Comparison of Approaches to Egomotion Computation

Tina Y. Tian[1] Carlo Tomasi[2] David J. Heeger[1]
[1]Department of Psychology and [2]Computer Science Department
Stanford University, Stanford, CA 94305
email: tian@white.stanford.edu, tomasi@cs.stanford.edu, heeger@white.stanford.edu

Abstract

We evaluated six algorithms for computing egomotion from image velocities. We established benchmarks for quantifying bias and sensitivity to noise, and for quantifying the convergence properties of those algorithms that require numerical search.

Our simulation results reveal some interesting and surprising results. First, it is often written in the literature that the egomotion problem is difficult because translation (e.g., along the X-axis) and rotation (e.g., about the Y-axis) produce similar image velocities. We found, to the contrary, that the bias and sensitivity of our six algorithms are totally invariant with respect to the axis of rotation. Second, it is also believed by some that fixating helps to make the egomotion problem easier. We found, to the contrary, that fixating does not help when the noise is independent of the image velocities. Fixation does help if the noise is proportional to speed, but this is only for the trivial reason that the speeds are slower under fixation. Third, it is widely believed that increasing the field of view will yield better performance. We found, to the contrary, that this is not necessarily true.

1 Introduction

The field of computer vision has witnessed a bewildering array of approaches to the fundamental problem of computing three-dimensional camera motion from the motion measured in the image plane. But systematic, quantitative comparisons of methods for the computation of camera motion have not been reported in the computer vision literature.

Our study is done in simulation because we want to be sure about the data, and because simulation makes it trivial to change one parameter at a time. For example, we report interesting conclusions about the effect of increasing field of view. This effect is related rather subtly to image resolution. The difficulty of carrying out such an investigation with real cameras should be obvious.

Systematically spanning the Cartesian product of all parameter intervals is both infeasible and hardly enlightening, so we concentrate here on an interesting subset of our simulation results. For similar reasons, we only compare a handful of algorithms. We picked methods based on how different their underlying principles are, rather than for their popularity or for the sake of exhaustiveness. If your favorite algorithm is not listed here, we hope that you will find in our comparison one whose general computational strategy is similar. Again, our goal is insight, not ranking. As a matter of fact, perhaps the clearest conclusion from our comparison is that algorithms cannot be ordered from worst to best. All the methods we have considered have strengths in some situations and weaknesses in others.

Our code (Matlab implementations of the algorithms themselves, and Matlab implementations of the simulation code) and all of our simulation results is on the World Wide Web at URL http://white.stanford.edu. If you believe that your favorite algorithm is better in some respect, then you will be able to run the very same set of simulations for direct comparison.

2 Methods

2.1 The Problem

As a camera moves with respect to a rigid scene, the image changes over time. The goal of egomotion computation is to estimate 3d motion from a sequence of images. Techniques for computing egomotion from image sequences can be categorized either as discrete-time methods or as instantaneous-time methods, depending on whether input is image displacement or image velocity. In this paper, we concentrate on instantaneous-time algorithms.

The image velocity, due to the motion of a camera with respect to a rigid scene, and under perspective projection, is given by the following familiar equation:

$$\mathbf{u}(\mathbf{x}) = \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -x_2 \end{bmatrix} \left(\frac{\mathbf{T}}{Z(\mathbf{x})} + \mathbf{\Omega} \times \mathbf{x} \right). \qquad (1)$$

Here $\mathbf{u}(\mathbf{x})$ is the image velocity at image position $\mathbf{x} = (x_1, x_2, 1)^t$, $\mathbf{T}$ is translational velocity, $\mathbf{\Omega}$ is rotational velocity, Z is depth, and the focal length is taken (without loss of generality) to be 1. The egomotion problem is to estimate the 3d motion, $\mathbf{T}$ and $\mathbf{\Omega}$, from a collection of velocity vectors sampled at some (perhaps all) image positions.

