MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1594 November, 1996

Direct methods for estimation of structure and motion from three views

Gideon P. Stein and **Amnon Shashua**
gideon@ai.mit.edu shashua@cs.huji.ac.il

Abstract

We describe a new direct method for estimating structure and motion from image intensities of multiple views. We extend the direct methods of [7] to three views. Adding the third view enables us to solve for motion, and compute a dense depth map of the scene, directly from image spatio-temporal derivatives in a linear manner without first having to find point correspondences or compute optical flow.

We describe the advantages and limitations of this method which are then verified through simulation and experiments with real images.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for this research was provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-94-01-0994. G.P.S. is supported by Alphatech/ DARPA "Automatic Target Recognition" project 95009-5381 and TASC/ DARPA "MSTAR Match Module" project J-08011-S95042. A.S. is supported by US-IS Binational Science Foundation 94-00120/2 and by the European ACTS project AC074 VANGUARD.

1 Introduction

In this paper we present a new method for computing motion and dense structure from three views. This method can be viewed as an extension of the 'direct methods' of Horn and Weldon [7] from two views (one motion) to three views (two motions). These methods are dubbed 'direct methods' because they do not require prior computation of optical flow. As with other gradient methods we assume small image motions on the order of a few pixels.

Applying the constant brightness constraint [6] to the trilinear tensor of Shashua and Werman [12, 15] results in an equation relating camera motion and calibration parameters to the image gradients (first order only). We get one equation for each point in the image and we have a fixed number of parameters which results in a highly over-constrained set of equations. Starting with the general uncalibrated model we proceed through a hierarchy of reduced models first by assuming calibrated cameras and then by assuming the Longuett-Higgins and Prazdny small motion model [9]. This is described in section (2). We then show how to solve the simplified model for the motion parameters (section 3).

This method has advantages over both optical flow methods [9][10] and feature based methods [15]. We combine the information from all the points in the image and thus we avoid the aperture problem which makes computation of optical flow difficult. There is also no need to explicitly define feature points. Points which have little or no gradient simply contribute little to the least squares estimation. Information from all points that have gradients is used.

These advantages are highlighted in a scene where we have a set of vertical bars in front of a set of horizontal bars and behind everything a uniform background. In this case optical flow methods will fail because of the straight bars and the aperture problem. Point feature based methods fail because the intersections of the lines in the image, which will be detected as 'features' do not correspond to real features in space. Many natural scenes such as tree branches or man made objects such as window frames, lamp posts and fences often give rise to these problems.

Section (4) describes some of the implementation details needed to make the method work. Sections (5) and (6) show the results of applying this method to simulated and real images. The method is shown to produce useful results in both camera motion and depth estimation. Section (7) discusses some of the open questions and suggests possible solutions.

1.1 Previous Work

The 'direct methods' were pioneered by Horn and Weldon in [7]. Using only a single image pair one has N equations in $N+6$ unknowns, where N is the number of points in the image, so some added constraint is needed. Negahdaripour and Horn [11] present a closed form solution assuming a planar or quadratic surface. McQuirk [8] shows that, assuming a pure translation model, the subset of the image points with a nonzero spatial derivative but a zero time derivative gives the direction of motion. The FOE is on a line perpendicular to the gradient at these points. But by using only this subset of the points we are throwing away the information from most of the image. Heel [5] uses multiple images from an image sequence. He then employs Kalman filters to build up a structure model from more than one image pair but the core computation is fundamentally the same single image pair computation.

This work is based on the work of Shashua and Hanna [14]. Here we describe the results of implementing these ideas in practice. During the course of implementation various subtleties and limitations were discovered.

2 Mathematical Background

2.1 Notations

The pin-hole camera model in homogeneous coordinates is represented by a 3×4 camera matrix A producing the following relation: $p \cong A\boldsymbol{x}$ where $\boldsymbol{x}$ varies in 3D space, p varies over the 2D image plane, and $\cong$ denotes equality up to scale. Since only relative camera positioning can be recovered from image measurements, the first camera matrix (camera 0) can be represented by $[I;0]$. In a pair of views, $p = [I;0]\boldsymbol{x}$ and $p' \cong A\boldsymbol{x}$, the left 3×3 minor of A stands for a 2D projective transformation of the chosen plane at infinity and the fourth column of A stands for the epipole (the projection of the center of camera 0 on the image plane of camera 1). In particular, in a calibrated setting the 2D projective transformation is the rotational component of camera motion and the epipole is the translational component of camera motion.

