A Model for Shape and Motion Perception

Tarek M. Sobh

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania, Philadelphia, PA 19104

Abstract

In this paper we describe an efficient system for recovering the 3-D motion and structure of visual systems from an evolving image sequence. The technique utilizes the image flow velocities in order to recover the 3-D parameters. We develop a real-time algorithm for recovering the 2-D flow vectors on the image plane with sub-pixel accuracy. The flow estimates are then examined for the possibilities for errors, mistakes and uncertainties in the visual system. Uncertainty models are developed for the sensor and for the image processing techniques being used. Further filtering and a rejection mechanism are then developed to discard unrealistic flow estimates. The 2-D flow models are then converted into 3-D uncertainty models for motion and structure. We further develop an algorithm which iteratively improves the 3-D solution given two successive image frames and then discuss a multi-frame algorithm that improves the solution progressively by using a large number of image frames.

1 Introduction

The problem of recovering scene structure and the camera motion relative to the scene has been one of the key problems in computer vision. Many techniques have been developed for the estimation of structure and motion parameters [4,11,20,21,27,28] and a number of methods have been developed to estimate the 2-D image motion [1,6,12,14]. In this work we develop a system for shape and motion perception that utilizes the image motion velocities in order to compute the 3-D parametric uncertainty models for the world structure and motion. We start by addressing the problem of recovering the 2-D flow vectors with uncertainty in real time. Then, we use those velocities to compute the world uncertainty.

The work exámines closely the possibilities for errors, mistakes and uncertainties in the visual system. We divide the problem into four major *levels* for developing uncertainty models. The *sensor* level models deals with the problems in mapping 3-D features to pixel coordinates and the errors incurred in that process. We identify these uncertainties and suggest a framework for modeling them. The next level is the *extraction strategy* level, in which we develop models for the possibility of errors in the low-level image processing modules used for identifying features that are to be used in computing the 2-D evolution of the scene under consideration and computing the image flow. In the third level, we utilize the known geometric and mechanical properties to reject unrealistic estimates for 2-D movements that might have been obtained from the first two levels.

After having obtained 2-D models for the evolution of the hand/object relationship, we transform the 2-D uncertainty models into 3-D uncertainty models for the structure and motion of the entire scene. The fourth level uses the equations that govern the 2-D to 3-D relationship to perform the conversion. We also develop algorithms that could be used to solve the transformation with uncertainty efficiently and in real time. An algorithm is developed that improves the solution iteratively from a set of two image frames. Then, we discuss a system that improves the solution progressively by using a large number of image frames. A number of other closed-form solution algorithms are also discussed.

0-8194-0750-X/92/$4.00

2 Recovering Image Motion

Many techniques were developed to estimate the optic flow (the 2-D image motion vectors) [1,6,12,14, 16,30], we propose an algorithm for calculating the image flow and then we discuss a simpler version of the same algorithm for real time detection of the 2-D motion vectors. The image flow detection technique we use is based on the sum-of-squared-differences optic flow. We consider two images, 1 and 2. For every pixel (x, y) in image 1 we consider a pixel area N surrounding it and search a neighboring area S to seek a corresponding area in image 2 such that the sum of squared differences in the pixel gray levels is minimal as follows :

$$SSD(\acute{x}, \acute{y}) = \min_{\acute{x},\acute{y} \in S} \sum_{\Delta x, \Delta y \in N} [E(x + \Delta x, y + \Delta y) - \acute{E}(\acute{x} + \Delta x, \acute{y} + \Delta y)]^2$$

The image flow vector of pixel (x, y) then points from the center of N in the first image to the center of the best match in the second image. The search area S should be restricted for practicality measures. In the case of multiple best matches, we can use the one which implies minimum motion, as a heuristic favoring small movements. It should be noted that the accuracy of direction and magnitude of the optic flow determination depends on the sizes of the neighborhoods N and S.

There are three basic problems with this simple approach, one is that the sum of squared differences will be near zero for all directions wherever the graylevel is relatively uniform, the second is that it suffers from the so-called "aperture problem" even if there is a significant graylevel variation. To illustrate this point, consider a vertical edge moving to the right by one pixel distance, and suppose the N window size is 3×3 pixels and the S window size is 5×5 pixels, the squared-differences at an edge point reaches its maximum for three directions as indicated by the vectors (in pixel displacements); $(1, 0)$, $(1, -1)$ and $(1, 1)$. The third problem is that the scheme will only determine the displacement to pixel accuracy.

We solve the first problem by estimating the motion only at the pixels (as determined by the two-dimensional segmentation scheme) where the intensity changes significantly. The Sobel edge detector is applied to the first image to estimate the edge magnitude $M(x, y)$ and direction $D(x, y)$ for every pixel :

$$M(x, y) \approx \sqrt{E_x^2 + E_y^2}$$

$$D(x, y) \approx tan^{-1}\left(\frac{E_x}{E_y}\right)$$

where E_x and E_y are the partial derivatives of the first image with respect to x and y, respectively. The edge direction and magnitude is discretized depending on the size of the windows N and S. The motion is then estimated at only the pixels where the gradient magnitude exceeds the input threshold value. Motion ambiguity due to the aperture problem can be solved by estimating only the *normal* flow vector. It is well known that the motion along the direction of intensity gradient only can be recovered. Then we evaluate the SSD functions at only those locations that lie on the gradient directions and choose the one corresponding to the minimal SSD, if more than one minimal SSD exist we can choose the one corresponding to the smallest movement, as described above. The full flow vector can then be estimated by using the following equation which relates the normal flow vector $\vec{v}_n$ to the full flow vector $\vec{v}$.

$$\vec{v}_n = \vec{v}.\vec{n}$$

This method works under the assumption that the image motion is locally constant. Solving the over-determined linear system will result in a solution for the full flow. The least square error of the system can help us to decide whether the assumption is a reasonably valid one. On the other hand, full flow determination can be performed for small clusters of points in the image and a number of full flow estimates is then used for 3-D recovery.

