Principles and Techniques for Sensor Data Fusion

James L Crowley and Yves Demazeau
LIFIA (IMAG)
46 avenue Félix Viallet
F-38031 Grenoble Cédex, FRANCE

Abstract

This paper concerns a problem which is basic to perception: the integration of perceptual information into a coherent description of the world. In this paper we present perception as a process of dynamically maintaining a model of the local external environment. Fusion of perceptual information is at the heart of this process.

After a brief introduction, we review the background of the problem of fusion in machine vision. We then present fusion as part of the process of dynamic world modeling, and postulate a set of principles for the "fusion" of independent observations. These principles lead to techniques which permit perceptual fusion with qualitatively different forms of data, treating each source of information as constraints. For numerical information, these principles lead to specific well known tools such as various forms of Kalman filter and Mahalanobis distance. For symbolic information, these principles suggest representing objects and their relations as a conjunction of properties encoded as schema.

Dynamic world modeling is a cyclic process composed of the phases: predict, match and update. These phases provide a framework with which we can organise and design perceptual systems. We show that in the case of numerical measurements, this framework leads to the use of a form of Kalman filter for the prediction and update phases, while a Mahalanobis distance is used for matching. In the case of symbolic information, elements of the framework can be constructed with schema and production rules. The framework for perceptual information is illustrated with the architectures of several systems.

Outline:

1. Introduction.....1
1.1 Perception and Sensing 1
1.2 Background and State of the Art in Perceptual Fusion.....2
2 Fusion and Dynamic World Modeling.....3
2.1 A General Framework for Dynamic World Modeling.....4
2.2 Principles for Integrating Perceptual Information 5
3. Techniques for Fusion of Numerical Properties 7
3.1 State Representation: A Vector of Properties 7
3.2 Prediction: Discrete State Transition Equations 9
3.3 Matching Observation to Prediction: The Mahalanobis Distance 11
3.4 Updating: The Kalman Filter Update Equations.....13
3.5 Eliminating Uncertain Primitives and Adding New Primitives.....14
4. Fusion of Symbolic Properties.....14
4.1 Philosophical Foundations 15
4.2 Principles for Symbolic Fusion.....16
4.3 A Symbolic Form of the Predict, Match and Update Cycle 17
5 Example Systems Constructed in the Framework. 18
5.1 Dynamic World Modeling Using Ultrasound.....19
5.2 2D Edge Segment Following 25
5.3 Vertical Line Stereo System 28
5.4 World Modeling Using Ultrasound and Vertical Line Stereo 30
5.5 An Integrated Active Vision System 31
6. Conclusions.....35
References.....36

Principles and Techniques for Sensor Data Fusion

1. Introduction

The problem of combining observations into a coherent description of the world is basic to perception. In this paper, we present a framework for sensor data fusion and then postulate a set of principles based on experiences from building systems. We argue that for numerical data, techniques from estimation theory may be directly adapted to the problem. For symbolic data, these principles suggest an adaptation of certain existing techniques for the problem of perceptual fusion.

We start the paper by discussing the problem of perception and sensing, and describing some background from related scientific disciplines.

1.1 Perception and Sensing

Perception is not a goal in itself, but a means to obtain a certain behaviour by an agent (a thing which "acts"). In order to plan and execute actions, an intelligent agent must reason about its environment. For this, the agent must have a description of the environment. This description is provided by fusing "perceptions" from different sensing organs (or different interpretation procedures) obtained at different times.

We define perception as:

> The process of maintaining of an internal description of the external environment.

The external environment is that part of the universe which is accessible to the sensors of an agent at an instant in time. In theory, it would seem possible to use the environment itself as the internal model. In practice, this requires an extremely complete and rapid sensing ability. It is far easier to build up a local description from a set of partial sources of information and to exploit the relative continuity of the universe with time in order to combine individual observations.

We refer to the problem of maintaining an internal description of the environment as a that of "Dynamic World Modeling". By dynamic we mean that the description evolves over time based on information from perception. This description is a model, because it permits the agent to "simulate" the external environment. This use of model conflicts with "models" which a systems designer might use in building a system. This unhappy confusion is difficult to avoid given that the two uses of "model" are thoroughly embedded in the vocabulary of the scientific community. This confusion is particularly troublesome in the area of perceptual fusion, where a sensor "model" is necessary for the proper design of the system, and the result of the system is to maintain a world "model". Having

signaled this possible confusion, we will continue with the terminology which is common in the vision and robotics communities: the use of “model” for an internal description used by a system to reason about the external environment.

1.2 Background and State of the Art in Perceptual Fusion

Recent advances in sensor fusion from within the vision community have largely entailed the rediscovery and adaptation of techniques from estimation theory. These techniques have made their way to vision via the robotics community, with some push from military applications.

For instance, in the early 1980's, Herman and Kanade [30] combined passive stereo imagery from an aerial sensor. This early work characterized the problem as one of incremental combination of geometric information. A similar approach was employed by the author for incremental construction of world model of a mobile robot using a rotating ultrasonic sensor [14]. That work was generalized [13] 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". Driven by the needs of perception for mobile robotics, Brooks [5] and Chatila [11] also published ad-hoc techniques for manipulation of uncertainty.

In 1985, a pre-publication of a paper by Smith and Cheeseman was very widely circulated [45]. 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 [26] on the manipulation of uncertainty in robotics and perception. This thesis presents derivations of techniques for manipulating and integrating sensor information which are extensions of technique from estimation theory. Well versed in estimation theory, Faugeras and Ayache [27] contributed an adaptation of this theory to stereo and calibration. From 1987, a rapid paradigm shift occurred in the vision community, with techniques from estimation theory being aggressively adapted.

While most researchers applying estimation theory to perception can cite one of the references [45], [26] or [27] 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 [37] and Weiner [47]. Bucy [9] showed that the method of calculating the optimal filter parameters by differential equation could also be applied to non-stationary processes. Kalman [34] 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 [10], Jazwinski [32], and in particular by Melsa and

Sage [41]. These relations are reviewed in a recent paper by Brown et. al. [6], as well as in a book by Brammer and Siffling [4].