2.2 The Algorithms

Bruss and Horn: Bruss and Horn [1] applied a simple algebraic manipulation to remove depth from Eqn.1 and obtained a bilinear constraint on $\mathbf{T}$ and $\mathbf{\Omega}$ for each image pixel. Later MacLean and Jepson [10] derived exactly the same bilinear constraint (by applying a different algebraic manipulation), and expressed it as follows:

$$\mathbf{T}^t(\mathbf{x} \times \mathbf{u}(\mathbf{x})) + (\mathbf{T} \times \mathbf{x})^t(\mathbf{x} \times \mathbf{\Omega}) = 0 \qquad (2)$$

We chose the following method for using the bilinear constraint to estimate $\mathbf{T}$. From the bilinear constraint, a least-squares estimate of rotation can be obtained as a function of translation $\mathbf{T}$. Substituting this rotation estimate back into the bilinear constraint gives a nonlinear constraint on translation, $\mathbf{T}$. We estimated translation by minimizing this nonlinear constraint over all image velocities subject to $|\mathbf{T}| = 1$.

As we will see, this algorithm performed quite well in many of our simulations. However, a disadvantage of this algorithm is that it requires numerical optimization.

Jepson and Heeger: Rieger and Lawton [16] proposed a method based on motion parallax. If two 3d points have the same image location but are at different depths, then the vector difference between the two flow vectors is oriented toward the focus of expansion (FOE). The Rieger-Lawton algorithm locates the FOE from the local flow-vector differences. Hildreth [5] later modified the Rieger-Lawton algorithm to improve its performance. But a problem with both of these algorithms is that it is particularly difficult to measure flow vectors near occlusion boundaries.

Motion parallax is, fortunately, more general than the constraint used by Rieger and Lawton. Prazdny [15], for example, noted that the difference between any two (not necessarily adjacent) flow vectors gives a constraint on translation, independent of rotation.

Jepson and Heeger built upon these previous efforts and proposed a series of subspace methods for estimating egomotion [4, 6, 7]. The simplest of these is the so-called linear subspace method [6, 7]. Given optical flow sampled at N discrete points in the image, $\mathbf{x}^k$, $k = 1 \cdots N$, one can construct a set of constraint vectors, τ_i:

$$\tau_i = \sum_{k=1}^{N} c_{ik}[\mathbf{u}(\mathbf{x}^k) \times \mathbf{x}^k] \tag{3}$$

such that the τ_i vectors are orthogonal to $\mathbf{T}$, i.e., $\tau_i \cdot \mathbf{T} = 0$. The trick is to choose $\mathbf{c}_i = [c_{i1} \cdots c_{iN}]^t$ to be orthogonal to all quadratic polynomials of x_1^k and x_2^k. This choice for the $\mathbf{c}_i$ vectors effectively annihilates the rotational component of the image velocities. For N image velocity samples, there are $N - 6$ τ_i constraint vectors. The estimate of $\mathbf{T}$ is the eigenvector corresponding to the smallest eigenvalue of $\sum \tau_i \tau_i^t$.

The advantage of the linear subspace method is that $\mathbf{T}$ is computed directly without requiring iterative numerical optimization. The disadvantage is that this method does not make use of all of the available information ($N - 6$ linear constraints versus N bilinear constraints).

Tomasi and Shi: Tomasi and Shi [18] developed a method that uses motion parallax information in a rather different way. Their method estimates translation $\mathbf{T}$ from image deformations, defined as the change $\dot{\alpha}$ in the angular distance $\alpha = \arccos(\mathbf{x}^j \cdot \mathbf{x}^j)$ between pairs of image points $\mathbf{x}^i, \mathbf{x}^j$ as the camera moves.

