Linear Subspace Methods for Recovering Translational Direction

Allan D. Jepson
University of Toronto

David J. Heeger
NASA-Ames Research Center

Abstract

The image motion field for an observer moving through a static environment depends on the observer's translational and rotational velocities along with the distances to surface points. Given such a motion field as input we have recently introduced subspace methods for the recovery of the observer's motion and the depth structure of the scene. This class of methods involve splitting the equations describing the motion field into separate equations for the observer's translational direction, the rotational velocity, and the relative depths. The resulting equations can then be solved successively, beginning with the equations for the translational direction. Here we concentrate on this first step. In earlier work, a linear method was shown to provide a biased estimate of the translational direction. We discuss the source of this bias and show how it can be effectively removed. The consequence is that the observer's velocity and the relative depths to points in the scene can all be recovered by successively solving three **linear** problems.

This paper is the University of Toronto, Department of Computer Science, Technical Report: RBCV-TR-92-40, April 1992. Correspondence should be sent to A. Jepson, Department of Computer Science, University of Toronto, 6 Kings College Road, Toronto, Ontario M5S 1A4.

1 Introduction

The basic problem we consider is how to obtain reliable information on the motion of a camera, along with distances to various points in its environment, from measurements of image motion (optical flow) alone. Here we pursue subspace methods which have been recently introduced for solving this problem (see [5, 6]). The general approach involves splitting the problem into three subproblems, each of which can be solved in the following order. First, we obtain constraints which involve only the translational direction, $\vec{T}$, of the camera. These equations are independent of the camera's angular velocity, $\vec{\Omega}$, and do not involve knowing the distances to points in the scene. Secondly, given the resulting estimate for the translational direction, a set of linear equations can be obtained which involve only the rotational velocity as an unknown. Finally, given estimates of both the translational direction and rotational velocities of the camera, several methods are available for obtaining reliable information about the (relative) distances to various scene points (see [2, 9]).

Unlike many previously proposed algorithms, our approach to motion analysis applies to the general case of arbitrary motion with respect to an arbitrary scene. There is no assumption of smooth or planar surfaces. The results in our previous work [5, 6] demonstrate that our approach can be stable with respect to random errors in the flow field measurements, and that it performs quite favorably when compared with other proposed approaches. It is simple to compute and it is highly parallel, not requiring iteration and not requiring an initial guess. For a brief review of the existing literature see [6].

In this paper we are primarily concerned with simplifying the first step of the subspace methods in which the direction of translation is estimated independently of the rotational velocity and depths. The input data is taken to be a discrete set of optical flow vectors, say $\vec{u}(\vec{x}_k)$, for $k = 1, \ldots, K$, where $\vec{x}_k$ denotes the image position for the k^{th} sample. It is convenient to collect these two-vectors into a single $2K$-dimensional vector, $\vec{\mathcal{O}}$. With this notation, the constraints on $\vec{T}$ take the simple form

$$\vec{\Psi}_i(\vec{T}) \cdot \vec{\mathcal{O}} = 0, \quad \text{for } i = 1, \ldots, K-3, \tag{1.1}$$

where "$\cdot$" denotes the usual vector inner-product. In previous work [5, 6] we show how to compute the constraint vectors $\vec{\Psi}_i$, and show that these vectors are typically nonlinear functions of $\vec{T}$. We proposed that the nonlinear problem (1.1) can be solved for $\vec{T}$ simply by sampling the constraints on a mesh distributed over a hemisphere of possible orientations for $\vec{T}$, and then seeking points of least square error[5, 6].

In [7] we introduced an alternative method for finding the camera's translational direction, which avoids sampling $\vec{T}$-space at many points, and results in a *linear* system for the translational direction. The key observation to make is that it is possible to redefine the constraint vectors $\vec{\Psi}_i(\vec{T})$ in equation (1.1) in such a way that the first $K-6$ of them depend linearly on $\vec{T}$, while the remaining three constraint vectors are typically nonlinear functions of $\vec{T}$. From (1.1) we see that the first $K-6$ vectors now lead to linear constraints on the translational direction. The step in our previous method which involved sampling constraints of the form (1.1) over $\vec{T}$-space can be replaced by the construction of these linear constraints, followed by a standard *linear* least squares solver. As a result this new method represents a huge savings in the computational resources required to obtain an estimate for $\vec{T}$.

There is, of course, a price to pay for this short cut. We are not using all of the available information to obtain this estimate of $\vec{T}$ (i.e. we are omitting the three nonlinear constraints).

Therefore we can expect that the new approach will be more sensitive to errors in the input. Fortunately, we can show that for general scenes containing a rich depth structure this linear approach still does provide a robust estimate [7]. However our preliminary experiments indicated that a straight forward implementation of the linear approach provides a biased estimate of the translational direction. This bias was observed to be more severe for images having smaller angular extents. In this paper we analize the cause of this bias, and discuss a simple method for its removal. In related work Spetsakis [11] discusses the bias observed in a completely different method for structure from motion, and comments on the possibility of the biases having a common cause. This is an important area of research, which we do not pursue here.

2 Basic Algorithm

The basic situation we consider is an observer moving through a stationary environment. In the observer's coordinate frame this is equivalent to the scene undergoing a rigid motion, which is completely characterized by a translational velocity $\vec{T}$ and an angular velocity $\vec{\Omega}$. In particular, the instantaneous velocity of the point $\vec{X}(t)$ is

$$\frac{d\vec{X}}{dt} \equiv \vec{V} = \vec{T} + \vec{\Omega} \times \vec{X}. \tag{2.1}$$

Here we take $\vec{X} \equiv (X_1, X_2, X_3)$ to be a right-handed coordinate system fixed on the observer, with the nodal point of the imaging system at the origin. We set the focal length to f, and denote the image point at (X_1, X_2, f) by image coordinates $\vec{x} \equiv (x_1, x_2, 1) = (X_1/f, X_2/f, 1)$ (for vector operations later in this paper it is convenient to write $\vec{x}$ as a 3-vector). We put the transducer surface in front of the nodal point to avoid the need to reflect the image coordinates.

We assume that we are given the optical flow (image velocity) at a set of image positions, $\{\vec{x}_k\}$, for a single frame of the image sequence. From this information we wish to compute the direction of the translational velocity. As mentioned in the introduction, given this direction we are left with linear problems for each of the rotational velocity and the inverse relative depths. Here we consider only this first step, namely the computation of the translational direction.

For the algorithm proposed in [7], the optical flow data at each point is first expanded into the 3-vector

$$\vec{q}(\vec{x}_k) = Q(\vec{x}_k)\vec{u}(\vec{x}_k), \tag{2.2}$$

where

$$Q(\vec{x}) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \\ x_2 & -x_1 \end{pmatrix}. \tag{2.3}$$

Notice that this preprocessing step is local; each image velocity is transformed separately.

The sample points $\{\vec{x}_k\}$ are subdivided into N (usually overlapping) patches. Let the n^{th} image patch consist of sample points $\{\vec{x}_k\}_{k=1}^{K}$. Then for each patch we define a particular coefficient vector, $\vec{c}_n \equiv (c_{1n}, \ldots, c_{Kn})^T$. The details of the computation of a suitable coefficient vector are given in Appendix A. For our purposes here we only note that the coefficients have been normalized so that

$$\sum_{k=1}^{K} c_{kn}^2 = 1. \tag{2.4}$$

Given the transformed optical flow vectors $\vec{q}(\vec{x}_k)$, and the coefficient vector $\vec{c}_n$ for the n^{th} image patch, we next build the *translation constraint vector*

$$\vec{\tau}_n \equiv \begin{pmatrix} \vec{c}_n \cdot (q_1(\vec{x}_1), \ldots, q_1(\vec{x}_K))^T \\ \vec{c}_n \cdot (q_2(\vec{x}_1), \ldots, q_2(\vec{x}_K))^T \\ \vec{c}_n \cdot (q_3(\vec{x}_1), \ldots, q_3(\vec{x}_K))^T \end{pmatrix}. \tag{2.5}$$

That is, the i^{th} component of $\vec{\tau}_n$ is obtained by taking the inner-product of the coefficient vector $\vec{c}_n$ with the vector formed from the i^{th} component of $\vec{q}(\vec{x})$ sampled over the image patch. The particular construction of the coefficient vectors $\vec{c}_n$ ensures that the resulting translation constraint vector, $\vec{\tau}_n$, is perpendicular to the true translational direction (see Appendix A).