We will occasionally use tensorial notations, which are briefly described next. We use the covariant-contravariant summation convention: a point is an object whose coordinates are specified with superscripts, i.e., $p^i = (p^1, p^2, ...)$. These are called contravariant vectors. An element in the dual space (representing hyperplanes — lines in $\mathcal{P}^2$), is called a covariant vector and is represented by subscripts, i.e., $s_j = (s_1, s_2, ...)$. Indices repeated in covariant and contravariant forms are summed over, i.e., $p^i s_i = p^1 s_1 + p^2 s_2 + ... + p^n s_n$. This is known as a contraction. For example, if p and s represent a coincident point and line in $\mathcal{P}^2$, then $p^i s_i = 0$. Vectors are also called 1-valence tensors. 2-valence tensors (matrices) have two indices and the transformation they represent depends on the covariant-contravariant positioning of the indices. For example, a_i^j is a mapping from points to points, and hyperplanes to hyperplanes, because $a_i^j p^i = q^j$ and $a_i^j s_j = r_i$ (in matrix form: $Ap = q$ and $A^\top s = r$); a_{ij} maps points to hyperplanes; and a^{ij} maps hyperplanes to points. When viewed as a matrix the row and column positions are determined accordingly: in a_i^j and a_{ji} the index i runs over the columns and j runs over the rows, thus $b_j^k a_i^j = c_{ki}$ is $BA = C$ in matrix form. An outer-product of two 1-valence tensors (vectors), $a_i b^j$, is a 2-valence tensor c_i^j whose i, j entries are $a_i b^j$ — note that in matrix form $C = ba^\top$.

Matching image points across three views will be denoted by p, p', p''; the coordinates will be referred to as p^i, p'^j, p''^k, or alternatively as non-homogeneous image coordinates $(x, y), (x', y'), (x'', y'')$.

2.2 Stratification of Direct Motion Models

Three views, $p = [I; 0]\boldsymbol{x}, p' \cong A\boldsymbol{x}$ and $p'' \cong B\boldsymbol{x}$, are known to produce four trilinear forms whose coefficients are arranged in a tensor representing a bilinear function of the camera matrices A, B:

$$\alpha_i^{jk} = t'^j b_i^k - t''^k a_i^j \tag{1}$$

where $A = [a_i^j, t'^j]$ (a_i^j is the 3×3 left minor and t' is the fourth column of A) and $B = [b_i^k, t''^k]$. The tensor acts on a triplet of matching points in the following way:

$$p^i s_j^\mu r_k^\rho \alpha_i^{jk} = 0 \tag{2}$$

where s_j^μ are any two lines (s_j^1 and s_j^2) intersecting at p', and r_k^ρ are any two lines intersecting p''. Since the free indices are μ, ρ each in the range 1,2, we have 4 trilinear equations (which are unique up to linear combinations). More details can be found in [4, 12, 15, 13].

Geometrically, a trilinear matching constraint is produced by contracting the tensor with the point p of image 0, *some* line coincident with p' in image 1, and *some* line coincident with p'' in image 2. In particular, we may use the tangent to the iso-brightness contour at p' and p'', respectively. In other words, one can recover in principle the camera matrices across three views in the context of the "aperture" problem, as noticed by [16]. Alternatively, if we represent the tangent to the corresponding iso-brightness contours by the instantaneous spatio-temporal derivatives at p (shown next), we will get a constraint equation involving the unknowns α_i^{jk} and the spatio-temporal derivatives at each pixel — the constraint is linear in the unknowns. This constraint was introduced by [14] under the name "Tensor Brightness Constraint". This is briefly derived next.

We can describe any line, in particular a line through p', as a linear combination of the vertical $(1, 0, -x')$ and horizontal $(0, 1, -y')$ lines. Let the coefficients of the linear combination be the components of the image gradient I_x, I_y at (x, y) in image 0, then the line s' has the form:

$$S' = \begin{pmatrix} I_x \\ I_y \\ -x'I_x - y'I_y \end{pmatrix}$$

The contribution of x', y' can be removed by using the constant brightness equation due to [6]:

$$u'I_x + v'I_y + I'_t = 0 \tag{3}$$

where $u' = x - x'$, $v' = y - y'$ and I'_t is the discrete temporal derivative at (x, y), i.e., $I_1(x, y) - I_0(x, y)$ where I_1 and I_0 are the image intensity values of the second and first images, respectively. Following substitution we obtain,

$$S' = \begin{pmatrix} I_x \\ I_y \\ I'_t - xI_x - yI_y \end{pmatrix} \tag{4}$$

Likewise, the tangent to the iso-brightness contour at p'' is,

$$S'' = \begin{pmatrix} I_x \\ I_y \\ I''_t - xI_x - yI_y \end{pmatrix} \tag{5}$$

where I''_t is the temporal derivative between images 0 and 2. The tensor brightness constraint is therefore:

$$\boxed{s''_k s'_j p^i \alpha_i^{jk} = 0.} \tag{6}$$

We have one such equation for each point on the image where s''_k and s'_j can be computed from the image gradients and $p = (x, y, 1)^T$ are the (projective) image coordinates of the point in image 0. We wish to solve for α_i^{jk} which combines the motion and camera parameters

