Measurement and Integration of 3-D Structures By Tracking Edge Lines[1]

James L. Crowley (LIFIA)
Patrick Stelmaszyk (ITMI)
Thomas Skordas (ITMI)
Pierre Puget (ITMI)[2]

LIFIA (IMAG)
I.N.P.Grenoble
46 Ave Félix Viallet
38031 Grenoble, France

I.T.M.I.
Ave des Prés
38240 Meylan, France

1 January 1992

The International Journal of Computer Vision
Vol 8, No. 2, July 1992.

---

[1]This work was sponsored by the Commission of the European Communities DG XIII through Project ESPRIT P940 and BR 3038.

[2]Pierre Puget's current address is D-LETI, CENG, 38000 GRENOBLE, France.

Table of Contents

ABSTRACT

This paper describes techniques for dynamically modeling the 2-D appearance and 3-D geometry of a scene by integrating information from a moving camera. These techniques are illustrated by the design of a system which constructs a geometric description of a scene from the motion of a camera mounted on a robot arm.

A framework for dynamic world modeling is described. The framework presents the fusion of perceptual information as a cyclic process composed of three phases: Predict, Match and Update. A set of mathematical tools are presented for each of these phases. The use of these tools is illustrated by the design of a system for tracking edge lines in image coordinates and inferring the 3-D position from a known camera motion.

The movement of edge-lines in a sequence of images is measured by tracking to maintain an image-plane description of movement. This description is composed of a list of edge-segments represented as a parametric primitive. Each parameter is composed of an estimated value, a temporal derivative, and a covariance matrix. Line segment parameters are updated using a Kalman filter. The correspondence between observed and predicted segments is determined by a nearest-neighbor matching algorithm using distance between parameters normalized by covariance. It is observed that constraining the acceleration of edge-lines between frames permits the use of a very simple matching algorithm, thus yielding a very short cycle time.

Three dimensional structure is computed using the correspondence provided by the 2-D segment tracking process. Fusion of 3-D data from different view points provides an accurate representation of the geometry of objects in the scene. An extended Kalman filter is applied to the inference of the 3-D position and orientation parameters of 2-D segments. This process demonstrates that 2-D tracking provides the information for an inexpensive technique for estimating 3-D shape from motion.

Results from several image sequences taken from a camera mounted on a robot arm are presented to illustrate the reliability and precision of the technique.

1. Introduction

This paper presents a framework for incrementally modeling the contents of a scene by integrating successive observations. This framework has been developed through a sequence of projects during the last decade, including several systems for dynamically modeling the free-space around a mobile robot ultrasonic range sensors [Crowley 85], and a system for mobile robot perception using vertical line stereo matching [Crowley et. al. 90]. Through-out these projects we have developed a methodology for constructing perceptual systems and refined a set of mathematical tools for such systems. This paper presents the resulting framework, reviews the tools for such systems and illustrates these tools with the design of a system for tracking edge lines in a sequence of images and constructing a 3-D model from the image-plane displacement of these edge lines.

Early versions of the 2-D and 3-D modeling systems have been reported in conference papers. The design of the system for 2-D tracking and experimental measures of its reliability were described in [Crowley et al. 88]. The system for incrementally modeling the 3-D structure of objects was described in [Skordas 89] as well as in [Crowley-Stelmaszyk 90]. This paper includes a number of improvements to these systems since their earlier description.

The 2-D and 3-D modeling systems described below permit us to make several conclusions about the role of prediction and estimation in perception:

1) Real time tracking can be based on a very simple "nearest-neighbor" matching algorithm. A first-order predictive system makes possible a very simple matching algorithm.

2) Tracking preserves correspondence. The correspondence between image features established by tracking can be preserved and propagated throughout a system, thus avoiding a more costly matching between 3-D structures. This is particularly useful in real time stereo systems.

3) Once an estimate has been established for a 3-D structure, the position and properties of the structure may be recursively estimated from successive observations without explicit computation of depth, using an extended Kalman filter.

4) A perceptual system can be designed using a layered architecture of predict-match-update cycles (described below). The model maintained within each cycle serves as an observation for cycles in a more abstract coordinate space.

These conclusions were observed as a result of building systems for dynamic modeling applying tools from estimation theory within a framework described in the next section. The work described in this paper is an example of the role of estimation theory in computer vision.

In the following section we review the theoretical foundations and mathematical tools for our work in dynamic world modeling. We present world modeling as a continuously operating cyclic process composed of the phases: predict, match and update. We also describe techniques from estimation theory for each of these processes. In section 3 we apply this framework to the problem of tracking edge segments in an image sequence. In section 4 we show how the tracking process can be exploited to provide the 3-D position of segments by a process of motion stereo. In section 5 we extend the tracking process to 3-D, and show how such 3-D tracking can take the form of an extended Kalman filter. Section 6 provides the results of experimental evaluation, followed by discussions and conclusions in section 7.

2. Theoretical and Mathematical Foundations

The section begins with description of dynamic world modeling as an iterative process of integrating observations into an internal description. Techniques are then presented for each of the phases of the cyclic process.

2.1 A framework for Perception: The Predict, Match and Update Cycle

Dynamic world modeling is a cyclic process in which a model of a scene is elaborated by integrating (or fusing) successive observations. This sequence of observations may come from multiple sensors as well as from a-priori descriptions of the scene.

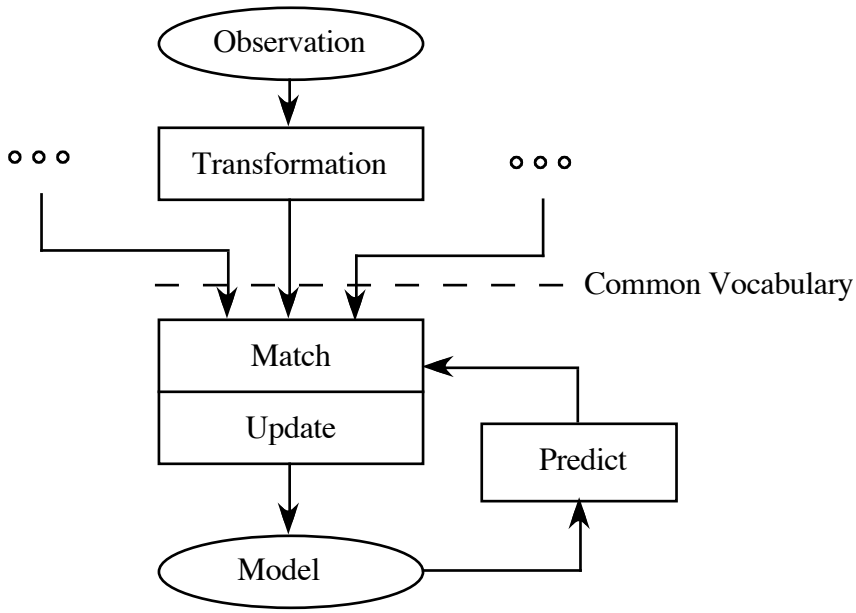

Figure 2.1 A General Framework for Dynamic World Modeling.

A general framework for dynamic world modeling is illustrated in figure 2.1. In this framework, independent observations are "transformed" into a common coordinate space and vocabulary. These

observations are then integrated (fused) into a model (or internal description) by a cyclic process composed of three phases: Predict, Match and Update.

Predict: The current state of the model is used to predict the state of the external world at the time that the next observation is taken.

Match: The transformed observation is brought into correspondence with the predictions. Such matching requires the predictions and observations be transformed to the same coordinate space.

Update: The update phase integrates the observed information with the predicted state of the model to create an updated description of the environment composed of hypotheses.

The update phase serves both to add new information to the model as well as to remove "old" information. During the update phase, information which is no longer within the "focus of attention" of the system, as well as information which has been found transient or erroneous, is removed from the model. This process of "intelligent forgetting" is necessary to prevent the internal model from growing without limits.

This framework can be applied at every level of abstraction within a perceptual system. In particular, such a process may be used in the 2D description of images, in the 3D scene modeling system, and in the symbolic scene description process.

2.2 The Use of Estimation Theory for Fusion and Tracking in Vision

Recent advances in sensor fusion within the vision community have largely entailed the rediscovery and adaptation of techniques from estimation theory. Our work on 2-D tracking was partly inspired by Gennerey, who has shown that the measurement of the motion of points in an image sequence could be based on a Kalman filter [Gennerey 82]. We adopted a similar approach for incremental construction of world model of a mobile robot using a rotating ultrasonic sensor [Crowley 85]. That work was generalized [Crowley 84] to present fusion as a cyclic process of combining information from logical sensors. The importance of an explicit model of uncertainty was recognized, but the techniques were for the most part "ad-hoc". In the same period, Herman and Kanade [Herman-Kanade 86] combined passive stereo imagery from an aerial sensor to design a system for "incremental modeling". Driven by the needs of perception for mobile robotics, Brooks [Brooks 85] and Chatila [Chatila 85] also published ad-hoc techniques for manipulation of uncertainty.

In 1986, a pre-publication of a paper by Smith and Cheeseman was very widely circulated [Smith-Cheeseman 87]. In this paper, the authors argue for the use of Bayesian estimation theory in vision and robotics. An optimal combination function was derived and shown to be equivalent to a simple form of Kalman filter. At the same period, Durrant Whyte completed a thesis [Durrant-Whyte 87] on the manipulation of uncertainty in robotics and perception. This thesis presents derivations of

techniques for manipulating and integrating sensor information which are extensions from estimation theory. Faugeras and Ayache [Faugeras et. al. 86] contributed an adaptation of this theory to stereo and calibration. Matthies et. al. [Matthies et. al. 87] have demonstrated recovery of depth from lateral displacement of points in a dense image sequence using a Kalman filter by tracking gray level values. From 1987, a paradigm shift occurred in the vision community, with techniques from estimation theory being increasingly adapted.

While most researchers applying estimation theory to perception can cite one of the references [Smith Cheeseman 87], [Durrant Whyte 87] or [Faugeras et. al. 86] for their inspiration, the actual techniques were well known to some other scientific communities, in particular the community of control theory. The starting point for estimation theory is commonly thought to be the independent developments of Kolmogorov [Kolmogorov 40] and Weiner [Weiner 49]. (Weiner's work during the 1940's concerned the estimation of flight paths and could only be published after the war.) Bucy [Bucy 59] showed that the method of calculating the optimal filter parameters by differential equation could also be applied to non-stationary processes. Kalman [Kalman 60] published a recursive algorithm in the form of difference equations for recursive optimal estimation of linear systems. With time, it has been shown that these optimal estimation methods are closely related to Bayesian estimation, maximum likelihood methods, and least squares methods. These relationships are developed in textbooks by Bucy and Joseph [Bucy-Joseph 68], Jazwinski [Jazwinski 70], and Melsa and Sage [Melsa-Sage 71]. These relations are reviewed in a recent paper by Brown [Brown et al. 89], as well as in a book by Brammer and Siffling [Brammer-Siffling 89].

