Quantitative Depth Recovery from Time-Varying Optical Flow in a Kalman Filter Framework

John Barron[1], Wang Kay Jacky Ngai[1], and Hagen Spies[2]

[1] Department of Computer Science, University of Western Ontario
London, Ontario, Canada
ngai@csd.uwo.ca,barron@csd.uwo.ca
[2] ICG-III: Phytosphere, Research Center Jülich,
52425 Jülich, Germany
h.spies@fz-juelich.de

Abstract. We present a Kalman filter framework for recovering depth from the time-varying optical flow fields generated by a camera translating over a scene by a known amount. Synthetic data made from ray traced cubical, cylinderal and spherical primitives are used in the optical flow calculation and allow a quantitative error analysis of the recovered depth.

Keywords: Depth Map, Depth from Optical Flow, Kalman Filter, 3D Camera Motion, Quantitative Error Analysis

1 Introduction

We consider the problem of depth recovery from monocular image sequences when the 3D camera motion is either known [10] or recovered from a Motion and Structure algorithm [2]. Using the assumption of local planarity, inverse depth can first be computed from the measured optical flow, then integrated into surface normals in small neighbourhoods and tracked over time in a Kalman filter framework [7,6].

There are a number of approaches for computing depth from optical flow/image derivatives in the literature. Piecewise planarity has been used by Faugeras and Lustman [5] to solve for motion and structure from point displacements using recursive least squares. Barron et al. [1] also used the planarity assumption to recover motion and structure from time varying optical flow. More recent work has used Kalman filtering as a way to recover depth values from time varying flow where the camera motion is known. Kalman filtering seems especially appropriate here as, from empirical observations by the 1^{st} author, optical flow error (as computed from Lucas and Kanade's algorithm [8]) has Gaussian mean-zero error. This was conjectured after examining plots of optical flow error for a number of images that looked Gaussian. Also, use of the Lorentzian robust

estimator [4] with the Lucas and Kanade calculation shows that the best results approached the standard least squares Lucas and Kanade results: least squares is optimal for Gaussian mean-zero noise.

2 Literature Survey

Some of the Kalman filter approaches include those by Matthies et al. [12], Heel [9] Xiong and Shafer [15] and Hung and Ho [10]. Matthies et al.'s approach [12] is a Kalman filter-based algorithm for recovering the depth map of a 3D scene from 2 frame image displacements, using the sum of squared differences (SSD) method to integrate the squared intensity difference between two shifted images over a small area to obtain an estimate of the optical flow, $(v_1, v_2)^T$, and its covariance matrix, P_m, for each pixel of the image. A Kalman filter tracks the inverse depth using predictions about how the depth changes. Heel [9] computed depth directly from image intensity derivatives in a Kalman filter framework by assuming local constant depth. If the camera motion is unknown it can also be estimated in a least squares framework. Given $(X(t), Y(t), Z(t))$ $(X(t+1), Y(t+1), Z(t+1))$ can be estimated and bilinear interpolation is then used to find the Z value at pixel locations. Further interpolation and extrapolation are used to compute dense fields. Heel has tested his algorithm on planar images with unknown translation with good results. Xiong and Shafer [15] use an extended Kalman filter to track the sensor's motion and the depth of each image pixel. The Levenberg-Marquardt method is used here to perform the nonlinear minimization to get an initial estimate of the motion and depth parameters. Their experiments include estimating camera motion and surface depth in a 51 image sequence generated by a moving camera over a scene of a straw hat, with realistic subjective surface results. Hung and Ho [10] use image intensity derivatives in a Kalman filter framework for known camera translation. At each time, the depth map is warped using the known camera motion and used to compute depth at pixel points. An experiment with a scene consisting of a coke can, a small box and a planar poster as background, show that with smoothing, good subjective depth can be recovered.