Starting from the general model (27 parameter model) of the constraint equation one can introduce a hierarchy of reduced models, as follows. By enforcing small-angle rotation on the camera motions, i.e., $A = [I + [w']_x; t']$ and $B = [I + [w'']_x; t'']$ where w', w'' are the angular velocity vectors and $[\cdot]_x$ is the skew-symmetric matrix of vector products, the tensor brightness constraint is reduced to a 24-parameter model which in matrix form looks like:

$$I''_t {S'}^\top t' - I'_t {S''}^\top t'' + {S'}^\top [t' {w''}^\top] V'' - {S''}^\top [t'' {w'}^\top] V' = 0, \tag{7}$$

where

$$V' = p \times S' = \begin{pmatrix} -I_y + y(I'_t - xI_x - yI_y) \\ I_x - x(I'_t - xI_x - yI_y) \\ xI_y - yI_x \end{pmatrix}$$

and

$$V'' = p \times S'' = \begin{pmatrix} -I_y + y(I''_t - xI_x - yI_y) \\ I_x - x(I''_t - xI_x - yI_y) \\ xI_y - yI_x \end{pmatrix}$$

If, in addition, we enforce infinitesimal translational motion (the Longuett-Higgins & Prazdny [9] motion model), which results in the image motion equations:

$$u' = \frac{1}{z}(t'_1 - xt'_3) - w'_3 y + w'_2(1 + x^2) - w'_1 xy \tag{8}$$

$$v' = \frac{1}{z}(t'_2 - yt'_3) + w'_3 x - w'_1(1 + y^2) + w'_2 xy$$

then S has the simpler form:

$$S = \begin{pmatrix} I_x \\ I_y \\ -xI_x - yI_y \end{pmatrix} \tag{9}$$

and

$$V = \begin{pmatrix} -I_y - y(xI_x + yI_y) \\ I_x + x(xI_x + yI_y) \\ xI_y - yI_x \end{pmatrix} \tag{10}$$

and we obtain a 15-parameter model of the following form:

$$I''_t S^T t' - I'_t S^T t'' + S^T [t' w''^T - t'' w'^T] V = 0 \tag{11}$$

We now have one such equation for each point in the image. It is a set of bilinear equations in the unknowns t', t'', w', w''. After solving for the camera motions (to be described later in the paper) we can solve for the dense depth map from the equations:

$$KS^\top t' + V^\top w' + I'_t = 0 \tag{12}$$

$$KS^\top t'' + V^\top w'' + I''_t = 0 \tag{13}$$

where $K = \frac{1}{z}$ denotes inverse of the depth at each pixel location. Equations (12)(13) are obtained by substituting (eq. 8) in equation (3) and rearranging the terms. See [7] for more details.

3 Solving the bilinear equation

3.1 The pure translation case

In the pure translation case equation (11) becomes:

$$I_t'' S^T t' - I_t' S^T t'' = 0 \tag{14}$$

We have one such equation for each image point and we can write it out in the matrix form:

$$At = 0 \tag{15}$$

where:

$$t = \begin{pmatrix} t_x' & t_y' & t_z' & t_x'' & t_y'' & t_z'' \end{pmatrix}^T \tag{16}$$

and A is an $N \times 6$ matrix with the $i'th$ row (corresponding to the $i'th$ pixel) given by:

$$\begin{pmatrix} I_t'' S_{i1} & I_t'' S_{i2} & I_t'' S_{i3} & I_t' S_{i1} & I_t' S_{i2} & I_t' S_{i3} \end{pmatrix} \tag{17}$$

Since we wish to avoid the trivial solution $t = 0$ we will add the constraint $||t|| = 1$. The least squares problem now maps to problem of finding $||t|| = 1$ that minimizes:

$$t^T A^T A t = 0 \tag{18}$$

The solution is the eigenvector of $A^T A$ corresponding to the smallest eigenvalue.

3.2 The general small motion case

In the general case we are confronted with the bilinear equation (11). There is no standard way to solve these bilinear problems and we have chosen to treat the 6 translation parameters and the 9 outer product terms $[t'w''^T - t''w'^T]$ as 15 intermediate parameters. Although they are not independent if we act as if they are, then they can be solved for as in section (3.1) but with a 15×15 matrix A. After recovering the 15 intermediate parameters we compute w' and w'' from $[t'w''^T - t''w'^T]$ using the computed translation.

This is the general idea but it is not that simple. If we consider the $N \times 15$ matrix A we note that the columns of A are not independent.

Lemma 1 *The matrix A is of rank ≤ 13.*

Proof: The 7th, 11th and 15th elements along each row add up to zero.

$$\begin{aligned} A(i,7) + A(i,11) + A(i,15) &= \\ &= S_{i1}V_{i1} + S_{i2}V_{i2} + S_{i3}V_{i3} \\ &= S \cdot V \\ &= S \cdot P \times S \\ &= 0 \end{aligned}$$

Therefore the vector

$$c_0 = (0,0,0,0,0,0,1,0,0,0,1,0,0,0,1)^T \tag{19}$$

is in the null space of A. The correct solution vector is also in the null space of A. Therefore the null space of A has $rank \geq 2$ and A is of $rank \leq 13$. ∎

The matrix A is in theory of rank 13 not 14. Suppose we found a vector b_0 in the null space of A such that $c_0 \times b_0 \neq 0$. The two vectors c_0 and b_0 span the null space of A and the desired solution vector b is a linear combination of the two:

$$b = b_0 + \alpha c_0 \tag{20}$$

In order to find α given c_0 and b_0 we must apply some constraint. We choose to enforce the constraint that the matrix $[t'w''^T - t''w'^T]$ be of rank 2.

