Linear Structure From Motion

Inigo Thomas
Eero Simoncelli

MS-CIS-94-61
GRASP LAB 383

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA USA

Linear Structure From Motion

Inigo Thomas **Eero Simoncelli**

GRASP Laboratory, and
Institute for Research in Cognitive Science
University of Pennsylvania
3401 Walnut Street, Philadelphia, PA 19104

email: {inigo,eero}@grip.cis.upenn.edu

Abstract

Determining the structure of the world and the motion of the observer from image changes has been a central problem in computer vision for over fifteen years. Since the early work on *Structure from Motion* (SFM) by Longuet–Higgins [4] and Pradzny [6], several techniques have been developed to compute the motion of the camera, the shape of moving objects, or distances to points in the world. However, the image changes are *non–linearly* related to camera motion and distances to points in the world. Thus, solving the problem typically requires non–linear optimization techniques that can be unstable or computationally inefficient. Linear algorithms are preferable since they are computationally advantageous, and since linear estimation is much better understood than non–linear estimation. Our paper describes an unbiased, completely linear algorithm for Structure–from–Motion. This work is similar to that of Jepson & Heeger [3] except that we employ spherical projection. The use of a spherical imaging geometry allows a simpler and more intuitive derivation of the algorithm, and produces an unbiased estimator. Experimental results are provided that demonstrate the performance of the algorithm.

Keywords: motion analysis, egomotion estimation, structure-from-motion, subspace techniques

1 Introduction

Determining world structure and egomotion from image motion has been a central problem in computer vision for over fifteen years. Since the early work on *Structure from Motion* (SFM) by Longuet–Higgins [4] and Pradzny [6], several techniques have been developed to compute the motion of the camera, the shape of moving objects, or distances to points in the world (cf. [9] for a review). Under perspective projection, the image change – or optical flow – is *non–linearly* related to camera motion and distances to points in the world. Thus, solving the problem typically requires non–linear optimization techniques that can be unstable or computationally inefficient. A fully linear algorithm would be preferable on various grounds, since linear estimation is computationally advantageous and is much better understood than non–linear estimation. The main contribution of this paper is a simple unbiased, linear estimator for direction of heading (DOH) and SFM. The algorithm is based on a simple, geometrically intuitive representation of optical flow in a spherical coordinate system.

Two linear algorithms have recently been attempted for SFM. One of them – due to Tomasi & Kanade [11] – assumes an orthographic projection model (or variations of this model such as para–perspective in later work), thus eliminating the nonlinear effects of perspective projection. Although this assumption makes it possible to reconstruct the shape of the environment using a linear algorithm, the required camera models are very restrictive and break down when the scene is made up of objects close to the camera.

A less restrictive approach has been taken by Jepson and Heeger [3], who use the full perspective projection model. They cancel the component of flow due to rotation of the camera by projecting flow onto a particular linear subspace. The result of this operation, which annihilates the rotation, is then observed to be linearly related to the translation direction. The Jepson/Heeger algorithm is very similar to the two algorithms we develop in this paper. The primary difference is that we use a

spherical imaging system and spherical coordinates instead of a planar image and Cartesian coordinate system. Their choice of imaging and coordinate system makes the derivation more complicated, and makes it more difficult to gain an intuition for the underlying geometric relationship between optical flow and DOH. Furthermore, since the planar imaging geometry does not treat all directions homogeneously (the optical axis is special), the Jepson/Heeger algorithm biases the DOH estimate towards the optical axis.

The spherical geometry we employ overcomes these difficulties.[1] Spherical projection has the advantage of treating all viewing directions homogeneously, and it allows us to write the optical flow in a particularly simple form. A planar perspective image can be directly converted into an image obtained under spherical projection and vice versa, assuming a calibrated camera (i.e., known focal length and optical axis).

We present two linear solutions for estimation of translation direction in this paper. The first solution is extremely simple and efficient but requires optical flow measurements at points diametrically across from each other on the spherical imaging surface.[2] The second algorithm has no such constraints on the location of image points, and can utilize a spherically imaged camera or a calibrated standard camera. In either case, there is a class of surfaces for which the algorithm will fail; we explicitly parameterize these surface classes. Finally, we test the performance of the complete SFM algorithm experimentally.

2 Optical Flow on a Sphere

Let us assume that the visible world is projected onto a sphere, and that the surface of the sphere is our image. Such a projection of the world onto a sphere is known

[1] Spherical projection is a camera model used rarely in SFM research. An exception is that of Nelson and Aloimonos [5].

[2] This requires, ideally, a 360° field–of–view imager, and at the very least a few points in opposite directions. However, this situation can be approximated by two standard cameras mounted back to back.

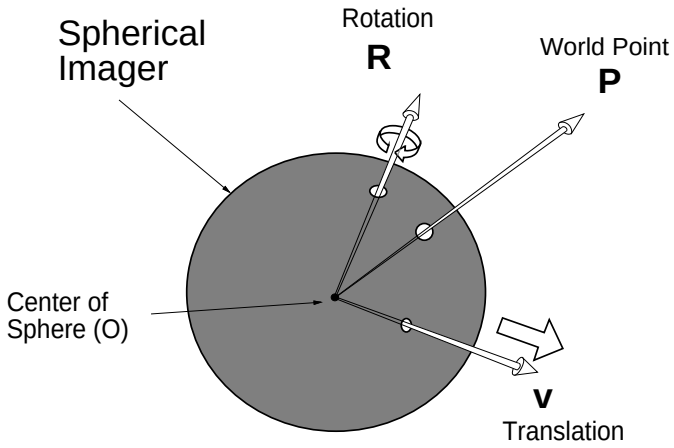


Figure 1: The geometry of structure from motion under spherical projection

as *Spherical Projection* in geometry [1]. A typical CCD camera obtains an image equivalent to a portion of such a sphere whereas the retina of the human eye is approximately equivalent to half of the spherical imager. Although we explore a model involving the entire sphere, the solutions that we will propose will only require portions of the sphere. Figure 1 depicts the spherical imager and the projection of a single 3D point P on the sphere.