These techniques from estimation theory provide a theoretical foundation for the processes which compose the proposed computational framework for fusion in the case of numerical data. An alternative approach for such a foundation is the use of minimum energy or minimum entropy criteria. An example of such a computation is provided by a Hopfield net [31]. The idea is to minimize some sort of energy function that expresses quantitatively by how much each available measurement and each imposed constraint are violated [38]. This idea is related to regularization techniques for surface reconstruction employed by Terzopoulos [46]. The implementation of regularization algorithms using massively parallel neural nets has been discussed by Marroquin, Koch et. al. [36], Poggio and Koch [43] and Blake and Zisserman [3].

Estimation theory techniques may be applied to combining numerical parameters. In this paper, we propose a computational framework which may be applied to numeric or symbolic information. In the case of symbolic information, the relevant computational mechanisms are inference techniques from artificial intelligence. In particular, fusion of symbolic information will require reasoning and inference in the presence of uncertainty using constraints.

The Artificial Intelligence community has developed a set of techniques for symbolic reasoning. In addition to brute force coding of inference procedures, rule based "inference engines" are widely used. Such inference may be backward chaining for diagnostic problems, consultation, or data base access as in the case of MYCIN [8]. Rule based inference may also be forward chaining for planning or process supervision, as is the case in OPS5 [28], [7]. Forward and backward chaining can be combined with object-oriented "inheritance" scheme as is the case in KEE and in SRL. Groups of "experts" using these techniques can be made to communicate using black-board systems, such as BB1 [29]. For perception, any of these inference techniques must be used in conjunction with techniques for applying constraint based reasoning to uncertain information.

Several competing families of techniques exist within the AI community for reasoning under uncertainty. Automated Truth Maintenance Systems [24] maintain chains of logical dependencies, when shifting between competing hypotheses. The MYCIN system [8] has made popular a set of ad-hoc formulae for maintaining the confidence factors of uncertain facts and inferences. Duda, Hart and Nilsson [25] have attempted to place such reasoning on a formal basis by providing techniques for symbolic uncertainty management based on Bayesian theory. Shafer has also attempted to provide a formal basis for inference under uncertainty by providing techniques for combining evidence [44]. A large school of techniques known as "Fuzzy Logic" [48] exist for combining imprecise assertions and inferences.

2 Fusion and Dynamic World Modeling

This section presents a general framework for dynamic world modeling. The problem of perceptual fusion is identified as fundamental to this process.

The section begins with description of dynamic world modeling as an iterative process of integrating observations into an internal description. This process provides a framework within which to examine problems of perceptual fusion. Using this framework, a set of principles for fusing perceptual information are elaborated. These principles are then illustrated in the following sections by presenting techniques for each of the phases of the cyclic process which make up the framework.

2.1 A General Framework for Dynamic World Modeling

A general framework for dynamic world modeling is illustrated in figure 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.

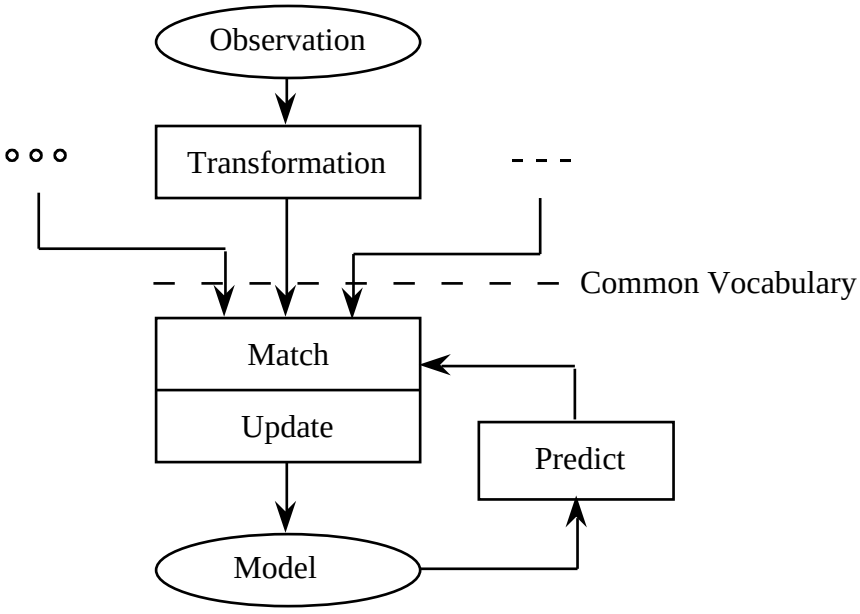

Figure 1. A Framework for Dynamic World Modeling.

Predict: In the prediction phase, 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: In the match phase, the transformed observation in brought into correspondence with the predictions. Such matching requires that the observation and the prediction express information

which is qualitatively similar. Matching requires that the predictions and observations be transformed to the same coordinate space and to a common vocabulary.

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.

We have demonstrated systems in which this framework is applied to maintain separate models of the environment with different update rates and information at different levels of abstraction. For example, in the SAVA Active Vision System [22], dynamic models are maintained for edge segments in image coordinates, for 3D edges in scene coordinates and of symbolic labels of recognized objects. In the MITHRA surveillance robot system [17], a geometric model of the environment is maintained in world coordinates, and a separate symbolic model is maintained based on searching the geometric model to detected expected objects.

From building systems using this framework, we have identified a set of principles for integrating perceptual information. These principles follow directly from the nature of the cyclic process for dynamic world modeling.

2.2 Principles for Integrating Perceptual Information

Experience from building systems for dynamic world modeling have led us to identify a set of principles for integrating perceptual information. These principles follow directly from the nature of the "predict-match-update" cycle presented in figure 1.

Principle 1) Primitives in the world model should be expressed as a set of properties.

