International Journal of Computer Vision 37(3), 203–230, 02000

Extracting Structure from Optical Flow Using the Fast Error Search Technique

SRIDHAR SRINIVASAN
Microsoft Corporation, One Microsoft Way, Redmond, WA 98052
sridhsri@microsoft.com

Abstract. In this paper, we present a globally optimal and computationally efficient technique for estimating the focus of expansion (FOE) of an optical flow field, using *fast partial search*. For each candidate location on a discrete sampling of the image area, we generate a linear system of equations for determining the remaining unknowns, viz. rotation and inverse depth. We compute the least squares error of the system without actually solving the equations, to generate an error surface that describes the goodness of fit across the hypotheses. Using Fourier techniques, we prove that given an $N \times N$ flow field, the FOE, and subsequently rotation and structure, can be estimated in $\mathcal{O}(N^2 \log N)$ operations. Since the resulting system is linear, bounded perturbations in the data lead to bounded errors.

We support the theoretical development and proof of our technique with experiments on synthetic and real data. Through a series of experiments on synthetic data, we prove the correctness, robustness and operating envelope of our algorithm. We demonstrate the utility of our technique by applying it for detecting obstacles from a monocular sequence of images.

Keywords: structure from motion, fast partial search, focus of expansion, optical flow

1. Introduction

The extraction of the three-dimensional structure of a moving scene from a sequence of images is termed the *structure from motion* (*SFM*) problem. The solution to this problem is a key step in the computer vision tasks of robotic navigation, obstacle avoidance, time to collision, recognition of solid objects, and surveillance. Recent interest in this area has been sparked by the desire to build 3D models from video for virtual reality, video conferencing, manufacturing and medical applications. The proliferation of powerful computing machinery and video streams has transitioned the problem from a purely theoretical domain to one of practical interest. Mathematical analysis of SFM proves the non-linear interdependence of structure and motion given observations on the image plane. While this problem has received considerable attention by researchers, the proposed solutions tend to have several shortcomings. Yet, it is well recognized that the human visual system performs the task of simultaneously estimating motion and the qualitative depth structure of a moving scene quite efficiently and accurately.

In a strict sense, SFM is not a single, well-defined problem. Rather it is a class of problems encompassing a wide range of possible imaging scenarios, camera motions, and object models. Various types of camera motion give rise to various simplifications, or additional constraints, which can be efficiently exploited by a well-matched algorithm. Certain members of the SFM class of problems have been well-studied, and excellent solutions have emerged over the past several years. Calibrated stereo, where a depth map is computed from two views of 3D scene with the motion being a lateral camera shift, and global perspective motion generated by a purely rotating camera, or an

This work was performed while the author was affiliated with the center for Automation Research at the University of Maryland, College Park.

arbitrarily moving camera viewing a 3D plane, are two well-studied subclasses of SFM. It is well established that lateral camera shift gets ambiguated with camera pitch and yaw, particularly when the camera has a small field of view (Young and Chellappa, 1992; Mitiche, 1994). A single solution to the entire class of SFM problems has eluded researchers, due to the disparity of its member sub-problems. In lieu of a general-purpose SFM solution, it has been proposed that "one should design algorithms specifically for their problem domains" (Oliensis, 1997).

Psychophysical experiments on human subjects reveal that the first stage of processing in the human visual system is the estimation of the motion field. Methods for estimating the structure from the motion field are referred to as *differential* techniques, as opposed to *discrete* methods which rely on two or more distinct views of the scene. The *optical flow*, which is defined as the 2-D projection of the 3D motion field, is composed of the horizontal and vertical velocity fields, $u(x, y)$ and $v(x, y)$. These are related to the motion and scene depth according to

$$\begin{aligned} u(x,y) &= (-t_x + xt_z)g(x,y) + xy\omega_x \\ &\quad - (1+x^2)\omega_y + y\omega_z \\ v(x,y) &= (-t_y + yt_z)g(x,y) + (1+y^2)\omega_x \\ &\quad - xy\omega_y + x\omega_z \end{aligned} \tag{1}$$

where the translational and rotational motion vectors are (t_x, t_y, t_z) and $(\omega_x, \omega_y, \omega_z)$ respectively. $g(x, y) = 1/Z(x, y)$ is the inverse scene depth, and all linear dimensions are normalized in terms of the focal length f of the camera. Such a pair of equations exists for every point (x, y) on the image plane at which optical flow can be determined. Assuming there exist M such points, there are $2M$ equations and $M + 5$ independently determinable unknowns. Of the latter, M unknowns are related to scene depth and the remaining five are the determinable motion parameters. One obvious means of simplifying the system is to eliminate the depth map $g(x, y)$ from the above equations. Such an operation gives rise to a nonlinear system of equations in five determinable unknowns. In practice, however, the elimination of depth is very sensitive to noise in the data u, v, which is compounded by the nonlinear nature of the resulting equations.

With this backdrop, it is now possible to formulate a better definition for the problem we address in this paper. Here, we consider a 2 frame solution for SFM. The motion is arbitrary, the only restriction being that the forward component of translation dominates the lateral components. We assume that the correspondences are determined externally (e.g. by using matching or optical flow). We place no restrictions on the permissible structure of the scene, or on the field of view of the camera. We do require, however, that the focal length be known reasonably well.

This variant reflects one of the most interesting SFM scenarios, and retains the mathematical challenges of general SFM. The prior proposed solutions to this problem, which are discussed in the next section, tend to have several shortcomings. In contrast, the algorithm proposed here is an accurate and efficient technique for estimating the Focus of Expansion (FOE). This attains the globally optimal FOE estimate (the nature of optimality is discussed further ahead) in a deterministic number of compute operations independent of the nature of input data. The algorithm is applicable across a wide range of depth map structures and fields of view. Its performance in the presence of noise is exemplary.

In the proposed technique, the FOE is hypothesized to lie within a bounded square on the image plane. For each candidate location on a discrete sampling of the plane, we generate a linear system of equations for estimating the remaining unknowns which are the rotational velocity and inverse depth map. We compute the least squares error of the system *without actually solving the equations*, to generate an error surface that describes the goodness of fit as a function of the hypothesized focus of expansion. Our technique exploits the symmetry of (1) and uses Fourier techniques to vastly reduce the computational burden. The minimum of the error surface occurs at a discrete location very close to the true FOE. For an optical flow field of size $N \times N$, the order of complexity in computing the error surface, and consequently of estimating the FOE, is $\mathcal{O}(N^2 \log N)$, and is independent of the specific input data. Since the resulting system is linear, bounded perturbances in the optical flow estimates lead to a deterministic, bounded offset of the error surface minimum from zero. Thus, noise resilience is inherent to this linear formulation.

2. Literature Review

The pioneering psychophysical experiments by Wallach and O'Connell (Wallach and O'Connell, 1953) and Gibson (Gibson, 1955, 1957) hypothesize the

simultaneous recoverability of structure and depth from an optical flow field. This is quantified and substantiated by Ullman (Ullman, 1979), Nakayama and Loomis (Nakayama and Loomis, 1974), and Koenderink and van Doorn (Koenderink and van Doorn, 1976). The early approaches by Longuet-Higgins and Prazdny (Longuet-Higgins, 1981; Longuet-Higgins and Prazdny, 1980), and later by Waxman and Ullman (Waxman and Ullman, 1985) derive closed-form solutions to the SFM problem by examining derivatives of the optical flow field, with an underlying assumption of smoothness of the 3D surface. Bruss and Horn (Bruss and Horn, 1983) and Adiv (Adiv, 1985) propose nonlinear solutions which minimize the squared error between the observed and predicted flows. While both the approaches are iterative, the former is global while the latter subdivides the flow field into smooth patches with a mechanism that allows them to be combined. Linear approaches to SFM can be traced back to the two-view solution formulated by Tsai and Huang (Tsai and Huang, 1981) based on point correspondences. Linear methods are applied to motion fields by Zhuang et al. (Zhuang et al., 1984, 1988), Waxman et al. (Waxman et al., 1987), and Mitiche et al. (Mitiche et al., 1987). Prazdny (Prazdny, 1981) subdivides the SFM problem by first hypothesizing the rotational component of motion, and then solving for the remaining variables by means of a nonlinear system of equations. In a related manner, Lawton (Lawton, 1983) searches for the translation direction at discrete points on the surface of a unit sphere he terms the *translation direction sphere*. A good summary of past literature and algorithms is provided in the books by Mitiche (Mitiche, 1994) and Weng et al. (Weng et al., 1991).

Among the more recent SFM literature are the solutions due to Heeger and Jepson (Heeger and Jepson, 1992) and Gupta and Kanal (Gupta and Kanal, 1995), both of which eliminate depth and rotation variables from (1) to solve for translation using a linear constraint. Direct techniques, first proposed by Aloimonos and Brown (Aloimonos and Brown, 1984), and later expanded by Negahdaripour and Horn (Negahdaripour and Horn, 1987) and Horn and Welden (Horn and Weldon, 1988), altogether bypass computation of the motion field by operating on raw luminance data. Several additional cues have been used to formulate a solution to the SFM problem, prominent among which are stereo images, stereo flow fields, and long sequences. These are not pertinent to the current work and are not discussed further. One important constraint that has gone largely unnoticed (expect as a means of resolving ambiguities in the form of multiple solutions) is the non-negativity of depth. This has been used in the recent approaches in a direct manner by Fermüller and Aloimonos (Fermuller and Aloimonos, 1995, 1997), and more indirectly by Fejes and Davis (Fejes and Davis, 1998).

Algorithms that eliminate the depth field $g(x, y)$ from (1) (Tsai and Huang, 1981; Zhuang et al., 1984, 1988; Waxman et al., 1987; Mitiche et al., 1987; Gupta and Kanal, 1995) are not very stable, although Gupta and Kanal (Gupta and Kanal, 1995) have claimed experimental results demonstrating some amount of noise resilience. Moreover, these algorithms discard valuable information in the form of nonlinear equality constraints which are not simple to enforce. Likewise, assuming smoothness of the depth field or optical flow field is not always valid, more so when there is noise in the flow estimates. Thus, differentiating noisy flow fields (Longuet-Higgins and Prazdny, 1980; Waxman and Ullman, 1985) in order to solve SFM is highly undesirable. On the other hand, the method due to Fejes and Davis (Fejes and Davis, 1998) requires strong variations in the underlying depth map (and thereby in the optical flow field) to work at all. Nonlinear optimization-based solutions (Bruss and Horn, 1983; Adiv, 1985; Negahdaripour and Horn, 1987) are relatively stable in the presence of noise. However, minimizing a nonlinear cost function exposes the solution to the pitfalls of local minima and slow convergence. The drawback of search-based methods (Prazdny, 1981; Lawton, 1983) is their slow speed of execution.

A significant deviation of our work from conventional techniques is that the error surface we generate gives us a distributed representation of the confidence, quantified in terms of the mean squared error over the solution space. In a similar manner, Simoncelli argues for a distributed representation of the optical flow field of a sequence as an alternative to a unique solution (Simoncelli, 1993). He shows that the computed error of a candidate solution is related to its likelihood of being the true solution. In a Bayesian framework, a distributed representation allows for error covariance propagation. Likewise, we claim that the error surface embeds information regarding both the presence of the true FOE and a confidence measure. A relatively flat error surface indicates low confidence in the solution whereas a sharp dip indicates high confidence.

In order to measure the merit of our approach, we use four criteria, viz. correctness, robustness, operating

range and utility. At a minimum, any valid solution to a problem must be correct, given a perfect input data set. This is our first criterion, which we prove first in theory and then demonstrate on artificially generated perfect flow fields. Next, we show the robustness of our method by experiments on noisy and sparse flows. In order to traverse the operating range of the algorithm, we evaluate its performance over a range of fields of view and depth map structures. Finally, we consider a real-world application calling for 3D SFM to demonstrate the utility of our approach. This application is the detection of obstacles from a sequence of images gathered by a monocular camera mounted on a moving vehicle.

This paper is organized as follows: Section 3 introduces the concept of partial search and the assumptions of our approach. The theory and derivations of our technique are explained in Section 4, with some details provided in the Appendix. Section 5 extends our technique by relaxing certain assumptions made earlier. The experiments on synthetic and real data are detailed in Section 6. Finally, we conclude with a few remarks in Section 7.

3. Problem Formulation

In the remainder of this paper, we will assume a camera-centered coordinate system whose origin coincides with the center of the imaging lens and whose *XY*-plane is parallel to the image plane. We will also assume that the focal length of the lens is known accurately, and that the linear distance unit is normalized by the focal length, i.e. $f = 1$. The Z axis lies along the optical axis and the image plane lies on $Z = -1$, which is the focal plane of the camera. Equation (1) can be rewritten as

$$\begin{aligned} u(x, y) &= (x - x_f)h(x, y) + xy\omega_x \\ &\quad - (1 + x^2)\omega_y + y\omega_z \\ v(x, y) &= (y - y_f)h(x, y) + (1 + y^2)\omega_x \\ &\quad - xy\omega_y - x\omega_z \end{aligned} \tag{2}$$

where $(x_f, y_f) \stackrel{\text{def}}{=} (\frac{t_x}{t_z}, \frac{t_y}{t_z})$ is known as the *focus of expansion* (*FOE*) and $h(x, y)$ is the scaled inverse depth map of the scene being imaged, $h(x, y) = \frac{t_z}{Z(x,y)}$.

3D motion algorithms aim at computing the 3D motion parameters of the camera, viz. $t = (t_x, t_y, t_z)$ and $\omega = (\omega_x, \omega_y, \omega_z)$. It can be seen that t can only be recovered up to a scale factor, i.e. if $(t = t_0, \omega = \omega_0)$ is a solution, so is (mt_0, ω_0). The 3D structure of the scene $h(x, y)$ can be recovered, once the 3D motion parameters are known. Many techniques build on the first step in which $h(x, y)$ is eliminated from (2), but this makes the solution very sensitive to noise in flow estimates. In order to achieve noise resilience and yet keep the computational demands of the solution within reasonable bounds, we wish to devise a linear solution to the SFM problem. The technique we propose is based on fitting a linear model to each candidate hypothesis over a bounded search space.

3.1. Assumptions

We will begin the analysis by enumerating the assumptions which simplify the understanding of our approach. As we progress, we will loosen some of the assumptions until only the essential ones remain. Further, we will argue that the assumptions are realistic and commonly hold in practice. In the following paragraphs describing the assumptions, the first sentence is the initial assumption made, for ease of understanding. The second sentence, which is *italicized*, is the minimum necessary to prove our method. The remainder of the item is a discussion of the assumption.

- The input flow field is defined over a square region, of size $N \times N$, where $N = 2^k$. *Can be fully relaxed.* With a square field whose side is a power of 2, we can use the fast Fourier transform (FFT) in some of our later computations. Other sizes do not change the formulation, but might lead to certain bounded inefficiencies in the computational requirement.
- The input flow field is dense. *Can be fully relaxed.* We first prove our result by assuming that a dense flow field is available for the $N \times N$ image. Later, we discuss the handling of sparse flow fields. However, the complexity of the proposed method is related to the size of the image for which the flow is available and not to the number of equation sets (i.e. the number of points at which the optical flow is known). Thus, the number of computations required to solve the problem given a sparse flow field is equal to the corresponding count given a dense flow field.
- The FOE lies within the area defined by the input flow field. *The FOE lies within a bounded region.* The primary aim of this effort is directed toward situations

where there is a large forward translational motion, like vehicle navigation. In this case, the FOE almost always lies within the image. In extreme situations where the FOE is very large or at infinity, t_z is very small in comparison with one or both of t_x and t_y. When t_z is small, Eq. (1) can be approximated as

$$\begin{aligned} u(x, y) &\approx -t_x g(x, y) + xy\omega_x - (1 + x^2)\omega_y + y\omega_z \\ v(x, y) &\approx -t_y g(x, y) + (1 + y^2)\omega_x - xy\omega_y + x\omega_z \end{aligned} \tag{3}$$

For this system of equations, there are only $M + 4$ independently determinable variables since there is still a scale ambiguity in the system. However, that is not all. Vision systems in practice tend to have small fields of view, or equivalently, large focal lengths. In this case, $x, y < 1$ and the contribution of the second terms in (3) is small, leading to the first-order approximation

$$\begin{aligned} u(x, y) &\approx -t_x g(x, y) - \omega_y + y\omega_z \\ v(x, y) &\approx -t_y g(x, y) + \omega_x - x\omega_z \end{aligned} \tag{4}$$

The instability of system (3) is clear when one notes the occurrence of multiple solutions in its approximation (4). Assume that $\{t_x, t_y, \omega_x, \omega_y, \omega_z, g_0, g_1, \ldots, g_{M-1}\}$ is a valid solution for the M instances of (4). Then the sets $\{t_x, t_y, \omega_x + t_y\delta, \omega_y - t_x\delta, \omega_z, g_0 + \delta, g_1 + \delta, \ldots, g_{M-1} + \delta\}$ are also valid solutions for all values of $\delta > 0$. The positivity constraint on δ ensures that depths are non-negative. An analysis of the instability of SFM under various conditions is provided in Adiv (Adiv, 1989) and Young and Chellappa (Young and Chellappa, 1992).