Clearly the choice of α will have no affect the first 6 elements of the vector b. Let us arrange the last 9 elements of b, b_0 and c_0 into the corresponding 3×3 matrices B, B_0 and C_0. We are now looking for an α such that:

$$Rank(B_0 - \alpha C_0) = 2 \tag{21}$$

Since C_0 in our case is the identity matrix, the solution for α is given by the eigenvalues of B_0. We choose the one with the smallest absolute value. The vector b_0 is the eigenvector corresponding to the second smallest eigenvalue of the matrix $A^T A$. (The vector c_0 corresponds to the smallest.)

Based on the previous arguments the algorithm for finding the motion parameters is as follows:

1. Compute the $N \times 15$ matrix A from equation (11).
2. Find the eigenvector corresponding to the second smallest eigenvalue of the matrix $A^T A$. This is b_0.
3. The first 6 elements of b_0 are the two translations, t' and t''.
4. Arrange the other 9 elements of b_0 into a 3×3 matrix B_0.
5. Construct a rank 2 matrix B from B_0:
$$B = B_0 - \alpha I \tag{22}$$
6. Solve:
$$[t'w''^T - t''w'^T] = B \tag{23}$$
for w' and w'', given t' and t'' from step (3).

3.3 The singular case:

This method fails when the two motions are in the same (or opposite) directions. This is obvious in the pure translation case. If the second translation vector t'' is proportional to the translation vector t' then equation (13) is simply a scaled version of equation (12) adding no new information. Under current research is a way to overcome this problem (section 7) and initial results look promising.

4 Implementation details

4.1 Iterative refinement and coarse to fine processing

The constant brightness constraint is a linearized form of the Sum Square Difference (SSD) criteria. The linear solution can be thought of as a single iteration of Newton's method applied to the problem. Iterative refinement is performed as follows: First one calculates motion and depth using the above equations. Then, using the depth and motion, images 1 and 2 are warped towards image 0. A correction to the depth and motion is computed using the warped images. In the ideal case, as the final result the warped images should appear nearly identical to image 0.

We have to be careful here. If at every iteration we compute only the correction to the motion we will end up trying to compute very small values of δt. At this point our equations will be badly conditioned and highly affected by noise. The idea is to devise our iterations in such a way as to always be computing a gradually improving estimate of the translation t rather than the incremental improvements, but using temporal derivatives from closer and closer images.

Let Ψ_0, Ψ_1 and Ψ_2 be the three image. Assume we have $\hat{K}$, $\hat{t}^j$, $\hat{w}^j$, from the previous iteration. The image motions $\hat{u}^j$ and $\hat{v}^j$ can be computed using $\hat{K}$, $\hat{t}^j$ and $\hat{w}^j$ in equation (8). These are then used to warp images Ψ_1 to $\hat{\Psi}_1$ and Ψ_2 to $\hat{\Psi}_2$. After warping, the images satisfy the brightness constraint equation:

$$I_x du' + I_y dv' + \hat{I}_t' = 0 \qquad (24)$$
$$I_x du'' + I_y dv'' + \hat{I}_t'' = 0$$

where the temporal derivatives at each pixel are given by:

$$\hat{I}_t' = \hat{\Psi}_1 - \Psi_0 \qquad (25)$$
$$\hat{I}_t'' = \hat{\Psi}_2 - \Psi_0$$

and du^j, dv^j are the (still unknown) differences between computed image motions and the real image motions:

$$du^j = u^j - \hat{u}^j \qquad (26)$$
$$dv^j = v^j - \hat{v}^j$$

Let:

$$\alpha^j = I_x \hat{u}^j + I_y \hat{v}^j \qquad (27)$$

which can also be written as:

$$\alpha^j = \hat{K} S^T \hat{t}^j + V^T \hat{w}^j \qquad (28)$$

Substituting equations (27) and (26) in equation (24) we get:

$$I_x u + I_y v + (\hat{I}_t' - \alpha') = 0 \qquad (29)$$
$$I_x u + I_y v + (\hat{I}_t'' - \alpha'') = 0$$

Substituting equation (8) in equation (29) we get modified versions of the equations (12) and (13)

$$K S^T t' + V^T w' + (I_t' - \alpha') = 0 \qquad (30)$$
$$K S^T t'' + V^T w'' + (I_t'' - \alpha'') = 0$$

We start our first iteration with $\hat{K}$, $\hat{t}^j$, $\hat{w}^j$ all zero and therefore $\alpha = 0$ as well.

In order to deal with image motions larger than 1 pixel we use a Gaussian pyramid for coarse to fine processing [2][3].

4.2 Computing the depth, smoothing and interpolation.

After recovering the camera motions, equations (12) and (13) give us a way of computing the depth at every point where $S^T t'$ or $S^T t''$ are non zero. In order to combine the information from both images and to interpolate over areas where the image gradients are small we chose an interpolation scheme called Local Weighted Regression. This method was chosen because it is simple to implement but could very well be replaced by other methods such as RBF's, B-splines or thin-plate interpolation.