3 A New Approach

We also assume that the direction of sensor translation, $\hat{u}$, and its rotation, $\boldsymbol{\omega}$, are known or can be computed by some other means [2]. In the case where the true sensor translation $\boldsymbol{U}$ is known, we can compute the absolute 3D depth map; otherwise we can compute relative depth. Interpolation of surface orientation values at non-pixel image locations is avoided by assuming local planarity, a seemingly reasonable assumption everywhere in the image except at depth discontinuities. Our use of planarity is different than Faugeras and Lustman's [5]: we use it to avoid having to compute non-pixel correspondences. Given image velocity $\boldsymbol{v}(\boldsymbol{Y}, t)$ we believe it valid most of the time to assume the correspondence is at round$(\boldsymbol{Y} + \boldsymbol{v}\delta t)$ at time $t + \delta t$.

We consider a setup consisting of a single camera taking the images while it is moving through a static 3D scene. The standard image velocity equations [11] relate a velocity vector measured at image location $\boldsymbol{Y} = (y_1, y_2, f) = f\boldsymbol{P}/X_3$, [i.e. the perspective projection of a 3D point $\boldsymbol{P} = (X_1, X_2, X_3)$], to the 3D sensor translation $\boldsymbol{U}$ and 3D sensor rotation $\boldsymbol{\omega}$. We can rewrite the standard equation for the image velocity $\boldsymbol{v} = (v_1, v_2)$ as $\boldsymbol{v}(\boldsymbol{Y}, t) = \boldsymbol{v}_T(\boldsymbol{Y}, t) + \boldsymbol{v}_R(\boldsymbol{Y}, t)$ where $\boldsymbol{v}_T$ and $\boldsymbol{v}_R$ are the translational and rotational components of image velocity:

$$\boldsymbol{v}_T(\boldsymbol{Y}, t) \;=\; A_1(\boldsymbol{Y})\frac{\boldsymbol{U}}{X_3} \quad \text{and} \quad \boldsymbol{v}_{\mathrm{R}}(\boldsymbol{Y}, \mathrm{t}) \;=\; \mathrm{A}_2(\boldsymbol{Y})\boldsymbol{\omega}(\mathrm{t}) \tag{1}$$

and

$$A_1(\boldsymbol{Y}) \;=\; \begin{pmatrix} -f & 0 & y_1 \\ 0 & -f & y_2 \end{pmatrix} \text{ and } \mathrm{A}_2(\boldsymbol{Y}) \;=\; \begin{pmatrix} \frac{y_1 y_2}{f} & -(f + \frac{y_1^2}{f}) & y_2 \\ (f + \frac{y_2^2}{f}) & -\frac{y_1 y_2}{f} & -y_1 \end{pmatrix}. \tag{2}$$

We define the depth scaled camera translation as

$$\boldsymbol{u}(\boldsymbol{Y}, t) \;=\; \frac{\boldsymbol{U}(t)}{||\boldsymbol{P}(t)||_2} \;=\; \hat{u}\mu(\boldsymbol{Y}, t), \tag{3}$$

where $\hat{u} = \hat{U} = (u_1, u_2, u_3)$ is the normalized direction of translation and $\mu(\boldsymbol{Y}, t) = \frac{||\boldsymbol{U}||_2}{||\boldsymbol{P}||_2} = \frac{f||\boldsymbol{U}||_2}{|X_3||\boldsymbol{Y}||_2}$ is the depth scaled sensor speed at $\boldsymbol{Y}$ at time t. We refer to μ as **scaled** speed. Note that translational image velocity, $\boldsymbol{v}_T$, is bilinear in $\hat{u}$ and μ, making the image velocity equations non-linear. The focal length f is assumed to be known via some camera calibration scheme. If we define 2 vectors:

$$\boldsymbol{r}(\boldsymbol{Y}) = (r_1, r_2) = |\boldsymbol{v} - A_2(\boldsymbol{Y})\boldsymbol{\omega}| \quad \text{and} \tag{4}$$

$$\boldsymbol{d}(\boldsymbol{Y}) = (d_1, d_2) = |A_1(\boldsymbol{Y})\hat{u}|\frac{||\boldsymbol{Y}||_2}{f}, \tag{5}$$

where $|\boldsymbol{A}|$ means each element in the vector is replaced by its absolute value. Then we can solve for μ from the image velocity equation, which can now be written as:

$$\boldsymbol{r} - \boldsymbol{d}\mu = (r_1, r_2) - (d_1, d_2)\mu = 0. \tag{6}$$

μ has solutions $\frac{r_1}{d_1}$ or $\frac{r_2}{d_2}$ or (best) a weighted average:

$$\mu = \frac{\left(\frac{r_1|v_1|}{d_1} + \frac{r_2|v_2|}{d_2}\right)}{|v_1| + |v_2|}. \tag{7}$$

We use the magnitudes of v_1 and v_2 to weight the calculation: we expect μ values computed from the larger velocity component magnitude to be more reliable.