Secondly, when performing 3D SFM over a long sequence of frames, the location of the FOE at the previous time instant is known after the first frame for which the motion field is recovered. If it can be assumed that the FOE does not change drastically between frames, the search window for the FOE at the current frame can be shifted and re-centered around the estimated FOE at the previous frame. Under the relaxed assumption that the FOE must lie within a bounded area, the re-centering strategy will work.

Finally, it must be borne in mind that the output of our algorithm is not a single solution; rather, it is an error metric for the entire search space. When the FOE lies outside the search space, we expect that the shape of the error surface will show a dip in the direction of the true FOE. In our experiments, we have observed that when the true FOE lies outside the search area, the minimum is attained at its periphery, indicating the likelihood of the true minimum lying beyond. This flags an error condition and provides a candidate offset by which to shift the search area.

In summary, the only non-trivial situation where we expect the proposed method to fail is when the following three conditions simultaneously hold: (i) wide field of view, (ii) very small forward translation, and (iii) no available estimate of the FOE from the prior frame that can form a reasonable guess to recenter the search area. Even in this situation, if the minimum error is attained at the search boundary (which is very likely to be the case), we can iteratively re-center and locate the error surface minimum. In the next section, we develop the theory behind our approach.

4. Approach

Let the true FOE be (x_f, y_f). Assuming that the flow field is of size $N \times N$ and all N^2 flow estimates are available, the optical flow at pixel location $i, j \in \{0, 1, \ldots, N-1\}^2$ is given by

$$\begin{aligned} u_{i,j} &= (x_{i,j} - x_f)h_{i,j} + x_{i,j}y_{i,j}\omega_x \\ &\quad - \left(1 + x_{i,j}^2\right)\omega_y + y_{i,j}\omega_z \\ v_{i,j} &= (y_{i,j} - y_f)h_{i,j} + \left(1 + y_{i,j}^2\right)\omega_x \\ &\quad - x_{i,j}y_{i,j}\omega_y - x_{i,j}\omega_z \end{aligned} \tag{5}$$

where $x_{i,j} = \delta[i - (N-1)/2]$, $y_{i,j} = \delta[j - (N-1)/2]$, and $\{h, u, v\}_{i,j} = \{h, u, v\}(x_{i,j}, y_{i,j})$. δ is the inter-pixel distance in focal length units. The transformation between the pixel coordinate system point (i, j) and the normalized 3D coordinate system point (x, y) on the image plane is reversible and is given by

$$\begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} \frac{x}{\delta} + \frac{N-1}{2} \\ \frac{y}{\delta} + \frac{N-1}{2} \end{pmatrix} \tag{6}$$

Thus, the optical center lies at $(\frac{N-1}{2}, \frac{N-1}{2})$ in the pixel coordinate system. Without loss of generality, we will switch between coordinate systems to simplify notation

wherever necessary. Define

$$r_{i,j} = \begin{pmatrix} x_{i,j}y_{i,j} & -\left(1+x_{i,j}^2\right) & y_{i,j} \end{pmatrix}'$$

$$s_{i,j} = \begin{pmatrix} 1+y_{i,j}^2 & -x_{i,j}y_{i,j} & -x_{i,j} \end{pmatrix}'$$

$$\mathbf{Q} = \begin{bmatrix} r'_{0,0} \\ s'_{0,0} \\ r'_{0,1} \\ s'_{0,1} \\ \vdots \\ r'_{N-1,N-1} \\ s'_{N-1,N-1} \end{bmatrix}$$

$$h = (h_{0,0} \quad h_{0,1} \ldots h_{N-1,N-1})'$$

$$\mathbf{u} = (u_{0,0} \quad v_{0,0} \quad u_{0,1} \quad v_{0,1} \quad \ldots \quad u_{N-1,N-1} \quad v_{N-1,N-1})' \tag{7}$$

$$\mathbf{P}(x_f, y_f) = \begin{bmatrix} x_{0,0}-x_f & 0 & 0 & & \\ y_{0,0}-y_f & 0 & 0 & & \\ 0 & x_{0,1}-x_f & 0 & & \mathbf{0} \\ 0 & y_{0,1}-y_f & 0 & & \\ 0 & 0 & x_{0,2}-x_f & & \\ 0 & 0 & y_{0,2}-y_f & & \\ & & & \ddots & \vdots \\ & \mathbf{0} & & \ldots & x_{N-1,N-1}-x_f \\ & & & & y_{N-1,N-1}-y_f \end{bmatrix}$$

$$\mathbf{A}(x_f, y_f) = [\mathbf{P}(x_f, y_f) \quad \mathbf{Q}]$$

$$\mathbf{x}_0 = \begin{bmatrix} h \\ \omega \end{bmatrix}.$$

We will drop the argument (x_f, y_f) of $\mathbf{P}(x_f, y_f)$ and $\mathbf{A}(x_f, y_f)$ where it is obvious. The above definitions allow us to consolidate the motion equations for all individual flow vectors in the brief form

$$[\mathbf{P}(x_f, y_f)\mathbf{Q}]\begin{bmatrix} h \\ \omega \end{bmatrix} = \mathbf{u} \tag{8}$$

i.e.

$$\mathbf{A}(x_f, y_f)\mathbf{x}_0 = \mathbf{u}. \tag{9}$$

Replacing the unknowns x_f, y_f and $\mathbf{x}_0$ by the hypothesized variables x_h, y_h and $\mathbf{x}$ we get the general condition

$$\mathbf{A}(x_h, y_h)\mathbf{x} \rightarrow \mathbf{u} \tag{10}$$

where the true solution exactly satisfies (10). The arrow denotes the condition where the left hand side is maximally close in value to the right hand side. The use of an equality sign instead of an arrow to represent this condition is common though incorrect, since the condition may not always be satisfied exactly.

We now define a squared error cost function $C(x_h, y_h, \mathbf{x})$ as

$$C(x_h, y_h, \mathbf{x}) = \|\mathbf{A}(x_h, y_h)\mathbf{x} - \mathbf{u}\|_2^2 \tag{11}$$

Since

$$C(x_h, y_h, \mathbf{x}) \geq 0 \tag{12}$$

$$C(x_f, y_f, \mathbf{x}_0) = 0 \tag{13}$$

(i) the true solution to the system (10) minimizes the cost function $C(\ )$, and (ii) all minimizers of $C(\ )$ satisfy (10) exactly. Thus, we have reduced the original problem to

$$\min_{x_h, y_h, \mathbf{x}} C(x_h, y_h, \mathbf{x}) \tag{14}$$

which can be decomposed as

$$\min_{x_h, y_h} \min_{\mathbf{x}} C(x_h, y_h, \mathbf{x}) \tag{15}$$

The inner minimization occurs at the least squares (LS) solution $\mathbf{x}_{\mathrm{LS}}$ of $\mathbf{A}(x_h, y_h)\mathbf{x} \rightarrow \mathbf{u}$. Later, we will prove uniqueness of $\mathbf{x}_{\mathrm{LS}}$. Indeed, $C(\ )$ can have more than one minimizer. In the separable form (15), existence of multiple minima is indicated by existence of multiple values of x_h, y_h (and thereby of $\mathbf{x}$) attaining the minimum for $C(\ )$. In this situation, search over the entire search space will pick out all the valid solutions. However, it is not difficult to see that even with coarse discretization, the number of free variables is too large to permit exhaustive search. Referring to Appendix A.1, we see that partial search is an ideal technique for solving (15).

In order to perform partial search, we set $\{x_h, y_h\}$ to be the search component and $\mathbf{x}$ to be the dependent component. We discretize the search component space at midway locations between four pixels over the entire image area, in line with our assumption that the FOE lies within the image. Later, we will relax the search component space to be any uniformly discretized rectangular set of points, not necessarily within the image. We do not dispute that one could potentially construct a pathological counter-example that lets the minima "slip through" the lattice formed by discrete values. But it is our belief that such situations do not occur in practice. We will experimentally justify the discretization process.

The LS solution $\mathbf{x}_{\mathrm{LS}}$ of $\mathbf{A}(x_h, y_h)\mathbf{x} \to \mathbf{u}$ satisfies

$$\mathbf{A}'\mathbf{A}\mathbf{x}_{\mathrm{LS}} = \mathbf{A}'\mathbf{u} \tag{16}$$

$$\begin{bmatrix}\mathbf{P}'\mathbf{P} & \mathbf{P}'\mathbf{Q}\\ \mathbf{Q}'\mathbf{P} & \mathbf{Q}'\mathbf{Q}\end{bmatrix}\mathbf{x}_{\mathrm{LS}} = \begin{bmatrix}\mathbf{P}'\\ \mathbf{Q}'\end{bmatrix}\mathbf{u} \tag{17}$$

where the arguments of $\mathbf{A}$ and $\mathbf{P}$ have been dropped. $\mathbf{D} \stackrel{\mathrm{def}}{=} \mathbf{P}'\mathbf{P}$ is a diagonal matrix, given by

$$\begin{aligned}\mathbf{D} &= \mathrm{Diag}\{\bar{x}_i^2 + \bar{y}_i^2\}\\ \mathbf{D}^{-1} &= \mathrm{Diag}\left\{\frac{1}{\bar{x}_i^2 + \bar{y}_i^2}\right\}\end{aligned} \tag{18}$$

where $\bar{x}_i$ and $\bar{y}_i$ are functions of (x_h, y_h):

$$\begin{aligned}\bar{x}_i &= x_{\lfloor i/N \rfloor, i \bmod N} - x_h\\ \bar{y}_i &= y_{\lfloor i/N \rfloor, i \bmod N} - y_h\end{aligned} \tag{19}$$

We shall assume here that $\bar{x}_i$ and $\bar{y}_i$ are never zero. This can be achieved by ensuring that the discretization grid for FOE hypotheses is not coincident with the pixel sampling grid. Even this restriction can be overcome, but we shall deal with it later, assuming for now that $\mathbf{D}$ is never singular. One way of guaranteeing this is to pick the hypothesized FOE to lie midway between pixels. When $\mathbf{D}$ is nonsingular, $\mathbf{x}_{\mathrm{LS}}$ is unique. Applying appropriate pre-multiplying matrices, we can manipulate the system as shown:

$$\begin{aligned}&\begin{bmatrix}\mathbf{D} & \mathbf{P}'\mathbf{Q}\\ \mathbf{Q}'\mathbf{P} & \mathbf{Q}'\mathbf{Q}\end{bmatrix}\mathbf{x} = \begin{bmatrix}\mathbf{P}'\\ \mathbf{Q}'\end{bmatrix}\mathbf{u}\\ \Rightarrow &\begin{bmatrix}\mathbf{I} & \mathbf{D}^{-1}\mathbf{P}'\mathbf{Q}\\ \mathbf{Q}'\mathbf{P} & \mathbf{Q}'\mathbf{Q}\end{bmatrix}\mathbf{x} = \begin{bmatrix}\mathbf{D}^{-1}\mathbf{P}'\\ \mathbf{Q}'\end{bmatrix}\mathbf{u}\\ \Rightarrow &\begin{bmatrix}\mathbf{I} & \mathbf{D}^{-1}\mathbf{P}'\mathbf{Q}\\ \mathbf{0} & \mathbf{Q}'\mathbf{Q} - \mathbf{Q}'\mathbf{P}\mathbf{D}^{-1}\mathbf{P}'\mathbf{Q}\end{bmatrix}\mathbf{x} = \begin{bmatrix}\mathbf{D}^{-1}\mathbf{P}'\\ \mathbf{Q}'(\mathbf{I} - \mathbf{P}\mathbf{D}^{-1}\mathbf{P}')\end{bmatrix}\mathbf{u}\end{aligned} \tag{20}$$

Introducing matrices $\mathbf{M} \in \Re^{2N \times 2N}$ and $\hat{\mathbf{M}} \in \Re^{3\times 3}$ defined as

$$\begin{aligned}\mathbf{M} &\stackrel{\mathrm{def}}{=} (\mathbf{I} - \mathbf{P}\mathbf{D}^{-1}\mathbf{P}')\\ &= \mathrm{BlockDiag}\left\{\frac{1}{\bar{x}_i^2 + \bar{y}_i^2}\begin{bmatrix}\bar{y}_i^2 & -\bar{x}_i\bar{y}_i\\ -\bar{x}_i\bar{y}_i & \bar{y}_i^2\end{bmatrix}\right\}\end{aligned} \tag{21}$$

$$\hat{\mathbf{M}} \stackrel{\mathrm{def}}{=} \mathbf{Q}'\mathbf{M}\mathbf{Q} \tag{22}$$

we get

$$\begin{aligned}&\begin{bmatrix}\mathbf{I} & \mathbf{D}^{-1}\mathbf{P}'\mathbf{Q}\\ \mathbf{0} & \hat{\mathbf{M}}\end{bmatrix}\mathbf{x} = \begin{bmatrix}\mathbf{D}^{-1}\mathbf{P}'\\ \mathbf{Q}'\mathbf{M}\end{bmatrix}\mathbf{u}\\ \Rightarrow &\begin{bmatrix}\mathbf{I} & \mathbf{D}'\mathbf{P}'\mathbf{Q}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}\mathbf{x} = \begin{bmatrix}\mathbf{D}^{-1}\mathbf{P}'\\ \hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M}\end{bmatrix}\mathbf{u}\\ &\Rightarrow \mathbf{x} = \begin{bmatrix}\mathbf{D}^{-1}\mathbf{P}'(\mathbf{I} - \mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M})\\ \hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M}\end{bmatrix}\mathbf{u}\end{aligned} \tag{23}$$

The error $\mathbf{A}\mathbf{x} - \mathbf{u}$ is

$$\begin{aligned}\mathbf{A}\mathbf{x} - \mathbf{u} &= [\mathbf{P} \quad \mathbf{Q}]\begin{bmatrix}\mathbf{D}^{-1}\mathbf{P}'(\mathbf{I} - \mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M})\\ \hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M}\end{bmatrix}\mathbf{u} - \mathbf{u}\\ &= (\mathbf{M}\mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M} - \mathbf{M})\mathbf{u}\end{aligned} \tag{24}$$

giving the squared error

$$\begin{aligned}\|\mathbf{A}\mathbf{x} - \mathbf{u}\|^2 &= \mathbf{u}'(\mathbf{M}\mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M} - \mathbf{M})^2\mathbf{u}\\ &= \mathbf{u}'\mathbf{M}^2\mathbf{u} + \mathbf{u}'\mathbf{M}\mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M}^2\mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M}\mathbf{u}\\ &\quad -2\mathbf{u}'\mathbf{M}^2\mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M}\mathbf{u}\end{aligned} \tag{25}$$

Again, note that $\mathbf{A}$, $\mathbf{M}$ and $\hat{\mathbf{M}}$ are functions of (x_h, y_h). It can be easily verified that $\mathbf{M}$ is idempotent, i.e. $\mathbf{M}^2 = \mathbf{M}$. (25) simplifies to

$$\|\mathbf{A}\mathbf{x} - \mathbf{u}\|^2 = \mathbf{u}'\mathbf{M}\mathbf{u} - \mathbf{u}'\mathbf{M}\mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M}\mathbf{u} \tag{26}$$

The most naive strategy for computing the least squared error, or equivalently, of performing the inner minimization in (15), is to explicitly solve the linear system for the unknown $\mathbf{x}$ and use this estimate to evaluate the squared error. Without assuming any sparseness or symmetry in the coefficient matrix $\mathbf{A}$, the system can be solved by matrix inversion. Keeping in mind that $\mathbf{x} \in \Re^{N^2+3}$, the order of complexity of estimating $\mathbf{x}$ in this manner is $\mathcal{O}(N^6)$.[1] Exploiting the structure of $\mathbf{A}$ leads to dramatic improvements. From (21) and (22), it can be seen that computing matrices $\mathbf{M}$ and $\hat{\mathbf{M}}$ requires $\mathcal{O}(N^2)$ operations, which is the complexity of estimating $\mathbf{x}$ and the squared error as well. Taking into account the outer minimization search leads to an overall complexity of $\mathcal{O}(N^8)$ for the matrix inversion method and $\mathcal{O}(N^4)$ by exploiting symmetry. In addition, examining (26) reveals that the only data-dependent term is $\mathbf{u}$. Thus, given sufficient memory, the data-independent terms can be pre-computed and stored, for each value of (x_h, y_h). However, even with this strategy, the overall complexity cannot be brought down below $\mathcal{O}(N^4)$.

In what forms the core of this effort, we will show that the structure in **A** can be further exploited so that the errors can be computed *directly*, without computing the solution **x** explicitly. Moreover, the least squared errors for all the candidate hypotheses can be computed in a single step, which leads to an overall complexity of $\mathcal{O}(N^2 \log N)$. At first, this seems ridiculous since factoring out N^2 from the complexity introduced by the outer search leaves $\mathcal{O}(\log N)$ which is insufficient even for vector addition. Yet, it is the simultaneous estimation of all errors in the search space that allows such a low overall complexity. We introduce the notion of *Fast Computability* in the following section.