For any monocular observer all visual information is obtained from the imager, in our case the surface of the sphere. When the observer moves, the image on the sphere typically changes. The goal of structure–from–motion algorithms is to determine the observers motion and the location of points in the world based on these changes. For the purposes of this work we will assume that the world is rigid, resulting in a single relative motion between the observer and the world. However, the observer can arbitrarily rotate and move (translate) during motion; an example of a rotation axis and translation direction are shown in Figure 1. The distances of points in the world (e.g. P) are reckoned from the center of the sphere (O).

The instantaneous change in the image due to observer motion is referred to as *Optical Flow*. It can be decomposed into a change due to translation and a change due to rotation. Let us represent the optical flow of the ith 3D point P (cf. Figure 1)

as a vector $\vec{\mathbf{f}}_i$ on the surface of the sphere:

$$\vec{\mathbf{f}}_i \;=\; \left[\frac{1}{p_i}\,(\hat{\mathbf{v}} \times \hat{\mathbf{p}}_i) \times \hat{\mathbf{p}}_i\right] \;+\; \left[\vec{\mathbf{R}} \times \hat{\mathbf{p}}_i\right] \tag{1}$$

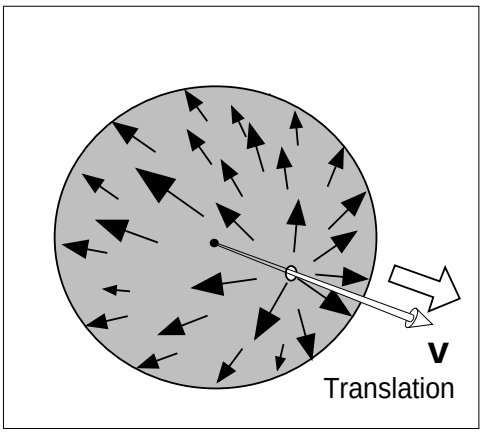
The first term corresponds to the translational component, where $\hat{\mathbf{v}}$ is the direction of translation of the observer, and $\vec{\mathbf{p}}_i$ is the vector from the center of the sphere to the point P in the world (p_i is the distance to P from O, and $\hat{\mathbf{p}}_i$ is the unit direction to P from O). The second term is the rotational component; the direction of $\vec{\mathbf{R}}$ is the axis of rotation and the magnitude of $\vec{\mathbf{R}}$ is the angle of rotation. Note that the rotational flow ($\vec{\mathbf{R}} \times \hat{\mathbf{p}}_i$) is not dependent on the distance from the point P to the observer, whereas the distance (p_i) *does* appear in the translational flow. Furthermore, only the *direction* of translation ($\vec{\mathbf{v}}$) matters in the translational flow, since a rescaling of the translation vector accompanied by an equivalent rescaling of the distances p_i produces no change in the optical flow.[3]

Figure 2 exemplifies the translational flow, the rotational flow, and the sum of the two flows, across the sphere. The undecomposed flow (i.e. the full optical flow) forms the input to the algorithms developed here, since the components of the flow are not independently measurable. Our goal is to compute the direction of translation ($\hat{\mathbf{v}}$), the rotation ($\vec{\mathbf{R}}$), and the distance (p_i) from this input.

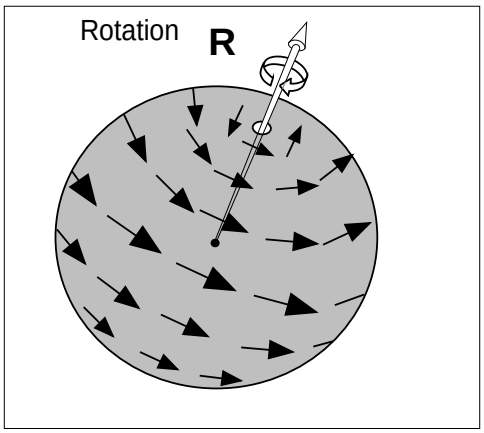
3 A Linear Algorithm for Rotation and Distances

In the following we show that given the translation direction, finding rotation is a simple linear operation. This result has also been shown by Heeger & Jepson [2]. In Section 4, we develop linear algorithms for computing translation direction.

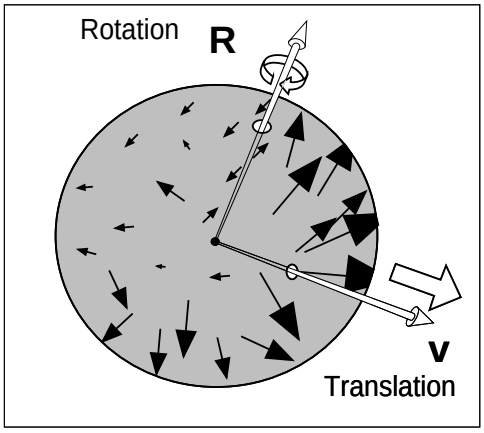
[3]Since the magnitude of translation cannot be obtained from the information on the image, this magnitude $\|\vec{\mathbf{v}}\|$ has been set to a constant (1) in the equation. For example, the flow generated by a certain translational magnitude is the same as the flow generated while moving twice as fast but in a new world which is identical in all respects other than that the points from the observer are twice as far. If either the magnitude of translation or the scale of translation is known independently, the 3D model of the world could be scaled up or down appropriately.