4 Planar Orientation from Relative Depth

We are interested in computing the local surface orientation as a unit normal vector, $\hat{\alpha} = (\alpha_1, \alpha_2, \alpha_3)$ from μ values. Consider two 3D points, $\boldsymbol{P_1} = (X_{11}, X_{12}, X_{13})$ and $\boldsymbol{P_2} = (X_{21}, X_{22}, X_{23})$, with images $\boldsymbol{Y}_1 = \left(\frac{fX_{11}}{X_{13}}, \frac{fX_{12}}{X_{13}}, f\right)$ and $\boldsymbol{Y}_2 = \left(\frac{fX_{21}}{X_{23}}, \frac{fX_{22}}{X_{23}}, f\right)$. If they lie on the same 3D plane then:

$$\frac{\hat{\alpha} \cdot \boldsymbol{Y_1}}{\hat{\alpha} \cdot \boldsymbol{Y_2}} = \frac{X_{32}}{X_{13}}. \tag{8}$$

This equation gives the ratio of the 3^{rd} coordinates (X_3) of two 3D points in terms of their image locations and their planar surface orientation (assuming they lie on a common 3D plane). From the definition of $\mu = \frac{||\boldsymbol{U}||_2}{||\boldsymbol{P}||_2} = \frac{f||\boldsymbol{U}||_2}{|X_3|||\boldsymbol{Y}||_2}$ we can write

$$X_3 = \frac{f||\boldsymbol{U}||_2}{\mu||\boldsymbol{Y}||_2}. \tag{9}$$

yielding:

$$\frac{\hat{\alpha} \cdot \boldsymbol{Y_1}}{\hat{\alpha} \cdot \boldsymbol{Y_2}} = \frac{\mu_1||\boldsymbol{Y}_1||_2}{\mu_2||\boldsymbol{Y}_2||_2}. \tag{10}$$

From the planar equation $\hat{\alpha} \cdot \boldsymbol{P} = \hat{\alpha} \cdot \frac{X_3}{f}\boldsymbol{Y} = c$ and using equation (9) we obtain:

$$\hat{\alpha} \cdot \boldsymbol{Y} = \frac{c\mu||\boldsymbol{Y}||_2}{||\boldsymbol{U}||_2} \tag{11}$$

We can solve for $\frac{\hat{\alpha}}{c}$ by setting up a linear system of equations, one for each pixel in a $n \times n$ neighbourhood where planarity has been assumed and using a standard least squares solution method [14].

5 The Overall Calculation

If we assume that $\hat{u}$ (or indeed $\boldsymbol{U}$) and $\boldsymbol{\omega}$ are knowns, we need only concern ourselves with the surface orientation step of the calculation.

At the initial time, $t = 1$:

1. Given $\hat{u}$ and $\boldsymbol{\omega}$, we compute all the μ's as described above (see [2] for one way of computing $\hat{u}$ and $\boldsymbol{\omega}$).
2. In each $n \times n$ neighbourhood centered at a pixel (i, j) we compute $(\frac{\hat{\alpha}}{c})_{(i,j)}$ at that pixel using equations (7) and (11). We call these computed $\frac{\hat{\alpha}}{c}$'s the measurements and denote them as $\boldsymbol{g}_{M_{(i,j)}}$.