4.1. Fast Computability

A few preliminary definitions and theorems are necessary before we proceed with the proof of our technique. Fast Computability is a novel concept introduced by us, that deals with the evaluation of a quantity which is defined over an appropriate space. In this discussion, the quantity being evaluated is a scalar valued 2D field. The physical relevance of this field is that its value at a certain coordinate is a measure of the goodness of fit subject to a certain hypothesis. The effect of shifting the hypothesis is measured by re-examining the field at different coordinates. Thus, if the field is evaluated over the entire coordinate space, this is tantamount to examining the performance of all possible hypotheses. In practice, we choose to evaluate such a field over a bounded and discrete space, corresponding to a finely sampled set of hypotheses.

Fast Computability is the property of the quantity or field that permits its evaluation in a rapid manner, using mathematical properties of the field. We define the threshold for Fast Computability to be $\mathcal{O}(N^2 \log N)$. In other words, if an *entire* $N \times N$ field can be evaluated in $\mathcal{O}(N^2 \log N)$ computational steps, it is said to be Fast Computable. Clearly, the minimum computational complexity for evaluating an $N \times N$ field is bounded on the lower side by $\mathcal{O}(N^2)$. The label *Fast* is inspired by the Fast Fourier Transform (FFT), which in 2D is an $\mathcal{O}(N^2 \log N)$ operation. The apparent similarity in label and order of complexity is no mere coincidence. Our approach towards evaluating the goodness of fit field breaks down the quantity into a series of convolutions, which are in turn implemented efficiently using FFTs. This is covered in depth in the following paragraphs.

Definition 1. Let $\mathbf{S} \in \Re^{N \times N}$. The cyclic shift $\mathbf{S}[i_0, j_0]$ of $\mathbf{S}$ by (i_0, j_0) is defined by

$$\mathbf{S}[i_0, j_0]_{i,j} = \mathbf{S}_{(i+i_0) \bmod N, (j+j_0) \bmod N}$$

Definition 2. The lexical ordering of a matrix $\mathbf{S} \in \Re^{N \times N}$, denoted by $\mathcal{S} \in \Re^{N^2 \times N^2}$, is defined by

$$\mathcal{S}_{i,j} = [\mathcal{L}(\mathbf{S})]_{i,j} = \begin{cases} 0 & i \neq j \\ \mathbf{S}_{\lfloor i/N \rfloor, i \bmod N} & i = j \end{cases}$$

$\mathcal{L}$ is the lexical ordering operator. The inverse operation is defined only on diagonal matrices:

$$[\mathcal{L}^{-1}(\mathrm{Diag}(s_k))]_{i,j} = s_{iN+j}$$

We denote the space of such diagonal matrices by $\mathcal{L}^{N^2}(\Re)$, where the argument $\Re$ denotes the space of each element s_i of the matrix $\mathcal{S}$.

Definition 3. The cyclic shift of a lexically ordered matrix $\mathcal{S} \in \mathcal{L}^{N^2}(\Re)$, by (i_0, j_0), is

$$\mathcal{S}[i_0, j_0] = \mathcal{L}(\mathcal{L}^{-1}(\mathcal{S}[i_0, j_0])) = \mathcal{L}(\mathbf{S}[i_0, j_0]) \quad (27)$$

Definition 4. The quantity $q(i_0, j_0)$ is said to be Fast Computable (FC) if $q(i_0, j_0)$ can be evaluated $\forall i_0, j_0 \in \{0, 1, \ldots, N-1\}$ in $O(N^2 \log N)$ computational steps.

Theorem 1. *Let* $a, b \in \Re^{N^2}$ *and* $\mathcal{S} \in \mathcal{L}^{N^2}(\Re)$. *The quantity* $q(i_0, j_0) \in \Re$ *defined by* $q(i_0, j_0) = a' \mathcal{S}[i_0, j_0] b$ *is FC.*

Proof:

$$\begin{aligned} q(i_0, j_0) &= a' \mathcal{S}[i_0, j_0] b \\ &= \sum_{i=0}^{N^2-1} a_i b_i \mathcal{S}_{i,i}[i_0, j_0] \\ &= \sum_{i=0}^{N^2-1} a_i b_i \mathbf{S}_{\lfloor i/N \rfloor, i \bmod N}[i_0, j_0] \\ &= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} a_{iN+j} b_{iN+j} \mathbf{S}_{i,j}[i_0, j_0] \end{aligned}$$

$$= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \mathbf{C}_{i,j} \mathbf{S}_{i,j}[i_0, j_0] \tag{28}$$

$$= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \mathbf{C}_{i,j} \mathbf{S}_{(i+i_0) \bmod N, (j+j_0) \bmod N} \tag{29}$$

where the $N \times N$ matrix $\mathbf{C}$ is comprised of component-wise products, $\mathbf{C}_{i,j} = a_{iN+j} b_{iN+j}$. (29) is a 2-D spatial correlation of the "image" $\mathbf{C}$ with a sliding "template" $\mathbf{S}$. Each element of the resultant matrix after correlation is the value of $q(i_0, j_0)$ for the appropriate shift. This operation can be performed in $O(N^2 \log N)$ operations using Fourier domain techniques.

Let $\mathcal{F}$ be the discrete Fourier transform operator. Form the $N \times N$ matrix Q *s.t.* $Q_{i,j} = q(i, j)$. We have

$$\mathcal{F}_Q(k, l) = \mathcal{F}_{\mathbf{C}}(-k, -l) \mathcal{F}_{\mathbf{S}}(k, l) \tag{30}$$

where the discrete Fourier transforms $\mathcal{F}_Q$, $\mathcal{F}_{\mathbf{C}}$ and $\mathcal{F}_{\mathbf{S}}$ are defined on the matrices $\mathbf{C}$, $\mathbf{S}$ and Q respectively. We can use the Fast Fourier Transform to compute the terms in (30), when $N = 2^k$, in $\mathcal{O}(N^2 \log N)$ steps. The product in Fourier space takes $\mathcal{O}(N^2)$ operations, requiring an overall complexity of $\mathcal{O}(N^2 \log N)$ for the computation of $q(i_0, j_0)$. □

We now extend Theorem 1 to a more complicated situation where each scalar element of the above data structures, including matrices and vector, is replaced by a *doublet*. The doublet corresponding to a scalar matrix entry is defined as a 2×2 submatrix and a doublet of a vector component as a 2-vector. In other words, each scalar element in the matrices is replaced by a 2×2 real matrix, and each scalar element in the vectors by a 2×1 real vector. The concepts of cyclic shift, lexical ordering and inverse lexical ordering are redefined below.

Definition 5. Let

$$\mathbf{S}_{i,j} = \begin{pmatrix} \mathbf{S}_{i,j_{00}} & \mathbf{S}_{i,j_{01}} \\ \mathbf{S}_{i,j_{10}} & \mathbf{S}_{i,j_{11}} \end{pmatrix} \in \Re^{2 \times 2}$$

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_{0,0} & \mathbf{S}_{0,1} & & \mathbf{S}_{0,N-1} \\ \mathbf{S}_{1,0} & \mathbf{S}_{1,1} & \cdots & \mathbf{S}_{1,N-1} \\ & \vdots & \ddots & \\ \mathbf{S}_{N-1,0} & \mathbf{S}_{N-1,1} & & \mathbf{S}_{N-1,N-1} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{S}_{0,0_{00}} & \mathbf{S}_{0,0_{01}} & \mathbf{S}_{0,1_{00}} & & \mathbf{S}_{0,N-1_{01}} \\ \mathbf{S}_{0,0_{10}} & \mathbf{S}_{0,0_{11}} & \mathbf{S}_{0,1_{10}} & \cdots & \mathbf{S}_{0,N-1_{11}} \\ \mathbf{S}_{1,0_{00}} & \mathbf{S}_{1,0_{01}} & \mathbf{S}_{1,1_{00}} & & \mathbf{S}_{1,N-1_{01}} \\ & \vdots & & \ddots & \\ \mathbf{S}_{N-1,0_{10}} & \mathbf{S}_{N-1,0_{11}} & \mathbf{S}_{N-1,1_{10}} & & \mathbf{S}_{N-1,N-1_{11}} \end{bmatrix}$$

$$\in \Re^{2N \times 2N} \tag{31}$$

The cyclic shift $\mathbf{S}[i_0, j_0]$ of $\mathbf{S}$ by (i_0, j_0) is defined by

$$\mathbf{S}[i_0, j_0]_{i,j} = \mathbf{S}_{(i+i_0) \bmod N, (j+j_0) \bmod N}$$

i.e.

$$\mathbf{S}[i_0, j_0] = \begin{bmatrix} \mathbf{S}_{i_0 \bmod N, j_0 \bmod N} & \cdots & \mathbf{S}_{i_0 \bmod N, (N-1+j_0) \bmod N} \\ \vdots & \ddots & \\ \mathbf{S}_{(N-1+i_0) \bmod N, j_0 \bmod N} & & \mathbf{S}_{(N-1+i_0) \bmod N, (N-1+j_0) \bmod N} \end{bmatrix} \tag{32}$$

Definition 6. The lexical ordering of the doublet matrix $\mathbf{S}$ shown in (31), denoted by $\mathcal{S} \in \Re^{2N^2 \times 2N^2}$, is defined by

$$\mathcal{S} = \text{BlockDiag}\left\{\mathbf{S}_{\lfloor i/N \rfloor, i \bmod N}\right\} = \text{BlockDiag}\left\{\begin{pmatrix} \mathbf{S}_{\lfloor i/N \rfloor, i \bmod N_{00}} & \mathbf{S}_{\lfloor i/N \rfloor, i \bmod N_{01}} \\ \mathbf{S}_{\lfloor i/N \rfloor, i \bmod N_{10}} & \mathbf{S}_{\lfloor i/N \rfloor, i \bmod N_{11}} \end{pmatrix}\right) \tag{33}$$

In keeping with Definition 2, we will denote the space of permissible $\mathcal{S}$ matrices which constitute the lexical ordering of a doublet matrix by $\mathcal{L}^{N^2}(\Re^{2 \times 2})$.

Definition 7. The cyclic shift of a lexically ordered matrix $\mathcal{S} \in \mathcal{L}^{N^2}(\Re^{2 \times 2})$ is defined by (27).

Theorem 2. *Let* $\mathcal{S} \in \mathcal{L}^{N^2}(\Re^{2 \times 2})$, *and* $a, b \in \Re^{2N^2}$. *The quantity* $q(i_0, j_0) = a' \mathcal{S}[i_0, j_0] b$ *is FC.*

Proof: Writing out a and b in terms of their components

$$\begin{aligned} a &= (a_{0,0} \;\; a_{0,1} \;\; a_{1,0} \;\; a_{1,1} \;\; \ldots \;\; a_{N-1,0} \;\; a_{N-1,1})' \\ b &= (b_{0,0} \;\; b_{0,1} \;\; b_{1,0} \;\; b_{1,1} \;\; \ldots \;\; b_{N-1,0} \;\; b_{N-1,1})' \end{aligned} \tag{34}$$

and using

$$\mathcal{S}[i_0, j_0] = \text{BlockDiag}\left\{\mathbf{S}_{\lfloor i/N \rfloor, i \bmod N}[i_0, j_0]\right\}$$

we get

$$\begin{aligned}
q(i_0, j_0) &= a'\mathcal{S}[i_0, j_0]b \\
&= \sum_{i=0}^{N^2-1} \big\{ a_{i,0}b_{i,0}\mathbf{S}_{\lfloor i/N \rfloor, i \bmod N_{00}}[i_0, j_0] + a_{i,0}b_{i,1}\mathbf{S}_{\lfloor i/N \rfloor, i \bmod N_{01}}[i_0, j_0] \\
&\quad + a_{i,1}b_{i,0}\mathbf{S}_{\lfloor i/N \rfloor, i \bmod N_{10}}[i_0, j_0] + a_{i,1}b_{i,1}\mathbf{S}_{\lfloor i/N \rfloor, i \bmod N_{11}}[i_0, j_0] \big\} \\
&= \sum_{i=0}^{N^2-1} \sum_{2} \begin{bmatrix} a_{i,0}b_{i,0}\mathbf{S}_{\lfloor i/N \rfloor, i \bmod N_{00}}[i_0, j_0] & a_{i,0}b_{i,1}\mathbf{S}_{\lfloor i/N \rfloor, i \bmod N_{01}}[i_0, j_0] \\ a_{i,1}b_{i,0}\mathbf{S}_{\lfloor i/N \rfloor, i \bmod N_{10}}[i_0, j_0] & a_{i,1}b_{i,1}\mathbf{S}_{\lfloor i/N \rfloor, i \bmod N_{11}}[i_0, j_0] \end{bmatrix} \\
&= \sum_{2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \begin{bmatrix} \mathbf{C}_{i,j_{00}}\mathbf{S}_{i,j_{00}}[i_0, j_0] & \mathbf{C}_{i,j_{01}}\mathbf{S}_{i,j_{01}}[i_0, j_0] \\ \mathbf{C}_{i,j_{10}}\mathbf{S}_{i,j_{10}}[i_0, j_0] & \mathbf{C}_{i,j_{11}}\mathbf{S}_{i,j_{11}}[i_0, j_0] \end{bmatrix} \\
&= \sum_{2} \begin{bmatrix} \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \mathbf{C}_{i,j_{00}}\mathbf{S}_{i,j_{00}}[i_0, j_0] & \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \mathbf{C}_{i,j_{01}}\mathbf{S}_{i,j_{01}}[i_0, j_0] \\ \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \mathbf{C}_{i,j_{10}}\mathbf{S}_{i,j_{10}}[i_0, j_0] & \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \mathbf{C}_{i,j_{11}}\mathbf{S}_{i,j_{11}}[i_0, j_0] \end{bmatrix}
\end{aligned} \tag{35}$$

$\sum_2$ denotes the sum of the four entries of the 2×2 matrix summand. $\mathbf{C} \in \Re^{2N \times 2N}$ is formed by the componentwise product of a and b:

$$\begin{aligned}
\mathbf{C}_{i,j} &= \begin{pmatrix} \mathbf{C}_{i,j_{00}} & \mathbf{C}_{i,j_{01}} \\ \mathbf{C}_{i,j_{10}} & \mathbf{C}_{i,j_{11}} \end{pmatrix} \\
&= \begin{pmatrix} a_{k,0}b_{k,0} & a_{k,0}b_{k,1} \\ a_{k,1}b_{k,0} & a_{k,1}b_{k,1} \end{pmatrix}, \quad k = iN + j
\end{aligned}$$

Each of the entries of the summand matrix in (35) is of the form (28) and is FC according to Theorem 1. The four components of the N^2 signals for each pair (i_0, j_0) can be summed in $O(N^2)$ operations. The overall complexity of evaluating $q(i_0, j_0)$ over the permissible values of i_0 and j_0 is $\mathcal{O}(N^2 \log N)$. Hence, $q(i_0, y_0)$ is FC. □

4.2. Non-Cyclic Shift

The last step in setting the stage to prove fast computability of the squared error is extending Theorem 2 to non-cyclic shifts. When a space-limited data sequence is shifted across a viewing window, points that were undefined earlier appear within the window. In a cyclic shift, the data points of the original sequence that disappear from the viewing area are wrapped around to fill the locations within the newly visible area. However, when the shift is not cyclic, it is necessary to define its behavior, especially with regard to how emerging areas are filled in.