Taken over all N image patches, each with its own $\vec{\tau}_n$, we now find that the translational direction $\vec{T}$ of the observer must satisfy the linear equation

$$\begin{pmatrix} \vec{\tau}_1^{\,T} \\ \vec{\tau}_2^{\,T} \\ \vdots \\ \vec{\tau}_N^{\,T} \end{pmatrix} \vec{T} = \vec{0}. \tag{2.6}$$

We can solve this in a least squares sense by accumulating the following 3×3 symmetric matrix

$$D \equiv \sum_{n=1}^{N} w_n \vec{\tau}_n \vec{\tau}_n^{\,T}. \tag{2.7}$$

Here we have included a weight, w_n, for each $\vec{\tau}$ vector, the choice of which is discussed further below. The least squares estimate for the translational direction is then given by the eigenvector for the smallest eigenvalue of this 3×3 matrix D. We have found it is also useful to look at the separation between the various eigenvalues of D, to detect situations for which one or more components of the translational direction are poorly determined by the linear constraints.

An important special case for the above computation is when the optical flow is sampled on a regularly spaced grid, and the image patches are all taken to have the same sampling pattern (eg. a $l \times l$ square grid). In such a case the coefficient vectors $\vec{c}_n$ can be precomputed and taken to be the same for each patch (see Appendix A). For a regular sampling pattern, then, the overall algorithm for the computation of the translational direction simplifies to the following. First the optical flow data $\vec{u}(\vec{x})$ at each sample point is transformed to a 3-vector, $\vec{q}(\vec{x})$, through a multiplication by the 3×2 matrix $Q(\vec{x})$. Three images, $q_i(\vec{x})$, are formed from each component of this vector. Each of these images is then convolved with the same precomputed mask made up of the coefficients of $\vec{c}_n$, and the three resulting images together provide an image of translation constraint vectors. That is, for each spatial position there is a translation constraint vector $\vec{\tau}_n$ known to be perpendicular to the observer's translational direction. These constraint vectors are finally compiled into the 3×3 matrix D, from which

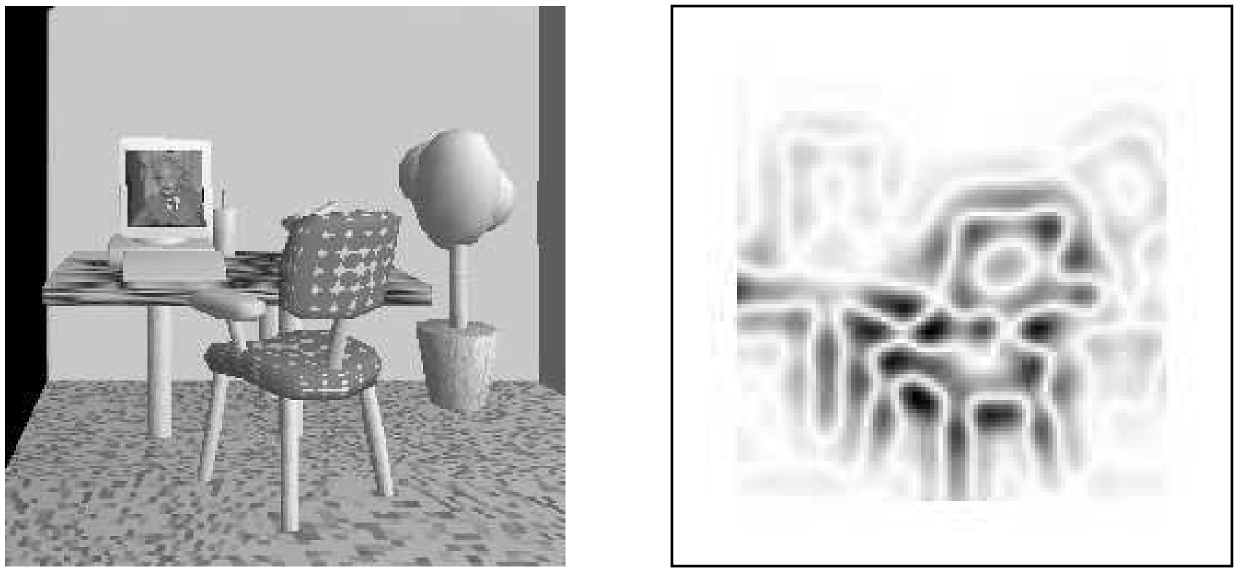

Figure 3.1: (Left) Graphics image of an office (128×128 pixels). (Right) Image formed from the norm of the recovered $\vec{\tau}(\vec{x})$, with larger norms depicted by darker grey levels. Here the motion field is as in Figure 3.2 (left).

the translational direction can easily be obtained. It is also possible, of course, to avoid explicitly forming any of these intermediate images, and instead directly build the matrix D from the results of an equivalent linear transformation applied to the optical flow for each image patch. We choose to present the algorithm in the expanded form primarily to emphasize the simplicity of the computations involved.

3 Direct Implementation.

In this section we examine the performance of the basic algorithm on a simple test problem. Our previous work showed that the algorithm appears to be fairly robust, but suffers from a bias in the recovered translational direction. We review the observation of a bias here through a straight forward implementation. In subsequent sections we use these results for comparison with approaches designed to reduce or remove the bias.

Due to the ease in which we can obtain well controlled test data, we consider only computer generated data in this paper. Clearly for the results on such sequences to generalize to real sequences we need to model the noise properties of optical flow measurement techniques. Fleet and Jepson [4], for example, report optical flow measurements with roughly Gaussianly distributed errors having a magnitude about 5% of the length of the optical flow vector. These error results were also obtained from simulated scenes, and therefore should be taken as a *lower* bound on the sorts of errors we can expect in practice (see also [1, 3]).

The optical flow data we use is generated from the depth map for the computer generated scene shown in Figure 3.1. The range of projected distances along the optical axis (i.e. the range of values in the "Z-buffer") is roughly a factor of two for this scene. Since we expect the accuracy of the results to depend on the visual angle, we allow the focal distance to vary and keep the Z-buffer fixed. In Figure 3.2 we show a motion field generated using this Z-buffer,

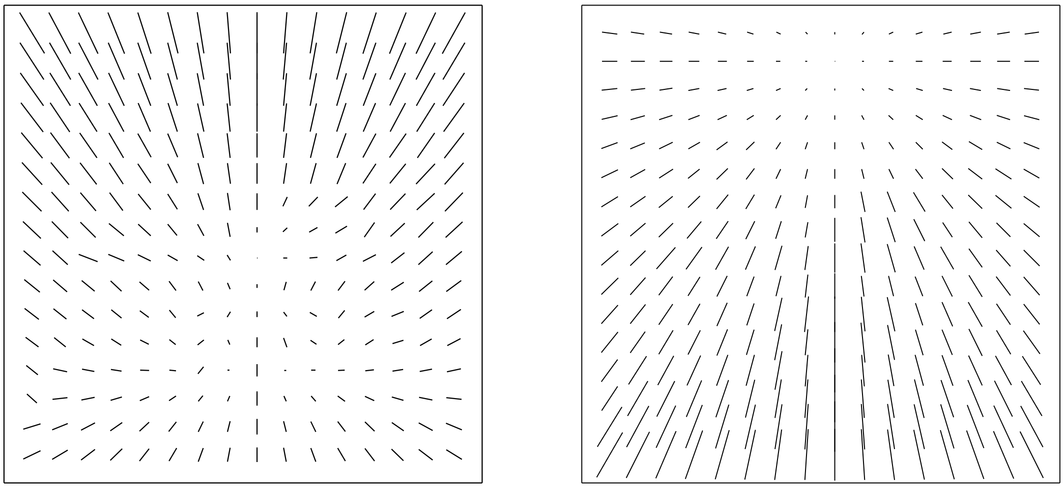

Figure 3.2: (Left) Flow field for the office scene, with translational velocity $\vec{T} = (0, -1, 2)$ (towards and up), and with a fixation point at the center of image. (Right) The component of the flow field due to only the translational motion. Note the focus of expansion, which is located at the true direction of translation.

a 60 degree field of view, and a translational velocity of $(0, -1, 2)$ (image coordinate x_1 is to the right, x_2 is down, and the third component is forward along the optical axis). The rotational velocity was chosen so that the camera was fixating at the center of the image. Figure 3.2 (left) shows the flow field, while only the translational component is presented in Figure 3.2 (right). Note that there is a very significant contribution to the motion field from the rotational component. This is important since the algorithm is designed to project out the effects of the rotational motion.

