Determining Optical Flow for Large Motions Using Parametric Models in a Hierarchical Framework

J.L. Barron M. Khurana
barron@csd.uwo.ca mmk@csd.uwo.ca
Dept. of Computer Science
The University of Western Ontario
London, Ontario, N6A 5B7

Abstract

The problem of computing optical flow (image velocity) for large image motions has been addressed by a number of researchers, including Bergen et al. [5], who proposed using a number of parametric models in a hierarchical (pyramid) scheme to compute large motions using image intensity derivatives computed from "warped" image patches at each pyramid level. We have implemented a modified version of their algorithm for four parametric models (0^{th} order, i.e. assuming constant image velocity in a local neighbourhood, 1^{st} order, i.e. assuming at most an affine transformation of image velocity in a local neighbourhood, and 2^{nd} order, i.e. assuming image velocities are computed on either planar and curved environmental surfaces). While the description of Bergen et al.'s algorithm's is less detailed than desirable in some places, we followed it as faithfully as we could (with only minor modifications). We present two novel ideas in this paper. First, we present a quantitative analysis of the error obtained at each level in the pyramid (using Fleet's angle error measure [7]) and second, we propose four ways of thresholding on the optical flow field at each level in the pyramid, to avoid projecting bad image velocities down the pyramid. We investigate the use of condition numbers, determinants and eigenvalues of the least squares integration matrix or of Gaussian curvature of the local (warped) image patches as thresholds. We show that as velocity is propagated down the pyramid using our thresholding scheme it becomes more accurate for controlled optical flow field densities.

1 Introduction

A fundamental problem in Computer Vision is the computation of image motion, which is the perspective projection of individual 3D scene points moving relative to an observer (camera) onto a 2D planar imaging surface. Optical flow is an approximation of this 2D image motion, where a number of assumptions, such as that all illumination changes are due entirely to motion, the scene surfaces are Lambertian and image motion is locally translational, are made.

Another approach to computing image motion involves computing correspondences between "interest" points (candidate match points), such as corner points or other locally discriminable points in (usually) two images and using these correspondences as the flow. Generally, a smoothness constraint is imposed, for example, see the algorithms of Anandan [1] or Singh [17, 18] to ensure that a dense (100%) flow is obtained. In general, good interest points are difficult to find and without a smoothness constraint, sparse flows result, for example, see Barnard and Thompson [2]. A recent study [4] has shown that at least for slow motions (2-4 pixels/frames) local differential methods are better that correspondence methods, even those with smoothness constraints. As well, differential methods employing smoothness constraints, for example Horn and Schunck [11], weren't as good as local differential methods, for example, Lucas and Kanade [13], for slow image motions.

The approach to optical flow that we investigate here (reported in detail in a MSc thesis [12]) is local and hierarchical (allowing large image motions to be handled) and follows closely the work of Bergen et al. [5]. In contrast, there is the hierarchical differential work of Glazer [10, 9], which imposes a smoothness constraint at each level in the pyramid and is effectively hierarchical Horn and Schunck.

2 Background

The motion constraint equation is given as

$$I_x v_x + I_y v_y + I_t = 0, \tag{1}$$

where I_x, I_y and I_t are spatio-temporal intensity derivatives and $\mathbf{v} = (v_x, v_y)$ is the image velocity. For this equation to be applicable the motion must be locally translational, the intensity must be preserved over time and the image must be continuous over space and time in order to allow reliable derivatives to be computed (without undue aliasing effects). For large motions this last constraint is usually not satisfied.

The motion constraint equation defines a line in (v_x, v_y) space and any point on that line satisfies equation (1). The velocity with the smallest magnitude is normal to this line and is called the normal velocity,

$$\mathbf{v_n} = \frac{-I_t \nabla I}{||\nabla I||_2^2} = v_n \hat{\mathbf{n}}, \tag{2}$$

where $\nabla I = (I_x, I_y)$ is the spatial gradient that gives the unit direction of the normal velocity, $\hat{\mathbf{n}} = \frac{\nabla I}{\|\nabla I\|_2}$, and $v_n = \frac{-I_t}{\|\nabla I\|_2}$ is the magnitude of that normal velocity. Thus the motion constraint equation can be re-written as

$$\mathbf{v} \cdot \hat{\mathbf{n}} = v_n. \quad (3)$$

If we assume a constant image velocity $\mathbf{v}$ in some small image neighbourhood then two sets of spatio-temporal intensity derivatives (I_x, I_y and I_t) at different neighbourhood points gives a linear system of equations which can easily be solved (in the least squares sense if more then two sets of derivatives are available) if they are linearly independent. This overcomes the aperture problem inherent in the use of normal velocities. This approach is essentially the constant velocity model [13] or 0^{th} order parametric model [5].

We use Bergen et al.'s hierarchical framework and parametric model methodology [5] here to handle large image motions and overcome aliasing. We are interested in evaluating the various parametric models with respect to the different types of image data used and the accuracy they provide. We also investigate adding various thresholds to the hierarchical processing to obtain more accurate (but less dense) flows.

3 Hierarchical Framework

The hierarchical processing can be understood in terms of four steps:

1. Gaussian pyramid construction,
2. Image velocity (optical flow) calculation,
3. Image warping and
4. Coarse to fine refinement,

which we describe in detail in the subsections below.

3.1 Gaussian Pyramid

Image pyramids are multiresolution representations of an image that provide a range of coarse to fine views of the image. The coarser images are blurred and subsampled (slowing the image motion). We build our Gaussian pyramid in the standard way [6][1]; Level 0 is the original image, Level 1 is constructed by blurring Level 0 with a 2D separable Gaussian filter (with a standard deviation of 1) and then subsampling the blurred image by 2 in the images' dimensions. Level i is built from Level $i-1$ in a similar manner, by blurring and subsampling. In this way an image motion of 20 pixels/frame at level 0 is slowed to 10 pixels/frame at Level 1, 5 pixels/frame at Level 2 and 2.5 pixels/frame at Level 3 (most likely, the root of the pyramid in this case). Note that a 512×512 image at Level 0 becomes a 64×64 image at level 3.