To obtain sub-pixel accuracy, we can fit a one-dimensional curve along the direction of the gradient for all the SSD values obtained. A polynomial of the degree of the number of points used along the gradient can be used to obtain the best precision. However, for an S window of size 7×7 pixels or less and an N window of size 3×3 or so, a quadratic function can be used for efficiency and to avoid optimizational instabilities for higher order polynomials. The subpixel optimum can be obtained by finding the minimum of the function used and using the displacement at which it occurred as the image flow estimate. To avoid probable discontinuities in the SSD values, the image could be smoothed first using a gaussian with a small variance.

A simpler version of the above algorithm can be implemented in real-time using a multi-resolution approach [30]. We can restrict the window size of N to 3×3 and that of S to 5×5, and perform the algorithm on different levels of the gaussian image pyramid. A gaussian pyramid is constructed by the successive applications of gaussian low-pass filtering and decimation by half. The pyramid processor, PVM-1 is capable of producing complete gaussian pyramid from a 256 by 256 image in one video frame ($\frac{1}{30}$ of a second). Maxvideo boards can be used for the simultaneous estimation of image flow at all the levels of the pyramid for all the pixels. Image flow of 1 pixel at the second level would correspond to 2 pixels in the original image, 1 pixel displacement at the third level would correspond to 4 pixels in the original image, and so on. The level with the smallest least square fitting error of the normal flow can be chosen to get the full flow and the motion vector is scaled accordingly. This method is crude in the sense that it only allow image flow values of 1,2,4 or 8 pixel displacement at each pixel, but it can be used for detecting fast movements.

By either using a flow recovery algorithm or a feature identification and tracking algorithm, we end up having a set of values for 2-D displacements of a number of pixels. The problem is how can we model the uncertainty in those 2-D estimates, which are to be used later for 3-D parameter recovery. For example, if the estimate is - for a specific 3-D feature - that pixel (x_i, y_j) has moved to pixel (x_m, y_n), then the problem reduces to finding space probability distributions for the four indices. The sensor acquisition procedure (grabbing images) and uncertainty in image processing mechanisms for determining features are factors that should be taken into consideration when we compute the uncertainty in the optic flow.

3 Sensor Uncertainties

In this section and the next two sections we develop and discuss modeling the uncertainties in the recovered 2-D displacement vectors. There are many sources of errors and ways to model uncertainties in image processing [29]. As mentioned in the section describing techniques for recovering the image flow, the uncertainty in the recovered values results from sensor uncertainties and noise and from the image processing techniques used to extract and track features. When dealing with measurements of any sort, it is always the case that the measurements are accompanied by some error. Mistakes also occur, where mistakes are not large errors but failures of a system component or more. A description of errors, mistakes and modeling them can be found in [2,3].

In this section we discuss errors in image formation. The camera is used to grab and register images of the scene, so we need to know errors in mapping from the 3-D world features to the 2-D domain which we use in forming 3-D hypothesis about the scene under supervision. The accuracy, precision

and modeling uncertainty of the camera as our sensor is an important issue and the first step towards forming a full uncertainty model for recovering the 3-D models.

As a lot of the image processing algorithms compute derivatives of the intensity function, noise in the image will be amplified and propagated throughout the imaging process. The goal of this treatment is to find a distribution for the uncertainty of mapping a specific 3-D feature into a specific pixel value. In other words, if the feature 2-D position was discovered to be (i, j), then the goal is to find a 2-D distribution for i and j, assuming that there is no uncertainty in the technique used to extract the 2-D feature, the technique's uncertainty will be discussed in the next section. The end product of modeling the sensor uncertainty is to be able to say a statement like : "The 3-D feature F is located in the 2-D pixel position (i, j) with probability p_1 or located in the 2-D pixel position $(i, j+1)$ with probability p_2 or .... *given* that the registered location is (l, m), such that $p_1 + p_2 + ... + p_n = 1$, and $\not{B}$ error in the 2-D feature recovery mechanism."

3.1 Image Formation Errors

The errors in the image formation process are basically of two different kinds. The first type is a spatial error, the other type is a temporal error. The spatial error due to the noise characteristics of a CCD transducer can be due to many reasons, among which are dark signatures and illumination signatures. The technique to be used is to take a large number of images, we can denote the image intensity function as a 3-D function $I(u, v, t)$, with spatial arguments u and v and temporal argument t. The sample mean of the image intensities over N time samples can be denoted by $\overline{I}(u, v)$.

$$\overline{I}(u, v) = \frac{1}{N} \sum_{t=1}^{N} I(u, v, t)$$

The spatial variance in a 5×5 neighborhood of the means is computed by:

$$s^2(u, v) = \sum_{i=-2}^{2} \sum_{j=-2}^{2} (\overline{I}(u+i, v+j) - \overline{I}(u, v))^2$$

The dark signature of the camera can be determined by computing $\overline{I}(u, v)$ of each pixel with the lens cap on. It will be found that a small number of pixels will have non-zero mean and non-zero variance. The specific pixel locations are blemished and should be registered. The uniform illumination is computed by placing a nylon diffuser over the lens and computing the mean and variance. It will be noticed that due to digitizing the CCD array into a pixel array of different size, and the difference in sample rates between the digitizer and camera, the border of the image will have different mean and variance from the interior of the image. Some "stuck" pixels at the location of the blemished pixels will also be noted. The contrast transfer function will also be noted to vary at different distances from the center of the lens.

