References

[1] P. Anandan, "A unified perspective on computational techniques for the measurement of visual motion", Image Understanding Workshop, pp.719-732, 1987.

[2] N. Ayache and O. Faugeras, "Building, registrating and fusing noisy visual maps", IEEE Int. Conf. on Rob and Automation, pp. 73-82, 1987.

[3] H. Durrant-Whyte, "Integration, Coordination and Control of Multi-sensor robot systems", Kluwer Academic lishers, 1988.

[4] A. Elfes, "Using occupancy grids for mobile robot perception and navigation", IEEE Computer, pp. 46-57, 1989.

[5] A.Gelb, Applied Optimal Estimation, The MIT Press, 1974.

[6] R. Jain, "Environment models and information assimilation", IBM Almaden Research Lab. Tech. Report, 1989

[7] L. Matthies, T. Kanade and R. Szeliski, "Kalman Filter-based algorithms for estimating depth from image quences", Int. Journal of Comp. Vision, 3, pp.209-236, 1989.

[8] S. Moezzi and T. Weymouth, "A computational model for dynamic vision", Proc. of the Int. Conf. of Robotics Automation, May 1990.

[9] H. Moravec, "Sensor fusion in certainty grids for mobile robots", AI Magazine, pp. 61-74, Summer 1988.

[10] P.J.Rousseuw, "Least Median of Squares Regression", J. Amer. Stat. Assoc. 79,388, pp. 871-880, 1984.

[11] G. Shafer, "A mathematical theory of evidence", Princeton University Press, 1976.

[12] A. Tirumalai, R. Jain and B. Schunck, "Evidential reasoning for building environment maps", CSE-TR-67-90, Dep of Elec. Engg. and Comp. Sci., Univ. of Michigan, Ann Arbor, Michigan 48109.

[13] A. Tirumalai, B. Schunck and R. Jain, "Robust dynamic stereo with incremental disparity map refinement", Inter national Workshop on Robust Computer Vision, Oct. 1990.

[14] H. L. Van Trees, "Decision, Estimation and Modulation Theory", MIT Press, 1967.

[15] L. Wesley, "Evidential knowledge-based computer vision", Optical Engineering, Vol. 25 No. 3, pp. 363-379, March 1986.

An Efficient System for 3-D Object Recognition

Tarek M. Sobh and Tarek K. Alameldin

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

Abstract

This paper presents an efficient system for recovering the structural properties of objects. We build adaptive maps for interpreting the object structure. The System is decomposed into four major modules, the *sensing interface* acquires visual data and performs low and intermediate vision tasks for enhancing the acquired image sequences. A *knowledge base* contains different visual primitives and the possible exploratory actions. The *map building and decision making* module gets its input from the sensing interface and utilizes the different predicates that are stored in the knowledge base in order to resolve possible uncertainties. The decisions made in the *map building and decision making* module are then utilized by the *controller* module which resolves possible inconsistencies due to physical limitations of the sensing devices. The above process is repeated until the map contains a minimal number of interpretations that cannot be reduced further.