In the remaining sections of this chapter we show how techniques from estimation theory may be applied to the predict-match-update cycle, in the case where the model is composed of parametric primitives.

2.3 Representation: Parametric Primitives

A dynamic world model, M(t), is a list of primitives which describe the "state" of a part of the world at an instant in time t.

Model: $M(t) \equiv \{ P_1(t), P_2(t), ..., P_m(t)\}$

A model may also include "grouping" primitives which assert relations between lower level primitives. Examples of such groupings include connectivity, co-parallelism, junctions and symmetries [Horaud et al. 90]. Such groupings constitute symbolic properties and are not addressed in this paper.

Each primitive P(t) in the world model, describes a local part of the world as a conjunction of estimated properties, $\hat{X}(t)$, plus a unique ID and a confidence factor, CF(t).

Primitive : $P(t) \equiv \{ID, \hat{X}(t), CF(t)\}$

The ID of a primitive acts as a name by which the primitive may be referred. The confidence factor, CF(t), permits the system to control the contents of the model. Newly observed segments enter the model with a low confidence. Successive observations permit the confidence to increase, where as if the segment is not observed in the succeeding cycles, it is considered as noise and removed from the model. Once the system has become confident in a segment, the confidence factor permits a segment to remain in existence for several cycles, even if it is obscured from observations. Experiments have lead us to use a simple set of confidence "states" represented by integers. The number of confidence states depends on the application of the system.

The properties, $\hat{X}(t)$, may be either a numeric value or a symbolic label from a finite class of symbols. In this paper we confine our discussion to numeric properties, for which we may employ the mathematical techniques from estimation theory.

A primitive represents the local state of a part of the world as an association of a set of N properties, represented by a vector , X(t).

$$X(t) = \{ x_1(t), x_2(t) \ldots x_n(t)\}.$$

The actual state of the external world, X(t), is estimated by an observation process which is assumed to be corrupted by random noise, N(t). The world state, X(t), is not directly knowable, and so our estimate is taken to be an expected value.

$$\hat{X}(t) = E\{X(t) + N(t)\}$$

At each cycle, the modeling system produces an estimate $\hat{X}(t)$ by combining a prediction, $Y^*(t)$, with an observation Y(t). The difference between the predicted vector $Y^*(t)$ and the observed vector Y(t) provides the basis for updating the estimate $\hat{X}(t)$, as described below.

In order for the modeling process to operate, both the primitive, $\hat{X}(t)$ and the observation, Y(t) must be accompanied by an estimate of their uncertainty. This uncertainty may be seen as an expected deviation between the estimated vector, $\hat{X}(t)$, and the true vector, X(t). Such an expected deviation is approximated as a covariance matrix $\mathbf{C}^{\wedge}(t)$[1] which represents the expected difference between the estimate and the actual world state.

---

[1]In the control theory literature, it is traditional to use P(t) for this covariance. Unfortunately, this conflicts with our use of P(t) as a point in the image or scene, as well as for a primitive in the world model.

$$\hat{\mathbf{C}}(t) \equiv E\{[X(t) - \hat{X}(t)]\,[X(t) - \hat{X}(t)]^T\}$$

Modeling this precision as a covariance makes available a number of mathematical tools for matching and integrating observations. The uncertainty estimate is based on a model of the errors which corrupt the prediction and observation processes. Estimating these errors is both difficult and essential to the function of our system.

The uncertainty estimate provides two crucial roles in our system:

1) It provides the tolerance bounds for matching observations to predictions, and

2) It provides the relative strength of prediction and observation when calculating a new estimate.

Because $\hat{\mathbf{C}}(t)$ determines the tolerance for matching, system performance will degrade rapidly if we under-estimate $\hat{\mathbf{C}}(t)$. On the other hand, overestimating $\hat{\mathbf{C}}(t)$ merely increases the computing time for finding a match.

2.4 Prediction: Discrete State Transition Equations

The prediction phase of the modeling process projects the estimated vector $\hat{X}(t)$ forward in time to a predicted value, $X^*(t+\Delta T)$. This phase also projects the estimated uncertainty $\hat{\mathbf{C}}(t)$ forward to a predicted uncertainty $\mathbf{C}^*(t+\Delta T)$. Such projection requires estimates of the temporal derivatives for the properties in $\hat{X}(t)$, as well as estimates of the covariances between the properties and their derivatives. These estimated derivatives can be included as properties in the vector $\hat{X}(t)$.

In the following, we will describe the case of a first order prediction; that is, only the first temporal derivative is estimated. Higher order predictions follow directly by estimating additional derivatives. We will illustrate the techniques for a primitive composed of two properties, $x_1(t)$ and $x_2(t)$. We employ a continuous time variable t to mark the fact that the prediction and estimation may be computed for a time interval, ΔT, which is not necessarily constant.

Temporal derivatives of a property are represented as additional components of the vector X(t). Thus, if a system estimates N properties, the vector X(t) is composed of 2N components: the N properties and N first temporal derivatives. It is not necessary that the observation vector, Y(t), contain the derivatives of the properties to be estimated. One of the surprising aspects of the Kalman filter is that we can iteratively estimate the derivatives of a property using only observations of its value. Furthermore, because these estimates are developed by integration, they are more immune to noise than instantaneous derivatives calculated by a simple difference.

Consider a property, $\hat{x}(t)$, of the vector $\hat{X}(t)$, having variance $\hat{\sigma}_x^2$. A first order prediction of the value

$x^*(t+\Delta T)$ requires an estimate of the first temporal derivatives, $\hat{x}'(t)$.

$$\hat{x}'(t) \equiv \frac{\partial \hat{x}(t)}{\partial t}$$

The evolution of X(t) can be predicted by a Taylor series expansion. To apply a first order prediction, all of the higher order terms are grouped into an unknown random vector V(t), approximated by an estimate, $\hat{V}(t)$. The term $\hat{V}(t)$ models the effects of both higher order derivatives and other unpredicted phenomena. V(t) is defined to have a variance (or energy) of **Q**(t).

$$\mathbf{Q}(t) = E\{V(t)\, V(t)^T\}$$

When V(t) is unknown, it is assumed to have a

expected value of zero. However, in some situations it is possible to estimate the perturbation from knowledge of commands by an associated control system. In this case, an estimated perturbation vector $\hat{V}(t)$ and its uncertainty, $\hat{\mathbf{Q}}(t)$ may be included in the prediction equations.

Thus each term is predicted by:

$$x^{*}(t+\Delta T) = \hat{x}(t) + \frac{\partial \hat{x}(t)}{\partial t}\Delta T + \hat{v}(t)$$

Let us consider a vector, $\hat{X}(t)$, composed of the properties $\hat{x}_1(t)$ and $\hat{x}_2(t)$ and their derivatives.

$$\hat{X}(t) \equiv \begin{bmatrix} \hat{x}_1(t) \\ \hat{x}_1'(t) \\ \hat{x}_2(t) \\ \hat{x}_2'(t) \end{bmatrix}$$

The time increment ΔT is included in the transition matrix, φ.

$$\varphi \equiv \begin{bmatrix} 1 & \Delta T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta T \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This gives the prediction equation in algebraic form as:

$$X^{*}(t+\Delta T) := \varphi\, \hat{X}(t) + \hat{V}(t) \qquad (2.1)$$

Predicting the uncertainty of $X^{*}(t+\Delta T)$ requires an estimate of the covariance between each property, $\hat{x}(t)$ and its derivative.

An estimate of this uncertainty, $\hat{\mathbf{Q}}_x(t)$, permits us to account for the effects of unmodeled derivatives when determining matching tolerances. This gives the second prediction equation:

$$\mathbf{C}_x^{*}(t+\Delta T) := \varphi\, \hat{\mathbf{C}}_x(t)\, \varphi^{T} + \hat{\mathbf{Q}}_x(t) \qquad (2.2)$$

2.5 Matching Observation to Prediction: The Mahalanobis Distance

The predict-match-update cycle presented in this paper simplifies the matching problem by applying the constraint of temporal continuity. That is, it is assumed that during the period ΔT between observations, the deviation between the predicted values and the observed values of the estimated primitives is small enough to permit a trivial "nearest neighbor" matching.

Let us define a matrix ${}^{Y}\mathbf{H}_X$ which transforms the coordinate space of the estimated state, X(t), to the

coordinate space of the observation.

$$Y(t) = {}^{Y}\mathbf{H}_{X}\, X(t).$$

The matrix ${}^{Y}\mathbf{H}_{X}$ constitutes a "model" of the sensing process[1] which predicts an observation, Y(t) given knowledge of the properties X(t). Estimating ${}^{Y}\mathbf{H}_{X}$ is a crucial aspect of designing a world modeling system. The model of the observation process, ${}^{Y}\mathbf{H}_{X}$, should not be assumed to be perfect. In machine vision, the observation process is typically perturbed by photo-optical, optical and electronic effects. Let us define this perturbation as W(t). In most cases, W(t) is unknown, leading us to estimate

$$\hat{W}(t) \equiv E\{W(t)\} = 0$$

and

$$\hat{\mathbf{C}}_{y}(t) \equiv E\{\, W(t)\, W(t)^{T}\}$$

To illustrate this process, suppose that we can observe the current value of the parameters but not their derivatives. In this case ${}^{Y}\mathbf{H}_{X}$, can be used to yield a vector removing the derivatives from the predicted properties. The two-property, first order vector used in the example from the previous section would give a prediction ${}^{Y}\mathbf{H}_{X}$ of:

$$\begin{bmatrix} y_1^*(t) \\ y_2^*(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1^*(t) \\ x_1^{*\prime}(t) \\ x_2^*(t) \\ x_2^{*\prime}(t) \end{bmatrix}$$

Of course ${}^{Y}\mathbf{H}_{X}$ may represent any linear transformation. In the case where the estimated state and the observation are related by a transformation, F(X), which is not linear, ${}^{Y}\mathbf{H}_{X}$ is approximated by the first derivative, or Jacobian, of the transformation, ${}^{Y}\mathbf{J}_{X}$.

$${}^{Y}\mathbf{J}_{X} = \frac{\partial F(X)}{\partial X}$$

Let us assume a predicted model $M^*(t)$ composed of a list of primitives, $P^*_n(t)$, each containing a parameter vector, X*(t), and an observed model O(t) composed of a list of observed primitives, $P_m(t)$, each containing the parameters Y(t). The match phase determines the most likely association of observed and predicted primitives based on the similarity between the predicted and observed properties. The mathematical measure for such similarity is to determine the difference of the

---

[1] ${}^{Y}\mathbf{H}_{X}$ is sometimes known as the "Sensor Model". This use of the word "model" by the estimation theory community creates an unfortunate conflict of terms with the vision community.

properties, normalized by their covariance. This distance, normalized by covariance, is a quadratic form known as the squared Mahalanobis distance.