Temporal noise characteristics can also be identified by taking a number of experiments and notice the time dependency of the pixels intensity function. In our treatment and for our modeling purposes we concentrate on the spatial distribution of noise and its effect on finding the 2-D uncertainty in recovering a 3-D feature location in the pixel array.

3.2 Calibration and Modeling Uncertainties

Methods to compute the translation and rotation of the camera with respect to its coordinates, as well as the camera parameters, such as the focal length, radial distortion coefficients, scale factor and

the image origin, have been developed and discussed in the literature [5,15,24]. In this section we use a static camera calibration technique to model the uncertainty in 3-D to 2-D feature locations. In particular we use the sequence of steps used to transform from 3-D world coordinates to computer pixel coordinates in order to recover the pixel uncertainties, due to the sensor noise characteristics described previously.

A sequence of computational steps can be used for a coplanar set of points in order to obtain the rotation and translation matrices, in addition to the camera parameters. The input to the system are two sets of coordinates, (X_f, Y_f), which are the computer 2-D pixel image coordinates in frame memory and (x_w, y_w, z_w), which are the 3-D world coordinates of a set of coplanar points impressed on a piece of paper with known inter-point distances. A discussion of the exact mathematical formulation of the inter-step computations to find all the parameters can be found in [5].

Our approach is to treat the whole camera system as a black box and make input/output measurements and develop a model of its parametric behaviour. The next step is to utilize the recovered camera parameters and the number of 3-D points which we created in order to formulate a distribution of the 2-D uncertainty.

The strategy used to find the 2-D uncertainty in the features 2-D representation is to utilize the recovered camera parameters and the 3-D world coordinates (x_w, y_w, z_w) of the known set of points and compute the corresponding pixel coordinates, for points distributed throughout the image plane a number of times, find the actual feature pixel coordinates and construct 2-D histograms for the displacements from the recovered coordinates for the experiments performed. The number of the experiments giving a certain displacement error would be the z axis of this histogram, while the x and y axis are the displacement error. Different histograms can be used for different 2-D pixel positions distributed throughout the image plane. The three dimensional histogram functions are then normalized such that the volume under the histogram is equal to 1 unit volume and the resulting normalized function is used as the distribution of pixel displacement error, thus modeling the sensor uncertainty. The black box approach is thus used to model errors in a statistical sense.

4 Image Processing Uncertainties

In this section we describe a technique by which developing uncertainties due to the image processing strategy can be modeled. In addition, we end the discussion by combining both the sensor uncertainties developed in the previous section and the models developed in this section to generate distribution models for the uncertainty in estimating the 2-D motion vectors. These models are to be used for determining the full uncertainty in recovering the 3-D structure and motion.

We start by identifying some basic measures and ideas that are used frequently to recognize the behaviour of basic image processing algorithms and then proceed to describe the technique we use in order to compute the error model in locating certain features from their 2-D representation in the pixel array. We concentrate on modeling the error incurred in extracting edges, as edge extraction is a very popular mechanism that is used for both identifying feature points and also for computing 2-D contours of the object under supervision. When we discussed flow recovery techniques before, it was discussed in details that the optic flow recovery algorithm using local matching works well for the intensity boundaries and not for the inside regions.

4.1 Edge Extraction Uncertainties

Edge extraction strategies and methods to evaluate their performance qualitatively and quantatively have been presented and discussed in the literature [8,10,17,19]. There are many types of edges,

ideal, ramp and noisy edges are only three of them. Different curvatures in the edges also constitute another dimension to be taken into consideration when it comes to asserting the types of edges that exist.

The goal of developing the error models for edge extraction to to be able to say a statement like : "Given that the 2-D feature recovered using the edge recovery S is in pixel position (x, y), then there is a probability that the feature was originally at pixel position $(x+1, y)$ with probability p_1 or .... etc. due to the noise in the pixel image, such that $p_1 + p_2 + ... + p_n = 1$." The problem is to find the probabilities.

It should be obvious that there may be different types of noises and also different levels of those types that might vary at different locations in the sensor image plane. This adds to the different models that we might have to construct. Our approach is to use ideal, that is, synthesized edges of different types, locations and also orientations in image frames then corrupt them with different kinds and levels of noises. We know the ideal edge points from the ideal image, for which we shall use the edge detector that is to be used in the experiment. The corrupted images will then be operated upon by the detector and the edge points located. The edge points will differ from the ideal image edge points. The problem reduces to finding corresponding edge points in corrupted and ideal images then finding the error along a large number of edge points. A 2-D histogram is then constructed for the number of points with specific displacement errors from the ideal point. The volume of the histogram is then normalized to be equal to 1, the resulting 3-D function is the 2-D probability density function of the error of displacements.