Since the image deformations are independent of camera rotation, one can derive the following bilinear constraint on $\mathbf{T}$ and the two depth values $Z(\mathbf{x}^i), Z(\mathbf{x}^j)$:

$$\dot{\alpha} = \sin\alpha[Z(\mathbf{x}^j), Z(\mathbf{x}^i), 0][\mathbf{x}^i, \mathbf{x}^j, \mathbf{w}^{ij}]^{-T}\mathbf{T}$$

where $\mathbf{w}^{ij} = (\mathbf{x}^i \times \mathbf{x}^j)/||\mathbf{x}^i \times \mathbf{x}^j||$.

The combined bilinear constraints for a subset of all possible point pairs were minimized and solved for $\mathbf{T}$ using the variable projection method [17] on the unit sphere $|\mathbf{T}| = 1$. The minimization involves solving for 3 translation parameters and N depth parameters, where N is the number of points. When N is large, this algorithm is much more expensive than our implementation of the Bruss and Horn algorithm, which solves for only 3 parameters.

Prazdny: All of the algorithms discussed so far begin by estimating $\mathbf{T}$. Once it is known, $\mathbf{T}$ can be plugged back into Eqn.2 to estimate $\mathbf{\Omega}$. Prazdny [13] proposed an algorithm that instead estimates rotation first. From a triple of image points, the following constraint on rotation parameters (independent of translation and depth) was derived by algebraic manipulation on Eqn.1:

$$\mathbf{n}_3 \cdot (\mathbf{n}_1 \times \mathbf{n}_2) = 0$$

where $\mathbf{n}_i = (\mathbf{\Omega} \times \mathbf{s}_i + \mathbf{v}_i) \times \mathbf{s}_i$, and $\mathbf{s}_i$ and $\mathbf{v}_i$ denote image coordinate and velocity at unit spherical retina, respectively, (i = 1,2,3).

In Prazdny's original implementation, these constraints were combined locally and three third-order polynomial equations of three unknowns were solved numerically. In our implementation, we combined all of the constraints throughout the image. Each triple of points came from a triangle of the Delaunay Triangulation of all the points. We found, however, that different triangulations resulted in inconsistent estimates, so we decided to use a fixed uniform sampling grid in order to fix the triangulation. The simplex method was used to do the optimization.

Kanatani A: The so-called epipolar constraint serves as the basis for several linear discrete-time egomotion algorithms [9, 19, 3, 20]. Let $\mathbf{X}$ and $\mathbf{X}'$ be the positions of a surface point before and after a camera motion. The rigid motion constraint relates these two positions, $\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{T}$, where $\mathbf{R}$ is a rotation matrix and $\mathbf{T}$ is a translation vector. The epipolar constraint states that the vectors $(\mathbf{R}\mathbf{X})$, $\mathbf{T}$ and $\mathbf{X}'$ all lie in the same plane.

Since image velocity is the infinitesimal limit of a finite image displacement, there is an instantaneous-time version of the epipolar constraint [11, 22, 8]. Based on this instantaneous-time epipolar constraint, Zhuang, Thomas, Ahuja, and Haralick [22] proposed a linear algorithm for egomotion estimation. Kanatani [8] later reformulated the instantaneous-time epipolar constraint in terms of essential parameters and twisted flow (a rotated version of the velocity vector). Since Zhuang et al.'s algorithm and Kanatani's algorithm are equivalent to one another, we chose Kanatani's algorithm as representative for this class of algorithms.

Kanatani B: Least-squares estimates of $\mathbf{T}$ are, as we shall see, systematically biased. Kanatani [8] analyzed the statistical bias using a simple Gaussian noise model, then proposed a method (called the renormalization method) that removes the bias by automatically compensating for the unknown noise.

Summary: Altogether the six algorithms that we have chosen differ from one another in a variety of ways: algorithms that compute translation-first versus those that compute rotation-first, algorithms that do and do not require numerical optimization, algorithms that do and do not attempt to compensate for bias, and algorithms based on motion parallax versus those based on the epipolar constraint. We will concentrate on these differences between the algorithms when interpreting the simulation results.

