MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1190 March, 1990

Direct Estimation of Structure and Motion from Multiple Frames

Joachim Heel

Abstract: This paper presents a method for the estimation of scene structure and camera motion from a sequence of images. This approach achieves fundamental improvements over existing methods in the following respects:

- No computation of optical flow or feature correspondences is required. Structure and motion are obtained directly from gradients of image brightness.
- The method processes image sequences of arbitrary length and exploits the inherent redundancy for a significant reduction in error over time.
- No restrictions or assumptions are made about the camera motion or the surface structure. Both quantities are fully recovered using this method.

Our method combines the "direct" motion vision approach of Horn, Weldon and Negahdaripour with the theory of recursive estimation. Most importantly: it really works and we show results on a variety of real image sequences.

Acknowledgements: This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology and the Waikiki Surfing and Computer Vision Society of the University of Hawaii. Support for the AI Laboratory's Artificial Intelligence Research is provided in part by the Advanced Research Projects Agency under contract number DACA 76-85-K-0685 and under Office of Naval Research contract N00014-85-K-0124.

1 Introduction

In this paper we address the problem of estimating the 3D structure of a scene and the relative motion of a camera with respect to that scene from a sequence of images acquired by the camera. This problem, also referred to as the structure and motion estimation problem has been studied extensively in computer vision with rather limited success in the general case. We show how the use of multiple frames allows us not only to recover both structure and motion but also to do so in a way that is robust and practical for real images.

Knowledge of scene structure and camera motion is useful for the navigation of autonomous vehicles on land, undersea and in space. Information about the 3D structure of the scene is also of interest for the localization and recognition of objects.

Almost all exisiting approaches to solving the structure and motion estimation problem have at least one of the following two properties:

1. They use the optical flow field or feature matches as input.
2. They use only two sequential frames out of an image sequence.

Both of these properties represent severe limitations, as we discuss below. The purpose of this paper is to show how both limitations can be overcome to produce practical and accurate estimates of structure and motion.

1.1 Feature and optical flow based structure and motion estimation

Optical flow is a term that describes computational approximations to the projection of a 3D velocity field into the image plane (see section 1.4 for details). A number of methods are available for computing this vector field of image velocities: Anandan [3], Horn and Schunck [26], Hildreth [24], Nagel and Enkelmann [37], Heeger [20] and many others. The computation of optical flow is computationally very expensive and the optical flow usually exhibits significant systematic and random error.

Features are locations in the image that are easily identified such as edges, corners or other distinguished points in the brightness array. Feature-based methods first determine such locations in two frames and then match features between frames. Conceptually, the optical flow field is a set of dense feature matches, one match for each pixel in the image. Conversely, one can view feature matching methods as optical flow computation at a few selected locations where the probablity of obtaining a good optical flow estimate is large. Therefore, the problems we encounter in feature matching are essentially the same as in optical flow except that we focus on those locations where these problems are least noticeable by sacrificing the availability of dense information.

Despite the difficulties in computing optical flow or feature matches both are very intuitive intermediate representations with an exactly quantifiable relationship to the parameters of rigid body motion and scene structure (assuming that the optical flow prefectly matches the motion field). Unfortunately, however, this relationship is nonlinear and no closed-form solution for the general case exists. Among the methods which use optical flow or feature matches are Bruss and Horn [10], Adiv [1], Tsai and Huang [49], Longuet-Higgins

and Prazdny [31], Ullman [50], Heel [22], Mathies, Szelsiki and Kanade [34], Subbarao [47], Spetsakis and Aloimonos [45], Broida and Chellappa [8].

1.2 Two-frame structure and motion estimation

Features matches and optical flow are generally computed from two sequential frames in an image sequence. As a consequence, the information that can be extracted from a single optical flow field is limited to a snapshot of short duration. Since both image brightness measurement and optical flow computation introduce error, such a single snapshot cannot provide very accurate information. In addition, entire sequences of images are usually available, possibly in a continuous fashion, in the motion imaging situation. Repeatedly applying a two-frame algorithm to sequential pairs of images does not exploit the noise-reduction potential available in the form of redundant information.

Examples of two-frame motion estimation are Bruss and Horn [10], Adiv [1], Tsai and Huang [49], Longuet-Higgins and Prazdny [31], Weng, Huang and Ahuja [54], Fennema and Thompson [13], Waxman and Wohn [53], Roach and Aggarwal [43], Mitiche [36], Negahdaripour, Weldon and Horn [40], [28].

1.3 Direct and multiframe methods

Recently a number of approaches have been proposed to overcome the aforementioned difficulties. Some researchers have developed techniques that extract motion and structure information without computation of the optical flow. Best known are the "direct" methods of Horn, Weldon and Negahdaripour [40], [41], [28]. The basis of this work is the "brightness change constraint equation" which links image brightness values to motion and structure parameters. Despite this simplification, the resulting equations are still nonlinear and no closed-form solution exists in the general cases. The authors show how motion or structure can be recovered in various special cases such as purely translational motion or planar surfaces.

Aloimonos, Herve and Ito [2], [29] suggested a method for planar surfaces and developed the concept of a "linear feature" for direct motion estimation. This work also makes use of the brightness change constraint equation mentioned above. Kanatani [30] uses "feature functionals" to determine 8 flow parameters from which the motion and normal of a planar surface may be obtained.

The idea of using multiple frames in order to reduce the influence of noise has become increasingly popular recently. Ullman [50] presented a method termed the "incremental rigidity scheme" which operated iteratively on a sequence of images to recover structure. It required feature extraction and matching.

Broida and Chellappa [9] suggested the use of a Kalman filter for iterative improvement of estimates for structure and motion parameters over a sequence of images. They also used a sequence of matched features as input for their algorithm. In addition, a model for the object in the scene was necessary which contained information about the relative location of feature points.

Another recursive depth estimation procedure for more than two frames was suggested

by Bharwani, Riseman and Hanson [5]. Using feature matches and known translational motion depth is then recovered.

Stuller and Krishnamurthy [46] also used a Kalman filtering approach in which the projected translational motion of an object is formulated in terms of a dynamical systems. Then a recursive estimation scheme is used to recover the 2D translational motion.

Dickmans [11], [12] has investigated a Kalman filtering based motion analysis algorithm which he termed "4D dynamic scene interpretation". The algorithm and a matching hardware architecture have been used successfully for the guidance of vehicles in real traffic environments.

Backed by encouraging experimental results, Matthies, Szeliski and Kanade [35], [34] presented a practical and simple method for estimating structure using a Kalman filter based algorithm over a sequence of images. They succeeded in recovering dense depth maps of unprecedented quality from real images under the assumption of known purely translational motion parallel to the image plane.

Heel [21], [22], [23] independently developed a Kalman filter based method for the dense estimation of structure similar to the one by Matthies, Szeliski and Kanade. This "dynamic motion vision" had the additional capability of estimating the parameters of rigid body motion rather than requiring them to be known. It also placed no restriction on the nature of the motion.

A prominently different approach was the "epipolar image" method by Bolles and Baker [7] (also Yamamoto [56]). This method is particularly interesting because it does not require optical flow computation or conventional feature matching and makes use of multiple frames. For translational motion parallel to the image plane, a horizontal slice through a sequence of images provides important information not only about rigid body motion and structure but also about occlusion and segmentation. More recently [4] Baker and Bolles have generalized this method to handle more complex motions and to work incrementally as new images become available. In this case, the method becomes similar to previously mentioned Kalman filtering methods.

1.4 Direct dynamic motion vision

In this paper we present an algorithm for the dense estimation of structure and motion that

- does not require the computation of optical flow or feature matches
- uses an image sequence of arbitrary length to reduce the effect of noise.

In doing so, this method overcomes the obstacles that previous approaches encountered for estimating structure and motion efficiently and robustly.

We build on the "direct" work of Horn, Weldon and Negahdaripour to avoid computation of the optical flow. We combine it with the Kalman filter recursive estimation technique to continuously update and improve structure information over time, as new measurements from images become available. A least-squares motion estimation procedure is interleaved with the stages of the filter, so that both motion and structure are obtained simultaneously, which was not previously possible in the case of general surfaces and motions.

The paper is organized as follows: We briefly review the direct approach to motion vision and the theory of recursive estimation as they apply to our application. We then formulate the dynamic motion and structure estimation algorithm by combining direct depth measurement with Kalman filtering. After the outline of the approach, each stage of the algorithm, depth measurement, update, prediction, motion estimation and smoothing/filling in are discussed in detail, analyzed with respect to their algorithmic complexity and evaluated experimentally. Finally, the results which may be obtained using this method are evaluated in an extensive experimental section using a variety of real image sequences.

2 The Direct Motion Vision Approach

2.1 The motion imaging situation

The traditional approach to motion vision as outlined by Longuet-Higgins and Prazdny [31] was to compute the *optical flow* as an intermediate representation of motion information and then compute rigid body motion and structure parameters using the optical flow. Optical flow denotes a computational approximation to the projection of the 3D velocity field of a scene into the image plane.

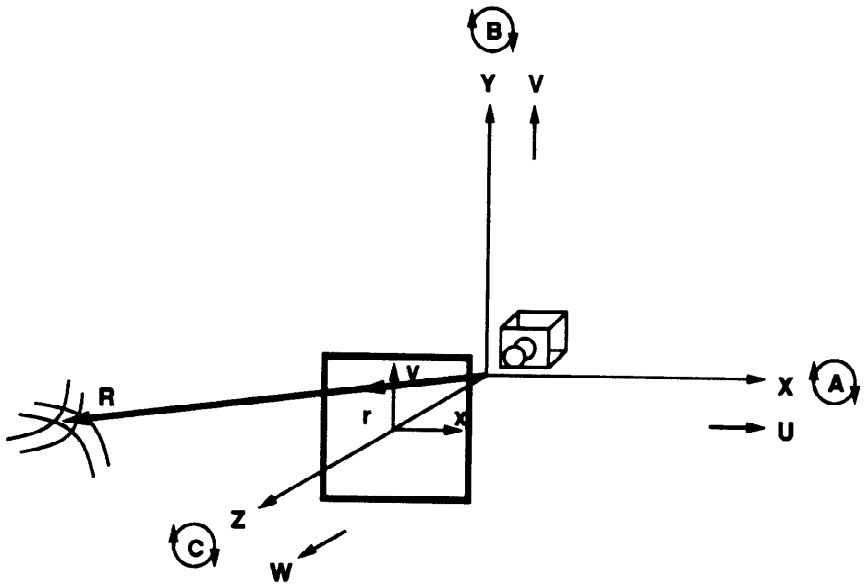

Figure 1: Coordinate system and projection geometry in the motion imaging situation.

In what follows we assume that only one independently moving rigid body is in the image area of interest. This is the case when a camera moves through a static environment for example. If multiple moving objects are present a segmentation into regions of single independently moving objects must be performed prior to applying this formalism.

Suppose we denote the relative motion of the camera with respect to the scene by a translation $\mathbf{t} = [U, V, W]^T$ and a rotation $\boldsymbol{\omega} = [A, B, C]^T$. Choose a coordinate system as shown in figure 1 with the origin at the camera's center of projection (principal distance f) and the optical axis aligned with the z axis $\hat{\mathbf{z}} = [0, 0, 1]^T$. A point P at location $\mathbf{R} = [X, Y, Z]^T$ in the scene is imaged at pixel location $\mathbf{r} = [x, y, f]^T$. The value Z is referred to as the *depth* of point P.

Given this situation, the point P will move according to

$$\dot{\mathbf{R}} = -\mathbf{t} - \boldsymbol{\omega} \times \mathbf{R} \tag{1}$$

where the dot denotes temporal differentiation. We can think of the vector $\dot{\mathbf{R}}$ as a 3D motion vector that is attached to a point on the surface of the scene in view. The collection of these vectors for all surface points constitutes a 3D velocity vector field. Suppose this velocity field is now projected into the image plane via the equation

$$\mathbf{r} = \frac{f}{Z}\mathbf{R} \tag{2}$$

of perspective projection. We differentiate equation (2) and then substitute (1) to obtain

$$\dot{\mathbf{r}} = -\hat{\mathbf{z}} \times (\mathbf{r} \times (\frac{1}{f}\mathbf{r} \times \boldsymbol{\omega} - \frac{1}{Z}\mathbf{t}) \tag{3}$$

The vector $\dot{\mathbf{r}}$ describes the motion of the projection of P into the image plane. The collection of $\mathbf{r}$ vectors at every point in the image plane is a vector field of image velocities commonly referred to as the *motion field*. This is the motion information that an observer could perceive given a perfect motion "measurement" device.

2.2 Motion and structure from optical flow

We investigate the motion field equation (3) in more detail. This vector equation for $\dot{\mathbf{r}}$ consists of three components, the third of which is identical to zero, since the third component of $\mathbf{r}$ is the constant focal length f. The remaining two components of the image velocity $(\dot{x}, \dot{y}) = (u, v)$ can be written as

$$u = \frac{-fU + xW}{Z} + \frac{1}{f}(Axy - B(x^2 + f^2) + Cyf) \tag{4}$$

$$v = \frac{-fV + yW}{Z} + \frac{1}{f}(A(y^2 + f^2) - Bxy - Cxf). \tag{5}$$

These motion field equations can be established at every pixel (x, y) in the image plane. The classical assumption is that estimates of the image velocities (u, v) can somehow be obtained from an input image sequence. The task is then to compute the motion $\mathbf{t} = [U, V, W]^T$, $\boldsymbol{\omega} = [A, B, C]^T$ as well as the depth Z given the velocities (u, v).

Note that the motion $\mathbf{t}$ and $\boldsymbol{\omega}$ is a global quantity, i.e. there is one rigid body motion for all points in the image plane. The depth Z however, may vary spatially. In an image region of $n \times n$ pixels we therefore have $2(n \times n)$ motion field equations for $n \times n + 6$ unknowns. We also observe that translation and depth are determined only up to a common scale factor i.e. if $\mathbf{t}$ and Z are a solution to the motion field equations then so are $k\mathbf{t}$ and kZ, for any $k \neq 0$. We can therefore at best hope to recover the direction of translation and the depth relative to that translation. Finally, the fact that the unknowns Z, $\mathbf{t}$ and $\boldsymbol{\omega}$ are related in a non-linear fashion in equations (4), (5) makes the motion and structure estimation from optical flow hard and, to date, unsolvable in closed-form.

2.3 Optical flow

Estimating the optical flow (u, v) from images is the other part of the traditional motion vision problem. We wish to determine a vector field (u, v) (i.e. a vector at every point

(x, y) in the image) that approximates the motion-field (4), (5) as closely as possible. To accomplish this we note that a particular optical flow vector points from the projection of a point in one image to the projection of the point in the subsequent image. Conceptually, an optical flow vector therefore connects corresponding points in sequential images.

It is generally assumed that the brightness of corresponding points in two subsequent images is the same. Horn and Schunck [26] formulated this in

$$\frac{dE}{dt} = 0 \tag{6}$$

where E denotes the brightness of a point in the image and t is the time. This can be expanded into the *brightness change constraint equation*

$$\frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = E_x u + E_y v + E_t = 0 \tag{7}$$

where E_x, E_y, E_t are the brightness derivatives in spatial and temporal directions and (u, v) is the optical flow introduced above. The brightness gradient may be obtained by a finite differences approximation for the derivatives directly from the images.

The brightness change constraint equation links the brightness values E measured by the camera system to the optical flow values (u, v) needed for motion and structure estimation. We see that with one brightness change constraint equation at every image point (x, y) and two unknowns (u, v) we are dealing with an underconstrained problem. We must therefore introduce an additional constraint such as the smoothness of the optical flow field (as done by Horn and Schunck [26] and Hildreth [25]) in order to solve equation (7) for the desired (u, v) everywhere.

Nevertheless, optical flow computation is both expensive in terms of computation time and subject to both systematic error, such as the aperture problem, and random errors due to image noise. The necessity to compute the optical flow has been a major obstacle on the path to real-time motion vision.

We might add that criticism of the validity of the brightness change constraint assumption (7) has been voiced (see Verri and Poggio [51], Schott [44]). Indeed, equation (7) is only a simplified approximation of reality which is subject to errors in most real situations. The main advantage lies in its simplicity. We will later evaluate the amount of error in the brightness change constraint equation for our specific application.

2.4 Skipping optical flow: The direct approach

As we have outlined, the estimation of scene structure and relative camera motion was traditionally a two-stage process consisting of the computation of optical flow from brightness values, followed by the recovery of motion and structure via the motion field equations. In this approach, the optical flow is no more than an intermediate representation, the computation of which involves a number of difficulties.

Horn, Negahdaripour and Weldon [38], [40], [27], [28], [39], [42] were among the first to point out that the computation of optical flow was not a necessary step in the process of determining structure and motion from image sequences. The main advantage of this

approach are tremendous improvements in computational efficiency since the computation of optical flow is expensive as we have pointed out.

Conceptually, the idea is as follows: The brightness change constraint equation (7) links brightness values to optical flow. The motion field equations (4), (5) link optical flow to rigid body motion and structure. Instead of computing optical flow from (7) and then using it in (4), (5) to obtain motion and structure we plug the motion field equations (4), (5) into the brightness change constraint equation (7) and obtain one equation which links image brightness gradients to motion and structure parameters.

We can rewrite and simplify the resulting equation. For a detailed derivation refer to Horn and Weldon [28]. We obtain

$$\frac{\mathbf{s} \cdot \mathbf{t}}{Z} + \mathbf{v} \cdot \boldsymbol{\omega} + E_t = 0 \tag{8}$$

in which

$$\mathbf{s} = \begin{bmatrix} -fE_x \\ -fE_y \\ xE_x + yE_y \end{bmatrix} \quad \text{and} \quad \mathbf{v} = \begin{bmatrix} E_y f + y(xE_x + yE_y)/f \\ -E_x f - x(xE_x + yE_y)/f \\ yE_x - xE_y \end{bmatrix}. \tag{9}$$

Note in (9) that $\mathbf{s}$ and $\mathbf{v}$ contain only measureable or known quantities such as the derivatives of brightness E_x, E_y, E_t and the location (x, y) at which we are evaluating the constraint. The unknowns in equation (8) are therefore the motion $\mathbf{t}$ and $\boldsymbol{\omega}$ and the depth Z - precisely the quantities we are interested in.

In this formulation, the task is to compute $n \times n$ values of Z and 6 values for $\mathbf{t}$ and $\boldsymbol{\omega}$ (actually only 5 due to the scale-factor ambiguity) from $n \times n$ nonlinear equations. In their work Horn, Negahdaripour and Weldon showed how motion and structure parameters may be estimated using equation (8) for a number of special cases (pure translation, planar surfaces etc.). A major difficulty they encountered was the fact that motion and depth are linked nonlinearly in (8) and cannot both be recovered in general. Note, however, that if one of either motion or depth is known, the other quantity is easily obtained from linear equations. This observation is the key to the iterative estimation of both structure and motion from a sequence of frames.

3 Recursive Estimation and Dynamic Motion Vision

The structure and motion estimation problem as we have previously outlined it is inherently instantaneous. Two subsequent frames are commonly used to compute the optical flow so that any structure or motion result obtained from this optical flow is based on information from those two time instances only. In the direct approach we use two subsequent frames to compute the brightness gradients E_x, E_y and E_t. Structure and motion recovered from a direct method corresponds to the information from those two frames only.

As we will see, the motion and in particular the structure information obtainable from an instantaneous snapshot is rather noisy in general. Conceptually information from multiple frames may be useful to reduce this noise and produce a useful estimate over time. There are two fundamentally different ways in which multiple frame measurements can be integrated.

1. Global or batch methods. In this case, the data from n frames is collected and then input to the integration procedure. We expect this to produce the most accurate result for the given set of frames. However, it necessitates storage of large amounts of data and does not produce any result until all frames have been acquired.

2. Recursive or iterative methods. Here we maintain a current estimate of the quantities of interest which are updated each time a new frame becomes available. The accuracy of such a method should be inferior to a global technique and convergence/stability becomes an issue. However, no storage is required and an estimate is available at every time instant.

Due to the continuous nature of visual information acquisition and the large amounts of data involved we focus here on a recursive estimation method for temporal integration.

3.1 Recursive Estimation Theory

There are many in-depth treatments of recursive estimation theory such as Gelb [17]. Knowledge of dynamical systems theory is also useful and provided by standard textbooks such as Luenberger [32], Franklin and Powell [15], Föllinger [14], Willsky [55] and many others. We present here an extremely simplified summary of essential concepts in recursive estimation theory which we will use for time-continuous motion vision. We will use notation to match the motion vision problem to make the applicability of this theory to our problem domain apparent.

Suppose our task is to estimate a (scalar) quantity $Z(k)$ from a sequence of measurements Z_k taken at discrete points k in time. Suppose further that the Z_k are generated by a stochastic process

$$Z_k = Z(k) + n_k \tag{10}$$

where n_k is zero mean Gaussian noise of known variance p_k. Finally, we know that the quantity Z changes over time according to the difference equation

$$Z_{k+1} = f(Z_k). \tag{11}$$

The goal is to compute at every time instance k an estimate $\hat{Z}_k$ which is as close as possible to the true value $Z(k)$. Kalman formulated and solved this problem in a far more general case (see the literature cited above) and the resulting algorithm is referred to as a *Kalman filter.*

The Kalman filter maintains an estimate $\hat{Z}_k$ and a variance $\hat{p}_k$. When a new measurement Z_k with variance p_k becomes available we perform the following *update* operation (the arrow denotes an assignment):

$$\hat{Z}_k \leftarrow \frac{Z_k/p_k + \hat{Z}_k/\hat{p}_k}{1/p_k + 1/\hat{p}_k} \tag{12}$$

$$\hat{p}_k \leftarrow \frac{1}{1/p_k + 1/\hat{p}_k} \tag{13}$$

In words: the new estimate of Z is a weighted sum of the old estimate and the new measurement with the variances as weights. If, for example, a very noisy measurement is received, its variance p will be large compared to the variance of the old estimate $\hat{p}$. As a consequence the measurement will have very little influence on the new value of the estimate. Conversely, a high-quality measurement will dominate the weighted sum due to its small variance and cause the new estimate to be very similar to its value.