[1] We also tried a Laplacian pyramid but with slightly less accurate results.

3.2 Image Velocity Calculation

The computation of image velocity can be viewed as three steps [4]:

1. Presmoothing the images to reduce noise and aliasing effects,
2. Computation of spatio-temporal intensity derivatives (and thus normal velocities) and
3. Integration of normal velocities within some neighbourhood into full image velocities.

Below we describe how the first two steps can be combined into one step and then how the integration step is performed for the four parametric models.

3.2.1 Prefiltering and Derivative Calculation

Bergen et al. [5] use just 2 images to estimate temporal derivatives. It was shown in Barron et al. [4] that using better differentiation kernels than simple pixel differences was crucial to obtaining good optical flow results. They used 4-point central differences after presmoothing the images with a spatio-temporal Gaussian filter with standard deviation of 1.5. This required a total of 15 images for 1 optical flow field calculation. Recently, Simoncelli [16] proposed some matched/balanced filters that combine presmoothing and differentiation to produce more accurate derivatives and yet require only 5 input images. One additional modification that we use is to presmooth the images in space and time by a simple averaging kernel $(\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$, thus, in total, we require 7 images instead of 15, when spatio-temporal Gaussian filtering is used. Table 1 below shows the 5-point kernels for Simoncelli's smoothing (p_5) and differentiation (d_5) filters. After modified smoothing, I_x is computed by applying p_5 in the t dimension, then p_5 to those results in the y dimension and finally d_5 to those results in the x dimension. I_x and I_t are computed in a similar appropriate manner. Table 2 shows the error results for the diverging tree sequence used in [4] for Lucas and Kanade's method (with eigenvalue threshold $\lambda_2 = 1.0$). Comparable improvements in accuracy were obtained by Simoncelli [16] for the Yosemite sequence over those reported for the same sequence in [4].

n	p_5	d_5
0	0.036	-0.108
1	0.249	-0.283
2	0.431	0.0
3	0.249	0.283
4	0.036	0.108

Table 1: Simoncelli's 5-point Matched/Balanced Kernels

3.2.2 Normal Velocity Integration

Given the spatio-temporal derivatives, I_x, I_y and I_t computed as described in the previous section (and

Filter	Average Error	Standard Deviation	Density
Spatio-Temporal Gaussian and 4-point Central Differences [4]	1.94	2.06	48.2%
Simoncelli Matched	0.92	1.14	54.4%
Simoncelli Modified and Matched	0.72	0.81	49.4%

Table 2: Angle error, standard deviation and density for the diverging tree sequence using various prefiltering and differentiation kernels for Lucas and Kanade's method.

hence, the normal velocities) we integrate small neighbourhoods of these values into image velocities using one of four parametric models [5] which we summarize below:

1. **Constant Velocity Model:** We assume velocity is constant in local image neighbourhoods, that is first and second order velocity derivatives are zero, i.e.

$$\mathbf{v}(\mathbf{x},\mathbf{y}) = \mathbf{v}(\mathbf{0},\mathbf{0}) = (u, v). \quad (4)$$

   This is the model used by Lucas and Kanade [13].

2. **First Order Model:** We assume no more than first order variation in local velocity neighbourhoods, i.e.

$$\mathbf{v}(x,y) = \mathbf{v}(0,0) + \frac{\partial \mathbf{v}}{\partial x}x + \frac{\partial \mathbf{v}}{\partial y}y. \quad (5)$$

   This is the model used by Fleet and Jepson [8, 7].

3. **Second Order Model:** We assume both first and second order variation in local velocity neighbourhood, i.e.

$$\mathbf{v}(x,y) = \mathbf{v}(0,0) + \frac{\partial \mathbf{v}}{\partial x}x + \frac{\partial \mathbf{v}}{\partial y}x + \frac{1}{2}\frac{\partial^2 \mathbf{v}}{\partial x^2}x^2 + \frac{\partial^2 \mathbf{v}}{\partial x \partial y}xy + \frac{1}{2}\frac{\partial^2 \mathbf{v}}{\partial y^2}y^2. \quad (6)$$

4. **Planar Model:** In the event that the local surface is planar, then

$$\frac{\partial^2 \mathbf{v}}{\partial x^2} = (2\frac{\partial^2 v}{\partial x \partial y}, 0) \quad \text{and} \quad \frac{\partial^2 \mathbf{v}}{\partial y^2} = (0, 2\frac{\partial^2 u}{\partial x \partial y}), \quad (7)$$

   and the second order model becomes:

$$\mathbf{v}(x,y) = \mathbf{v}(0,0) + \frac{\partial \mathbf{v}}{\partial x}x + \frac{\partial \mathbf{v}}{\partial y}y + \left(\frac{\partial^2 v}{\partial x \partial y}x^2 + \frac{\partial^2 u}{\partial x \partial y}xy, \frac{\partial^2 v}{\partial x \partial y}xy + \frac{\partial^2 u}{\partial x \partial y}y^2\right). \quad (8)$$

   These two 2^{nd} models are those proposed by Waxman and Wohn in their Velocity Functional method [19].

In all cases we solve for $\mathbf{v}(0,0)$ (and its 1^{st} and 2^{nd} order derivatives if present in the model) in the least squares sense using equations of the form $v_n(x,y) = \mathbf{v}(x,y) \cdot \hat{\mathbf{n}}$ in small local image neighbourhoods. For $n \times n$ neighbourhoods we solve $n^2 \times 2$, $n^2 \times 6$, $n^2 \times 8$ or $n^2 \times 12$ systems of equations, i.e. we simply have to invert 2×2, 6×6, 8×8 or 12×12 matrices.