2.3 Simulation Methods

In all of the simulations, the focal length was set to 1. All other distances and sizes were specified in units of focal lengths. A random cloud of points was placed in the simulated 3d space in front of the simulated camera. The depth range was 2 to 8 (in units of focal length). Unless otherwise stated, the entire image subtended 90 degrees of visual angle. Various combinations of translation and rotation were chosen (see below) and Eqn.1 was used to compute the image velocities.

Zero-mean Gaussian noise of various amounts was added to each component of each velocity vector. We used two noise models in which: (1) the standard deviation of the noise was independent of image velocity, or (2) the standard deviation was proportional to the average speed, averaged across the entire image and averaged across all possible depth values (since the depth values were chosen randomly). For most of the simulations, unless otherwise stated, the first (constant) noise model was used. The noise level can be specified either in units of focal length or in units of pixels. A noise level of 0.1 pixels, for example, means that the standard deviation of the Gaussian noise was 0.1 pixels in a 512×512 image.

The default translational speed was chosen as follows. We considered a camera translating along the X-axis and rotating around the Y-axis with a fixed (see below) angular velocity. We then picked the point at the center of the random depth cloud and chose the translational speed so that the camera would fixate on that point. We used this *same* translational speed even when we varied the translation direction and/or the rotation axis. The rotation rate was fixed (0.23^0/frame) in our experiments.

For most of the algorithms (except Prazdny's algorithm), we used sparse flow data (50 randomly chosen sample points). For Prazdny's algorithm, we found that different triangulations resulted in inconsistent estimates, so we decided to use a fixed uniform (7x7) sampling grid of sample points in order to fix the triangulation.

2.4 Benchmarks

We aimed to study the bias and noise-sensitivity of the estimates for noisy flow data. The bias and the sensitivity were measured as the mean and the standard deviation of the estimates over a number of trials. One-thousand trials (each with 50 velocity vectors) were performed for each simulation condition.

Translation bias was computed for each simulation condition as the angle between the true translation direction, $\mathbf{T}$, and the "average" of the 1000 estimated translation direction vectors, $\hat{\mathbf{T}}_i$. The "average" vector, $\bar{\mathbf{T}}$, was chosen to be the unit vector that was closest (measured in angular units) to the 1000 estimates. In particular, $\bar{\mathbf{T}}$ was chosen to minimize:

$$\sum_{n=1}^{N} \cos^{-1}(\hat{\mathbf{T}}_i \cdot \bar{\mathbf{T}}), \tag{4}$$

subject to $|\bar{\mathbf{T}}| = 1$, and where $N = 1000$.

Translation sensitivity was computed as:

$$\sqrt{\frac{1}{N-1} \sum_{n=1}^{N} \left[\cos^{-1}(\hat{\mathbf{T}}_i \cdot \bar{\mathbf{T}})\right]^2}, \tag{5}$$

which is the standard deviation of the distribution of angles between each estimated translation vector and the "average" vector.

To quantify rotation bias, we first computed the average of the rotation estimates. Rotation matrices, $\mathbf{R}$ and $\bar{\mathbf{R}}$, were constructed from the true solution and the average estimate, respectively. The "difference rotation" matrix was defined as: $\Delta\mathbf{R} = \mathbf{R}^t \bar{\mathbf{R}}$, that is, the rotation matrix that takes the average estimate to the true solution. This "difference rotation" matrix $\Delta\mathbf{R}$ can be characterized by an axis and an angle. We used the angle of the "difference rotation" matrix as our measure of bias. This angle is computed from $\Delta\mathbf{R}$ as:

$$\cos^{-1}\left[\frac{\mathrm{Tr}(\Delta\mathbf{R}) - 1}{2}\right], \tag{6}$$

where $\mathrm{Tr}(\Delta\mathbf{R})$ is the trace of the matrix.