Given these measurements, $\boldsymbol{g}_{M_{(i,j)}}$, we can recompute the $\mu_{(i,j)}$'s as:

$$\mu(i,j) = \frac{(\boldsymbol{g}_{M_{(i,j)}} \cdot \boldsymbol{Y}_{(i,j)})||\boldsymbol{U}||_2}{||\boldsymbol{Y}_{(i,j)}||_2} \tag{12}$$

The recomputed μ values are more "smoothed" than the actual measurements and better represent the scene shape. These $\mu(i,j)$ values are currently the best estimate of the scene's dense shape. Note that we can obtain μ values for pixels with no optical flow, these are computed from the image velocities in its neighbourhood (which are assumed to result from the same local planar patch).

At time $t = 2$:

1. Given the measurements or best estimates of $\hat{u}$ and $\boldsymbol{\omega}$, we compute μ at each pixel location and then compute all $\boldsymbol{g}_{M_{(i,j)}}$'s in the same way described above for the new optical flow field. Using the image velocity measurements at time $t = 2$, we use the best estimate of surface orientation at time $t = 1$ at location $\boldsymbol{Y} - \boldsymbol{v}$ ($\Delta t = 1$) plus the measurement at $\boldsymbol{Y}$ and its covariance matrix to obtain a new best estimate at $\boldsymbol{Y}$ at time $t = 2$. We do this at all $\boldsymbol{Y}$ locations (where possible), recompute the μ values via equation (12) and output these as the 3D shape of the scene.

At time $t = i$ we proceed as for time $t = 2$, except we use the best μ estimates from time $t = i - 1$ instead of time $t = 1$ in the Kalman filter updating.

Note that if the true sensor translation $\boldsymbol{U}$ is known, the absolute 3D depth X_3 can be computed everywhere from the filtered μ values:

$$X_3 = \frac{f||\boldsymbol{U}||_2}{\mu||\boldsymbol{Y}||_2}. \tag{13}$$

6 The Kalman Filter Equations

We note here that the components of $\frac{\hat{\alpha}}{c}$ in equation (11) are not independent, thus we have a covariance matrix with non-zero off diagonal elements in the Kalman filter equations. [In [2], the components of all 2D and 3D vectors, $\hat{u}$ and $\boldsymbol{\omega}$ respectively were treated as 1D variables in the Kalman filter framework.]

If we assume $\hat{u}$ and $\boldsymbol{\omega}$ are known, we can compute $\frac{\hat{\alpha}}{c}$ at each pixel (i,j) as outlined above, assuming local planarity. For the purposes of understanding the equations below we will subscript symbols with a M to indicate measured quantities (computed from the image velocities), P to indicate predicted quantities and C to indicate the computed quantities (the current best estimates). We use $\boldsymbol{g}$ to denote $\frac{\hat{\alpha}}{c}$. It is a 3D quantity, so we need an initial predicted value and an initial covariance matrix at time $t = 0$:

$$\boldsymbol{g}_{P_{(i,j)}} = \boldsymbol{0}, \tag{14}$$

$$C_{P_{(i,j)}} = \begin{bmatrix} \infty & 0 & 0 \\ 0 & \infty & 0 \\ 0 & 0 & \infty \end{bmatrix}. \tag{15}$$

This definition of C_P says that initially the coefficients of $\boldsymbol{g}$ are independent and we have no confidence in their estimates.

For each time $t = 1, 2, 3, ...$ at all pixels (i, j), we use equations (7) and (11) to make a measurement of $\boldsymbol{g}_{M_{(i,j)}}$ and an estimate of its covariance matrix $C_{M_{(i,j)}}$, respectively. Then using the previous best surface orientation estimate at time $t - \Delta t$ at image location $\boldsymbol{Y} - \boldsymbol{v}\Delta t$, denoted by (i^-, j^-), the Kalman filter equations are computed [12] as follows:

$$K_{(i,j)} = C_{P_{(i^-,j^-)}} \left[C_{P_{(i^-,j^-)}} + C_{M_{(i,j)}} \right]^{-1}, \tag{16}$$

$$\boldsymbol{g}_{C_{(i,j)}} = \boldsymbol{g}_{P_{(i^-,j^-)}} + K_{(i,j)} \left(\boldsymbol{g}_{M_{(i,j)}} - \boldsymbol{g}_{P_{(i^-,j^-)}} \right) \quad \text{and} \tag{17}$$