The optical flow data was obtained by adding isotropic, Gaussianly distributed noise to the motion field. We used a noise amplitude such that each component of the error has a standard deviation equal to 10% of the length of the motion field vector (so the norm of the error is a factor of $\sqrt{2}$ larger than this). The actual amplitude of the noise is not particularly important. Indeed we have done some tests using up to 30% noise and observed a graceful degradation in performance. However, to deal with computed optical flow data we need to contend with noise that is correlated across nearby image locations, and with noise distributions which have longer tails signifying a higher probability of outliers. We do not deal with these issues here, although in Section 6 we do show the results of our algorithm given computed optical flow data.

Figure 3.1 (right) shows the norm of $\vec{\tau}$ generated by the algorithm. Here the patches for the optical flow data were chosen to be square 15×15 sampling patterns, with the samples each separated by a distance of two pixels. The coefficient vector was chosen to be the modified DOG mask discussed in Appendix A. The same mask was used for all patches and the translation constraint vectors $\vec{\tau}(\vec{x})$ were generated using the convolution algorithm described above. (Here we find it convenient to refer to the $\vec{\tau}$ generated for a particular patch as $\vec{\tau}(\vec{x})$, where $\vec{x}$ is the image coordinates for the center of the patch.) The norm of the resulting translation constraint vectors, $\vec{\tau}(\vec{x})$, is essentially the absolute value of the response of the DOG mask when applied to the inverse depth data with a general modulation

Angular Extent (deg)	60	40	20	10	5
Error in Mean (deg)	7.0	12	19	24	26
Error T_1 (deg)	0.3 ± 0.1	0.2 ± 0.1	0.08 ± 0.04	0.03 ± 0.02	0.01 ± 0.01
Error T_2 (deg)	6.5 ± 0.6	11.4 ± 0.4	18.6 ± 0.2	23.6 ± 0.1	25.5 ± 0.01

Table 3.1: Results from the basic algorithm

by the length of $\vec{x} \times \vec{T}$ (see [7]). (No value was computed around the border of the image where the convolution mask would have extended beyond the edge of the image.) As we discussed in [7], the translation constraint vector $\vec{\tau}(\vec{x})$ has a large amplitude only where there is significant nonplanar variation in the depth structure. The amplitude response could clearly be useful for other operations such as image segmentation.

For a 60 degree field of view (measured along the horizontal center line), and 10% input error, the recovered translational direction was $(-0.007, -0.329, 0.944)$. The exact direction (after scaling) is $(0.0, -0.447, 0.894)$. The computed solution represents an error of about 13% in the recovered translational direction, or about 7 degrees of visual angle. Note that the vertical component of $\vec{T}$ (i.e. T_2) has the largest error, while the component of the error in the horizontal direction T_1 is only 0.4 degrees of visual angle. Thus the recovered translational direction is skewed about 7 degrees towards the optical axis. Twenty separate runs were done (each taking less than a minute on a Silicon Graphics 4D/340VGX), and the standard deviation of the error measure (in degrees) was found to be less than a degree (see Table 3.1). In particular, we found that the method has a significant bias (roughly 7 degrees) in the recovered translational direction towards the orientation of the optical axis.

The same Z-buffer and 3D motion parameters were used for various other angular extents. The results of 20 runs for each extent are summarised in Table 3.1. For example, for a 20 degree field of view the recovered translational direction is strongly skewed towards the optical axis, with the average recovered translational direction having a a 19 degree error. The standard deviation in the radial direction (T_2) was only about 0.2 degrees of visual angle, which is negligible with respect to the mean error of 18.6 degrees in this component. As a result we can safely conclude that the error in the mean represents represents a strong bias in the method rather than random fluctuations. From Table 3.1 it is clear that the bias increases as the angular extent is reduced. A second important point is that the horizontal component of the translational direction (i.e. T_1) is recovered quite accurately even for a 20 degree angular extent. For significantly smaller extents, such as the 10 degree and 5 degree cases reported in Table 3.1, the recovered translational direction is skewed to within a few degrees of the optical axis, and thus an accurate horizontal component (nearly zero) may not be significant here.

4 Source of the Bias

We have seen that the variance of the estimates for the direction of translation is small, but the basic method is seriously flawed by a significant bias. Note that in the absence of noise our method is exact, the various constraint vectors $\vec{\tau}(\vec{x})$ provide linear constraints on the translational direction. (Indeed, given the motion field to single precision floating point accuracy our program returns an estimate of the translational direction accurate to six digits.) The bias is clearly not a consequence of analytical errors. Moreover, only linear operations are used to construct the constraint vectors. Therefore we should expect that the bias can be understood through a simple linear analysis of the effects of noise in the optical flow on the resulting constraint vectors.

For our simulated optical flow it is relatively straight forward to compute the covariance of the constraint vectors $\vec{\tau}(\vec{x})$. In particular, since we are assuming the noise at different pixels is independent,

$$E[(\vec{\tau}-E[\vec{\tau}])(\vec{\tau}-E[\vec{\tau}])^T] = \sum_{i=1}^{K} c_i^2 E[(\vec{q}_i - E[\vec{q}_i])(\vec{q}_i - E[\vec{q}_i])^T].$$

Here $E[\vec{\tau}]$ and $E[\vec{q}_i]$ represent the expected values of the vectors $\vec{\tau}$ and $\vec{q}_i$. The last term in the above expression is the covariance of $\vec{q}_i$. By equation (2.2) this covariance can be rewritten in terms of the 3×2 matrix $Q(\vec{x}_i)$ and the covariance of the noise for the optical flow. Since the optical flow noise is assumed to be isotropic, its covariance is simply a constant, $\rho^2||u(\vec{x}_i)||^2$, times the 2×2 identity matrix. (For 10% noise in each component of the optical flow we use $\rho = 0.10$.) As a result, we can now rewrite the above equation as

$$E[(\vec{\tau}-E[\vec{\tau}])(\vec{\tau}-E[\vec{\tau}])^T] = \rho^2 \sum_{i=1}^{K} ||u(\vec{x}_i)||^2 c_i^2 Q(\vec{x}_i) Q^T(\vec{x}_i).$$

For our purposes here it is convenient to approximate this covariance matrix using the assumption that the norm of the optical flow is roughly constant over the sample points $\vec{x}_i$ used in any single patch. In particular, we replace the term $||\vec{u}(\vec{x}_i)||^2$ in the above sum with the weighted average

$$\sigma_u^2 = \sum_{i=1}^{K} c_i^2 ||u(\vec{x}_i)||^2 \tag{4.1}$$

(recall (2.4) which requires the coefficients c_i to be normalized such that the sum of their squares is one). This provides the approximate covariance matrix

$$C_n = \rho^2 \sigma_u^2 \sum_{i=1}^{K} c_i^2 Q(\vec{x}_i) Q^T(\vec{x}_i). \tag{4.2}$$

The sum in equation (4.2) is over terms which only depend on the particular sample points $\vec{x}_i$, and not on the optical flow.

Using equation (2.3), the sum in (4.2) can be shown to be

$$\sum_{i=1}^{K} c_i^2 Q(\vec{x}_i) Q^T(\vec{x}_i) = \begin{pmatrix} 1 & 0 & -x_1^n \\ 0 & 1 & -x_2^n \\ -x_1^n & -x_2^n & \alpha+\beta \end{pmatrix}. \tag{4.3}$$

Here x_1^n and x_2^n are the first and second components of the center $\vec{x}^{\,n}$ of the n^{th} patch, which is defined in general by

$$\vec{x}^{\,n} = \sum_{i=1}^{K} c_{in}^2 \vec{x}_i.$$

(For the symmetric modified DOG masks used in this paper, $\vec{x}^{\,n}$ is just the position of the center pixel of the mask when applied to the n^{th} patch.) Also $\alpha = (x_1^n)^2 + (x_2^n)^2$ and the term β in (4.3) is given by

$$\beta = \sum_{i=1}^{K} c_i^2 [(x_{1i} - x_1^n)^2 + (x_{2i} - x_2^n)^2].$$

We see β is simply a measure of the variance of the sample points weighted by the coefficients c_i. For the case in which the $\vec{\tau}$ vectors are computed using convolution with the same set of coefficients, the value of β is simply a constant independent of $\vec{x}^{\,n}$.