4.2 Computing 2-D Motion Uncertainty

In this section we describe how to combine sensor and strategy error models to compute models for the recovered image flow values. To simplify the idea, let's assume that we have recovered a specific feature point (x_1, y_1) in an image grabbed at time instant t and the corresponding point (x_2, y_2) at time $t+1$. The problem is to figure out the distribution of v_x. As an example, to explain the procedure, let's assume that from the 3-D sensor distribution we have have computed the marginal density function of the x coordinate of x_1 in the point:

$$f_X(x) = \int_R f_{X,Y}(x, y)dy$$

where R is all the possible y values within the sensor uncertainty model.
The same process is applied for the strategy distribution and another function is recovered. To simplify things, lets assume that for both distributions there is an equal probability equal to $\frac{1}{3}$ that the x coordinate is the same, or shifted one position to the left or the right. Combining the spatial information of both distributions as a convolution process would produce a triangle-like distribution, which is the error probability density function of having the 3-D feature x 2-D coordinate in the recovered image 2-D x position. Further more, assume that x_2 distribution is the same.

The problem reduces to finding the distribution of the optic flow x component, using these two combined distributions. As an example, if $x_1 = 10$ and $x_2 = 22$, then all probability statements can be easily computed, a set of some of these probability statement is shown :

$$P(v_x = 8) = P((x_1 = 12) \wedge (x_2 = 20)) = \frac{1}{9} \times \frac{1}{9} = \frac{1}{81}$$

$$P(v_x = 9) = P(((x_1 = 12) \wedge (x_2 = 21)) \vee ((x_1 = 11) \wedge (x_2 = 20))) = (\frac{1}{9} \times \frac{2}{9}) + (\frac{2}{9} \times \frac{1}{9}) = \frac{4}{81}$$

$$P(v_x = 10 | x_1 = 10) = \frac{P(x_1=10 \wedge x_2=20)}{P(x_1=10)} = \frac{\frac{3}{9} \times \frac{1}{9}}{\frac{3}{9}} = \frac{1}{9}$$

Consequently, all distributions and expected values can be computed from the combination of the sensor level and strategy level uncertainty formulation. Those flow models are then passed to the higher levels for 3-D recovery. In the next section we discuss a method for refining the measured 2-D motion vectors and we then proceed to formulate the 3-D uncertainty modeling of parameters.

5 Refining Image Motion

In this section we describe a method to refine the recovered 2-D motion vectors on the image plane. Having obtained from the sensor and extraction strategy uncertainty levels distribution estimates for the image flow of the different features, we now try to eliminate the unrealistic ones. Faulty estimates can results from noise, errors or mistakes in the sensor acquisition process and/or visual problems like occlusion, modeling the uncertainties in the previous two levels may still leave room for such anomalies.

We can assume that the features to be tracked lie on a planar surface or that segmenting the objects as polyhedra is simple, although the modification would be very simple to allow for arbitrary 3-D positions of the feature distribution. Since we might know a-priori some information about the mechanical capabilities and limitations and geometric properties of the system under observation, also about the rate of visual sampling for the camera, since we actually control that, we might be able to assert some limits on some of the visual parameters in our system.

We know equations that govern the behaviour of the image flow as a function of the structure and 3-D motion parameters, as follows :

$$v_x = (1 - px - qy)\left(x\frac{V_Z}{Z_o} - \frac{V_X}{Z_o}\right) + \left[xy\Omega_X - \left(1 + x^2\right)\Omega_Y + y\Omega_Z\right]$$

$$v_y = (1 - px - qy)\left(y\frac{V_Z}{Z_o} - \frac{V_Y}{Z_o}\right) + \left[\left(1 + y^2\right)\Omega_X - xy\Omega_Y - x\Omega_Z\right]$$

Which are second degree functions in x and y in three dimensions, $v_x = f_1(x, y)$ and $v_y = f_2(x, y)$.

In addition, we know upper and lower limits on the coefficients p, q, V_X, V_Y, V_Z, Ω_X, Ω_Y, Ω_Z and Z_o, as we might know some of the mechanical abilities of the system being observed. So the problem reduces to constructing the three dimensional envelopes for v_x and v_y as the worst case estimates for the flow velocity and rejecting any measured values that lie outside that envelope.

As an example, we write the equation governing the maximum v_x value in the first quadrant of the $x - y$ plane (x^+, y^+).

$$v_{x_{max}} = \left(-\frac{fV_{X_s}}{Z_{o_s}} - f\Omega_{Y_s}\right) + \left(\frac{V_{Z_l}}{Z_{o_s}} + \frac{max(p_l V_{X_l}, p_s V_{X_s})}{Z_{o_s}}\right)x + \left(\frac{max(q_l V_{X_l}, q_s V_{X_s})}{Z_{o_s}} + \Omega_{Z_l}\right)y$$

$$+ \left(\frac{\Omega_{X_l}}{f} - \frac{min(q_l V_{Z_s}, q_s V_{Z_l})}{fZ_{o_s}}\right)xy - \left(\frac{min(p_l V_{Z_s}, p_s V_{Z_l})}{fZ_{o_s}} + \frac{\Omega_{Y_s}}{f}\right)x^2$$

where the subscripts s and l denote lower and upper limits, respectively. At first sight the problem of determining the maximum value of v_x seems to be a constrained non linear optimization problem, which is true, however, assuming that the upper and lower limits of the coefficients are equal in magnitude and opposite in directions (except for Z_o, which is used only as $Z_{o_s}^+$) makes the input to the *max* and *min* functions in the above equations always equal and thus providing one more degree of freedom in choosing the parameters and making the choice consistent throughout the equation.