3.2.3 Image Warping

The image velocity parametric models introduced in the previous section are accurate for small motions but, in the case of large motions, images have to be **warped** before they are processed. Image warping is performed by using a computed flow field (that we hope is reasonably accurate) as the initial velocities at each pixel (x, y) in the sequence. The corresponding regions in the image sequence at the appropriate frames and displacements from (x, y) are collected together to make the image domain differentiable in space and time. Each region collected is processed as if it were one single image patch for the purposes of computing the intensity derivatives at that pixel. This allows an image velocity calculation for that region which can be added to the initial estimated velocity to obtain a more accurate velocity value. Velocities are float vectors so we use bilinear interpolation to obtain the actual grayvalues at each pixel (as suggested by Bergen et al. [5]). Bilinear interpolation uses the grayvalues at the four nearest integer neighbours to compute the grayvalue at a given pixel at a noninteger position. For subpixel location (x, y), where i is the integer value of x and j is the integer value of y, we have following four nearest neighbours, (i, j), $(i + 1, j)$, $(i + 1, j + 1)$ and $(i, j + 1)$. If I_1, I_2, I_3 and I_4 are the grayvalues at points (i, j), $(i + 1, j)$, $(i + 1, j + 1)$, $(i, j + 1)$ respectively, then the intensity of subpixel (x, y) can be computed as follows:

$$I(x,y) = (1-v)(1-u)I_1 + u*I_2 + v(1-u)I_4 + uI_3, \quad (9)$$

where $u = x - i$ and $v = y - j$. A $5 \times 5 \times 5$ cube of grayvalues is collected about each displaced pixel using estimated image velocities, from which intensity derivatives I_x, I_y, and I_t are computed. These intensity derivatives are then used by one of the parametric models to compute a **warped** image velocity which is added to the initial velocity estimate to obtain a new (and hopefully more accurate) image velocity for that pixel.

3.3 Coarse to Fine Refinement

Coarse to fine hierarchical algorithms have a control strategy that starts at the top of the pyramid and proceeds level by level to the bottom of the pyramid and involves single level image velocity calculations and projections at each level. The steps involved are:

1. Compute the image velocities at the root level. Here we are assuming that the image motion has been blurred and slowed enough by the pyramid that any of the 4 parametric models will be sufficient to obtain a reasonable flow field. We use a neighbourhood size of $2n+1$, where $n=2$ in the integration step of the least square velocity calculation.

2. Project each computed velocity (after doubling it) at location (i,j) to locations $(2i,2j)$, $(2i+1,2j)$, $(2i,2j+1)$ and $(2i+1,2j+1)$. We use simple linear interpolation (averaging) to obtain the projected velocities at locations $(2i+1,2j)$, $(2i,2j+1)$ and $(2i+1,2j+1)$. That is, the velocity at $(2i+1,2j)$ is computed as the average of the velocities at $(2i,2j)$ and $(2i+2,2j)$, the velocity at $(2i+1,2j+1)$ is computed as the average the velocities at $(2i,2j)$ and $(2i+2,2j+2)$, etc. We also double the value of n at each projection step to take into account the enlarging of the aperture as we proceed down the pyramid.

3. Given the projected velocities around a pixel, we warp the images to remove the effect of the velocities and then perform a warped image velocity calculation. This warped velocity is added to the projected velocity and taken as the estimated velocity for that level.

4. These projected velocities are then doubled and projected 1-to-4 to the next level as described in the previous step. Warping is then performed again, etc.

5. This projection and warping is continued until the bottom image of the pyramid is reached. The estimated velocity there is taken as the final optical flow field.

4 Thresholding

One novel aspect of our work is the addition of thresholding to the hierarchical processing. Thresholding a computed optical flow field at some level in the pyramid allows for the removal (non-projection) of "bad" velocities that cannot be fixed by warping. This results in sparser but more accurate flow fields. We investigate the use of 4 thresholds in this paper:

1. Condition Number: a number between 1.0 and ∞ that indicates how well conditioned the least squares matrix is (the lower the better). We use the singular value decomposition of the matrix to obtain its L_2 condition matrix, κ [15].

2. Determinant: the determinant of the least squares matrix is computed using Crout's LU decomposition method [15]. Determinant values near 0 indicate singularity while large determinant values may indicate robustness.

3. Smallest Eigenvalue: We use a Jacobi transformation to compute the eigenvalues/eigenvectors of the least squares directions [15]. The eigenvectors are mutually orthogonal and their eigenvalues indicate how important data in their directions are to the overall computation. Small eigenvalues indicate a potentially poor calculation.

4. Gaussian Curvature: This is defined as $I_{xx}I_{yy} - I_{xy}^2$ where these 2^{nd} order derivatives are computed using differentiation with respect to x and y on I_x and I_y values. Large Gaussian curvature values should indicate large local intensity structure, where good image velocity calculations can occur.

At the root level of the pyramid it is easy to choose a threshold that is reasonably robust for a number of images. For example, Barron et al. [4] found that an eigenvalue threshold of 1.0 worked well for most images for their implementation of Lucas and Kanade's method [13]. However, in hierarchical thresholding, the threshold must necessarily change from level to level in the pyramid. Consider condition number thresholding. At the root level of the pyramid a threshold value of 10 seems good for most of the images we examined. However, after one projection (i.e. the next lower level in the pyramid) the 5 images are almost completely registered at each pixel, with the consequence that the condition numbers of the individual integration matrices values grow much larger. We tried, by trial and error, to find reasonably good condition number thresholds at each level but were not successful. This trial and error strategy was also tried for the other thresholds with a similar lack of success. Instead we adopted the following threshold selection strategy:

- At the root we selected a preset percentage of the "best" velocities according to the threshold values. For example, for this paper's results we use median thresholding (50% cutoff) for the root flow field.

- At at lower levels, we used a second much higher preset percentage of best thresholds, in this paper 90%. The rational here is that the best velocities have been projected from the root so less velocities at subsequent levels should be rejected, as most likely, more of them are also good.

