. It should, however, be pointed out that in the simulated experiments a consistent, rather small bias was observed in the estimates produced by the algorithm. I expect that two sources are responsible for this observed bias. The first is likely the sequential estimation of the motions. The energy in the second motion component biases the SLIDE estimate when it is assumed that only one line is present in the magnitude spectrum. This would explain the observation that the estimates of the cloud velocities have smaller biases, since these are typically extracted in the second pass of the algorithm, where only one motion is left over. The second major source of bias, I suspect, is the finite resolution of the data.

Finally, it should be noted that existing iterative image-domain techniques such as those of Refs. 3 and 21 work better for rather large motions and that their convergence is a strong function of the image content. Furthermore, their computational cost is rather high. The present algorithm, while also sensitive to the image content, is specifically intended as a faster technique for situations in which a large number of frames are available, and it allows for the possibility of more than two superimposed motions.

ACKNOWLEDGMENTS

The author thanks the anonymous reviewer, whose comments and suggestions have helped improve the content and the presentation of this paper.

REFERENCES AND NOTES

1. B. K. P. Horn, *Robot Vision* (MIT Press, Cambridge, Mass., 1986).
2. M. Shizawa and K. Mase, "Principles of superposition: a common computational framework for analysis of multiple motion," in *Proceedings of the IEEE Workshop on Visual Motion* (IEEE, New York, 1991), pp. 289–295.
3. J. R. Bergen, P. Burt, R. Hingorani, and S. Peleg, "A three-frame algorithm for estimating two-component image motion," IEEE Trans. Pattern Anal. Mach. Intell. **14**, 886–896 (1992).
4. D. Kersten, "Transparency and the cooperative computation of scene attributes," in *Computational Models of Visual Processing*, M. Landy and J. A. Movshon, eds. (MIT Press, Cambridge, Mass., 1991), pp. 209–228.
5. J. K. Aggarwal and N. Nandhakumar, "On the computation of motion from sequences of images—a review," Proc. IEEE **76**, 917–935 (1988).
6. P. J. Burt, R. Hingorani, and R. J. Kolczynski, "Mechanisms for isolating component patterns in the sequential analysis of multiple motion," in *Proceedings of the IEEE Workshop on Visual Motion* (IEEE, New York, 1991), pp. 187–193.
7. M. Irani, B. Rousso, and S. Peleg, "Computing occluding and transparent motions," Int. J. Comput. Vis. **12**, 5–16 (1994).
8. D. J. Heeger, "Model for the extraction of image flow," J. Opt. Soc. Am. A **4**, 1455–1471 (1987).
9. B. Porat and B. Friedlander, "A frequency domain algorithm for multiframe detection and estimation of dim targets," IEEE Trans. Pattern Anal. Mach. Intell. **12**, 398–401 (1990).
10. L. Bruton and N. Bartley, "Three-dimensional image processing using the concept of network resonance," IEEE Trans. Circuits Syst. **CAS-32**, 664–672 (1985).