The update formulas (12) and (13) may also be obtained from a maximum likelihood estimation of Z and its variance given the previous estimate and the measurement under the assumption that the estimate $\hat{Z}_k$ and measurement Z_k are uncorrelated.

Notice the following feature of the variance update process (13): Suppose without loss of generality that $\hat{p}_k < p_k$. Now (since variances are positive)

$$\frac{1}{1/p_k + 1/\hat{p}_k} < \frac{1}{1/\hat{p}_k} = \hat{p}_k < p_k \tag{14}$$

so the updated variance is smaller than both the current variance and the measurement variance. In other words, variance decreases strictly during the update procedure. This captures the notion that the quality of the estimate improves over time and indicates that later measurements will have less and less influence on the resulting estimate.

An additional difficulty is introduced by the fact that $Z(k)$ may vary with time. It is therefore not possible to simply combine the result $\hat{Z}_k$ obtained through an update at time k with a measurement obtained at time $k+1$. We must first *predict* an estimate for time $k+1$ using the known temporal behavior:

$$\hat{Z}_{k+1} \leftarrow f(\hat{Z}_k) \tag{15}$$

$$\hat{p}_{k+1} \leftarrow \left(\frac{\partial f}{\partial Z}\right)^2 \hat{p}_k. \tag{16}$$

The original Kalman filter and the proofs for optimality require the function f to be linear. We will see that f is indeed linear in our application.

We can visualize the operation of the Kalman filter with the help of a block diagram as shown in figure 2. Note the simplicity and locality of both update and prediction stages which make this estimation particularly appealing from an algorithmic point of view.

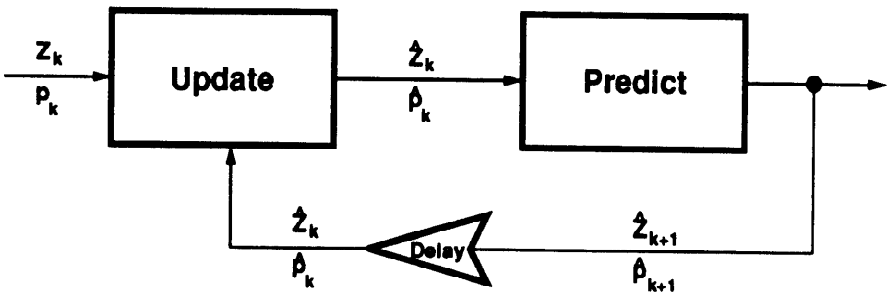

Figure 2: Block diagram of a Kalman filter.

3.2 Dynamic Motion Vision

In this section we will outline how the theory of recursive estimation can be combined with the direct approach to motion vision. The result is a time-continuous estimation procedure which not only extends the one-frame case to multiple frames but allows us to recover both motion and structure by exploiting the iterative nature of the Kalman filter, something that was not possible previously. We refer to this technique as "dynamic motion vision".

Suppose for now the rigid body motion vectors $\mathbf{t}$ and ω are known. In this case, the brightness change constraint equation (8) contains only one unknown and can be solved for the depth Z. By evaluating it, we can therefore obtain a value of Z at every point (x, y) in the image and at every time instance k. Due to noise and errors in the brightness change constraint assumptions, this value will differ from the true value of Z of the point which is imaged at that pixel. Further, the value of Z at a particular pixel changes over time due to the relative motion between object and observer. Using the equations of rigid body motion, this change in depth can be derived precisely.

The recursive estimation task can therefore be formulated as follows. We wish to estimate the depth Z at a particular pixel (x, y) in the image plane. For each new image we can compute a noisy measurement Z_k using the brightness change constraint equation. This measurement can be used to update an estimate $\hat{Z}_k$ according to the Kalman filter update rule. As the next frame $k+1$ becomes available, the depth value changes due to the rigid body motion: $Z_{k+1} = f(Z_k)$. This known transformation can then be used to predict the estimate for the depth value $\hat{Z}_{k+1}$ in the next iteration. A block diagram depicting this operation is shown in figure 3.

Note that the previous formulation describes the recursive estimation of Z at one particular pixel location (x, y). In general we will be interested in obtaining depth information for every pixel in the image. To accomplish this we can imagine a Kalman filter positioned at every pixel location where it continuously estimates the depth. During the prediction stage, some interaction will take place between these spatially distributed Kalman filters when the projections of real world points of which depth is being estimated move to new image locations due to rigid body motion.

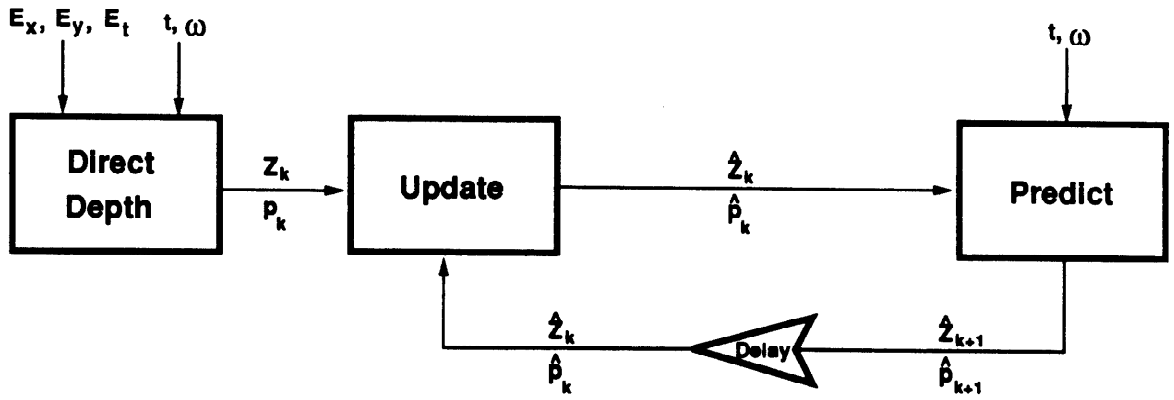

Figure 3: Block diagram of Kalman filter in motion vision.

So far, we have assumed that the motion parameters t and ω are known. This may be the case in a number of practical situations, in particular when the camera is mounted on vehicle or robot. In this case, the method outlined above is a complete solution to the recursive structure estimation problem. The even more interesting problem, however, arises, when the motion is also unknown and must be estimated. The key idea is the following. When depth Z is known at every location (x, y), we can solve the brightness change constraint equation (8) for the motion vectors t and ω. However, depth is unknown - it is precisely what the filter is estimating. Now, in each iteration of the filter we have a current estimate of the depth $\hat{Z}$. The idea is then to use the current filter estimate $\hat{Z}$ for the computation of motion from the brightness gradient using equation (8). This process can be initialized by starting with an arbitrary constant value (since the motion estimation adjusts for the unknown scale factor) for Z everywhere.

This procedure of alternatively estimating depth and motion relies on the fact that motion estimation is rather insensitive to errors in depth as we show in our experiments. Conceptually, we can think of the motion vectors as parameters for our filter. Inserting the motion estimation between update and prediction stage of the filter therefore corresponds to an online adaptation of the filter paramters. Such an adaptive filter is of course not in line with the classical Kalman filter theory and theoretical properties derived for the latter do not extend in a straightforward way.

Figure 4 shows the integration of motion estimation into the dynamic motion vision process. This block diagram represents the structure which we have implemented and the components of which we will describe in detail in the following sections.

For the sake of clarity we have omitted another block which should be positioned between update and prediction part of the filter. Our implementation allows us to perform a variety of spatial noise reduction operations. This was motivated by two facts:

1. The filter estimation process does not take spatial smoothness into account: each filter

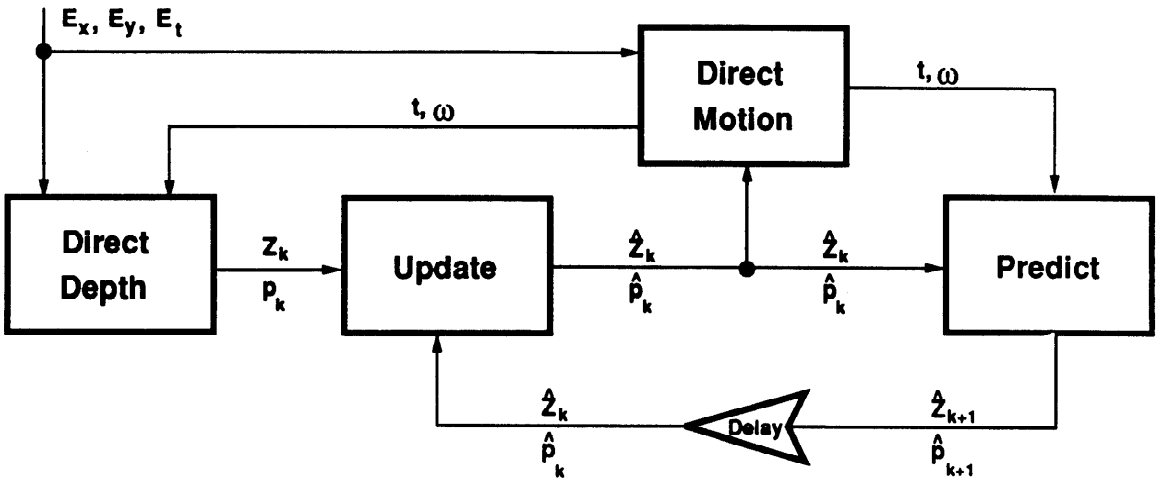

Figure 4: Block diagram of dynamic motion vision with motion estimation.

at each pixel operates independently. A spatial smoothing step may be desirable to enforce a certain amount of smoothness.

2. Most any image contains regions of uniform brightness in which no depth information is obtainable from motion. Mathematically this is seen by the fact that all components of the brightness gradient are zero, or very small, in such a region thereby providing little or no constraint on the value of the depth. Even repeated measurements will provide poor data with large variances in such regions. In such cases it may be desirable to fill in data from adjacent regions of lower variance.

Filling in and smoothing are simply cosmetic operations which do not contribute to the underlying idea of temporal noise reduction via recursive estimation. They are therefore purely optional in our implementation.

The following sections contain detailed descriptions of the individual modules used in the dynamic motion vision process.

4 Direct Depth Estimation

The first stage of the dynamic motion and structure estimation algorithm is the computation of depth from the input image data. As shown in the structural block diagram in figure 4, the input to the depth estimation stage are the brightness derivatives E_x, E_y and E_t and the motion $\mathbf{t}$ and ω. The output is the depth Z and its variance at every pixel (x, y).

In this section, we describe how we compute the gradient of image brightness. We then show how depth may be obtained in a straightforward way from brightness gradients and motion. Using data from real images we will show that this straightforward solution is quite unsatisfactory. We then suggest a way to improve the solution method to produce better results and illustrate the achievable improvement with more image data.

4.1 Brightness gradient computation

We are given discrete samples $E_{i,j,k}$ of the brightness function $E(x, y, t)$. The objective is to estimate the first derivatives of E with respect to all of its independent variables. The simplest approximation is obtained using central differences

$$E_x(i,j,k) = (E_{i+1,j,k} - E_{i-1,j,k})/2\Delta x \tag{17}$$

$$E_y(i,j,k) = (E_{i,j+1,k} - E_{i,j-1,k})/2\Delta y \tag{18}$$

$$E_t(i,j,k) = (E_{i,j,k+1} - E_{i,j,k-1})/2\Delta t, \tag{19}$$

where Δx and Δy denote the physical spacing of pixels on the CCD chip in horizontal and vertical directions and Δt denotes the time elapsed between the acquistion of two subsequent images.

Horn and Schunck [26] suggested an alternative in which a derivative value is not computed from a difference across a single pixel but as the average of four differences at neighboring pixels:

$$\begin{aligned} E_x(i,j,k) = {} & (E_{i+1,j,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i,j+1,k} + \\ & E_{i+1,j,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i,j+1,k+1})/4\Delta x \end{aligned} \tag{20}$$

$$\begin{aligned} E_y(i,j,k) = {} & (E_{i,j+1,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i+1,j,k} + \\ & E_{i,j+1,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i+1,j,k+1})/4\Delta y \end{aligned} \tag{21}$$

$$\begin{aligned} E_t(i,j,k) = {} & (E_{i,j,k+1} - E_{i,j,k} + E_{i+1,j,k+1} - E_{i+1,j,k} + \\ & E_{i,j+1,k+1} - E_{i,j+1,k} + E_{i+1,j,k+1} - E_{i+1,j+1,k})/2\Delta t \end{aligned} \tag{22}$$

Notice that the derivative approximations are computed from pixel values at the corners of corresponding grid squares in subsequent images as shown in figure 5.

The pixels form a spatio-temporal cube. The derivative values computed from equations (20), (21), (22) are therefore the estimates of derivative at the center of this cube - inbetween frames and inbetween pixels. Since all computations in the direct dynamic motion vision algorithm are carried out on gradients (the images can be discarded once the gradients have been computed), the relative position of pixels and gradient values poses no difficulty during the computation. Note however, that the gradient field will have one row and one column less than the images used for its computation.

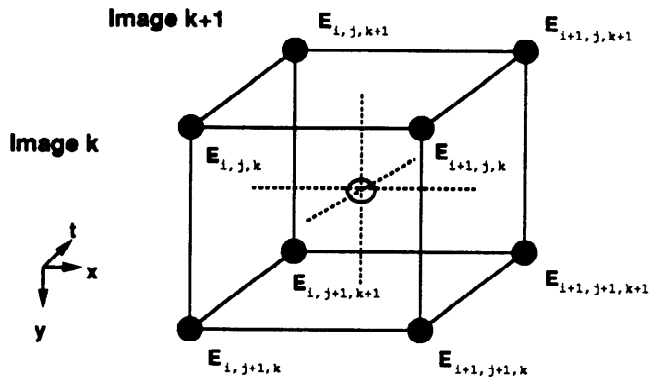

Figure 5: Location of pixel values for gradient computation

4.2 Single pixel depth computation

We have shown how the brightness gradient E_x, E_y, E_t may be computed and we are assuming that the motion vectors t and ω are known. To compute the depth we use the brightness change constraint equation (8)

$$\frac{\mathbf{s} \cdot \mathbf{t}}{Z} + \mathbf{v} \cdot \omega + E_t = 0 \tag{23}$$

which we can solve for the desired depth Z

$$Z = -\frac{\mathbf{s} \cdot \mathbf{t}}{E_t + \mathbf{v} \cdot \omega} \tag{24}$$

To obtain an estimate of the variance in the value of Z we assume that the brightness E at every pixel is corrupted by noise n of variance σ_E^2 which is identically distributed at every pixel and mutually uncorrelated between pixels. The noise in the eight brightness values used in the computation of the brightness derivatives propagates through the derivatives and appears in the depth value. Unfortunately, the relationship between depth and brightness values is nonlinear and rather complex. We will outline briefly how a formula for the variance $p = \sigma_Z^2$ may be obtained.

As mentioned, eight brightness values $E_i, i = 1, \ldots 8$ at the corners of a spatio-temporal cube contribute to the value of Z:

$$Z = f(E_1, \ldots, E_8) \tag{25}$$

Under the assumption that the nonlinear function f can be locally approximated by the first-order terms of its Taylor series, the variance in Z is given by

$$p = \sigma_E^2 \sum_{i=1}^{8} \left(\frac{\partial Z}{\partial E_i}\right)^2. \tag{26}$$

To compute the necessary partial derivatives we first determine the immediate relationship between Z and the brightness derivatives by plugging in the values for **s** and **v** from equation (9) into the depth equation (24). We find

$$Z = \frac{aE_x + bE_y}{E_t + cE_x + dE_y} \tag{27}$$

in which

$$\begin{aligned} a &= fU - xW & (28)\\ b &= fV - yW & (29)\\ c &= (xyA - (f^2 + x^2)B)/f + yC & (30)\\ d &= ((f^2 + y^2)A - xyB)/f - xC & (31) \end{aligned}$$

(Recall that f is the principal distance, $\mathbf{t} = [U, V, W]^T$ and $\omega = [A, B, C]^T$). Next, we plug in the formulas for the partial derivatives E_x (20), E_y (21), E_t (22) which would require more space than we have available here. However, after differentiating the result with respect to the eight brightness values according to (26), some simplification can be done and we obtain

$$p = \frac{\sigma_E^2}{2}\left((\frac{1}{\Delta x})^2(\frac{\partial Z}{\partial E_x})^2 + (\frac{1}{\Delta y})^2(\frac{\partial Z}{\partial E_y})^2 + (\frac{1}{\Delta t})^2(\frac{\partial Z}{\partial E_t})^2\right) \tag{32}$$

where

$$\begin{aligned} \frac{\partial Z}{\partial E_x} &= \frac{aE_t + (ad - bc)E_y}{(E_t + cE_x + dE_y)^2} & (33)\\ \frac{\partial Z}{\partial E_y} &= \frac{bE_t - (ad - bc)E_x}{(E_t + cE_x + dE_y)^2} & (34)\\ \frac{\partial Z}{\partial E_y} &= -\frac{aE_x + bE_y}{(E_t + cE_x + dE_y)^2}. & (35) \end{aligned}$$

In conclusion, the derivation is extremely complex and the resulting formula is expensive to compute. Moreover, the result is only valid approximately due to the Taylor series. It would be extremely useful to find a simpler approximation that serves the same purpose: quantifying the quality of the measurement Z.

We will now briefly investigate the performance of the depth estimation scheme outlined above. This is necessary, since we are relying on the brightness change constraint assumption. We investigated two scenes in which the camera translated before a planar surface (a poster) that was parallel to the image plane. Images of the scenes are shown in figures 6 a) and b). We will refer to them as "wave" and "onnanoko".

Planar scenes are particularly useful for the evaluation we intend to perform since we can use the mean, variance and histogram of all the computed depth values to present results compactly. In the case of the wave image a motion of $\mathbf{t} = [1.5, 0, 3]$ mm was used, the plane was at a distance of 700 mm initially. The depth was then computed using the

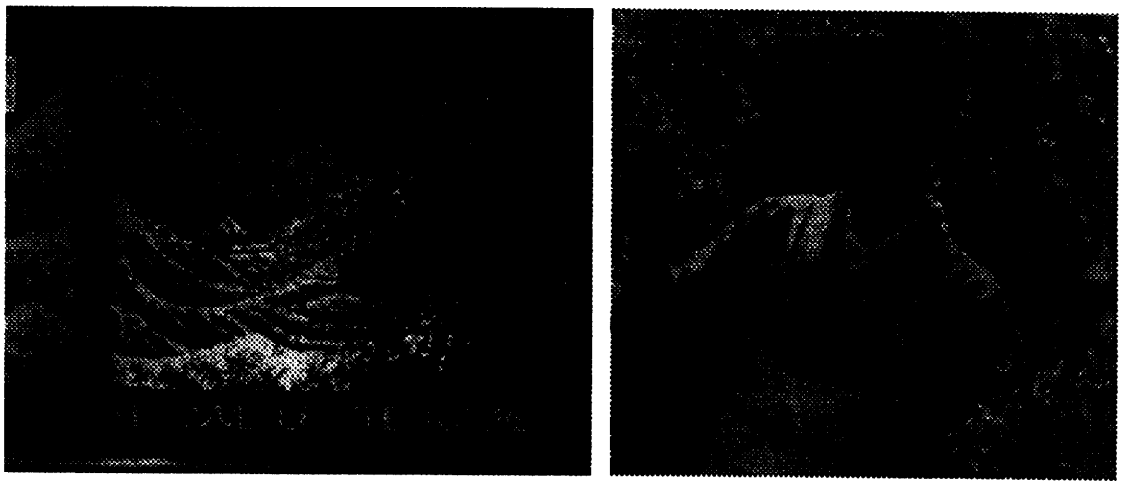

Figure 6: The planar scenes: a) "wave" b) "onnanoko".

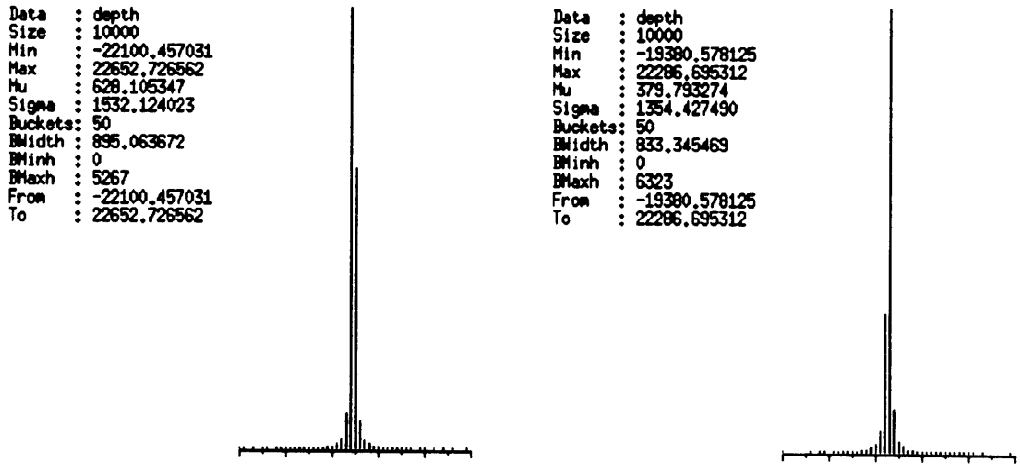

Figure 7: Statistical analysis of depth values computed for planar scenes: a) wave scene b) onnanoko scene.

formulas derived above and the known values for the motion. The resulting histogram is shown in figure 7 a).