Thus the problem becomes simply to write eight equations as the above one for each of v_x and v_y, to draw the function in each of the four quadrants for maximum and minimum envelopes. It should be noted that the maximum absolute possible value of the image flow is minimal at the origin of the camera image plane and increases quadratically as the distance increases from the center.

6 Recovering 3-D Parameters

In this section we describe different techniques for recovering the 3-D parameters. In particular, we utilize the refined 2-D motion distributions that were computed in the previous levels in order to achieve a robust estimation of the three dimensional motion and structure vectors of the scene under observation. We develop some techniques for finding estimates of the required parameters and discuss mathematical formulations that will enable us to determine the 3-D parametric distributions.

The problem of recovering scene structure and the camera motion relative to the scene has been one of the key problems in computer vision. Many techniques have been developed for the estimation of structure and motion parameters (Tsai and Huang [23], Weng et al. [28] etc.). A lot of existing algorithms depend on evaluating the motion parameters between two successive frames in a sequence. However, recent research on structure and motion has been directed towards using a large number of frames to exploit the history of parametric evolution for a more accurate estimation and noise reduction (Ullman [26], Grzywacz and Hildreth[11] etc.)

Next, we describe a method for recovering the 3-D motion and orientation of the planar surface from an evolving image sequence. The algorithm utilizes the image flow velocities in order to recover the 3-D parameters. First, we develop an algorithm which iteratively improves the solution given two successive image frames. The solution space is divided into three subspaces - the translational motion, the rotational motion and the surface slope. The solution of each subspace is updated by using the current solution of the other two subspaces. The updating process continues until the motion parameters converge, or until no significant improvement is achieved.

Second, we further improve the solution progressively by using a large number of image frames and the ordinary differential equations which describe the evolution of motion and structure over time. Our algorithm uses a weighted average of the expected parameters and the calculated parameters using the 2-frame iterative algorithm as current solution and continues in the same way till the end of the frame sequence. Thus it keeps track of the past history of parametric evolution.

The solution is further improved by exploiting the temporal coherence of 3-D motion. We develop the ordinary differential equations which describe the evolution of motion and structure in terms of the current motion/structure and the measurements (the 2-D motion vectors) in the image plane. As an initial step we assume that the 3-D motion is piecewise uniform in time. The extended Kalman filter can then be used to update the solution of the differential equations.

6.1 A 3-D Recovery Algorithm

One can model an arbitrary 3-D motion in terms of stationary-scene/moving-viewer. The optical flow at the image plane can be related to the 3-D world as indicated by the following pair of equations (In case of a planar surface), for each point (x, y) in the image plane :

$$v_x = (1 - px - qy)\left(x\frac{V_Z}{Z_o} - \frac{V_X}{Z_o}\right) + \left[xy\Omega_X - \left(1 + x^2\right)\Omega_Y + y\Omega_Z\right]$$

$$v_y = (1 - px - qy)\left(y\frac{V_Z}{Z_o} - \frac{V_Y}{Z_o}\right) + \left[\left(1 + y^2\right)\Omega_X - xy\Omega_Y - x\Omega_Z\right]$$

where v_x and v_y are the image velocity at image location (x, y), (V_X, V_Y, V_Z) and $(\Omega_X, \Omega_Y, \Omega_Z)$ are the translational and rotational velocity vectors of the observer, p and q are the planar surface orientations. The situation becomes, for each point, two equations in eight unknowns, namely, the scaled translational velocities V_X/Z_o, V_Y/Z_o and V_Z/Z_o, the rotational velocities Ω_X, Ω_Y and Ω_Z and the orientations p and q. Differential methods could be used to solve those equations by differentiating the flow field and by using approximate methods to find the flow field derivatives. The existing methods for computing the derivatives of the flow field usually do not produce accurate results. Our algorithm uses a discrete method instead, i.e, the vectors at a number of points in the plane is determined and the problem reduces to solving a system of nonlinear equations.

It should be noticed that the resulting system of equations is nonlinear, however, it has some linear properties. The rotational part, for example, is totally linear, also, for any combination of two spaces among the rotational, translational and slope spaces, the system becomes linear. For the system of equations to be consistent, we need the flow estimates for at least four points, in which case there will be eight equations in eight unknowns.

6.1.1 Two-Frame Algorithm

The algorithm takes as input the estimate of the flow vectors at a number of points ≥ 4 obtained from motion between two images. It iterates updating the solution of each subspace by using the solution of the other two subspaces. Each update involves solving a linear system, thereby it requires to solve three linear systems to complete a single iteration. This process continues until the solution converges, or until no significant improvement is made. The algorithm proceeds as follows :

1. Set p, $q = 0$;
   input the initial estimate for rotation ;
   Solve the linear system for translation;
2. Use the translation and rotation from step 1 ;
   Solve the linear system for the slope ;
3. Set i=1;
   While (i $\leq$ Max. Iterations) and (no convergence) Do
   Solve for the rotations using latest estimates of translations, p and q;
   Solve for the translations using latest estimates of rotations, p and q;
   Solve for p, q using latest estimates of translations and rotations;
   end While ;

6.1.2 Complexity Analysis

As we mentioned earlier, one should notice in the equations relating the flow velocities with the slope, rotational and translational velocities that they are "quasi-linear" , if one can say so. The equations exhibit some linear properties. This suggests that a purely iterative technique for solving non-linear equations might not be an excellent choice, since, the variables are linearly related in some way. To think of a way of "inverting" the relations might be a good start, although to do that without a framework based on iterating and gravitating towards a solution is not a good idea.