A model primitive expresses an association of properties which describe the state of some part of the world. This association is typically based on spatial position. For example the co-occurrence of a surface with a certain normal vector, a yellow color, and a certain temperature. For numerical quantities, each property can be listed as an estimate accompanied by a precision. For symbolic entities, the property slot can be filled with a list of possible values, from a finite vocabulary. This association of properties is known as the "state vector" in estimation theory.

Principle 2) Observation and Model should be expressed in a common coordinate system.

In order to match an observation to a model, the observation must be "registered" with the model. This typically involves transforming the observation by the "inverse" of the sensing process, and thus implies a reliable model of the sensor geometry and function.

When no prior transformation exists, it is sometimes possible to infer the transformation by matching the structure of an observation to the internal description. In the absence of a priori information, such a matching process can become very computationally expensive. Fortunately, in many cases an approximate registration can be provided by knowledge that the environment can change little between observations.

The common coordinate system may be scene based or observer based. The choice of reference frame should be determined by considering the total cost of the transformations involved in each predict-match-update cycle. For example, in the case of a single stationary observer, a sensor based coordinate system may minimize the transformation cost. For a moving observer with a model which is small relative to the size of the observations, it may be cheaper to transform the model to the current sensor coordinates during each cycle of modeling. On the other hand, when the model is large compared to the number of observations, using an external scene based system may yield fewer computations.

Transformations between coordinate frames generally require a precise model of the entire sensing process. This description, often called a “sensor model”, is essential to transform a prediction into the observation coordinates, or to transform an observation into a model based coordinate system. Determining and maintaining the parameters for such a “sensor model” is an important problem which is not addressed in this paper.

Principle 3) Observation and model should be expressed in a common vocabulary.

A perceptual model may be thought of as a data base. Each element of the data base is a collection of associated properties. In order to match or to add information to a model, an observation needs some be transformed to the terms of the data base in order to serve as a key. It is possible to calculate such information as needed. However since the information is used both in matching and in updating, it makes more sense to save it between phases. Thus we propose expressing the observation in a subset of the properties used in the model.

An efficient way to integrate information from different sensors is to define a standard "primitive" element which is composed of the different properties which may be observed or inferred from different sensors. Any one sensor might supply observations for only a subset of these properties. Transforming the observation into the common vocabulary allows the fusion process to proceed independent of the source of observations.

Principle 4) Properties should include an explicit representation of uncertainty.

Dynamic world modeling involves two kinds of uncertainty: precision and confidence. Precision can be thought of as a form of spatial uncertainty. By explicitly listing the precision of an observed property, the system can determine the extent to which an observation is providing new information to a model. Unobserved properties can be treated as observations which are very imprecise. Having a model of the sensing process permits an estimate of the uncertainties to be calculated directly from the geometric situation.

Principle 5) Primitives should be accompanied by a confidence factor.

Model primitives are never certain; they should be considered as hypotheses. In order to best manage these hypotheses, each primitive should include an estimate of the likelihood of its existence. This can have the form of a confidence factor between -1 and 1 (such as in MYCIN [8]), a probability, or even a symbolic state from a finite set of confidence state.

A confidence factor provides the world modeling system with a simple mechanism for non-monotonic reasoning. Observations which do not correspond to expectations may be initially considered as uncertain. If confirmation is received from further observation, their confidence is increased. If no further confirmation is received, they can be eliminated from the model.

The application of these principles leads to a set of techniques for the processes of dynamic world modeling. In the next section we discuss the techniques for the case of numerical properties, and provide examples from systems in our laboratory. This is followed by a discussion of the case of symbolic properties.

3. Techniques for Fusion of Numerical Properties

In the case of numerical properties, represented by a primitive composed of a vector of property estimates and their precisions, a well defined set of techniques exists for each of the phases of the modeling process. In this section we show that the Kalman filter prediction equations provides the means for predicting the state of the model, the Mahalanobis Distance provides a simple measure for matching, and the Kalman filter update equations provide the mechanism to update the property estimates in the model. We also discuss the problem of maintaining the confidence factor of the property vectors which make up the model.

3.1 State Representation: A Vector of Properties

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

A model may also include "grouping" primitives which assert relations between lower level primitives. Examples of such groupings include connectivity, co-parallelism, junctions and symmetry. Such groupings constitute abstractions which are represented as symbolic properties.

Each primitive, Pi(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 label by which the primitive may be identified and recalled. 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 with 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.

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

$\hat{X}(t) \equiv \{ \hat{x}_1(t), \hat{x}_2(t), \ldots \hat{x}_n(t)\}$.

The actual state of the external world, X(t), is estimated by an observation process ${}^{Y}\mathbf{H}_X$ which projects the world onto a observation vector Y(t). The observation process is generally corrupted by random noise, N(t).

$Y(t) = {}^{Y}\mathbf{H}_X \, X(t) + N(t)$.

The world state, X(t), is not directly knowable, and so our estimate is taken to be the expected value $\hat{X}(t)$ built up from observations. At each cycle, the modeling system produces an estimate $\hat{X}(t)$ by combining a predicted observation, $Y^*(t)$, with an actual 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 $\hat{\mathbf{C}}(t)$ which represents the square of the expected difference between the estimate and the actual world state.

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

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 such a system.

The uncertainty estimate provides two crucial roles:

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 $\hat{\mathbf{C}}(t)$ may increase the computing time for finding a match.

3.2 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. The Kalman filter permits us to 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 $\hat{\mathbf{X}}(t)$, having variance $\hat{\sigma}_X^2$. A first order prediction of the value x*(t+ΔT) requires an estimate of the first temporal derivative, $\hat{x}'(t)$.

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

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 zero mean, and thus is estimated to be zero. However, in some situation 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, $\hat{\mathbf{X}}(t)$, composed of the properties $\hat{x}_1(t)$ and $\hat{x}_2(t)$ and their derivatives.