We find that the distribution of depth values is extremely broad with a range of $[-22100, 22652]$. The standard deviation of 1532 is more than twice as large as the actual distance of the plane. Ideally, this distribution should have mean 700 and variance 0 (or at least small with respect to the mean). In addition we have many negative values which are physically impossible (depth must be positive). In short, the depth obtained by

this method is subject to extremely large errors and almost useless.

A similar observation can be made for the onnanoko scene. Here the motion was $\mathbf{t} = [1.5, 0, -3]$ mm with the planar surface at 500 mm initially. The results of the statistical analysis of the resulting depth map are shown in figure 7. The conclusion is the same: too much error to be useful.

4.3 Patch depth estimation

Apparently, the quality of results obtained by evaluating the brightness change constraint equation at individual pixels in real images is limited. The above results can be used directly to quantify the amount of error involved in the brightness change constraint assumption. This error will be significant in general.

From our experience with motion estimation via the brightness change constraint equation, however, (see section 6.2) we knew that it could be applied with great success in cases where data was taken from an entire image region and then used to estimate a global quantity in that region in a least-squares fashion. This prompted us to investigate depth estimation from a patch rather than a single pixel.

The idea is as follows. Pick a patch size n (typically a few pixels). Assume Z is constant within a given patch. This gives us $n \times n$ constraint equations

$$\frac{\mathbf{s} \cdot \mathbf{t}}{Z} + \mathbf{v} \cdot \boldsymbol{\omega} + E_t = 0 \tag{36}$$

that all contain the same Z. Find the value of Z that minimizes the sum of squared errors

$$\Phi(Z) = \sum_{x,y \epsilon P} (\frac{\mathbf{s} \cdot \mathbf{t}}{Z} + \mathbf{v} \cdot \boldsymbol{\omega} + E_t)^2. \tag{37}$$

in which P is the patch we have chosen. To solve the minimization problem we substitute $d = 1/Z$ to make the problem linear

$$\Phi'(d) = \sum_{x,y \epsilon P} (d(\mathbf{s} \cdot \mathbf{t}) + \mathbf{v} \cdot \boldsymbol{\omega} + E_t)^2. \tag{38}$$

We differentiate with respect to d

$$\frac{d\Phi'}{dd} = 2 \sum_{x,y \epsilon P} (d(\mathbf{s} \cdot \mathbf{t}) + \mathbf{v} \cdot \boldsymbol{\omega} + E_t)(\mathbf{s} \cdot \mathbf{t}) \tag{39}$$

to find the necessary condition for a minimum

$$d = -\frac{\sum_{x,y \epsilon P} (v \cdot \boldsymbol{\omega} + E_t)(\mathbf{s} \cdot \mathbf{t})}{\sum_{x,y \epsilon P} (\mathbf{s} \cdot \mathbf{t})^2}. \tag{40}$$

Through backsubstitution we finally obtain the expression for Z

$$Z = -\frac{\sum_{x,y \in P} (\mathbf{s} \cdot \mathbf{t})^2}{\sum_{x,y \in P} (v \cdot \omega + E_t)(\mathbf{s} \cdot \mathbf{t})} \tag{41}$$

We note that in the trivial case of a single-pixel patch we have

$$Z = -\frac{(\mathbf{s} \cdot \mathbf{t})^2}{(v \cdot \omega + E_t)(\mathbf{s} \cdot \mathbf{t})} = -\frac{\mathbf{s} \cdot \mathbf{t}}{v \cdot \omega + E_t} \tag{42}$$

which is the same as the formula (24) that we derived for the single pixel estimation when $\mathbf{s} \cdot \mathbf{t} \neq 0$.

The next task would be to derive an expression for the variance of Z in the same way demonstrated in the previous subsection. Only now the number of brightness values which contribute to Z, and which must therefore be used in the derivation of the variance, is $2(n+1)^2$ where n is the width and height of the patch. From our experience in the single pixel case we know that this will not only require an extremely lengthy derivation, it will also lead to a very complex expression for the variance. Instead, we choose to use a simple approximation to the variance values which exhibits the same properties that we require for the recursive estimation scheme.

Recall that the purpose of the variance is to indicate the quality of the measurement that it accompanies. In our case, the measurement (depth Z) is computed as a quotient. The magnitude of the denominator is of particular importance. When it is zero, the depth cannot be obtained at all. When it is small, the result is very sensitive to errors in the measurements that enter into the numerator. The significance of the denominator is reflected in the fact that it appears squared in the denominator of the expression for the variance in the single pixel case. In analogy to this we chose the following expression to approximate p:

$$p = \frac{n^4 S}{(\sum_{x,y \in P} (v \cdot \omega + E_t)(\mathbf{s} \cdot \mathbf{t}))^2} \tag{43}$$

where n is once again the width/height of the patch and S is a constant scale factor. The term n^4 is used to normalize the variance with respect to the patch size. The scale factor can be chosen arbitrarily. However, when the depth estimator is used as part of the recursive estimation scheme its choice must be coordinated with the initial value of the variance in the filter. Their relationship determines the convergence property of the filter.

4.4 Evaluating the patch depth estimation procedure

We now compare the patch depth estimation with the single pixel estimation using the same images and statistical analysis as above. Figures 8 a) and b) show the histograms of the depth values obtained from the wave sequences using patch sizes of 4 and 9. For better comparison with the single pixel case, the same histograms truncated to the range used for the single pixel histogram 7 are shown in figures 9 a) and b).

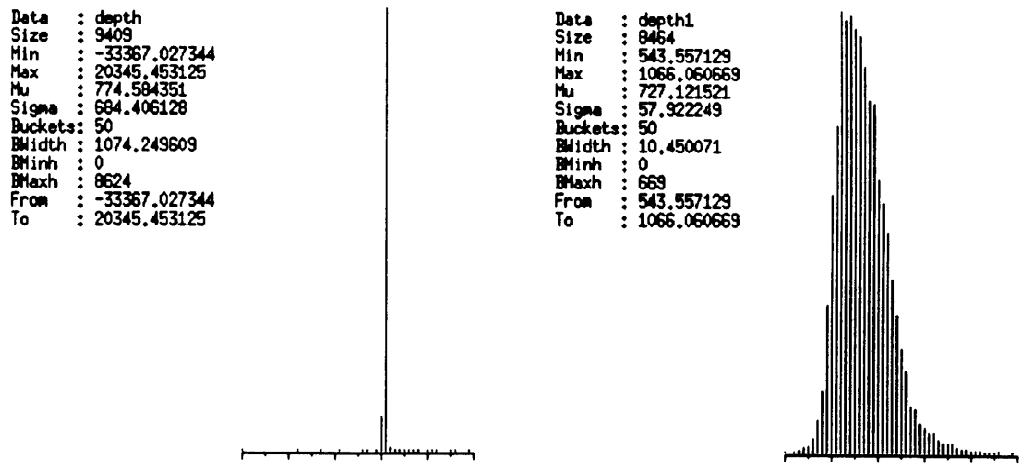

Figure 8: Statistical analysis of depth values computed for the wave scene using patch depth estimation with a) 4 pixel and b) 9 pixel patches.

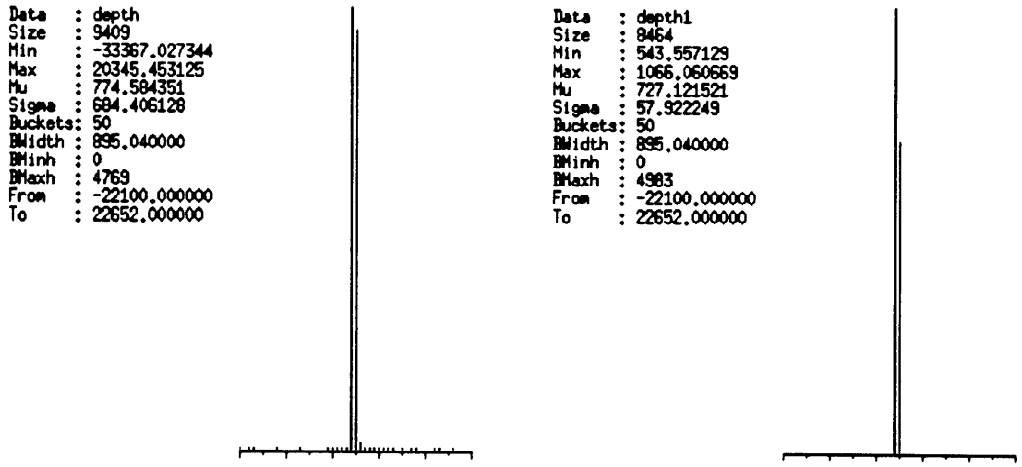

Figure 9: Statistical analysis of depth values computed for the wave scene using patch depth estimation with a) 4 pixel and b) 9 pixel patches. Histogram ranges truncated to that of the single pixel histogram above.

We expect that increasing patchsize will decrease the variance and range of the data and move the mean towards the true depth value of 700. Surprisingly, we find that for a patch size of 4 the range of depth values and the variance increase (figure 8 a) with respect to the single pixel case (figure 7 a). However, when both histograms are viewed at the same scale, it becomes apparent that this effect is due to a small number of outliers that appear in the patch estimation case. The distribution has in fact become narrower as we

had expected. In comparing the 9 pixel case 8 b), 9 b) we see that standard deviation and range have been reduced by 2 orders of magnitude with respect to the single pixel case - a significant improvement.

The appearance of the outliers can be explained theoretically. All outliers in these experiments were due to nearly vanishing denominators in the expression (41) for computing depth from patches. Compare this with the expression (24) used for depth from single pixel. In patches of uniform brightness the x and y derivatives of E which appear in the vectors **s** and **v** will be zero. In the single pixel estimation case of equation (24) this will not lead to a singularity unless E_t vanishes. In the patch case of equation (41), however, zero spatial derivatives cause a singularity.

Experiments run on the onnanoko sequence confirm these observations. Despite the appearance of outliers the distribution of depth values narrows considerably when larger patch sizes are employed. Although the improvement over the single-pixel depth estimation is considerable, estimating depth from two frames by such a procedure is rather unsatisfactory.

In practice, special measures should be taken to eliminate the depth outliers. A simple procedure is as follows: depth must be positive and is also bounded from above. Our implementation therefore includes a simple procedure which succeeds the patch depth estimation process. It searches for depth values that exceed the bounds and replace them by the average of neighbors that conform to the bounds.

4.5 Evaluating the variance computation procedure

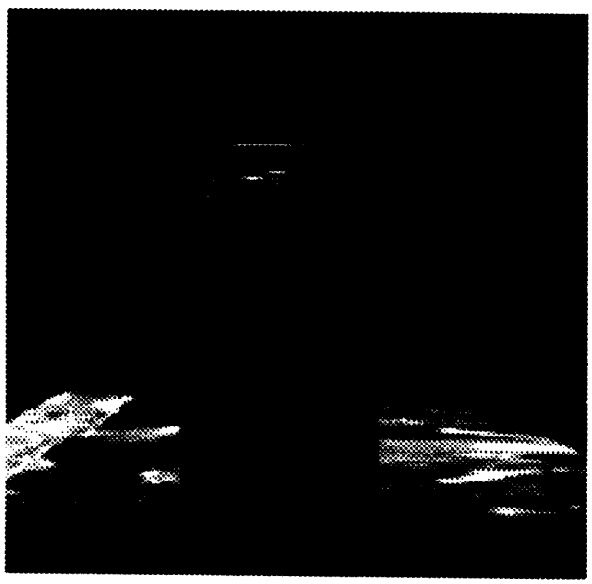

Figure 10: The pepsi scene

To evaluate the performance of the variance measure that we have introduced we make

use of yet another real image sequence. In this experiment shown in figure 10, a soda can was positioned on a table in front of the camera at a distance of 570 mm. The background is a plane parallel to the image plane at a distance of 1240 mm. The camera translated $\mathbf{t} = [1.5, 0, 0]$ mm between the two frames used to compute the gradient.

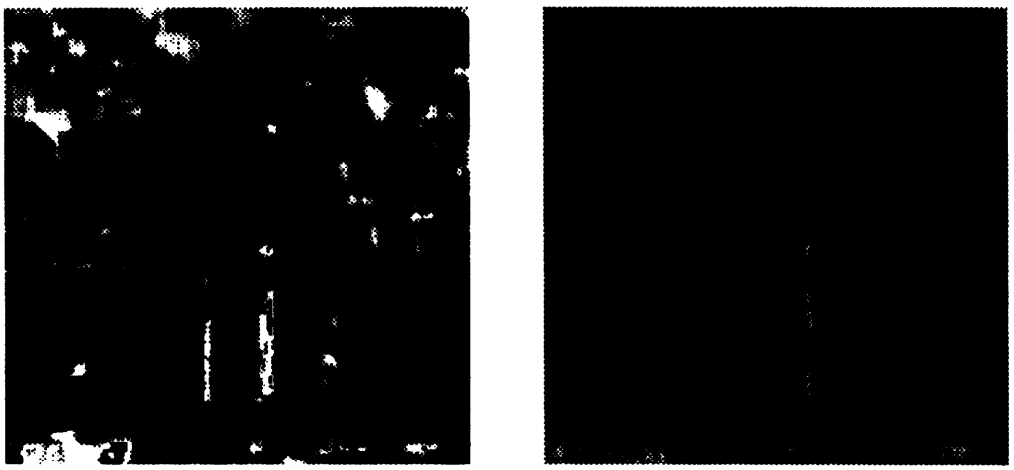

Figure 11: a) Depth and b) logarithm of variances from the pepsi scene using patch estimate with patch size 9.

Figures 11 a) and b) show the depth and variance computed by the patch estimator with a patch size of 9. The brightness of a point in these figures is proportional to the magnitude of the value at that location where dark points correspond to small values and light points correspond to large values. Due to the outliers in the depth map and the wide range of the variance values the brightness-encoded presentation of this data particularly difficult. For this reason the depth values have therefore been truncated to a range of [500, 3000]. This range contains over 97 % of all depth values. To present the variances we add 1 and take the natural logarithm. Figure 11 b) therefore illustrates the spatial distribution of variances qualitatively and the actual variation between the variances is orders of magnitudes larger than they appear thus enhancing the effects noted below.

Note the large errors in depth that appear on the lower right of the can and the frontal portion of the table the can is standing on. In comparing with the image we find that these regions are basically uniform in brightness so that a zero brightness gradient results. We therefore expect the depth estimate in those regions to be poor. As a consequence the variances there should be large. Note how the brighter areas in the variance figure pick out the areas in which the error in depth is large. Since the distribution of variances is in actuality orders of magnitude wider than the logarithmic intensity map suggests, poor depth estimates are very easily identified by their variance values. These observations not only support the approximation we have made to obtain the variance values but also suggests a way by which regions of poor depth estimates may be "filled in" using surrounding values of lower variance.

4.6 Algorithmic aspects of patch depth estimation

Finally a note on the algorithmic aspects of depth estimation using the direct patch method. The approach outlined above can be implemented in a straightforward way in time $O(n^2k^2)$ where n is the width/height of the image and k is the width/height of the patch. We simply compute the sums

$$S_1 = \sum_{x,y \epsilon P} (\mathbf{s} \cdot \mathbf{t})^2 \tag{44}$$

$$S_2 = \sum_{x,y \epsilon P} (\mathbf{v} \cdot \boldsymbol{\omega} + E_t)(\mathbf{s} \cdot \mathbf{t}) \tag{45}$$

for each of the $(n-k+1) \times (n-k+1)$ possible patches where each sum consists of $k \times k$ values.

Using the idea of running sums it is possible to reduce the computation time by a factor of k^2. Notice that the patch used to compute S_1 and S_2 for a given location (i,j) in the image differs only by one column of k elements from the patch used in the computation at location $(i+1,j)$. The idea is therefore to compute the full $k \times k$ sums only once (in the top left corner of the image for example) and to compute neighboring values by simply adding a new rightmost column and subtracting the leftmost column. The same idea can be used along the rows of the image, so that each pixel is only used once in the computation of all sums and the worst-case complexity is $O(n^2)$.

In practice, a circular buffer is used to hold the values of the sums in each column and row of the current patch. Some overhead is associated with maintaining this buffer so and errors will propagate through the entire image in this case. We therefore chose to apply the running sum technique only along each line of the image and recompute the $k \times k$ sum at the beginning of each line.

5 The Update Stage

As we see fr om the dynamic motion vision block diagram 4 the task of the update stage is to take as input a depth measurement Z and its variance p and combine it with a current estimate of depth $\hat{Z}$ and variance $\hat{p}$ to update the estimate.

5.1 The update equations

In our presentation of the recursive estimation theory we have introduced notation that will allow us to map the theoretical results directly onto the motion vision problem domain. The measurement Z_k in our case is the depth at a particular pixel (x, y) in the kth frame. The variance p_k is the variance of the depth. In the previous section we have outlined, how both values may be obtained directly from image brightness values.

It follows that one filter will maintain an estimate $\hat{Z}$ and its variance $\hat{p}$ at a particular pixel (x, y) in the image. We can think of having one filter at every pixel that estimates the value there. The update equation for each one of these filters is identical and extremely simple. We have

$$\hat{Z}_k \leftarrow \frac{Z_k/p_k + \hat{Z}_k/\hat{p}_k}{1/p_k + 1/\hat{p}_k} \tag{46}$$

$$\hat{p}_k \leftarrow \frac{1}{1/p_k + 1/\hat{p}_k} \tag{47}$$

which is taken directly from the simplified Kalman filter equations (12) and (13). The theoretical properties of this update process have been discussed in detail in subsection 3.1.

5.2 Evaluating the update process

To study the effect of the update procedure we show results obtained from the pepsi sequence introduced in the previous section (figure 10). The depth measurement Z_k and its variance p_k are computed from the brightness gradient using the patch depth estimate described in the previous section with a patch size of 9 pixels and using the known motion $\mathbf{t} = [1.5, 0, 0]$ mm. The resulting depth map is shown in figure 12.

For the update procedure we used a previous estimate $\hat{Z}_k$ corresponding to a frontal plane at 1000 mm distance. Since initially we have no information about the shape of the scene a flat depth map is a plausible way to start. The choice of initial variances $\hat{p}_k$ should be large so as to indicate the complete uncertainty about the current estimate. Since the square root of the variance is the standard deviation of the error we used a variance of $\hat{p} = 1000^2$.

Now we update the depth estimate and its variance according to the update equations (46), (47). The resulting depth map is shown in figure 12 b). As a result of the update, the flat depth map is almost completely replaced by the incoming measurement except in areas of large error as indicated by the variance values. locations. This is precisely the effect we expect from the update procedure and we can imagine how a repeated application of this process over time can significantly reduce the amount of error present in a single measurement.

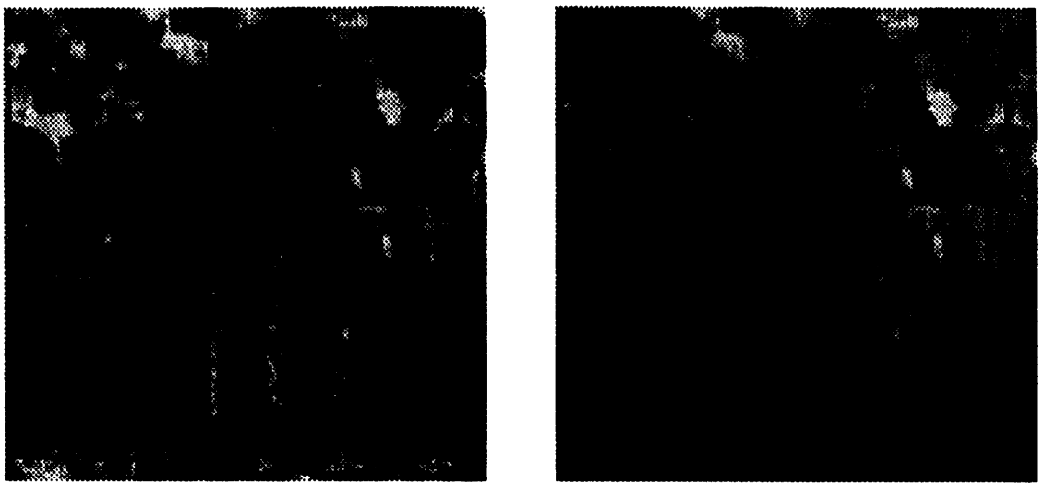

Figure 12: Depth maps from the pepsi scene a) measured using the patch estimator with patch size 9, b) after the update process with an initially flat depth map.

5.3 Algorithmic aspects of the update process

The serial complexity of the update process is easily seen to be $O(n^2)$ with n being the width/height of the image. Given $n \times n$ processors on a parallel machine this can be reduced to $O(1)$ since the update equations are local to each pixel. The filter update is indeed the fastest of all operations that constitute the dynamic motion vision algorithm.