This strategy of threshold selection means we end up with flow fields with a preset density (for those images with 100% textured backgrounds). For example, for a 3-level pyramid with an initial root cutoff percentage of 50% and a cutoff percentage of 90% for remaining 2 levels, we have densities of 50% at the root, 45% at the next level down and 40.5% at the lowest level (the original image).

5 Experimental Technique

In this section we describe what image data was used and how the error analysis was performed.

5.1 Image Data

We use three sets of image sequence pairs with large image motions, for which the correct flow is known,

allowing a quantitative error analysis. The first set contain 2 sequences for paning camera motion (translation perpendicular to the line of sight axis) and zooming camera motion (translation along the line of sight axis). Figure 1 shows an image from each of these sequences. The tree sequences were pro-

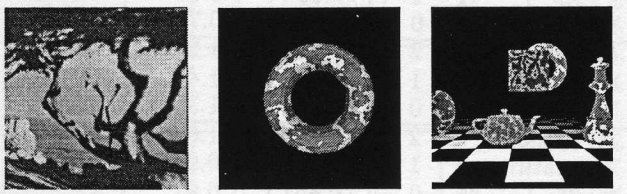

Figure 1: *An image of each of the the sets of image sequences pairs used for experimental results* **(left):** *a* 150×150 *image from the panning/zooming tree sequences,* **(center):** *an* 512×512 *image from the panning/zooming torus sequences and* **(right):** *an* 512×512 *image from the panning/zooming general scene sequences.*

duced by David Fleet [7]. We used every 3^{rd} image to obtain faster image motions. For the panning sequence, motions were 6-7 pixels/frame while for the zooming sequence motions had speeds of about 6 pixels/frame at the corners of the flows. To perform error analysis we simply divided all computed flows by 3 and compared with the correct known flow. The last two sets were produced by a program written by Stephen McKee [14]. These sequences consisting of panning/zooming with respect to a single textured torus object or panning/zooming with respect to general scene. The panning torus sequence has motions of about 10 pixels/frame while the zooming torus sequence has speed of up to 12 pixels/frame at the torus' outer edges. The general panning sequence had motions of about 15-17 pixels/frame while the general zooming sequence had speed of about 10 pixels/frame at the corners.

The correct flows for these six image sequences are shown in Figure 2.

5.2 Error Analysis

We use angular error to compute the error between the computed optic flow and the correct optic flow [7]. Because the velocity $\mathbf{v} = (u, v)^T$ can be represented as a spatiotemporal direction $\mathbf{v} = (u, v, 1)$ in units (pixel,pixel,frame) we can measure the angle error by considering velocity as a vector in space-time. In this view, estimated velocity $\mathbf{v_e} = (u_e, v_e)^T$ is represented as:

$$\hat{\mathbf{v}}_\mathbf{e} \equiv \frac{1}{\sqrt{u_e^2 + v_e^2 + 1}}(u_e, v_e, 1)^T \qquad (10)$$

and the correct velocity $\mathbf{v_c} = (u_c, v_c)^T$ is represented as

$$\hat{\mathbf{v}}_\mathbf{c} \equiv \frac{1}{\sqrt{u_c^2 + v_c^2 + 1}}(u_c, v_c, 1)^T. \qquad (11)$$

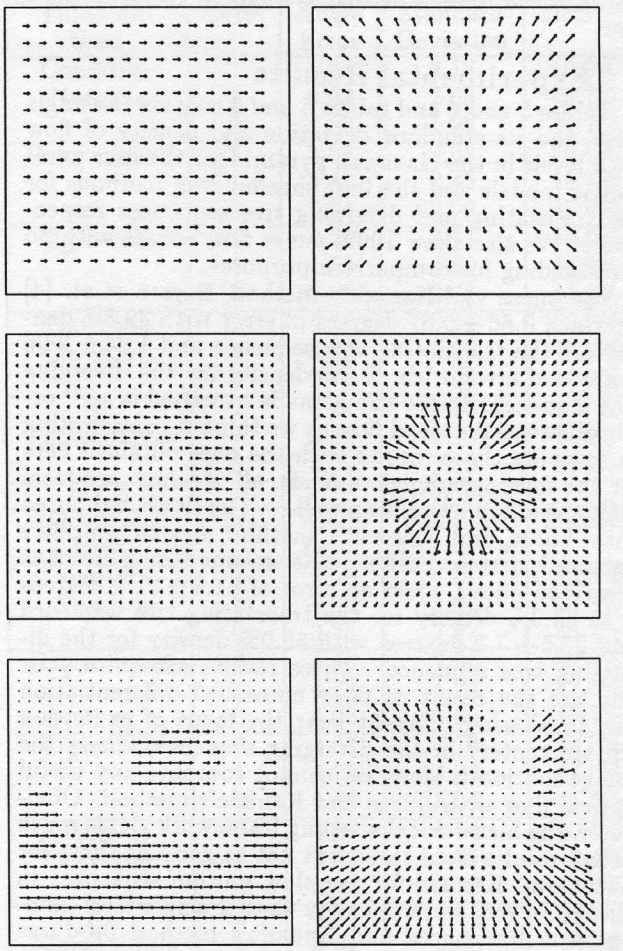

Figure 2: *The correct flow for each of the size image sequences:* **(top):** *the correct flows for the panning/zooming tree sequences,* **(middle):** *the correct flows for panning/zooming torus sequences and* **(bottom):** *the correct flows for the panning/zooming general scene sequences.*

The error in $\mathbf{v_e}$ is then found as

$$\psi = arccos(\hat{\mathbf{v}}_\mathbf{c} \cdot \hat{\mathbf{v}}_\mathbf{e}). \qquad (12)$$

This is the angle between $\hat{\mathbf{v}}_\mathbf{e}$ and $\hat{\mathbf{v}}_\mathbf{c}$. This error measure is less biased for small or large velocities [7]. To obtain error measures for flows in the pyramid, we expand those flows not at the bottom pyramid by projecting 1-to-4 and multiplying by 2 until the expanded flow and correct flow are the same size. In the event the final computed and projected flow is not exactly the same size as the correct flow (this happens when an image size is not a power of 2) we truncate some rows/columns of the computed flow to make it the same size as the correct flow. This allows measure-

ment analysis to proceed as outlined above but may result in some small density changes in the computed flow.