Definition 8. The non-cyclic shift $\mathbf{S}\langle i_0, j_0\rangle$ of matrix $\mathbf{S} \in \Re^{N \times N}$ by (i_0, j_0) is defined in terms of the superset matrix $\bar{\mathbf{S}} \in \Re^{2N \times 2N}$ by

$$\mathbf{S}\langle i_0, j_0\rangle_{i,j} = \bar{\mathbf{S}}_{i+i_0, j+j_0}, \quad \forall i_0, j_0 \in \{1, 1, \ldots, N-1\}.$$

Definition 9. The non-cyclic shift of a lexically ordered matrix $\mathcal{S} \in \mathcal{L}^{N^2}(\Re)$ is defined by

$$\mathcal{S}[i_0, j_0] = \mathcal{L}(\mathbf{S}\langle i_0, j_0\rangle)$$

Theorem 3. *Let $a, b \in \Re^{N^2}$ and $\mathcal{S} \in \mathcal{L}^{N^2}(\Re)$. The quantity $q(i_0, j_0) \in \Re$ defined by $q(i_0, j_0) = a'\mathcal{S}\langle i_0, j_0\rangle b$ is FC.*

Proof:

$$\begin{aligned}
q(i_0, j_0) &= a'\mathcal{S}\langle i_0, j_0\rangle b \\
&= \sum_{i=0}^{N^2-1} a_i b_i \bar{\mathbf{S}}_{\lfloor i/N \rfloor + i_0, i \bmod N + j_0} \\
&= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} a_{iN+j} b_{iN+j} \bar{\mathbf{S}}_{i+i_0, j+j_0}
\end{aligned} \tag{36}$$

Define $\mathbf{C} \in \Re^{2N \times 2N}$ by

$$\mathbf{C}_{i,j} = \begin{cases} a_{iN+j}b_{iN+j} & i, j \in \{0, 1, \ldots, N-1\} \\ 0 & \text{otherwise.} \end{cases}$$

Noting that

$$\hat{\mathbf{S}}_{i+i_0, j+j_0} = \hat{\mathbf{S}}_{(i+i_0) \bmod (2N), (j+j_0) \bmod (2N)}, \quad \forall i, j, i_0, j_0 \in 0, 1, \ldots, N-1 \tag{37}$$

we get

$$q(i_0, j_0) = \sum_{i=0}^{2N-1} \sum_{j=0}^{2N-1} \mathbf{C}_{i,j} \hat{\mathbf{S}}_{(i+i_0) \bmod (2N), (j+j_0) \bmod (2N)}, \quad \forall i_0, j_0 \in \{0, 1, \ldots, N-1\} \quad (38)$$

which is of the form (29), with $2N$ replacing N. Thus $q(i_0, j_0)$ can be computed in $\mathcal{O}((2N)^2 \log(2N)) = \mathcal{O}(N^2 \log N)$ operations, proving the theorem. Note that the Fourier technique will calculate $q(i_0, j_0)$ over a larger domain, viz. $\{0, 1, \ldots, 2N-1\}^2$, of which the useful values are lie within the "North-West" quarter of the matrix. □

We now extend Theorem 3 to the doublet space.

Theorem 4. *Let $a, b \in \Re^{2N^2}$ and $\mathcal{S} \in \mathcal{L}^{N^2}(\Re^{2\times 2})$. The quantity $q(i_0, j_0) \in \Re$ defined by $q(i_0, j_0) = a'\mathcal{S}\langle i_0, j_0\rangle b$ is FC.*

Proof: Writing out a and b in terms of their components as in (34) we get

$$\begin{aligned} q(i_0, j_0) &= a'\mathcal{S}\langle i_0, j_0\rangle b \\ &= \sum_{i=0}^{N^2-1} \Big\{ a_{i,0}\, b_{i,0} \big[\bar{\mathbf{S}}_{\lfloor i/N \rfloor + i_0, i \bmod N + j_0}\big]_{00} + a_{i,0} b_{i,1} \big[\bar{\mathbf{S}}_{\lfloor i/N \rfloor + i_0, i \bmod N + j_0}\big]_{01} \\ &\qquad + a_{i,1}\, b_{i,0} \big[\bar{\mathbf{S}}_{\lfloor i/N \rfloor + i_0, i \bmod N + j_0}\big]_{10} + a_{i,1} b_{i,1} \big[\bar{\mathbf{S}}_{\lfloor i/N \rfloor + i_0, i \bmod N + j_0}\big]_{11} \Big\} \\ &= \sum_{2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \begin{bmatrix} a_{iN+j,0} b_{iN+j,0} \big[\bar{\mathbf{S}}_{i+i_0, j+j_0}\big]_{00} & a_{iN+j,0} b_{iN+j,1} \big[\bar{\mathbf{S}}_{i+i_0, j+j_0}\big]_{01} \\ a_{iN+j,1} b_{iN+j,0} \big[\bar{\mathbf{S}}_{i+i_0, j+j_0}\big]_{10} & a_{iN+j,1} b_{iN+j,1} \big[\bar{\mathbf{S}}_{i+i_0, j+j_0}\big]_{11} \end{bmatrix} \end{aligned} \quad (39)$$

As in 3, we define $\mathbf{C} \in \Re^{4N \times 4N}$ by

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_{0,0} & \cdots & \mathbf{C}_{0,2N-1} \\ \vdots & \ddots & \\ \mathbf{C}_{2N-1,0} & & \mathbf{C}_{2N-1,2N-1} \end{bmatrix}$$

$$\mathbf{C}_{i,j} = \begin{cases} \begin{bmatrix} a_{iN+j,0} b_{iN+j,0} & a_{iN+j,0} b_{iN+j,1} \\ a_{iN+j,1} b_{iN+j,0} & a_{iN+j,1} b_{iN+j,1} \end{bmatrix} & i, j \in \{0, 1, \ldots, N-1\} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \text{otherwise.} \end{cases} \quad (40)$$

Using the reasoning in (37), we get

$$q(i_0, j_0) = \sum_{2} \begin{bmatrix} q_{00} & q_{01} \\ q_{10} & q_{11} \end{bmatrix}, \quad \forall i_0, j_0 \in \{0, 1, \ldots, N-1\}$$

$$\begin{aligned} q_{00} &= \sum_{i=0}^{2N-1} \sum_{j=0}^{2N-1} \mathbf{C}_{i,j_{00}} \big[\hat{\mathbf{S}}_{(i+i_0) \bmod (2N), (j+j_0) \bmod (2N)}\big]_{00} \\ q_{01} &= \sum_{i=0}^{2N-1} \sum_{j=0}^{2N-1} \mathbf{C}_{i,j_{01}} \big[\hat{\mathbf{S}}_{(i+i_0) \bmod (2N), (j+j_0) \bmod (2N)}\big]_{01} \\ q_{10} &= \sum_{i=0}^{2N-1} \sum_{j=0}^{2N-1} \mathbf{C}_{i,j_{10}} \big[\hat{\mathbf{S}}_{(i+i_0) \bmod (2N), (j+j_0) \bmod (2N)}\big]_{10} \\ q_{11} &= \sum_{i=0}^{2N-1} \sum_{j=0}^{2N-1} \mathbf{C}_{i,j_{11}} \big[\hat{\mathbf{S}}_{(i+i_0) \bmod (2N), (j+j_0) \bmod (2N)}\big]_{11} \end{aligned} \quad (41)$$

As before, $\sum_2$ denotes the sum over the four components of the summand matrix. Each of the terms q_{kl} is of the form (38) and is FC by Theorem 3. Since the components themselves can be summed in $\mathcal{O}(N^2)$ operations, the overall complexity in computing $q(i_0, j_0)$ over the domain $\{0, 1, \ldots, N-1\}^2$ is $\mathcal{O}(N^2 \log N)$, proving the theorem. □

Finally, we extend the above theorem to arbitrary finite-dimensional aggregations in the following results.

Corollary 1. *Let $\mathcal{S} \in \mathcal{L}^{N^2}(\Re^{2\times 2})$, $a \in \Re^{2N^2}$ and $B \in \Re^{2N^2 \times p}$ for some arbitrary integer $p \geq 1$. The quantity $q(i_0, j_0) = a'\mathcal{S}\langle i_0, j_0\rangle B$ is FC.*

Proof: Writing out $q(i_0, j_0)$ in terms of its components,

$$q(i_0, j_0) = \begin{pmatrix} a'\mathcal{S}[i_0, j_0]b_0 & a'\mathcal{S}[i_0, j_0]b_1 \\ \ldots & a'\mathcal{S}[i_0, j_0]b_{p-1} \end{pmatrix}$$

where $B = (b_0\ b_1 \ldots b_{p-1})$. Each of the components of $q(i_0, j_0)$ is FC. When p is a constant, the overall computational steps are still $O(N^2 \log N)$. □

Corollary 2. *Likewise, let* $A, B \in \Re^{2N^2 \times p}$, p *being an arbitrary constant.* $Q(i_0, j_0) = A'\mathcal{S}[i_0, j_0]B$ *is FC.*

Proof: As in Corollary 1. □

4.3. Basic Proof

Next, we show how the squared error given by Eq. (26) in Fast Computable, for every choice of (x_f, y_f). This enables the likely solution space to be searched exhaustively in $\mathcal{O}(N^2 \log N)$ steps. First, we use (6) to define the pixel coordinate (i_h, j_h) corresponding to the hypothesized FOE (x_h, y_h) as

$$\begin{pmatrix} i_h \\ j_j \end{pmatrix} = \begin{pmatrix} \frac{x_h}{\delta} + \frac{N-1}{2} \\ \frac{y_h}{\delta} + \frac{N-1}{2} \end{pmatrix}$$

We define the search space for (i_h, j_h) to be $\{\frac{1}{2}, \frac{3}{2}, \ldots, N - \frac{1}{2}\}^2$. The candidate solutions lie at half-pixel displacements along a regular grid covering the image area. In the discussion ahead, we will interchangably use the pixel and *XY* coordinate systems.

Lemma 1. $\mathbf{M} \in \mathcal{L}^{N^2}(\Re^{2\times 2})$

Proof: From (21) and Definition 6. □

Definition 10. The superset matrix $\bar{\mathbf{M}} \in \Re^{4N \times 4N}$ is a doublet matrix defined by

$$\bar{\mathbf{M}}_{i,j} = \frac{\begin{bmatrix} \left(j - N + \frac{1}{2}\right)^2 & -\left(i - N + \frac{1}{2}\right)\left(j - N + \frac{1}{2}\right) \\ -\left(i - N + \frac{1}{2}\right)\left(j - N + \frac{1}{2}\right) & \left(i - N + \frac{1}{2}\right)^2 \end{bmatrix}}{\left(i - N + \frac{1}{2}\right)^2 + \left(j - N + \frac{1}{2}\right)^2} \quad \forall i, j \in \{0, 1, \ldots, 2N-1\}. \quad (42)$$

In the above definition, the $\frac{1}{2}$ terms are included to stagger the grid. There is a one-to-one relation between the hypothesized location (i_h, j_h) and the index (i, j) according to

$$\{i_h, j_h\} \leftrightarrow \{i, j\} + \frac{1}{2} \quad (43)$$

The importance of staggering the sampling grid for the search component is seen from (42), where the $\frac{1}{2}$ terms prevent the denominator of the leading fraction from vanishing. However, it must be noted that this fix is merely cosmetic since even without staggering, the entry $\bar{\mathbf{M}}_{N,N}$ at the singular point can be defined using limits.

Lemma 2. $\mathbf{M}(x_h, y_h) = \mathbf{M}\langle N - 1 - \lfloor i_h \rfloor, N - 1 - \lfloor j_h \rfloor\rangle$ *to within half-pixel discretization.*

Proof: From (19), (21) and (6), we have

$$\mathcal{L}^{-1}(\mathbf{M}(x_h, y_h))_{i,j} = \frac{\begin{bmatrix} (y_{i,j} - y_h)^2 & -(x_{i,j} - x_h)(y_{i,j} - y_h) \\ -(x_{i,j} - x_h)(y_{i,j} - y_h) & (x_{i,j} - x_h)^2 \end{bmatrix}}{(x_{i,j} - x_h)^2 + (y_{i,j} - y_h)^2}$$

$$= \frac{\begin{bmatrix} \left(j - \frac{N-1}{2} - f y_h\right)^2 & -\left(i - \frac{N-1}{2} - f x_h\right)\left(j - \frac{N-1}{2} - f y_h\right) \\ -\left(i - \frac{N-1}{2} - f x_h\right)\left(j - \frac{N-1}{2} - f y_h\right) & \left(i - \frac{N-1}{2} - f y_h\right)^2 \end{bmatrix}}{\left(i - \frac{N-1}{2} - f x_h\right)^2 + \left(j - \frac{N-1}{2} - f y_h\right)^2} \quad (44)$$

Comparing (42) with (44), we see that $\mathcal{L}^{-1}(\mathbf{M})$ is a windowed version of the superset matrix $\hat{\mathbf{M}}$. The location of this window (k, l) is computed by equating the indices. In general, since the computed location may not be integral, rounding is performed. The following equation evaluates k. l is evaluated likewise.

$$\begin{aligned} k - N + \frac{1}{2} \rightarrow i - \frac{N-1}{2} - f x_h \\ k &= \left\lceil i + N - f x_h - \frac{N+1}{2} \right\rceil \\ &= i + (N-1) - \left\lfloor f x_h - \frac{N-1}{2} \right\rfloor \\ &= i + (N-1) - \lfloor i_h \rfloor \end{aligned} \quad (45)$$

Thus, upon discretization, we get

$$\mathcal{L}^{-1}(\mathbf{M}(x_h, y_h))_{i,j} = \bar{\mathbf{M}}_{i+(N-1)-\lfloor i_h \rfloor, j+(N-1)-\lfloor j_h \rfloor}$$

which gives the non-cyclic shift relationship in mixed coordinate system notation

$$\mathbf{M}(x_h, y_h) = \mathbf{M}\langle N - 1 - \lfloor i_h \rfloor, \quad N - 1 - \lfloor j_h \rfloor\rangle$$

This result is key to tying in fast computation with the SFM problem. Lemma 2 maps the FOE search space $[-\frac{N}{2}, \frac{N}{2}]^2$ to the discrete non-cyclic shifts i_0, $j_0 \in \{N-1, N-2, \ldots, 1, 0\}$ of $\mathbf{M}\langle i_0, j_0\rangle$. □

Lemma 3. $\hat{\mathbf{M}}$ *is FC.*

Proof:

$$\hat{\mathbf{M}} = \mathbf{Q}'\mathbf{M}(x_f, y_f)\mathbf{Q} = \mathbf{Q}'\mathbf{M}\langle i_0, j_0\rangle\mathbf{Q}$$

which is FC, from Corollary 2 and Lemma 2, since $\mathbf{Q} \in \Re^{2N^2\times 3}$. □

Lemma 4. $\hat{\mathbf{M}}^{-1}$ *is FC.*

Proof: From Lemma 3, $\hat{\mathbf{M}}$ can be evaluated over the search space in $\mathcal{O}(N^2 \log N)$ operations. A 3×3 matrix can be inverted in a constant number p of operations. Inverting $\hat{\mathbf{M}}$ over all the N^2 hypotheses takes $N^2 p$ operations. The overall complexity is still $\mathcal{O}(N^2 \log N)$ and hence $\hat{\mathbf{M}}^{-1}$ is FC. □

Lemma 5. $\mathbf{u}'\mathbf{M}\mathbf{u}$ *and* $\mathbf{Q}'\mathbf{M}\mathbf{u}$ *are FC.*

Proof: From Corollary 1 and Corollary 2. □

Theorem 5. *The squared error* (26) *is FC.*

Proof: Evaluating (26) in the manner indicated by the underbraces in the equation

$$\|\mathbf{A}\mathbf{x} - \mathbf{u}\|^2 = \underbrace{\mathbf{u}'\mathbf{M}\mathbf{u}} - \underbrace{\mathbf{u}'\mathbf{M}\mathbf{Q}}\underbrace{[\hat{\mathbf{M}}]^{-1}}\underbrace{\mathbf{Q}'\mathbf{M}\mathbf{u}} \quad (46)$$

groups the right-hand side as a sum or product of Fast Computable terms. The products are among 3×3 matrices and 3×1 vectors. The product and sum operations themselves require $O(N^2)$ computations to evaluate over the entire search space. The overall complexity is $O(N^2 \log N)$, and the squared error is FC. □

In (46), it can be seen that the terms $\mathbf{u}'\mathbf{M}\mathbf{Q}$ and $\mathbf{Q}'\mathbf{M}\mathbf{u}$ are transposes of each other. There are, in all, three quantities whose evaluation is based on the Fast Computation theorems, viz. $\mathbf{u}'\mathbf{M}\mathbf{u}$, $\mathbf{u}'\mathbf{M}\mathbf{Q}$ and $\mathbf{Q}'\mathbf{M}\mathbf{Q}$. Of these, the last term is data-independent. In our basic solution, therefore, $\hat{\mathbf{M}}$ and its inverse can be computed beforehand. Likewise, the Fourier transform of the superset matrix $\bar{\mathbf{M}}$, which is evaluated for computing $\hat{\mathbf{M}}$, is stored in memory for later use in computing $\mathbf{u}'\mathbf{M}\mathbf{u}$ and $\mathbf{u}'\mathbf{M}\mathbf{Q}$. Next, we list the steps of the basic algorithm.

4.4. Algorithm

- *Step 1*: Compute the superset matrix $\bar{\mathbf{M}}$ and its Fourier transform.
- *Step 2*: Compute $\hat{\mathbf{M}}\langle i_0, j_0\rangle = \mathbf{Q}'\mathbf{M}\langle i_0, j_0\rangle\mathbf{Q}$ and its inverse for all locations of the candidate solution.
- *Step 3*: Using the given dense optical flow field, compute $\mathbf{u}'\mathbf{M}\langle i_0, j_0\rangle\mathbf{u}$ and $\mathbf{u}'\mathbf{M}\langle i_0, j_0\rangle\mathbf{Q}$.
- *Step 4*: Form the products and compute the error for each candidate hypothesis of the FOE.
- *Step 5*: Pick the location of the FOE corresponding to the smallest squared error.
- *Step 6*: Repeat from step 3 for the next data set.