6 The Prediction Stage

The task of the prediction stage as we see from the block diagram of figure 4 is to determine how a depth and variance map at time k will appear to the observer at time $k+1$ assuming that the relative motion of observer and surface are given by the motion vectors $\mathbf{t}$ and ω. Informally this may be stated as follows: given an estimated depth map and a 3D motion, what will the depth map look like after the motion?

We state the problem more formally. Let (X, Y, Z) denote coordinates of a point in the real world and (x, y) denote the coordinates of its projection in the image plane. Given an array of depth values $Z(x_j, y_i)$ for $i = 0, \ldots, h-1$ and $j = 0, \ldots, w-1$ (i.e. on a regular grid) and a motion $\mathbf{t} = [U, V, W]^T$, $\omega = [A, B, C]^T$, what are the values of $\hat{Z}(x_j, y_i)$ after the given motion?

Since the Z values are known only at the grid points, it does not suffice to simply transform the 3D coordinates corresponding to the given motion. The transformed values may not coincide with grid points, they may have become occluded or moved out of the field of view. A resampling of the warped surface and hence some form of interpolation between the given samples is necessary.

An additional problem is the following: We interpret the given values of $Z(x_j, y_i)$ as normally distributed random variables with variance $p(x_j, y_i)$. What are the value of $p(x_j, y_i)$ after the warping? The answer to this question is a integral part of the dynamic motion vision algorithm. We present here the algorithm for warping the depth map $Z(x_j, y_i)$. The transformation of variances presents no fundamentally different challenges but is rather lengthy in nature so we chose to attach it as an appendix (Appendix A).

6.1 Warping the depth map: outline of the algorithm

The outline of the algorithm is as follows:

1. Using the equations of perspective projection we compute the 3D coordinates corresponding to the given depth values. Thus, we have the transformation
$$(x_{ij}, y_{ij}, Z_{ij}) \rightarrow (X_{ij}, Y_{ij}, Z_{ij}) \tag{48}$$
2. Using the equations of motion we tranform the 3D points into the new coordinate system. The transformation is
$$(X_{ij}, Y_{ij}, Z_{ij}) \rightarrow (X'_{ij}, Y'_{ij}, Z'_{ij}) \tag{49}$$
3. For each pixel (x, y) in the new coordinate system we seek to determine the location in 3D where the ray through (x, y) intersects a surface through the points computed in the previous step. The Z coordinate of the intersection becomes the Z value stored at (x, y). This procedure is referred to as resampling.
4. Grid values of Z that remained undetermined after the previous resampling step (parts of the surface which enter the field of view due to the motion) must be filled. We extrapolate from the known values in a straightforward way.

We now elaborate the details of the above algorithm.

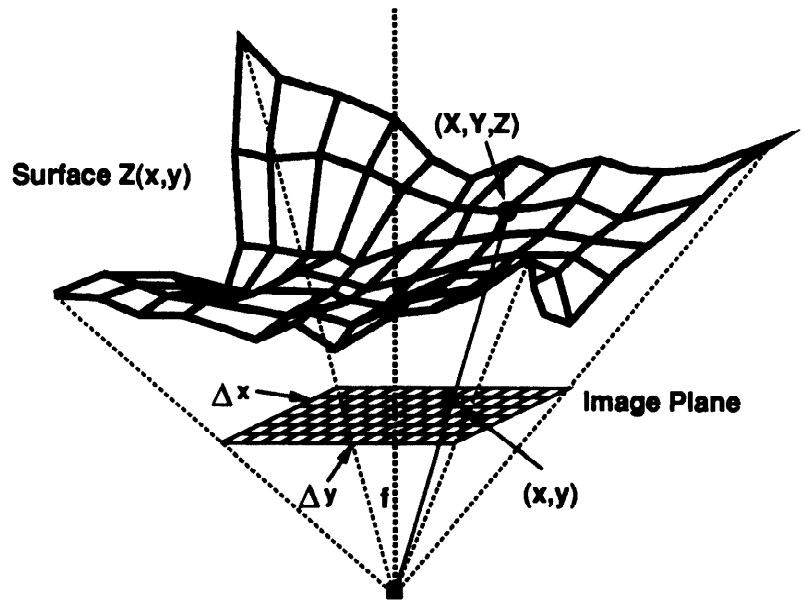

Figure 13: A depth surface $Z(x,y)$ before prediction warping.

6.1.1 The initial situation

We start with $Z(x_j, y_i) = Z_{ij}$ for $i = 0, \ldots, h-1$ and $j = 0, \ldots, w-1$ as the depth map. x_j and y_j are given by

$$x_j = (j - \frac{w-1}{2})\Delta x \qquad \text{and} \qquad y_i = (i - \frac{h-1}{2})\Delta y \tag{50}$$

where w, h are the dimensions of the camera pixel array and Δx, Δy are the spacings of pixels in horizontal and vertical directions. This situation is shown in figure 13. The motion is given as $\mathbf{t} = [U, V, W]^T$ and $\omega = [A, B, C]^T$.

6.1.2 Inverse Projection

We compute the 3D coordinates of all the points whose depth is stored in the depth map using inverse perspective projection. The goemetry of perspective projection along the x direction is shown in figure 14. We compute

$$X_{ij} = \frac{x_j Z(x_j, y_i)}{f} \qquad \text{and} \qquad Y_{ij} = \frac{y_i Z(x_j, y_i)}{f} \tag{51}$$

for $i = 0, \ldots, h-1$ and $j = 0, \ldots, w-1$. The focal length is denoted by f.

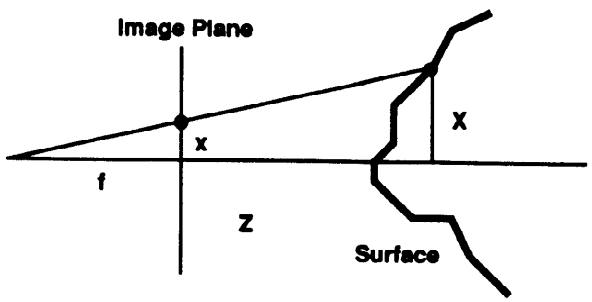

Figure 14: The perspective projection situation along the x direction.

6.1.3 Warping

We transform all the 3D points into the coordinate system obtained after preforming the given motion t, ω. Transformed points will be denoted by a prime. We have

$$\begin{bmatrix} X'_{ij} \\ Y'_{ij} \\ Z'_{ij} \end{bmatrix} = - \begin{bmatrix} U \\ V \\ W \end{bmatrix} - \begin{bmatrix} 1 & -C & B \\ C & 1 & -A \\ -B & A & 1 \end{bmatrix} \begin{bmatrix} X_{ij} \\ Y_{ij} \\ Z_{ij} \end{bmatrix} \tag{52}$$

for $i = 0, \ldots, h-1$ and $j = 0, \ldots, w-1$.

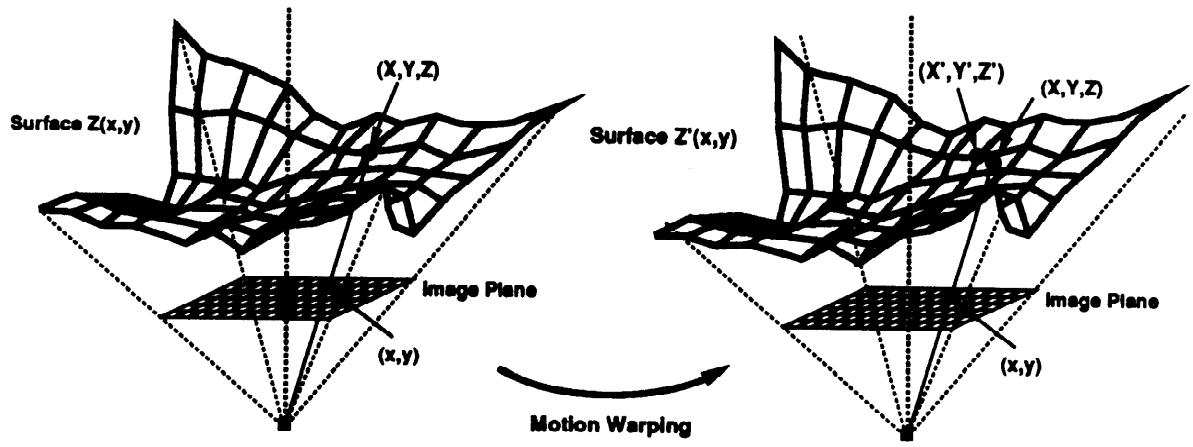

Figure 15: The motion warping.

The effect of this transformation is conceptually equivalent to moving the depth wire frame in space according to the given motion. The result can be visualized as in figure 15.

6.1.4 Resampling

Now we encounter the main problem in the prediction algorithm. If we project the points $(X'_{ij}, Y'_{ij}, Z'_{ij})$ back into the image plane they will most likely not coincide with the grid points as shown in figure 15. In other cases the backprojected points may have become occluded by other points or have moved out of the image plane. We handle these problems as follows: we interpret each point (X', Y', Z') obtained from the warping step as a sample of a 3D surface and the objective is to resample this surface at the grid points. This requires an assumption about the structure of the surface inbetween the sample points.

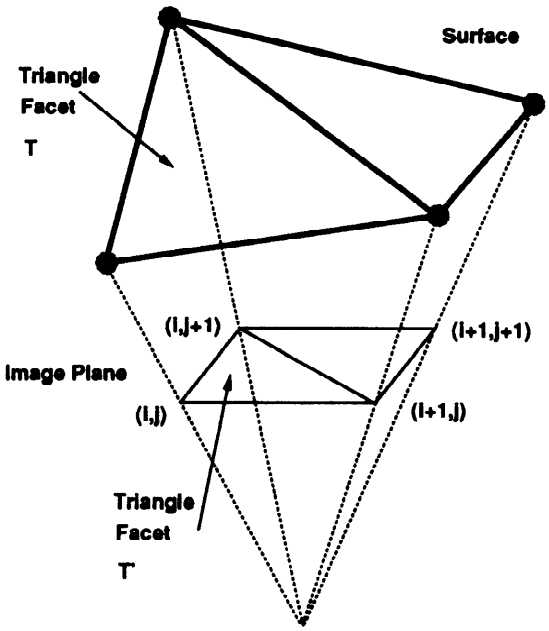

Figure 16: Triangular facet subdivision.

We will consider each grid square in the array. Such a square is given by its corner coordinates $(i,j),(i+1,j),(i,j+1),(i+1,j+1)$. For each of the four corners there is a warped 3D surface point $(X'_{ij}, Y'_{ij}, Z'_{ij})$ available. We divide the grid square into two triangles

$$(i,j),(i+1,j),(i,j+1) \tag{53}$$

$$(i+1,j+1),(i+1,j),(i,j+1) \tag{54}$$

as shown in figure 16. The three points in 3D corresponding to these corners define a plane in space which we will use as our surface approximation. Most importantly the spatial triangle T between these points prespectively projects into a triangle T' in the image plane. All grid points inside T' have rays which intersect the spatial triangle T.

The value of Z which we obtain by this intersection procedure, however, need not be the truly visible depth value, since the currently considered spatial triangle may be occluded by other triangles. But if we repeat the procedure of computing the intersection for all possible spatial triangles, then the smallest value of Z obtained at a given grid point will be the truly visible depth value.

1. **Determining the image plane projections of the spatial triangles**

   For every spatial triangle T we have the 3D coordinates of the corners $[X_1, Y_1, Z_1]^T$, $[X_2, Y_2, Z_2]^T$ and $[X_3, Y_3, Z_3]^T$. It is intuitively clear and can be shown analytically, that this spatial triangle T perspectively projects into a triangle T' in the image plane. The corners of this triangle are (x_1, y_1), (x_2, y_2) and (x_3, y_3) where

   $$x_k = f\frac{X_k}{Z_k} \qquad \text{and} \qquad y_k = f\frac{Y_k}{Z_k} \tag{55}$$

   for $k = 1, 2, 3$.

2. **Determining the grid points inside the triangle T'**

   The rays through all grid points (x, y) inside T' intersect the corresponding spatial triangle T i.e. the warped surface and it is these intersection points that we would like to determine. The first step is to compute all grid points inside T'.

   We first determine the bounding box of the given triangle. The x coordinates are between x_{min} and x_{max} given by

   $$x_{min} = \min(x_1, x_2, x_3) \qquad \text{and} \qquad x_{max} = \max(x_1, x_2, x_3) \tag{56}$$

   and similarly we compute bounds y_{min}, y_{max} on the y coordinates. This determines a square in the grid. The grid points inside this square have coordinates in the ranges $[i_{min}, i_{max}]$ and $[j_{min}, j_{max}]$ where

   $$j_{min} = ceil(\frac{x_{min}}{\Delta x} + \frac{w-1}{2}) \qquad \text{and} \qquad j_{max} = floor(\frac{x_{max}}{\Delta x} + \frac{w-1}{2}) \tag{57}$$

   $$i_{min} = ceil(\frac{y_{min}}{\Delta y} + \frac{h-1}{2}) \qquad \text{and} \qquad i_{max} = floor(\frac{y_{max}}{\Delta y} + \frac{h-1}{2}) \tag{58}$$

   in which $ceil(x)$ is the smallest integer larger than x and $floor(x)$ is the largest integer smaller than x. In addition, we must insure that the values j_{min}, j_{max}, i_{min} and i_{max} are actually grid points i.e. that they are in the range $[0, w-1]$ or $[0, h-1]$ respectively. If this is not the case, we set them to the closest value which is inside the grid.

   For all grid points with indices (i, j) in the computed range we must now determine, if they actually lie inside the triangle T'. We can use a simplified polygon containment test: Arrange the vertices of the triangle in counterclockwise order and then test if the candidate point lies in the left half plane of all of the three edges as we proceed in counterclockwise direction.

More formally, to test

$$x = (j - \frac{w-1}{2})\Delta x \tag{59}$$

$$y = (i - \frac{h-1}{2})\Delta y \tag{60}$$

for containment in (x_1, y_1) - (x_2, y_2) - (x_3, y_3) we first compute

$$(x_3 - x_1)(y_2 - y_1) - (y_3 - y_1)(x_2 - x_1). \tag{61}$$

If this is positive, then the given sequence of points is in clockwise order and we switch (x_2, y_2) and (x_3, y_3) to obtain counterclockwise order.

Now we compute the three values

$$(x_2 - x_1)(y - y_1) - (x - x_1)(y_2 - y_1) \tag{62}$$

$$(x_3 - x_2)(y - y_2) - (x - x_2)(y_3 - y_2) \tag{63}$$

$$(x_1 - x_3)(y - y_3) - (x - x_3)(y_1 - y_3) \tag{64}$$

If all three values are positive, the point (x, y) is inside the triangle. Each test determines containment in one left half-plane. For each spatial triangle T we thereby obtain a (possibly empty) set of image plane grid points (x, y) which have rays intersecting T.

3. Determining the intersections of the grid rays with the spatial triangles

For each of the grid points (x, y) the ray has the equation

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \\ f \end{bmatrix} \tag{65}$$

for positive λ. The plane through the spatial triangle T has the equation

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \alpha \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} + \beta \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} + (1 - \alpha - \beta) \begin{bmatrix} X_3 \\ Y_3 \\ Z_3 \end{bmatrix}. \tag{66}$$

In computing the intersection of the ray with the plane we obtain three linear equations for the three unknowns α, β and λ. Knowledge of λ is sufficient to determine the point of intersection.

The determinant of the system is

$$\begin{aligned} D = {} & (X_1 - X_3)[(Z_2 - Z_3)y - (Y_2 - Y_3)f] - \\ & (Y_1 - Y_3)[(Z_2 - Z_3)x - (X_2 - X_3)f] + \\ & (Z_1 - Z_3)[(Y_2 - Y_3)x - (X_2 - X_3)y] \end{aligned} \tag{67}$$

and the adjunct of λ is

$$\begin{aligned} D_\lambda &= -X_3[(Y_1 - Y_3)(Z_2 - Z_3) - (Y_2 - Y_3)(Z_1 - Z_3)] \\ &\quad +Y_3[(X_1 - X_3)(Z_2 - Z_3) - (X_2 - X_3)(Z_1 - Z_3)] \\ &\quad -Z_3[(X_1 - X_3)(Y_2 - Y_3) - (X_2 - X_3)(Y_1 - Y_3)]. \end{aligned} \tag{68}$$

So if D is nonzero we have

$$\lambda = \frac{D_\lambda}{D} \tag{69}$$

and the intersection point is given by

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \\ f \end{bmatrix} \tag{70}$$

The case where $D = 0$ is mathematically possible in two cases, only one of which can occur here. In the first case the ray and the plane do not intersect. This cannot occur since we have selected the rays in such a way that they must intersect the triangle.

The second case is where the ray lies in the plane defined by the spatial triangle. In this case there are are infinitely many solutions which will manifest itself in $D = 0$. In this case we would like to have the intersection point with the smallest value of Z. This point will lie on one of the three triangle edges. We can therefore compute the intersection of the ray with the three triangle edge segments (there must be at least two intersections) and select the one with the smallest depth value.

Each edge segment of the spatial triangle is given by two points in space $[X_1, Y_1, Z_1]^T$ and $[X_2, Y_2, Z_2]^T$. The equation of the line is

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \alpha \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} + (1 - \alpha) \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} \tag{71}$$

which we must intersect with the ray

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \\ f \end{bmatrix} \tag{72}$$

for $\lambda \geq 0$ and $0 \leq \alpha \leq 1$. This yields three equations for the two unknowns λ and α. Since we know (by the fact that $D = 0$) that the two must intersect, we can use any two of the equations and compute the determinant

$$d = x(Y_1 - Y_2) - y(X_1 - X_2) \tag{73}$$

as well as the adjuncts

$$d_\alpha = -yX_2 + xY_2 \tag{74}$$

$$d_\lambda = (X_1 - X_2)Y_2 - (Y_1 - Y_2)X_2 \tag{75}$$

If d is nonzero we have

$$\alpha = \frac{d_\alpha}{d} \qquad \text{and} \qquad \lambda = \frac{d_\lambda}{d} \tag{76}$$

If $\alpha < 0$ or $\alpha > 1$ the ray intersects the line outside the segment between the two given points and we try the next triangle edge. $\lambda < 0$ is not possible. Having tested all these conditions, the desired intersection is

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \\ f \end{bmatrix}. \tag{77}$$

Finally, if d is zero, then the triangle edge segment coincides with the ray and the desired point is the segment endpoint with the smallest depth value. In other words if $Z_1 < Z_2$ the intersection point is $[X_1, Y_1, Z_1]^T$ otherwise $[X_2, Y_2, Z_2]^T$.

4. **Creating the resampled depth map**

Each spatial triangle T may lead to depth values Z for a number of different grid point locations (x, y). Some grid points may have multiple values of Z assigned, if the ray through that point intersects the 3D surface more than once. Among these points we are interested in the one closest to the camera, for it will occlude all others. Hence we simply ignore all depth values Z at a particular point except the smallest one.

In practice we will initialize the depth map Z_{ij} with ∞ (some very large value) before resampling the warped depth surface as described above. Whenever a new value Z is computed for location (i, j) we replace Z_{ij} with Z if and only if $Z_{ij} > Z$. After each grid triangle has been considered, Z_{ij} will contain only the minimal values of depth computed at that point.

6.1.5 Extrapolating to unassigned grid points

It is easily conceivable that some grid points will not have been assigned any value of Z after the resampling step. This will be reflected by the fact that some depth map values are still ∞ after all grid triangles have been considered. Physically these points correspond to surface points which have become visible due to the relative motion of camera and scene, so that no depth information is available there.

In general, we can say nothing about the correct depth value at these locations. If, however, we assume that the surface is somewhat smooth then we can extrapolate the depth values at these points from known values at neighboring locations.

In our current algorithm, we search the depth map for unassigned locations (where $Z = \infty$). Suppose (i, j) is such a grid location. We then search all neighboring pixels (with coordinates differing by -1, 0 or 1 from (i, j)) for finite values of Z. Say we find n such pixels. Let the sum of the Z values of these pixels be ΣZ. Then we assign the value

$$Z = \frac{\Sigma Z}{n} \tag{78}$$

at location (i,j). If all neighboring pixels were also unassigned, we increase the radius of our search from 1 to 2 and try again. If the search radius exceeds the size of the image then none of the grid points has been assigned any value. This would mean that none of the previously visible surface part is visible in the current image.

6.1.6 Summary of the algorithm

Below we briefly summarize the steps of the depth prediction algorithm using the notation from above

1. Initialize the reinterpolation depth map to $\hat{Z}_{ij} = \infty$ for all i,j.

2. Determine the 3D coordinates of each point stored in the current depth map using the inverse perspective projection:

$$X_{ij} = \frac{x_{ij} Z_{ij}}{f} \qquad \text{and} \qquad Y_{ij} = \frac{y_{ij} Z_{ij}}{f} \tag{79}$$

3. Warp the 3D points into the new coordinate frame:

$$\begin{bmatrix} X'_{ij} \\ Y'_{ij} \\ Z'_{ij} \end{bmatrix} = - \begin{bmatrix} U \\ V \\ W \end{bmatrix} - \begin{bmatrix} 1 & -C & B \\ C & 1 & -A \\ -B & A & 1 \end{bmatrix} \begin{bmatrix} X_{ij} \\ Y_{ij} \\ Z_{ij} \end{bmatrix} \tag{80}$$

4. Project each 3D point back into the image plane

$$x'_{ij} = f \frac{X'_{ij}}{Z'_{ij}} \qquad \text{and} \qquad y'_{ij} = f \frac{Y'_{ij}}{Z'_{ij}} \tag{81}$$

5. Each grid square $(i,j)-(i+1,j)-(i,j+1)-(i+1,j+1)$ is divided into two triangles $(i,j)-(i+1,j)-(i,j+1)$ and $(i+1,j+1)-(i+1,j)-(i,j+1)$. For each such triangle the previous step has produced three corner points (x',y') in the warped image plane. We determine all the grid points in the warped grid which fall into this triangle. This is done by first computing the bounding box of the triangle and then testing each grid point in the bounding box for containment in the triangle. The ray through each grid point found to be in the triangle will intersect the corresponding spatial triangle.