(i) Translational Flow



(ii) Rotational Flow



(iii) Full Optical Flow = (i) + (ii)

Figure 2: Spherical Optical Flow. Translational flow is along longitudes emanating from the focus of expansion; the magnitude of translational flow depends on how far the point is from the observer. Rotational flow is along latitudes around the axis of rotation. Finally, the complete flow at any point is the sum of the translational flow and rotational flow; there usually is no obviously noticeable pattern in the flow field.

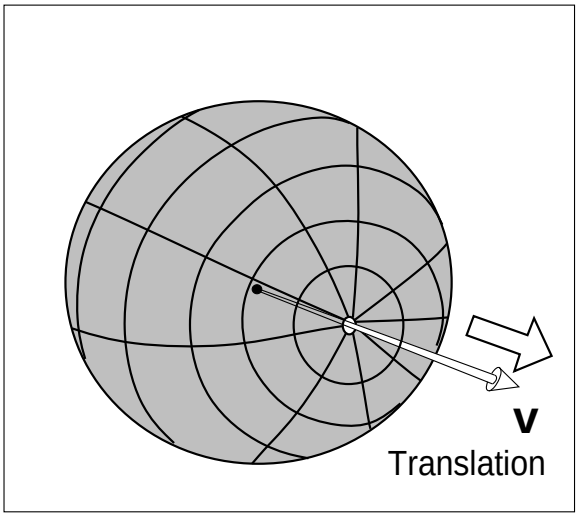


Figure 3: The vectors $(\hat{\mathbf{v}} \times \hat{\mathbf{p}}_i) \times \hat{\mathbf{p}}_i$ lie along the longitudes whereas the vectors $\hat{\mathbf{v}} \times \hat{\mathbf{p}}_i$ lie along the latitudes.

In order to find the rotation, we must isolate the flow due to rotation from the full optical flow by removing the translational flow. This would be straightforward if the distances to the points in the world *and* the translational direction were known: the vectors in Figure 2(i) could be subtracted from those in Figure 2(iii), resulting in the vectors in Figure 2(ii) – the rotational flow. However, we will show that rotation can be determined if we know only the translational direction.

A given translational direction uniquely specifies the direction of translational flow at any given point, as shown by the radial vectors in Figure 2(i). That is, in Equation (1) the vector $\hat{\mathbf{v}}$ specifies $(\hat{\mathbf{v}} \times \hat{\mathbf{p}}_i) \times \hat{\mathbf{p}}_i$, which lie along the longitudes in Figure 3. This means that the flow along the latitudes in Figure 3, perpendicular to the direction of translational flow, is not only independent of distances to points in the world but is purely due to rotation.[4]

The component of the full optical flow along the latitudes in Figure 3 is obtained by projecting the optical flow at any point (given in Equation (1)) onto the vector

[4]Note that this observation does not mean that all the rotational flow is orthogonal to the translational direction; typically there will be some component of rotation in the direction of the translational flow. We are merely separating out this component of flow, which is independent of rotation.

$\hat{\mathbf{v}} \times \hat{\mathbf{p}}_i$:

$$\vec{\mathbf{f}}_i \cdot (\hat{\mathbf{v}} \times \hat{\mathbf{p}}_i) = \mathbf{0} + \left[(\vec{\mathbf{R}} \times \hat{\mathbf{p}}_\mathbf{i}) \cdot (\hat{\mathbf{v}} \times \hat{\mathbf{p}}_\mathbf{i}) \right] \quad (2)$$

$$= \vec{\mathbf{R}} \cdot (\hat{\mathbf{p}}_i \times (\hat{\mathbf{v}} \times \hat{\mathbf{p}}_i)) \quad (3)$$

Note that the first term in Equation (1) is orthogonal to $\hat{\mathbf{v}} \times \hat{\mathbf{p}}_i$ and therefore its projection onto $\hat{\mathbf{v}} \times \hat{\mathbf{p}}_i$ is $\mathbf{0}$; the rewriting in Equation (3) follows from simple vector algebra.

Given the optical flow for a point $\hat{\mathbf{p}}_i$ on the image sphere, and given the value of the translation direction $\hat{\mathbf{v}}$ (to be calculated as in Section 4), we can denote the left hand side of Equation (3) as a new measurement, say m_i. Similarly, the quantity $\hat{\mathbf{p}}_i \times (\hat{\mathbf{v}} \times \hat{\mathbf{p}}_i)$ will be denoted by $\vec{\mathbf{n}}_i$. Equation (3) then can be written as:

$$\vec{\mathbf{R}} \cdot \vec{\mathbf{n}}_i = m_i \quad (4)$$

i.e.

$$\vec{\mathbf{R}} \cdot \vec{\mathbf{n}}_i - m_i = 0 \quad (5)$$

or

$$\begin{pmatrix} \vec{\mathbf{R}} & 1 \end{pmatrix} \begin{pmatrix} \vec{\mathbf{n}}_i \\ m_i \end{pmatrix} = 0 \quad (6)$$

Equation (6) relates $\vec{\mathbf{R}}$ to the calculatable entities $\vec{\mathbf{n}}_i$ and m_i for each point where optical flow is measured, resulting in one equation for every value of flow. We may estimate the rotation $\vec{\mathbf{R}}$ by solving this set of linear equations.[5]

Once the rotation and translation direction have been determined, it is straightforward to calculate the distance to any 3D point whose optical flow is known by rewriting Equation (1):

$$p_i = \frac{\|\vec{\mathbf{f}}_i - \vec{\mathbf{R}} \times \hat{\mathbf{p}}_i\|}{\|(\hat{\mathbf{v}} \times \hat{\mathbf{p}}_i) \times \hat{\mathbf{p}}_i\|} \quad (7)$$

[5]Denoting $\vec{Q} = \begin{pmatrix} \vec{\mathbf{R}} \\ 1 \end{pmatrix}$ and $\vec{\mathbf{s}}_i = \begin{pmatrix} \vec{\mathbf{n}}_i \\ m_i \end{pmatrix}$, Equation (6) reduces to the form of a plane through the origin (albeit in four dimensions): $\vec{\mathbf{Q}} \cdot \vec{\mathbf{s}}_i = 0$. If a matrix S is formed from the set of calculated vectors $\vec{\mathbf{s}}_i$, then $\vec{\mathbf{Q}}$ is obtained by finding the minimal-eigenvalue eigenvector of the matrix $S^T S$.