Rotation sensitivity was quantified as follows. Rotation matrices, $\hat{\mathbf{R}}_i$, were constructed from each rotation estimate. A rotation matrix, $\bar{\mathbf{R}}$, was computed as above, from the average of the rotation estimates. We computed the angular difference, θ_i between $\hat{\mathbf{R}}_i$ and $\bar{\mathbf{R}}$ for each i, using the above formulae. Then rotation sensitivity was computed as:

$$\sqrt{\frac{1}{N-1} \sum_{n=1}^{N} \theta_i^2}, \tag{7}$$

the standard deviation of the angular differences.

For algorithms that involve nonlinear optimization (search), we used the correct solution as the initial guess to study bias and sensitivity. In addition, we tested the convergence behavior of these algorithms by using initial guesses distributed uniformly in the parameter space.

3 Results

Translation Direction and Rotation Axis: We found in an extensive series of preliminary simulations (data not shown) that the axis of rotation had absolutely no impact on bias and sensitivity of these algorithms. For example, we ran one set of simulations with rotation about the Y-axis and another set of simulations with rotation about the Z-axis. We used the same random number seed to choose the depth values and the velocity noise. The translation estimates, on each individual trial, were exactly the same, regardless of rotation axis. At first, we found this quite surprising; it is often written in the literature that the egomotion problem is difficult because translation (e.g., along the X-axis) and rotation (e.g., about the Y-axis) produce similar image velocities. In retrospect, however, this result is not at all surprising. The five translation-first algorithms were each designed to eliminate rotation (by algebraic substitution, subspace projection, etc.).

In a preliminary set of simulations we also evaluated two other translation-first algorithms [14, 16] that were not designed to entirely/exactly eliminate rotation; the bias and sensitivity of these algorithms did depend on the rotation axis.

Since the performance of our six algorithms is invariant with respect to the rotation axis, we plot results for only a single rotation axis. Figure 1 plots bias and sensitivity, for sideways translation.

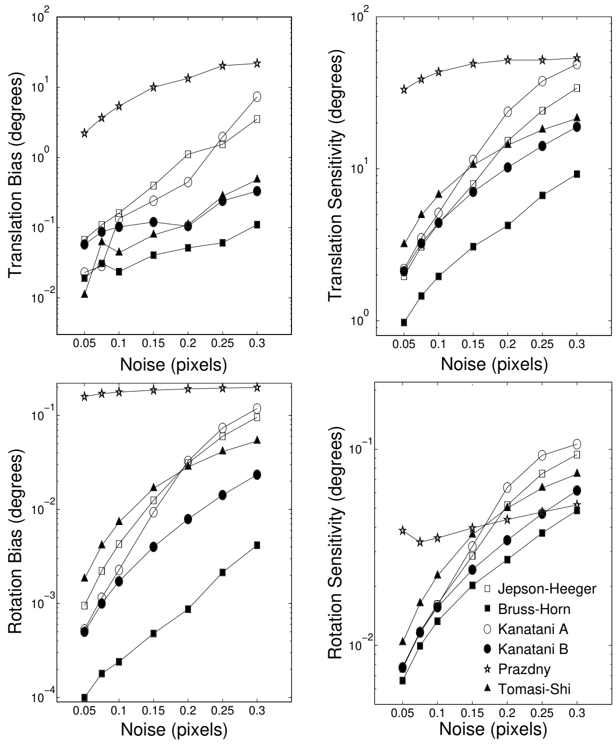

Figure 1: Bias and sensitivity for sideways translation and X-axis rotation.