6. For each grid point (i,j) inside of a triangle we compute the intersection of the ray through that gridpoint with the corresponding spatial triangle according to one of the three cases discussed in subsection 3. If the Z coordinate of the intersection is smaller than the current value of $\hat{Z}_{ij}$ we replace $\hat{Z}_{ij}$ by the new Z.

7. After all grid triangles have been considered, we search the depth map $\hat{Z}_{ij}$ for ∞. For each such grid point found we extrapolate Z from the closest gridpoints with $\hat{Z} \neq \infty$ as described above.

Using the procedure described above and some simple facts about the propagation of variances we can also determine the variances of the warped and resampled depth map. These variances are needed for the operation of the recursive estimation procedure in the next iteration. Due to the lengthy nature of the derivation we have attached it as Appendix A for the interested reader.

6.2 Algorithmic aspects of the prediction stage

Suppose that the depth map consists of $n \times n$ values. This means there are $2(n-1)^2$ triangular facets which must be considered for backprojection after the warping stage. In the worst case, the backprojection of each facet could subsume the entire image plane and therefore necessitate the update of n^2 depth and variance values. The worst case complexity is therefore $O(n^4)$. For most well-behaved surfaces and small motions, however, the the backprojection of each triangle will subsume a number of pixels which can be bounded by a small constant.

A parallel machine could devote one processor to the computation associated with one triangular facet. As we pointed out the result of this computation may effect all n^2 pixels in the image plane so $O(n^2)$ operations are necessary at each processor. The prediction part of the algorithm is computationally most expensive.

7 The Motion Estimation Stage

In our previous discussion we have seen that both the depth measurement and the Kalman filter prediction stage require knowledge of the parameters of rigid body motion $\mathbf{t}$ and $\boldsymbol{\omega}$. In some cases, the motion may actually be known from some external source, for example, when our camera is mounted on a vehicle for which the motion can be measured or commanded precisely. If so, this information can be used and the subsequently described motion estimation module is not needed. If, however, motion information is not available, the motion must be determined from the image sequence.

7.1 Least squares motion estimation

As we pointed out before, motion estimation using the brightness change constraint equation (8) is not possible since the depth Z is unknown. At this point, we can exploit the iterative nature of the recursive estimation process. At every point in time, the filter produces an estimate of the depth $\hat{Z}_k$. We use this estimate of the depth map as input to the motion estimation stage as shown in the block diagram 4. The task of the motion estimation stage is therefore: given a depth map $Z(x, y)$ and the brightness gradients E_x, E_y, E_t compute the motion $\mathbf{t}$ and $\boldsymbol{\omega}$.

If the depth Z is given, the brightness change constraint equation (8) contains 6 unknowns: the motion vectors $\mathbf{t}$ and $\boldsymbol{\omega}$. We obtain one such linear constraint equation for every pixel in our image. Instead of selecting 6 of these equations and solving them for the desired parameters we formulate a least-squares problem using the constraints at every pixel within an image patch in order to reduce the effect of noise (see Horn, Weldon [28]).

More formally, we have

$$\min_{\mathbf{t},\boldsymbol{\omega}} \sum_x \sum_y \left(\frac{\mathbf{s} \cdot \mathbf{t}}{Z} + \mathbf{v} \cdot \boldsymbol{\omega} + E_t\right)^2 \tag{82}$$

which we differentiate with respect to the motion vectors to obtain the necessary condition for a minimum

$$\left(\sum_x \sum_y \frac{\mathbf{s}\mathbf{s}^T}{Z^2}\right)\mathbf{t} + \left(\sum_x \sum_y \frac{\mathbf{s}\mathbf{v}^T}{Z}\right)\boldsymbol{\omega} = -\sum_x \sum_y \frac{E_t \mathbf{s}}{Z} \tag{83}$$

$$\left(\sum_x \sum_y \frac{\mathbf{v}\mathbf{s}^T}{Z}\right)\mathbf{t} + \left(\sum_x \sum_y \mathbf{v}\mathbf{v}^T\right)\boldsymbol{\omega} = -\sum_x \sum_y E_t \mathbf{v} \tag{84}$$

The above summations are carried out over all pixels in the image region of interest. This linear system (83), (84) of 6 equations can be easily computed from the brightness gradients and the given depth Z. It is noteworthy in this context that the system matrix is symmetric so that the number of coefficients to compute can be halved. Any standard technique (see [52]) can be used to solve the linear system.

7.2 Evaluating the motion estimation procedure

A crucial point is of course the fact that the depth map used in the motion estimation is not the true depth map but the depth map produced by the filter, which is subject to

errors. An important issue is therefore how the motion estimation procedure performs in the presence of errors in the depth map. This is difficult to evaluate, since true depth maps for scenes are rarely available. This is where the experiments on planar scenes come in handy. Planar depth maps are trivial to create and the true depth of the experimental scene can be measured quite accurately. Moreover the error in a typical depth estimate as it is produced by the patch depth estimation procedure is approximately Gaussian (figure 8) and can therefore be recreated easily in an experiment.

We used the scenes "wave" and "onnanoko" shown in figures 6 a) and b) of section 3.2. In each case we have frontal planes at 700 mm and 500 mm respectively. We created depth maps containing these values at every point (x, y) with additive Gaussian noise of varying standard deviation. We then used these depth maps and the brightness gradients computed from the images to the motion estimation procedure described above.

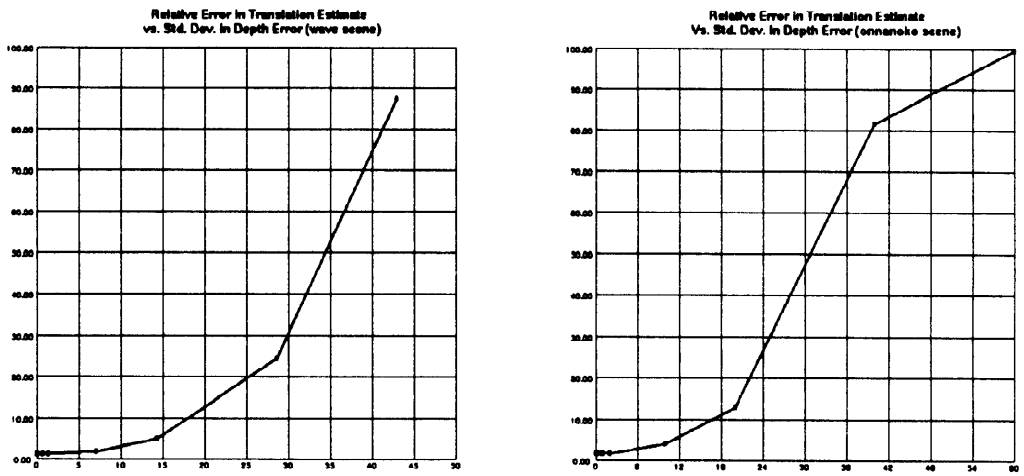

Figure 17: Relative error in the motion estimate vs. standard deviation of the depth error (normalized by true depth) a) wave scene, b) onnanoko scene. Both axes are labeled in percent.

We evaluate the error in the resulting motion estimate using the measure (both motions were pure translations):

$$\varepsilon = \frac{|\hat{\mathbf{t}} - \mathbf{t}|}{|\mathbf{t}|} \tag{85}$$

where $\hat{\mathbf{t}}$ is the estimate obtained for the translation vector and $\mathbf{t}$ is the true value. This is the normalized error in the vector. The measure ε was computed using depth maps with noise of standard deviation 0, 1, 5, 10, 50, 100, 200 and 300 mm. The resulting values of the error in the motion as a function of the noise standard deviation (normalized by the true depth) are shown in figures 17 a) and b) for the wave and onnanoko scenes respectively.

Both results show that 25% noise results in about 20% error in the motion estimate.

For the onnanoko scene with a true depth of 500 mm, for instance, this means, that when 68 % of all depth values are in the range $[375, 625]$, a motion estimate of $\hat{\mathbf{t}} = [1.419762, 0.046511, -2.713889]^T$ (vs. the actual $\mathbf{t} = [1.5, 0, -3.0]^T$) is obtained. This indicates that the motion estimation procedure is quite robust with respect to errors in depth.

This robustness of the motion estimator is extremely important for the convergence of the filtering routine. In the filtering procedure, the largest errors in depth occur initially, when no information about depth is available. Initially, we start with a flat depth map and may potentially have large errors in depth. Robustness of the motion estimator is essential to prevent divergence at that point.

The motion estimator can be viewed from another perspective. The motion vectors are parameters in the Kalman filter formulation. Adding the motion estimator is equivalent to fitting these parameters to the data in a least squares fashion in every iteration. The proposed approach is therefore a least-squares adaptive filter. A distinct advantage of this approach is the fact that no particular care must be taken in the choice of the initial depth value, since the motion estimator automatically adjusts the motion parameters to fit the depth data by exploiting the motion-depth ambiguity. The initial depth value does not determine whether or not the algorithm converges, it merely sets the absolute values of motion and depth values it will eventually converge to. If fixed (known) motion is used, this is not true.

The above experiments were carried out using purely translational motion only. Of course the formalism holds for arbitrary rotational and translational motion. However, we discovered, that the motion estimation does not exhibit the necessary robustness discussed above, when all 6 degrees of motion are admitted. This does not mean that rotational motion cannot be handled. Our experiments showed similarly stable results as shown above in the case of one rotational and 2 translational degrees of freedom (as in a mobile robot for example). For our presentation, we chose to restrict our experiments to purely translational motion only.

7.3 Algorithmic aspects of the motion estimation process

The complexity analysis is quite simple. Each sum in the formulas (83), (84) contains $O(n^2)$ elements so that $O(n^2)$ operations will be necessary. Speedups can be achieved by noting that the system matrix in these equations is symmetric. On a parallel machine the summations can be carried out in $O(log(n))$ time using a tree connection structure of the individual processors.

8 Smoothing and Filling In

8.1 High variance regions: the need for surface reconstruction

The adaptive recursive estimation scheme described so far has the following limitation: Given a sequence of measurements, it will choose an estimate that best matches the measurements according to their quality as indicated by the variances. Conceptually, we start with some initial value for the depth estimate and each measurement pulls the estimate towards the true value. The smaller the variance, the stronger the pull.

If, however, the measurements at a given point (x, y) in the image plane always exhibit a high variance, then the estimate will never move from its initial value. In essence, no information about the actual value is supplied to the filter so the estimate is never changed. A practical situation that demonstrates this is the pepsi scene. Shown in figure 18 are an image from the pepsi sequence, the depth map and the logarithm of the variance map.

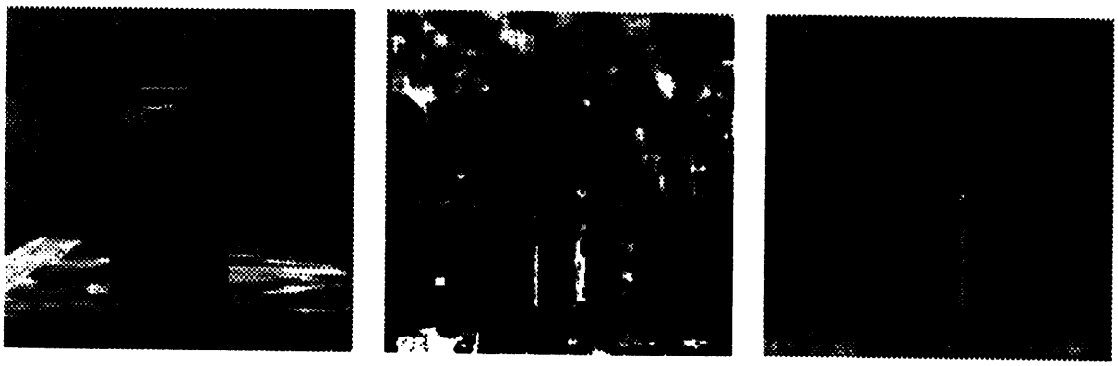

Figure 18: a) Pepsi scene b) Depth map computed with patch estimate (size 9) and c) logarithm of variances.

As we have already noted in section 3.2, the lower right portion of the pepsi can has large regions of uniform brightness. Large errors in the depth values result and the regions have high variance values (high brightness in the variance map). Moreover, the measurements obtained in these regions exhibit high variances throughout the entire sequence i.e. there is never any useful information available for depth estimation at these points.

From the recursive estimation point of view, this cannot be helped. It is, however, rather unsatisfactory as the result of an image analysis algorithm. To improve this situation we assume that the surface being observed exhibits some amount of smoothness so that "bad" values may be filled in by neighboring "good" values. In essence, we have a surface reconstruction problem. Note that the surface reconstruction is completely orthogonal to the dynamic motion vision algorithm from the estimation theoretical point of view. Its effects are an improvement of the obtained depth maps that make them more intuitively plausible.

8.2 MRF surface reconstruction and why it fails

Reconstructing smooth surfaces is a problem that has been dealt with extensively and we do not attempt to make a contribution to this field. Naive solutions such as smoothing or lowpass filtering the depth map fail when discontinuities are present (blurring of edges). Mathematical foundations for smooth surface reconstruction in the presence of discontinuities were laid by Geman and Geman [18]. Work by Grimson [19], Terzopoulos [48], Marroquin [33] and Blake and Zisserman [6] explored applications to vision. The stochastic relaxation algorithms used there are computationally quite intensive. Geiger and Girosi [16] recently developed a deterministic approximation to Markov random fields based on mean field theory which is computationally quite simple.

We briefly summarize the results from [16] as they apply to our surface reconstruction application. Given a field Z_{ij} on a regular lattice of points (i,j) we seek to determine a field $\tilde{Z}_{ij}$ which is

1. "close" to the input data by some measure,
2. locally "smooth" by some measure except at locations where Z_{ij} exhibits discontinuities.

Formally, these measures of "closeness" and "smoothness" are captured by an "energy function" which the desired surface function $\tilde{Z}_{ij}$ should minimize

$$E = E_{close} + E_{smooth} + E_{line} \tag{86}$$

where

$$E_{close} = \sum_{i,j}(\tilde{Z}_{ij} - Z_{ij})^2 \tag{87}$$

$$E_{smooth} = \alpha \sum_{i,j}[(\tilde{Z}_{i+1,j} - \tilde{Z}_{ij})^2(1 - h_{ij}) + (\tilde{Z}_{i,j+1} - \tilde{Z}_{ij})^2(1 - v_{ij})] \tag{88}$$

$$E_{line} = \gamma \sum_{i,j}(h_{ij} + v_{ij}) \tag{89}$$

The first term enforces "closeness" as before, the second "smoothness" except at locations where the "line processes" h or v are 1 (at discontinuities) and the third term is needed to prevent line processes from being created everywhere. Line process fields have value ranges $[0,1]$ at every site (i,j) where 1 indicates the presence and 0 the absence of a discontinuity at that location.

Using mean field theory, Geiger and Girosi derived the following iterative scheme to determine the field $\tilde{Z}_{ij}$ and the line processes v_{ij}, h_{ij} that minimize the above energy functional:

$$\tilde{Z}_{ij}^{n+1} = \tilde{Z}_{ij}^{n} - \omega[\tilde{Z}_{ij}^{n} - F(Z, \tilde{Z}^n, h^n, v^n, i, j)] \tag{90}$$

$$h_{ij}^{n+1} = H(f^n, h^n, i, j) \tag{91}$$

$$v_{ij}^{n+1} = V(f^n, v^n, i, j) \tag{92}$$

where

$$
\begin{aligned}
F(Z,\tilde{Z},h,v,i,j) &= Z_{ij} - \\
&\quad \alpha[(\tilde{Z}_{ij} - Z_{i,j-1})(1 - v_{i,j-1}) - (\tilde{Z}_{i,j+1} - Z_{i,j})(1 - v_{i,j}) + \\
&\quad (\tilde{Z}_{ij} - Z_{i-1,j})(1 - h_{i-1,j}) - (\tilde{Z}_{i+1,j} - Z_{ij})(1 - h_{i,j})] \qquad (93)
\end{aligned}
$$

$$
H(f,h,i,j) = \frac{1}{1 + e^{\beta(\gamma - \alpha(\tilde{Z}_{i+1,j} - \tilde{Z}_{ij})^2)}} \qquad (94)
$$

$$
V(f,v,i,j) = \frac{1}{1 + e^{\beta(\gamma - \alpha(\tilde{Z}_{i,j+1} - \tilde{Z}_{ij})^2)}} \qquad (95)
$$

and n is the iteration index.

The parameters α, β, γ and ω play different roles in this formulation. The values of β and values of ω were set to $\beta = 10$ and $\omega = 0.01$ as they were found to influence the convergence behavior, but not the final result. The relationship between α and γ determines the tradeoff between smoothing and line processes. Geiger and Girosi show that for absolute gradient values larger than $\sqrt{\gamma/\alpha}$ line processes will be created to inhibit smoothing. γ and α should therefore be chosen so that $\sqrt{\gamma/\alpha}$ corresponds to a threshold value above which one wishes to introduce a discontinuity and prevent smoothing.

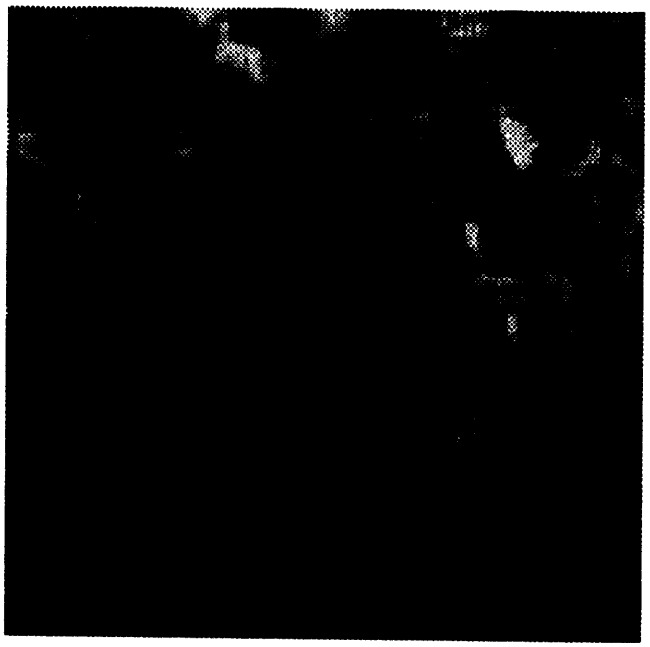

Figure 19: The depth map for the pepsi scene after one iteration of the filter. The patch depth estimator of size 9 and the motion estimator were used. The initial depth map is flat 1000 mm.

We have implemented the above iterative solution to the mean field approximation to an MRF for smoothing. Figure 19 shows the depth map result of the update stage. Figures

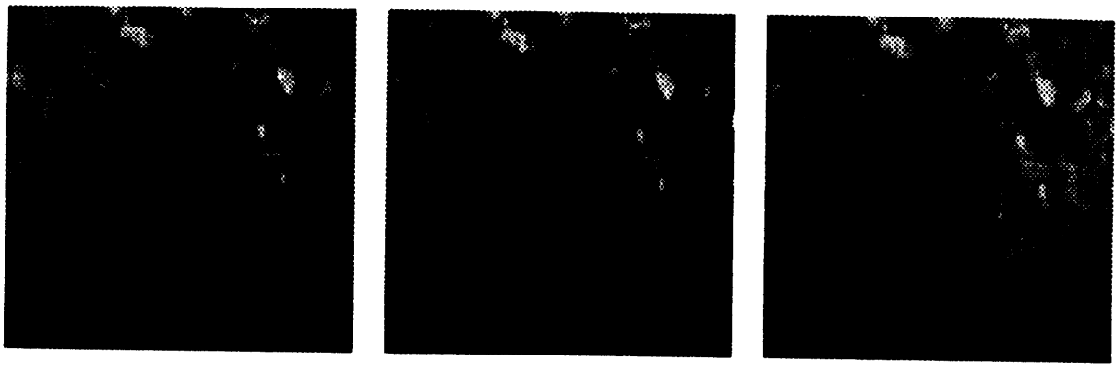

Figure 20: The depth maps from the pepsi scene with a) $\gamma = 160000$, b) $\gamma = 250000$ and c) $\gamma = 360000$ (see text).

20 a), b) and c) show the result of applying the MRF algorithm for 3 iterations with parameters

a) $\alpha = 4$, $\gamma = 160000$ (edge threshold = 200)

b) $\alpha = 4$, $\gamma = 250000$ (edge threshold = 250)

c) $\alpha = 4$, $\gamma = 360000$ (edge threshold = 300)

The input field range is [459, 2967]. Figures 21 a), b) and c) show the line processes as

$$e_{ij} = h_{ij} + v_{ij} \tag{96}$$

where black dots indicate the location of a boundary and white points are locations where smoothing was enabled.