We have shown that instantaneous rotation and distances to points in the world can be extracted from optical flow without resorting to non–linear optimization techniques. Provided that translation direction can be calculated linearly – as we will show — we will have a complete linear SFM algorithm.

4 Linear Computation of Translation

We now turn to a description of computing translation direction. Our observation is that there is a simple linear operation which transforms optical flow into a new representation wherein the linearities of translational flow become intuitively apparent. We term the new representation *angular flow* and denote it by $\vec{\mathbf{a}}$.

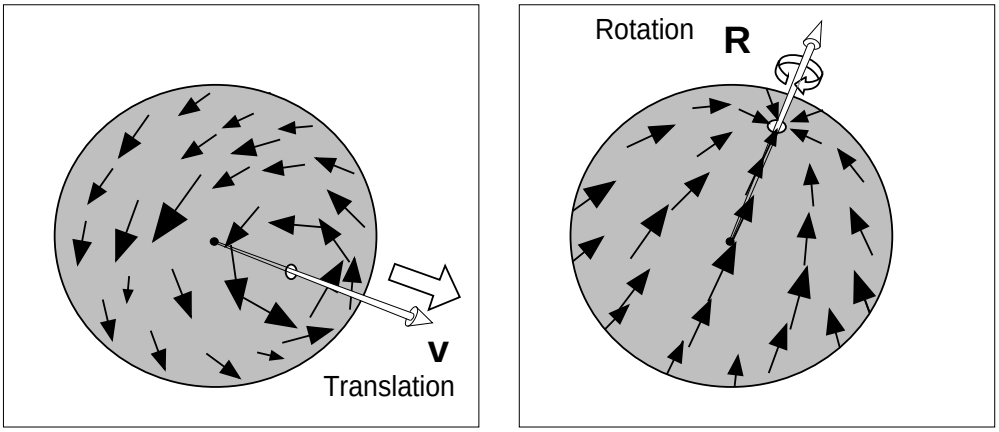
Angular flow at any given point $\hat{\mathbf{p}}_i$ on the surface of the sphere is obtained by taking the cross product of known optical flow at the point ($\vec{\mathbf{f}}_i$ in Equation (1)) with $\hat{\mathbf{p}}_i$ itself:

$$\vec{\mathbf{a}}_i = \left[\frac{1}{p_i}\left((\hat{\mathbf{v}} \times \hat{\mathbf{p}}_i) \times \hat{\mathbf{p}}_i\right) \times \hat{\mathbf{p}}_i\right] + \left[(\vec{\mathbf{R}} \times \hat{\mathbf{p}}_i) \times \hat{\mathbf{p}}_i\right] \quad (8)$$

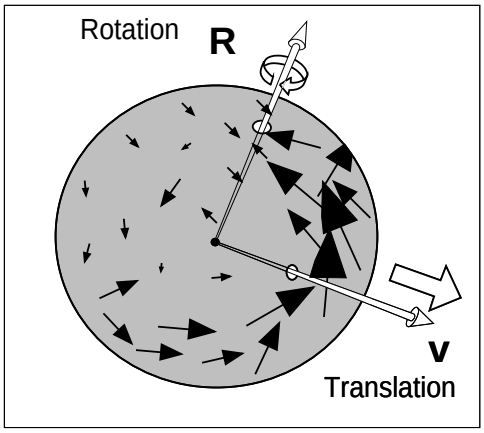
$$= \left[\frac{1}{p_i}\hat{\mathbf{v}} \times \hat{\mathbf{p}}_i\right] + \left[(\vec{\mathbf{R}} \times \hat{\mathbf{p}}_i) \times \hat{\mathbf{p}}_i\right] \quad (9)$$

The first term in Equation (8) simplifies, since taking the cross–product of a vector ($\hat{\mathbf{v}}$) three times with the unit vector ($\hat{\mathbf{p}}_i$) is equivalent to taking the cross–product just once. The angular flow vector for any point $\hat{\mathbf{p}}_i$ acts as an axis passing through the center of the sphere (O) about which $\hat{\mathbf{p}}_i$ rotates in order to generate the optical flow $\vec{\mathbf{f}}_i$ on the surface of the sphere. This representation of angular flow was first introduced in [10]. Note that the angular flow of a point can be readily obtained from the optical flow of the point, and vice versa.

Figure 4 exemplifies the angular flow which is obtained from the optical flow shown in Figure 2. The translational component of angular flow falls on latitude lines around an axis in the direction of translation, whereas the rotational component of



(i) Translational Angular Flow (ii) Rotational Angular Flow



(iii) Full Angular Flow = (i) + (ii)

Figure 4: Angular Flow. This new representation is merely a linear transformation of the flow in Figure 2. However, in this case it is the translational component that is along latitude-like rings, while the rotational component is along longitude–like great circles – structures that are opposite to what is noticed with optical flow.

angular flow lies on longitudes emanating from the point on the sphere pierced by the rotation axis. Note the striking reversal of symmetry between Figures 2 and 4; in Figure 2 the rotational flow falls along *latitudes*, whereas the translational flow lies along *longitudes*. That is, the patterns of the curves of translational and rotational flow are the opposite of each other.