The basic building block of the algorithm is a function that performs the Fast Computation $a'\mathbf{M}\langle i_0, j_0\rangle b$ for the arguments a and b. Since this is central to the approach, we have provided a detailed algorithm for this function, which we term *FastCompute*, in Appendix A.2. *FastCompute* requires 3 one-time FFTs and 4 FFTs for each evaluation involving vectors a and b. The 3 one-time FFTs are computed in step 1 of 4.4. Steps 2 and 3 require 24 and 16 FFT computations respectively, making up a total of 43 FFTs for the entire process.

5. Extensions

This section discusses a few extensions of the basic technique for FOE estimation, with the emphasis being on relaxing the assumptions introduced in Section 3.1.

5.1. Dealing with Sparsity

How does the proposed solution change if the velocity estimate u, v is not available for a particular point (i, j)? In such a situation the corresponding equation pair (5) is also not available. If, with no increase in computational complexity, we can replace (5) with a

set of equations that retains the coefficients of $h_{i,j}$ and yet does not influence the solution in any manner, we can extend our technique to both sparse flow fields as well as non-2^k image sizes.

Consider the equations

$$\begin{aligned} 0 &= (x_{i,j} - x_f)h_{i,j} + 0\omega_x + 0\omega_y + 0\omega_z \\ 0 &= (y_{i,j} - y_f)h_{i,j} + 0\omega_x + 0\omega_y + 0\omega_z \end{aligned} \tag{47}$$

The consistent solution of these equations is $h_{i,j} = 0$, with ω being indeterminate. Since the coefficients of ω are zero, appending this set of equations to our system does not influence the solution. Obviously, the depth at point $i,\ j$ cannot be estimated. Replacing (5) by (47) for every pixel at which the optical flow is not known preserves the structure of $\mathbf{P}(x_f, y_f)$, and the corresponding rows of $\mathbf{Q}$ must now be set to zero. With this substitution, the reasoning in Section 4 holds and the FOE can still be estimated in $\mathcal{O}(N^2 \log N)$ steps. It must be noted that Step 2 in Algorithm 4.4 must also be repeated for each frame.

The above substitution allows us to zero-pad a flow field if needed to make its size a power of 2. Moreover, since most optical flow techniques produce sparse flow fields, this substitution allows us to interface with these methods. There is no need to interpolate sparse flow fields—an operation which may lead to reduced accuracy. Yet another situation calling for zero-padding is where the search area is larger than the image, even when the latter is a power of 2 in size.

5.2. *Confidence Incorporation*

The reliability with which optical flow is determined across an image is dependent on several factors including local gradient information, and varies across the image area. Often, a space-varying confidence measure characterizing the reliability of optical flow can be computed from image data. This section deals with incorporating this information into the error surface computation.

Let $k_{i,j} \in \Re$ be the Bayesian confidence of optical flow at (i, j), i.e. $k_{i,j}$ is proportional to the standard deviation of the optical flow distribution at (i, j). We assume the x and y components of the computed flow are Gaussian with the same variance, and are independent. The likelihood function that must be minimized is given by

$$\begin{aligned} \sum_{i,j} k_{i,j}^2 \big[&\big(u_{i,j} - (x_{i,j} - x_f)h_{i,j} - x_{i,j}y_{i,j}\omega_x \\ &+ \big(1 + x_{i,j}^2\big)\omega_y - y_{i,j}\omega_z\big)^2 + \big(v_{i,j} - (y_{i,j} - y_f)h_{i,j} \\ &- \big(1 + y_{i,j}^2\big)\omega_x + x_{i,j}y_{i,j}\omega_y + x_{i,j}\omega_z\big)^2 \big] \end{aligned}$$

The confidence values can be incorporated into the matrix-vector formulation (10). This is easily achieved by multiplying each equation pair by its corresponding confidence value. Form the vector $\hat{\mathbf{u}}$ given by the componentwise product of $\mathbf{u}$ with $k_{i,j}$, as

$$\hat{\mathbf{u}} = (u_{0,0}k_{0,0}\ v_{0,0}k_{0,0} \ldots v_{N-1,N-1}k_{N-1,N-1})$$

and $\hat{\mathbf{Q}}$ likewise

$$\hat{\mathbf{Q}} = \begin{bmatrix} r'_{0,0}k_{0,0} \\ s'_{0,0}k_{0,0} \\ \vdots \\ s'_{N-1,N-1}\,k_{N-1,N-1} \end{bmatrix}$$

Then, the modified squared error cost function of the form (11) is given by

$$C(x_h, y_h, \mathbf{x}) = \|\hat{\mathbf{A}}(x_h, y_h)\mathbf{x} - \hat{\mathbf{u}}\|_2^2$$

where $\hat{\mathbf{A}} = [\mathbf{P}\hat{\mathbf{Q}}]$. Note that the effect of multiplying each equation by the confidence measure is split between $\hat{\mathbf{Q}}$ and the unknown $\hat{h} = (h_{0,0}k_{0,0} \ldots h_{N-1,N-1}k_{N-1,N-1})$, without affecting the structure of $\mathbf{P}$. Thus, the remainder of the reasoning follows, and confidence measures can be built into the estimation of FOE with no detriment to the overall performance.

Now, it is easy to see that (47) is a special case of weighting (5) by zero. In other words, points outside the image, or interior points at which the flow is unknown can be treated as points with zero confidence value. This extension may be construed to be weak, due to the inherent assumption that the x and y components are independently Gaussian, and have the same variance. In practice, the aperture effect leads to non-isotropic distributions of the flow field. Since points suffering from the aperture effect can contribute only one reliable equation, they might as well be dropped by setting the confidence to zero, with no significant loss of information. Therefore, although the assumption may break down, this extension is still applicable.

5.3. *Offset of Search Window*

If it is known by some means that the FOE is present in an area around $x_{\text{off}}, y_{\text{off}}$, our method can be made more effective by incorporating this knowledge into the formulation. Assume that the offset in the pixel coordinate system is $i_{\text{off}}, j_{\text{off}}$ corresponding to $x_{\text{off}}, y_{\text{off}}$. The superset matrix $\hat{\mathbf{M}}$ is redefined as

$$\bar{\mathbf{M}}_{\mathbf{i},\mathbf{j}} = \frac{\begin{bmatrix} \left(j - j_{\text{off}} - N + \frac{1}{2}\right)^2 & -\left(i - i_{\text{off}} - N + \frac{1}{2}\right)\left(j - j_{\text{off}} - N + \frac{1}{2}\right) \\ -\left(i - i_{\text{off}} - N + \frac{1}{2}\right)\left(j - j_{\text{off}} - N + \frac{1}{2}\right) & \left(i - i_{\text{off}} - N + \frac{1}{2}\right)^2 \end{bmatrix}}{\left(i - i_{\text{off}} - N + \frac{1}{2}\right)^2 + \left(j - j_{\text{off}} - N + \frac{1}{2}\right)^2} \tag{48}$$

$\forall i, j \in \{0, 1, \ldots, 2N-1\}$. This definition offsets the search window by $i_{\text{off}}, j_{\text{off}}$. The remainder of the computation process remains the same. However, the offset must be added to $\arg\min_{x_h, y_h} \|\mathbf{A}(x_h, y_h)\mathbf{x} - \mathbf{u}\|^2$ when computing the FOE.

Shifting the search area by an offset allows the fusion of external information into our algorithm. For example, while processing a sequence of images, the search window can be re-centered at the FOE estimate of the previous frame. If it is assumed that acceleration is small from frame to frame, such re-centering improves the probability of finding the true FOE within the search area. Alternatively, if a kinematic model exists for the camera platform, the velocities can be predicted from the past behavior. This provides a useful starting guess for the FOE.

Finally, we claim without proof that the overall complexity of estimating the FOE hypothesized to lie in a $M_x \times M_y$ area (potentially offset from the center), given a flow field of size $N_x \times N_y$ (potentially sparse), is $\mathcal{O}(M_2 N_2 \log M_2 N_2)$, where $M_2 = 2^{\lceil \log_2(M_x + N_x) \rceil}$ and $N_2 = 2^{\lceil \log_2(M_y + N_y) \rceil}$. Although this is within the same order of magnitude for images whose aspect ratio is bounded, this expression is derived by tightening the Fast Computability proofs.

6. Experiments and Results

In this section, we describe our experiments on evaluating the partial search FOE estimation algorithm. Our experiments comprise two phases, viz. a first phase which measures quantitative performance on synthetic data generated with known parameters, followed by a second phase which examines qualitative performance on real-world imagery with emphasis on useful applications. Using synthetic data allows us to accurately characterize performance over a range of situations. In the first phase, the optical flow field is synthesized. On real data, the flow is obtained from a sequence of images using matching. The synthetic data experiments demonstrate the correctness, operating range and robustness of our algorithm. The utility of the algorithm in solving real-world problems is shown using real data.

During the course of the experiments, the proposed algorithm is applied to the flow field and the location corresponding to the minimum error in (26) is picked as the FOE $(\hat{x}, \hat{y})$ (or equivalently, $(\hat{i}, \hat{j})$ in the pixel coordinate system.) A correction of $\frac{1}{2}$ pixel is applied to each direction, to undo the effect of staggering (43). Once the FOE is estimated, the angular velocity and depth map can be obtained by solving linear equations. From (23), we can see that the LS estimate for ω is

$$\hat{\omega} = \hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M}\mathbf{u}$$

which is a product of terms that have already been calculated. Therefore, the estimate of rotation is available with no extra computation.

The depth map takes some additional effort to recover. Using (23), we get

$$h = \mathbf{D}^{-1}\mathbf{P}'(\mathbf{I} - \mathbf{Q}\hat{\mathbf{M}}^{-1}\mathbf{Q}'\mathbf{M})\mathbf{u}$$

which requires the computation of $\mathbf{D}^{-1}\mathbf{P}'\mathbf{Q}$ and $\mathbf{D}^{-1}\mathbf{P}'\mathbf{u}$. These take $\mathcal{O}(N^2)$ operations.

6.1. *Experiments with Synthetic Data*

The least squared error as a function of the hypothesized FOE is denoted by E, and is referred to as the *error surface*. The minimum value of the error surface, which occurs at $(\hat{i}, \hat{j})$, is denoted by $E_{\min}$.

Incorporating the round-off error in estimating the FOE, the error in the FOE estimate, ϵ, is given by

$$\epsilon = \min\left\{ \left\| \begin{matrix} \hat{i} - i_0 \\ \hat{j} - j_0 \end{matrix} \right\|, \left\| \begin{matrix} \hat{i} - i_0 \\ \hat{j} - j_1 \end{matrix} \right\|, \left\| \begin{matrix} \hat{i} - i_1 \\ \hat{j} - j_0 \end{matrix} \right\|, \left\| \begin{matrix} \hat{i} - i_1 \\ \hat{j} - j_1 \end{matrix} \right\| \right\}$$

where $\{i, j\}_0 = \lfloor \{i, j\}_f - \frac{1}{2} \rfloor + \frac{1}{2}$ and $\{i, j\}_1 = \lceil \{i, j\}_f - \frac{1}{2} \rceil + \frac{1}{2}$. The $\frac{1}{2}$ terms compensate for staggering of the hypothesis grid. This expression provides a 4-pixel neighborhood associated with zero error, if the true FOE lies off the $\frac{1}{2}$-pixel offset grid lines along each axis. When the true FOE is exactly on a grid point, the neighborhood shrinks to one pixel.

In many favorable situations, our algorithm estimates the FOE with no error. We use a secondary error metric, ϵ_ω, which is the total angular error, given by

$$\epsilon_\omega = \|\hat{\omega} - \omega\|$$

to characterize and differentiate between cases where ϵ does not provide sufficient characterization.

Given a good estimate of the FOE, the depth map is essentially recovered by solving a set of linear equations. This process is neither novel nor particularly interesting from a research perspective, and can be expected to have predictable results. For instance, poor values of the optical flow severely impair the relevant depth map estimate. The depth map is very sensitive to perturbations in the optical flow as well as the FOE estimate, especially near the latter. Later, we examine the recovered depth maps for the case of noisy and dense optical flow fields.

6.1.1. Generation of Synthetic Data. Using (5), we generate optical flow fields corresponding to chosen inverse depth maps, FOEs and rotational velocities. The depth map, in the real world, is comprised of largely smooth regions bounded by sharp discontinuities. The size of these regions varies widely. In order to closely model the real world, the depth map is generated using a fractal model. This is motivated by the use of fractal models for luminance images which, like depth maps, are also composed of largely smooth regions and discontinuities. The Fourier transform method is used to generate the depth map fractal. In this method, the transform magnitude is chosen to be exponential, of the form $(f_x^2 + f_y^2)^{-d/2}$. The phase is random, uniformly distributed in $[0, 2\pi]$. We call d the *fractal exponent*. Upon inverse Fourier transformation, the resulting images (both real and imaginary components of the transform) are fractals. Choosing smaller fractal exponents gives more intricate fractals. The fractals so generated are used as synthetic inverse depth maps.

Table 1. Parameters used in generating synthetic data sets *A* and *B*.

Set	N	f	i_f	j_f	ω_x	ω_y	ω_z	d
A	256	400	102.0	51.0	−2.0	5.0	−8.0	1.5
B	256	400	127.5	201.5	5.0	3.0	4.0	1.7

For our experiments, we chose $N = 256$. We generated two baseline data sets *A* and *B* with parameters shown in Table 1. The angular velocities are in m rad, and the focal length f is in pixel units. The FOE is shown in the pixel coordinate system, with the origin located at the top left corner of the image frame. The i axis is down the rows and j axis across the columns. The FOE in *Set B* is located at a valid grid point while the FOE in *Set A* is located exactly midway between four grid points. Besides, *Set A* displays a relatively large rotation along the optical axis. On the other hand, the dominant rotation, for *Set B*, is along the image plane axes. In addition, the FOE coordinates are well distributed in $[0, N]$. The focal length of 400 pixels reflects a normal field of view of 35 degrees.

Figure 1(a)–(d) depicts *Set A*: the underlying fractal depth map is shown as a grayscale image in (a), the rotational component of the flow field in (b), the translational component alone in (c), and the overall synthesized flow in (d). Close areas are bright while areas at infinity are black in Fig. 1(a). The confounding of rotational and translational flow is clear from (b), (c) and (d). In particular, there are areas in Fig. 1(d) where the flow field alternates direction. Such inflections occur at points where neither the rotation nor translation is dominant. The underlying inverse depth map and total flow field of *Set B* is shown in Fig. 1(e) and (f) respectively.

6.1.2. Sensitivity Analysis on Synthetic Data. Next, we look at the performance of our algorithm on the synthetic data sets. Sets *A* and *B* serve as the benchmark. For analyzing the sensitivity of our algorithm to a particular parameter p, we generate an ensemble of flow fields using different values of p, keeping the remaining parameters fixed to those of *A* and *B*. To begin with, we assume that the focal length is known accurately. In Section 6.1.8, we examine the effect of errors in the focal length estimate.

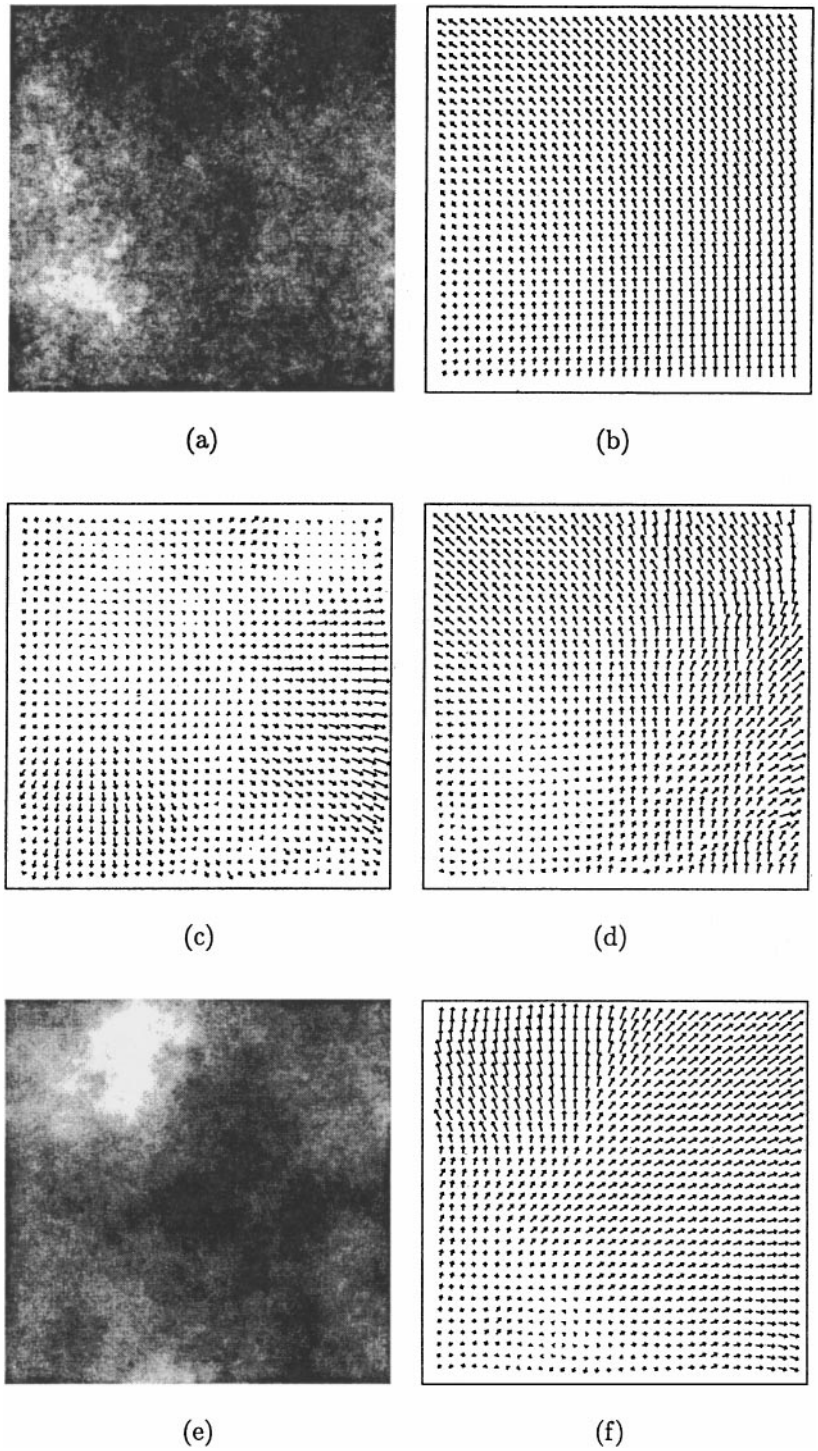

Figure 1. Set *A*: (a) shows the underlying fractal inverse depth map, (b) the rotational flow, (c) the translational flow and (d) the total optical flow field. (e) and (f) are the underlying fractal inverse depth map and total optical flow field generated from set *B* respectively.