Only few iterations were used but they suffice to reveal a very important fact. While the depth maps are virtually identical to the observer, the discontinuity map shows that strong discontinuities are established along the regions of constant brightness on the can surface. They prohibit information from propagating into the regions of high variance. So while smoothing is achieved the filling in cannot be accomplished with the MRF scheme, because variance information is not incorporated.

8.3 Filling in high variance regions

One could devise various schemes to incorporate the variance information into the MRF approach in particular by varying α and γ spatially according to the variance. This, however is not straightforward and leads to a significant complication of the algorithm. Instead, we found a very simple procedure to be useful in solving the problem of filling in information.

Shown in figure 22 are the variance (not in logarithmic scale) corresponding to the updated depth map of figure 19 and a histogram of these variances. We observe (this observation holds for all the variance maps we investigated) that the distribution is exponentially-shaped with the majority of the values concentrated at the lower end. In fact, over 90 %

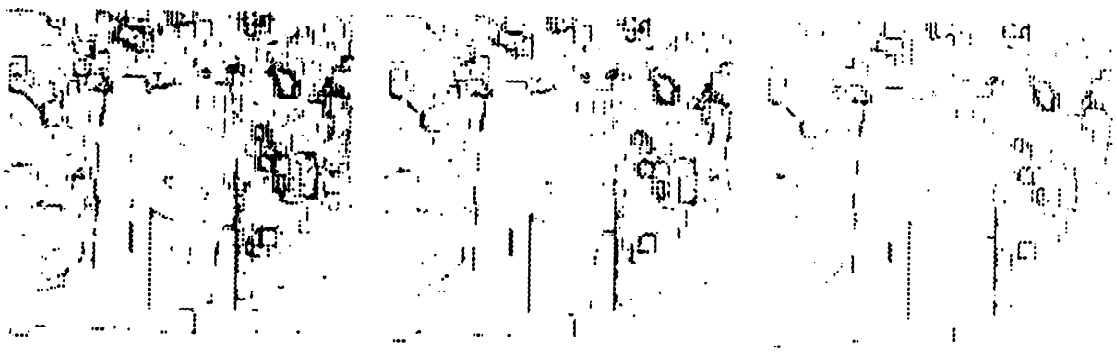

Figure 21: The line processes from the pepsi depth map with a) $\gamma = 160000$, b) $\gamma = 250000$ and c) $\gamma = 360000$ (see text).

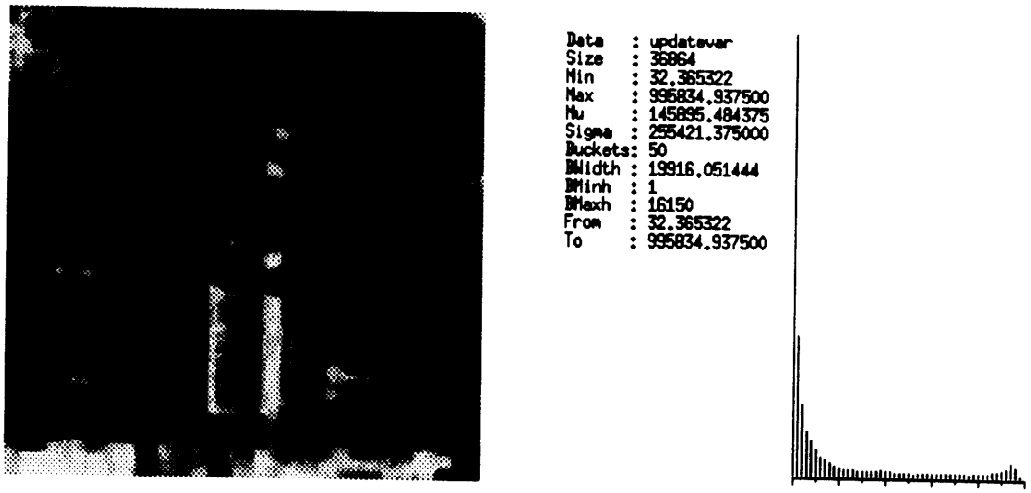

Figure 22: The variance map after update and a histogram of the variances.

of the values are between the min and min $+\sigma$ where σ is the standard deviation of the distribution and min is the smallest variance value observed. In other words: "bad" or high variance values can be clearly identified from the distribution.

Having determined such a threshold on the variances we fill in depth values that exhibit a variance value above the threshold by repeating the following "brushfire-fill" algorithm, until all variance values are below the threshold:

1. Initially, set $p'_{ij} = p_{ij}$ and $Z'_{ij} = Z_{ij}$ for all (i, j) where Z and p are the input depth and its variance.

2. For all sites (i, j) that have variance value p_{ij} above the threshold set the new variance

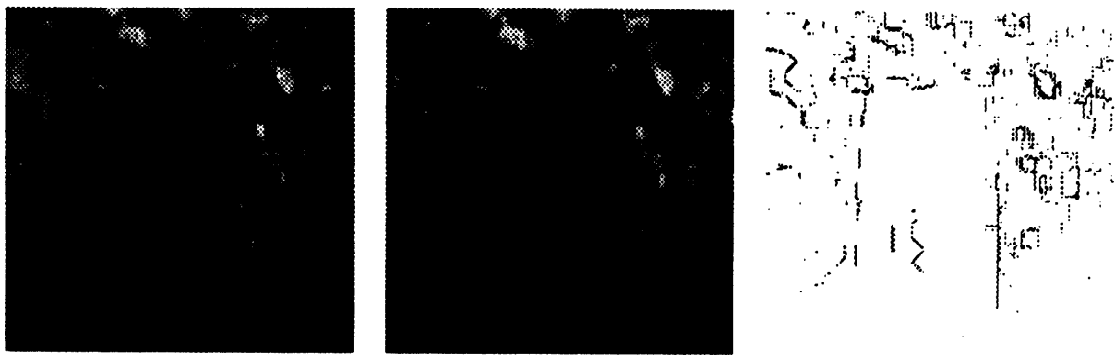

Figure 23: a) The result of filling in the updated depth map with threshold $\min + \sigma$, b) MRF smoothing of filled in depth map, c) line processes

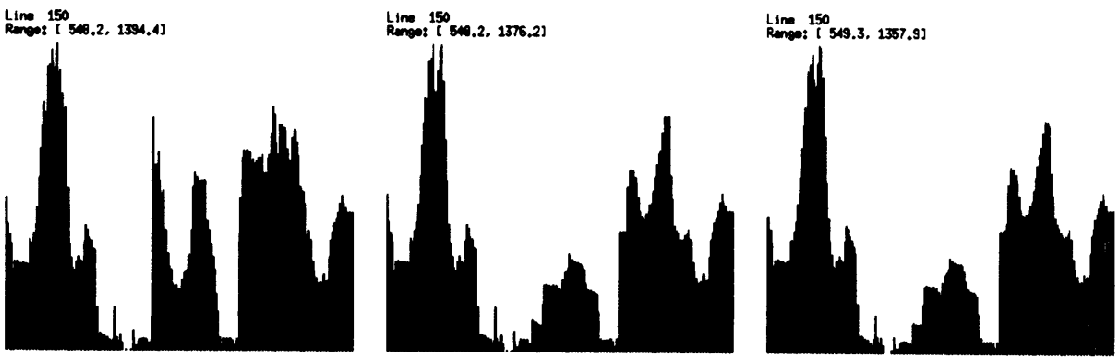

Figure 24: Line number 150 of a) the update depth map, b) the filled in depth map, c) the filled and MRF-smoothed depth map.

value p'_{ij} to be the average of the variances at the eight-connected neighbors that are below the threshold and the new depth Z'_{ij} to be the average of the depth values Z_{ij} at those sites.

3. Copy $p_{ij} = p'_{ij}$ and $Z_{ij} = Z'_{ij}$ for all (i, j). Continue with step 2.

The effect will be that in each step information will propagate by one pixel from low variance regions into high variance regions until all low variance information has been filled in. Figure 23 a) shows the result of filling in when a variance threshold of $\min + \sigma = 255000$ is used. The improvement over the input depth map 19 and all the MRF depth maps 20 is clearly visible. Figure 23 b) and c) show the result of processing the filled in depth map with the MRF smoother (Parameters $\alpha = 4$, $\beta = 10$, $\gamma = 250000$, $\omega = 0.01$).

A more detailed illustration of the effect of filling in and smoothing is provided by examining values along a cross-section of the depth map. Figure 24 a) shows the values

along line number 150 of the updated depth map of figure 19. This line is precisely 3/4 from the top of the depth map. Figure 24 b) shows the same line after the filling operation and figure 24 c) shows the line after MRF-smoothing the filled-in depth map. We see how the filling-in step significantly improves the depth estimate in the center of the line corresponding to the can. The MRF smoothing step then attenuates local variation while clearly preserving the major discontinuity.

This filling-in operation followed by MRF smoothing as described is available an optional "add-on" in our implementation of the dynamic structure and motion estimation algorithm. It is not a necessary component. As we have demonstrated, however, it is important to take care in performing filling or smoothing operations when the quality or "information content" varies among the field values under consideration.

8.4 Algorithmic aspects of the filling/smoothing stage

The filling stage requires at least one value in the input depth map to have variance above the threshold. In the worst case, this depth value could be located in a corner of the depth map in which case it would take $n-1$ iterations to fill the entire depth map. In each iteration n^2 sites are checked so that the worst-case complexity is $O(n^3)$. This scenario, however, is rather unrealistic. In practice the size of the largest high variance region can be bounded by some small number k and the complexity is $O(n^2k)$. On a parallel machine the filling operation can be accomplished in $O(log(k))$ time.

The complexity analysis of the MRF operation is quite intricate in general since convergence becomes an issue. If, however, we limit ourselves (as we have done) to a fixed number of iterations (say m) the complexity is easily seen to be $O(n^2m)$ on a serial machine and $O(m)$ on a parallel machine.

9 Experiments

In the previous sections we have seen the effect of the individual modules that constitute the dynamic motion vision algorithm. In this section we evaluate the complete estimation system on several real image sequences. The goal of this experimental evaluation is to determine, where possible, the amount of error in the depth and motion estimates produced by the algorithm and to investigate the amount of improvement that is obtained by this multiframe algorithm over a two-frame algorithm. Since true depth maps are rarely available for real scenes, the depth estimates must be assessed in a more qualitative manner by the reader.

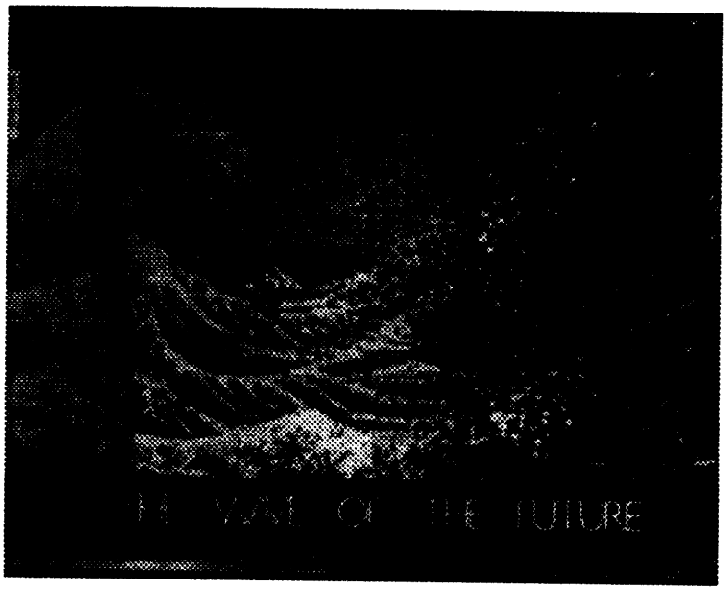

Figure 25: The wave experiment scene

9.1 The "wave" experiment

To obtain measurable results from both depth and motion estimation we used a simple scene with a planar surface parallel to the image plane. The scene is shown in figure 25. The camera translated by $\mathbf{t} = [1.5, 0, 3.0]$ mm between frames. Initial values of $\hat{Z}_0 = 1000$ and $\hat{p}_0 = 1000^2$ were used. Figures 26 a) and b) show 3D structure plots of the depth maps after the 1st and 10th filter iteration respectively. In both cases the depth range shown is $[767, 1499]$ which is the range of depth values after 1 iteration. No filling-in or MRF smoothing was used in order to investigate the effect of the filter alone. The noise reduction achieved by use of the filter is clearly visible.

Due to the simplicity of the experiment, we can quantify the effect of the filter. Since the scene is a frontal plane, the ideal depth map contains constant values everywhere. In practice, the depth values will deviate from the true constant value. The effect of the filter

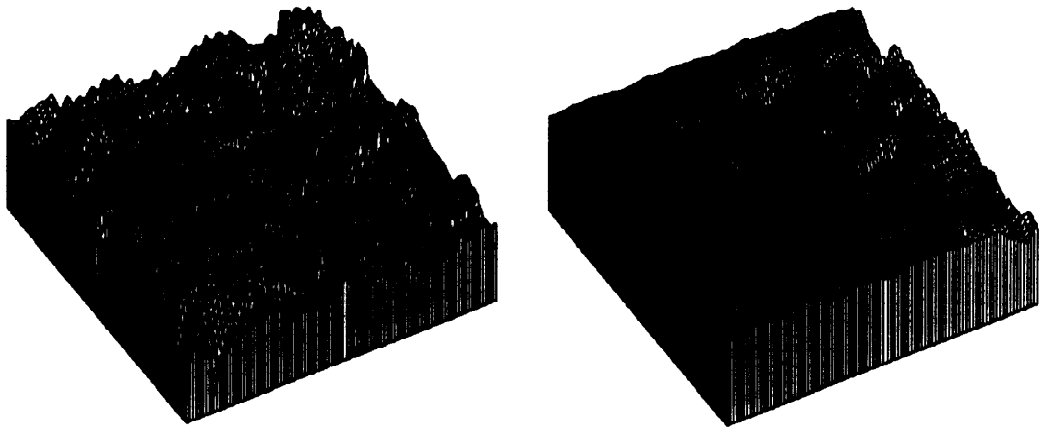

Figure 26: Structure plots of the depth maps from the wave scene after a) 1 iteration, b) 10 iterations of the filter. The depth range shown is [767, 1499] in both cases.

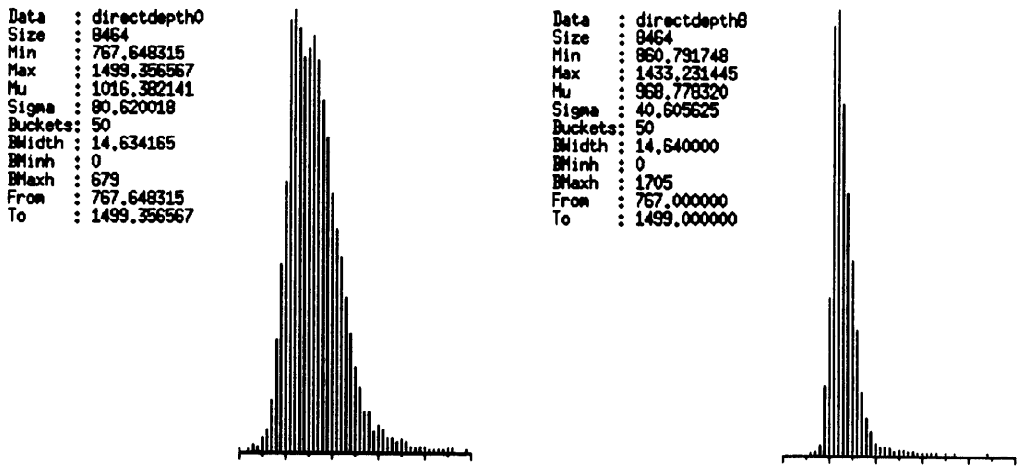

Figure 27: Statistical analysis of the depth maps from the wave scene after a) 1 iteration, b) 9 iterations of the filter.

should be to reduce the standard deviation of the distribution of depth values over time. Figure 27 a) and b) show the distributions of depth values after 1 and 9 iterations respectively, both shown in the range [767, 1499] for comparison. As we see the expected variance reduction has been achieved. More precisely, the variance is halved after 9 iterations.

To see the development over time we have plotted the variance of the values in the depth map over the frame number in figures 28. The filter performs satisfactorily and as

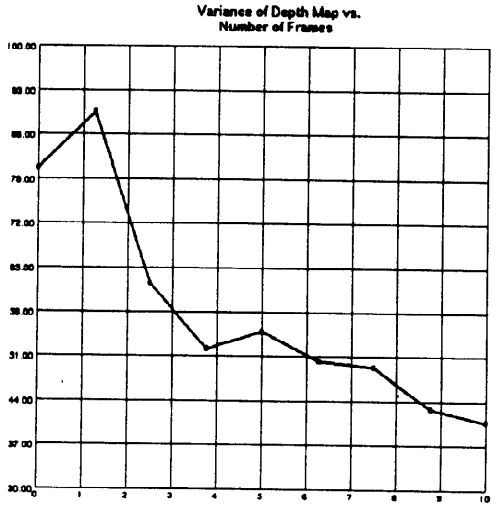

Figure 28: Development of variance in the depth map as a function of the frame number.

expected under all evaluation criteria. From this experiment, it is hard to evaluate the convergence or robustness of the proposed method, since the initial depth map happens to be the true depth map. The following experiment explores this issue.

9.2 The "pepsi" experiment

In this experiment, a soda can was placed on a table at 570 mm, before a background parallel to the image plane at 1240 mm as shown in figure 29. The camera translated $\mathbf{t} = [1.5, 0, 0]$ mm between frames. As we have pointed out in previous sections, this scene poses a number of difficulties since it contains large regions of uniform brightness as well as depth discontinuities. Initial values of $\hat{Z}_0 = 1000$ and $\hat{p}_0 = 1000^2$ were used.

Figures 30 a), b) and c) show the depth maps obtained after 1, 4, and 8 iterations of the algorithm. In these depth maps, brightness is proportional to depth (close values are dark, distant values are bright). MRF smoothing with 3 iterations, $\alpha = 4, \gamma = 250000$ and filling in of high-variance regions were applied as described. These depth maps show how the estimate improves over time, how the shape of the can becomes more distinct and the noise in the background is attenuated. Repeated application of the filling-in operation supplies regions of high variance with plausible values. Even the slanted surface of the the table on which the can stands is recovered.

Figures 31 a) and b) show the 3d structure plots of the depth maps after 1 and 8 iterations. Note that the bottom border of the depth map corresponds to the top left border in the structure plot so that the top of the can is visible to the observer. These plots show how the shape of the scene gradually emerges from the noise. The filled in regions can be made out in the final structure as they still differ slightly from the surrounding areas.

Figure 29: The pepsi scene

Figure 32 shows the components of the translation vector computed during the dynamic motion vision procedure for the pepsi scene. Since the true depth map is not available, it is not possible to scale the recovered translation vector to compare it with the actual motion. Instead, we scale the recovered vectors to be unit vectors and determine, how far the recovered motion deviates from pure x translation direction. We see that the error in all components is in the range of 2 - 3 % except in the first iteration. This is plausible, since the initial depth is flat and differs considerably from the true depth. This result for the motion estimation confirms the robustness of the motion estimation stage with respect to errors in depth.

Finally we use the pepsi experiment to compare the adaptive filtering approach to the case of known motion. If the true motion of $\mathbf{t} = [1.5, 0, 0]$ mm is known, the motion estimation stage can be bypassed and the filter is non-adaptive. For the same parameter settings as used to obtain the depth maps of figure 30 the filter produces the depth maps shown in figure 33 a), b) and c) after 1, 4 and 8 iterations respectively. They do not differ significantly from the results of the adaptive filter, in fact, the adaptive results seem to be "better".