Now consider the relationship between the angular flow vectors and the translation direction, $\hat{\mathbf{v}}$. Equation (9) shows that the translational component of angular flow (first term in the equation) is always orthogonal to $\hat{\mathbf{v}}$. The entire set of translational

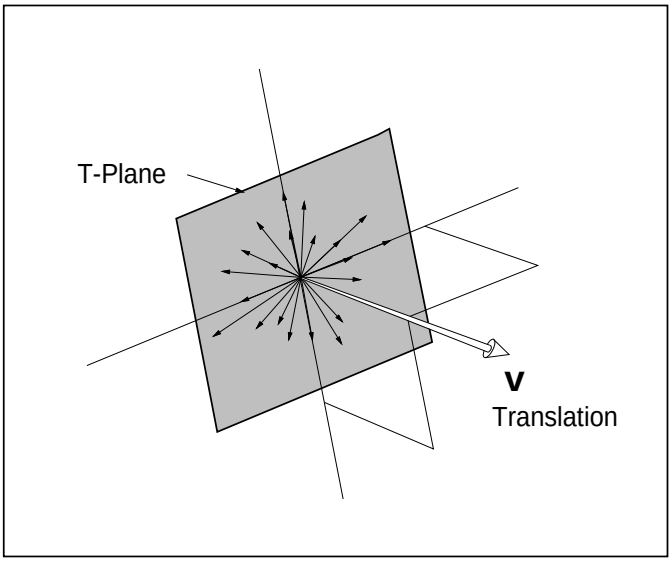


Figure 5: The translational component of an angular flow vector always lies on the T-plane, normal to the direction of translation.

angular flow vectors lies on a plane orthogonal to the direction of translation $\hat{\mathbf{v}}$. Furthermore, any linear combination of these vectors will lie on the same plane. We shall henceforth refer to this plane as the *T-plane* (cf. Figure 5).

In practice the rotational component is not zero, hence resulting in angular flow vectors that need not lie on the T-plane. Our goal is to compute linear combinations of the angular flow vectors such that contain *no rotational component.* As we have stated, such linear combinations will still remain on the T-plane, and we can apply simple eigenvalue techniques to estimate the translation direction. The goal of the following sections is to describe two simple linear operators that combine angular flow vectors so as to annihilate the rotational component of flow.

4.1 Linear Solution I: Antipodal Pairs of Points

The crucial observation for this solution is that the rotational component of angular flow is the same at opposite points of the sphere. This is confirmed first by intuition; for example, a point directly in front and a point directly at the back of you will move in the image by the same amount for any given rotation. The intuition is also

verified by a straightforward examination of the second term in Equation (9): a pair of opposite (or antipodal) points $(\hat{\mathbf{p}}_i, -\hat{\mathbf{p}}_i)$ have the identical rotational angular flow. Therefore, subtracting the angular flows of any two antipodal points results in a set of vectors $\vec{\mathbf{d}}_i$ that lie in the T-plane, but are independent of rotation $\vec{R}$.

Thus, we have established a linear operation (subtraction of antipodal angular flows) which when applied to optical flow, cancels out the effect of rotation. Solving for the translation direction then reduces to finding the normal to the plane containing the vectors $\vec{\mathbf{d}}_i$.[6]

There are two problems with this approach. The first problem is that the number of constraints is only half of the number of measurements. This large reduction in dimensionality is being used to eliminate just three rotation parameters. This means that we are not using the full set of constraints available to us, and it suggests that there may be surface configurations that will "fool" the algorithm.

Inspection reveals an example of this: the algorithm will not work if the antipodal points are equally far away from the observer; in such a case $\vec{\mathbf{d}}_i$ is zero everywhere and the T–plane cannot be established. This would occur, for example, when the observer is traveling down a hallway looking directly ahead.

The second problem is that measuring flow at antipodal points necessitates the use of hardware that can obtain images of the world at antipodal points. Although such imaging hardware is closely approximated in animals having antipodally arranged eyes (e.g., rabbits), such artifical imaging systems are not readily available.

4.2 Linear Solution II: Arbitrary Sets of Points

In this section we consider more general linear combinations of angular flows of an arbitrary set of points such that the resulting vectors lie on the T–plane, and the rotational flow has been removed.

[6] If D is a matrix of the measurable $\vec{\mathbf{d}}_i$ then an estimate for $\hat{\mathbf{v}}$ is the minimal-eigenvalue eigenvector of $D\,D^T$, which is a linear computation [7].

Consider an n-dimensional weighting vector, $\vec{w}$, with components w_i. The weighted sum of the angular flows is then:

$$\sum_i^n w_i \, \vec{\mathbf{a}}_i = \sum_i^n w_i \left(\frac{1}{p_i} \hat{\mathbf{v}} \times \hat{\mathbf{p}}_i \right) + \sum_i^n w_i \, (\vec{\mathbf{R}} \times \hat{\mathbf{p}}_i) \times \hat{\mathbf{p}}_i \tag{10}$$

If we denote the unit vector $\hat{\mathbf{p}}_i$ by (x_i, y_i, z_i) in an arbitrary Cartesian coordinate system, we can expand the rotational term of Equation (10) using matrix notation and rewrite the equation as:

$$\sum_i^n w_i \, \vec{\mathbf{a}}_i = \sum_i^n w_i \left(\frac{1}{p_i} \hat{\mathbf{v}} \times \hat{\mathbf{p}}_i \right) + \sum_i^n w_i \begin{pmatrix} 1 - x_i^2 & -x_i y_i & -x_i z_i \\ -x_i y_i & 1 - y_i^2 & -y_i z_i \\ -x_i z_i & -y_i z_i & 1 - z_i^2 \end{pmatrix} \vec{\mathbf{R}} \tag{11}$$

or

$$\sum_i^n w_i \, \vec{\mathbf{a}}_i = \sum_i^n w_i \left(\frac{1}{p_i} \hat{\mathbf{v}} \times \hat{\mathbf{p}}_i \right) + \begin{pmatrix} \sum_i^n w_i(1 - x_i^2) & -\sum_i^n w_i x_i y_i & -\sum_i^n w_i x_i z_i \\ -\sum_i^n w_i x_i y_i & \sum_i^n w_i(1 - y_i^2) & -\sum_i^n w_i y_i z_i \\ -\sum_i^n w_i x_i z_i & -\sum_i^n w_i y_i z_i & \sum_i^n w_i(1 - z_i^2) \end{pmatrix} \vec{\mathbf{R}} \tag{12}$$