11. K. S. Knudsen and L. T. Bruton, "Moving object detection and trajectory estimation in the transform/spatiotemporal mixed domain," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing* (IEEE, New York, 1992), Vol. 3-M.
12. S. A. Mahmoud, M. S. Afifi, and R. J. Green, "Recognition and velocity computation of large moving objects in images," IEEE Trans. Acoust. Speech Signal Process. **36**, 1790–1791 (1988).
13. S. A. Rajala, A. M. Riddle, and W. E. Snyder, "Application of the one-dimensional Fourier transform for tracking moving objects in noisy environments," Comput. Vision Graph. Image Process. **21**, 280–293 (1983).
14. M. H. Groves, S. A. Rajala, and W. E. Snyder, "Calculation of displacement fields by means of the motion detection transform," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing* (IEEE, New York, 1984), pp. 23.6.1, 23.6.4.
15. A. Kojima, N. Sakurai, and J. Ishigami, "Motion detection using 3-d FFT spectrum," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing* (IEEE, New York, 1993), Vol. V.
16. B. Girod and D. Kuo, "Direct estimation of displacement histograms," in *Proceedings of the Image Understanding and Machine Vision Conference* (IEEE, New York, 1989), pp. TuB3-1–TuB3-4.
17. H. K. Aghajan and T. Kailath, "SLIDE: subspace-based line detection," IEEE Trans. Pattern Anal. Mach. Intell. **16**, 1057–1073 (1994).
18. H. K. Aghajan and T. Kailath, "Sensor array processing techniques for super resolution multi-line-fitting and straight edge detection," IEEE Trans. Image Process. **2**, 454–465 (1993).
19. H. K. Aghajan and T. Kailath, "Subspace techniques for image understanding," in *Proceedings of the 28th Asilomar Conference on Signals, Systems, and Computers* (IEEE, New York, 1994), pp. 726–730.
20. In this paper the terms velocity vector and displacement vector are used interchangeably. Both are assumed to be in units of pixels per frame (ppf).
21. B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the IEEE Image Understanding Workshop* (IEEE, New York, 1981), pp. 121–130.
22. Note also that the frequency-domain approach solves the LS problem without resorting to the Taylor-series approximation.
23. G. T. Herman, *Image Reconstruction from Projections* (Academic, New York, 1980).
24. This linear system of equations is numerically well conditioned when the angles θ_1 and θ_2 are approximately 90° apart. Hence preference is given to orthogonal projection directions.
25. R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," Commun. ACM **15**, 11–15 (1972).
26. A. K. Jain, *Fundamentals of Digital Image Processing* (Prentice-Hall, Englewood Cliffs, N.J., 1989).
27. P. V. C. Hough, "Method and means for recognizing complex patterns," U.S. patent 3,069,654 (December 18, 1962).
28. C. M. Brown, "Inherent bias and noise in the Hough transform," IEEE Trans. Pattern Analysis Mach. Intell. **PAMI-5**, 493–505 (1983).
29. R. Roy, A. Paulraj, and T. Kailath, "ESPRIT: a subspace rotation approach to estimation of parameters of cissoids in noise," IEEE Trans. Acoust. Speech Signal Process. **34**, 1340–1342 (1986).
30. If this condition is violated, it may then be necessary to pick different projection angles and repeat the previous steps. This is a potential drawback of the proposed algorithm, which will be commented on in Section 6.
31. The spatial correlation of w_1 and w_2 may be larger when the projection directions are not orthogonal because, in general, the corresponding sums may have more than one term in common. This points out an advantage of using mutually orthogonal projection directions.
32. S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory* (Prentice-Hall, Englewood Cliffs, N.J., 1993).
33. Strictly speaking, it is more appropriate to state this result for the continuous case ($N = \infty$), but for consistency of notation summations are used instead of integrals.
34. J. E. Marsden, *Elementary Classical Analysis* (Freeman, San Francisco, 1974).
35. R. Jasinschi, "Intrinsic constraints in space–time filtering: a new approach to representing uncertainty in low-level vision," IEEE Trans. Pattern Anal. Mach. Intell. **14**, 353–366 (1992).
36. These images have a power spectrum that drops off (roughly) as $1/f^2$.
37. The definition of the SNR is
$$\mathrm{SNR(dB)} = \frac{1}{T}\sum_{t=0}^{T-1} 10\log_{10}\left\{\frac{\sum_{x,y}[f(x,y,t)-\bar{f}(t)]^2}{N^2\sigma^2}\right\},$$
where $\bar{f}$ is the spatial average of $f(x, y, t)$, N is the spatial dimension of f, and T is the number of frames. So the SNR is the average SNR per pixel across all frames.
38. In these displays, to facilitate viewing, the gray-scale color map is not linear.
39. This suggests that one may be able to pick the best projection directions *a priori* by computing the gradient on a sample image in several different directions.