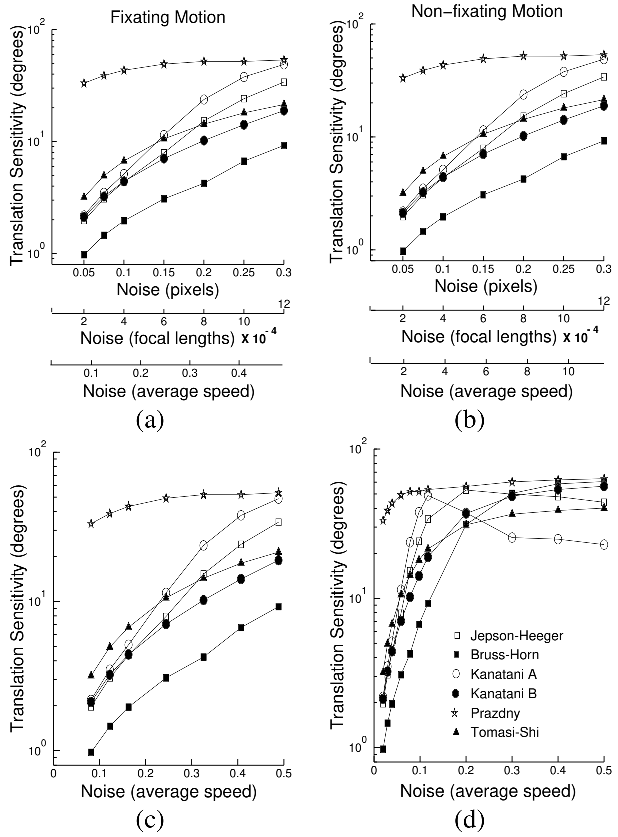

Figure 2: Translation sensitivity for fixating motion (a and c) and non-fixating (b and d) motions, as a function of noise. In (a) and (b) noise is independent of the image velocities, expressed in units of pixels, focal length, and as a percentage of the average speed of flow field. In (c) and (d) noise is proportional to the average speed of the flow field. Sensitivity of Jepson-Heeger and Kanatani-A in (d) goes down for large noise because translation bias is close to 90^0.

The performance of all six algorithms does, however, depend on translation direction. For all six algorithms, translation bias is smaller when the Z-component of translation is zero. Translation sensitivity also depends on the Z-component of translation, but not systematically. Some algorithms perform better and some worse for translation in the X- versus Z- directions.

Fixation: Figure 2 shows the translation sensitivity for fixating and non-fixating camera motions, as a function of noise. The translation was along the X-axis and rotation was either clockwise or counter-clockwise about the Y-axis. For the constant noise model (a and b), these two motions yield exactly the same sensitivity even though the average speed differs by a factor of four (in fact, the data are identical in figures 2a, 2b, and 2c). When the noise is proportional to average speed (c and d), fixating yields much lower sensitivity.

This effect, however, has to do entirely with the noise model, not the algorithms. Fixating results in slower image velocities. When the noise is proportional to image speed, this results in less noise. The first (constant) noise model is probably more representative of the behavior of most optical flow/feature tracking algorithms.

Field of View: Figure 3 shows the effect of varying the field of view (FOV). Results are shown only for the Jepson-Heeger (linear subspace) algorithm, but the conclusions are the same for all six algorithms. Increasing the FOV but fixing the number of pixels amounts to decreasing image resolution (that is, increasing the pixel size). There is a tradeoff between these two factors; increasing the FOV helps, but decreasing the image resolution hurts. We found that for sideways translation, the benefits typically outweight the drawbacks and you end up better off. But for straight-ahead translation, increasing the FOV typically yields worse results. Therefore, trying to increase the FOV simply by replacing the lens with one that has a shorter focal length is, in general, a bad idea.

To take advantage of the benefits of increasing field of view, one must also pay attention to image resolution. Obviously, it is impractical to replace the CCD chip each time you switch lenses. But for some applications, it may be possible to take advantage of wide FOV without losing image resolution. In particular, there has been much interest recently in producing a panorama from a sequence of images, e.g., collected by rotating a camera about its nodal point [2, 12]. Such panoramas have both high resolution and wide FOV, and are, therefore, ideally suited for egomotion estimation. We have found that the performance of the Jepson-Heeger (linear subspace) algorithm improves

