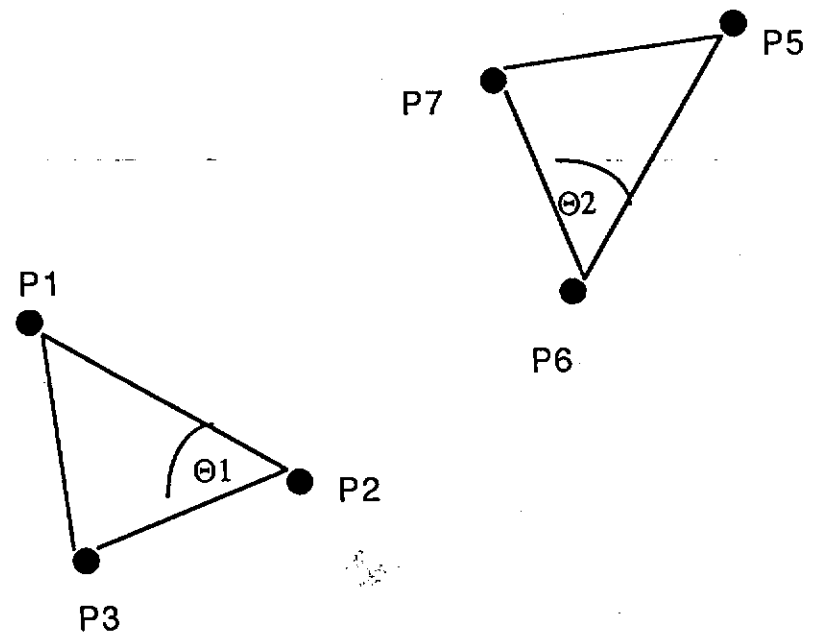

Figure 3. Angular invariance during motion.

Structure and Motion of Entire Polyhedra

Tarek M. Sobh and Tarek Alameldin

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

Abstract

We describe an efficient system for recovering the 3-D motion and structure of entire polyhedra from an evolving image sequence. The suggested technique utilizes the image flow velocities in order to recover the 3-D parameters. We develop a method for estimating the image flow velocities and an algorithm for computing the 3-D parameters given two successive image frames. The solution is then improved by using a large number of image frames and exploiting the temporal coherence of 3-D motion. We use the ordinary differential which describe the parametric evolution in terms of the current motion/structure and the measurements in the image plane. The extended Kalman filter is then used to update the solution. The process is started by segmenting the entire scene into a number of planar surfaces and then applying the above technique to each surface under consideration, the probable inconsistencies are then resolved.

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 [2,6]. 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 [3,5,7].

In this paper we describe a method for recovering the 3-D motion and orientation of entire polyhedra from an evolving image sequence. The algorithm utilizes the image flow velocities in order to recover the 3-D parameters. First, we develop a method to recover the image flow of the scene. Our method overcomes some of the shortcomings of the existing algorithms and is accurate to a large extent. The recovered 2-D motion vectors are then fed into an algorithm which iteratively improves the 3-D 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 is then used to update the solution of the differential equations. The scene is segmented into a number of planar surfaces and the above technique is applied to each surface under consideration. The inconsistencies in motion and structure estimates between different surfaces are then resolved.

2 Modeling and Flow Recovery

Many techniques were developed to estimate the optic flow (the 2-D image motion vectors) [8,9,11,12], 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 moving polyhedra. 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 polyhedra image 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 on the accuracy of our estimate.

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 [12]. 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.

The optical flow at the image plane can be related to the 3-D world, by using the stationary-scene/moving-viewer formulation as indicated by the following pair of equations originally derived by Longuet-Higgins and Prazdny [1], for each point (x, y) in the image plane :

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

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

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, and Z is the unknown distance from the camera to the object.
For planar surfaces, the Z function is simply $pX + qY + Z_o$, where 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, a pair of equations represents the flow at each point as follows :

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

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

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.

3 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.

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. The algorithm proposed, 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 62 multiplications at each 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.

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. 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 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 [4].

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

$$dZ_o = Z_o\,[(\Omega_Y + V_X)p - (\Omega_X - V_Y)q - V_Z]\,dt$$

$$dp = [p(\Omega_Y p - \Omega_X q) + (\Omega_Y + \Omega_Z q)]\,dt$$

$$dq = [q(\Omega_Y p - \Omega_X q) - (\Omega_X + \Omega_Z p)]\,dt$$

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

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

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, $\ddot{\mathbf{\Omega}} = \ddot{\mathbf{V}} = \mathbf{0}$. 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 :

$$\frac{d\acute{V_X}}{dt} = -\acute{V_X} \frac{1}{Z_o} \frac{dZ_o}{dt}, \; \frac{d\acute{V_Y}}{dt} = -\acute{V_Y} \frac{1}{Z_o} \frac{dZ_o}{dt} \quad \text{and} \quad \frac{d\acute{V_Z}}{dt} = -\acute{V_Z} \frac{1}{Z_o} \frac{dZ_o}{dt}$$

$$\frac{1}{Z_o} \frac{dZ_o}{dt} = -\acute{V_Z} - p v_x - q v_y$$

The extended Kalman filter is then used to update the solution of the differential equations. Where the state vector can be written as :

$$X = [\; \acute{V_X} \;\; \acute{V_Y} \;\; \acute{V_Z} \;\; \Omega_X \;\; \Omega_Y \;\; \Omega_Z \;\; p \;\; q \;]$$

and the measurement vector is expressed as :

$$Z = [\; v_x \;\; v_y \;\; \frac{\delta v_x}{\delta x} \;\; \frac{\delta v_y}{\delta x} \;\; \frac{\delta v_x}{\delta y} \;\; \frac{\delta v_y}{\delta y} \;\; \frac{\delta v_x}{\delta t} \;\; \frac{\delta v_y}{\delta t} \;]$$

The filtering mechanism is applied to each surface in the scene. Inconsistencies evolving between the motion and structure parameters are resolved by computing a global estimate for the camera motion parameters from more than one planar patch in a rigid body and then solving for the structure parameters for the remaining surfaces in the scene, while allowing for some small perturbations in the recovered motion values.

5 Conclusions

We described a method for recovering scene structure and motion of the observer. A system is developed utilizing the image flow velocities recovered within a large sequence of frames to determine the 3-D parameters. Parallel implementations could be designed, thus solving for the structure - motion parameters for each surface separately. In fact, solving the linear system at each iteration could also be parallelized. The recovery method has a variety of applications. It can be useful in vision-guided applications such as autonomous landing and navigation. It can also be utilized for robotics applications in the "Moving Blocks World".

References

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

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] S-L. Iu and K. Wohn, "Estimation of 3-D Motion and Structure Based on a Temporally Oriented Approach with the Method of Regression", *IEEE Workshop on Visual Motion*, March 1989, Irvine, CA, 273-281.

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

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

[10] 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.*

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

[12] 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.