The predicted parameter vector is given by:

$$Y_n^* := {}^{Y}\mathbf{H}_X \, X_n^*$$

with covariance

$$\mathbf{C}_{yn}^* := {}^{Y}\mathbf{H}_X \; \mathbf{C}_{xn}^* \; {}^{Y}\mathbf{H}_X^T$$

The observed properties are Y_m with covariance $\mathbf{C}_{ym}$. The squared Mahalanobis distance between the predicted and observed properties is given by:

$$D_{nm}^2 = \frac{1}{2} \{ (Y_n^* - Y_m)^T (\mathbf{C}_{yn}^* + \mathbf{C}_{ym})^{-1} (Y_n^* - Y_m) \}$$

For the case where a single scalar property is compared, this quadratic form simplifies to:

$$D_{nm}^2 = \frac{1}{2} \frac{(y_n^* - y_m)^2}{(\sigma_{y_n}^{*2} + \sigma_{y_m}^2)}$$

In the predict-match-update cycles described below, matching involves minimizing the normalized distance between predicted and observed properties or verifying that the distance falls within a certain number of standard deviations.

2.6 Updating: The Kalman Filter Update Equations

Having determined that an observation corresponds to a prediction, the properties of the model can be updated. The extended Kalman filter permits us to estimate a set of properties and their derivatives, $\hat{X}_n(t)$, from the association of a predicted set of properties, $Y_n^*(t)$, with an observed set of properties, $Y_m(t)$. It equally provides an estimate for the precision of the properties and their derivatives. This estimate is equivalent to a recursive least squares estimate for $X_n(t)$. The estimate and its precision will converge to a false value if the observation and the estimate are not independent.

The crucial element of the Kalman filter is a weighting matrix known as the Kalman Gain, $\mathbf{K}(t)$. The Kalman Gain may be defined using the prediction uncertainty $\mathbf{C}_y^{**}(t)$.

$$\mathbf{K}(t) := \mathbf{C}_x^*(t) \; {}^{Y}\mathbf{H}_X^T \, [\, \mathbf{C}_y^*(t) + \mathbf{C}_y(t)]^{-1} \qquad (2.3)$$

The Kalman gain provides a relative waiting between the prediction and observation, based on their relative uncertainties. The Kalman gain permits us to update the estimated set of properties and their derivatives from the difference between the predicted and observed properties:

$$\hat{X}(t) := X^*(t) + \mathbf{K}(t)\,[Y(t) - Y^*(t)] \tag{2.4}$$

The precision of the estimate is determined by:

$$\hat{\mathbf{C}}(t) := \mathbf{C}^*(t) - \mathbf{K}(t)\,{}^{Y}\mathbf{H}_X\,\mathbf{C}^*(t) \tag{2.5}$$

Equations (2.1) through (2.5) constitute the 5 equations of the Kalman Filter. In the following three sections, we apply these techniques to the predict-match-update cycles for tracking edge lines in the image plane and for inferring the 3-D scene coordinates for image features from the image plane motion.

3. Measuring Image Structure by Tracking Edge Segments

Correspondence of edges in a dense temporal sequence of images can be maintained by a very simple tracking process based on the predict-match-update cycle. Such a process has been described in [Crowley et. al. 88]. This tracking process is well debugged and has been used in a number of our projects. Real time hardware, capable of tracking up to 256 segments at 10 cycles per second, has recently been constructed using this algorithm [Chehikian 90].

In this section we describe the tracking process. We present a representation which permits line segment motions to be modeled as an independent set of four parameters. This is followed by sections which describe the processes for prediction of the model state, for matching observed segments to model segments, and for updating the flow model.

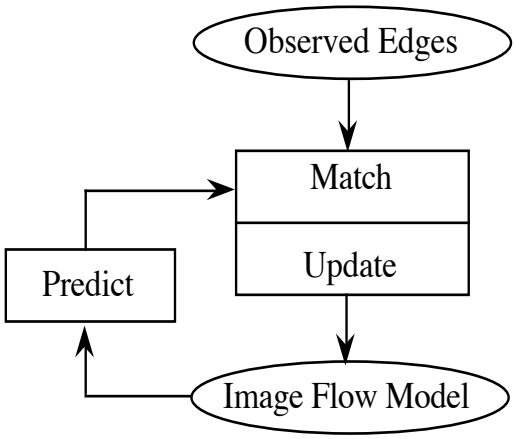

Figure 3.1. The Image Flow Modeling Process.

3.1 A Dynamic Model for Image Structure

The edge segment tracking process is illustrated by figure 3.1. For each time, t, that an image is observed, the state for an image flow model is predicted. The edge segments which are observed in the image are then matched to the predicted flow model. Matches between observed edge segments and the predicted flow model are used to update the flow model.

Newly observed edge-line segments are extracted from each image comprise the "observation" which is used to update the image flow model. As the observed edge segments for each new image are made available, the segments in the flow model "find their match" among the observed edge lines. The dynamic and geometric attributes of each model segment are updated using the corresponding observed segment. The simplicity of the matching and updating process is made possible by a parametric representation for edge lines which separates position information into its perpendicular and tangential components. This representation permits tracking to be performed as a set of four 1-D Kalman filters.

3.2 A Parametric Representation for Line Segments

A minimal representation for a line segment requires four parameters. The classic representation is the Cartesian coordinates of the two end-points. An alternative representation is to express a line segment in terms of a center point, half-length and orientation. This second representation provides advantages for matching and for flow estimation. None-the-less, the uncertainty of the x and y components of the center point are strongly correlated, and a proper estimation system requires a coupling these parameters, yielding a 4 by 4 covariance matrix which must be inverted for both matching and updating.

If we consider the effect of image noise on the process of extracting edge lines we can make a number of observations. In particular:

1) The length of a line segment is often unreliable, due to random effects which break edge lines into smaller segments. Thus, along the line direction, the position of a segment is unreliable and has a large uncertainty.

2) The perpendicular position of a line segment may be measured with precision. This value is often a parameter of the segment extraction process, and is usually on the order of a pixel.

3) The orientation of a line segment has a precision which is proportional to the ratio of the perpendicular precision and the length. Longer segments have a more precise orientation.

These observations show that the error statistics on the midpoint are strongly correlated with the edge-segment orientation, suggesting the use of a representation in which perpendicular position and tangential position of a line segment are made explicit and separate. Such a suggestion is reinforced by the well know "aperture effect" [Hildreth 82] in measuring motion. That is, for edges, it is generally only possible to directly measure perpendicular displacement. Tangential displacement is ambiguous because of the one-dimensional nature of edge lines. These considerations suggested a representation for edge lines in which the Cartesian expression of the midpoint, (x, y) is augmented by an expression in terms of the parameters (c, d) which represent the perpendicular and tangential distance from the origin.

$$d = x\,\mathrm{Cos}(\theta) - y\,\mathrm{Sin}(\theta)$$
$$c = x\,\mathrm{Sin}(\theta) + y\,\mathrm{Cos}(\theta)$$

The advantage of this representation is that it allows us to represent each parameter and its time derivative as a scalar estimate and a scalar variance. The disadvantage is a "lever-arm" effect. For segments which are not near the origin, a small error in θ can provoke a large variation c and d. The result can be a considerable number of correspondences incorrectly rejected during matching.

After considerable experimentation with a 4 parameter segment represented by (c, d, θ, h) we ultimately settled on a redundant representation composed of the 5 independently estimated parameters: (x_c, y_c, θ, h, c). The extra parameter, c, provides the perpendicular tolerance for matching, as well as a direct estimated of the perpendicular velocity of the segment.

Thus the representation for segments is composed of the following parameters:

x_c	The horizontal position of the center point.
y_c	The vertical position of the center point.
θ	The orientation of the segment.
h	The half-length of the segment.
c	The perpendicular distance of the segment from the origin.

In addition, the segment end-points P_1 and P_2 and the line equation coefficients, a and b, are maintained for each segment. The line coefficients are computed from the orientation using the formulas:

$$a = \sin(\theta)$$
$$b = -\cos(\theta).$$

3.3 Representation for Line Segments in the 2-D Model

In the flow model it is necessary to estimate the time derivative of each parameter along with its precision. A direct approach would lead us to a Kalman filter which estimates each line segment with an 8 dimensional vector (4 parameters and 4 derivatives) and an an 8 by 8 covariance matrix. We have

greatly simplified the computational load by separating the estimation into 5 independent estimations, for each of the properties in the vector S.

$$S \equiv \{x_c, y_c, \theta, h, c\}$$

For each parameter $x \in S$ we maintain a first order estimate vector $\hat{X}(t)$.

$$\hat{X}(t) \equiv \begin{bmatrix} \hat{x}(t) \\ \hat{x}'(t) \end{bmatrix}$$

In addition, for each property $x \in S$, it is necessary to represent its variance, the variance of the temporal derivative, and a covariance between the estimate and its temporal derivative

$$\mathbf{C}_{\hat{x}}(t) \equiv \begin{bmatrix} \hat{\sigma}_x^2 & \hat{\sigma}_{x'x} \\ \hat{\sigma}_{xx'} & \hat{\sigma}_{x'}^2 \end{bmatrix}$$

In addition to the attribute vector, S, each line segment contains a set of redundant parameters composed of its center point expressed in image coordinates, P, as well as its end points, P_1 and P_2. These redundant parameters are recomputed from the parameters in S during each cycle. Each segment also contains a confidence factor, CF, which expresses the confidence in the existence of a token in the model, and a unique identity, ID.

The confidence factor is expressed as an integer state, from the set {1, 2, 3, 4, 5}. CF = 1 represents a model segment which is very uncertain. CF = 5 represents a segment which has been reliably tracked for at least 3 frames. When the token for a segment is first created, it is added to the model with a confidence factor of CF = 3. During each update cycle, if a correspondence is found for a token, the CF is incremented by 1, up to a maximum value of 5. If no correspondence is found, the CF is decremented by 1, to a minimum value of 0. If the CF passes to 0, the token is removed from the model. The use of a confidence factor gives the flow model a degree of immunity to the temporary loss of edge lines in as many as four successive observation images.

The ID is a unique index which makes it possible to identify a segment at different times. In the following section, we describe a technique based on periodic "snapshots" of the flow model. The correspondence of segments in these "snapshots" is directly available from the segment ID. From the difference in position of a segment in two snapshots, we can calculate a 3D segment. The 3D segment conserves this ID, which makes it possible to directly associate a reconstructed 3D segment with a segment in the 3D model, without 3-D correspondence matching.

In summary, a line segment is represented in the model with the following parameters:

Horizontal Position: $\{ x_c, x_c', \sigma_x^2, \sigma_{xx'}, \sigma_{x'}^2 \}$

Tangential Position: $\{ y_c, y_c', \sigma_y^2, \sigma_{yy'}, \sigma_{y'}^2 \}$

Perpendicular Position: $\{ c, c', \sigma_c^2, \sigma_{cc'}, \sigma_{c'}^2 \}$

Orientation: $\{ \theta, \theta', \sigma_\theta^2, \sigma_{\theta\theta'}, \sigma_{\theta'}^2 \}$

Half Length: $\{ h, h', \sigma_h^2, \sigma_{hh'}, \sigma_{h'}^2 \}$

Confidence Factor: CF

Identity: ID

Line coefficients: $(a = \sin(\theta), b = -\cos(\theta))$.