This makes one think of applying a method which converges faster than a purely iterative scheme like Newton's method. However, the complexity of Newton's method is determined by the complexity

of computing the inverse Jacobian, which is of an order of N^3, or $N^{2.81}$ multiplications as the lower bound using Strassen's technique. In our case, since we have at least 8 equations in 8 unknowns, the complexity is of order $8^3 = 512$ multiplications at every iteration, and the method does not make any use of the fact that the set of equations at hand exhibits some linear properties.

The algorithm proposed, on the other hand, makes very good use of the fact that there are some linearity in the equations, by inverting the set of relations for each subspace at every iteration. The complexity at every iteration is of the order of the complexity of computing the pseudo-inverse which is of the order of ($3^3+3^3+2^3$) multiplications at each iteration, where the first 3 comes from solving the system for the rotational variables, the second 3 is for the translations, the last 2 is for p and q. This is equal to 62 multiplications at every iteration, which is significantly less than the 512 multiplications in a method like Newton's for example. It was noticed that the algorithm converged to solution in a very small number of iterations for most experiments we have conducted so far. The maximum number of iterations was 6.

Using the latest solution obtained from the two-frame analysis as the initial condition for the next two-frame problem in the image sequence would further decrease the complexity, as the next set of parameters would, most probably, be close in values to the current parameters, thus the number of iterations needed to converge to the new solution would decrease significantly.

6.1.3 Observations

- The algorithm is not sensitive to the initial condition of the orientation parameters. The plane is simply assumed to be a frontal one at the beginning. The slope parameters evolves with iterations.
- The algorithm is sensitive to input noise just like other existing algorithms, some experiments shows the sensitivity with respect to the change of viewing angle. Similarly, the algorithm performs better for a large number of points that are evenly distributed throughout the planar surface, than it does for clustered, smaller number of image points.
- It is proven that there exists dual solutions for such systems. However, if our method gravitates towards a "fixed point" in the solution space we can find the other explicitly in terms of the first one from the relations given by Waxman and Ullman [27].

6.1.4 Multi-Frame Algorithm

The ordinary differential equations that describe the evolution of motion and structure parameters are used to find the expression for the expected parameter change in terms of the previous parameter estimates. The expected change and the old estimates are then used to predict the current motion and structure parameters.

At time instant t, the planar surface equation is described by

$$Z = pX + qY + Z_o$$

To compute the change in the structure parameters during the time interval dt, we differentiate the above equation to get

$$\frac{dZ}{dt} = p\frac{dX}{dt} + X\frac{dp}{dt} + q\frac{dY}{dt} + Y\frac{dq}{dt} + \frac{dZ_o}{dt}$$

The time derivatives of (X, Y, Z) in the above expression are given by the three components of the vector $-(\mathbf{V} + \mathbf{\Omega} \times \mathbf{R})$ that represent the relative motion of the object with respect to the camera. Substituting these components for the derivatives and the expression $pX + qY + Z_o$ for Z we can get the exact differentials for the slopes and Z_o as

$$\begin{aligned} dZ_o &= Z_o\left[(\Omega_Y + V_X)p - (\Omega_X - V_Y)q - V_Z\right] dt \\ dp &= \left[p(\Omega_Y p - \Omega_X q) + (\Omega_Y + \Omega_Z q)\right] dt \\ dq &= \left[q(\Omega_Y p - \Omega_X q) - (\Omega_X + \Omega_Z p)\right] dt \end{aligned}$$

Using the above relations, we can compute the new structure parameters at time $t + dt$ as

$$\acute{p} = p + dp\ ,\ \acute{q} = q + dq \ \text{ and } \ \acute{Z}_o = Z_o + dZ_o$$

Thus the slope parameters evolve at time $t + dt$ as follows :

$$\begin{bmatrix} \acute{p} \\ \acute{q} \end{bmatrix} = \begin{bmatrix} p \\ q \end{bmatrix} + \begin{bmatrix} \Omega_Y p - \Omega_X q & \Omega_Z & \Omega_Y \\ -\Omega_Z & \Omega_Y p - \Omega_X q & -\Omega_X \end{bmatrix} \begin{bmatrix} p \\ q \\ 1 \end{bmatrix} \ dt$$

The new translational velocity $\acute{V}$ at time $t + dt$ can be found in the absence of accelerations from

$$\acute{\mathbf{V}} = \mathbf{V} + \mathbf{V} \times \mathbf{\Omega} dt$$

Dividing $\acute{V}$ by $\acute{Z}_o$ we get the new expected scaled translational velocity components at time $t + dt$ as follows :

$$\begin{bmatrix} \acute{V}_X \\ \acute{V}_Y \\ \acute{V}_Z \end{bmatrix} = \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} + \begin{bmatrix} -s & \Omega_Z & \Omega_Y \\ -\Omega_Z & -s & \Omega_X \\ \Omega_Y & -\Omega_X & -s \end{bmatrix} \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} \ dt,$$

where s is expressed as follows :

$$s = (\Omega_Y + V_X)\, p - (\Omega_X - V_Y)\, q - V_Z$$

The expected rotational parameters at time $t + dt$ remain equal to their values at time t since

$$\acute{\mathbf{\Omega}} = \mathbf{\Omega} + \mathbf{\Omega} \times \mathbf{\Omega} dt = \mathbf{\Omega}$$

and thus

$$\left(\acute{\Omega}_X, \acute{\Omega}_Y, \acute{\Omega}_Z\right) = (\Omega_X, \Omega_Y, \Omega_Z)$$