$$C_{C_{(i,j)}} = C_{P_{(i^-,j^-)}} - K_{(i,j)} C_{P_{(i^-,j^-)}}. \tag{18}$$

Because $\boldsymbol{g}$, i.e. $\frac{\hat{\alpha}}{c}$, can be rotated by sensor rotation, the predicted values $\boldsymbol{g}_{P_{(i,j)}}$ must take this into account in their update:

$$\boldsymbol{g}_{P_{(i,j)}} = R(\boldsymbol{\omega}, t + \Delta t) R^T(\boldsymbol{\omega}, t) \boldsymbol{g}_{C_{(i,j)}} \quad \text{and} \tag{19}$$

$$C_{P_{(i,j)}} = C_{C_{(i,j)}}. \tag{20}$$

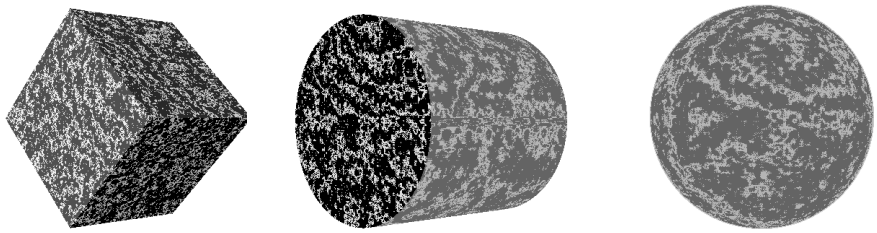
7 Abnormal Situations and Their Resolution

While tracking individual surface orientations, a number of situations may arise. It is possible that:

1. When tracking a surface orientation at a moving pixel, no surface orientation can be computed at the latest time, in which case tracking stops.
2. A new, untracked surface orientation can be computed, in which case tracking starts.
3. A surface orientation is tracked to a wrong pixel, in which case the tracking continues from that wrong pixel as no error recovery is possible or detectable.
4. The surface orientations at two different pixels with (perhaps) different surface orientations track to the same pixel, in which case the surface orientations are combined and then tracked as a single surface orientation (a weighted average using the covariance matrices as weighting matrices).

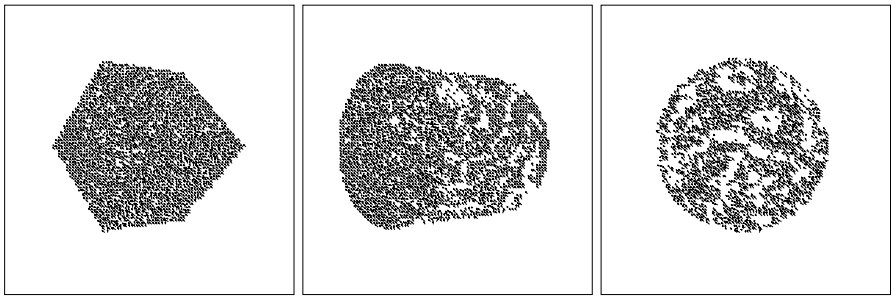
8 Generation of Synthetic Test Image Sequences

For the experiments, we use synthetic images because that allows for a quantitative error analysis on the estimated depth values. We generate a 30 image sequence of 512×512 images using each of three different 3D objects: a sphere, a cube and a cylinder (Figures 1a, 1b and 1c). For each experiment, a sequence of images of one of these three objects is generated while the synthetic camera is moving in a known way and the object is rendered by perspective projection onto the image plane. In all the experiments we performed, all the image sequences were generated using a camera translation of $(1, 1, 0)$ with a focal length of 1000. All the objects have a marble texture to facilitate optical flow computation. Although the synthetic images themselves are error free, the optical flow computed using them is not, as can be seen in Figure 2a through 2c.



(a) (b) (c)

Fig. 1. Synthetic test data: (a) A marble-texture cube with sides of length 300 with its center located at (0,0,1500), (b) A marble-texture cylinder with two end spheres of radius 200 and a wall of length 400 with its center located at (0,0,1500) and (c) A marble-texture sphere of radius 200 with its center located at (0,0,1300).



(a) (b) (c)

Fig. 2. The optical flow of the (a) cube, (b) cylinder and (c) sphere.