Table 2. Performance on dense, noiseless flow fields.

Set	$\hat{i}$	$\hat{j}$	ϵ	$\hat{\omega}_x$	$\hat{\omega}_y$	$\hat{\omega}_z$	ϵ_ω
A	101.5	51.5	0.00	−1.985	5.013	−7.995	0.020
B	127.5	201.5	0.00	5.000	3.000	4.000	0.000

6.1.3. Ideal Case. With a perfect flow field and accurate focal length estimate, our algorithm locates the FOE without error. The rotational error ϵ_ω is zero for *Set B* which lies on a valid grid point, but non-zero, though small, for *Set A*, which lies between grid points. These results are shown in Table 2. The rotational velocities and error ϵ_ω are in 10^{-3} rad. Contours of the error surface E are plotted in Fig. 2(a) and (b) for sets A and B respectively. The true location of the FOE is marked by a $+$ and the estimated location by a $\times$.

6.1.4. Performance with Sparse/Noisy Flows. In the real world, optical flow is seldom determined at all points in an image with certainty. Besides, the flow estimates are typically noisy. The sparsity of flow depends on the specific optical flow algorithm chosen, the presence of local high-frequency information, and the existence of a coherent motion across the image. We simulate a sparse flow field by randomly including or discarding the flow at a given pixel, according to an *i.i.d.* binary distribution. We realize noise in the flow field by adding an *i.i.d.* zero-mean Gaussian process with variance σ to each component of velocity. The noise level η is measured according to the angular error metric employed in Barron et al. (1994), given by

Table 3. FOE estimation error ϵ as a function of the noise and sparsity of the input flow field.

Noise		Sparsity				
σ	η (approx.)	100%	80%	60%	40%	20%
0.0	0.00	0.00	0.00	0.00	0.00	0.00
0.1	2.65	0.00	0.00	0.00	0.00	0.00
0.2	5.29	0.00	0.00	0.00	0.00	0.50
0.4	10.51	1.00	1.00	1.21	1.12	0.50
1.0	25.08	5.41	9.80	3.55	6.77	6.62

$$\eta = E\left[\cos^{-1}\left(\frac{\mathbf{v}_0 \cdot \mathbf{v}}{\|\mathbf{v}_0\|\|\mathbf{v}\|}\right)\right]$$

where $\mathbf{v}_0 = (u\ v\ 1)'$, $\mathbf{v} = \mathbf{v}_0 + (\eta_u\ \eta_v\ 0)'$ and $\eta_u, \eta_v \sim \mathcal{N}(0, \sigma)$. η, measured here in degrees, is insensitive to the magnitude of the motion vector and offers a normalized measure against which a range of velocities can be compared meaningfully.

We simulated flow fields corresponding to five combinations of sparsity, viz. 100%, 80%, 60%, 40% and 20%, and five combinations of noise level, $\sigma = 0, 0.1, 0.2, 0.4$ and 1.0. The angular error η corresponding to σ is approximately the same for a given σ,

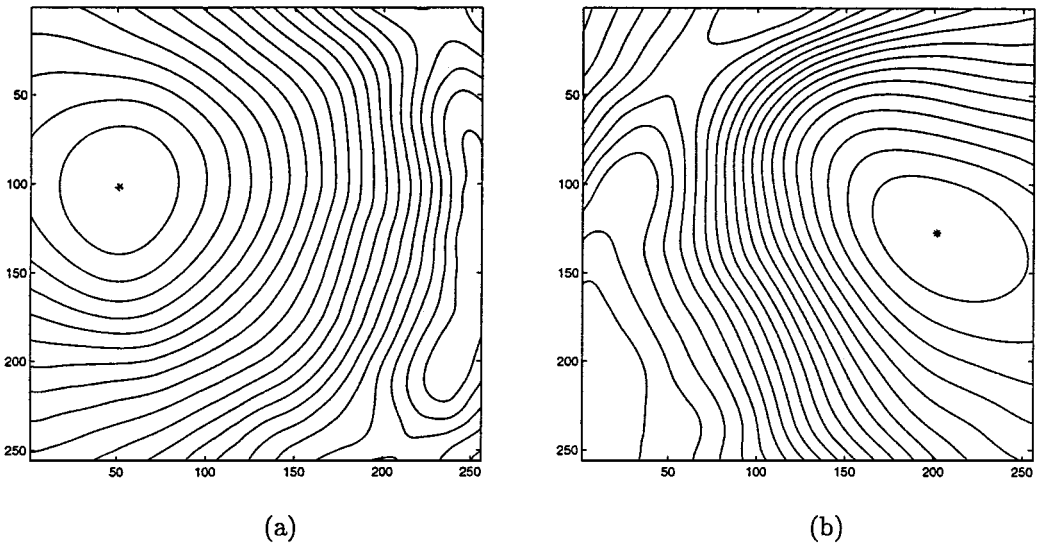

Figure 2. Ideal case performance: (a) and (b) show the error surface contours for sets A and B respectively. The true FOE is marked by a $+$ and the estimated FOE by a $\times$.

over sets A and B and over all sparsity levels. The error ϵ in estimating the FOE using our algorithm, averaged over sets A and B, is tabulated as a function of noise and sparsity in Table 3. Figure 3 represents the *worst case* scenario. The flow fields generated with $\sigma = 1.0$ are shown in Fig. 3, (a) *Set A* with 80% density and (b) *Set B* with 20% density. Figure 3(c) and (d) plot the error surface contours with the true FOE (+) and estimate (×) corresponding to (a) and (b) respectively, showing good compliance of our solution.

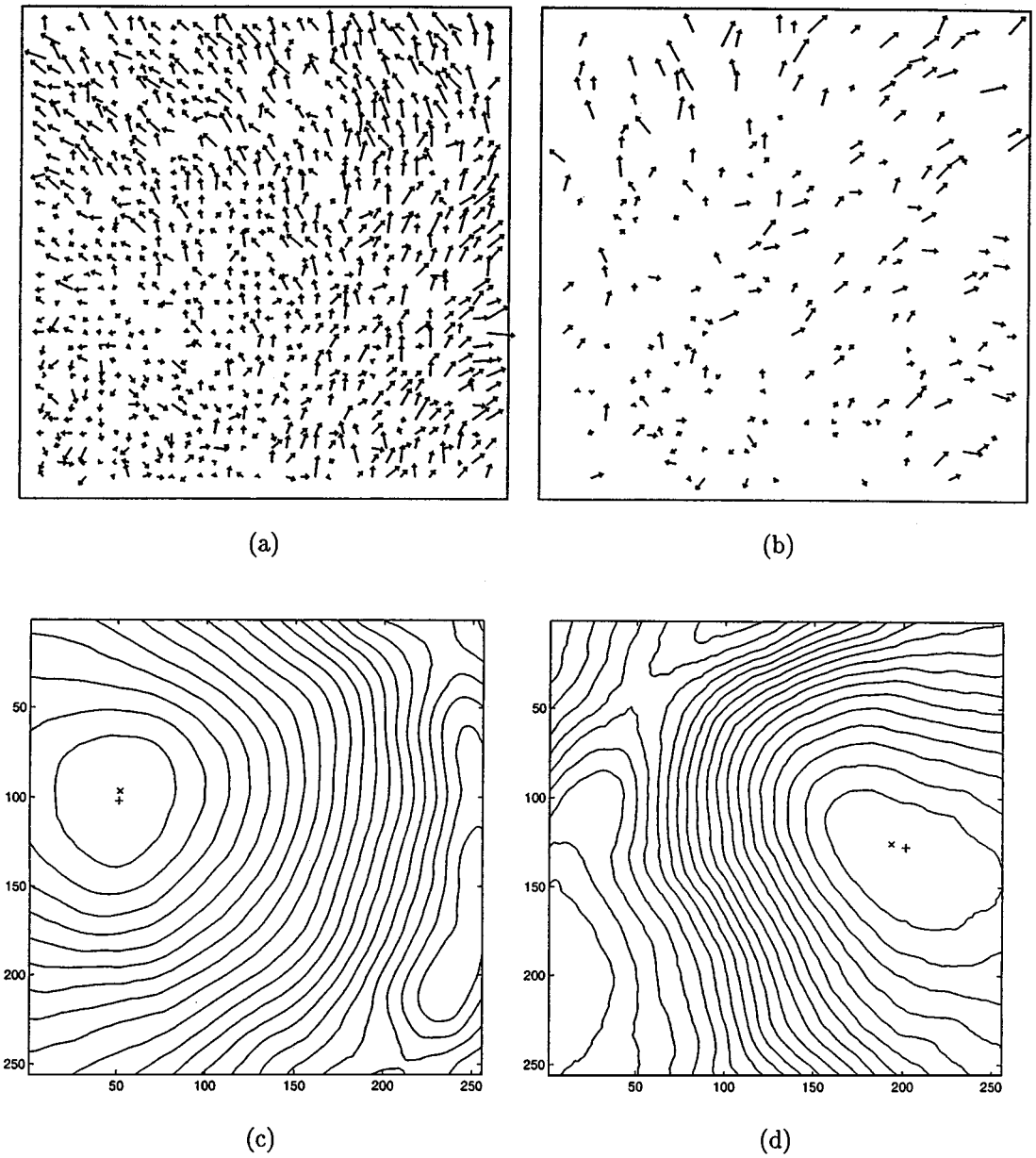

Figure 3. Worst-case performance with a noisy, sparse flow field: (a) *Set A* at 80% density, (b) *Set B* at 20% density, (c) and (d) error surface contours with true FOE (+) and estimated FOE (×) corresponding to (a) and (b) respectively. Zero-mean Gaussian noise with $\sigma = 1.0$ has been added to (a) and (b).

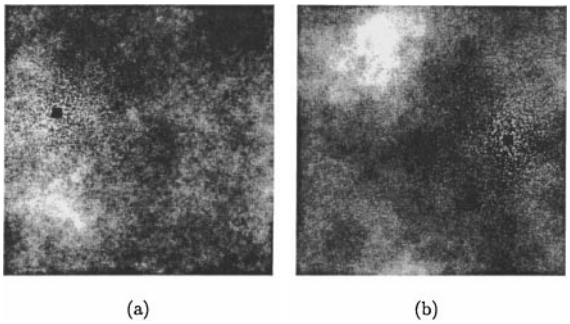

Figure 4. Depth map reconstruction: (a) and (b) are reconstructed scaled inverse depth maps from dense flows with $\sigma = 0.1$.

6.1.5. Depth Recovery. In the ideal case, when the input flow field is dense and error-free, the FOE is recovered with no error. The recovered depth map is also nearly perfect, except at points very close to the FOE. This is due to the influence of discretization on the coefficients of the inverse depth map in (2). When the input flow field is noisy, we can expect the recovered depth map to show significantly larger errors as compared to the FOE or rotation estimate. This is because the number of independent equations contributing to each depth map value is only 2, whereas all $\mathcal{O}(N^2)$ equations contribute to the unknown rotations in (5).

Figure 4 provides an insight into the depth recovery step. Dense flow fields corresponding to sets A and B, with additive noise of $\sigma = 1$ are generated. The corresponding angular error is approximately 2.65 degrees, which is a reasonable performance metric for a good motion estimation algorithm. The SFM technique is applied and scaled inverse depth recovered. The recovered depth map of set A is shown in Fig. 4(a), and that of set B in (b). A small region of 5×5 pixels around the estimated FOE is blanked out since the estimates in this area are rather meaningless. A visual comparison demonstrates excellent match with the original depth maps (Fig. 1(a) and (e)).

6.1.6. Effect of Depth Structure. Early 3D SFM techniques that assume a smooth flow field tend to fail when there are discontinuities introduced by busy depth maps. Likewise, the newer techniques that exploit depth non-negativity minimally require busy depth maps to work at all. One of the merits of our approach is its applicability to both of these extremes and to intermediate cases. We validate this claim by studying performance on flow fields simulated using several depth maps generated by a range of fractal exponents d. Moreover, we also consider a planar depth map for comparison, given by $h_{\text{planar}}(i, j) = C_0 + C_x i + C_y j$. An imaged 3D planar scene gives rise to this form of depth map.

Retaining the FOE, rotation and focal lengths of sets A and B, we generated flow fields for the planar and fractal depth maps with $d = 1.7$, 1.5, 1.3 and 1.1. The high-frequency content of the depth map increases with decreasing d. The results of FOE estimation using our algorithm are shown in Table 4. For both sets A and B, the FOE is estimated with zero error ϵ. The angular error ϵ_ω, indicated in 10^{-3} rad, is very small for A and zero for B. This is not surprising, considering that the true FOE for A lies between grid points. Figure 5(a)

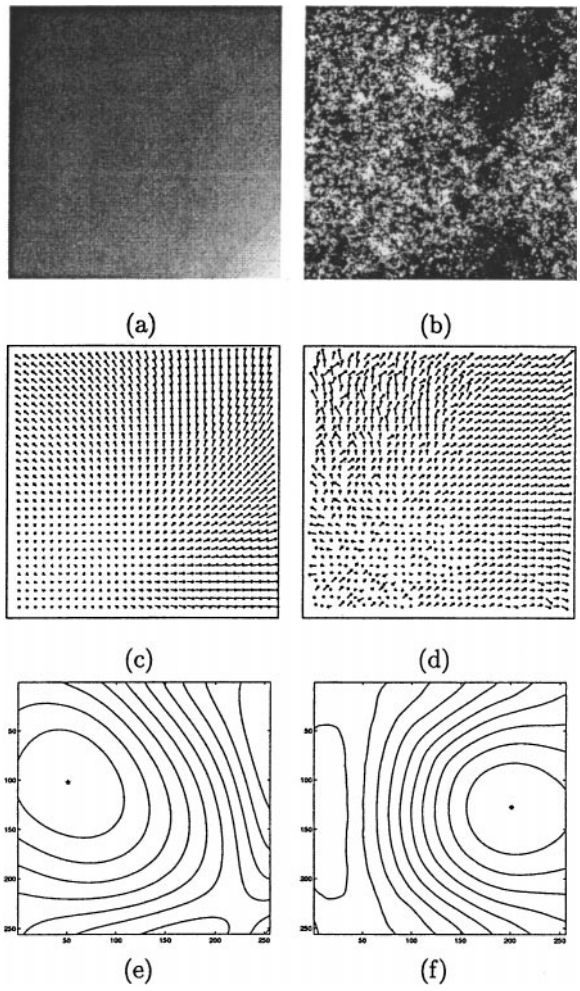

Figure 5. Performance vs. depth map structure: Two extreme depth maps are shown here. (a) shows the *planar* depth map and (b) is a fractal depth map with exponent 1.1. (c) and (d) show flow fields generated with parameter sets A and B and depth maps (a) and (b) respectively. Corresponding error contours, with true and estimated FOEs, are plotted in (e) and (f).