The structure of the approximate covariance matrix C_n is best illustrated by its eigenvalues and eigenvectors, which provide the variances and principle directions. The eigenvalues for C_n are just $(\rho\sigma_u)^2$ times the eigenvalues of the matrix given in (4.3), and the eigenvectors of the two matrices are identical. The eigenvalues of the matrix in (4.3) are easily found to be $\lambda_1 = 1$, and

$$\begin{aligned} \lambda_+ &= (1 + \alpha + \beta)\gamma, \\ \lambda_- &= \beta/((1 + \alpha + \beta)\gamma). \end{aligned} \tag{4.4}$$

Here

$$\gamma = \frac{1}{2}\left[1 + \sqrt{1 - 4\beta/(1 + \alpha + \beta)^2}\right].$$

The eigenvectors associated with these three eigenvalues are given (respectively) by the columns of the matrix

$$U = \begin{pmatrix} x_2^n & x_1^n & x_1^n \\ -x_1^n & x_2^n & x_2^n \\ 0 & \delta_+ & \delta_- \end{pmatrix}. \tag{4.5}$$

For simplicity, we have ignored the normalization factors for these column vectors. The constant δ_+ is given by $1 - \lambda_+$, and the second constant δ_- satisfies $\alpha = -\delta_+\delta_-$.

The significance of these equations is most easily seen by considering sampling patterns having a small angular extent. In this case β, a weighted variance of the sampling pattern, is much smaller than one. To be specific, for the sampling pattern used here based on a 15×15 array of taps applied at every other pixel, β is 1.5×10^{-3} for the 60 degree field of view. This decreases to 6.0×10^{-4} for the 40 degree field of view, and to only 3.5×10^{-5} for the 10 degree field of view. As a consequence, we find δ_+ is roughly given by $-\alpha$, and δ_- is roughly 1. In particular, from (4.5) we see that the eigendirections can be simply determined from the position of the center of the image sampling pattern, $\vec{x}^{\,n}$. The eigenvalues for these three directions are 1, $\lambda_+ = 1 + \alpha + O(\beta)$, and $\lambda_- = \beta/(1+\alpha) + O(\beta^2)$. For an image having an angular extent of 60 degrees (side to side) α ranges from zero to a maximum of 2/3, so the first two eigenvalues of C_n are roughly the same size. However, the third eigenvalue, corresponding to the sampling direction $\vec{x}^{\,n}$, is much smaller. The one standard deviation surface for the errors in $\vec{\tau}$ are ellipsoids having the shape of mildly elliptical "pancake", with the diameter along the minor axis of the ellipsoid much smaller than along the other two principal directions. (In fact the ratio of diameters is at least $1/\sqrt{\beta}$, which is 26 for a 60 degree field of view and 170 for 10 degrees.) In summary, we see the noise for the computed $\vec{\tau}$ vectors is far from isotropic, and the minor axis of each pancake is oriented along the direction of the center of the optical flow sampling pattern, $\vec{x}^{\,n}$.

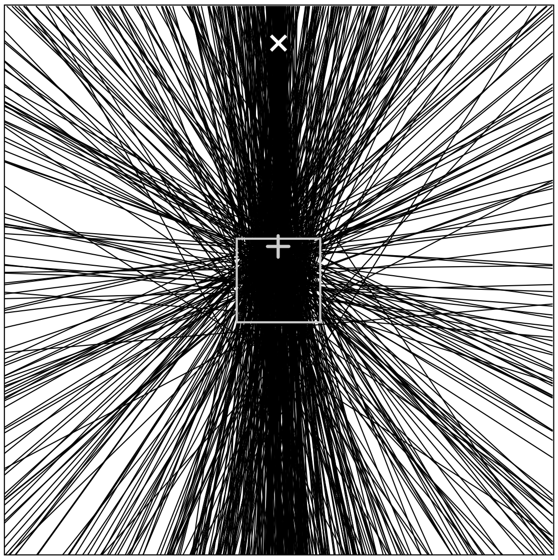

Figure 4.1: Translational constraint lines for $\vec{T}$, plotted in a 60 degree window centered on the optical axis (i.e. the same image coordinates as used for Figure 3.2). The true translational direction is indicated by the white "X". In order to accentuate the bias we used only a 10 degree field of view (grey box), and the recovered translational direction is indicated by the grey "+".

To illustrate one effect of this non-isotropic error distribution we plot a large sample of the resulting constraints in Figure 4.1. Here the $\vec{\tau}$ vectors were computed given a 10 degree field of view (depicted by the grey box in Figure 4.1), and 10% image noise added to the flow field generated using the same 3D motion parameters as in Figure 3.2 (left). For each sampled $\vec{\tau}$ vector we plot the set of translational directions $\vec{T}$ which are perpendicular to $\vec{\tau}$, as projected onto an image having the same optical axis as the original but with a 60 degree angular extent. The set of translational directions perpendicular to a particular $\vec{\tau}$ forms a straight line in this image. One other property of the constraint vectors is needed to understand the distribution shown in Figure 4.1, namely that the exact $\vec{\tau}$ vectors tend to be nearly perpendicular to the mean sampling direction. In particular, the exact constraint lines tend to pass close to the sampling direction [7]. Since the covariance of $\vec{\tau}$ is small in this direction, the noise does not perturb the position of the constraint lines very much near the sampling direction. As a result, in Figure 4.1 we see that almost all of the constraint lines pass through the ten degree field of view from which the optical flow data was sampled.

The recovered direction is (roughly) the point that minimizes the sum of squared distances to all of these straight lines, with each distance weighted by the squared length of the particular $\vec{\tau}$. For this case the recovered direction is about 24 degrees away from the true translational direction. As can be seen from Figure 4.1, one source of this bias may be the significant portion of constraint lines that are extremely noisy. These arise from cases where the length of $\vec{\tau}$ is not much longer than the expected length of the noise. Due to the non-isotropic nature of the noise even these constraint line will also typically pass close to the sampling direction, and the result is a radial pattern of noisy constraints as depicted in Figure 4.1. For such a radial pattern, points near the center of the pattern will tend to have a smaller square distance to all of the lines, and hence these noisy constraints can be

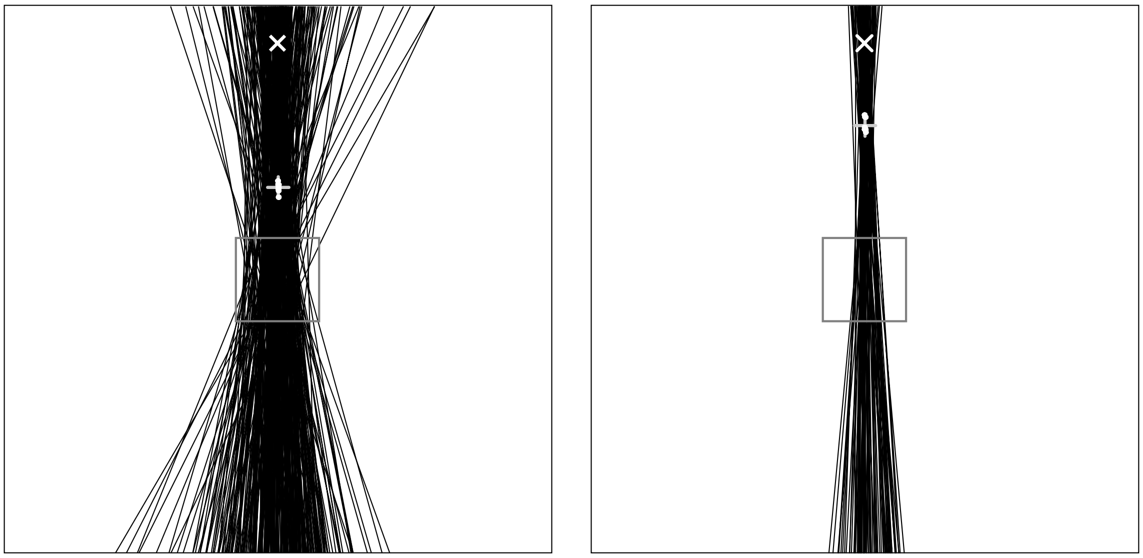

Figure 4.2: Translational constraint lines for the same case as in Figure 4.1, but for only those constraints having a signal noise ratio larger than 5 (left) or 30 (right). White "X"'s mark the true translational direction. A dot is used to indicate the result from each of 10 different noise samples, and they all lie on the vertical bars of the the grey "+"'s marking the mean recovered translational directions.

expected to provide some contribution to the bias.

Given the above estimate for the noise in a particular $\vec{\tau}$ vector it is a simple matter to implement a constraint on the signal to noise ratio of the constraint vectors used to estimate $\vec{\tau}$. For example, we use the standard deviation $\rho\sigma_u$ associated with the first eigenvalue of C_n, to define

$$SNR[\vec{\tau}(\vec{x})] = ||\vec{\tau}(\vec{x})||/[\rho\sigma_u]. \tag{4.6}$$

This estimate is used in two ways. First, we implement a simple threshold on the estimated SNR of each $\vec{\tau}$, if the estimate does not exceed a critical value then $\vec{\tau}$ is not used in the least squares problem. (In terms of equation (2.7) the weight w_n is set to zero.) For translation constraint vectors which exceed this threshold, we use the weight $w_n = (\rho\sigma_u)^{-2}$.