End-points $\{(x_1, y_1), (x_2, y_2)\}$

3.4 Predicting the State of Line Segments

Given an estimate set of parameters, $\hat{S}(t)$, the prediction follows directly from the prediction equations (1) and (2) developed above. For each $x \in S \equiv \{x_c, y_c, \theta, h, c\}$:

$$X^*(t+\Delta T) := \varphi \hat{X}(t) + \hat{V}(t)$$

and

$$\mathbf{C}_x^*(t+\Delta T) := \varphi \hat{\mathbf{C}}_x(t) \varphi^T + \hat{\mathbf{Q}}_x(t) \Delta T^4$$

where

$$\varphi \equiv \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix}$$

When the camera velocity is constant, the term $\hat{V}(t)$ is set to zero for the five parameters. When the camera undergoes an acceleration, a compensation term is added based on the movement and the estimated depth to the 3-D structure which corresponds to the structure in the 3-D model described below.

This prediction assumes that there is no unmodeled acceleration between update cycles. Indeed, accelerations exists and are a real source of uncertainty. To account for the possibility of accelerations, the uncertainty of each attribute is increased by a constant term, σ_{acc}^2, multiplied by the time interval to the fourth power. For each of the parameters, this term is included in the term $\hat{\mathbf{Q}}_x(t)$:

$$\hat{\mathbf{Q}}_x(t) \equiv \begin{bmatrix} \sigma_{acc}^2 & 0 \\ 0 & 0 \end{bmatrix}$$

3.5 Matching Predictions to Observations

When a flow model is matched to observations with sufficiently small time delay, matching can be

based on a nearest neighbor association between the predicted and observed tokens. This approach is self-reinforcing, in that if matching is based only on difference between attributes, it can be done with a very simple, and fast process. The simplicity of the matching is of fundamental importance, precisely because the cost of matching dominates the tracking process.

At the beginning of the matching process, the state of each token in the flow model is predicted based on the time delay, Δt, since the last model update. Each model token then scans the observed segment to find a best match. The state vector of each token is then updated with its match, as described in this section.

Because of the uncertainty of the mid-point position is strongly correlated with θ, matching can not be based on a simple comparison of attributes. We have developed a matching test in which each model token searches the list of observed tokens for its best match by testing for similar orientation, alignment, and overlap. If any test is failed, matching proceeds to the next observed token. The tests for orientation and co-linearity are made by testing to see if the difference in attributes is less than a threshold number of standard deviations. For overlap, the half length of the segments is used as a tolerance region.

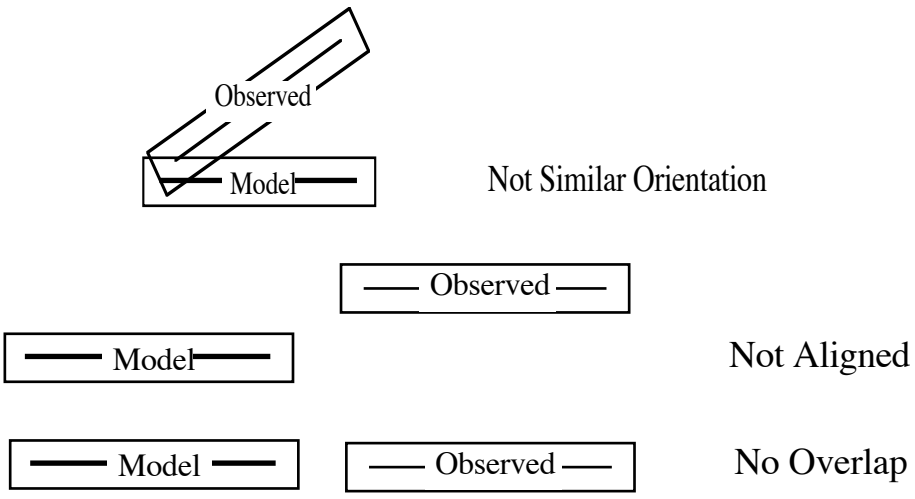

Figure 3.2 To match an observation, a predicted model segment must have a similar orientation, be co-linear and must overlap the observed segment. These figures illustrate these three criteria.

For model segment $Y_m^* \equiv \{ x_m, y_m, c_m, \theta_m, h_m, a_m, b_m\}$, and observation segment $Y_o \equiv \{x_o, y_o, c_o, \theta_o, h_o, a_o, b_o\}$, the test for similarity of orientation is:

$$(\theta_m - \theta_o)^2 \leq 2 (\sigma_{\theta m}^2 + \sigma_{\theta o}^2).$$

If the test is true, then the observed segment is tested for co-linearity with the model token by comparing the distance from the midpoint of each segment to the line of the other segment:

$$(a_m x_o + b_m y_o + c_m)^2 < \sigma_{cm}^2) \text{ AND}$$
$$(a_o x_m + b_o y_m + c_o)^2 < \sigma_{co}^2)) \text{ THEN}$$

If the observed segment passes this test then the segments are tested to see if they overlap. The test for overlap is made using the half length of the segments as an uncertainty along the line segment. Thus the test compares the sum of the half lengths to the difference between mid-points.

$$(x_m - x_o)^2 + (y_m - y_o)^2 \leq (h_m + h_o)^2$$

If an observed segment passes all three tests, then a similarity to the model segment is calculated, using the sum of the differences normalized by the standard deviations for orientation and length.

$$\text{Sim}(Y_m^*, Y_o) = \frac{(\theta_o - \theta_m)^2}{\sigma_{\theta o}^2 + \sigma_{\theta m}^2} + \frac{(x_m - x_o)^2 + (y_m - y_o)^2}{(h_m + h_o)^2}$$
$$+ \frac{(a_m x_o + b_m y_o + c_m)^2}{\sigma_{cm}^2} + \frac{(a_o x_m + b_o y_m + c_o)^2}{\sigma_{co}^2}$$

This similarity measure is a form of Mahalanobis distance, that is distance, normalized by variance. The observed token with the smallest value of the similarity measure is selected as matching the model token, and is used to update the token state vector and uncertainties.

3.6 Updating a Model Segment with an Observation

The flow model is updated by updating the attributes and confidence factor of each token for which a match was found, and reducing the confidence factor for tokens for which no match was found. This process is described in this section.

Given an observed edge line which matches a model token, the update process is based on equations (2.3), (2.4) and (2.5) above. For each $x \in S \equiv \{x_c, y_c, \theta, h, c\}$ the transformation from the predicted vector and its derivative to the coordinates of the observation vector is the row vector

$$^{Y}\mathbf{H}_X \equiv [\,1 \;\; 0\,]$$

For each $x \in S \equiv \{x_c, y_c, \theta, h, c\}$ we compute a Kalman Gain vector as:

$$K(t) := \mathbf{C}_x^{**}(t)\, {}^{Y}\mathbf{H}_X^T \left[\, {}^{Y}\mathbf{H}_X\, \mathbf{C}_x^{**}(t)\, {}^{Y}\mathbf{H}_X^T + \mathbf{C}_y^{*}(t)\right]^{-1}$$

For each parameter $x \in S = \{c, d, \theta, h\}$, a new estimated vector is obtained by

$$\hat{X}(t) := X^*(t) + K(t) [Y(t) - {}^{Y}\mathbf{H}_X X^*(t)]$$

A new estimated uncertainty is obtained from

$$\hat{\mathbf{C}}(t) := \mathbf{C}^*(t) - K(t) {}^{Y}\mathbf{H}_X \mathbf{C}^*(t)$$

If a match is found for a model segment, the confidence factor is incremented, to a maximum value of 5. If no match is found for a segment, the estimated parameters are set as the predicted parameters, and the confidence factor is decremented by 1 to a minimum of 0. If the value of CF descends to 0, then the token is removed from the model.

3.7 Adding New Observations to the Model

When an observed segment is determined to be the best match for a token, it is marked as matched. After all of the model tokens have been updated, and the model tokens with CF = 0 removed from the model, each unmatched observed segment is added to the model using default parameters.

New tokens are created using the observed value for the parameter estimates, and a temporal derivative of zero. The parameter covariances are set to large default values. The token confidence factor is set to 1. Thus in the next cycle, a new token has a significant possibility of finding a false match. False matches are rapidly eliminated, however, as they lead to incorrect predictions for subsequent cycles and a subsequent lack of matches. Because an observed token can be used to update more than one model token, such temporary spurious model tokens do not damage the parameters of other tokens in the flow model.

4. Estimating the 3-D Structure from 2-D Tracking

Tracking edge lines from a moving camera provides a number of useful capabilities. One property is that tracking suppresses image noise. A second property is that tracking preserves correspondence. Because the segment ID is furnished by 2-D tracking, this correspondence matching is reduced to a simple verification. Tracking removes the necessity for 3-D matching thus simplifying the composite modeling process.

In the following two sections we show how multiple observations of a scene, kept in correspondence by tracking, can be used to reconstruct and maintain a 3-D composite model from the motion of a camera mounted in the gripper of a robot manipulator. In this section we present the boot-strap phase, in which newly observed segments are projected to 3-D scene coordinates using standard stereo

equations. In the next section we describe how the Kalman Filter is used to refine the estimated position of such 3-D segments. By this form of motion stereo we are able to recover three dimensional edge-line segments which are integrated in a composite local model.

The system is illustrated in figure 4.1. The 3-D composite model is updated whenever the accumulated camera motion exceeds a threshold distance (approximately 5 cm in our experiments). The update phase for 2-D modeling is considerably faster than that of 3-D modeling. Two forms of 3-D inference are used in this system: boot-strap initialization and steady state maintenance. Segments from the 2-D flow model are treated as an observation and compared to the 2-D projection (prediction) of segments from the 3-D model. Correspondence is determined directly from the segment IDs and is verified by comparison of the orientation, alignment and overlap, as described above.