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

In matrix form, the prediction can be written as:

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

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

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

This gives the prediction equation in matrix form:

$$X^*(t+\Delta T) := \Phi \hat{X}(t) + \hat{V}(t) \qquad (1)$$

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:

$$\mathbf{C}_x^*(t+\Delta T) := \Phi \hat{\mathbf{C}}_x(t) \Phi^T + \hat{\mathbf{Q}}_x(t) \qquad (2)$$

3.3 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.

$$Y(t) = {}^Y\mathbf{H}_X\, X(t) + W(t)$$

The matrix ${}^Y\mathbf{H}_X$ constitutes a "model" of the sensing process 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, ${}^Y\mathbf{H}_X$, can 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 two properties 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 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:

$$D^2_{nm} = \frac{1}{2} \frac{(y^*_n - y_m)^2}{(\sigma^{*2}_{y_n} + \sigma^2_{y_m})}$$

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.

The rejection threshold for matching depends on the trade-off between the risk of rejecting a valid primitive, as defined by the X^2 square distribution and the desire to eliminate false matches. For example, for a single 1-D variable, to be sure to not reject 90% of true matches, the normalized distance should be smaller than 2.71. For 95% confidence, the value is 3.84. As the probability of not rejecting a good match goes up, so does the probability of false alarms.

3.4 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 $\mathbf{C}^*_y(t)$.

$$\mathbf{K}(t) := \mathbf{C}^*_x(t) \; {}^Y\mathbf{H}_X{}^T \left[\mathbf{C}^*_y(t) + \mathbf{C}_y(t) \right]^{-1} \qquad (3)$$

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:

$$\hat{X}(t) := X^*(t) + \mathbf{K}(t) \left[Y(t) - Y^*(t) \right] \qquad (4)$$

The precision of the estimate is determined by:

$$\hat{\mathbf{C}}(t) := \mathbf{C}^*(t) - \mathbf{K}(t) \; {}^Y\mathbf{H}_X \; \mathbf{C}^*(t) \qquad (5)$$

Equations (1) through (5) constitute the 5 equations of the Kalman Filter. For primitives composed of numerical properties, the Kalman filter equations provide the tools for our framework.

3.5 Eliminating Uncertain Primitives and Adding New Primitives to the Model

Each primitive in the world model should contain a confidence factor. In most of our systems we represent confidence by a discrete set of five states labelled as integers. This allows us to emulate a temporal decay mechanism, and to use arbitrary rules for transitions in confidence.

During the update phase, the confidence of all model primitives is reduced by 1. Then, during each cycle, if one or more observed token is found to match a model token, the confidence of the model token is incremented by 2, to the maximum confidence state. After all of the model primitives have been updated, and the model primitives with CF = 0 removed from the model, new model primitives are created for each unmatched observed primitives.

When no model primitive is found for an observed primitive, the observed primitive is added to the model with the observed property estimates and a temporal derivative of zero. The covariances are set to large default values and the confidence factor is set to 1. In the next cycle, a new primitive 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 primitive can be used to update more than one model primitive, such temporary spurious model primitives do not damage the estimates of properties of other primitives in the model.

In this section we have seen that in the case of numerical properties, estimation theory provides the mathematical tools for the processes which make up our perceptual framework. However, many perceptual problems require that a system be able to reason with properties which are abstracted from numerical measurements. How does this perceptual framework map onto the problem of fusing symbolic properties? This is the subject of the next section.

4. Fusion of Symbolic Properties

The framework for fusion of perceptual information described above can also be applied to symbolic properties. In this section we describe the symbolic equivalent of a dynamic world model and then present techniques for symbolic forms of the predict-match-update cycle. Such a system can be used to describe the external environment in terms of objects, relations and events.

We begin our discussion with by recalling certain basic principles about perception. We then develop the symbolic equivalent for each of the five principles which were presented in section 2.

We complete the section by considering techniques for a symbolic form of predict-match-update cycle. These techniques are illustrated with a system which is presented in section 5.5.

4.1 Philosophical Foundations

In 1781, Emanuel Kant [33] presented a theory of intelligence which has found increasing relevance as we build systems which interpret sensor data. At the lowest level, Kant distinguished the unknowable external world (called “noumena”) from perceptual stimuli resulting from that world (phenomena). Noumena are the source of perceptual stimuli and are not directly knowable. Noumena include such stable things as physical objects, as well as processes such as fire or a river. According to Kant, knowledge of the external world is limited to perceptual manifestations of noumena. Such manifestations, called phenomena, represent a partial projection of noumena. For example, our perception of the moon is limited to the pattern of light which reflects from one side of the moon to our eyes.

Kant proposed that humans organise their perceived phenomena into schemata. Schemata can represent individual phenomena as well as abstractions. Properties measured by sensors are phenomena. A vector of such properties is a form of schemata. To be faithful to Kant we should also include temporal sequences and images in our schemata.

Reasoning about the world requires an abstraction from phenomena. Abstractions can represent such things as categories of objects, relations between objects, action or events. Abstractions are represented by labels, commonly called symbols. A symbol is a "sign" which represents some "thing". The thing which is represented may be a raw perceptual phenomena or an abstraction from perceptual phenomena.

Symbols may be organised into networks connected by relations. Minsky's "frames" and many other AI tools provide modern examples of this idea. Networks of symbols and relations may also be represented by more abstract symbols. A basic problem is the definition of the vocabulary of symbols and relations for a domain. In most AI tools, the vocabulary of symbols and relations are specified by the programmer. Researchers who apply AI tools to perception usually discover that the design of the a-priori "knowledge" for a domain is an expensive and difficult task.

A popular philosophical problem in Cognitive Science and AI is the so called "symbol grounding problem". Simply stated, this problem asks: “what is the relation between the abstract symbols in a symbolic reasoning system and real world?”. In perception, this relation is provided by recognition procedures. Numerous pattern recognition techniques exist for detecting predefined categories of objects. Many vision techniques exist to detect objects and their relations.

Fusion of symbolic information is a problem of associating (or grouping) symbols which represent perceptual phenomena. As with numerical properties, an internal “symbolic” model is composed of a vector of properties. In modern terms such an association of properties is called a "schema" or frame or a unit. We will use the term "schema".