Table 4. Performance as a function of depth structure quantified by the fractal exponent.

d	$\hat{i}$	$\hat{j}$	ϵ	$\hat{\omega}_x$	$\hat{\omega}_y$	$\hat{\omega}_z$	ϵ_ω
Set *A*							
Planar	102.5	51.5	0.0	−1.989	4.985	−8.002	0.019
1.7	101.5	50.5	0.0	−2.014	5.014	−8.009	0.022
1.5	101.5	50.5	0.0	−1.985	5.013	−7.995	0.020
1.3	101.5	50.5	0.0	−2.012	5.015	−7.992	0.021
1.1	101.5	50.5	0.0	−2.009	5.014	−7.999	0.017
Set *B*							
All	127.5	201.5	0.0	5.000	3.000	4.000	0.000

and (b) show the extreme-case depth maps which are planar and fractal respectively, the latter with $d = 1.1$. The corresponding flows generated using sets A and B are plotted in (c) and (d). Finally, the error surface contours, together with true (+) and hypothesized (×) FOE, are shown in Fig. 5(e) and (f) respectively.

6.1.7. Performance vs. Field of View. Until now, we have restricted ourselves to flow fields generated by a camera with a "normal" field of view (FOV), which is typically between 30 and 45 degrees. The nature of the flow field varies considerably as the focal length, or equivalently, the FOV, changes. A robust solution to the 3D SFM problem must operate across a range of fields of view. With noiseless data, our algorithm locates the FOE without error for f between 200 and 800 pixels, corresponding to FOVs between 18 and 65 degrees. To facilitate a better understanding we examine the performance with noise in the flow field. We generated flow fields for sets A and B, with noise level η set to 10.0. This is achieved by choosing appropriate values of σ.

Table 5 provides the results of this experiment. The FOE is estimated with reasonable accuracy, and the rotational error (tabulated in 10^{-3} rad) is very small for all cases. As the focal length increases, so does the σ needed to achieve $\eta = 10.0$. As a consequence, ϵ also shows an upward trend. Figure 6 shows the extreme cases of our experiment. The synthesized noisy optical flow fields corresponding to $f = 200, 800$ and 800 pixels, using parameter sets A, A, and B, are shown in Fig. 6(a), (c) and (e) respectively. The error contours together with true (+) and estimated (×) FOE are shown in Fig. 6(b), (d) and (f). Despite obvious large perturbations in the flow, our algorithm performs well in locating the FOE.

6.1.8. Effect of Mis-Estimated Focal Length. In the above discussion, we have assumed that the focal length of the camera is known accurately. This is realistic in the real world as the physical parameters of the camera are either specified or can be measured using camera calibration algorithms. Nevertheless, characterizing the sensitivity of our algorithm to mis-estimated focal length is useful since this sensitivity determines the deviation of our FOE estimates when the focal length itself is known (or estimated) imprecisely. Furthermore, in situations where the focal length is altogether unknown, this study reveals what parameters, at a minimum, can be computed with some degree of reliability using an arbitrarily chosen focal length.

Our final experiment on synthetic data sets involves estimating the FOE from noiseless, dense flow fields using parameter sets A and B. Unlike the previous experiments, the focal length assumed in the

Table 5. Performance as a function of field of view.

Field of view		Noise level		Estimates						
f	θ (degree)	η	σ	$\hat{i}$	$\hat{j}$	ϵ	$\hat{\omega}_x$	$\hat{\omega}_y$	$\hat{\omega}_z$	$\epsilon\omega$
Set A										
200	65.2	10.0	0.408	101.5	51.5	0.00	−1.935	5.060	−7.959	0.097
280	49.1	10.0	0.368	101.5	50.5	0.00	−2.018	5.009	−8.011	0.023
560	25.8	10.0	0.512	101.5	51.5	0.00	−1.985	5.007	−8.018	0.024
800	18.2	10.0	0.820	94.5	49.5	7.07	−2.003	5.055	−7.921	0.096
Set B										
200	65.2	10.0	0.372	127.5	201.5	0.00	5.010	3.002	3.996	0.011
280	49.1	10.0	0.345	127.5	201.5	0.00	5.007	3.011	3.968	0.035
560	25.8	10.0	0.580	125.5	199.5	2.83	4.990	3.016	4.026	0.032
800	18.2	10.0	0.910	130.5	205.5	5.00	5.022	2.984	3.987	0.030

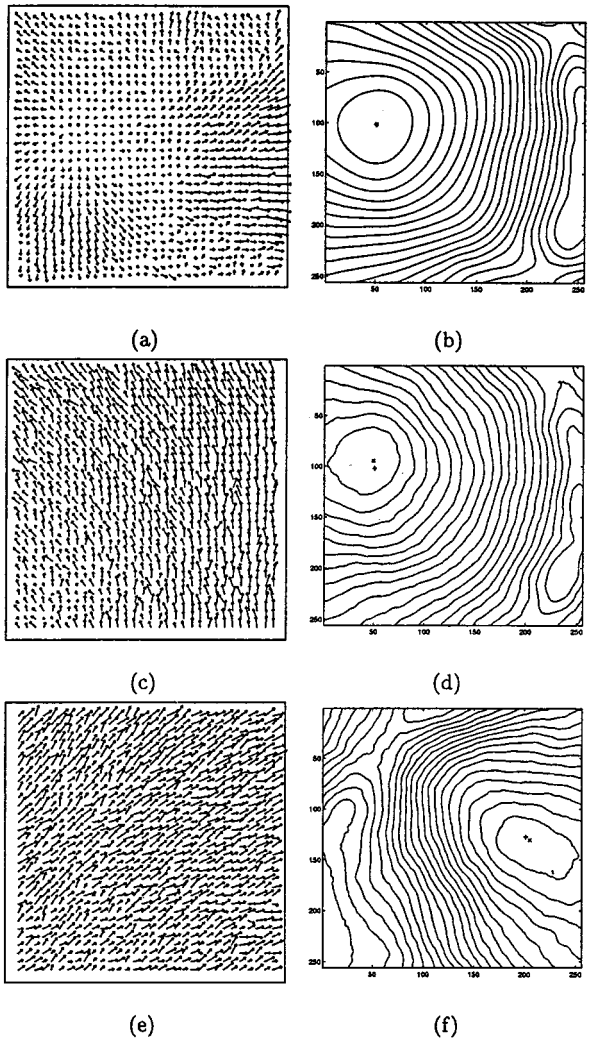

Figure 6. Performance vs. focal length: Synthesized noisy flow fields and error surface contours, generated with: (a) and (b) parameter set A, $f = 200$, (c) and (d) A, $f = 800$, and (e) and (f) B, $f = 800$. Noise $\eta = 10.0$ for all cases. The true location is marked by a $+$ and the estimated location by a $\times$.

computations is made to vary over the range 200–800 pixels in $2^{0.25}$ factor multiples, while the true focal length is fixed at 400 pixels. Table 6 provides a summary of the results of this experiment. Excluding the extreme wide angle case (200 pixels), the FOE estimate is very good. In the extreme case, the FOE is displaced by 15 and 11 pixels for A and B respectively. These results lead us to claim that the algorithm is relatively insensitive to the focal length, insofar as the FOE estimate is concerned. For the angular velocities, this does not hold. $\hat{\omega}_x$ and $\hat{\omega}_y$ are approximately scaled by the ratio of the true focal length to the chosen focal length. However the rotation along the optical axis is relatively robust to the choice of focal length.

Our experiments using synthetic flow fields illustrate the strengths and limitations of our algorithm, and provide useful insight regarding its domain of applicability. In particular, we have shown that our approach is robust to all the commonly encountered issues in flow field analysis, primary amongst which are noise and sparsity. It remains to be seen how our algorithm can be applied to real-world problems; this is our focus ahead. First, we discuss timing issues, and follow up with a real-world application.

6.2. Timings

The principal thrust of this paper is towards a tractable and robust solution to the structure from motion problem. Section 4 proves that partial search for the FOE has a complexity $\mathcal{O}(N^2 \log N)$, where N is the linear dimension of the flow field. The robustness issues have been demonstrated experimentally earlier in this section. In order to provide a better feel for the tractability of this approach, we consider the execution time of the core computation involving computing the error surface through 43 FFTs and picking the minimum. The observed timings are summarized in Table 7. The three standard workstation architectures compared here are the following:

- *Arch I*: DEC Alpha workstation, clocked at 275 MHz.
- *Arch II*: Pentium Pro workstation, clocked at 200 MHz.
- *Arch III*: Pentium II workstation, clocked at 450 MHz.

The FOE estimation algorithm was implemented in C++, with full floating point operations. Only compiler optimization was employed to speed up the implementation on Arch I. For Archs II and III, we used the Intel image processing library (ipl) routines to speed up the FFT computations alone.

It can be seen that it is possible to attain sub-second execution times for reasonably large flow fields. This makes it conceivable to use the partial search algorithm for real-time 3D motion analysis. The algorithm formulates the solution as a series of FFT computations. Dramatically increased operating speeds are possible using optical correlators for performing these steps. We believe that this method is a realistic approach to solving

Table 6. Performance as a function of incorrectly estimated focal length.

	Set *A*					Set *B*				
Focal length[a]	ϵ	$\hat{\omega}_x$ (-2.00)	$\hat{\omega}_y$ (5.00)	$\hat{\omega}_z$ (-8.00)	$\epsilon\omega$	ϵ	$\hat{\omega}_x$ (5.00)	$\hat{\omega}_y$ (3.00)	$\hat{\omega}_z$ (4.00)	$\epsilon\omega$
200	11.05	-3.16	9.80	-7.15	4.98	14.87	8.97	5.75	2.98	4.94
238	5.10	-2.97	8.28	-7.40	3.48	8.94	7.85	4.88	3.33	3.48
283	2.00	-2.66	7.03	-7.63	2.16	5.00	6.81	4.14	3.61	2.17
336	0.00	-2.34	5.93	-7.84	1.00	2.24	5.85	3.54	3.83	1.02
400	0.00	-1.99	5.01	-8.00	0.02	0.00	5.00	3.00	4.00	0.00
476	0.00	-1.71	4.22	-8.11	0.85	1.41	4.25	2.54	4.12	0.89
566	0.00	-1.44	3.55	-8.20	1.66	2.24	3.59	2.15	4.20	1.66
673	0.00	-1.22	2.99	-8.27	2.18	3.16	3.03	1.82	4.26	2.31
800	0.00	-1.03	2.50	-8.34	2.70	3.16	2.55	1.53	4.29	2.87

[a]True focal length: 400.

3D structure from motion. We have provided software[2] to support these claims, and for further research.

6.3. *Experiments on Real Data*

Although our technique for FOE estimation works well in theory and in simulations, applying it to solving real world problems presents a whole new set of challenges. In the data processing chain, our algorithm uses a pre-computed optical flow field and provides as output the FOE, rotational velocity and depth map (up to a scale factor). Thus, its performance is to some extent circumscribed by the accuracy of the flow field estimation pre-processing stage. While the FOE and rotational velocity estimates are reasonably robust to errors in the flow field, the same is not true of the depth map, as seen in the relevant experimental results.

We have applied the proposed SFM technique for solving several problems on real-world data, including 3D stabilization, moving object detection from a moving platform, rangefinding, obstacle detection and 3D model building (Srinivasan, 1998). Predictably, performance of our technique has been mixed, with applications requiring accurate rotations showing excellent results, and applications calling for dense depth map estimates being poor. In defence of our technique, we wish to note that we did not concentrate our efforts on improving the post-processing stages that are critical for applications like 3D model building.

Table 7. FOE estimation times on three common architectures.

Flow field size	Arch I	Arch II	Arch III
64×64	1.5 s	400 ms	180 ms
128×128	7.7 s	1.9 s	910 ms
256×256	40 s	8.5 s	4.1 s

In the remainder of this section, we will consider an application that does require the scene structure to be computed, but does not require heavy post-processing. This is the detection of obstacles from an image sequence gathered by a (mostly forward) moving camera. We use a variant of block matching (Srinivasan, 1998) to compute a reasonably dense flow field, though sub-pixel flow estimates are not very good. Algorithmically, this is an $\mathcal{O}(N^2)$ process. We perform SFM to generate a scaled inverse depth map. Next, we fit a ground plane to the computed inverse depth map. A plane in 3D shows up as a planar function relating the inverse depth to the image coordinate. Let the ground plane be given by $AX + BY + CZ = 1$. In the image coordinate system,

$$\begin{aligned} At_z\frac{X}{Z} + Bt_z\frac{Y}{Z} + Ct_z &= t_z\frac{1}{Z} \\ \Rightarrow t_z(Ax + By + Cf) &= h(x, y) \\ \Rightarrow ax + by + c &= h(x, y), \end{aligned} \tag{49}$$

which is a planar function for the (scaled) inverse depth $h(x, y)$. We fit a plane to the valid values of the computed inverse depth, using the least squares error criterion. While fitting the ground plane, we consider only the lower two-thirds of the image area, assuming, as a rule of thumb, that the top portion of the image looks above the horizon. We then look for deviations of the

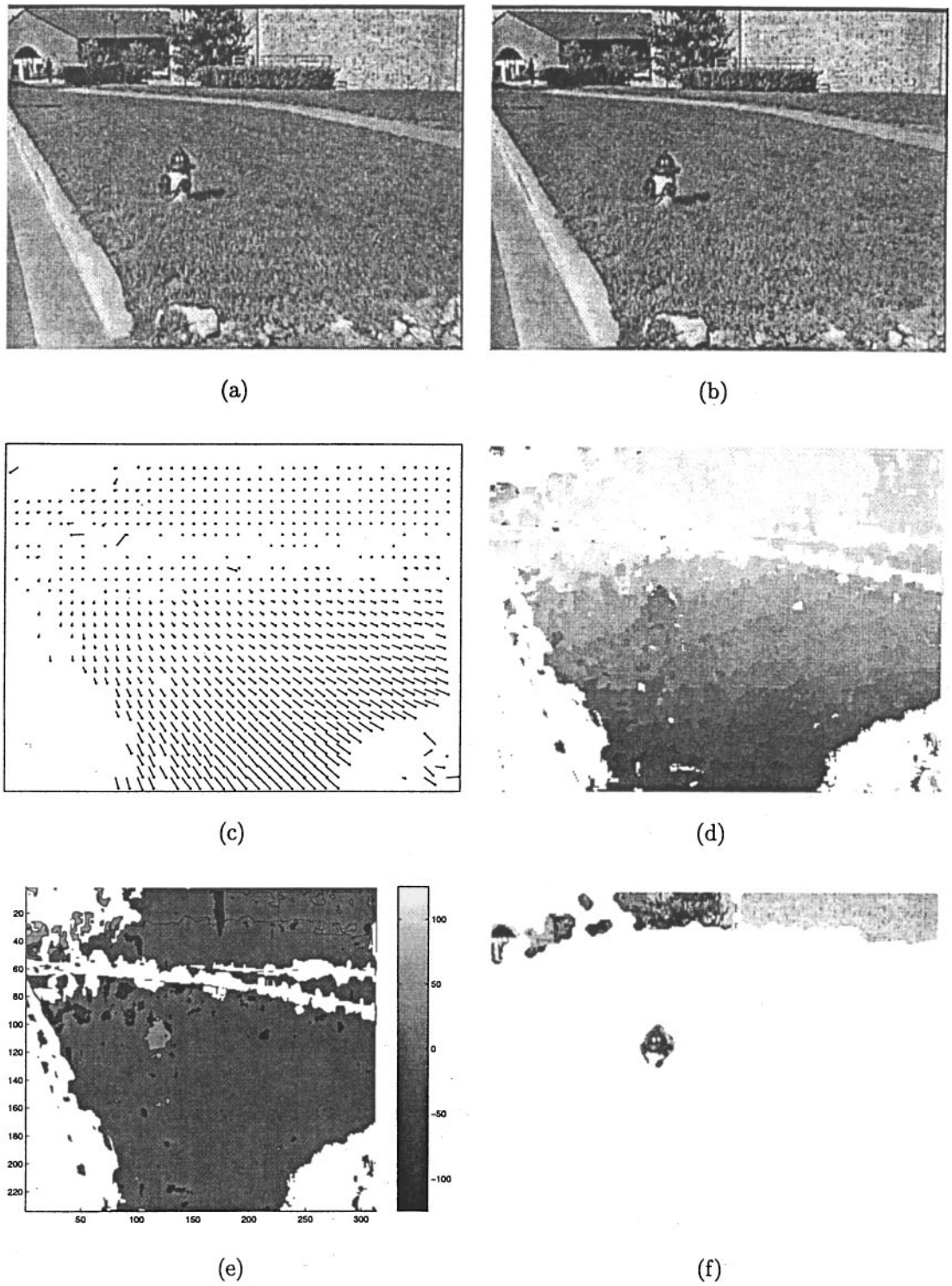

Figure 7. Obstacle detection: (a) and (b) two consecutive frames of the sequence, (c) computed optical flow, (d) inverse depth map, (e) deviation from ground plane, (f) located obstacles.

inverse depth map from the computed ground plane, that characterize obstacles. A robust planar fit may be necessary if large obstacles are present in the image. Some morphological operations are used to clean up the detected obstacles.

Figure 7 shows the results of an experiments on obstacle detection. (a) and (b) are consecutive frames for which the flow field is shown in (c). Subfigure (d) is the computed inverse depth map, where white regions are areas where no flow, and therefore no depth estimate, is available. Darker areas are closer to the camera. (e) is a contour plot of the magnitude of the deviation of the computed inverse depth from the planar fit. The final result showing detected obstacles superimposed on the original image is shown in (f).