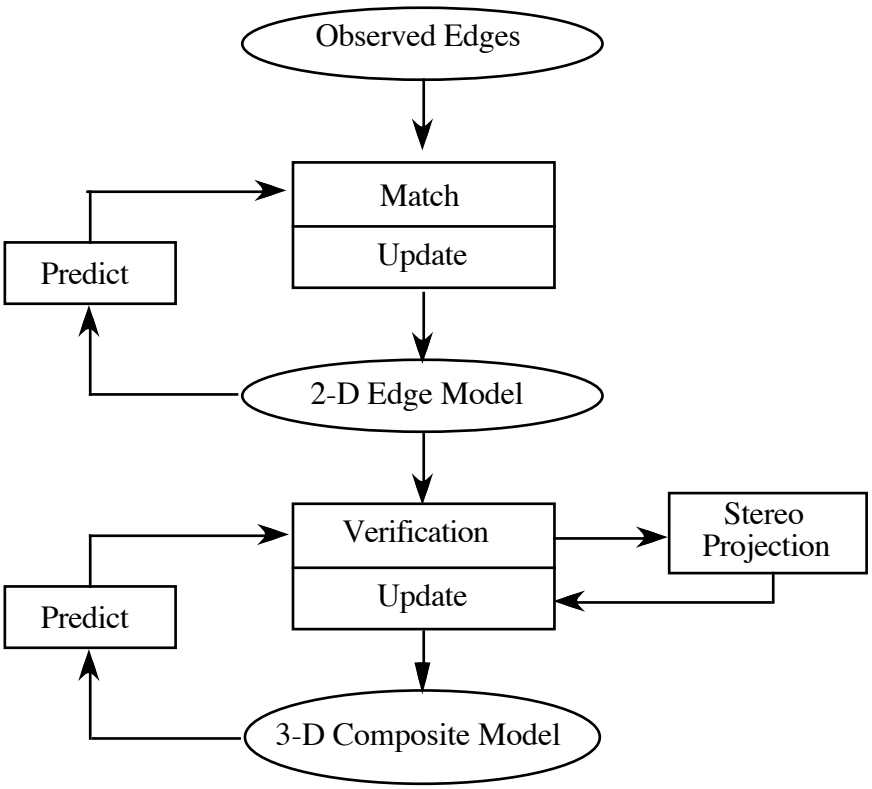

Figure 4.1 System for maintaining a 3-D Composite model of a scene from the motion of a camera on a robot arm.

Initialization: Observed segments for which no segment exists in the 3-D model, are passed to the initialization phase to be used to create a new 3-D segment for the model. The 3-D position of edge segments is initialized using a classical stereo solution. "Snapshots" of the flow model are saved a each update of the 3-D Model. Knowledge of the camera location at the time of each update permits us to compute the three dimensional locations for corresponding edge lines. These observed 3-D edge lines are then used to update the 3-D composite model. This process is described in section 4.3 below.

Maintenance: When a 3-D model segment exists for an observed segment, the parameters of the

observed segment are refined using an extended Kalman filter. This process is described in section 5 below.

The 3-D composite model uses a 3-D edge line primitive described in [Crowley 86]. These 3-D edge-line primitives contain an explicit estimate of the uncertainty of the 3-D positions. As with the flow model, a Kalman filter update equation is used to refine the estimated position and uncertainty. The resulting 3-D edge line segments are more reliable and more precise than individual observations of 3-D segments obtained from pairs of "snapshots" of the flow model.

In the following section we described the notation and set of coordinate systems that are required to reconstruct the three dimensional form of the scene from a camera mounted on a robot arm. This is followed by a description of the representation which is used to represent 3-D edge line segments. We then present the techniques for inferring the 3-D position of segments. Finally we describe the update process for the 3-D composite model.

4.1 Coordinate Systems and Notation

The derivation of the equations for recovery of 3-D structure from the movement of a camera on a robot arm requires that we define a set of coordinate systems, as well as homogeneous coordinate transformations between these coordinate systems.

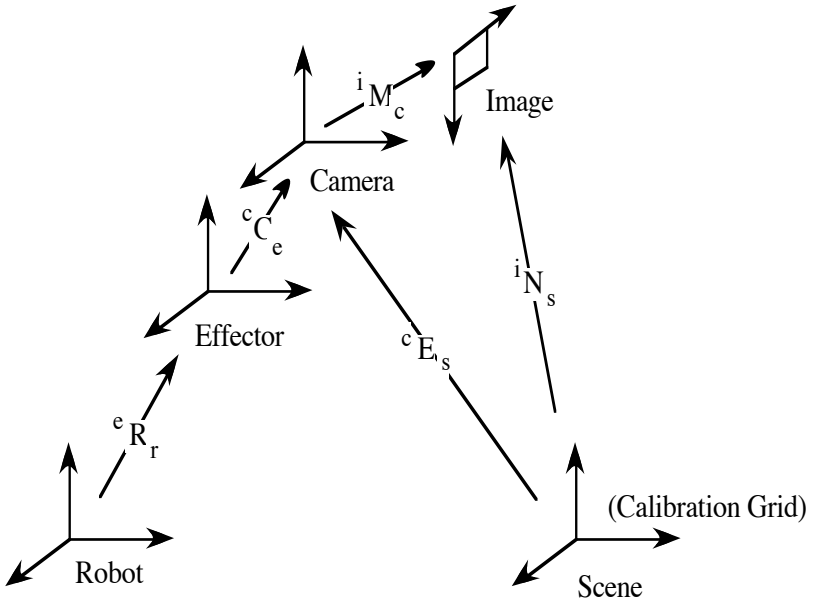

Figure 4.2 Coordinate Systems and Transformations

The transformations with which we are concerned are illustrated in figure 4.2. They are:

- ${}^{e}\mathbf{R}_{r}$ The transformation from robot to effector coordinates
- ${}^{c}\mathbf{C}_{e}$ The transformation from effector to camera coordinates

${}^{c}\mathbf{E}_{s}$ The transformation from scene to camera coordinates

${}^{i}\mathbf{M}_{c}$ The transformation from camera to image coordinates

${}^{i}\mathbf{N}_{s}$ The transformation from scene to image coordinates

Notice that the transformations ${}^{i}\mathbf{M}_{c}$ and ${}^{i}\mathbf{N}_{s}$ include the projective transformation and thus have no inverse.

Constructing an estimate of the position and uncertainty of 3-D contours in scene coordinates requires a model of the image formation process. Such a model is expressed as a composition of the intrinsic and extrinsic parameters of the camera. For the intrinsic camera parameters we employ a standard "central projection" model of image formation. The intrinsic parameters form a transformation, ${}^{i}\mathbf{M}_{c}$, which describes the projection of the point P_c in coordinates centered on the camera to a point in the image, wP_i, where w is the homogeneous variable.

$$wP_i = \begin{bmatrix} wx_i \\ wy_i \\ w \end{bmatrix} = {}^{i}\mathbf{M}_{c}\, P_c$$

To obtain the image coordinates for $w\,x_i, w\,y_i$ it is necessary to divide by w.

The extrinsic parameters describe a projection, ${}^{c}\mathbf{E}_{s}$, of a point from scene coordinates, P_s to camera center coordinates, P_c.

$$P_c = {}^{c}\mathbf{E}_{s}\, P_s$$

Together the intrinsic and extrinsic parameters describe a projection, ${}^{i}\mathbf{N}_{s}$, from scene coordinates to image coordinates,

$${}^{i}\mathbf{N}_{s} = {}^{i}\mathbf{M}_{c}\ {}^{c}\mathbf{E}_{s}$$

so that a point in the scene P_s is projected to a point in the image wP_i by

$$wP_i = {}^{i}\mathbf{N}_{s}\ P_s. \qquad (4.1)$$

The extrinsic camera parameters may be estimated as a composition of the position and orientation of the robot arm "tool" coordinates and the transformation from the tool coordinates to the camera. An estimate of the position and orientation of the robot arm tool coordinates is provided to us by the arm controller. The estimate of the position and orientation with respect to tools coordinates is a rigid transformation which can be calibrated when the system is initialized using a technique developed by Tsai and Lenz [Tsai et al. 87].

Modeling the scene from multiple view points requires that the recovered 3-D structure be expressed in a common coordinate system. For convenience, we have chosen to use the calibrated scene coordinate system, defined by the calibration grid. Let ${}^{r}\mathbf{R}_{e0}$ represent the robot position at the time of the calibration of the first image, and let ${}^{r}\mathbf{R}_{ek}$ represent the robot position at the time at which the flow model was updated from the k^{th} image. The extrinsic camera transformation for the k^{th} image may be computed from the robot effector position by:

$$ {}^{c}\mathbf{E}_{sk} = {}^{e}\mathbf{C}_{c}^{-1}\, {}^{r}\mathbf{R}_{ek}^{-1}\, {}^{r}\mathbf{R}_{e0}\, {}^{e}\mathbf{C}_{c}\, {}^{c}\mathbf{E}_{s}. $$

so that the projection of a point in the scene to a point in the image for camera position k is given by:

$$ {}^{i}\mathbf{N}_{sk} = {}^{i}\mathbf{M}_{c}\, {}^{e}\mathbf{C}_{c}^{-1}\, {}^{r}\mathbf{R}_{ek}^{-1}\, {}^{r}\mathbf{R}_{e0}\, {}^{e}\mathbf{C}_{c}\, {}^{c}\mathbf{E}_{s0} $$

This computation requires only one matrix inversion and two matrix multiplications. The terms ${}^{e}\mathbf{C}_{c}^{-1}$ and ${}^{r}\mathbf{R}_{e0}\, {}^{e}\mathbf{C}_{c}\, {}^{c}\mathbf{E}_{s}$ are computed at the time of calibration.

4.2 Initializing the 3-D Model

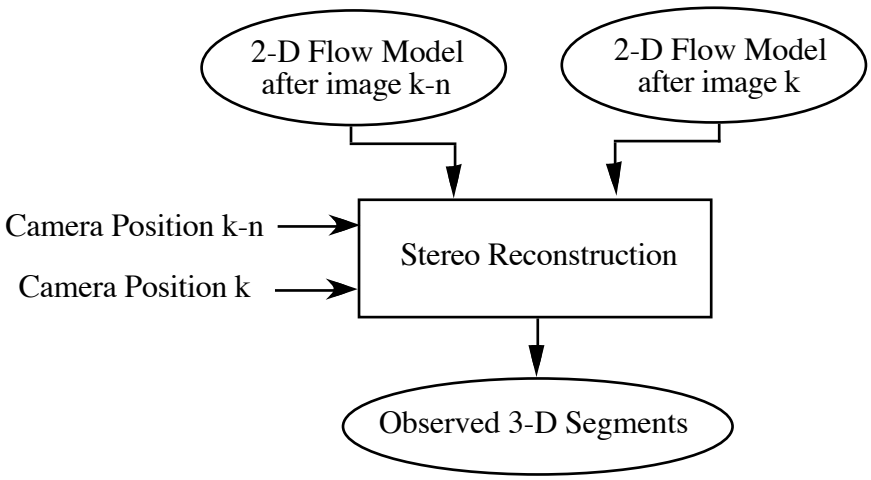

Figure 4.3 The 3-D Reconstruction Process.