6 Experimental Results

Tables 3 and 4 and tables 5 and 6 contain the angle error and its standard deviation and density of flow for 3 levels in the Gaussian pyramid for the four parametric models and the four thresholding methods for the translating and diverging tree sequences respectively. We also show 100% dense flow results with no thresholding for comparison purposes.

For Lucas and Kanade's method, Barron et al. [4] obtained 0.66 ± 0.67 degrees of error with 39.8% density for the translating tree sequence and 1.94 ± 2.06 degrees of error with 48.2% density for the diverging tree sequence. Lucas and Kanade's method is a 1^{st} order differential optical flow so we think it is interesting to compare those results with the ones obtained here for the four parametric models, all of which are local differentiable methods as well. The flows computed with Lucas and Kanade's method and Simoncelli's balanced/matched differentiation/smoothing filters as described above, yield an error of 51.8 ± 31.9 degrees with 63.3% density for the translating tree sequence and 27.5 ± 7.2 degrees with 55.0% density for the diverging tree sequence. These results are quite poor and are the result of poor numerical differentiation due to aliasing. Except near the focus of expansion (image center) in the diverging sequence, where the image motion is slow, we cannot compute any useful optical flow with Lucas and Kanade's method. Obviously, the hierarchical warping framework yields much better results than these but not as good as those for Lucas and Kanade for the slow motion sequences in [4]. However, we emphasize that a single-level computation like Lucas and Kanade's method produces meaningless results for the fast motion sequences. We also note that some of our results approach the Lucas and Kanade optical flow results in accuracy and density even though the image motion is 3 times greater.

We found that the 2×2 (constant velocity) parametric model give the best results, even though the textured frames of the sequence are those of a moving plane and so best fit the 8×8 (planar surface) model. The error at the root level is always the highest (with 50%) density) and is always reduced at each lower level. Hierarchical processing does indeed improve accuracy with little loss in density of flow. The densities deviate from the predicted ones at the lower levels for a number of reasons. We used outlier thresholding (all velocities with a magnitude greater than 40 were eliminated, this accounts for less than for than 1% reduction in the flow density). The expansion of the flow at intermediate levels in the pyramid into a 150×150 flow field either increased or decreased the density depending on whether or not there were image velcities at or near the edges of the flow field. As well, flow fields that were initially sparse are projected to the next lowest level by simply multiplting by 2 and copying to the 4 corresponding locations at the lower level; no bilinear interpolation using projected neighbouring velocities could be done and this increased

2×2		Translating Tree		
Threshold Used	Level	Angle Error	Standard Deviation	Density
Condition Number	2	1.39	1.12	50.0%
	1	1.15	1.05	44.1%
	0	0.97	0.89	41.7%
Determinant	2	1.37	1.11	50.0%
	1	1.03	0.86	48.0%
	0	0.84	0.70	47.0%
Eigenvalue	2	1.51	1.25	50.0%
	1	1.51	1.47	44.2%
	0	1.35	1.36	42.0%
Gaussian Curvature	2	1.43	1.00	50.0%
	1	0.93	0.92	42.7%
	0	0.86	0.97	39.9%
No Thresholding	2	1.45	1.12	100%
	1	1.03	1.09	100%
	0	0.95	1.03	100%
6×6		Translating Tree		
Threshold Used	Level	Angle Error	Standard Deviation	Density
Condition Number	2	1.66	1.42	50.0%
	1	1.51	1.51	48.2%
	0	1.30	1.42	46.5%
Determinant	2	1.69	1.46	50.0%
	1	1.32	1.36	48.1%
	0	1.08	1.17	46.4%
Eigenvalue	2	1.78	1.37	50.0%
	1	1.61	1.39	48.0%
	0	1.31	1.14	46.1%
Gaussian Curvature	2	1.89	1.49	50.0%
	1	1.19	1.09	42.7%
	0	1.13	1.00	39.9%
No Thresholding	2	1.84	1.51	100%
	1	1.28	1.25	100%
	0	1.17	1.10	100%

Table 3: Angle error, standard deviation and density for the 2×2 and 6×6 parametric models for the translating tree sequence for 3 levels in the Gaussian pyramid.

inaccuracy. Bilinear interpolation is important to the projection process. It effectively smooths the image velocity field and increases accuracy. Lastly, we note that occasionally the flow fields with no thresholding were often more accurate than those with thresholding; this shows that sometimes thresholding may not work (and less accurate velocities were retained while more accurate ones were eliminated) but more likely, it shows the effect of always doing bilinear interpolation in the projection step for non-thresholded flow calculations versus not always doing it (when thresholding is used and some image velocities are not completely surrounded by other projected velocities). If less accurate velocities are retained at the root level flows it was more likely they would be retained at lower levels in the pyramid also. If a velocity with large error is projected down the pyramid then warping may compound the error and the flow may become worse. On

8 × 8		Translating Tree		
Threshold Used	Level	Angle Error	Standard Deviation	Density
Condition Number	2	1.91	1.62	50.0%
	1	1.67	1.65	47.6%
	0	1.44	1.54	45.8%
Determinant	2	2.11	2.07	50.0%
	1	1.86	2.21	48.1%
	0	1.48	1.93	46.4%
Eigenvalue	2	2.22	2.23	50.0%
	1	1.43	1.28	42.7%
	0	1.37	1.21	39.9%
Gaussian Curvature	2	2.23	1.82	50.0%
	1	1.67	1.88	42.7%
	0	1.56	1.69	39.9%
No Thresholding	2	2.33	2.22	100%
	1	1.67	1.88	100%
	0	1.56	1.69	100%
12 × 12		Translating Tree		
Threshold Used	Level	Angle Error	Standard Deviation	Density
Condition Number	2	2.44	2.14	50.0%
	1	2.05	2.47	46.9%
	0	1.67	2.11	44.9%
Determinant	2	2.50	2.57	50.0%
	1	2.18	2.72	47.3%
	0	1.34	1.40	4.7%
Eigenvalue	2	2.32	2.25	50.0%
	1	2.12	2.07	46.6%
	0	1.89	1.98	44.5%
Gaussian Curvature	2	2.67	2.94	50.0%
	1	1.78	1.90	42.7%
	0	1.59	1.72	39.9%
No Thresholding	2	2.71	2.77	100%
	1	1.98	1.97	100%
	0	1.83	1.87	100%

Table 4: Angle error, standard deviation and density for the 8 × 8 and 12 × 12 parametric models for the translating tree sequence for 3 levels in the Gaussian pyramid.