A schema is a named association of numeric and symbolic properties. A schema may contain relations to other schema (such as ISA and Part-Of hierarchies), as well as procedures for operating on its properties. A dynamic world modeling system for symbolic information is concerned with instantiating and maintaining a collection of schema which describe and predict the external environment. The maintenance process can be formed using a predict-match-update cycle. Let us examine how the principles enumerated in section 2 can be applied to symbolic interpretation.

4.2 Principles for Symbolic Fusion.

The first of our principles was stated as:

Principle 1) Primitives in the world model should be expressed as a set of properties.

Schema provide just such a representation. The properties may be symbolic labels or numerical measures.

Principle 2) Observation and Model should be expressed in a common coordinate system.

For numerical data, a common coordinate system serves as a basis for data association. In a symbolic description, this principle translates to saying that the information must somehow be associated. This association may be on the basis of spatial or temporal coordinates, or it may be on the basis of a relationship between properties. Spatial location remains a powerful technique for associating information.

Principle 3) Observation and model should be expressed in a common vocabulary.

The equivalent to a common coordinate system and common vocabulary at the symbolic level is a "context". A context is a collection of symbols and relations which are used to describe a situation. Knowing the context provides a set of symbols and relations which can be expected. This permits description to proceed by a process of prediction and verification.

Principle 4) Properties should include an explicit representation of uncertainty.

As with numerical properties, symbolic properties have two kinds of uncertainties: precision and confidence. The classic AI method for representing precision is to provide a list of possible values.

Such lists are used both for symbolic properties and for relations. Constraints are applied by intersecting lists of possible values.

Principle 5) Primitives should be accompanied by a confidence factor.

A schema is an assertion about the external environment. This assertion may be more or less certain. The degree of confident in the truth of a schema can be represented by a confidence factor. As with numerical techniques, this confidence factor applies to the association of symbols, and increases or decreases as supporting or conflicting evidence is detected. Numeric techniques for estimating confidence include probabilistic techniques [42], the MYCIN style confidence factors [8], and fuzzy logic [48]. Many researchers define a set of ad-hoc discrete confidence "states".

4.3 A Symbolic Form of the Predict, Match and Update Cycle

In this section we examine techniques for the cyclic process of predict-match and update for a symbolic description composed of schema. Our discussion is based on the use of a production system to dynamically maintain an internal model composed of schema. A similar architecture may be implemented with various other AI tools.

Prediction

The key to prediction is context. The prediction phase applies a-priori information about the context to predict the evolution of schemas in the model as well as the existence and location of new schema. An important role of the prediction phase is to select the perceptual actions which can detect the expected phenomena.

The prediction phase can be used with both pre-attentive and post-attentive recognition. In pre-attentive recognition, the prediction phase can be used to "enable" or "arm" the set of pre-attentive cues which will be processed. In post-attentive recognition, prediction triggers procedures for detecting and locating instances of expected phenomena [2].

Match

The match phase associates new perceptual phenomena with predictions from the internal model. As with numerical properties, a primary tool for matching is spatial location. Matching may also be an association based on similar properties.

Update

The update phase combines the results of prediction and observation in order to reconstruct the internal model. Production rules provide one method to express the world knowledge needed to update the internal model.

Control of Perception.

The internal model may be thought of as a form of short term memory. The computational cost of the predict, match and update phases depends on the quantity of information in this model. Imposing a fixed cycle time on the model update process has the effect of determining a limit on the quantity of information (the number of schema) that can be placed in the model at any time. This problem of controlling the contents of the model is sometimes called the problem of "control of perception [22].

Prediction plays a crucial role in control of perception. Expectations generated by the context determine what objects and relations will be fed back into the match and update cycle. But context alone is not sufficient. The current task of the system determines what information is needed and thus which part of the context is attended to by prediction, matching and updating. Pre-attentive information must also be controlled based on prediction, or else the match and update phases would be flooded with too much information.

To illustrate the use of this framework to construct a symbolic description system, in the next section we describe an object recognition system which we have recently constructed. More detailed discussions of control of perception are presented in [20] and [22].

5 Example Systems Constructed in the Framework.

The perceptual framework presented above has been refined and demonstrated through the construction of several perceptual and robotic systems over the last 10 years.

Mobile Robotics

The framework was originally developed for dynamic world modeling for a mobile robot. The first such system maintained a model of the local environment for a mobile robot using a rotating ultrasonic range sensor [14]. Shortly thereafter, the framework was adapted to the problem of surface modeling using the light-stripe data [15], leading to a 3D composite surface model based on the use of estimation theory. These techniques were then refined and used to model the environment for a mobile robot equipped with 24 ultrasonic range sensors mounted in fixed positions around a robot vehicle [19]. This system, which fuses ultrasound, odometry and a-priori information, is briefly described in section 5.1.

Passive Vision

In the construction of a real time vision system, we applied a variation of this framework to the problem of tracking edge segments in image sequences [18]. The result was a real-time "token tracker", which is described in section 5.2. The token tracking system was generalized to provide a system which incrementally models a 3D scene using 3-D edge segments [23]. The token tracker was equally adapted to provide the input to a vertical line stereo system [21], described in section 5.3. The vertical 3-D edges from the vertical line stereo system have been combined with horizontal edges from ultrasonic modeling to maintain a description of the local environment. This system is described in section 5.4.

Integrated Active Vision

Very recently, similar techniques have been demonstrated in an integrated active vision system [22]. Edge segments from the right and left image are tracked using the token tracking process. The edge models are then fed into a stereo matching process to maintain a 3-D scene model composed of edges of all orientation. Both the 2-D and 3-D models are used by a object recognition process. The 2-D, 3-D and symbolic descriptions are each based on the perceptual framework described above. The integrated active vision system is briefly described in section 5.5.