The rationale for this choice of weight is that, for a general translational velocity $\vec{T}$, the minimum distance between $\vec{T}$ and the plane perpendicular to $\vec{\tau}$ is $|\vec{\tau} \cdot \vec{T}|/||\vec{\tau}||$. The product of the square of this distance and the square of the SNR for $\vec{\tau}$ gives $(\vec{\tau}(\vec{x}^{\,n}) \cdot \vec{T})^2/(\rho\sigma_u)^2$. This is precisely the contribution we obtain from the least squares objective function $\vec{T}^{\,T} D\vec{T}$ defined in Section 2, given that the weight w_n is chosen as above. Thus, we are weighting the distance between $\vec{T}$ and the plane perpendicular to $\vec{\tau}$ by the signal to noise ratio defined in (4.6).

The results of using this weighting scheme are shown in Figure 4.2. The input data were the same as for Figure 4.1. The threshold has successfully eliminated the wildly incorrect constraints, and clearly larger values of the threshold provide a tighter distribution of constraint lines near the true translational direction. However, the use of a threshold has reduced but not removed the bias. Indeed, the results of 10 runs are shown in each half of Figure 4.2. The scatter of the individual results is significantly less than the mean error (its about the size of the vertical bar of the "+" marking the mean recovered direction). Table

Angular Extent (deg)	60	40	20	10	5
Error in Mean (deg)	3.3	5.0	11	19	24
Error T_1 (deg)	0.08 ± 0.1	0.05 ± 0.1	0.03 ± 0.07	0.1 ± 0.03	0.1 ± 0.01
Error T_2 (deg)	3.0 ± 0.7	4.5 ± 0.7	10 ± 0.7	19 ± 0.5	24 ± 0.5

Table 4.1: Results of using a threshold of 5.0 on the signal to noise ratio $SNR[\vec{\tau}]$, and using the scalar weights w_n.

4.1 provides similar results from 20 runs using different random noise samples, for each of several given focal lengths. While there is a significant improvement in the performance over the basic algorithm (compare Table 3.1), the bias is still severe for smaller angular extents.

5 Removing the Bias

In the previous section we noted the pancake shaped distribution for noise in the $\vec{\tau}$ vectors, and the property that the exact constraint vectors are nearly perpendicular to the sample direction. As we discussed above, these two properties contribute to the bias in the recovered translational direction, in part through the behaviour of the constraint vectors having a low signal to noise ratio. However, even for relatively high signal to noise ratios the bias was still observed. In this section we briefly discuss the reason for this remaining bias, and demonstrate a simple method for its removal.

Consider a single constraint vector $\vec{\tau}(\vec{x}^{\,n})$ having a large signal to noise ratio, and assume it is nearly perpendicular to the sampling direction $\vec{x}^{\,n}$, as is typical. The plane perpendicular to this constraint vector forms the set of all translational velocities that are perfectly consistent with $\vec{\tau}(\vec{x}^{\,n})$. As discussed in the previous section, we are minimizing the objective function $\vec{T}^{\,T} D\vec{T}$, which is simply the sum of squared distances between the translational velocity $\vec{T}$ and the planes perpendicular to $\vec{\tau}(\vec{x}^{\,n})$, for $n = 1, \ldots, N$, with each squared distance weighted by the square of the SNR for $\vec{\tau}(\vec{x}^{\,n})$. The important point is that this term depends only on the relative angle between $\vec{T}$ and $\vec{\tau}(\vec{x}^{\,n})$, and not on the orientation of $\vec{T}$ around the axis aligned with $\vec{\tau}(\vec{x}^{\,n})$. The contribution of the n^{th} patch to the overall objective function is therefore rotationally symmetric about the translation constraint vector $\vec{\tau}(\vec{x}^{\,n})$. As we show below, it is precisely this symmetry which causes the significant bias.

Note that the rotational symmetry does not model the non-isotropic character of the noise in the recovered constraint vectors. In particular, the variance of $\vec{\tau}(\vec{x}^{\,n})$ was seen to be much smaller in the direction of the mean sample point of the n^{th} patch than in the perpendicular directions. The plane of translational directions $\vec{T}$ that are perpendicular to this constraint vector, therefore, will not wobble very much across the sampling direction for different noise samples. Instead the dominant effect of the noise will be to rotate the plane of translational velocities around the sampling direction. Roughly speaking we should expect the rotation around the sampling direction to have a variance that is $1/\beta$ times the variance across the viewing direction (i.e. the ratio of the eigenvalues of C_n). Recall that β is 1.5×10^{-3} for the 60 degree field of view and smaller for narrower fields. Thus an appropriate objective function should weight perturbations in $\vec{T}$ from the constraint plane much more heavily for translational directions nearly parallel to the sampling direction than for directions nearly perpendicular both the sampling direction and $\vec{\tau}(\vec{x}^{\,n})$. The ratio of the

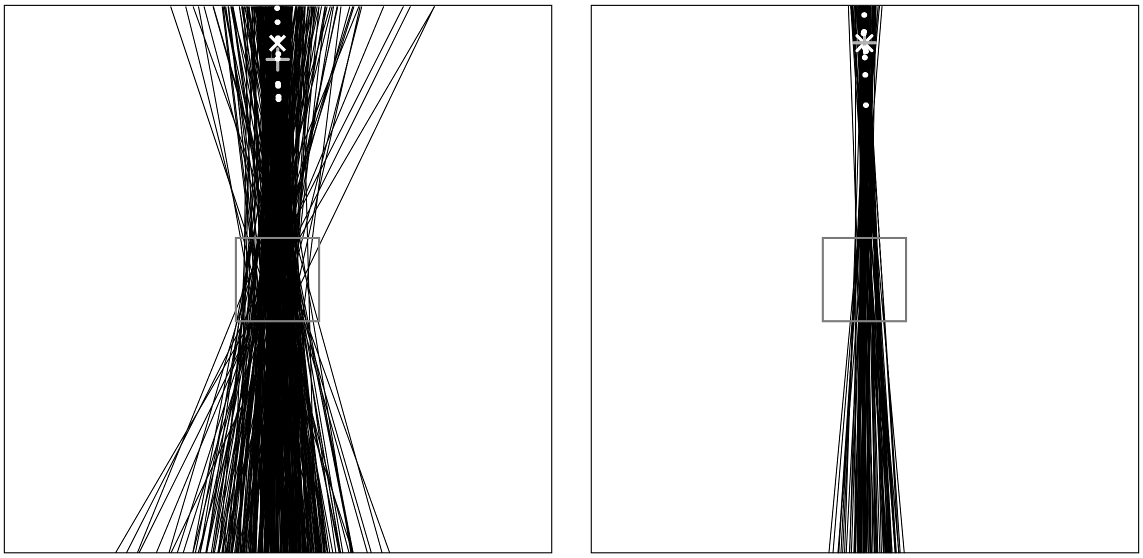

Figure 5.1: Translational constraint lines for the same case as in Figure 4.1, with a signal to noise threshold of 5 (left) and 30 (right). The dithering method was used on ten different runs for each threshold (dots), and the mean recovered translational direction is given by the grey "+".

weights on the same size error occurring at these two extremes should be $1/\sqrt{\beta}$, which for our current application increases from 26 for the 60 degree field of view, to 344 for the 5 degree field of view!

In terms of plots such as Figures 4.1 and 4.2, the appropriate cost function can be thought of in terms of the displacement between the translational direction $\vec{T}$ (in the image plane) and the depicted constraint line. The same perpendicular displacement should cost much more when it occurs near the sampling direction than when it occurs further away. By using a rotationally symmetric cost function we are in effect under-weighting perturbations near the sampling direction. Given a narrow aperture image we are consistently under-weighting the perturbations near the optical axis and, considering the magnitude of the appropriate weighting, we should expect a strongly biased estimate.