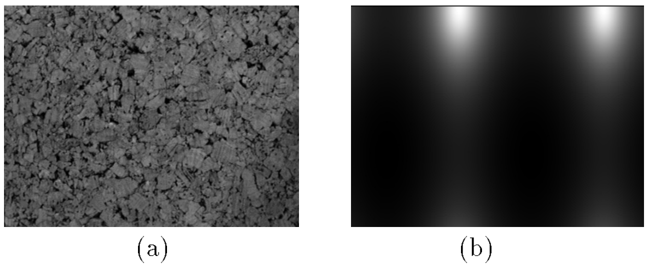


Figure 1: *The texture image (a) and simulated depth map(b) used for simulation experiments.*

Equation (31) shows the cost function used to compute the depth at a given point:

$$\min \arg_K \sum_{x,y \in R} \sum_j \beta(x,y) |S^T t^j|^p \left(K S^T t^j + V^T w^j + I_t^j \right)^2 \qquad (31)$$

The sum is over a region R and over the two motions $j = 1, \ldots, 2$. The windowing function $\beta(x,y)$ allows one to increase the weight of the closer points. We used a function created by convolving two box filters together. It is a crude approximation to a Gaussian. The $|S^T t^j|^p$ term reduces the weight of points which have a small gradient or where the gradient is perpendicular to that camera motion since these cases are highly affected by noise. We used $p = 1$. The size of the region R depends on the amount of smoothing and interpolation required. During the iteration process we used a region of 5×5 or 7×7. In order to get 'prettier' results, after the last iteration, we replaced K which is a locally constant depth model with a locally planar model $K(x,y) = K_x x + K_y y + K_0$ and increased the region of support to 30×30.

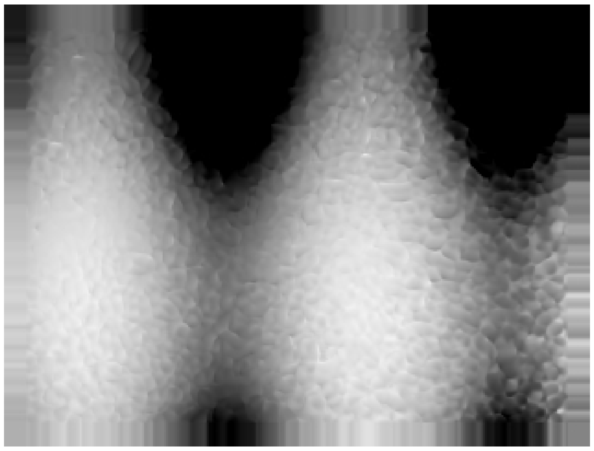
5 Simulation

To test the method we took the 320×240 texture image (fig. 1a) and defined the depth at each point in the image according to the equation:

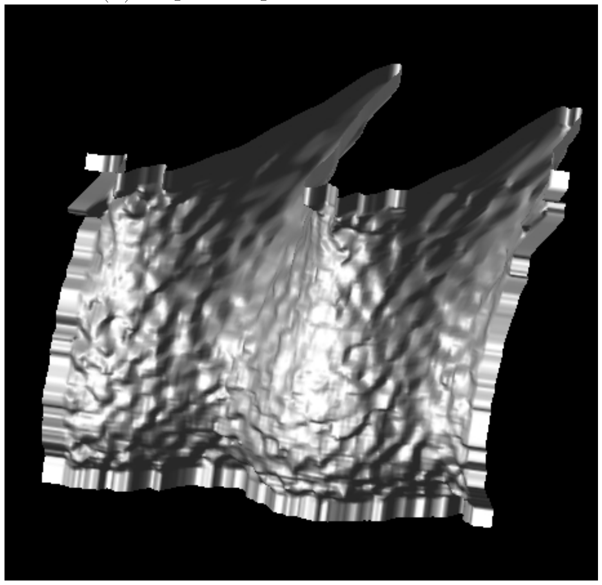
$$Z(x,y) = 1000 + 400 * \left(\sin(\frac{x}{25}) + \sin(\frac{y}{50}) \right) \qquad (32)$$

The depth map is shown in figure (1b). We then warped it according to two given motions and equation (8). We used the original image and the two warped images as input to our algorithm and computed the motion and depth. The motion was chosen so that the image motion was 8 pixels or less. The first image was warped by a translation of $t' = (2000, 0, 500)$ and no rotation. The second image was warped by a rotation $w'' = (0.5, 0, 0)$ and a translation $t'' = (0, 2000, 0)$. The rotation and translation were chosen so that they combined to reduce the overall motion. The focal length was $f = 50$.