The goal then is to find weights such that the rotational component of the combination is zero independent of the value of $\vec{\mathbf{R}}$. When the rotational component of Equation (12) vanishes, the remaining translational component consists of $\sum_i^n w_i \; {}^1\!/_{p_i} \hat{\mathbf{v}} \times \hat{\mathbf{p}}_i$, which continues to lie on the T–plane. Note that only six of the nine terms in the matrix of Equation (12) are unique. In order for the rotational flow to be zero for any value of $\vec{\mathbf{R}}$, each of the six unique terms must be zero. This gives rise to six equations:

$$\begin{aligned} \sum_i^n w_i(1 - x_i^2) = 0 \qquad & \sum_i^n w_i x_i y_i = 0 \\ \sum_i^n w_i(1 - y_i^2) = 0 \qquad & \sum_i^n w_i x_i z_i = 0 \\ \sum_i^n w_i(1 - z_i^2) = 0 \qquad & \sum_i^n w_i y_i z_i = 0 \end{aligned} \tag{13}$$

In the above equations, x_i, y_i and z_i are known. The only unknowns are the n weights w_i.

Since there six linear equations for n unknowns, we should be able to find at least $n-6$ linearly independent solutions. Any value of $\vec{\mathbf{w}}$ that satisfies the following equation is a valid solution:[7]

$$\vec{\mathbf{w}}^T \begin{pmatrix} 1 & x_1^2 & y_1^2 & x_1 y_1 & x_1 z_1 & y_1 z_1 \\ 1 & x_2^2 & y_2^2 & x_2 y_2 & x_2 z_2 & y_2 z_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n^2 & y_n^2 & x_n y_n & x_n z_n & y_n z_n \end{pmatrix} = 0 \qquad (14)$$

Solving for the different solution sets of weights $\vec{\mathbf{w}}$ reduces to solving this linear system of equations. The solutions constitute the "left nullspace" of the above $n \times 6$ matrix. In general, there are $n-6$ solutions for $\vec{\mathbf{w}}$ since the rank of the matrix in Equation (14) is 6. This follows from the fact that the quadratic terms in general span a 6 dimensional space, and the remaining dimensions of the above matrix $(n-6)$ constitute the dimensions of its left null space.

Each solution of weights $\vec{\mathbf{w}}$ gives rise to a single new vector that lies on the T–plane. Finding the translation direction now reduces to finding the vector orthogonal to these vectors.[8]. This means that we now have a fully linear solution for direction of heading $\hat{\mathbf{v}}$, for an arbitrary set of points at which angular flow is measured. We can now apply this calculated value of translation to the previously described linear algorithm to determine rotation and subsequently the distances to points in the world.

[7]Equation (14) contains the same information as the six equations in 13, and makes use of the fact that $z_i^2 = 1 - x_i^2 - y_i^2$, since $\hat{\mathbf{p}}_i$ is a unit vector.

[8]If we represent the set of $n-6$ new 3D vectors by a single $(n-6) \times 3$ matrix C then the direction of translation $\hat{\mathbf{v}}$ is the null space of the 3×3 matrix $C^T C$.

5 Analysis of the Linear Algorithm for Translation

Now that we have a linear algorithm for arbitrary sets of points, we will consider two important issues that concern its performance.

1. Problematic Surfaces. Although the rotation parameter ($\vec{\mathbf{R}}$) is only a three-dimensional entity, the linear algorithm must annihilate a six-dimensional subspace (cf. Equation (14)). This linear annihilation operation therefore eliminates some portion of angular flow that is due to translational motion. It is important to determine the class of surfaces for which the translational motion component could be completely removed; such surfaces will be parameterized in this section.

2. Robustness to noise. If the optical flow measurements are noisy, how does this effect the estimates of translation direction? In particular, an important question is whether the estimated translation is unbiased. In this paper we touch on these issues on briefly. In future work, we would like to express the uncertainty of the translation estimate as a function of the covariance of error in the optical flow estimates.

5.1 Problematic Surfaces

Recall that the linear algorithm computes linear combinations of angular flow vectors with a weighting $\vec{\mathbf{w}}$ that obeys Equation (14). The translational component of angular flow from Equation (9) is:

$$\vec{\mathbf{a}}_i^v = \frac{1}{p_i}\hat{\mathbf{v}} \times \hat{\mathbf{p}}_i \tag{15}$$

We wish to determine the distances to points in the world, p_i, such that $\vec{\mathbf{w}} \cdot \vec{\mathbf{a}}^v$ is zero. In these situations, the translational flow will be canceled by the same

operation which removes rotational angular flow, thereby preventing the calculation of the translational direction $\hat{\mathbf{v}}$. Since the numerator on the right side of Equation (15) contains only *linear* components of $\hat{\mathbf{p}}_i$, i.e. (x_i, y_i, z_i) and since the weighting vector $\vec{\mathbf{w}}$ is constructed to cancel *quadratic* components of $\hat{\mathbf{p}}$, the translational flow will be canceled when the distance to the points is:

$$\begin{aligned} p_i &= \frac{1}{\alpha_x\ x_i + \alpha_y\ y_i + \alpha_z z_i} \\ &= \frac{1}{\vec{\alpha} \cdot \hat{\mathbf{p}}} \end{aligned} \tag{16}$$

where $\vec{\alpha} = (\alpha_x, \alpha_y, \alpha_z)$, is any fixed vector in space.