9.3 The "cup" experiment

In this experiment a complex scene was constructed. A cup, a staple-remover and several books were placed on a table before a flat background. The layout of the scene is

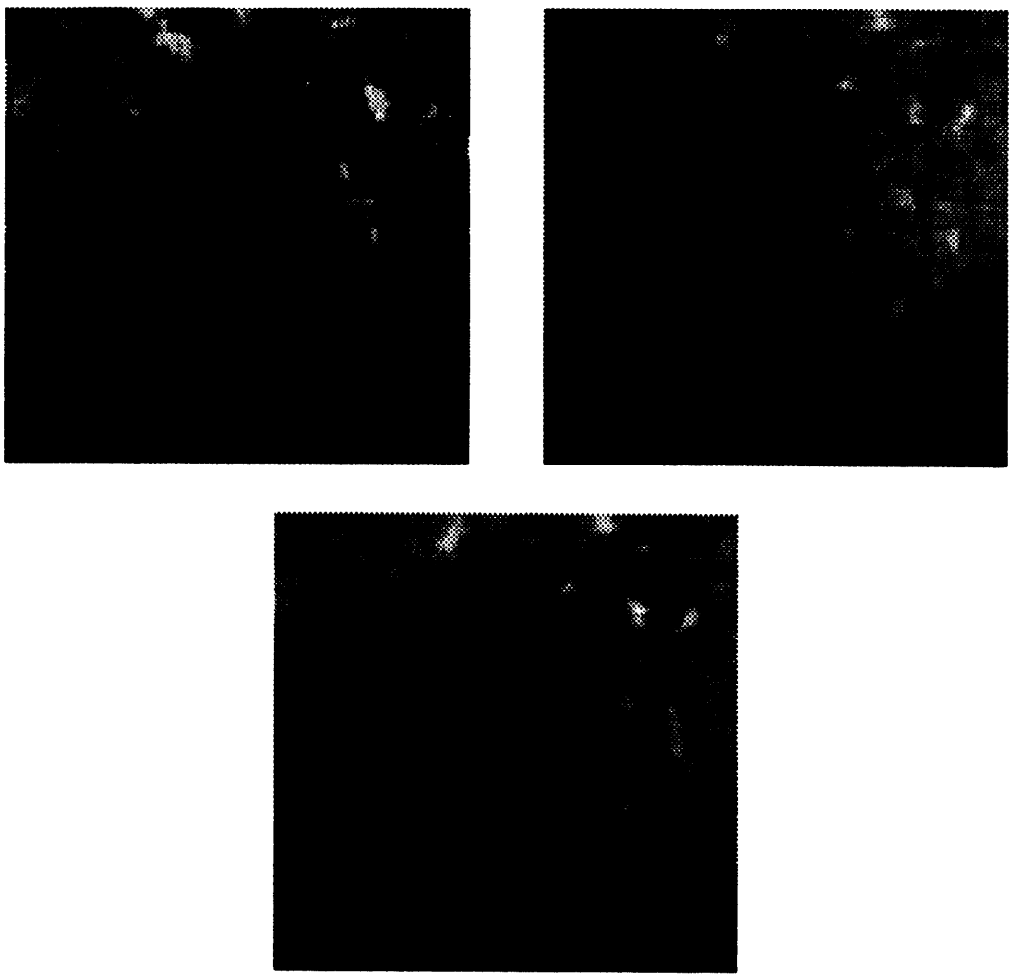

Figure 30: The depth maps from the pepsi scene after a) 1 iteration, b) 4 iterations, c) 8 iterations of the algorithm.

shown in figure 34 a) and figure 34 b) shows an image from the sequence taken by a camera translating $\mathbf{t} = [2, 0, 4]$ mm between frames.

Figures 35 a), b) and c) show the depth maps computed by the dynamic motion vision algorithm after 1, 4 and 8 iterations respectively. The same initial values $\hat{Z}_0 = 1000$ and $\hat{p}_0 = 1000^2$ as in the previous experiments were used. MRF smoothing with 2 iterations and parameters $\alpha = 4$, $\gamma = 40000$ (edge threshold magnitude 100) and filling-in were in effect.

Figure 36 a) shows the components of the recovered translation vector as a function of the frame number. As before the translation vector was normalized to be a unit vetor. The true translation vector in this diagram is therefore $[0.447, 0, 0.894]$.

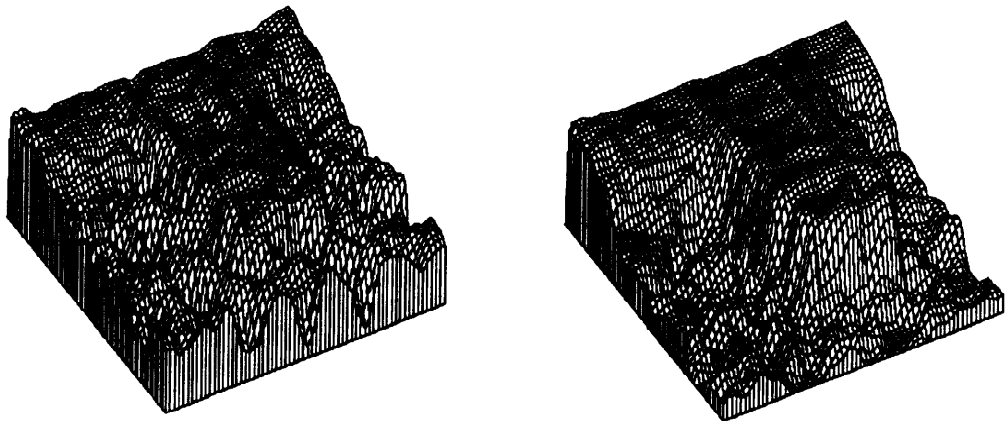

Figure 31: Structure plots for the pepsi scene after a) 1 iteration, b) 8 iterations.

Finally, figure 37 shows the line processes produced by the MRF smoother in the 9th iteration of the algorithm.

In evaluating the experimental results we find that the algorithm recovers the structure of all elements in the scene. Even the staple-remover in the left foreground is recovered, despite the fairly uniform brightness in that image region. The bottom region of the cup is not only uniform in brightness but also contains a specularity. The filling algorithm identifies this region of high variances and fills in plausible information. Now look at the top of the cup in the image. Barely visible is a white spoon which was placed in the cup. It appears clearly in the depth map. It seems to have widened. This can be explained by the fact that we are using patches of 9 pixels to recover depth, so that a certain growth of small regions across depth discontinuities should be expected. The results from this scene illustrate the practical applicability of the approach and rival results obtainable from stereo.

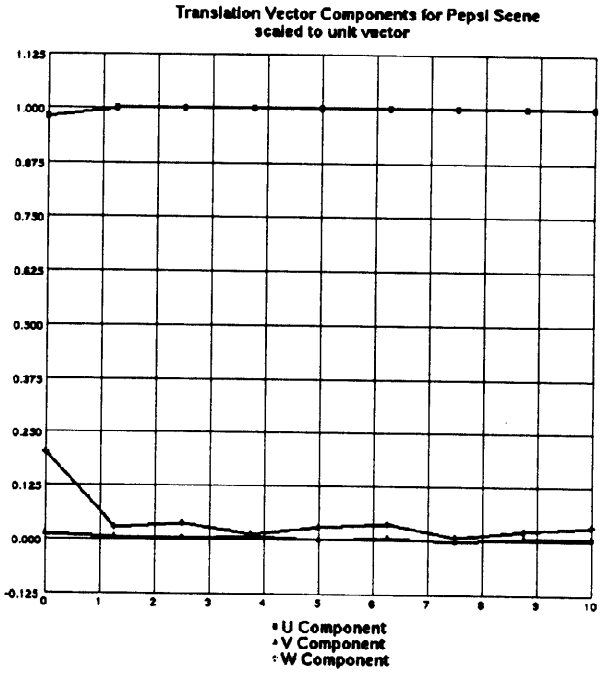

Figure 32: The components of the translation vector estimate. The translation vector has been scaled to be a unit vector.

10 Conclusion and Future Work

We have presented a multiframe algorithm for the dense estimation of structure and motion information. Key features of this direct dynamic motion vision method are

1. A dense structure or depth map is computed.
2. The parameters of rigid body motion are computed.
3. No restriction or assumption is made concerning the structure of the scene
4. No restriction or assumption is made conderning the motion other that the resulting motion in the image plane is limited to only a few pixels.
5. The algorithm operates iteratively over a sequence of arbitrary length and produces an updated depth estimate which best incorporates information from all preceding frames according to the quality.
6. The algorithm is based on the Kalman filter recursive estimation technique and becomes adaptive when motion estimation is included.
7. Using a filling and smoothing algorithm, plausible structure estimates can be obtained even in image regions where no information is available.

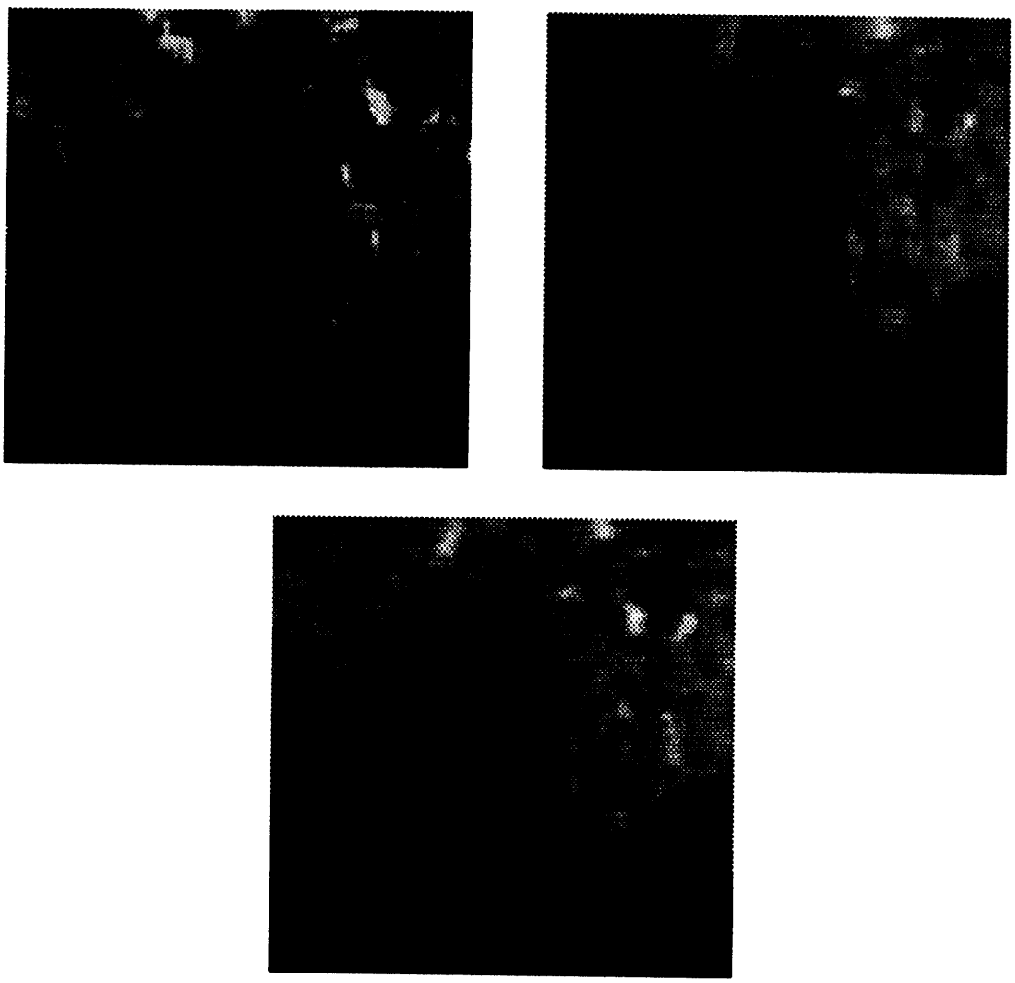

Figure 33: The depth maps from the pepsi scene after a) 1 iteration, b) 4 iterations, c) 8 iterations of the algorithm when motion is known.

8. No time-consuming computation of optical flow or feature matches is required. The algorithm has great potential for real-time implementation as all of the operations involved are local.

9. The algorithm provides robust results for both structure and motion as shown on a variety real images.

The key assumption we have made is the brightness change constraint assumption (7). We have in fact shown, that its validity at individual pixels is not guaranteed at all, but that reasonable information may be obtained from this assumption when image patches are considered. In combination with the recursive estimation procedure for error attentuation

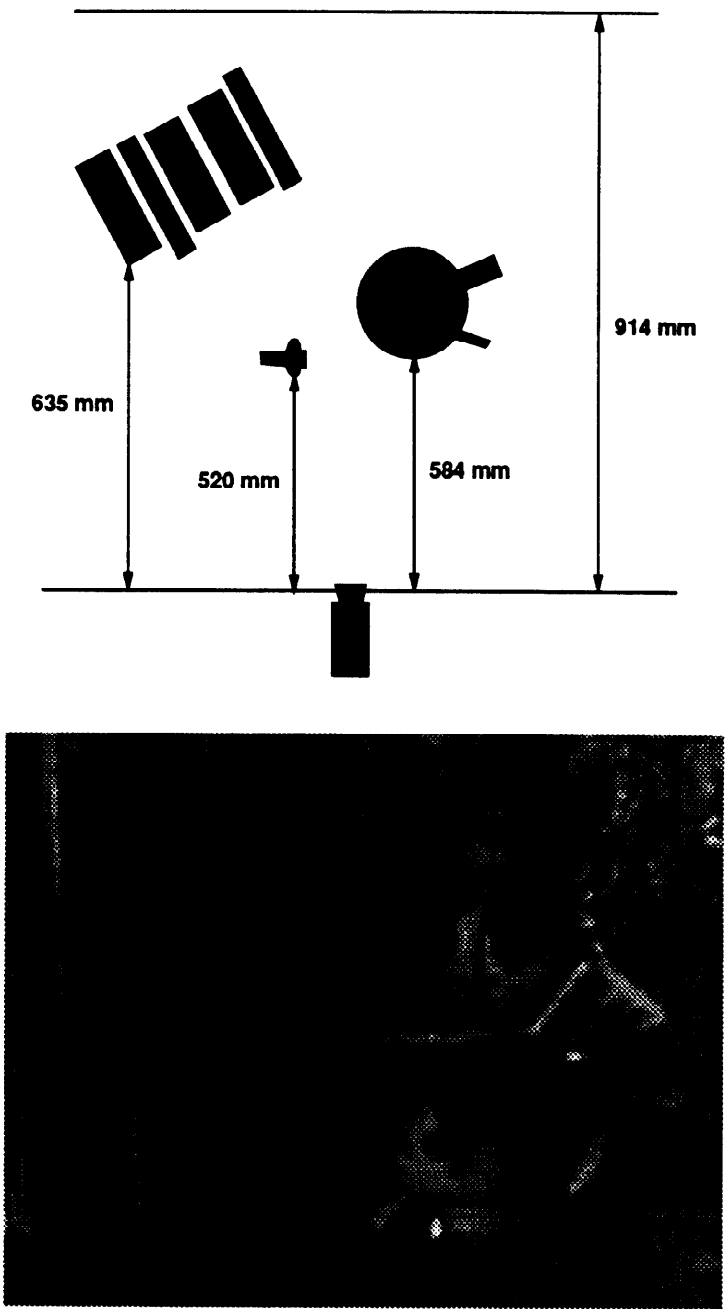

Figure 34: a) The layout of the cup experiment and b) an image from the cup sequence.

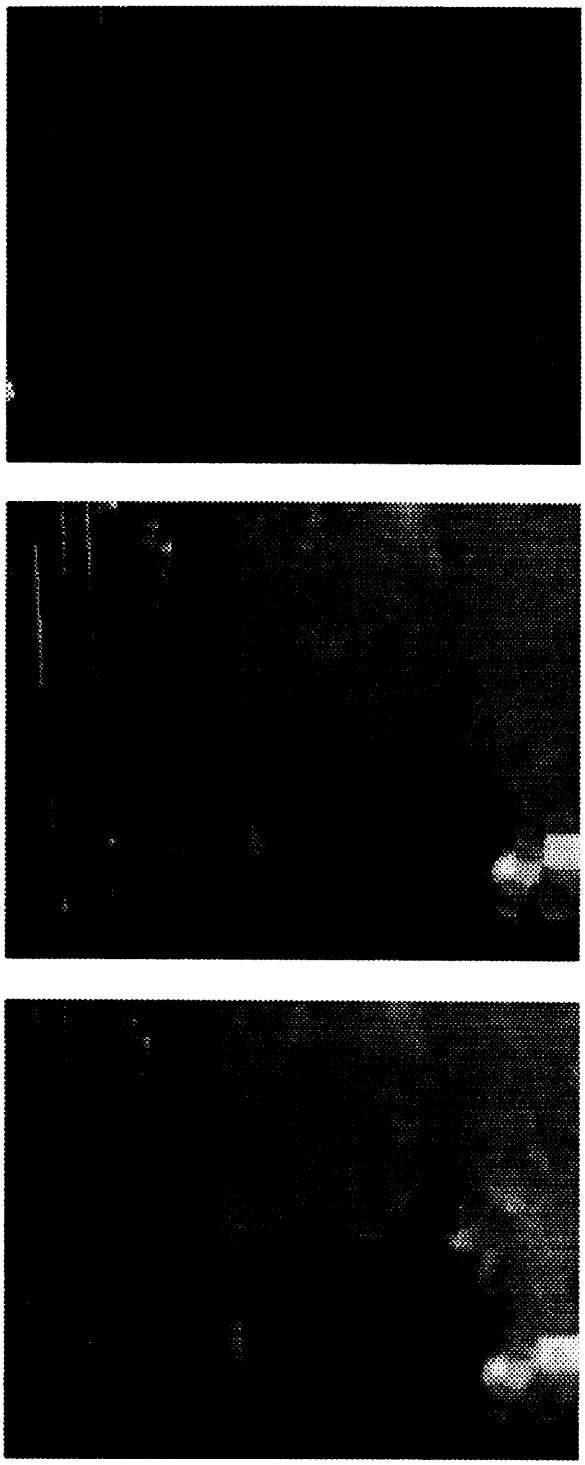

Figure 35: a) The depth maps computed by the algorithm for the cup scene after a) 1 iteration, b) 4 iterations, c) 9 iterations.

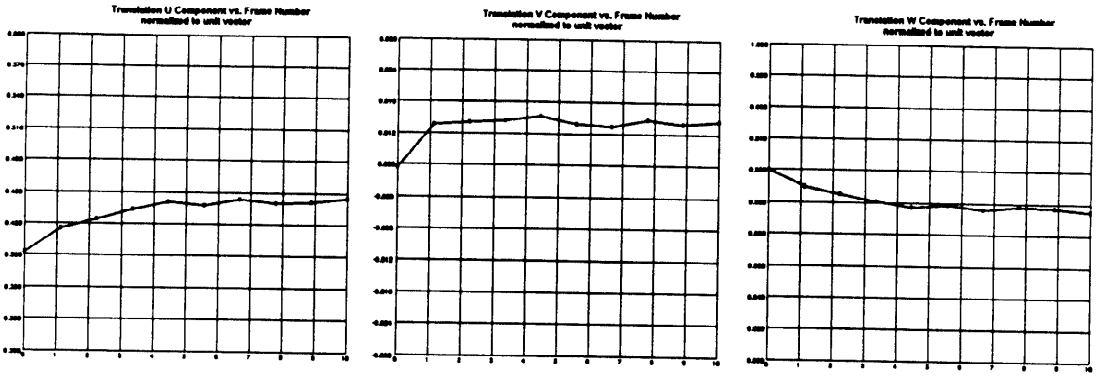

Figure 36: U, V and W components of the recovered translation vectors as functions of the frame number. Translation vectors have been normalized to unit vectors. True unit motion is [0.447, 0, 0.894].

Figure 37: The line processes produced by the MRF smoother in the last iteration of the dynamic structure estimation algorithm.

over multiple frames the experimental results seem to confirm the validity of this assumption in the given application.

A limitation is the fact that only one independently moving rigid body may be visible in the scene. To overcome this, it would be necessary to segment the image into regions corresponding to independently moving objects and then run the algorithm on each region individually. An idea that we are currently pursuing is to integrate segmentation into the

dynamic estimation process. The line processes available from the MRF smoothers are a first step in this direction, but more work is necessary.

An alternative to the proposed iterative method is a global or batch approach which collects information from the entire sequence before estimating motion and structure. As we have mentioned, such methods should exhibit higher accuracy and should be evaluated in comparison with the presented approach.

The most interesting aspect for future work lies in the following observation. The depth update procedure which incorporates information from a new measurement reads (eq. 46)

$$\hat{Z}_k \leftarrow \frac{Z_k/p_k + \hat{Z}_k/\hat{p}_k}{1/p_k + 1/\hat{p}_k} \tag{97}$$

Suppose measurements of depth are available from several independent sources such as stereo, shape from shading, motion or even non-visual sensors. If each such measurement Z_i is accompanied by an estimate of its quality (variance) p_i, then they can all be incorporated into the current depth estimate in a straightforward way

$$\hat{Z}_k \leftarrow \frac{\hat{Z}_k/\hat{p}_k + \sum_{i=1}^{n} Z_{ik}/p_{ik}}{1/\hat{p}_k + \sum_{i=1}^{n} 1/p_{ik}} \tag{98}$$

This is the basis for a theory of temporal integration which we intend to investigate by combining depth information from various visual sources over time which has great potential for future work.

Acknowledgements

I would like to thank Berthold Horn and Shahriar Negahdaripour for their support and insights which helped make this research happen.

A Appendix: Warping the variance values in the prediction stage

It remains to determine the variance values of the warped depth map. In essence, the warped values of Z are some function of the input values of Z i.e. the output value is a random variable which is some function of several input random variables. More formally: we interpret the given values of $Z(x_j, y_i)$ as normally distributed random variables with variance $p(x_j, y_i)$. What are the value of $p(x_j, y_i)$ after the warping?

A.1 Variance Propagation

Let us first establish some basic facts about propagation of variances. Let Z_1, Z_2 be two uncorrelated random variables with variances $\sigma_{Z1}^2, \sigma_{Z2}^2$. Then the random variable

$$Z = aZ_1 + bZ_2 \tag{99}$$

has the variance

$$\sigma_Z^2 = a^2\sigma_{Z1}^2 + b^2\sigma_{Z2}^2 \tag{100}$$

The only assumption made here is that Z_1, Z_2 are uncorrelated.

In the more general case Z can be an arbitrary function of several random variables, say

$$Z = f(Z_1, Z_2). \tag{101}$$

In this case we approximate f by its Taylor series around the point of interest and neglect all but the first-order terms. Then we apply the above rule for a linear combination of random variables to obtain

$$\sigma_Z^2 = (\frac{\partial f}{\partial Z_1})^2\sigma_{Z1}^2 + (\frac{\partial f}{\partial Z_2})^2\sigma_{Z2}^2 \tag{102}$$

where the dertivatives must be evaluated at the particular point (Z_1, Z_2) of interest. This relationship is easily extended to n independent variables. The assumptions made here are zero correlation between Z_1 and Z_2 and the fact that $f()$ can be approximated by the first terms of its Taylor series near Z_1, Z_2.