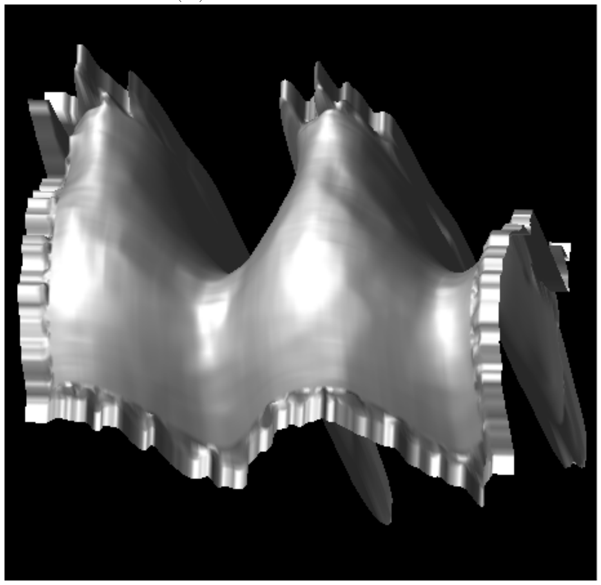
We used 4 levels of coarse to fine processing and 2 iterations at each level. Figure (2a) shows the reconstructed depth map at the finest level. Figure (2b, 2c) shows the 3D rendering of the surface using a local constant depth fit and a local planar fit respectively. Apart from a few outliers around the edges the surface is qualitatively correct. Table (1) shows the given motion and



(a) Depth map - local constant fit.



(b) Local constant fit.



(c) Local planar fit.

Figure 2: *The estimated $K(x,y) = \frac{1}{Z}$ (inverse depth) from simulation.*

Table 1: Estimated camera motion in simulation experiments

Translation		
	real	estimate
FOE 1:	(360, 120)	(369.4, 119.8)
FOE 2:	(160, inf)	(170.8, 6097)
Rotation		
	real	estimate
W1	(0,0,0)	(-0.0018, 0.00034, 0.0012)
W2	(-0.5,0,0)	(-0.48, 0.0055, 0.0096)

Table 2: Image coordinates of two example points in the three images

	Image 0	Image 1	Image 2
Head	(255,206)	(254,195)	(264,206)
Cork	(72,291)	(70,283)	(78,291)

the resulting motion estimates. The results are shown using a 7×7 region for depth estimation. Varying the size of the region from 3×3 to 11×11 had less than 1% effect on the motion estimation.

6 Experiments with real images

6.1 Experimental procedure

The images were taken with a Phillips $\frac{1}{3}inch$ CCD video camera and an $8mm$ lens. Image capture was performed using the SGI Indy built in frame grabber at 640×480 pixel resolution. Figure (3a) shows one of three images used for the experiment. The depth in the image ranged from $450mm$ to $750mm$. The camera was mounted on a lightweight tripod. After taking the first image the camera was lowered $3mm$ and a second image was taken. The tripod was then moved a few millimeters to the right for the third image. No special care was taken to ensure precise motion. Image motions were $6-11pixels$. Table (2) shows the measured image coordinates for two points in the three images: one on the head and one on the cork panel on the image left.

6.2 Results

The results are shown for the case where images were processed using 4 levels of coarse to fine processing with 2 iterations at each level. Varying the number of iterations from 1 through 4 had no qualitative impact but using a single iteration caused a small change in the resulting motion estimates. A 7×7 region was used for the local constant depth fit at all levels. Table (3) shows the estimated camera motions. It shows that the first motion was along the Y axis and the second motion along the X axis and that for both cases the rotation was negligible. This is qualitatively correct. We do not have accurate ground truth estimates.

Figure (3b) shows the recovered depth map (in fact this shows $K(x,y) = \frac{1}{Z}$). Figure (3c) shows a 3D rendering of the surface $K(x,y)$. The K values have been scaled by 100.0. The rendering uses an orthographic projection. In order to get smoother and more visually

Table 3: Motion estimates from real images.

FOE 1	(-603.7, -8055)
FOE 2	(16302, 300.2)
W1	(0.00022, -0.00055, -0.0217)
W2	(-0.00027 -0.00017 -0.00062)

pleasing results a local planar fit was used for the final stage using a 30×30 region of support. The results are shown in figures (4a, 4b). There is noticeable smoothing and overshoots at depth discontinuities and the tip of the nose. In (4b) the texture was removed for clarity.

7 Discussion and future work

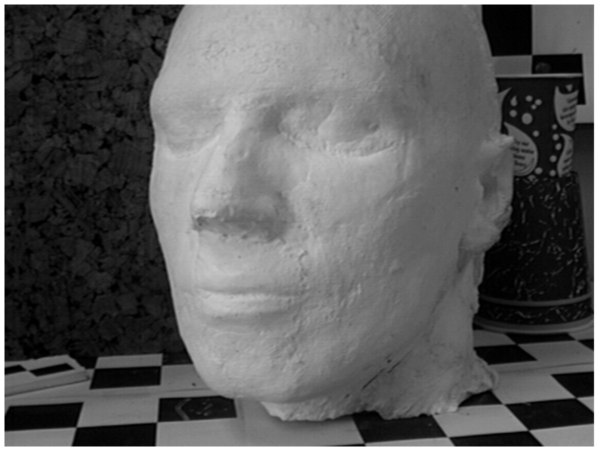
We have presented a new method for recovering structure and motion from 3 views. This method does not require feature correspondence or optical flow. We have shown, using simulation and experiments with real images, that the method can qualitatively recover depth and motion in the general, small motion case. These results are promising but more experiments are needed to test the accuracy of the motion estimation.