2 × 2		Diverging Tree		
Threshold Used	Level	Angle Error	Standard Deviation	Density
Condition Number	2	3.64	2.18	50.0%
	1	3.93	2.68	44.7%
	0	2.89	1.92	40.1%
Determinant	2	3.93	2.68	50.0%
	1	3.42	2.42	48.4%
	0	2.94	2.09	47.0%
Eigenvalue	2	4.25	3.11	50.0%
	1	4.09	2.99	44.2%
	0	3.66	2.66	41.2%
Gaussian Curvature	2	4.11	3.13	50.0%
	1	3.20	2.35	43.4%
	0	2.76	1.92	40.8%
No Thresholding	2	4.41	2.96	100%
	1	3.59	2.43	100%
	0	3.05	2.15	100%
6 × 6		Diverging Tree		
Threshold Used	Level	Angle Error	Standard Deviation	Density
Condition Number	2	4.48	4.21	50.0%
	1	4.04	3.64	48.3%
	0	3.35	2.82	38.7%
Determinant	2	4.46	4.34	50.0%
	1	4.73	4.24	47.0%
	0	4.03	3.54	46.1%
Eigenvalue	2	4.61	4.10	50.0%
	1	4.73	4.24	47.0%
	0	4.03	3.54	45.9%
Gaussian Curvature	2	4.59	4.54	50.0%
	1	3.66	3.21	43.4%
	0	3.33	2.79	40.8%
No Thresholding	2	4.84	4.11	100%
	1	3.91	3.06	100%
	0	3.47	2.94	100%

Table 5: Angle error, standard deviation and density for the 2 × 2 and 6 × 6 parametric models for the diverging tree sequence for 3 levels in the Gaussian pyramid.

the other hand, velocities with small errors at the root level usually get corrected with projection and warping and become better.

Tables 7, 8, 9 and 10 show the flow field error for four levels in the pyramid for the Pan/Zoom Torus sequences and the Pan/Zoom Gerneral sequences for the four parametric models for condition number thresholding. Generally, these were the best results (over other thresholding options) and so to save space we only show these. Other error results were about 20% or so higher and the no-thresholding results were often quite bad as outliers were computed in image areas that were nearly uniform in intensity. (Note that the tree images are 100% textured but these images are not; much of the background is uniformily black.) Again, we note that both angle error and density always decrease from the flows at the root of the pyramid to those at the bottom of the pyramid. Because these image sequences are generated from scenes with objects having actual depth variation in them, higher order models often do better than the constant (2×2) parametric model. For the Pan Torus and Zoom Torus sequences the 6 × 6 and 12 × 12 models do the best while the 2×2 models gives the worst result. Interestingly, the torus is a curved object so we should expect local 1^{st} and 2^{nd} order variations in the local velocity neighbourhoods (which is exactly what is found). For the General Pan sequence the 8 × 8 model seems the best with the 6×6 model being a close second. For the General Zoom the 6 × 6 model is again the best with the 2 × 2 model being the worst. Figure 3 shows the flow fields for the 6 × 6 model for the Pan Torus and Zoom Torus sequences and the 6×6 model for the Pan General and Zoom General sequnces. It is interesting to look at one of these results in more detail. Figure 4 shows the computed flow fields for the 4 levels in the

8×8		Diverging Tree		
Threshold Used	Level	Angle Error	Standard Deviation	Density
Condition Number	2	5.24	4.43	50.0%
	1	4.48	3.55	48.1%
	0	3.81	3.13	46.2%
Determinant	2	5.75	5.15	50.0%
	1	4.94	4.42	48.2%
	0	4.61	3.69	46.7%
Eigenvalue	2	5.11	3.86	50.0%
	1	5.40	4.83	39.7%
	0	5.14	5.00	36.0%
Gaussian Curvature	2	5.56	5.16	50.0%
	1	4.27	3.57	43.4%
	0	3.85	3.39	40.8%
No Thresholding	2	5.73	4.69	100%
	1	4.57	3.61	100%
	0	4.13	3.74	100%
12×12		Diverging Tree		
Threshold Used	Level	Angle Error	Standard Deviation	Density
Condition Number	2	5.86	4.49	50.0%
	1	5.52	4.46	47.8%
	0	4.87	4.28	46.0%
Determinant	2	5.76	4.70	50.0%
	1	5.49	4.44	47.6%
	0	4.31	3.81	24.2%
Eigenvalue	2	6.42	4.47	50.0%
	1	6.41	4.48	47.9%
	0	5.93	4.27	46.2%
Gaussian Curvature	2	5.58	4.68	50.0%
	1	5.05	4.29	43.4%
	0	4.31	3.89	40.8%
No Thresholding	2	6.30	4.49	100%
	1	5.59	4.23	100%
	0	4.99	4.25	100%

Table 6: Angle error, standard deviation and density for the 8×8 and 12×12 parametric models for the diverging tree sequence for 3 levels in the Gaussian pyramid.

pyramid. As we can see (with the aid of tables 9 and 10), the flow get more accurate but less dense as we go from the root flow to the bottom flow. However, a problem can be seem from the results: if a small velocity at the bottom image in the pyramid is smoothed and halved at each step up the pyramid, then at the root, its velocity is insignificant and it may, in fact, be thresholded out. For example, if the velocity magnitude is 1 pixel/frame at the bottom then it would be $\frac{1}{8}$ pixel/frame at the root and may not be considered a good velocity by our thresholding criteria. This problem occurs for all zoom sequences where velocity magnitude ranges from 0 at the FOE to a large value at the corners of the image. Currently, we have no way of re-introducing velocities at some level in the pyramid when that velocity has been eliminated by thresholding at a higher level.