9 Experimental Results and Discussion

Error was measured for relative depth μ using the exact μ values: hence the use of the term absolute relative error. Optical flow was computed using (see Figures 2a, 2b and 2c) Lucas and Kanade's algorithm [8] with differentiation performed as proposed by Simoncelli [13]. Figures 3a to 3c shows the histograms of the error distribution for the cube data for the 1^{st}, 22^{nd} and 25^{th} frames. Figures 4a to 4c shows the histograms of the error distribution for the cylinder data for the 1^{st}, 22^{nd} and 25^{th} frames. Figures 5a to 5b shows the histograms of the error distribution for the cylinder data for the 1^{st}, 17^{th} and 25^{th} frames. Note that at depth discontinuities, where the local planarity assumption is definitely violated, no depth is recovered as the Kalman filter rejects those values as unreliable. This suppose image velocity can be computed at a depth discontinuity. Normally Lucas and Kanade optical flow does not yield such velocity values.

Table 1 shows the average absolute relative error for the 3 objects for a number of ranges. The quantitative error results were best for the cube (the most planar) and worst for the sphere (the most curved). Obviously, the planarity assumption is not always valid and we plan on replacing this with a higher order

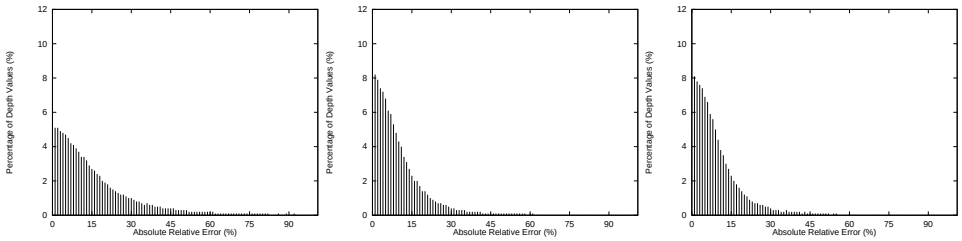


Fig. 3. The histogram of the absolute relative errors of the estimated depth values at the (a) 1^{st}, (b) 22^{nd} and (c) 24^{th} images of the cube sequence using our approach.

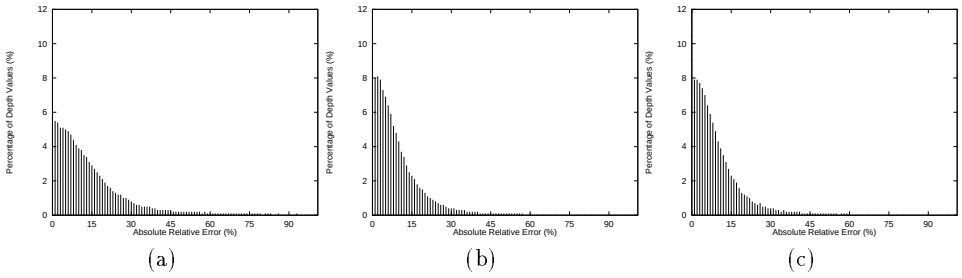


Fig. 4. The histogram of the absolute relative errors of the estimated depth values at the (a) 1^{st}, (b) 22^{nd} and 24^{th} images of the cylinder sequence using our approach.

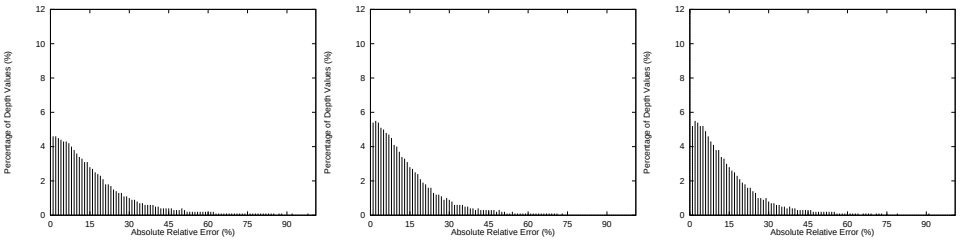
parametric model. We have already seen that higher order parametric models are sometimes necessary to get better optical flow [3].