5.1 Dynamic World Modeling Using Ultrasound

In this section we briefly describe a dynamic world modeling system for a mobile robot based on 24 ultrasonic range sensors. This system is described in greater detail in [19]. The architecture of the system is shown in figure 2.

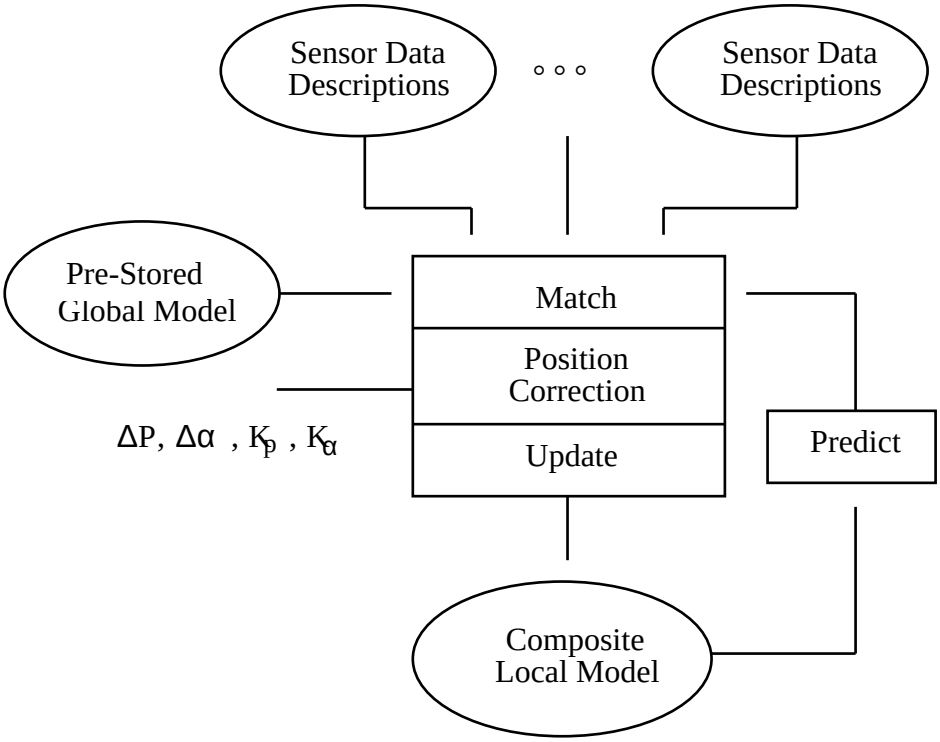

Figure 2 Dynamic World Modeling by Fusing Ultrasound, Odometry and A-priori Information

The modeling system was implemented for a rectangular mobile robot equipped with a ring of 24 ultrasonic range sensors. Each of the four sides of the vehicle carries a set of three parallel sensors. Each corner also carries a set of three sensors mounted with orientations of 30°, 45° and 60°. Range data from all 24 sensors are acquired continuously acquired and projected to "points" in world coordinates, accompanied by an uncertainty covariance. This raw range data is used as the basis for reactive obstacle detection and avoidance.

The position and orientation of the sensors with respect to the origin of the robot are defined in a sensor configuration table. For each sensor, the sensor configuration table gives:

- r The distance from the robot's origin to the sensor.
- γ The angle from the robot's axis to the sensor
- β The orientation of the sensor with respect to the robot's axis.

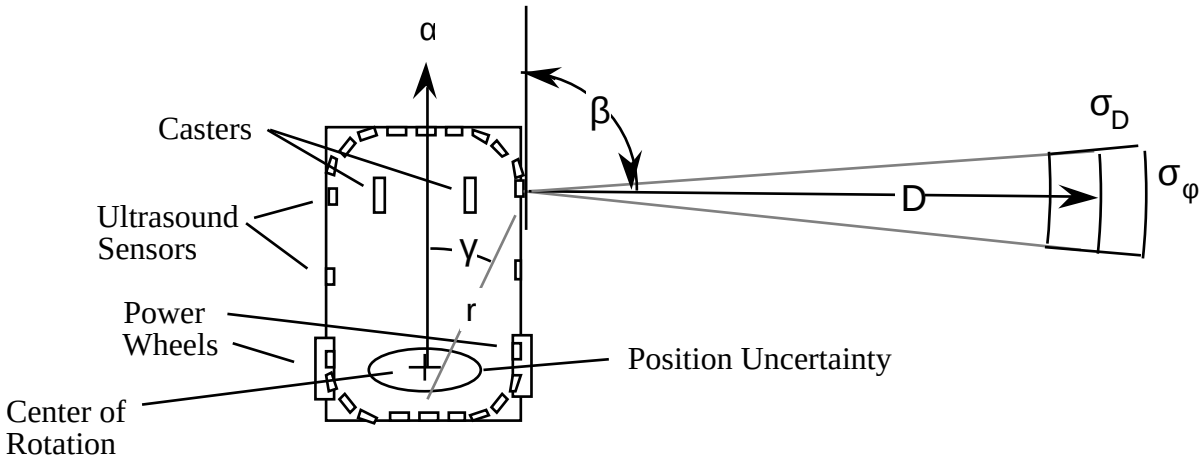

Figure 3 Projection of a Range Reading to External Coordinates.

A sensor data description process reads range measurements from the sonar table, as well as the estimated position of the robot from the vehicle controller. With this information, the depth measure, D, for each sensor, s, is projected to external coordinates, (x_S, y_S), using the estimated position of the robot, (x, y, α), as shown in figure 3.

$$x_S = x + r\ \mathrm{Cos}(\gamma + \alpha) + D\ \mathrm{Cos}(\beta + \alpha)$$
$$y_S = y + r\ \mathrm{Sin}(\gamma + \alpha) + D\ \mathrm{Sin}(\beta + \alpha)$$

All three of the angles, α, β and γ are imprecise. To simplify the analysis, let us define the absolute angle of the sensor as φ.

$$\varphi = \alpha + \beta + \gamma$$

In order to combine data from different viewpoints and sensors, we must estimate the inherent precision of the data. We have developed a model of an ultrasonic range sensor which predicts that an echo comes from an arc shaped region defined by an uncertainty in orientation, σ_φ and an uncertainty in depth σ_D.