In order to correct this problem we need to use the non-isotropic character of the covariance matrix for the constraint vectors. A standard approach is to use the covariance matrix to generate a cost function in the following manner. Consider the space of all constraint vectors that are consistent with a given candidate translational direction $\vec{T}$. This is just the plane perpendicular to $\vec{T}$. If this was the true translational direction, then the exact $\vec{\tau}$ must be in this plane. A natural choice for the cost of such translational direction is the minimum squared distance between the measured constraint vector $\vec{\tau}(\vec{x}^{\,n})$ and this plane, where the distance metric is based on the covariance matrix C_n. That is, the cost is given by the minimum value

$$\min(\vec{\tau} - \vec{\tau}(\vec{x}^{\,n}))^T C_n^{-1} (\vec{\tau} - \vec{\tau}(\vec{x}^{\,n})), \quad \text{for } \vec{\tau} \cdot \vec{T} = 0.$$

This problem is easily solved, and we find the minimum value corresponds to

$$\frac{1}{(\vec{T}^{\,T} C_n \vec{T})} (\vec{\tau}(\vec{x}^{\,n}) \cdot \vec{T})^2.$$

Angular Extent (deg)	60	40	20	10	5
Error in Mean (deg)	0.2	0.2	0.3	0.5	8.8
Error T_1 (deg)	0.08 ± 0.2	0.03 ± 0.14	0.01 ± 0.12	0.03 ± 0.09	0.03 ± 0.11
Error T_2 (deg)	0.2 ± 0.7	0.14 ± 0.8	0.3 ± 1.4	0.5 ± 5.4	8 ± 22

Table 5.1: Results of the dithering method using a threshold of 5.0 on the signal to noise ratio, and the scalar weight w_n.

That is, the appropriate cost function can be viewed in the same general form $\vec{T}^{\,T} D(\vec{T})\vec{T}$, but for $D(\vec{T})$ defined by

$$D(\vec{T}) = \sum_{n=1}^{N} w_n(\vec{T}) \vec{\tau}(\vec{x}^{\,n}) \vec{\tau}(\vec{x}^{\,n})^{\,T}, \tag{5.1}$$

with the weights

$$w_n(\vec{T}) = \frac{1}{(\vec{T}^{\,T} C_n \vec{T})} \tag{5.2}$$

Notice the weights depend on the translational velocity $\vec{T}$ itself, and thus in general the objective function $\vec{T}^{\,T} D(\vec{T})\vec{T}$ is no longer quadratic. We note in passing that by changing the sign of this objective function we obtain an approximation to the log likelihood of the translational direction $\vec{T}$, given all the constraint vectors $\vec{\tau}(\vec{x}^{\,n})$ which pass the condition on the signal to noise ratio. Thus we can view this approach as a maximum likelihood method, as has been suggested in [12].

There are, of course, techniques for finding locally optimal values of non-quadratic objective functions. However, they require initial guesses and iterations, or alternatively, we might consider directly sampling the unit sphere of possible translational directions on a mesh to locate the minimum. The latter approach is similar to our earlier non-linear approach for finding the translational direction [5], and its use would largely eliminate the benefit of having linear constraints on the translational direction.

We would therefore like a quadratic objective function which effectively implements the one above. At first glance this seems like an impossibility. However, the key idea is that we are free to *add* noise to the constraint vectors $\vec{\tau}(\vec{x}^{\,n})$ in order to alter their covariance matrices; such a process is called dithering. If we can alter the covariance matrices C_n such that they are all scalar multiples of the same matrix, say C_0, then a simple (non-isotropic) rescaling of $\vec{T}$ will allow the nonquadratic objective function $\vec{T}^{\,T} D(\vec{T})\vec{T}$ to be written in an equivalent quadratic form. Note that we cannot do this rescaling trick with the original covariance matrices C_n since they are not all scalar multiples of each other. In particular, their one standard deviation surfaces were shown to be pancake shaped, with the minor axis of the pancake oriented along the mean sampling direction. As the mean sampling direction changes, the minor axis changes, and the various covariance matrices cannot be scalar multiples of each other. However, by dithering, we can pad the various pancakes so they all have the same shape.

In this paper we simply dither the constraint vectors to ensure that the resulting one standard deviation surfaces are all nearly spherical. For small angular extents this is rather crude, in that the minor axes of the various ellipsoids are nearly aligned to begin with, and a significantly smaller amount of dithering would suffice. However, for 90 degree fields of view we would be faced with dithering to spherical surfaces, as implemented here.

In particular, consider a computed constraint vector $\vec{\tau}(\vec{x}^{\,n})$ and the estimated covariance matrix C_n of the form described in Section 4. Two eigenvalues of C_n were shown to be roughly $\rho^2\sigma_u^2$, while the third eigenvalue was a factor of λ_- smaller. The third eigendirection was shown to be roughly the mean sampling direction $\vec{x}^{\,n}$ for the n^{th} patch. The dithering we use is simply to add a random component to each constraint vector, $\vec{\tau}(\vec{x}^{\,n})$, in the mean sampling direction $\vec{x}^{\,n}$, with an amplitude given by a mean zero Gaussian process having variance of $\sigma_{dith}^2 = \rho^2\sigma_u^2(1.0 - \lambda_-)$. That is,

$$\vec{d}_n = \vec{\tau}(\vec{x}^{\,n}) + N(0;\sigma_{dith}^2)\vec{x}^n/||\vec{x}^n||. \tag{5.3}$$

The result of this dithering is that the covariance matrix of $\vec{d}_n$ has the minimum eigenvalue of $(\rho\sigma_u)^2$, which occurs twice, while the remaining eigenvalue is a factor of $(1+\alpha)$ larger (see equation (4.4)). Since α is between 0 and 2/3 for all of the cases we are interested in here, we conclude that the covariance matrices of $\vec{d}_n$ are roughly isotropic. (We could of course dither in two directions to make the covariance perfectly isotropic, but remember that C_n is only an approximation of the true covariance. Also, our results below indicate that this extra dithering in not necessary in practice.)

Each constraint vector is used to generate a dithered vector $\vec{d}_n$, and these dithered constraint vectors are then used to generate

$$D = \sum_{n=1}^{N} w_n \vec{d}_n \vec{d}_n^{\,T}.$$

Here the weights are chosen exactly as in Section 4, namely w_n is zero if the signal to noise ratio of $\vec{\tau}(\vec{x}^{\,n})$ is below a threshold, and otherwise $w_n = (\rho\sigma_u)^{-2}$. Finally, the eigenvalues of D are found, and the eigenvector corresponding to the minimum eigenvalue provides the estimate for the translational direction. (We also check the ratio of the eigenvalues to detect cases in which only partial information on the translational direction is recovered.)

The results of this process are shown in Figure 5.1 and Table 5.1. For all practical purposes, the bias is gone. The results in Table 5.1 show that the translation direction can be recovered quite accurately even down to a 10 degree angular extent. The standard deviation of the estimate in the radial direction increases as the angular extent is reduced, as is expected since the constraint lines become more parallel. However, the mean error in the method is consistently less than the standard deviation of the responses, indicating that the bias has virtually been eliminated. (There should be some bias left, since the dithered constraints do not have perfectly isotropic covariances, but it now appears to be negligible.) The horizontal component of the translational direction was recovered to within a fraction of a degree for all cases.

For the angular extent of 5 degrees we are approaching, or within, the domain of applicability of techniques which rely on an orthographic approximation, such as the factoring approach discussed in [13]. At this extreme we find that the eigenvalues values of the matrix D are in the proportion (300:1.2:1). The fact that the last two eigenvalues have nearly the same magnitude indicates that the translational direction is only weakly constrained within the plane spanned by the last two eigendirections. The first eigendirection of D provides a robust constraint that the true translational direction lies somewhere on the plane perpendicular to it, which is accurate to about a tenth of a degree. The separation of the smallest

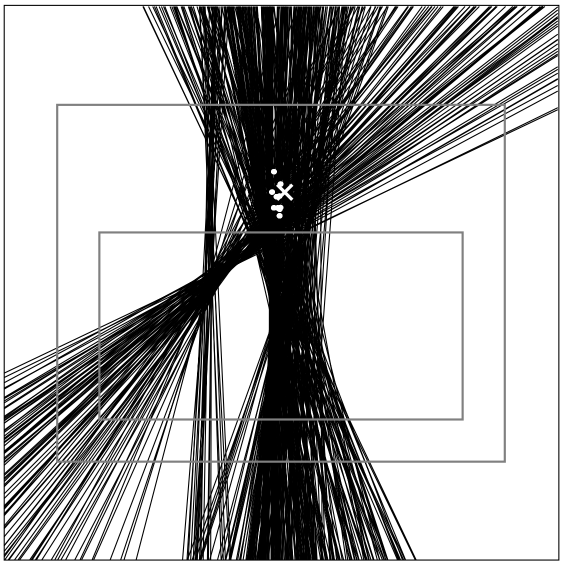

Figure 6.1: Results for frame 7 of the Yosemite sequence. The dithering method was used on ten different runs (dots). The error in the mean recovered translational direction is 1.5 degrees.

two eigenvalues increases as the angular extent increases, indicating that the translational direction becomes more strongly constrained in this plane. In particular, the proportions of the eigenvalues of D for the 10 degree extent are typically (300:1.6:1), for 20 degrees we have (250:3:1), and finally for the 60 degree case the proportions are (200:8:1). Thus, the proportions appear to be useful diagnostic of the nature of the information provided about the translational direction. We note in passing that the ratios of the eigenvalues of D, generated by the basic method and the weighted method discussed above, does not provide such a useful diagnostic. For example, in the 5 degree field of view the basic method produces the proportions (5×10^3:240:1), while the simple weighting produces (1.2×10^4:48:1). Both results indicate that the methods strongly constrain the translational direction to a unique line, as is verified by the small standard deviations of the observed responses (see Tables 3.1 and 4.1). The trouble is that a large part of this constraint is due entirely to the bias in the method. Therefore situations in which the true constraints are nearly defective are not identified by the eigenvalues of the matrix D generated by the biased methods. By destroying the bias, the dithering method avoids this problem.