10 Conclusions

We have presented a new algorithm to compute dense accurate depth using a Kalman filter framework. We need to test our algorithm for camera motions including rotation and to use real data (preferably with ground truth). Lastly, we plan to integrate this algorithm into the Kalman filter based motion and structure algorithm designed earlier [2].

References

1. Barron J.L., Jepson A.D. and Tsotsos J.K. (1990), "The Feasibility of Motion and Structure From Noisy Time-Varying Image Velocity Information", Int. Journal of Computer Vision, 5:3, pp. 239-269.
2. Barron J.L. and Eagleson R. (1996), "Recursive Estimation of Time-Varying Motion and Structure Parameters", *Pattern Recognition*, Vol. 29, No. 5, May, pp. 797-818.



(a) (b) (c)

Fig. 5. The histogram of the absolute relative errors of the estimated depth values at the 1^{st}, (b) $17th$ and (c) $24th$ images of the sphere sequence using our approach.

Percent Relative Error	>15%	5% - 15%	<5%
1st depth map (Cube)	42.92%	34.70%	22.38%
22nd depth map (Cube)	21.47%	41.80%	36.73%
25th depth map (Cube)	20.47%	42.35%	37.18%
1st depth map (Cylinder)	49.85%	29.90%	20.25%
22nd depth map (Cylinder)	37.38%	33.81%	28.81%
24th depth map (Cylinder)	36.65%	34.33%	29.02%
1st depth map (Sphere)	65.66%	20.79%	13.55%
17th depth map (Sphere)	61.25%	23.40%	15.35%
24th depth map (Sphere)	61.29%	23.01%	15.7%

Table 1. The percentage of the estimated depth values that have certain absolute relative errors in the experiments for the cube, cylinder and sphere.

3. Barron J.L. and Khurana M. (1997), "Determining optical flow for large motions Using Parametric Models in a Hierarchical Framework, Vision Interface, Kelowna, B.C., May, pp47-56.
4. Black M.J. and Anandan P. (1996), "The Robust Estimation of Multiple Motions: Parametric and Piecewise-Smooth Flow fields", Computer Vision and Image Understanding, 63:1, pp. 75-104.
5. Faugeras O.D. and Lustman F. (1985), "Let Us Suppose that the World is Piecewise Planar", 3^{rd} Intl. Symposium on Robotics, France, pp. 33-39
6. Gelb A. (1974), "Applied Optimal Estimation", MIT Press.
7. Kalman, R. E. (1960), "A new approach to linear filtering and prediction problems", Trans. of the ASME - Journal of Basic Engineering, pp35-45.
8. Lucas, B. and Kanade, T. (1981), "An iterative image registration technique with an application to stereo vision", *Proc. DARPA IU Workshop*, pp. 121-130.
9. Heel J. (1990), "Direct Dynamic Motion Vision", *Proc. IEEE Conf. Robotics and Automation*, pp. 1142-1147.
10. Hung Y.S. and Ho H.T. (1999), "A Kalman Filter Approach to Direct Depth Estimation Incorporating Surface Structure", *IEEE PAMI*, June, pp. 570-576.
11. Longuet-Higgins H.-C. and Prazdny K, (1980), "The Interpretation of a Moving Retinal Image", Proc. R. Soc. Lond B 208, pp. 385-397.

12. Matthies L., Szeliski R. and Kanade T. (1989), "Kalman Filter-Based Algorithms for Estimating Depth from Image Sequences", Int. J. Computer Vision 3:3, pp. 209-238.
13. Simoncelli E.P. (1994), "Design of multi-dimensional derivative filters", IEEE Int. Conf. Image Processing, Vol. 1, pp790-793.
14. Press W.H., Flannery B.P., Teukolsky S.A., and Vetterling W.T. (1992), "Numerical Recipes in C (2^{nd} edition)", Cambridge University Press, pp671-675.
15. Xiong Y. and Shafer S. (1995), "Dense Structure from a Dense Optical Flow Sequence", Int. Symposium on Computer Vision, Coral Gables, Florida, pp1-6.