The extrinsic camera parameters for arbitrary viewing positions permit us to use standard stereo reconstruction equations. This process is illustrated in figure 4.3. The composition of intrinsic and extrinsic camera parameters describes the projection of a scene point to an image point. Using homogeneous coordinates, this relation has the form:

$$ \begin{bmatrix} wx_i \\ wy_i \\ w \end{bmatrix} = {}^{i}\mathbf{N}_{s} \begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} $$

Because we can not invert ${}^{i}\mathbf{N}_{s}$, we are obliged to deal algebraically with the individual relations. Let ${}_{1}N_{sk}$, ${}_{2}N_{sk}$, and ${}_{3}N_{sk}$ represent the first, second third rows of the k^{th} transformation from scene to image, ${}^{i}\mathbf{N}_{sk}$.

For two observations (1 and 2) of a scene point we obtain the image points $P_{i1} = (x_{i1}, y_{i1})$ and $P_{i2} = (x_{i2}, y_{i2})$. By algebra we can deduce the relations [Toscani-Faugeras 86] for (x_{i1}, y_{i1}) and transformations ${}^{i}\mathbf{N}_{s1}$ and ${}^{i}\mathbf{N}_{s2}$:

$$x_{i1} = \frac{{}_{1}N_{s1} \bullet P_s}{{}_{3}N_{s1} \bullet P_s} \qquad y_{i1} = \frac{{}_{2}N_{s1} \bullet P_s}{{}_{3}N_{s1} \bullet P_s} \tag{4.1}$$

$$x_{i2} = \frac{{}_{1}N_{s2} \bullet P_s}{{}_{3}N_{s2} \bullet P_s} \qquad y_{i2} = \frac{{}_{2}N_{s2} \bullet P_s}{{}_{3}N_{s2} \bullet P_s} \tag{4.2}$$

or equivalently

$$x_{i1} ({}_{3}N_{s1} \bullet P_s) - ({}_{1}N_{s1} \bullet P_s) = 0 \qquad y_{i1} ({}_{3}N_{s1} \bullet P_s) - ({}_{2}N_{s1} \bullet P_s) = 0$$
$$x_{i2} ({}_{3}N_{s2} \bullet P_s) - ({}_{1}N_{s2} \bullet P_s) = 0 \qquad y_{i2} ({}_{3}N_{s2} \bullet P_s) - ({}_{2}N_{s2} \bullet P_s) = 0$$

Each equation describes a plane in scene coordinates that passes through a column or row of the image. This provides us with a set of four equations for recovering the three unknowns of P_s.

A common problem with edge line segments is the phenomena of "breaking". Although the token tracking process reduces this phenomena, we must still assure that the end-points of the segments correspond to the same physical point. To do this, we project the epipolar line from each end point into the other image, to determine the part of the two segments which is common to the two segments, as illustrated in figure 4.4.

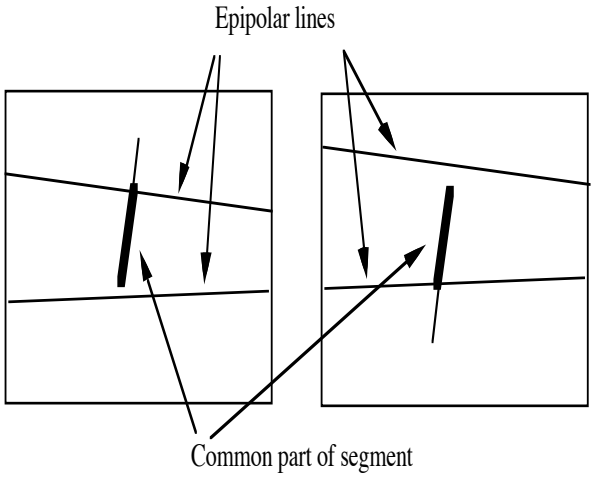

Figure 4.4 Determination of the common part of the two line segments. The epipolar lines from each end points are projected onto the other image. The part which is common to the two line segments is illustrated as a dark line.

The stereo reconstruction equations are applied to the end points of the common part of the segment to recover an observed 3-D edge segment which is represented as a pair of 3-D points and their uncertainty. These 3-D points are used to initialize the 3-D composite model. To describe the projection of the uncertainty we must describe the representation for 3-D segments.

5. Refining the 3-D Model with Multiple Observations

Once a 3-D segment has been entered in the composite model, its parameters and uncertainty can be refined by additional observations using the Kalman filter equations developed in section 2. Since our system involves a static scene, there is no estimation of the temporal derivatives. The prediction phase projects the parameters of the 3-D segment into the current image coordinates using a Jacobian transformation based on the perspective transformation ${}^{i}\mathbf{N}_{sk}$. Correspondence is furnished directly by the segment ID's, and is verified by the 2-D matching tests for similar orientation, co-linearity and overlap described above. The 3-D segment parameters are then updated using the Kalman filter equations (2.3), (2.4) and (2.5).

5.1 Representation for 3-D Lines Segments.

A line segment in the 2-D flow model corresponds to a line segment in the 3-D scene. In order to apply Kalman filter estimation techniques to 3-D reconstruction we require a representation which expresses a segment in terms of a minimum of parameters. However there are many minimal representations for 3-D line segments. In [Ayache 89] a 3-D line is represented in terms of the intersection of two planes. In [Crowley-Ramparany 87] a direct analog of the 2-D segment used above above is introduced; 3-D segments are represented as a midpoint, a direction, a half length, the 3-D uncertainty of the midpoint (perpendicular to the segment) and a 3-D uncertainty of the direction.

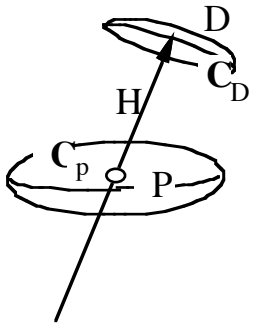

Figure 5.1 Representation for a 3-D segment and its Uncertainties.

In the 3-D model, segments are represented by the following parameters.

P The center point of the segment (x, y, z).

$\mathbf{C}_p$ The 3x3 covariance of the center point .

ΔP_s An un-normalized direction vector.

CF The confidence of segment.

ID: The Identity of the segment (from the 2-D Model).

Based on these parameters, several redundant parameters can be computed.

D The normalized direction (expressed as Δx, Δy, and Δz, normalized to unit length.)

$\mathbf{C}_D$ The 3x3 covariance of the direction

H The half length of the segment.

These parameters are initially calculated from the 3-D end-points P_1, P_2 and their covariances, $\mathbf{C}_1$, $\mathbf{C}_2$, by:

$$P_s = \frac{(P_1 + P_2)}{2}$$

$$\mathbf{C}_p = \frac{(\mathbf{C}_1 + \mathbf{C}_2)}{4}$$

$$\Delta P_s = \frac{1}{2} (P_2 - P_1)$$

From which we can calculate the half length and unit direction vector and its covariance as:

$$H = \| \Delta P_s \|$$

$$D = \frac{\Delta P_s}{\| \Delta P_s \|}$$

$$\mathbf{C}_D = \frac{(\mathbf{C}_1 + \mathbf{C}_2)}{\| \Delta P_s \|^2}$$

The covariance of the direction, $\mathbf{C}_D$, is an approximation of the uncertainty angle with respect to each of the axes.

5.2 Predicting 2-D from 3-D

The prediction phase projects the mid-points of the 3-D segment into the current image coordinates using the current transformation ${}^{i}\mathbf{N}_{sk}$.

$$w\,P_i = {}^{i}\mathbf{N}_{sk}\,P_s$$

To project the half-length, direction and covariances, we need a linear transformation from scene to image coordinates. Such a transformation is provided by writing the projections equations for x_i, y_i, and computing their derivative with respect to each of the scene coordinates, as shown in equations 4.1 and 4.2. That is, for the three lines of ${}^{i}\mathbf{N}_{sk}$ represented by ${}_1N_{sk}$, ${}_2N_{sk}$, and ${}_3N_{sk}$

$$x_i = F_{x_i}(x_s, y_s, z_s) = \frac{{}_1N_{sk} \bullet P_s}{{}_3N_{sk} \bullet P_s}$$

$$y_i = F_{y_i}(x_s, y_s, z_s) = \frac{{}_2N_{sk} \bullet P_s}{{}_3N_{sk} \bullet P_s}$$

from which we can define an observation function:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = F(x_s, y_s, z_s) = \begin{bmatrix} F_{x_i}(x_s, y_s, z_s) \\ F_{y_i}(x_s, y_s, z_s) \end{bmatrix}$$

The Jacobian of this transformation ${}^{y}\mathbf{J}_x$ gives an approximate linear transformation from scene to image coordinates for a particular value of scene coordinates.

$${}^{y}\mathbf{J}_x = \begin{bmatrix} \frac{\partial F_{x_i}(x_s, y_s, z_s)}{\partial x_s} & \frac{\partial F_{x_i}(x_s, y_s, z_s)}{\partial y_s} & \frac{\partial F_{x_i}(x_s, y_s, z_s)}{\partial z_s} \\ \frac{\partial F_{y_i}(x_s, y_s, z_s)}{\partial x_s} & \frac{\partial F_{y_i}(x_s, y_s, z_s)}{\partial y_s} & \frac{\partial F_{y_i}(x_s, y_s, z_s)}{\partial z_s} \end{bmatrix} \qquad (5.1)$$

This approximation lets us project the direction vector to image coordinates by:

$$\Delta P_i^* \equiv \begin{bmatrix} \Delta x_i^* \\ \Delta y_i^* \end{bmatrix} := {}^{Y}\mathbf{J}_X \;\Delta P_s$$

from which we can compute:

$$h_i^* = \| \Delta P_i^* \|$$

and

$$\theta^* = \mathrm{Tan}^{-1}\left(\frac{\Delta y_i^*}{\Delta x_i^*}\right)$$

The covariance of the center point can then be projected to image coordinates by:

$$\mathbf{C}_i^* := {}^Y\mathbf{J}_X \; \mathbf{C}_s^* \; {}^Y\mathbf{J}_X^T$$

The uncertainty perpendicular to the segment may be computed using the coefficient vector as

$$\sigma_c^{*2} = D^T \mathbf{C}_i^* \; D$$

where the line equation coefficients define a vector perpendicular to the segment

$$D = \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \mathrm{Sin}(\theta^*) \\ -\mathrm{Cos}(\theta^*) \end{bmatrix}$$

The uncertainty of the orientation is predicted by

$$\sigma_\theta^* = \mathrm{Tan}^{-1}\left(\frac{\sigma_c^*}{h^*}\right)$$

The correspondence is verified by comparing the predicted parameters to the 2-D model parameters using the tests described in section 3.5.