6 Yosemite Revisted

While the results of the dithering algorithm are an impressive improvement over the other methods reported here, they are based on the use of simulated optical flow data. As discussed above, there are several additional difficulties we might expect to be confronted with given real data. The two primary concerns are: 1) optical flow measurements are typically sparse; and 2) given an environment with occlusion boundaries the optical flow measurements will typically have numerous outliers (see [3]).

There are two possible approaches to dealing with sparse data. In particular, the data values might be interpolated on a uniform grid. In this case we can apply our approach directly to the interpolated values, but we need to be concerned about correlations and structure introduced in the errors through the interpolation process itself. Alternatively, we might consider using only the sparse set of image points at which reliable estimates of optical flow can be obtained. In this case the coefficient vector $\vec{c}_n$ cannot be precomputed since it depends on the particular locations of the sample points. These coefficients must satisfy six linear equations (see Appendix A), and an efficient method for solving these equations is an important area for further research.

There are also several possible approaches for dealing with the second problem, namely that optical flow methods must be expected to generate the occasional outlier. For example, given that some information about the 3D motion can be obtained with a crude method, we might consider using this information to identify outliers in the data. Methods from robust statistics could have an important application here.

For the purposes of this paper we simply demonstrate the existing algorithm on an optical flow field computed for the (synthetic) Yosemite image sequence. This sequence has been used with our non-linear algorithm, which produced an error in the translational direction of about 4 degrees [5]. Here we use a different optical flow field, namely one provided by Eero Simoncelli generated using the method described in [10]. The field has been interpolated and sampled on a regular grid, so the convolution method can be directly applied. Due to errors in the interpolant near the borders of the image and near the horizon, we found it necessary to crop the flow field. In Figure 6.1 the outside frame represents our usual 60 degree field of view, the next largest rectangle represents the frame of the original Yosemite sequence, and the inside rectangle represents the field of view in which we kept the optical flow data.

The results of using 10 different noise samples in the dithering are shown in Figure 6.1. To get this quality of result we needed to use a large convolution mask. In particular we used a 31×31 square array formed by modifying a DOG mask with a center standard deviation of 3 pixels and 6 for the surround (see Appendix A). Successive taps for this mask were separated by 4 pixels, since it was found that smaller masks generated only a planar constraint on the translational direction. An additional 20 runs were performed (each run took 30 seconds on a SGI 4D/340VGX and computed about 1,000 translation constraint vectors), and they generated a mean translational direction with an error of 1.8 degrees. The standard deviation over these twenty runs was 1.3 degrees in the vertical direction and 0.5 degrees horizontally. These results compare favourably to those of our previous nonlinear method.

In summary, we have identified the cause of the bias in the basic linear subspace algorithm for the recovery of the translational direction. Moreover, we have demonstrated a simple method for its removal. The new method appears to be fairly robust and capable of generating useful diagnostics of its performance. Further work needs to be done before it can be generally applied to computed optical flow fields. However, the simplicity of the underlying approach, the absence of extraneous local minima, and no need for an initial guess or iterations, all provide strong motivating factors for pursuing this approach further.

Acknowledgements

Thanks to NSERC Canada and NASA for financial support. The first author is also a member of the Institute for Robotics and Intelligent Systems (IRIS) and wishes to acknowledge support of the Networks of Centers of Excellence Program of the Government of Canada and the participation of PRECARN Associates Inc. Finally, thanks to Eero Simoncelli for providing the Yosemite flow, and to Alex Pentland and Brad Horowitz for providing the office scene.

Appendix A.

Here we discuss the conditions on the coefficient vector $\vec{c}$ needed to ensure that the resulting translation constraint vectors $\vec{\tau}$ are orthogonal to the true translational direction. To do this we first need the basic equations for the motion field induced by a rigid motion. Image velocity, $\vec{u}(\vec{x})$, is defined as the derivatives, with respect to time, of the image coordinates of the projection of a scene point $\vec{X}(t)$. Using the rigid motion equation (2.1), and the perspective projection equation

$$\vec{x}(t) = \frac{1}{X_3(t)}\vec{X}(t),$$

we find that the image velocity is given by (see [6])

$$\vec{u}(\vec{x}) = p(\vec{x})A(\vec{x})\vec{T} + B(\vec{x})\vec{\Omega}. \tag{A.1}$$

Here $p(\vec{x}) = 1/X_3$ is inverse depth, $\vec{T}$ is the translational velocity and $\vec{\Omega}$ is the rotational velocity. Also, the matrices A and B are given by

$$A(\vec{x}) = \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -x_2 \end{bmatrix}$$
$$B(\vec{x}) = \begin{bmatrix} -(x_1x_2) & (1+x_1^2) & -x_2 \\ -(1+x_2^2) & (x_1x_2) & x_1 \end{bmatrix}.$$

The $A(\vec{x})$ and $B(\vec{x})$ matrices depend only on the image position $\vec{x}$, not on any of the unknowns. (Note that the image position $\vec{x}$ is measured per unit focal length, so there is an implicit dependence on f.)

The first step of our algorithm involves the premultiplication of the image velocity samples by the matrix $Q(\vec{x})$ defined in (2.3). Using (A.1) above, the resulting vector $\vec{q}(\vec{x})$ is given by

$$\vec{q}(\vec{x}) = p(\vec{x})Q(\vec{x})A(\vec{x})\vec{T} + Q(\vec{x})B(\vec{x})\vec{\Omega}. \tag{A.2}$$

Here the product $Q(\vec{x})A(\vec{x})\vec{T}$ is easily seen to be

$$Q(\vec{x})A(\vec{x})\vec{T} = \begin{pmatrix} 0 & 1 & -x_2 \\ -1 & 0 & x_1 \\ x_2 & -x_1 & 0 \end{pmatrix}\vec{T} = \vec{T} \times \vec{x}, \tag{A.3a}$$

where "$\times$" denotes the usual vector cross product. Similarly, a straight forward calculation shows that

$$Q(\vec{x})B(\vec{x})\vec{\Omega} = \begin{pmatrix} -(1+x_2^2) & x_1x_2 & x_1 \\ x_1x_2 & -(1+x_1^2) & x_2 \\ x_1 & x_2 & -(x_1^2+x_2^2) \end{pmatrix}\vec{\Omega} = \vec{x} \times (\vec{x} \times \vec{\Omega}). \tag{A.3b}$$

Taken together, we find the simple expression for $\vec{q}(\vec{x})$

$$\vec{q}(\vec{x}) = (\vec{T} \times \vec{x})p(\vec{x}) + \vec{x} \times (\vec{x} \times \vec{\Omega}). \tag{A.4}$$

This form is sufficient to motivate the constraints on the coefficient vector $\vec{c}$.