Hence, the equation that defines the problematic surfaces is now:

$$\begin{aligned} \vec{\mathbf{p}} &= p_i\ \hat{\mathbf{p}} \\ &= \frac{\hat{\mathbf{p}}}{\vec{\alpha} \cdot \hat{\mathbf{p}}} \end{aligned} \tag{17}$$

This class of surfaces is easier for us to interpret when written as a standard "Z-map" under planar perspective projection. Let the planar image coordinates be (a, b), and the associated depth map $Z(a, b)$. Then we can relate the spherical and planar imaging geometries via the equation $\hat{\mathbf{p}} = \frac{(a,b,1)}{\|(a,b,1)\|}$ (we assume, without loss of generality, a unit focal length). We can now write an expression for the depth map of the singular surfaces:

$$\begin{aligned} Z(a,b) &= \vec{\mathbf{p}}_z \\ &= \frac{\hat{\mathbf{p}}_z}{\vec{\alpha} \cdot \hat{\mathbf{p}}} \\ &= \frac{1}{\vec{\alpha} \cdot (a, b, 1)}. \end{aligned}$$

Fortunately, most such surfaces will generally cause a problem only *instantaneously*, except for the case of a plane perpendicular to the translation direction

$(\vec{\alpha} = (0, 0, 1))$.

5.2 Effect of Noise

Now we address the second question: how is the algorithm affected by noise? Jepson and Heeger [3] have answered this question in the case of a planar imager. They found that their algorithm determined a translation direction that was *biased* toward the optical axis. They were able to eliminate this bias by adding a noise vector perpendicular to the imaging plane to each of the flow vectors, with variance equal to the noise already present in the flow estimate.

The algorithm we have described is unbiased if (a) the field–of–view is 180° or (b) if the noise in the flow is 3D symmetric Gaussian. However, this type of noisy input cannot be expected. Typically, when optical flow is computed on the sphere, the errors lie on the tangent plane to the sphere at that point. At best, we can try to estimate the variance of these errors in the tangent plane (see e.g. [8]). However, if we keep a full viewing angle of 180°, then there is no bias in the estimated translation direction, even if the input noise were not symmetrical.

The algorithm is no longer unbiased if the viewing angle is less than 180°. For example, if we use a viewing angle of 90°, the error in the flow vectors would all lie within 45° of the optical axis. Since the algorithm seeks a plane perpendicular to the flow vectors, this will bias the estimate toward the optical axis. Thus, our linear algorithm is a *biased* one when used with a restricted viewing angle, and an *unbiased* one with a 180° degree field–of–view imager.

6 Experiments

6.1 Simulations

We first tested the algorithm on simulated angular flow fields. We (i) chose a viewing angle, (ii) generated a random collection of viewing directions $\hat{\mathbf{p}}_i$ lying within this

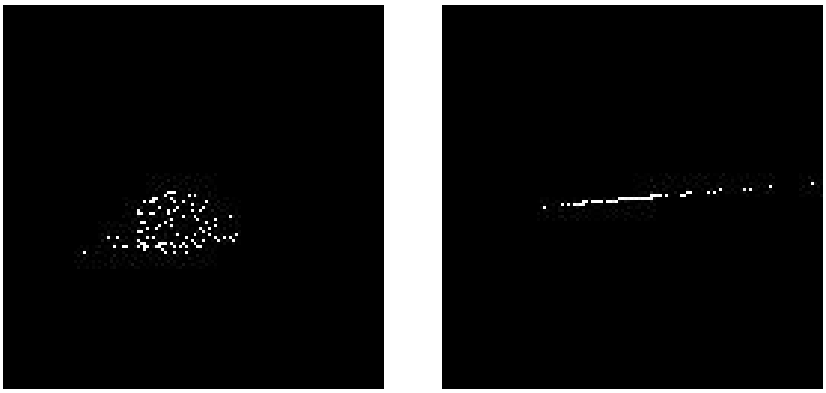


Figure 6: Projection of the angular flow vectors onto a plane whose normal is perpendicular to the true direction of translation. At the left is full angular flow, including both a rotation and translation component. On the right are the angular flow vectors after annihilation of the rotational flow. These vectors lie on a plane in 3D whose normal is the correct direction of translation.

viewing angle, (iii) generated a set of random distances, p_i, where $1/p_i$ was uniformly distributed in the interval $[0, 1/3f]$ (f being the camera's focal length). In this noise-free case, the algorithm works accurately as expected. Figure 6 illustrates the linear operation that removes the rotational component of flow.

Next, we added 3D Gaussian random noise with standard deviation $\sigma = 0.2$ in each direction to the flow vectors. These noise vectors were then re-projected back onto the imaging sphere (that is, the component in the $\hat{\mathbf{p}}_i$ direction was removed). We tried this for a full viewing angle (180°) and a limited viewing angle (60°). As expected, the estimates for the full viewing angle are unbiased, whereas the estimates in the restricted view are biased toward the optical axis. This bias in the result for a limited viewing angle is illustrated in figure 7.

6.2 Realistic Image Sequence

We ran the algorithm on the "Yosemite fly-through" image sequence. This sequence was generated by re-projecting real imagery onto a real depth map. One frame of the sequence is shown in figure 8. The viewing angle is approximately 45°. When we used the algorithm with the correct angular flow[9], the estimate was within 1.8° of the

[9]The angular flow can be computed, since the camera motion and depth map of the sequence are known.