This crescent shaped uncertainty region may be approximated by a covariance which approximates σ_φ and σ_D. This covariance is be expressed in Cartesian coordinates. The transformation from a circular coordinate system to a Cartesian coordinate system is given by:

$$X = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} D\ \mathrm{Cos}(\alpha+\beta) \\ D\ \mathrm{Sin}(\alpha+\beta) \end{bmatrix}$$

To map the covariance from circular to Cartesian coordinates, we need the Jacobian of this transformation :

$${}^{X}\mathbf{J}_{D\varphi} \triangleq \frac{\partial \begin{bmatrix} x \\ y \end{bmatrix}}{\partial (D, \varphi)} = \begin{bmatrix} \mathrm{Cos}(\varphi) & -D\mathrm{Sin}(\varphi) \\ \mathrm{Sin}(\varphi) & D\mathrm{Cos}(\varphi) \end{bmatrix}$$

The transformation of this elliptical region to Cartesian coordinates is given by:

$$\mathbf{C}_s = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} = {}^{X}\mathbf{J}_{D\alpha}{}^{T} \begin{bmatrix} \sigma_D & 0 \\ 0 & \sigma_\varphi \end{bmatrix} {}^{X}\mathbf{J}_{D\alpha}$$

From the needs of local modeling, the most important criteria for the sensor data uncertainty is that it be larger than any true errors. Thus each reading in the sonar horizon is represented by the triple $(x_S, y_S, \mathbf{C}_S)$ expressed in external coordinates.

The modeling process begins by constructing a description of the raw sensor data. This description serves to filter sensor noise by detecting range measurements which are mutually consistent. It also provides a representation with which the estimated position and orientation of the robot may be constrained. Finally, it provides a form of "object constancy" at the level of the geometric description of the environment. Such object constancy makes it possible for the mobile vehicle to react to a number of situations without requiring a symbolic interpretation of the model.

Raw ultrasonic range data, the local model and the pre-stored global model are each described as parametric line segments [16], represented with a data structure, illustrated in figure 4. This parametric representation contains a number of redundant parameters which are useful during matching and updating.

A parametric line segment is a structure composed of a minimal set of parameters and a set of redundant parameters. The minimal set of parameters are:

- c: The perpendicular distance from the segment to the origin.
- d: The distance from the perpendicular projection of the origin to the mid-point of the segment.
- θ: The orientation of the line segment.
- h: The half-length of the line segment.
- σ_θ: The uncertainty (standard deviation) in the orientation.
- σ_c: The uncertainty in position perpendicular to line segment.

The redundant parameters for a line segment are:

- P: The mid-point of the line segment in external coordinates (x, y).
- P_r: The end-point to the right of the segment.
- P_l: The end-point to the left of the segment.

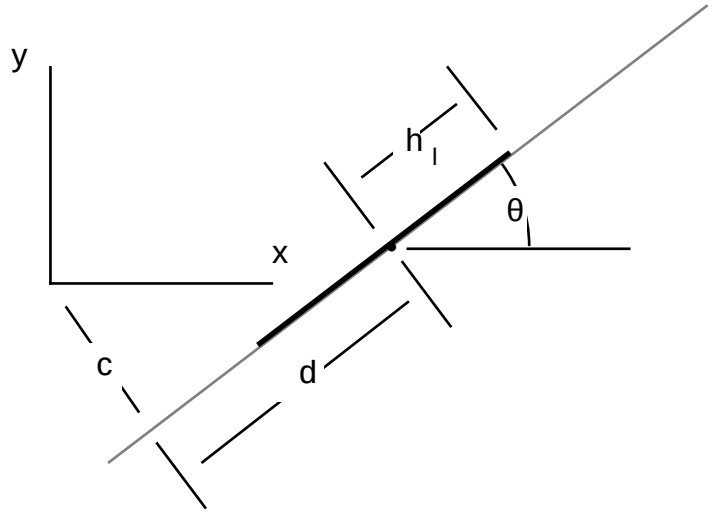

Figure 4. The Parametric Representation for a Line Segment.

The parameters (c, d) are equivalent to rotating the segment by an angle of -θ about the origin so that the mid-point lies on the x axis. That is:

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

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 in the composite model. In addition to the above parameters, line segments contain a "type" field. Type fields may contain one of the following values:

Observed: The segment was generated uniquely by observation and does not correspond to a segment in the global model.

Fixed: A segment from the global model which is known to be unmovable. The label "fixed" is used to represent walls and heavy furniture.

Movable: A segment from the global model which is known to be movable. Typically used for light furniture which might be displaced in the environment.

The type field is included on both recalled and observed line segments so that the update process can propagate this label to the composite model segments. It is expected that the vocabulary of types will evolve with the robot's functionalities.

Segments which are recalled or observed have no temporal component. Thus, by default, the temporal derivatives are zero and are not explicitly represented. In order to track segments in the composite model, it is necessary to be able to predict the location of the segments at a specific time

interval. Such tracking requires an estimate of the temporal derivative of the position and orientation of segments.

Let us express a minimum set of parameters for a segment as the vector:

$$S = [c, d, \theta, h].$$

The parameters of S are each represented by a vector a, and its covariance, C_a. The vector A is composed of an estimate of the parameter, a, plus the estimate of the first temporal derivative, a'. The covariance Ca composed of the variance of the estimate, σ_a^2, the covariance between the estimate and its temporal derivative, $\sigma_{aa'}$, and the variance of the temporal derivative, $\sigma_{a'}^2$.

$$A = \begin{bmatrix} a \\ a' \end{bmatrix} \qquad C_A = \begin{bmatrix} \sigma_a^2 & \sigma_{aa'} \\ \sigma_{aa'} & \sigma_{a'}^2 \end{bmatrix}$$

where

$$a' = \frac{\partial a}{\partial t}$$

The confidence of existence of segments is represented by a set of confidence states, noted CF, and labelled by the integers 1 through 5, as described above in section 3.5. Segments are also labelled with a unique "ID". The ID of a segment provides a label by which segments may be "referenced" by processes outside of the modeling cycle.