5.3 Updating the Parameters of a 3-D Segment

Updating the parameters in the 3-D composite model follows directly from the Kalman filter equations developed in section 2 using the Jacobian transformation developed in equation (5.1). The midpoint positions uncertainty, $\mathbf{C}_i(t)$, is formed from the variance of the midpoint.

$$\mathbf{C}_i(t) = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$$

This covariance permits us to compute a Kalman gain vector using equation 2.3.

$$\mathbf{K}(t) := \mathbf{C}_s(t) \; {}^Y\mathbf{J}_X^T \left[\mathbf{C}_i^*(t) + \mathbf{C}_i(t) \right]^{-1}$$

and then to update the midpoint, direction, and covariance using equations 2.4 and 2.5.

$$\hat{P}_s(t) := P_s^*(t) + \mathbf{K}(t) \left[P_i(t) - P_i^*(t) \right]$$

$$\Delta\hat{P}_s(t) := \Delta P_s^*(t) + \mathbf{K}(t) \left[\Delta P_i(t) - \Delta P_i^*(t) \right]$$

and

$$\hat{\mathbf{C}}_S(t) := \mathbf{C}_S^*(t) + \mathbf{K}(t)\ {}^Y\mathbf{J}_X\ \mathbf{C}_S^*(t).$$

5.4 Managing the Confidence of Composite Model Segments

The confidence factor in the 3-D composite model is maintained in a manner which is similar to that in the 2-D model, with one important difference. This difference concerns the elimination of segments which are occluded. Segments which reach a confidence value of CF_{max} (a value of 5 in our current implementation) do not have their CF reduced when they are no longer observed. In this way, the system can construct a model of a 3-D object which contains faces which are not simultaneously visible.

At the end of the update phase for a set of observed segments, any segment for which no correspondence was observed, and for which the CF has a value of less than CF_{max} has its confidence reduced by 1. If the CF of a segment drops below 1, the segment is removed from the composite model. Thus a segment must be present in at least 5 observations in order to be considered reliable enough to be preserved in the model.

6. Experimental Evaluation

We have performed a number of experiments in 3-D scene modeling with our system, using a camera mounted on a robot arm. The results of some of these experiments are described in this section. We first present some results from an early experiment which illustrated the function of the entire system. We then describe some more recent experiments designed to determine the limits to the precision of the system.

6.1 Experimental Set-up

Our experiments used a CCD camera equipped with a 12.5 mm lens and mounted in the gripper of the arm. Video signals from the camera are digitized using a frame buffer/digitizer board mounted in the bus of a work-station. A six axis robot arm is linked with a robot controller capable of providing the location of the robot gripper simultaneously with the acquisition of each image. This information allows us to reconstruct the 3-D geometry of the scene.

The object used for each experiment is located at a distance of about 30 cm from the camera. The camera follows a roughly circular trajectory which passes over the object. Camera displacements are less than 1 cm per image, with rotations under 5° per image. The camera is moved to each image position, and then an image is acquired (stop-and-go motion). Each sequence is composed of between 90 and 130 images, taken in 3 or 4 trajectories. The trajectories are determined automatically so as to

assure a nearly perpendicular movement for each of the major 3-D edge segments on the object.

Edge points were detected by a version of the Canny operator designed and programmed by R. Deriche [Canny 86], [Deriche 87]. Edge points were chained and segmented by a chaining program realized by G. Giraudon. The 3-D model is updated after every fifth update of the 2-D model.

6.2 An Example of the Complete Process

One of the image sequences with which we have debugged the system is composed of 90 images of an electrical switch box. Figure 6.1 shows an example of a raw image from this sequence. Figure 6.2 shows the edge segment description from every 5th image from image 15 to 90. Figure 6.3 shows the 2-D line segments that have been tracked for every 5th image for images 30 through 55. Segments are displayed with their ID number. Figure 6.4 shows the evolution of the 3-D model. On the left are the 3-D segments obtained after image 15. The middle shows the result after integrating image 40. The right shows the result after integration of all 90 images.

To evaluate the accuracy of these results we have built a software tool which permits us to easily compare the measurements of physical structures on the object to the same structures in the estimated model. Among other measures, this tool displays the difference in orientation ($\Delta\theta$) and the perpendicular distance (dist) between two segments. Table 6.1 shows the evolution of these measurements for pairs of segments after the observation of images 40, 65, and 90. These values are compared to the values measured on the physical object ("exact Values").

Segments		Exact Values		Image 40		Image 65		Image 90	
		dist	$\Delta\theta$	dist	$\Delta\theta$	dist	$\Delta\theta$	dist	$\Delta\theta$
55	26	0.0	90.0	0.4	99.7	0.2	93.1	0.2	91.8
126	27	38.0	90.0	23.9	98.3	35.2	92.7	36.8	89.8
137	26	76.0	90.0	73.7	87.5	68.5	87.9	71.2	90.1
137	126	22.0	0.0	19.0	3.0	21.4	0.5	21.0	0.3
27	40	38.0	0.0	37.9	3.3	38.2	0.5	38.4	0.9
55	40	50.0	0.0	46.9	2.8	47.9	0.5	48.6	0.5

Table 6.1 Perpendicular distance (dist) in mm and difference in orientation ($\Delta\theta$) in degrees between pairs of segments after updating the 3-D model with images 40, 65, and 90.

6.3 Second Example: A Cubic-like Object

Figure 6.5 shows images 80 and 130 from a sequence of 130 images of a cube-like object. Figure 6.6 shows the edge segments extracted from every fifth image from image 15 to 130. Three trajectories

were used in this sequence. The first trajectory is images 1 to 70. The second trajectory for images 71 to 100 and the third trajectory for images 101 to 130. Figure 6.7 shows all the 2-D edge segments which were tracked from images 45 through 70. The ID is superimposed over each segment.

Figure 6.8 demonstrates the evolution of the 3-D model during reconstruction. The left side shows the reconstruction after the 70th image. The middle shows the reconstruction after the 100th image and the right side shows the reconstruction after the 130th image. The accuracy of the reconstruction is illustrated in the table 6.2.

Segments		Exact Values		Image 30		Image 60		Image 100		Image 130	
		dist	Δθ	dist	Δθ	dist	Δθ	dist	Δθ	dist	Δθ
35	53	50.0	0.0	–	–	–	–	46.0	3.6	48.5	0.5
30	35	50.0	0.0	–	–	–	–	51.7	0.1	51.7	0.1
7	22	50.0	0.0	–	–	50.5	1.9	50.7	1.1	50.7	1.1
3	22	50.0	0.0	–	–	49.7	1.5	49.6	0.9	49.6	0.9
2	7	50.0	0.0	50.6	1.2	50.0	0.4	50.1	0.2	50.1	0.2
3	53	0.0	90.0	–	–	–	–	1.1	92.1	2.4	90.9

Table 6.2 Perpendicular distance (dist) in mm and difference in orientation (Δθ) in degrees between pairs of segments after updating the 3-D model with images 30, 60, 100 and 130. A "-" indicates that one of the segments had not yet been integrated into the model because of occlusion.

Segments		Exact Values		Image 30		Image 75		Image 100		Image 130	
		dist	Δθ	dist	Δθ	dist	Δθ	dist	Δθ	dist	Δθ
11	12	20.0	0.0	20.4	0.4	20.5	0.3	20.5	0.3	20.5	0.3
11	13	0.0	90.0	0.1	85.0	0.3	88.7	0.3	88.7	0.3	88.7
46	58	20.0	0.0	–	–	21.0	4.8	20.6	2.9	19.3	0.1
4	13	60.0	0.0	64.3	4.5	64.4	1.4	64.4	1.4	64.4	1.4
12	67	117.0	0.0	–	–	–	–	–	–	114.7	1.0
13	31	132.0	90.0	–	–	133.2	89.9	133.2	89.9	133.0	89.7

Table 6.3 Perpendicular distance (dist) in mm and difference in orientation (Δθ) in degrees between pairs of segments after updating the 3-D model with images 30, 75, 100 and 130 for the third object.

6.4 Third Test Object: A Metal Part

A thirds sequence of images illustrates the reconstruction of a metal part from 130 images. Figure 6.9 shows an image of the part from this sequence. Figure 6.10 shows the edge segments from every 5th image. As with the second example, the sequence is composed of three trajectories. Figure 6.11 shows

the 2-D model after every fifth update phase from images 5 through 30. Figure 6.12 shows the evolution of the 3-D model after the 30th, 75th and 130th image. Table 6.3 shows the accuracy of reconstruction of selected segments during the sequence.

7. Discussion

The problem of dynamically maintaining a description of the local environment is fundamental to perception. A necessary aspect of a system is the ability to integrate perceptual information either from different sources, or from the sequential observations with the same source. Construction of such systems has led us to propose a framework for perceptual integration, and to develop a set of techniques for building such systems. We have constructed dynamic world modeling systems using ultrasonic range sensors [Crowley 89a] and vertical line stereo [Crowley 90a] using these techniques.

Dynamic world modeling can be seen as a cyclic process composed of the phases: predict, match and update. We began the paper by describing how techniques from estimation theory can be adapted to this process in the case of dynamic modeling of numeric properties. Direct application of the Kalman filter and the Mahalanobis distance to even such simple problems as edge line tracking can lead to the need to invert large numbers of 8 by 8 matrices for each cycle of the predict-match-update process. In order to build systems that function in real time, we have shown how edge segment tracking can be simplified by approximating a single large first order estimator with four small estimators by using a clever line segment representation. The line segment representation based on mid-point, direction, length has made it possible for us to construct a co-processor "token-tracking" card which operates at near video rates. The token-tracking process has become a standard building block for real time vision systems at our two laboratories.

The Kalman Filter equations permit observations of a small number of properties to serve as constraints for a vector composed of a larger number of properties. This fact is counter-intuitive and easily mis-understood. We illustrated this principle in both the 2-D and 3-D modeling systems. In the 2-D token tracking system, observations of the value of position, length and orientation permitted us to estimate the first temporal derivatives of these values. In the 3-D modeling system, observations of edge segments in the 2-D model served as constraints to refine estimates of the 3-D position and orientation of segments.

Tracking preserves correspondence. This simple fact alone justifies the use of 2-D tracking in a vision system. Avoiding the problem of searching for 3-D correspondence has made it possible for us to add a 3-D tracking process which infers and maintains a 3-D image description from the positions of the 2-D edge segments. The 3-D system exploits the estimates from the 2-D tracking system as observations. This architecture shows that it is possible to compose perceptual systems with multiple independent predict-match-update loops operating in different perceptual coordinates and at different levels of abstraction.

The reliable function of the system depends on accurate calibration of the camera, and the robot arm, and the position of the camera on the robot. The implementation and debugging of this system have led us to invest several man years in a sequence of calibration techniques [Puget-Skordas 90]. The development of stable and reliable calibration has proven to be much harder than anyone connected with this project had expected.

As seen in section 6, when these components are well calibrated, the 3-D form which is recovered quickly converges to precisions on the order of a millimeter. When the system is not properly calibrated, the result is an increased error in the projection from 3-D to 2-D for matching. If the uncertainty associated with observations is sufficiently large, the system will continue to function, but will converge much less rapidly. If the uncertainties associated with observations are too small, then matching will fail and the geometric model will contain a smaller number of segments.

