3-D Perception and Recognition Under Uncertainty

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 with unknown 3-D shape. In this approach, we build adaptive maps for interpreting the object structure under uncertainty. We decompose the system into four major modules: sensing interface, knowledge base, map building and decision making, and the controller. The *sensing interface* acquires visual data and performs low and intermediate vision tasks for enhancing the acquired image sequences. The *knowledge base* contains different visual primitives and the possible exploratory actions. The *map building and decision making* module 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. The above process is repeated until the map stored in the third module contains a minimal number of three dimensional interpretations that cannot be reduced further. Motion primitives are used as sensing actions. 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 motion and structure estimates. This system is capable of recognizing the object structure efficiently and adaptively.

1 Introduction

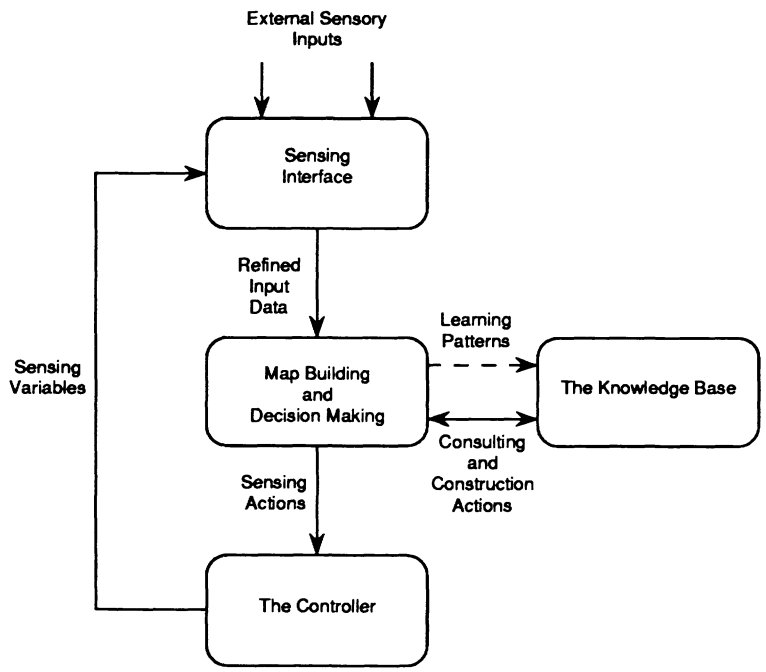
We address the problem of recovering the structural properties of three-dimensional objects. We emphasize in this work the recognition of polyhedra-like objects, our system can be easily extended to general surfaces. The idea of active sensing [13,14] is utilized by the approach used in building the 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

0-8194-0752-6/92/$4.00

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 more movements and that the current interpretation is the best possible one.



The flow diagram of the suggested system

The work examines closely the possibilities for errors 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. We then 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.

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 coordinate 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.`

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

4 Map Building and Decision Making

The map building and decision making module receives the "refined" sensory data from the sensing interface and interacts with the knowledge base in order to construct a solid 3-D model for the viewed object. It uses the 2-D positions, orientations and velocity of the scene to decide what are the features that it should be looking for, or in other words, which unexplored area should it try to view next. In the knowledge base tree, a list of possible 3-D structures are stored with priority vectors attached to each one, the knowledge base uses the data from the decision module and tells it 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 :

[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

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

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

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 :

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

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

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, $\dot{\vec{\Omega}} = \dot{\vec{V}} = \mathbf{0}$. 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 or 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 Uncertainty Modeling

In this section 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. 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.

5.1 Sensor Uncertainties

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 $\not\exists$ error in the 2-D feature recovery mechanism."

5.1.1 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.

5.1.2 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. 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.

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.

5.2 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.

5.2.1 Edge Extraction Uncertainties

Edge extraction strategies and methods to evaluate their performance qualitatively and quantatively have been presented and discussed in the literature. 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.

5.2.2 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.

5.3 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.

5.4 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 a two-frame iterative technique or a 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.

6 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.

7 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.