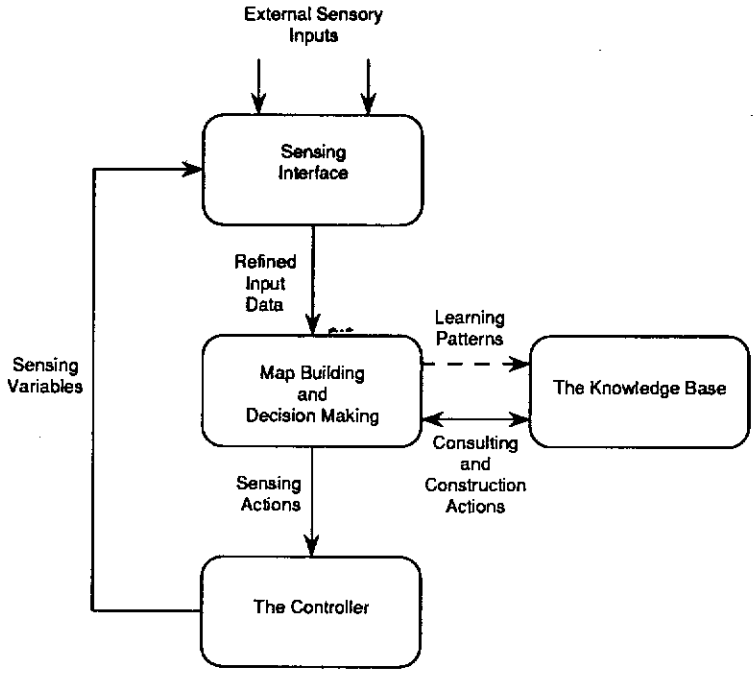
1 Introduction

We address the problem of recovering the structural properties of three-dimensional objects. We emphasize in this work the recognition of polyhedra-like objects, our system can be easily extended to general surfaces. The idea of active sensing [13,14] is utilized by the approach used in building the system. A sequence of images is acquired by a camera mounted on a robot arm, the movements of the sensor (the camera) is guided by the interpretation of the previously acquired sensor inputs throughout the sequence. We build a set of "maps" that are adaptively updated using an existing knowledge base and the sensor inputs. Sensing actions are stopped when a coherent map that cannot be further reduced or improved is built.

The system consists of four basic modules, as depicted in the figure. The sensing interface acquires the images using the camera and also performs some basic low and intermediate level vision tasks. Those tasks include segmenting the image into meaningful surfaces, also our system uses motion primitives in order to recover the structural properties, thus the interface performs the optic flow detection for the relevant surfaces. The interface produces enhanced image sequences and other data that are to be used by the next module, which makes exploratory decisions and constructs the 3-D properties of the viewed object.

The map building and decision making module constructs the 3-D model of the viewed object and is responsible for deciding the next sensory actions, which are mostly exploratory movements. The knowledge base is to be consulted and updated periodically by the map building module to decide on the following actions. Some learning patterns are passed to the knowledge base in case the observer registered a new 3-D pattern that is not anticipated or expected in the knowledge tree. The controller module transforms sensing actions generated by the decision module into a set of allowable three dimensional positioning vectors for the observer's camera. Physical limitations due to the precision and reachability capabilities of the observer's robot arm are resolved and overcome by this module.

The recognition process is stopped when the map building module decides there is no need for mo movements and that the current interpretation is the best possible one.



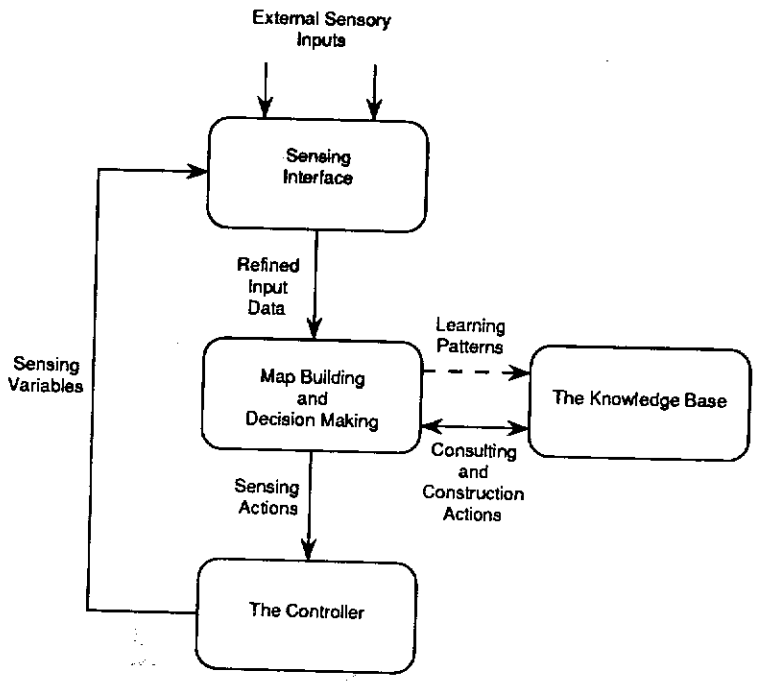
The flow diagram of the suggested system

2 The Sensing Interface

The sensing interface is responsible for acquiring images through the camera and for performing some low and intermediate level vision tasks. The module starts by segmenting the view into the possible patches of planar surfaces, edge detection and the image histogram are utilized for this purpose, the interface might make some movements with predetermined displacements in order to recover a crude estimate of the depth of some points within the viewed region. The interface always stores the previous (to the current) image in the sequence so that it can also performs a computation for the image flow, which we discuss next.