A major cause of failure is unmodeled accelerations. Such accelerations are due to abrupt motions of the camera or of objects in the scene, for which the acceleration is beyond the acceleration uncertainty a_{cc}. For motions which are induced by the robot arm, such accelerations may be inferred from the arm motion, and entered into the tracking process via the term V(t) in equation 2.1. When such terms are due to the acceleration of objects, then can only be handled by proper initialization of the observation uncertainty.

These critical "observation" uncertainties are crucial to the function of the system. At the current time, the uncertainties for the segment tracking process are "tuned" by hand. That is, when the uncertainties are large, the number of tracked segments remains stable, but the precision for tracking is decreased. When the uncertainties are too small, the number of tracked segments decreases rapidly. This provides a method to "tune" the uncertainties for a particular situation. The process is made to run, and the number of segments with a confidence factor greater than 3 are displayed. The uncertainties are then systematically reduced until the number of segments in the model begins to decrease.

The techniques for incremental 3-D modeling presented above are not clearly superior to a stereo system for 3-D vision. They should be considered as complementary. Indeed, we are currently refining a vision system which uses the 2-D tracking and incremental 3-D modeling systems described above in conjunction with a 3-D inference from stereo cameras. The results described above demonstrate the power of the use of the predict-match-update cycle for dynamic world modeling, as well as the generality of the Kalman filter equations and the Mahalanobis distance for constructing perceptual systems which use numerical properties.

Acknowledgments

The work described in this paper was partly financed by the European Commission DG XIII as part of

projects ESPRIT P940 and BR 3038. Complementary funding has recently been received from the Region Rhône-Alpes. The Token Tracker was implemented by Christophe Discours in collaboration with Olivier Faugeras of INRIA. A first version of the 3-D reconstruction algorithm was developed in collaboration with Fano Ramparany and was integrated and tested by Jean-Noel Soulier and Laurent Lefort. The latest version, reported here, was programmed by Renaud Zigmann. Much of the recent refinement of the token tracker, as well as its hardware realization have been performed by Serge DePaoli. A special thanks goes to Nicholas Ayache, whose crticisms and support helped us through the many versions of this system. Parts of this paper were written using the facilities of the laboratory of Prof. Tsuji of Univ. of Osaka, in Japan.

Bibliography

[Ayache 89] Ayache, N. "Construction et Fusion de Représentations Visuelles 3D", Thèse de Doctorat d'Etat, Universtié Paris-Sud, centre d'Orsay, 1988.

[Brammer-Siffling 89] Brammer, K. and G. Siffling, Kalman-Bucy Filters, Artech House Inc., Norwood Mass, 1969.

[Brooks 85] Brooks, R. A., "Visual Map Making for a Mobile Robot", 1985 IEEE Conference on Robotics and Automation, 1985.

[Brown et al. 89] Brown, C. and H. Durrant Whyte, J. Leonard and B. Y. S. Rao, "Centralized and Decentralized Kalman Filter Techniques for Tracking, Navigation and Control", DARPA Image Understanding Workshop, May 1989.

[Bucy 59] Bucy, R. S. "Optimum finite filters for a special non-stationary class of inputs", Internal Rep. BBD-600, Applied Physics Laboratory, Johns Hopkins University.

[Bucy 68] Bucy R. S. and P. D. Joseph, Filtering for Stochastic Processes, with applications to Guidance, Interscience New York, 1968.

[Canny 86] Canny, J., "A Computational Approach to Edge Detection", IEEE Trans. on P.A.M.I., Vol PAMI-8, No. 6, Nov. 1986.

[Chatila 85] Chatila, R. and J. P. Laumond, "Position Referencing and Consistent World Modeling for Mobile Robots", "1985 IEEE Conf. on Robotics and Automation", St. Louis, March 1985.

[Chehikian 89] Chehikian, A., P. Stelmaszyk and P. Depaoli, "Hardware Evaluation Process for Tracking Edges Lines", Workshop on Industrial Applications of Machine Intelligence and Vision.Tokyo, 1989.

[Chehikian 90] Chehikian, A., S. Depaoli, and P. Stelmaszyk, "Real Time Token Tracker", EUSIPCO, Barcelona, Sept. 1990 .

[Crowley 84] Crowley, J. L., "A Computational Paradigm for 3-D Scene Analysis", IEEE Conf. on Computer Vision, Representation and Control, Annapolis, March 1984.

[Crowley 85] Crowley, J. L., "Navigation for an Intelligent Mobile Robot", IEEE Journal on Robotics and Automation, 1 (1), March 1985.

[Crowley 86] Crowley, J. L., "Representation and Maintenance of a Composite Surface Model", IEEE International Conference on Robotics and Automation, San Francisco, Cal., April, 1986.

[Crowley et. al. 88] Crowley, J. L., P. Stelmaszyk and C. Discours, "Measuring Image Flow by Tracking Edge-Lines", Second ICCV, Tarpan Springs, Fla. 1988.

[Crowley 89] Crowley, J. L., "World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging", 1989 IEEE Conference on Robotics and Automation, Scottsdale AZ, May 1989.

[Crowley et. al. 90] Crowley, J. L., P. Bobet and K. Sarachik, "Dynamic World Modeling using Vertical Line Stereo", First European Conference on Computer Vision, (ECCV-1) Antibes, France, 1990.

[Crowley-Stelmaszyk 90] Crowley, J. L. and P. Stelmaszk, "Measurement and Integration of 3-D Structures By Tracking Edge Lines", First European Conference on Computer Vision, (ECCV-1) Antibes, France, 1990.

[Crowley-Ramparany 87] Crowley, J. L. and F. Ramparany, "Mathematical Tools for Manipulating Uncertainty in Perception", AAAI Workshop on Spatial Reasoning and Multi-Sensor Fusion", Kaufmann Press, October, 1987.

[Deriche 87] Deriche, R., "Using Canny's Criteria to Derive a Recursively Implemented Optimal Edge Detector", International Journal of Computer Vision, Vol 1(2), 1987.

[Durrant Whyte 87] Durrant-Whyte, H. F., "Consistent Integration and Propagation of Disparate Sensor Observations", Int. Journal of Robotics Research, Spring, 1987.

[Faugeras-Toscani 86] Faugeras, O. D. ,G. Toscani, "The Calibration Problem for Stereo. Computer Vision and Pattern Recognition, pp 15-20, Miami Beach, Florida, USA, June 1986.

[Faugeras, et. al. 86] Faugeras, O. D. , N. Ayache, and B. Faverjon, "Building Visual Maps by

Combining Noisey Stereo Measurements", IEEE International Conference on Robotics and Automation, San Francisco, Cal., April, 1986.

[Gennery 82] Gennery, D. B., "Tracking Known Three Dimensional Objects", Proc. of the National Conference on Artificial Intelligence (AAAI-82), Pittsburgh, 1982.

[Herman-Kanade 86] Herman M. and Kanade T. Incremental reconstruction of 3D scenes from multiple complex images. Artificial Intelligence vol-30, pp.289-341, 1986.

[Hildreth 82] Hildreth, E. C., The Measurement of Visual Motion, 1983 ACM Distinguished Dissertation, MIT Press, Cambridge Mass. 1982.

[Horaud et al. 90] Horaud, P., F. Veillon, and T. Skordas, "Finding Geometric and Topological Structures in Image", First European Conference on Computer Vision, (ECCV-1) Antibes, France, 1990.

[Huang 83] Huang T. H., Image Sequence Processing and Dynamic Scene Analysis, Springer Verlag, Berlin, 1983.

[Jazwinski 70] Jazwinski, J. E., Stochastic Processes and Filtering Theory, Academic Press, New York, 1970.

[Kalman 60] Kalman, R. E. "A New Approach to Linear Filtering and Prediction Problems", Transactions of the ASME, Series D. J. Basic Eng., Vol 82, 1960.

[Kalman 61] Kalman, R. E. and R. S. Bucy, "New Results in LInear Filtering and Prediction Theory", Transaction of the ASME, Series D. J. Basic Eng., Vol 83, 1961.

[Kolmogorov 40] Kolmogorov, A. N., "Interpolation and Extrapolation of Stationary Random Sequences", Bulletin of the Academy of Sciences of the USSR Math. Series, Vol 5., 1941.

[Marr 80] Marr, D. and E. C. Hildreth, "Theory of Edge Detection", Proc. of the Roy. Soc. Lond. B, Vol 207, 1980.

[Matthies et. al. 87] Matthies, L., R. Szeliski, and T. Kanade, "Kalman Filter-based Algorithms for Estimating Depth from Image Sequences", CMU Tech. Report, CMU-CS-87-185, December 1987.

[Melsa-Sage 71] Melsa, A. P. and J. L. Sage, Estimation Theory, with Applications to Communications and Control, McGraw-Hill, New York, 1971.

[Puget-Skordas 90] Puget P. and T. Skordas, "Calibrating a Mobile Camera", Image and Vision Computing, Vol 8, No. 4, Nov. 1990.

[Ramaparany 89] Ramparany, F., "Perception Multi-sensorielle de la Structure Geometrique d'une Scene", Thèse de Doctrat, INPG, Feb 1989.

[Skordas et. al. 89] Skordas, T., P. Puget, R. Zigmann and N. Ayache, "Building 3-D Edge-Lines Tracked in an Image Sequence", Proc. 2nd Conf. on Intelligence Autonomous Systems, Amsterdam, December 1989.

[Smith-Cheeseman 87] Smith, R. C. and P. C. Cheeseman, "On the Estimation and Representation of Spatial Uncertainty", International Journal of Robotics Research 5 (4), Winter, 1987.

[Tsai 85] Tsai, R. Y., "An Efficient and Accurate Camera Calibration Technique for 3-D Machine Vision", Technical Report, IBM T. J. Watson Reearch Center, 1986.

[Weiner 49] Weiner, N., Extrapolation, Interpolation and Smoothing of Staitionary Time Series, John Wiley and Sons, New York., 1949.

Figure 6.1 An example of a raw image from this sequence.

Figure 6.2 shows the edge segment description from every 5th image from image 15 to 90.

Figure 6.3 shows the 2-D line segments that have been tracked for every 5th image for images 30 through 55.

Figure 6.4 shows the evolution of the 3-D model. On the left are the 3-D segments obtained after image 15. The middle shows the result after integrating image 40. The right shows the result after integration of all 90 images.

Figure 6.5 Images 80 and 130 from a sequence of 130 images of a cube-like object.

Figure 6.6 The edge segments extracted from every fifth image from image 15 to 130.

Figure 6.7 All the 2-D edge segments which were tracked from images 45 through 70.

Figure 6.8 The evolution of the 3-D model during reconstruction. The left side shows the reconstruction after the 70th image. The middle shows the reconstruction after the 100th image and the right side shows the reconstruction after the 130th image.

Figure 6.9 An image of the part from this sequence.

Figure 6.10 The edge segments from every 5th image.

Figure 6.11 The 2-D model after every fifth update phase from images 5 through 30.

Figure 6.12 shows the evolution of the 3-D model after the 30th, 75th and 130th image.