Our first multi-frame algorithm uses a weighted average of the expected parameters at time $t+dt$ from the above equations and the calculated parameters using the two-frame iterative algorithm as the solution at time $t+dt$, and continues in the same way until the end of the frame sequence. Thus it keeps track of the past history of parametric evolution. We further develop the first multi-frame algorithm to exploit the temporal coherence of 3-D motion. We develop the ordinary differential equations which describe the evolution of motion and structure in terms of the current motion/structure and the two-dimensional flow vectors in the image plane. We assume that the 3-D motion is piecewise uniform in time, i.e, $\dot{\mathbf{\Omega}} = \dot{\mathbf{V}} = \mathbf{0}$. We then use the equations expressing the time derivative of the slope derived above and the fact that the derivative of the rotational velocities is zero and develop the following expressions for the scaled translational velocities and the depth Z_o :

$$\frac{dV_X'}{dt} = -V_X' \frac{1}{Z_o} \frac{dZ_o}{dt}, \quad \frac{dV_Y'}{dt} = -V_Y' \frac{1}{Z_o} \frac{dZ_o}{dt} \quad \text{and} \quad \frac{dV_Z'}{dt} = -V_Z' \frac{1}{Z_o} \frac{dZ_o}{dt}$$

$$\frac{1}{Z_o} \frac{dZ_o}{dt} = -V_Z' - pv_x - qv_y$$

The extended Kalman filter can then be used to update the solution of the differential equations. Parallel implementations could be designed for the system, thus solving for the structure - motion parameters for each surface separately. In fact, solving the linear system at each iteration could also be parallelized.

6.2 Other Algorithms

There are other non-iterative techniques for recovering the 3-D parameters resulting from 2-D motion between two frames. The methods that will be mentioned here rely on specific assumption regarding the scene's geometry and/or world actions. Assuming that the actual relations between feature points that lie on some of the scene's planar surfaces are well defined than a closed form solution for the structure parameters and depth can be estimated by using a method like the one described by Fischler and Bolles [9]. The motion parameters can then be easily recovered by solving a linear system in six parameters.

It should be noticed that we try to use alternative methods in order to solve linear equations, the motive behind that is the fact that linear systems can be solved in a pseudo-real time framework for a relatively small number of feature points and in addition a closed form solution always results. Another idea is to assume that some surfaces are frontal at the time of capturing the frame to be processed with the previous one, thus p and q are equal to zero, and the problem reduces to solving a linear system in six parameters for the motion parameters, while the depth is easily computed by knowing the 3-D distance between any two feature points.

The assumption here being that the camera always locates itself to a position in which some object surfaces are frontal with respect to the camera image plane, and that the movements while the camera is moving and during computations is negligible. Other formulations may attempt to find pseudo-close form solution of the non-linear second order system and other assumptions, like the absence of rotational and/or translational motion reduces the complexity significantly.

6.3 Recovering 3-D Uncertainties

Having discussed methods for computing the three dimensional motion vectors and structure parameters between two image frames, we now use the same formulations described earlier for 3-D recovery but using 2-D error distributions as estimates for motion and/or feature coordinates in order to compute 3-D uncertainty distributions for the real world motion vectors and structure instead of singular values for the world parameters.

As an example to illustrate the idea, let's assume a linear system of equations as follows :

$$x + 3y = z_1$$

$$2x + y = z_2$$

The solution of this system is very easily obtained as :

$$x = \frac{3}{5} z_2 - \frac{1}{5} z_1$$

$$y = \frac{2}{5}z_1 - \frac{1}{5}z_2$$

That is, a linear combination of the right hand side parameters. If the parameters z_1 and z_2 were random variables of known probability distributions instead of constants, then the problem becomes slightly harder, which is, to find the linear combination of those random variables as another random variable. The obvious way of doing this would be to use convolutions and the formula :

$$P_{X_1+X_2}(y) = \sum_R P_{X_1,X_2}(x, y-x)$$

for the sum of two random variables X_1, X_2 for any real number y and/or the formula for linear combinations over the region R, which is for all x such that $P_{X_1,X_2}(x, y-x) > 0$. Using the moment generating function or the characteristic function seems also to be a very attractive alternative. The moment generating function M of a linear combination of random variables, for example X_1, X_2 can be written as :

$$M_{aX_1+bX_2+c}(t) = e^{ct}\left(M_{X_1}(at)M_{X_2}(bt)\right)$$

for independent random variables X_1, X_2. That is, the problem of solving linear systems on the form $Ax = b$, where b is a vector of random variables, may be reduced to finding closed form solutions for x in terms of the random parameters (using any elimination technique) and then manipulating the results and finding different expectations using moment generating or characteristic functions.

The solutions we suggest to this problem of finding the random variable solution of the 3-D parameters utilize the techniques we described in the previous two subsections. Using either the two-frame iterative technique or the closed form algorithms, it should be noticed that the problem reduces to either solving multi-linear systems or a single one. In that case, using elimination and characteristic functions for computing the required expectations and distributions is straight forward.