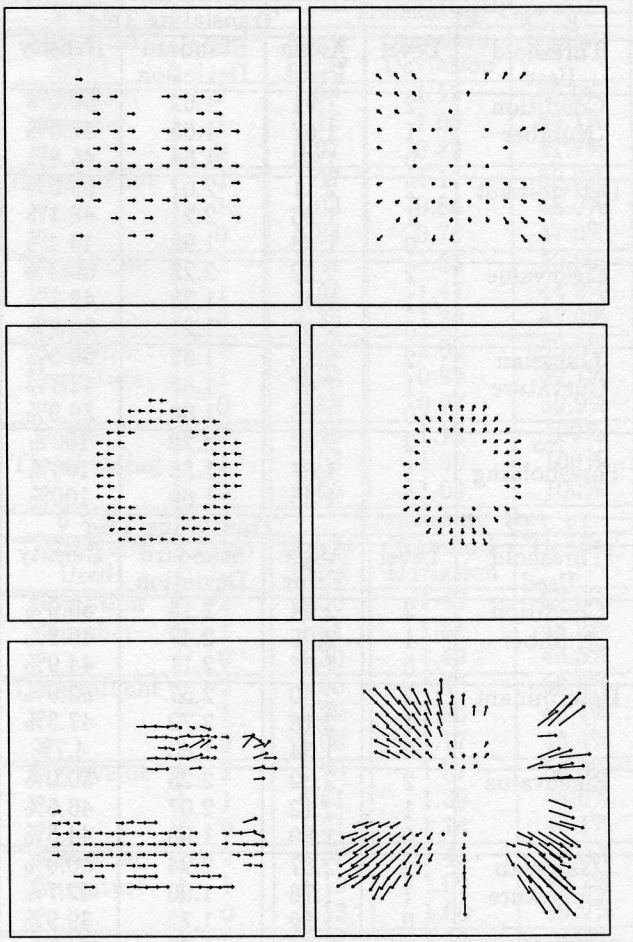

Figure 3: Flow fields computed by our hierarchical warping framwork **(top):** Translating and Diverging tree flow fields for the 2×2 (constant velocity) parametric model, **(middle):** Panning and Zooming torus flow fields for 6×6 (affine) parametric model and **(bottom):** Panning and Zooming general scene flow fields for 6×6 parametric model. Condition number thresholding was used for all flows.

7 Conclusions

In conclusion, we re-iterate our major findings. We have implemented a modified version of Bergen et al.'s differential hierarchical warping algorithm [5]. We used Simoncelli's differentiation/low-pass filtering [16] and 7 images, instead of just 2 images and simple pixel differences (as Bergen et al. did). We performed a quantitative error analysis on our flow results (not done by Bergen et al.) and showed improved accuracy as we processed flow down the pyramid. We introduced the notion of thresholding in the hierarchical framework. Thresholding has been shown to be a much needed tool for optical flow computations [4] and has again proved itself successful here. Lastly, our

Parametric Model	Level	Pan Torus Angle Error	Standard Deviation	Density
2 × 2	3	5.93	11.86	32.8%
	2	4.06	9.47	29.4%
	1	2.98	7.61	26.4%
	0	2.32	6.29	23.6%
6 × 6	3	3.84	8.56	29.4%
	2	2.54	5.97	26.5%
	1	1.81	3.67	23.83%
	0	1.53	2.40	21.24%
8 × 8	3	4.18	8.47	28.6%
	2	2.93	6.17	25.6%
	1	2.52	5.16	23.1%
	0	2.04	3.82	20.6%
12 × 12	3	3.19	5.93	38.0%
	2	2.24	3.60	21.6%
	1	1.88	2.69	19.5%
	0	1.59	1.87	17.4%

Table 7: Angle error, standard deviation and density for the various parametric models for the panning torus sequences for 4 levels in the Gaussian pyramid when condition number thresholding is used.

Parametric Model	Level	Zoom Torus Angle Error	Standard Deviation	Density
2 × 2	3	14.01	18.35	25.6%
	2	10.00	14.04	23.0%
	1	7.64	11.19	20.7%
	0	6.49	9.58	18.4%
6 × 6	3	8.87	12.79	22.5%
	2	6.25	8.81	20.2%
	1	5.10	6.31	18.2%
	0	4.29	3.61	16.2%
8 × 8	3	8.84	11.69	21.9%
	2	7.14	9.49	19.7%
	1	5.95	7.74	17.8%
	0	5.07	5.85	15.8%
12 × 12	3	7.04	7.49	18.2%
	2	5.99	5.23	16.3%
	1	5.27	4.18	14.7%
	0	4.85	3.17	13.1%

Table 8: Angle error, standard deviation and density for the various parametric models for the zooming torus sequence for 4 levels in the Gaussian pyramid when condition number thresholding is used.

work has shown the usefulness of generating complex synthetic image sequences with known correct flow fields for testing optical flow work [14].

Further work is needed. First, the algorithm should be extended to allow the re-introduction of small velocities in lower levels in the pyramid even if they have been thresholded out at higher levels. The evaluation of thresholding values as confidence values (as done for eigenvalues and slow image motions in Barron and Eagleson's motion and structure work [3]) is needed.