7. Conclusions

The 3D structure from motion problem is very interesting both from the theoretical and the application points of view. Although in theory 3D motion and depth can be recovered simultaneously from a flow field, the solution has proven to be difficult. 3D SFM provides valuable cues for depth estimation, 3D stabilization, robotic navigation, obstacle avoidance, time to collision and virtual reality model generation. But a theoretically sound, robust and computationally tractable solution has eluded researchers. In this paper, we have presented what we believe is a viable solution to the problem.

Our motivation in this work has been to come up with an elegant solution that fully exploits the linear dependence of the optical flow on the focus of expansion and on the scaled inverse depth map. The fundamental result in this paper is a theoretical proof of our claim that a partial search for the focus of expansion is computationally equivalent to performing a constant number of 2D FFTs. Our experimental results on a wide variety of synthetic data representing noise in the flow field, sparsity of the computed flow, uncertainty in the focal length estimate, type of underlying depth map and field of view demonstrates the correctness, robustness and performance envelope of our algorithm. We show the utility of our algorithm for performing obstacle detection. Our experiments validate the theory behind our approach, and our claims.

3D SFM is an old problem and has received much attention over the years. In parallel, the computational power available to the image analyst has steadily increased over time. At this juncture, it is viable to use fast search-based techniques to solve computer vision problems. An advantage of our algorithm is its ready portability to digital signal processors (DSPs) and optical correlators. Having a robust solution to 3D SFM will allow future researchers to concentrate their efforts on higher level vision steps, such as primitive modeling and logical inference, in building computer vision systems.

Appendix

A.1. Partial Search

Exhaustive search has seldom been accepted by theoreticians as the best technique for arriving at the solution of a set of equations, due to its perceived inelegance and lack of mathematical structure. However, in practice exhaustive search is used when other methods fail, and when it can be performed in a reasonable amount of time. Suppose the nonlinear set of equations for which a solution is desired is given by

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \quad \mathbf{x} \in \Re^M, \quad \mathbf{0} \in \Re^K, \quad K \geq M \tag{A1}$$

Exhaustive search for a solution $\mathbf{x}_e$ involves (i) enumerating a finite set of candidate solutions $\mathcal{X} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots\}$ that adequately cover the solution space, (ii) computing an error metric (e.g. $\|\mathbf{f}(\mathbf{x_i})\|^2$) which associates each candidate solution $\mathbf{x}_i$ with a compliance measure, and (iii) locating the minimum error and corresponding candidate solution, which for the squared error metric is

$$\mathbf{x}_e = \arg \min_{\mathbf{x} \in \mathcal{X}} \{\|\mathbf{f}(\mathbf{x})\|^2\} \tag{A2}$$

In general, the order of complexity of exhaustive search is proportional to the cardinality $|\mathcal{X}|$ of $\mathcal{X}$. This can get unmanageably large as the dimensionality M of $\mathbf{x}$ increases.

An intermediate approach to searching for all the components of the solution is to enumerate only a few components and solve for the remaining components based on the hypothesis. For example in (A1), assume that the argument $\mathbf{x}$ can be partitioned as

$$\mathbf{x} = \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix}, \quad \mathbf{a} \in \Re^{M_1}, \quad \mathbf{b} \in \Re^{M_2}, \quad M_1 + M_2 = M$$

and given $\mathbf{a}_i$, (A1) can be solved for the remaining components $\mathbf{b}_i$ with a small number of computations. We will call $\mathbf{a}$ the *search component* and $\mathbf{b}$ the dependent component of $\mathbf{x}$. Partial search of a solution $\mathbf{x}_p$ is performed by (i) enumerating a finite set of candidate partial solutions $\mathcal{A} \in \{\mathbf{a}_0, \mathbf{a}_1, \ldots\}$ that adequately

cover the search component space, (ii) computing the dependent component $\mathbf{b}_i$ corresponding to each $\mathbf{a}_i \in \mathcal{A}$ that satisfies (or closely satisfies) (A1), (iii) computing an error metric, for example $\|\mathbf{f}([\mathbf{a}_i' \mid \mathbf{b}_i']')\|^2$ for the squared error case, and (iv) picking the candidate solution corresponding with the minimum error. Step (ii) can be defined as a minimization over the continuous space of permissible $\mathbf{b}$ of the same squared error metric to formulate the partial search solution $\mathbf{x}_p$ as

$$\begin{aligned}\mathbf{x}_p &= \begin{pmatrix}\mathbf{a}^* \\ \mathbf{b}^*\end{pmatrix} \\ \mathbf{a}^* &= \arg\min_{\mathbf{a}\in\mathcal{A}} \min_{\mathbf{b}} \left\|\mathbf{f}\begin{pmatrix}\mathbf{a} \\ \mathbf{b}\end{pmatrix}\right\|^2 \\ \mathbf{b}^* &= \arg\min_{\mathbf{b}} \left\|\mathbf{f}\begin{pmatrix}\mathbf{a}^* \\ \mathbf{b}\end{pmatrix}\right\|^2\end{aligned} \tag{A3}$$

Partial search separates the problem into a search and a minimization. In general, the complexity of the original problem is proportional to the cardinality $|\mathcal{A}|$ of $\mathcal{A}$, and to the number of operations required to compute $\mathbf{b}_i$ given $\mathbf{a}_i$. It may be possible, depending on the problem, to simplify certain steps in the partial search procedure, leading to further reduction in complexity. This paper approaches the SFM problem through partial search and exploits the symmetry of the problem to minimize the computational requirement.

A.1.1. Partial Search for SFM. When the optical flow u, v is known at M points $\{(x_0, y_0), (x_1, y_1), \ldots, (x_{M-1}, y_{M-1})\}$, we get M instances of (2) with $2M$ linearly independent equations and $M+5$ unknowns in all. An exhaustive search for a solution must consider the combinatorial multiplicity of the $M+5$ unknowns. This gets unwieldy even when $M=5$, which is the minimum number of equation sets needed for a unique solution. For the sake of argument, assuming that there are L possible discretized permissible values for each of the scalar variables, exhaustive search computation requires $\mathcal{O}(L^{M+5})$ operations. A crude search with 5% uncertainty in each variable (i.e. $L=10$) and with $M=5$ observed points gives rise to 10^{10} combinations, which escalates to over 10^{15} when the number of observations doubles to 10. Exhaustive search is clearly not an alternative, and indeed we require that the number of search combinations be independent of the number of observations since the latter can potentially be huge.

In order to restrict the search to a finite number of dimensions, the search component $\mathbf{a}$ must not include the depth variables $h(x_i, y_i)$. Of the remaining five variables, the rotational components are linearly related to the observation, which in this case is the flow field. Thus, in addition to the scaled inverse depth map, rotation forms an ideal candidate for including in the dependent component. The search component comprises x_f and y_f which are the focus of expansion, and the dependent component includes the rotation and depth variables which number $M+3$ in all. It can be seen in (2) that the dependent component and observation are linearly related when the search component or FOE is fixed. Given a search component $\mathbf{a}_i = (x_{f_i}, y_{f_i})'$, the dependent component $\mathbf{b}_i$ can be solved for by means of a $(M+3)\times(M+3)$ matrix inversion which takes at most $\mathcal{O}(M^3)$ steps. The worst-case computational requirement of this partial search approach is $\mathcal{O}(M^3L^2)$, which is combinatorially smaller than $\mathcal{O}(L^{M+5})$. Yet even this number is too large for practical use. Section 4 analyzes the linear system obtained when choosing a candidate search component, with the eventual aim of reducing the order of complexity of partial search.

A.2. FastCompute Specification

Objective: To compute $q(i_0, j_0) = a'\mathbf{M}\langle i_0, j_0\rangle b, \forall i_0, j_0 \in \{0, 1, \ldots, N-1\}$.
Input: Vectors $a, b \in \Re^{2N^2}$.
Output: Matrix $Q \in \Re^{N\times N}$, where $Q_{i,j} = q(i, j)$.
Internal Data: $\bar{\mathbf{M}}$, given by (42).
Data Structures: Matrices $M_{00}, M_{01}, M_{11} \in \Re^{2N\times 2N}$, initialized to

$$\begin{aligned}M_{00_{i,j}} &= \alpha_{i,j}\left(j - N + \frac{1}{2}\right)^2 \\ M_{01_{i,j}} &= -\alpha_{i,j}\left(i - N + \frac{1}{2}\right)\left(j - N + \frac{1}{2}\right) \\ M_{11_{i,j}} &= \alpha_{i,j}\left(i - N + \frac{1}{2}\right)^2 \\ \alpha_{i,j} &= \frac{1}{\left(i - N + \frac{1}{2}\right)^2 + \left(j - N + \frac{1}{2}\right)^2}\end{aligned} \tag{A4}$$

Matrices $C_{00}, C_{01}, C_{11}, \bar{Q} \in \Re^{2N\times 2N}$, initialized to zero.
Complex valued matrices $\hat{M}_{00}, \hat{M}_{01}, \hat{M}_{11}, \hat{C}_{00}, \hat{C}_{01}, \hat{C}_{11}, \hat{Q} \in \mathcal{C}^{2N\times 2N}$.

Steps

1. Initialize data structures as above.
2. Compute the discrete Fourier transforms $\hat{M}_{00}$, $\hat{M}_{01}$ and $\hat{M}_{11}$ of M_{00}, M_{01} and M_{11} respectively.
3. Write out a and b as in (34).
4. Redefine C_{00}, C_{01}, C_{11} in the upper quarter as

$$\left.\begin{aligned} C_{00_{i,j}} &= a_{iN+j,0}\, b_{iN+j,0} \\ C_{01_{i,j}} &= a_{iN+j,0}\, b_{iN+j,1} + a_{iN+j,1}\, b_{iN+j,0} \\ C_{11_{i,j}} &= a_{iN+j,1}\, b_{iN+j,1} \end{aligned}\right\} \quad \forall i, j \in \{0, 1, \ldots, N-1\} \quad \text{(A5)}$$

5. Compute the discrete Fourier transforms $\hat{C}_{00}$, $\hat{C}_{01}$ and $\hat{C}_{11}$ of C_{00}, C_{01} and C_{11} respectively.
6. Set $\hat{Q}$ to

$$[\hat{Q}]_{k,l} = [\hat{C}_{00}]^*_{k,l}\,[\hat{M}_{00}]_{k,l} + [\hat{C}_{01}]^*_{k,l}\,[\hat{M}_{01}]_{k,l} + [\hat{C}_{11}]^*_{k,l}\,[\hat{M}_{11}]_{k,l} \quad \text{(A6)}$$

   since for a real signal $x(n)$, its Fourier transform $\hat{x}(k)$ displays conjugate symmetry, i.e. $\hat{x}(k) = \hat{x}^*(-k)$. Note that this requires complex arithmetic. This summation performs, in effect, the sum over components ($\sum_2$) in 4 since $\mathcal{F}^{-1}(x) + \mathcal{F}^{-1}(y) = \mathcal{F}^{-1}(x + y)$.
7. Compute the inverse discrete Fourier transform $\bar{Q}$ of $\hat{Q}$.
8. Populate the output matrix Q copying from $\bar{Q}$:

$$Q_{i,j} = \bar{Q}_{i,j}, \quad i, j \in \{0, 1, \ldots, N-1\} \quad \text{(A7)}$$

Analysis: Step 2 involves 3 FFTs which need to be computed only once for the entire SFM process. Steps 5 and 6 require 3 and 1 FFT respectively, giving an overall complexity of 3 one-time FFTs and 4 FFTs per *FastCompute* operation.

Acknowledgments

The author wishes to thank Drs. Rama Chellappa, Venu Govindu and Azriel Rosenfeld, and the anonymous reviewers for their comments and criticism.

Notes

1. More accurately, $\mathcal{O}(N^{2\log_2 7})$ (Press et al., 1992)!
2. ftp://ftp.cfar.umd.edu/pub/shridhar/software

References

Adiv, G. 1985. Determining 3-D motion and structure from optical flow generated by several moving objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7(4):384–401.

Adiv, G. 1989. Inherent ambiguities in recovering 3-D motion and structure from a noisy flow field. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(5):477–489.

Aloimonos, Y. and Brown, C. 1984. Direct processing of curvilinear sensor motion from a sequence of perspective images. In *IEEE Workshop on Computer Vision, Representation and Control*, Annapolis, MD, pp. 72–77.

Barron, J., Fleet, D., and Beauchemin, S. 1994. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77. Software and test sequences available at ftp.csd.uwo.ca/pub/vision.

Bruss, A. and Horn, B. 1983. Passive navigation. *Computer Vision, Graphics and Image Processing*, 21(1):3–20.

Fejes, S. and Davis, L. 1998. What can projections of flow fields tell us about visual motion. In *International Conference on Computer Vision*, Mumbai, India, pp. 979–986.

Fermuller, C. and Aloimonos, Y. 1995. Qualitative egomotion. *International Journal of Computer Vision*, 15(1/2):7–29.

Fermuller, C. and Aloimonos, Y. 1997. On the geometry of visual correspondence. *International Journal of Computer Vision*, 21(3):223–247.

Gibson, J. 1955. *The Perception of the Visual World*. Houghton Mifflin.

Gibson, J. 1957. Optical motion and transformations as stimuli for visual perceptions. *Psychological Review*, 64(5):288–295.

Gupta, N. and Kanal, L. 1995. 3-D motion estimation from motion field. *Artificial Intelligence*, 78(1/2):45–86.

Heeger, D. and Jepson, A. 1992. Linear subspace methods for recovering translation direction. Technical Report RBCV-TR-92-40, University of Toronto. www.cs.utoronto.ca/~jepson/abstracts/dither.html.

Horn, B. and Weldon, E., Jr. 1988. Direct methods for recovering motion. *International Journal of Computer Vision*, 2(1):51–76.

Koenderink, J. and vanDoorn, A. 1976. Local structure of movement parallax of the plane. *Journal of the Optical Society of America A*, 66(7):717–723.

Lawton, D. 1983. Processing translational motion sequences. *Computer Vision, Graphics and Image Processing*, 22(1):116–144.

Longuet-Higgins, H. 1981. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135.

Longuet-Higgins, H. and Prazdny, K. 1980. The interpretation of a moving retinal image. *Proceedings of the Royal Society of London B*, B-208:385–397.

Mitiche, A. 1994. *Computational Analysis of Visual Motion*. Plenum.

Mitiche, A., Zhuang, X., and Haralick, R. 1987. Interpretation of optical flow by rotational decoupling. In *IEEE Workshop on Computer Vision, Representation and Control*, Miami Beach, FL, pp. 195–200.

Nakayama, K. and Loomis, J.M. 1974. Optical velocity patterns, velocity-sensitive neurons and space perception. *Perception*, 3:63–80.

Negahdaripour, S. and Horn, B. 1987. Direct passive navigation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(1):168–176.

Oliensis, J. 1997. A critique of structure from motion algorithms. Technical Report www.neci.nj.com/homepages/oliensis/Critique.html, NECI.

Prazdny, K. 1981. Determining the instantaneous direction of motion from optical flow generated by a curvilinearly moving observer. *Computer Vision, Graphics and Image Processing*, 17(3):238–248.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. 1992. *Numerical Recipes in C*, 2nd edn. Cambridge University Press: Cambridge, UK.

Simoncelli, E.P. 1993. Distributed representation and analysis of visual motion. Ph.D. Thesis, MIT.

Srinivasan, S. 1998. Image sequence analysis-estimation of optical flow and focus of expansion, with applications. Ph.D. Thesis, University of Maryland, College Park.

Tsai, R. and Huang, T. 1981. Estimating 3-D motion parameters of a rigid planar patch I. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 29(12):1147–1152.

Ullman, S. 1979. The interpretation of structure from motion. *Proceedings of the Royal Society of London B*, B-203:405–426.

Wallach, H. and O'Connell, D.N. 1953. The kinetic depth effect. *Journal of Experimental Psychology*, 45:205–217.

Waxman, A., Kamgar-Parsi, B., and Subbarao, M. 1987. Closed-form solutions to image flow equations for 3D structure and motion. *International Journal of Computer Vision*, 1(3):239–258.

Waxman, A. and Ullman, S. 1985. Surface structure and three-dimensional motion from image flow kinematics. *International Journal of Robotics Research*, 4(3):72–94.

Weng, J., Hwang, T.S., and Ahuja, N. 1991. *Motion and Structure from Image Sequences*. Springer-Verlag: Berlin.

Young, G. and Chellappa, R. 1992. Statistical analysis of inherent ambiguities in recovering 3-D motion from a noisy flow field. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(10):995–1013.

Zhuang, X., Huang, T., Ahuja, N., and Haralick, R. 1984. Rigid body motion and the optic flow image. In *First IEEE Conference on AI Applications*, pp. 366–375.

Zhuang, X., Huang, T., Ahuja, N., and Haralick, R. 1988. A simplified linear optical flow-motion algorithm. *Computer Vision, Graphics and Image Processing*, 42(3):334–344.