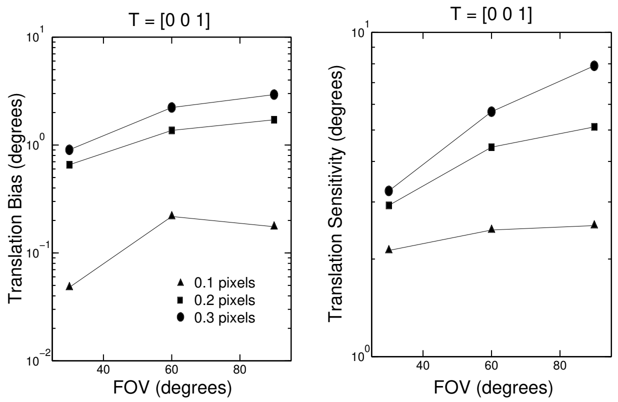

Figure 3: Translation bias and sensitivity of the Jepson-Heeger linear subspace method as a function of field of view for straight ahead translation and rotation around the X-axis, with noise levels of 0.1, 0.2, 0.3 pixels.

dramatically for very wide FOV (180 deg or wider).

Convergence: We also examined the convergence properties of the egomotion algorithms that required iterative numerical search (Bruss and Horn, Tomasi and Shi, Prazdny). In preliminary simulations we found that Prazdny's algorithm requires a very close initial guess, because it involves optimizing a system of third-order polynomials of three unknowns. We did not study the convergence properties of that algorithm further.

For the other two (Bruss and Horn, Tomasi and Shi) algorithms we performed an extensive series of simulations. Noise-free data was used. We densely sampled the unit hemisphere space of translation direction, and started a search from each translation sample. This gave us a set of translation estimates. We then chose a set of bins on the unit hemisphere, with binsize proportional to the angular distance from the correct solution, and counted the number of estimates that fell into each bin. Although there was no noise added, we ran 50 trials with different random-cloud depth structures. For each of the 50 depth structures, we started searches from each of 78 initial guesses, for a total of 3900 searches. Ideally, we would like there to be only one (global) minimum. We found, for both algorithms, that there are multiple local minima, but not many (typically, fewer than 5).

We proceeded to study where these local minima are located with respect to the correct solution. For straight ahead motion, the Bruss and Horn algorithm nearly always converged to the correct solution. For sideways motion, this algorithm converged to the correct solution about 50% of the time (over all random depth clouds and over all initial guesses). The rest of the time, the search got stuck in local minima that were all nearly 90^0 away from the correct solution. There is a good reason for the local minima to be at 90^0. Note that $\mathbf{T} \propto \mathbf{x}$ is an *exact* solution for the bilinear constraint in Eqn.2, i.e., each constraint line passes through the image point from which it is formed. For sideways translation direction, therefore, a point near the center of the image will be a reasonable solution (near all of the constraint lines).

The Tomasi and Shi algorithm converged to the correct solution about 70% of the time, for both straight ahead and sideways motions. The rest of the time, the search got stuck in local minima that were widely distributed in their angular errors.

In addition, we counted the number of times that the search failed to converge after 400 iterations. The Bruss and Horn algorithm and the Tomasi and Shi algorithm each failed 0.37% and 18% percent of the time, respectively.

4 Summary and Future Work

Here we list the main results of our simulations:

- The bias and sensitivity of all six algorithms are invariant with respect to rotation axis. This is a desirable property that is not true for all egomotion algorithms, but is true for these six algorithms.
- Performance does depend on translation direction, but not systematically.
- Fixating does not help when the noise is independent of the image velocities. Fixation does help if the noise is proportional to speed, but this is only for the trivial reason that the speeds are slower under fixation.
- Increasing the field of view does not necessarily improve performance.