7 Conclusions

We have suggested algorithms for the quick estimation of the 3-D uncertainties in the structure and motion of a moving scene. Our system utilizes the image flow velocities in order to recover the motion and structure estimates of the uncertainty in the 3-D world parameters. Sensor and strategy uncertainties are used and propagated in order to recover the 3-D distributions. The system can be used for a variety of application ranging from simple recognition tasks to tracking objects in a robust and stable manner. It allows for error recovery, as the uncertainty modelling presents estimates for the correctness of the recovered parameters.

References

[1] P. Anandan, "A Unified Perspective on Computational Techniques for the Measurement of Visual Motion". In *Proceedings of the* 1^{st} *International Conference on Computer Vision*, 1987.

[2] H. L. Anderson, *GRASP lab. Camera Systems and Their Effects on Algorithms*, Technical Report MS-CIS-88-85 and GRASP lab. TR 161, University of Pennsylvania, 1988.

[3] R. Bajcsy, E. Krotkov and M. Mintz, *Models of Errors and Mistakes in Machine Perception*, Technical Report MS-CIS-86-26 and GRASP lab. TR 64, University of Pennsylvania, 1986.

[4] J. L. Barron, A. D. Jepson and J. K. Tsotsos, "The Feasibility of Motion and Structure from Noisy Time-Varying Image Velocity Information", *International Journal of Computer Vision*, December 1990.

[5] N. M. Benahmed, *Camera Calibration for Dynamic Environment.* M.S. Thesis, Department of Electrical Engineering, University of Pennsylvania, 1989.

[6] P. J. Burt, C. Yen, and X. Xu, "Multiresolution Flow-Through Motion Analysis". In *Proceedings of the 1983 IEEE Conference on Computer Vision and Pattern Recognition.*

[7] P. J. Burt, et al., "Object Tracking with a Moving Camera", *IEEE Workshop on Visual Motion*, March 1989.

[8] E. S. Deutsch and J. R. Fram, "A Quantitative Study of the Orientation Bias of some Edge Detector Schemes", *IEEE Trans. Comput.*, C-27, No. 3, March 1978.

[9] M. Fischler and R. C. Bolles, *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*, Readings in Computer Vision, Morgan Kaufmann Publishers, 1987.

[10] J. R. Fram and E. S. Deutsch, "On the Quantitative Evaluation of Edge Detection Schemes and Their Comparison with Human Performance", *IEEE Trans. Comput.*, C-24, No. 6, June 1975.

[11] N. M. Grzywacz and E. C. Hildreth, *The Incremental Rigidity Scheme for Recovering Structure from Motion: Position vs. Velocity Based Formulations*, MIT A.I. Memo No. 845, October 1985.

[12] D. J. Heeger, *Models for Motion Perception.* Ph.D. Thesis, Computer and Information Science Department, University of Pennsylvania, September 1987.

[13] J. Heel, "Dynamic Motion Vision", In *Proceedings of the SPIE Conference on Computer Vision*, November 1989.

[14] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow", *Artificial Intelligence*, vol. 17, 1981, pp. 185-203.

[15] A. Izaguirre, P. Pu and J. Summers, "A New Development in Camera Calibration: Calibrating a Pair of Mobile Cameras", In *Proceedings of the International Conference on Robotics and Automation*, pp. 74-79, 1985.

[16] D. Keren, S. Peleg and A. Shmuel, *Accurate Hierarchical Estimation of Optic Flow*, TR–89–9, Department of Computer Science, The Hebrew University of Jerusalem, June 1989.

[17] E. Krotkov, *Results in Finding Edges and Corners in Images Using the First Directional Derivative*, Technical Report MS-CIS-85-14 and GRASP lab. TR 37, University of Pennsylvania, 1985.

[18] H. C. Longuet-Higgins and K. Prazdny, *The interpretation of a moving Retinal Image*, Proc. Royal Society of London B, 208, 385-397.

[19] T. Peli and D. Malah, "A Study of Edge Detection Algorithms", *Computer Graphics and Image Processing*, vol. 20, 1982, pp. 1-21.

[20] T. M. Sobh and K. Wohn, "Recovery of 3-D Motion and Structure by Temporal Fusion". In *Proceedings of the* 2^{nd} *SPIE Conference on Sensor Fusion*, November 1989.

[21] M. Subbarao and A. M. Waxman, *On The Uniqueness of Image Flow Solutions for Planar Surfaces in Motion*, CAR–TR–113, Center for Automation Research, University of Maryland, April 1985.

[22] C. Tomasi and T. Kanade, "Shape and Motion without Depth", CMU-CS-90-128, School of Computer Science, Carnegie Mellon University, May 1990.

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

[24] R. Y. Tsai, "An Efficient and Accurate Camera Calibration Technique for 3-D Machine Vision", IBM Report.

[25] S. Ullman, "Analysis of Visual Motion by Biological and Computer Systems", *IEEE Computer*, August 1981.

[26] S. Ullman, *Maximizing Rigidity: The incremental recovery of 3-D structure from rigid and rubbery motion*, AI Memo 721, MIT AI lab. 1983.

[27] A. M. Waxman and S. Ullman, *Surface Structure and 3-D Motion From Image Flow: A Kinematic Analysis*, CAR–TR–24, Center for Automation Research, University of Maryland, October 1983.

[28] 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.

[29] R. Wilson and G. H. Granlund, "The Uncertainty Principle in Image Processing", IEEE Trans. PAMI, Vol. 6, No. 6, November 1984.

[30] K. Wohn and S. R. Maeng, "Real-Time Estimation of 2-D Motion for Object Tracking", In *Proceeding of the SPIE Conference on Intelligent Robotics*, November 1989.