We develop a method to recover the image flow of the scene. Our algorithm overcomes some of the shortcomings of the existing algorithms and is accurate to a large extent. 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

The recognition process is stopped when the map building module decides there is no need for more movements and that the current interpretation is the best possible one.



The flow diagram of the suggested system

2 The Sensing Interface

The sensing interface is responsible for acquiring images through the camera and for performing some low and intermediate level vision tasks. The module starts by segmenting the view into the possible patches of planar surfaces, edge detection and the image histogram are utilized for this purpose, the interface might make some movements with predetermined displacements in order to recover a crude estimate of the depth of some points within the viewed region. The interface always stores the previous (to the current) image in the sequence so that it can also performs a computation for the image flow, which we discuss next.

We develop a method to recover the image flow of the scene. Our algorithm overcomes some of the shortcomings of the existing algorithms and is accurate to a large extent. 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 is used for efficiency and to avoid optimizational instabilities for higher order polynomials. The subpixel optimum is 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 sensing interface performs some moment computations for the different planar patches in the scene, in order to recover their 2-D orientation and positions in the camera plane. The recovered 2-D vectors, positioning and orientation of the different surface, with a label for each surface and a crude estimation for the Z co-ordinate of some points is then fed into the next module. A flag indicating that the required area to be viewed is not viewable might be passed from the controller module and should be reported to the map building module. The sensing interface uses the 3-D positioning vector supplied by the controller module to position the camera.

3 The Knowledge Base

This module interacts with the map building and decision making module. The decision module consults the knowledge base in order to construct the 3-D model of the viewed object. The knowledge base stores information about different 3-D solid models in a knowledge tree. Each node in the tree contains a primitive description of an incomplete model, each branch represents a rule concerning junctions and the possibility of viewing a specific feature *following* the observation of a parent feature, and each leaf represents a 3-D object model. When the knowledge base receives information from the map building module (such as 3-D orientations and how the scene was recognized from a specific position) it tries to find the intermediate node that matches best [1] the visual information.

Then, it constructs the actions to be sent to the decision and map building module by using the different rules stored in the knowledge tree branch of the current node, and the sensor parameters. These actions might be qualitative ones like :

`move_up, move_down, move_right, move_left, move_forward, rotate_direction.`

[1] If it finds more than one node, it chooses one of them at random or according to a pre-specified Bayesian decision rule and pushes the other nodes in stack. It continues exploring that node until it can reach an object interpretation, otherwise, it starts exploring the nodes in the stack

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 is used for efficiency and to avoid optimizational instabilities for higher order polynomials. The subpixel optimum is 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 sensing interface performs some moment computations for the different planar patches in the scene, in order to recover their 2-D orientation and positions in the camera plane. The recovered 2-D vectors, positioning and orientation of the different surface, with a label for each surface and a crude estimation for the Z co-ordinate of some points is then fed into the next module. A flag indicating that the required area to be viewed is not viewable might be passed from the controller module and should be reported to the map building module. The sensing interface uses the 3-D positioning vector supplied by the controller module to position the camera.

3 The Knowledge Base

This module interacts with the map building and decision making module. The decision module consults the knowledge base in order to construct the 3-D model of the viewed object. The knowledge base stores information about different 3-D solid models in a knowledge tree. Each node in the tree contains a primitive description of an incomplete model, each branch represents a rule concerning junctions and the possibility of viewing a specific feature *following* the observation of a parent feature, and each leaf represents a 3-D object model. When the knowledge base receives information from the map building module (such as 3-D orientations and how the scene was recognized from a specific position) it tries to find the intermediate node that matches best [1] the visual information.

Then, it constructs the actions to be sent to the decision and map building module by using the different rules stored in the knowledge tree branch of the current node, and the sensor parameters. These actions might be qualitative ones like :

`move_up, move_down, move_right, move_left, move_forward, rotate_direction.`

[1] If it finds more than one node, it chooses one of them at random or according to a pre-specified Bayesian decision rule and pushes the other nodes in stack. It continues exploring that node until it can reach an object interpretation, otherwise, it starts exploring the nodes in the stack

he sensor parameters include the focal length, the viewing angle, and the other camera parameters, so it includes the link lengths of the manipulator arm. The map building module processes those tions and sends new information as learning patterns to create more intermediate and leaf nodes in e knowledge base tree and more branching rules. Both the map building and the knowledge base odules interact until a 3-D object model is achieved. The map building module can update the nowledge base tree when it finds an object that was not stored in the knowledge base tree before.