The Bruss and Horn algorithm, based on the bilinear constraint in Eqn. 2, exhibited the best overall performance with respect to bias and sensitivity. However, it requires numerical optimization, and there are multiple local minima. Jepson and Heeger's linear-subspace method is closely related to the bilinear constraint used by Bruss and Horn's algorithm. However, the Jepson-Heeger algorithm is more sensitive to noise because it uses only a subset of the constraints. The two Kanatani algorithms do not perform as well as the Bruss and Horn algorithm because the Kanatani algorithms are based on the epipolar (coplanarity) constraint. The epipolar constraint is weaker than the bilinear (rigidity) constraint because there are non-rigid motions that satisfy the epipolar constraint. The Tomasi and Shi algorithm is relatively insensitive to the direction of translation, but has local minima and often fails to converge. Prazdny's algorithm appears to have the worst noise sensitivity. We believe that this is because it involves minimizing third-order polynomials.

The most important consideration, of course, is whether a given vision system is good enough to be used in a non-trivial, realistic robotics application. To determine that, we will need to perform extensive tests on real image sequences. Toward that end, we are installing and calibrating an experimental apparatus for the accurate measurement of the most important performance parameters of motion sequence analysis systems. The core of the setup is a stage that moves a camera in a controlled way with high accuracy. This apparatus will be used to digitize a library of carefully calibrated image sequences. These sequences will be used to validate the simulation results reported in this paper. In addition we plan to make the library of calibrated image sequences available on the World Wide Web.

References

[1] A. R. Bruss and B. K. Horn. Passive navigation. *Computer Graphics and Image Processing*, 21:3–20, 1983.

[2] S E Chen. QuickTime VR: An image-based approach to virtual environment navigation. In *Proceedings of ACM SIGGRAPH*, pages 29–38, Los Angeles, 1995.

[3] O D Faugeras, F Lustman, and G Toscani. Motion and structure from motion from point and line matches. In *Proceedings of the First International Conference on Computer Vision*, pages 25–34, London, 1987.

[4] D. J. Heeger and A. D. Jepson. Subspace methods for recovering rigid motion. i. algorithm and implementation. *International Journal of Computer Vision*, 7(2):95–117, 1992.

[5] E. C. Hildreth. Recovering heading for visually-guided navigation. *Vision Research*, 32(6):1177–1192, 1992.

[6] A D Jepson and D J Heeger. A fast subspace algorithm for recovering rigid motion. In *Proceedings of IEEE Workshop on Visual Motion*, pages 124–131, Princeton, NJ, 1991.

[7] A D Jepson and D J Heeger. Linear subspace methods for recovering translation direction. In L Harris and M Jenkin, editors, *Spatial Vision in Humans and Robots*, pages 39–62. Cambridge University Press, New York, 1993.

[8] K. Kanatani. 3-d interpretation of optical flow by renormalization. *International Journal of Computer Vision*, 11(3):267–282, 1993.

[9] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(10):133–135, 1981.

[10] W. J. MacLean, A. D. Jepson, and R. C. Frecker. Recovery of egomotion and segmentation of independent object motion using the em algorithm. In *Proceedings of the 5th British Machine Vision Conference*, pages 13–16, York, UK, 1994.

[11] S. Maybank. The angular velocity associated with the optical flow field arising from motion through rigid environment. *Proc. Roy. Soc. London. A*, 401:317–326, 1985.

[12] L McMillan and G Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of ACM SIGGRAPH*, pages 39–46, Los Angeles, 1995.

[13] K. Prazdny. Egomotion and relative depth map from optical flow. *Biological Cybernetics*, 36:87–102, 1980.

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

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

[16] J. H. Rieger and D.T. Lawton. Processing differential image motion. *J. Opt. Soc. Ame. A*, 2(2):354–359, 1985.

[17] A. Ruhe and P. A. Wedin. Algorithms for separable nonlinear least squares problems. *SIAM Review*, 22(3):318–337, 1980.

[18] C. Tomasi and J. Shi. Direction of heading from image deformations. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 422–427, New York, 1993.

[19] R Y Tsai and T S Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:13–27, 1984.

[20] J. Weng, T.S. Huang, and N. Ahuja. Motion and structure from two perspective views: Algorithms, error analysis, and error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):451–467, 1989.

[21] J. Weng, N. Ahuja, and T.S. Huang. Optimal motion and structure estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):864–884, 1993.

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