In particular we seek coefficients which ensure that $\vec{\tau} \cdot \vec{T}$ vanishes,. for the translation constraint vector defined by

$$\vec{\tau} = \sum_{i=1}^{K} c_i \vec{q}(\vec{x}_i). \tag{A.5}$$

It is useful to first consider only the component of $\vec{q}$ due to the translational velocity (i.e the first term on the right hand side of (A.4)). Notice that the inner product of this term with $\vec{T}$ must vanish, for all image positions $\vec{x}_i$, since the cross product $\vec{T} \times \vec{x}_i$ is perpendicular to $\vec{T}$. Thus, only the second term in (A.4) contributes, and we find

$$\vec{\tau} \cdot \vec{T} = \sum_{i=1}^{K} c_i \vec{T} \cdot [\vec{x}_i \times [\vec{x}_i \times \vec{\Omega}]]. \tag{A.6}$$

Here $\vec{T} \cdot [\vec{x}_i \times [\vec{x}_i \times \vec{\Omega}]]$, is simply a quadratic polynomial in terms of the image coordinates, whose coefficients depend on the particular values of $\vec{T}$ and $\vec{\Omega}$ (compare equation (A.3b)). In fact, it can be shown that any quadratic polynomial can be represented by a particular choice of $\vec{T}$ and $\vec{\Omega}$. Therefore a sufficient condition for $\vec{\tau} \cdot \vec{T}$ to vanish is that the coefficients $\vec{c}$ must be orthogonal to the samples of any quadratic polynomial.

We can write this condition on the coefficients using a basis for the quadratic polynomials on the image plane, say $\{1, x_1, x_2, x_1^2, x_1x_2, x_2^2\}$. This basis is evaluated at each image sample point, $\vec{x}_i$ for $i = 1, \ldots, N$, and the results are collected into the i^{th} column of a $6 \times N$ matrix F. The condition that $\vec{c}$ is perpendicular to the samples of any quadratic polynomial is then equivalent to the condition that it is perpendicular to the samples of each of the basis polynomials, that is

$$F\vec{c} = \vec{0}. \tag{A.7}$$

Moreover, for generic sampling patterns, F is of full rank, so we can expect equation (A.7) to have a $K - 6$ dimensional space of solutions. Note that the 6-dimensional space of quadratic polynomials is invariant under affine deformations. Therefore the solution vectors of (A.7), namely $\vec{c}$, can also be taken to be invariant of affine deformations of the sampling pattern. As mentioned in Section 2, this invariance is important for the convolution form of our algorithm.

A couple of other methods can be understood from this formulation. In particular, the original nonlinear method described in [5, 6] is equivalent to choosing the coefficients $\vec{c}$ such that they annihilate the samples of all polynomials of the same general form $\vec{T} \cdot [\vec{x} \times [\vec{x} \times \vec{\Omega}]]$. The difference is that we treated $\vec{T}$ as fixed (our candidate translational direction), and only varied the rotational velocity $\vec{\Omega}$. As a result, we required only three conditions on the coefficients instead of six. However the extra three conditions depend on the particular choice of $\vec{T}$, and thus the coefficients $\vec{c}$ also turn out, in general, to be functions of $\vec{T}$. In the notation of this paper, the consequence of this is that the translation constraint vectors $\vec{\tau}$ themselves depend on $\vec{T}$, and we obtain nonlinear constraints of the form $\vec{\tau}(\vec{T}) \cdot \vec{T} = 0$ on the translational direction. In general we note that *a priori* constraints on the translational or rotational velocities can be used to restrict the space of polynomials that the coefficients must be orthogonal to, thereby increasing the number of independent choices for the coefficients.

A second variant is due to da Vitorio Lobo and Tsotsos [8]. They consider only sampling patterns for which the sample points all lie on a line in the image plane passing through the projection of the true translational direction. That is, $\vec{x}_i = \vec{x}(s_i)$ where

$$\vec{x}(s) = (\vec{T} + s\vec{x}_0), \tag{A.8}$$

and the vector $\vec{x}_0$ is assumed to have a zero third component. Substitution of this expression for the sample points into (A.6) provides a polynomial that is quadratic in s, with the constant term

$$\vec{T} \cdot [\vec{T} \times [\vec{T} \times \Omega]] = 0.$$

Therefore, in this special case (A.6) takes the form

$$\vec{\tau} \cdot \vec{T} = \sum_{i=1}^{K} c_i s_i \left[a_0(\vec{T}, \vec{\Omega}) + a_1(\vec{T}, \vec{\Omega}) s_i \right],$$

for some coefficients a_0 and a_1. In particular the *product* $c_i s_i$ must be perpendicular to the samples of any *linear* polynomial in s (i.e. the above expression for $\vec{\tau} \cdot \vec{T}$ should vanish for any a_0 and a_1). This constraint can be satisfied given three or more distinct points $\{\vec{x}(s_i)\}_{i=1}^{3}$ that are colinear and have the true translational direction on the same line. In such a case, constant values for $c_i s_i$, $i = 1, \ldots, K$ can be computed for a particular sampling geometry. Finally, in order to eliminate the (unknown) terms s_i, it is convenient to first expand the Q matrix into the form

$$Q(\vec{x}(s)) = Q(\vec{T}) + s \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ -x_{2,0} & x_{1,0} \end{pmatrix}.$$

Upon substitution of this expansion into $\vec{\tau} \cdot \vec{T}$ we find the terms depending on $Q(\vec{T})$ do not contribute to the inner product. Moreover, the remaining terms involve only the product $c_i s_i$, which are in fact the constant coefficients computed above. Therefore, we see the method of da Vitorio Lobo and Tsotsos reduces to a constant coefficient case in a novel way.

Returning to the method studied in this paper, we consider the constraint (A.7) on the coefficient vector. As we mentioned above this constraint is independent of any affine transformation of the sampling points, and depends only on the particular pattern of sample points in the patch. Therefore in situations where the patches are all chosen to be the same, such as the square $l \times l$ patches used in this paper, the *same* coefficient vector $\vec{c}$ can be precomputed and used for all the patches. Here we use a coefficient vector constructed by modifying a DOG (Difference of Gaussians) mask so that it satisfies (A.7). The center standard deviation of the original DOG was 1.5 pixels, and the surround was set to 3 pixels. A 15×15 set of filter taps was used. These DOG coefficients were modified by adding a quadratic polynomial which was windowed by the surround Gaussian (i.e. a Hermite function). The particular Hermite function added was uniquely determined by the constraint (A.7). The original DOG required only a small modification. In addition, the result was normalized so that

$$\sum_{k=1}^{K} c_{kn}^2 = 1. \tag{A.9}$$

Finally, in Section 6 we used an DOG operator of twice the size (31×31), with twice the center and surround standard deviations listed above. This mask was modified in a similar way to satisfy (A.7).

References

[1] J.L. Barron, D.J. Fleet, S.S. Beauchemin, T.A. Burkitt, Performance of optical flow techniques, TR299, Dept. of Computer Science, University of Western Ontario, 1992 (see also CVPR 1992).

[2] R.C. Bolles, H.H. Baker, and D.H. Marimont, Epipoloar-plane image analysis: An approach to determine structure from motion, *Int. J. of Comp. Vis.*, 1 (1987), pp.7-55.

[3] D.J. Fleet, Measurement of Image Velocity , Kluwer, Boston (1992).

[4] D.J. Fleet and A.D. Jepson, Computation of component image velocity from local phase information, Int. J. of Comp. Vision, 5:1 (1990), pp. 77-104.

[5] D.J. Heeger and A.D. Jepson, Simple method for computing 3D motion and depth, Proc. ICCV 1990, Osaka, Japan, pp.96-100.

[6] D.J. Heeger and A.D. Jepson, Subspace methods for recovering rigid motion I: Algorithm and implementation, *Int. J. of Comp. Vision*, V.7, N.2 (1992), pp.95-117.

[7] A.D. Jepson and D.J. Heeger, A fast subspace algorithm for recovering rigid motion, Proc. IEEE Workshop on Visual Motion, Princeton (1991), pp. 124-131.

[8] N. da Vitoria Lobo and J.K. Tsotsos, Using collinear points to compute egomotion and detect nonrigidity, IEEE CVPR Proc., Hawaii (1991), pp. 344-350.

[9] L. Matthies, R. Szeliski, and T. Kanade, Kalman filter-based algorithms for estimating depth from image sequences, *Int. J. of Comp. Vis.*, 3 (1989), pp.209-238.

[10] E.P. Simoncelli, E.H. Adelson, and D.J. Heeger, Probability distributions of optical flow, IEEE CVPR Proc., Hawaii (1991), pp. 310-315.

[11] M.E. Spetsakis, Models of Statistical Visual Motion Estimation. Tech. Rep. CS-91-06, Dept. of Computer Science, York University, North York, Ontario (1991).

[12] M.E. Spetsakis and J. Aloimonos, Optimal computing of structure from motion using point correspondences in two frames, Proc. of IEEE ICCV, Florida (1988), pp. 449-453, 684.

[13] C. Tomasi and T. Kanade, Factoring image sequences into shape and motion, Proc. IEEE Workshop on Visual Motion, Princeton (1991), pp. 21-28.