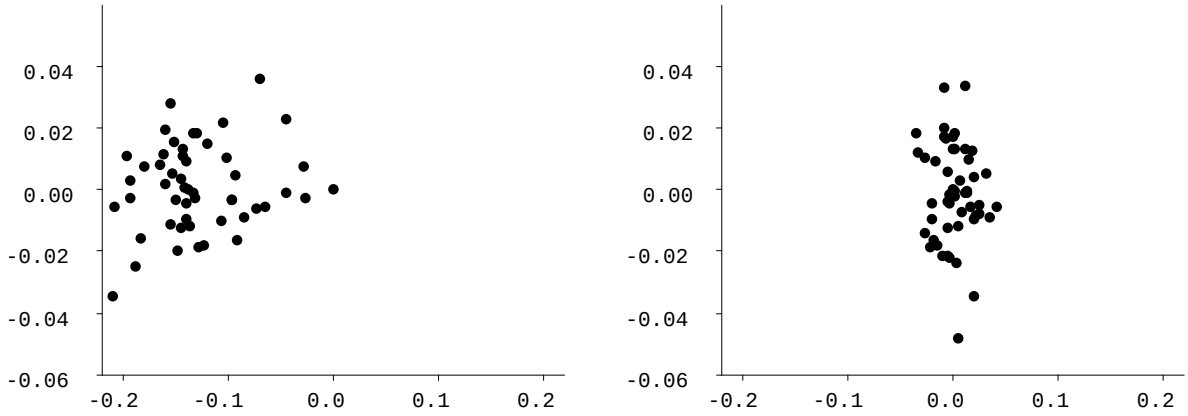


Figure 7: Scatter plot of the $x-$ and $y-$ components of error in the estimated translation direction. The coordinate system has been rotated so that the correct translation is at $(0,0)$, and the optical axis is at $x = -0.67$. The viewing angle for the left plot is 60°, and the viewing angle for the right plot is 180°. We generated angular flow vectors for a fixed translation, rotation, and randomized depth map. We added Gaussian white noise to these flow vectors, projecting out the component in the $\hat{\mathbf{p}}_i$ direction, and computed an estimate of translation. Shown are 50 such estimates. Notice that the estimates for the smaller field of view are *biased* toward the optical axis. The full field of view produces an unbiased set of estimates.

correct direction of translation.

We also tested the algorithm with optical flow recovered as in [8]. As expected, this produced an estimate biased toward the optical axis, due to the restricted viewing angle.

7 Conclusion

We have described a linear algorithm for recovering egomotion and depth from image flow. The algorithm is developed in the context of a spherical imaging geometry. We have discussed two types of failure of the algorithm: 1) a class of surfaces in the world for which the algorithm cannot recover translational motion, and 2) a bias in the translation estimate for restricted viewing angles. The first of these is not a serious problem because these surfaces occur rarely, and even when they do occur, the motion of the camera ensures that they occur only momentarily. The second problem only occurs for restricted viewing angles: a hemispherical imaging surface will produce unbiased estimates.

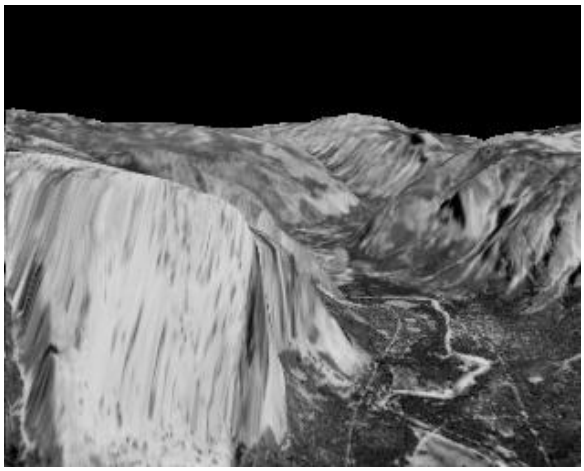


Figure 8: One frame of the "Yosemite fly-through" sequence. This sequence was generated by Lyn Quam at SRI.

Acknowledgments. We wish to acknowledge Prof. R. Bajcsy for her comments on this research, and Dr. A. Vainikka for her criticsms on the writing. We also wish to acknowledge the following grants: ARPA Grant N00014-92-J-1647, ARO Grant DAAL03-89-C-0031PRI, NSF Grant CISE/CDA-88-22719, and NSF Grant STC SBR8920230.

References

[1] M. P. Do Carmo. *Differential Geometry of Curves and Surfaces.* Prentice–Hall, Inc., Englewood Cliffs, New Jersey, 1976.

[2] D. Heeger and A. Jepson. Subspace methods for recovering rigid motion i: Algorithm and implementation. *IJCV*, 7(2):95–117, 1992.

[3] A. Jepson and D. Heeger. *Linear subspace methods for Recovering translational direction.* Cambridge University Press, 1992.

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

[5] R. Nelson and J. Aloimonos. Finding motion parameters from spherical motion fields (or the advantages of having eyes in the back of your head). *Biological Cybernetics*, 58:261–273, 1988.

[6] K. Prazdny. On the information in optical flow. *Computer Vision, Graphics and Image Processing*, 22:239–259, 1983.

[7] W. H. Press, B. P. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C, The Art of Scientific Computing.* Press Syndicate of the University of Cambridge, 1988.

[8] E. P. Simoncelli, E. H. Adelson, and D. J. Heeger. Probability distributions of optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition*, Mauii, Hawaii, June 1991.

[9] I. Thomas. *Reducing Noise in 3D Models Recovered from a Sequence of 2D Images.* PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, MA, 1993.

[10] I. Thomas, E. Simoncelli, and R. Bajcsy. Spherical retinal flow for a fixating observer. In *Proceedings of the Workshop on Visual Behaviors*, pages 37–44, Seattle, WA, June 1994.

[11] C. Tomasi and T. Kanade. Factoring image sequences into shape and motion. In *Proc. IEEE workshop on visual motion*, pages 21–28, Princeton, NJ, 1991.