Map Building and Decision Making

he map building and decision making module receives the "refined" sensory data from the sensing terface and interacts with the knowledge base in order to construct a solid 3-D model for the ewed object. It uses the 2-D positions, orientations and velocity of the scene to decide what are e features that it should be looking for, or in other words, which unexplored area should it try o view next. In the knowledge base tree, a list of possible 3-D structures are stored with priority ectors attached to each one, the knowledge base uses the data from the decision module and tells t where is the place that it should be exploring next. The decision module passes to the knowledge base a set of 3-D structure parameters for the different segments. Next, we elaborate on the method we use to recover the 3-D parameters from the 2-D vectors supplied by the sensing interface.

This 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 [3,5,7,15].

We describe an efficient system for recovering the 3-D structure of the polyhedra from an evolving image sequence. The suggested technique utilizes the image flow velocities in order to recover the 3-D parameters. We develop 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. 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 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. Differential methods could be used to solve these 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 equations, a pair of equations represents the flow at each point as follows :

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

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

Using the latest solution obtained from the two-frame analysis in an iterative scheme 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 system's solution is better for a large number of points that are evenly distributed throughout the planar surface, than it is for clustered, smaller number of image points. The ordinary differential equations that describe the evolution of motion and structure parameters can be 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 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 :

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

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

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

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

Using the latest solution obtained from the two-frame analysis in an iterative scheme 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 system's solution is better for a large number of points that are evenly distributed throughout the planar surface, than it is for clustered, smaller number of image points. The ordinary differential equations that describe the evolution of motion and structure parameters can be 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 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 :

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

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 utilize the extended Kalman filter to update the solution of the differential equations. The filtering mechanism is applied to each surface in the scene, which can be implemented in parallel.

The current 3-D parameters are used, in conjunction with the estimates of the surfaces viewed at previous steps, to build a solid 3-D for the viewed part of the object until the current time instant and then used to check the knowledge base to decide which part, we should probably view next. If a new pattern is recognized, it can be used to add more leaves and/or intermediate nodes to the base to help in future recognition processes. If the map (solid model) developed can be no longer improved, either due to the fact that it actually matches something in the knowledge base, or due to the fact that we can't view the required part of any subsequent areas of the scene (which can be indicated by a set of flags passed by the sensing interface) then the recognition process is stopped and the current map is returned as the best possible model.

5 The Controller

The controller module utilizes the decisions made in the map building and decision making module to decide upon the optimal next position for the observer. Based upon the sensor parameters and precision and the robot arm capabilities and reachability, the controller solves the inverse kinematics problem and supplies a configuration vector to the robot arm and also a set of flags to indicate the impossibility of reaching a specific sensing position, if that is what the decision module output implies.

The controller should use the information received from the decision module to decide whether the computed sensing parameters will cause *future* problems for the expected set of viewing positions. The controller produces *quantitative* entities based on *qualitative* entities supplied by the decision module. Zooming, focusing, thresholding level and window sizes decisions should be made by the controller too and passed to the sensing interface along with the position vector and any other information relevant to the sensing process.

6 Conclusions

We have discussed an efficient system for recovering the structural properties of 3-D objects. The system builds models as the sensing process goes on, guided by a knowledge base of possible interpretations that is updated during recognition. We utilize motion primitives for recovering the structure of surfaces. The system could be expanded by allowing for general surfaces and using other shape from X modules, as ways of updating structures, shape from stereo, shading, contours, textures and/or an integrated version of some or all can be used for robustness. Other sensing devices can be used and a scheme could be implemented for combining the information from different sensors and supplying the refined data for consulting the knowledge base efficiently. Subsequently, the controller module will have to control in a distributed way the different sensors so that to obtain maximum information at a low cost of operating the sensors during a specific period of time. Uncertainty in

the sensor data could be studied to allow for the low cost - accurate recognition of different objects using the proposed system.

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* 1^{st} *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.

[13] R. Bajcsy, "Active Perception", *Proceedings of the IEEE*, Vol. 76, No. 8, August 1988.

[14] J. Aloimonos and A. Bandyopadhyay, "Active Vision". In *Proceedings of the* 1^{st} *International Conference on Computer Vision*, 1987.

[15] T. M. Sobh and T. K. Alameldin, Structure and Motion in the "Moving Blocks World". In *Proceedings of the 1990 IASTED International Conference on Adaptive Control and Signal Processing*, October 1990.