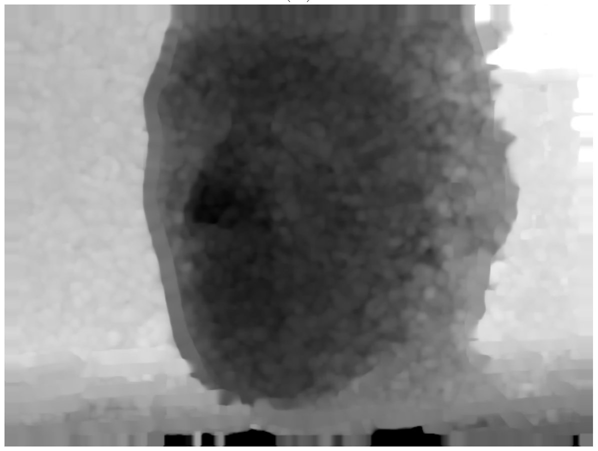
Occlusions do not pose a significant problem in the motion estimation since they take up only a small part of the total image area. The depth estimates obtained along the occlusion boundary will be inaccurate.

In general one can only obtain a depth estimate where there is a gradient. In order to get a dense depth map one must perform some form of interpolation. The local constant or linear fit which we have used is superficially similar to the smoothness assumptions employed by optical flow techniques to overcome the aperture problem. We could benefit from some details of these techniques such as adaptive window size. The main difference is that here, the smoothness does not affect our motion estimation and is for cosmetic reasons, while for optical flow methods the smoothness assumption is key to the whole process. Another difference is that we are smoothing the depth image for which it might make sense to assume a local planar or quadratic model. This is very much application dependent.

There are two important theoretical questions left open. The first involves solving the bilinear equation (11). We have chosen to enforce a rank 2 constraint on the matrix $[t'w''^T - t''w'^T]$ but there might be other possible constraints to use. Furthermore, we make the matrix rank 2 by subtracting αI where α is the eigenvalue closest to zero. But perhaps we should use one of the other two eigenvalues. On the practical side, it might be best to solve equation (11) as a nonlinear optimization problem with the linear solution as the initial guess.

The second open problem deals with the current limitation that the two motions cannot be collinear since this has been shown (section 3.3) to be a singular case. On the other hand no such limitation exists in the discrete case [15]. There is a question whether this is a general phenomena resulting from using the constant brightness equation or whether this is specific to the LH model. There are two avenues to proceed. We can go back to


(a)


(b)

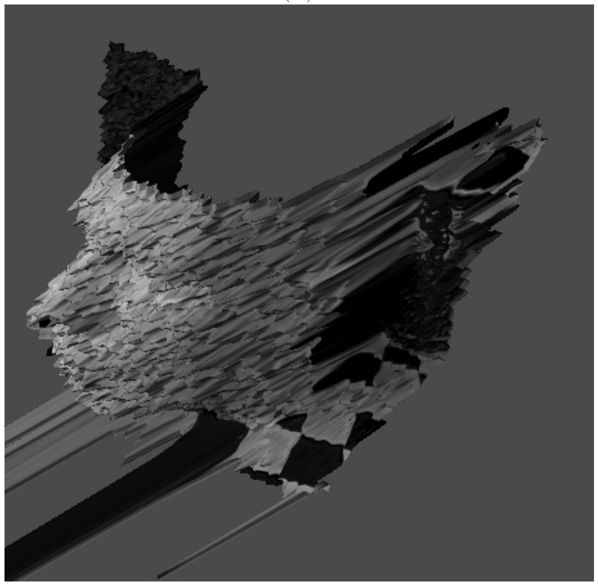
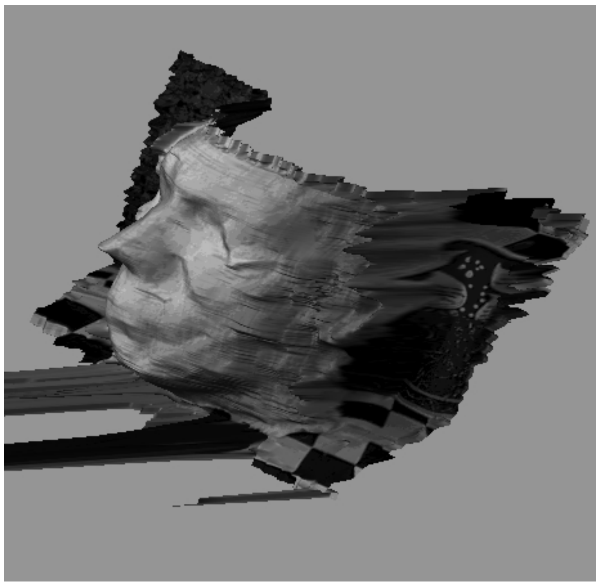
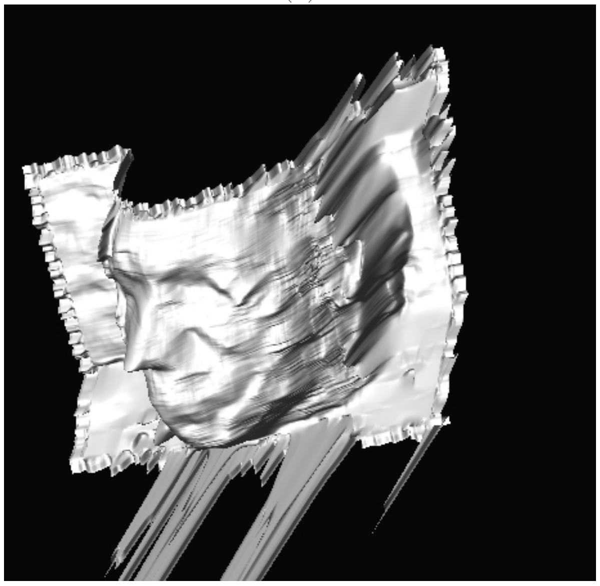