A.2 Variances of the warped depth values

Having established how variances propagate through functions we need merely determine the functional relationship between input and output depth values and propagate the variances through these functions. Let us determine where in our algorithm we actually compute an output value of Z and how it is computed.

There are two ways in which an output value of Z can be computed. The first is in step 6 of the algorithm. There we compute the intersection of the ray through a pixel location (x, y) with a spatial triangle. The spatial triangle is determined by the 3D coordinates of the corner points (X_i', Y_i', Z_i') for $i = 1, 2, 3$. So our output value of Z is some function (which is determined by the interpolation procedure) of the corner coordinate values. Each corner coordinate is the result of warping one point (X_i, Y_i, Z_i) from the input surface. The

warping function is simply a linear combination of the input point coordinates. Finally we note that the X and Y components of each original point are obtained by inverse perspective projection

$$X_i = \frac{x_i Z_i}{f} \quad \text{and} \quad Y_i = \frac{y_i Z_i}{f}. \tag{103}$$

Thus, each Z value obtained in step 6 of the algorithm is a function of three depth values Z_1, Z_2, Z_3 in the input depth map. We determine this functional relationship below.

The second possibility is when the depth value is obtained by extrapolating in step 7 of the algorithm. In this case a depth value Z is computed as the average of some depth values $Z_1, \ldots Z_k$ in its immediate neighborhood. This is a fairly simple linear relationship. The variance propagation for this case is also discussed in detail below.

A.2.1 Variances of interpolated depth values

We determined above that each output depth value Z is a function of three input depth values Z_1, Z_2, Z_3. Let us begin by determining the relationship between these values and the corresponding values (X_1', Y_1', Z_1'), (X_2', Y_2', Z_2'), (X_3', Y_3', Z_3') after warping. We combine the equations of motion

$$\begin{bmatrix} X_i' \\ Y_i' \\ Z_i' \end{bmatrix} = - \begin{bmatrix} U \\ V \\ W \end{bmatrix} - \begin{bmatrix} 1 & -C & B \\ C & 1 & -A \\ -B & A & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \tag{104}$$

with the inverse perspective projection

$$X_i = \frac{x_i Z_i}{f} \tag{105}$$

$$Y_i = \frac{y_i Z_i}{f} \tag{106}$$

to obtain

$$X_i' = -U + b_{xi} Z_i \tag{107}$$

$$Y_i' = -V + b_{yi} Z_i \tag{108}$$

$$Z_i' = -W + b_{zi} Z_i \tag{109}$$

where

$$b_{xi} = x/f + Cy/f - B \tag{110}$$

$$b_{yi} = Cx/f + y/f + A \tag{111}$$

$$b_{zi} = Bx/f - Ay/f + 1. \tag{112}$$

After the motion warping, we compute the output depth by interpolation. As we recall, there are three cases by which Z can be obtained within the interpolation procedure.

1. There is exactly one point of intersection between the ray and the spatial triangle.

2. The ray lies in the same plane as the spatial triangle and it has at most one point in common with each edge of the spatial triangle.

3. The ray lies in the same plane as the spatial triangle and it coincides with one edge of the spatial triangle.

In the first case, the output Z value is computed as

$$Z = \lambda f = \frac{D_\lambda}{D} f \tag{113}$$

where

$$\begin{aligned} D &= (X_1 - X_3)[(Z_2 - Z_3)y - (Y_2 - Y_3)f] - \\ &\quad (Y_1 - Y_3)[(Z_2 - Z_3)x - (X_2 - X_3)f] + \\ &\quad (Z_1 - Z_3)[(Y_2 - Y_3)x - (X_2 - X_3)y] \end{aligned} \tag{114}$$

and

$$\begin{aligned} D_\lambda &= -X_3[(Y_1 - Y_3)(Z_2 - Z_3) - (Y_2 - Y_3)(Z_1 - Z_3)] \\ &\quad +Y_3[(X_1 - X_3)(Z_2 - Z_3) - (X_2 - X_3)(Z_1 - Z_3)] \\ &\quad -Z_3[(X_1 - X_3)(Y_2 - Y_3) - (X_2 - X_3)(Y_1 - Y_3)] \end{aligned} \tag{115}$$

where the primes have been omitted. If we use the above equations (107) - (109) to replace the variables in these expressions, we find

$$D = aZ_1Z_2 + bZ_1Z_3 + cZ_2Z_3 \tag{116}$$

where

$$a = b_{x1}(b_{z2}y - b_{y2}f) - b_{y1}(b_{z2}x - b_{x2}f) + b_{z1}(b_{y2}x - b_{x2}y) \tag{117}$$

$$b = b_{x1}(b_{y3}f - b_{z3}y) - b_{y1}(b_{x3}f - b_{z3}x) + b_{z1}(b_{x3}y - b_{y3}x) \tag{118}$$

$$c = -b_{x3}(b_{z2}y - b_{y2}f) + b_{y3}(b_{z2}x - b_{x2}f) - b_{z3}(b_{y2}x - b_{x2}y). \tag{119}$$

Further we have

$$D_\lambda = aZ_1Z_2 + bZ_1Z_3 + cZ_2Z_3 + dZ_1Z_2Z_3 + eZ_1Z_3^2 + fZ_2Z_3^2 \tag{120}$$

where

$$a = Ua_1 - Va_2 + Wa_3 \tag{121}$$

$$b = Ub_1 - Vb_2 + Wb_3 \tag{122}$$

$$c = Uc_1 - Vc_2 + Wc_3 \tag{123}$$

$$d = -b_{x3}a_1 + b_{y3}a_2 - b_{z3}a_3 \tag{124}$$

$$e = -b_{x3}b_1 + b_{y3}b_2 - b_{z3}b_3 \tag{125}$$

$$f = -b_{x3}c_1 + b_{y3}c_2 - b_{z3}c_3 \tag{126}$$

in which we have abbreviated

$$a_1 = b_{y1}b_{z2} - b_{y2}b_{z1} \quad (127)$$
$$b_1 = b_{z1}b_{y3} - b_{y1}b_{z3} \quad (128)$$
$$c_1 = b_{y2}b_{z3} - b_{y3}b_{z2} \quad (129)$$
$$(130)$$
$$a_2 = b_{x1}b_{z2} - b_{x2}b_{z1}$$
$$b_2 = b_{x3}b_{z1} - b_{x1}b_{z3} \quad (131)$$
$$c_2 = b_{x2}b_{z3} - b_{x3}b_{z2} \quad (132)$$
$$(133)$$
$$a_3 = b_{x1}b_{y2} - b_{x2}b_{y1}$$
$$b_3 = b_{x3}b_{y1} - b_{x1}b_{y3} \quad (134)$$
$$c_3 = b_{x2}b_{y3} - b_{y2}b_{x3}. \quad (135)$$

As a result of these tedious manipulations we are now able to express an interpolated value of Z as a function of three input depth values:

$$Z = Z(Z_1, Z_2, Z_3) = f\frac{D_\lambda(Z_1, Z_2, Z_3)}{D(Z_1, Z_2, Z_3)} \quad (136)$$

If the variances for the input depth values are p_1, p_2, p_3, we assume that they are uncorrelated and the above functional relationship can be approximated by the first terms of the Taylor series we can use the variance propagation (102)

$$p = (\frac{\partial Z}{\partial Z_1})^2 p_1 + (\frac{\partial Z}{\partial Z_2})^2 p_2 + (\frac{\partial Z}{\partial Z_3})^2 p_3 \quad (137)$$

to determine the output variance p. It remains to determine the partial derivatives

$$\frac{\partial Z}{\partial Z_i} = f\frac{\partial}{\partial Z_i}\frac{D_\lambda}{D} = \frac{\frac{\partial D_\lambda}{\partial Z_i}D - D_\lambda\frac{\partial D}{\partial Z_i}}{D^2}. \quad (138)$$

The partial derivatives of D are obtained easily from (116)

$$\frac{\partial D}{\partial Z_1} = aZ_2 + bZ_3 \quad (139)$$
$$\frac{\partial D}{\partial Z_2} = aZ_1 + cZ_3 \quad (140)$$
$$\frac{\partial D}{\partial Z_3} = bZ_1 + cZ_2 \quad (141)$$

and similarly those for D_λ from (120)

$$\frac{\partial D_\lambda}{\partial Z_1} = aZ_2 + bZ_3 + dZ_2Z_3 + eZ_3^2 \quad (142)$$
$$\frac{\partial D_\lambda}{\partial Z_2} = aZ_1 + cZ_3 + dZ_1Z_3 + fZ_3^2 \quad (143)$$
$$\frac{\partial D_\lambda}{\partial Z_3} = bZ_1 + cZ_2 + dZ_1Z_2 + 2eZ_1Z_3 + 2fZ_2Z_3. \quad (144)$$

Note that a, b and c are computed differently for D and D_λ. We must also take care not to confuse the coefficient f used in the expression for D_λ with the focal length f. This result enables us to compute the variance p of every output depth value Z obtained by interpolation in the case where the ray has exactly one point in common with the spatial triangle.

In the second case listed above we consider how a depth value is obtained if the interpolation ray is parallel to the spatial triangle but does not coincide with one of its edge segments. In this case we recall that Z is obtained by interpolating between the two endpoints (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2) of the triangle edge segment:

$$Z = f\frac{d_\lambda}{d} \tag{145}$$

with

$$d = x(Y_1 - Y_2) - y(X_1 - X_2) \tag{146}$$

and

$$d_\lambda = (X_1 - X_2)Y_2 - (Y_1 - Y_2)X_2. \tag{147}$$

All coordinate components are after the motion warping although the primes have been omitted. We express both d and d_λ in terms of the depth values of the endpoints Z_1 and Z_2 before motion warping as done before:

$$d = aZ_1 + bZ_2 \tag{148}$$

where

$$a = xb_{y1} - yb_{x1} \tag{149}$$
$$b = yb_{x2} - xb_{y2} \tag{150}$$

and similarly

$$d_\lambda = aZ_1 + bZ_2 + cZ_1Z_2 \tag{151}$$

in which

$$a = b_{y1}U - b_{x1}V \tag{152}$$
$$b = b_{x2}V - b_{y2}U \tag{153}$$
$$c = b_{x1}b_{y2} - b_{y1}b_{x2}. \tag{154}$$

We compute the partial derivatives

$$\frac{\partial d}{\partial Z_1} = a \tag{155}$$
$$\frac{\partial d}{\partial Z_2} = b \tag{156}$$

and

$$\frac{\partial d_\lambda}{\partial Z_1} = a + cZ_2 \tag{157}$$

$$\frac{\partial d_\lambda}{\partial Z_2} = b + cZ_1 \tag{158}$$

which we need for

$$\frac{\partial Z}{\partial Z_1} = \frac{\frac{\partial d_\lambda}{\partial Z_1} d - d_\lambda \frac{\partial d}{\partial Z_1}}{d^2} \tag{159}$$

$$\frac{\partial Z}{\partial Z_2} = \frac{\frac{\partial d_\lambda}{\partial Z_2} d - d_\lambda \frac{\partial d}{\partial Z_2}}{d^2}. \tag{160}$$

Then the variance p of the output Z becomes

$$p = (\frac{\partial Z}{\partial Z_1})^2 p_1 + (\frac{\partial Z}{\partial Z_2})^2 p_2. \tag{161}$$

In the third and last case the interpolation ray coincides with at least one edge segment of the spatial triangle. Suppose the depth values of the endpoints are Z_1, Z_2 before motion warping. After motion warping they become

$$Z_1' = -W + b_{z1} Z_1 \qquad \text{and} \qquad Z_2' = -W + b_{z2} Z_2 \tag{162}$$

Recall that the algorithm assigned the smaller of the two values Z_1', Z_2' as the output value Z. Hence, if $Z_1' < Z_2'$ then the output variance would be

$$p = b_{z1}^2 p_1 \tag{163}$$

otherwise

$$p = b_{z2}^2 p_2. \tag{164}$$

From the point of view of implementation we see that the computation of variances cannot simply be added to the depth computation outlined in the previous section. A completely different approach is necessary in order to have the b_{xi}, b_{yi}, b_{zi} values available in the interpolation stage. Although the depth computation is identical the abstraction from the underlying geometry makes it less intuitive.

A.2.2 Variances of extrapolated depth values

A number of output depth values are obtained through extrapolation. As described above we use a very simple extrapolation procedure. In which an unassigned depth value is set to the average of its assigned neighbors. This extrapolation was motivated by assuming certain smoothness of the surface. Nevertheless, the choice of variance at this location should indicate the fact, that no information was available there so that subsequent updates can contributed maximally to the value at this point. This is particularly important when objects enter the field of view that exhibit a depth discontinuity with respect to the previously visible objects. We therefore assign ∞ (some large value) as the variance of extrapolated points.

References

[1] G. Adiv. Determing 3-d motion and structure from optical flow generated by several moving objects. COINS Technical Report 84-07, University of Massachusetts, Amherst, April 1984.

[2] J. Aloimonos and J.-Y. Herve. Correspondenceless detection of depth and motion for a planar surface. Technical Report CAR-TR-357, Computer Vision Laboratory, Center for Automation Research, University of Maryland, April 1988.

[3] P. Anandan. Computing dense displacement fields with confidence measures in scenes containing occlusion. COINS Technical Report 84-32, University of Massachusetts, Amherst, December 1984.

[4] H. H. Baker and R. C. Bolles. Generalizing epipolar-plane image analysis on the spatiotemporal surface. *International Journal of Computer Vision*, 3, 1989.

[5] S. Bharwani, E. Riseman, and A. Hanson. Refinement of environment depth maps over multiple frames. In *Proceedings of the Workshop on Motion: Representation and Analysis*, Charleston, S. C., May 1986.

[6] A. Blake and A. Zisserman. *Visual Reconstruction.* MIT Press, 1987.

[7] R. C. Bolles and H. H. Baker. Epipolar-plane image analysis: A technique for analyzing motion sequences. In *IEEE Proceedings of the Third Workshop on Computer Vision: Representation and Control*, Bellaire, MI, October 1985.

[8] T. J. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(1), January 1986.

[9] T. J. Broida and R. Chellappa. Kinematics and structure of a rigid object from a sequence of noisy images. In *Proceedings of the IEEE Workshop on Motion: Representation and Analysis*, Charleston, SC, May 1986.

[10] A. R. Bruss and B. K. P. Horn. Passive navigation. *Computer Vision, Graphics, and Image Processing*, 21, 1983.

[11] E. D. Dickmans. 4d-szenenanalyse mit integralen raum-/zeitlichen modellen. In *Mustererkennung 1987, 9. DAGM Symposium*, Braunschweig, W. Germany, September/October 1987.

[12] E. D. Dickmans. Subject-object discrimination in 4d-dynamic scene interpretation for machine vision. In *Proceedings Workshop on Visual Motion*, Irvine, CA, March 1989.

[13] C. L. Fennema and W. B. Thompson. Velocity determination in scenes containing several moving objects. *Computer Graphics and Image Processing*, 9(4), April 1979.

[14] O. Foellinger. *Regelungstechnik.* Dr. Alfred Huethig Verlag, Heidelberg, 1984.

[15] G. F. Franklin and J. D. Powell. *Digital Control of Dynamic Systems.* Addison-Wesley, Reading, MA, 1980.

[16] D. Geiger and F. Girosi. Parallel and deterministic algorithms for mrfs: surface reconstruction and integration. AI Memo 1114, MIT Artificial Intelligence Laboratory, June 1989.

[17] A. Gelb (Ed.). *Applied Optimal Estimation.* The MIT Press, Cambridge, MA, 1974.

[18] S. Geman and D. Geman. Stochastic realxation. gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6, 1984.

[19] W. E. L. Grimson. *From Images to Surfaces.* The MIT Press, 1981.

[20] D. J. Heeger. Optical flow using spatiotemporal filters. *International Journal of Computer Vision*, 1(4), 1987.

[21] J. Heel. Dynamical systems and motion vision. AI Memo 1037, MIT Artificial Intelligence Laboratory, April 1988.

[22] J. Heel. Dynamic motion vision. In *Proceedings of the DARPA Image Understanding Workshop*, Palo Alto, CA, May 1989.

[23] J. Heel. Dynamic motion vision. In *Proceedings SPIE Conference on Advances in Intelligent Robotics Systems*, Philadelphia, PA, November 1989.

[24] E. C. Hildreth. Computations underlying the measurement of visual motion. *Artificial Intelligence*, 23, 1984.

[25] E. C. Hildreth. *The Measurement of Visual Motion.* The MIT Press, 1984.

[26] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17, 1981.

[27] B. K. P. Horn and E. J. Weldon, Jr. Computationally-efficient methods for recovering translational motion. In *Proceedings of the IEEE First International Conference on Computer Vision*, London, England, June 1987.

[28] B. K. P. Horn and E. J. Weldon, Jr. Direct methods for recovering motion. *International Journal of Computer Vision*, 2, 1988.

[29] E. Ito and J. Aloimonos. Is correspondence necessary for the perception of structure from motion? Technical Report CAR-TR-340, Computer Vision Laboratory, Center for Automation Research, University of Maryland, January 1988.

[30] K. Kanatani. Structure from motion without correspondence: General principle. In *Proceedings Image Understanding Workshop*, Miami, FL, December 1985.

[31] H. C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. In S. Ullman and W. Richards, editors, *Image Understanding 1984*. Ablex, 1984.

[32] D. G. Luenberger. *Introduction to Dynamic Systems*. John Wiley and Sons, Inc., 1979.

[33] J. L. Marroquin. Deterministic bayesian estimation of markovian random fields with applications to computational vision. In *Proceedings of the International Conference on Computer Vision*, London, England, June 1987.

[34] L. Matthies, R. Szeliski, and R. Kanade. Incremental estimation of dense depth maps from image sequences. In *Proceedings Computer Vision and Pattern Recognition*, Ann Arbor, MI, June 1988.

[35] L. Matthies, R. Szeliski, and R. Kanade. Kalman filter-based algorithms for estimating depth from image sequences. In *Proceedings of the DARPA Image Understanding Workshop*, Cambridge, MA, April 1988.

[36] A. Mitiche. On kineopsis and computation of structure and motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8, 1986.

[37] H.-H. Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(5), September 1986.

[38] S. Negahdaripour and B. K. P. Horn. Direct passive navigation: Analytical solution for planes. In *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 1986.

[39] S. Negahdaripour and B. K. P. Horn. A direct method for locating the focus of expansion. AI Memo 939, MIT Artificial Intelligence Laboratory, January 1987.

[40] S. Negahdaripour and B. K. P. Horn. Direct passive navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9, January 1987.

[41] S. Negahdaripour and B. K. P. Horn. Using depth-is-positive constraint to recover translational motion. In *Proceedings of the Workshop on Computer Vision*, Miami Beach, FL, November 1987.

[42] S. Negahdaripour and B. K. P. Horn. A direct method for locating the focus of expansion. *Computer Vision, Graphics and Image Processing*, Vol. 46(3), June 1989.

[43] J. W. Roach and J. K. Aggarwal. Determining the movement of objects from sequences of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(6), November 1980.

[44] J.-P. Schott. *Three-Dimensional Motion Estimation Using Shading Information in Multiple Frames*. PhD thesis, Massachusetts Institute of Technology, 1985.

[45] M. Spetsakis and J. Aloimonos. Optimal computing of structure from motion using point correspondences in two frames. Technical Report CAR-TR-389, Computer Vision Laboratory, Center for Automation Research, University of Maryland, September 1988.

[46] J. Stuller and G. Krishnamurthy. Kalman filter formulation of low-level television motion estimation. *Computer Vision, Graphics and Image Processing*, 21(2), February 1983.

[47] M. Subbarao. *Interpretation of Visual Motion: A Computational Study.* Morgan Kaufmann Publishers, Inc., 1988. Research Notes in Artificial Intelligence.

[48] D. Terzopoulos. Multilevel computational processes for visual surface reconstruction. *Computer Vision, Graphics and Image Processing*, 24, 1983.

[49] R. Y. Tsai and S. T. Huang. Estimating three-dimensional motion parameters of a rigid planar patch. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-29(6), December 1981.

[50] S. Ullman. Maximizing rigidity: The incremental recovery of 3-d structure from rigid and rubbery motion. AI Memo 721, MIT Artificial Intelligence Laboratory, June 1983.

[51] A. Verri and T. Poggio. Regularization theory and shape constraints. AI Memo 916, MIT Artificial Intelligence Laboratory, September 1986.

[52] W. T. Vetterling, S. A. Teukolsky, W. H. Press, and B. P. Flannery. *Numerical Recipes.* Cambridge University Press, 1986.

[53] A. M. Waxman and K. Wohn. Contour evolution, neighborhood deformation, and global image flow: Planar surfaces in motion. *International Journal of Robotics Research*, 4(3), 1985.

[54] J. Weng, T. S. Huang, and N. Ahuja. 3-d motion estimation, understanding, and prediction from noisy image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(3), May 1987.

[55] A. S. Willsky. *Digital signal processing and control and estimation theory.* The MIT Press, Cambridge, MA, 1979.

[56] M. Yamamoto. The image sequence analysis of three-dimensional dynamic scenes. Technical Report UDC 681.3.056, Electrotechnical Laboratory, Agency of Industrial Science and Technology, Japan, May 1988.