Parametric Model	Level	General Pan Angle Error	Standard Deviation	Density
2 × 2	3	19.02	31.98	39.2%
	2	13.50	26.91	35.0%
	1	13.94	27.25	31.5%
	0	14.48	27.25	28.1%
6 × 6	3	16.52	28.17	37.9%
	2	13.55	26.12	33.9%
	1	12.48	25.31	30.5%
	0	11.82	24.52	27.2%
8 × 8	3	16.20	27.40	37.6%
	2	12.50	23.40	33.5%
	1	11.49	21.97	30.1%
	0	10.69	20.87	26.9%
12 × 12	3	25.38	36.19	33.8%
	2	21.60	33.32	30.0%
	1	20.69	32.90	27.0%
	0	20.19	32.94	24.03%

Table 9: Angle error, standard deviation and density for the various parametric models for the panning torus sequence for 4 levels in the Gaussian pyramid when condition number thresholding is used.

Parametric Model	Level	General Zoom Angle Error	Standard Deviation	Density
2 × 2	3	14.14	24.22	48.9%
	2	10.72	20.27	43.9%
	1	9.54	19.53	39.4%
	0	9.22	19.72	35.1%
6 × 6	3	13.18	22.33	46.2%
	2	10.27	18.73	41.6%
	1	8.12	15.50	37.4%
	0	6.80	13.14	33.4%
8 × 8	3	14.86	22.48	46.0%
	2	11.89	19.67	44.2%
	1	9.92	17.39	37.1%
	0	8.57	15.93	33.0%
12 × 12	3	13.75	19.07	40.3%
	2	11.19	15.75	35.9%
	1	9.77	14.61	32.2%
	0	8.71	14.22	28.8%

Table 10: Angle error, standard deviation and density for the various parametric models for the zooming torus sequence for 4 levels in the Gaussian pyramid when condition number thresholding is used.

Lastly, the motion constraint equation assumes local image translation with no intensity discontinuities due to occlusion. In the case of significant image rotation, warping would fail, suggesting the need for some kind of non-linear image warping. McKee's program is capable of producing such image sequences for testing purposes, all that is needed is a new warping strategy.

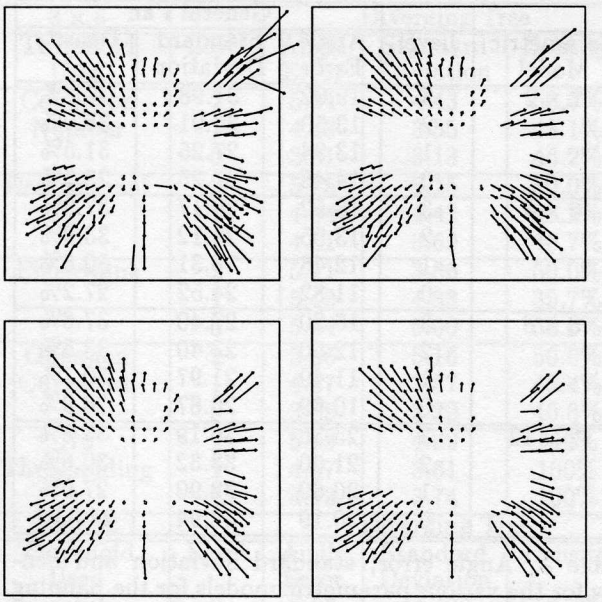

Figure 4: *Expanded flow fields for the General Zoom sequence for* **(top left):** *the 64 × 64 flow at Level 3, the root,* **(top right):** *the 128 × 128 flow at Level 2,* **(bottom left):** *the 256 × 256 flow at Level 1 and* **(bottom right):** *the 512 × 512 flow at Level 0, the bottom.*

Acknowledgements

We gratefully acknowledge financial support from NERSC and from the Department of Computer Science at the University of Western Ontario.

References

[1] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *IJCV*, 2:283–310, 1989.

[2] S. T. Barnard and W. B. Thompson. Disparity analysis of images. *IEEE PAMI*, 2(4):333–340, 1980.

[3] J. L. Barron and R. Eagleson. Recursive estimation of time-varying motion and structure parameters. *Pattern Recognition*, 29(5):797–818, May 1996.

[4] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *IJCV*, 12(1):43–77, 1994.

[5] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proceedings of ECCV, Santa Margherita, Italy*, pages 237–252. Springer-Verlag, May 1992.

[6] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. on Communications*, 31:532–540, 1983.

[7] D. J. Fleet. *Measurement of Image Velocity.* Kluwer Academic Publishers, Norwell, 1992.

[8] D. J. Fleet and A. D. Jepson. Computation of component image velocity from local phase information. *IJCV*, 5(1):77–104, 1990.

[9] F. C. Glazer. Hierarchical gradient-based motion detection. In *DARPA Proceedings of Image Understanding Workshop*, pages 733–748, Los Angeles, California, February 1987.

[10] Frank C. Glazer. *Hierarchial Motion Detection.* PhD thesis, Univ. of Massachusetts, Feb 1987. COINS-87-02.

[11] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–204, 1981.

[12] Meeta Khurana. Determining optical flow for large motions using differential techniques in a hierarchical framework. Master's thesis, University of Western Ontario, September 1996.

[13] B. D. Lucas and T. Kanade. An iterative image-registration technique with an application to stereo vision. In *Proceedings of IJCAI*, pages 674–679, Vancouver, British Columbia, 1981.

[14] Stephen McKee. Modelling the motion and dynamics of computer generated images for optical flow analysis. Master's thesis, University of Western Ontario, June 1996. (also see TR-498, Dept. of Computer Science, University of Western Ontario).

[15] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C.* Cambridge University Press, 1988.

[16] Eero P. Simoncelli. Design of multi-dimensional derivative filters. *ICIP*, 1:790–793, 1994.

[17] A. Singh. An estimation-theoretic framework for image flow computation. In *Proceedings of ICCV*, pages 168–177, Osaka, Japan, December 1990.

[18] A. Singh. *Optic flow computation: A unified perspective.* IEEE Computer Society Press, 1992.

[19] A. M. Waxman and K. Wohn. Contour evolution, neighborhood deformation, and global image flow: planar surfaces in motion. *Int. J. of Robotics Res.*, 4(3):95–108, 1985.