(c)

Figure 3: *One of the three input images (a) and the estimated $K(x,y) = \frac{1}{Z}$ inverse depth map (b) and 3D rendering of the surface with K scaled by 100.0 (c). Uses 7×7 region and a local constant depth model.*



(a)



(b)

Figure 4: *3D rendering of the estimated surface* $K(x, y) = \frac{1}{Z}$ *(inverse depth) scaled by 100.0. Uses* 30×30 *region and a locally planar depth model. Note the overshoot along the depth discontinuities around the head.*

more complex motion models described in section (2), the 24 parameter model or even the full 27 parameter 'Tensor Brightness Constraint' equation.

Alternatively one can look again at the 'Constant Brightness Equation' (3). In our implementation we calculate I_x and I_y at image 0. I_x and I_y and therefore the vector S are the same for both image pairs. If instead, the derivatives were computed at image 1 and image 2 the equations (12) and (13) would give us independent equations even in the collinear motion case.

Acknowledgments

Left blank for blind review.

References

[1] **Avidan, S. and Shashua, A.**, "Novel View Synthesis in Tensor Space", to be published SIGGRAPH96.

[2] **Bergen, J.R., Anandan, P., Hanna, K.J. and Hingorani, R.** "Hierarchical model-based motion estimation", In *Proceedings EECV*, Santa Margherita Ligure, Italy, June (1992)

[3] **Burt, P.J. and Adelson, E.H.** "The Laplacian pyramid as a compact image code" *IEEE Transactions on Communications*, 31:532-540, (1983)

[4] **Hartley, R.** "A linear method for reconstruction from lines and points" In *Proceedings ICCV*, 882-887, Cambridge, MA, June (1995)

[5] **Heel, J** "Direct Estimation of Structure and Motion from Multiple Frames" AI memo 1190, March (1990)

[6] **Horn, B.K.P. and Schunk, B.G.** "Determining optical flow" *Artificial Intelligence*, 17:185-203 (1981)

[7] **Horn, B.K.P. and Weldon, E.J.** "Direct methods for recovering motion" *IJCV* 2:51-76, (1988)

[8] **McQuirk, I.S.** "An Analog VLSI Chip for Estimating the Focus of Expansion" AITR-1577,(1996)

[9] **Longuett-Higgins, H.C. and Prazdny, K.** "The interpretation of a moving retinal image." *Proceedings of the Royal Society of London B*, 208:385-397,(1980)

[10] **Lucas, B.D. and Kanade, T.** "An iterative image registration technique with an application to stereo vision" In *Proceedings IJCAI*, 674-679, Vancouver, (1981)

[11] **Negahdaripour, S. and Horn, B.K.P** "Direct passive navigation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):168-176,(1987)

[12] **Shashua, A.** "Algebraic functions for recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):779–789, 1995.

[13] **Shashua, A. and Anandan, P.** "Trilinear Constraints Revisited: Generalized Trilinear Constraints and the Tensor Brightness Constraint", *Proceedings of the ARPA Image Understanding Workshop*, February 1996, Palm Springs, CA.

[14] **Shashua, A. and Hanna, K.J.** "The Tensor Brightness Constraints: Direct Estimation of Motion Revisited" Technion Technical Report, Haifa, Israel, November (1995)

[15] **Shashua, A. and Werman, M.**, "Trilinearity of Three Perspective Views and its Associated Tensor", In *Proceedings of ICCV 95*, 920-925 Boston, MA, USA, June (1995)

[16] **Spetsakis, M.E. and Aloimonos, J.** "A unified theory of structure from motion", In *Proceedings Image Understanding Workshop*, 1990.

[17] **Wang, J. Y. A and Adelson, E. H.** " Representing Moving Images with Layers. " *IEEE Transactions on Image Processing Special Issue: Image Sequence Compression*, 3(5):625-638, September (1994).