In summary, the non-redundant parameters of a segment in the composite model are:

Perpendicular Position:	[c, c', σ_c^2, $\sigma_{cc'}$, $\sigma_{c'}^2$]
Tangential Position:	[d, d', σ_d^2, $\sigma_{dd'}$, $\sigma_{d'}^2$]
Orientation:	[θ, θ', $\sigma_\theta 2$, $\sigma_{\theta\theta'}$, $\sigma_{\theta'}^2$]
Half Length:	[h, h', σ_h^2, $\sigma_{hh'}$, $\sigma_{h'}^2$]
Type:	One of {Observed, Fixed, Movable}
Confidence Factor:	CF
Identity:	ID

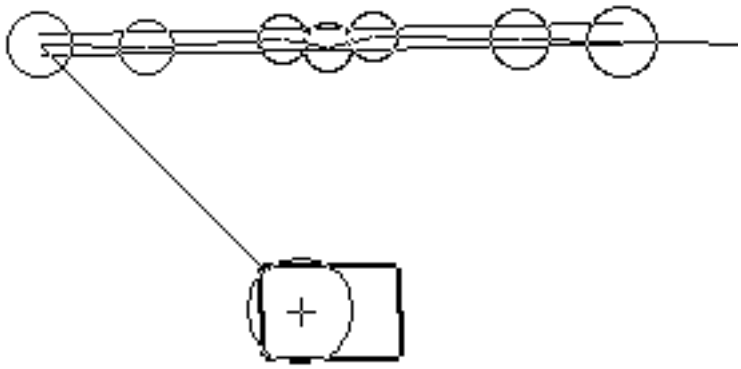

Figure 5 A vehicle (rectangle with cross) with an circular uncertainty in position of 40 cm (circle surrounding cross) is shown detecting a line segment. The top image shows a set of range data in which a line segment has been detected. The projections of ultrasound readings are illustrated as circles of radius σ_D. The projections provide the vertices of the sonar horizon. The detected segment is illustrated by a pair of parallel line at $\pm \sigma_C$.

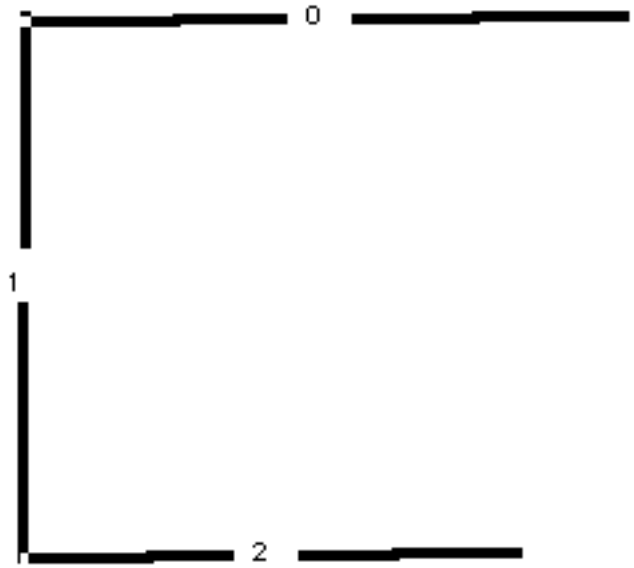

Figure 6. Continuing the example begun in figure 5, this crop from a screen dump shows three segments from the composite local model just before the model was updated by the segment detected in figure 5. An integer number indicates the identity of each segment. The width indicates the confidence (CF = 3 for all segments in this example).

5.2 2D Edge Segment Following

The problem of tracking edge lines can be solved with the same framework and the same mathematics as world modeling using ultra-sound. Our process for 2-D edge tracking has been described in [18]. This tracking process has been used in a number of our projects over the last few years. Real time hardware, capable of tracking up to 256 segments at 10 cycles per second, has recently been constructed using this algorithm [12].

Edge lines in this system are expressed in the"midpoint-Distance-Length" representation described above. For edge tracking, this representation corresponds to the uncertainty in position due to the well known aperture effect. That is, for a segment, perpendicular velocity, $\frac{\partial c}{\partial t}$ may be precisely determined, while tangential velocity, $\frac{\partial d}{\partial t}$ is inherently imprecise. Separating the mid-point position into c and d parameters allows us to represent each parameter and its time derivative as a scalar estimate and a scalar variance. Segments are represented with:

P_m	The mid-point, (x,y) of the segment.
c	The perpendicular distance of the segment from the origin.
d	The distance from the perpendicular intercept of the origin to the mid-point of the segment.
θ	The orientation of the segment.
h	The half-length of the segment.

During the update phase, we update a minimum set of of parameters:

$$\mathbf{S} = \{c, d, \theta, h\}.$$

A problem with this representation is that the parameters c and d depend on the orientation θ. The further a point is from the origin, the more an error in θ can lead to errors in c and d. For this reason we shift the origin of our image coordinates to the center of the image. We also employ a similarity measure based on the (x, y) coordinates of a segment during matching.

Prior to matching, the position of the mid-point (x, y), is computed from (c, d), to provide a redundant set of parameters. For segments to match they must have similar orientation, be roughly co-linear and overlap. The variances provide the tolerance for similarity of orientation and colinearity. The segment half length provides the tolerance for overlap. Each model token searches the list of observed tokens for its best match by testing for similar values orientation, alignment and overlap. If any test is failed, matching proceeds to the next observed token. The tests for orientation and alignment are made by testing to see if the difference in attributes is less than three standard deviations. For overlap, the half length of the segments is used as a tolerance region.

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 colinearity with the model token by comparing the distance from the mid-point of each segment to the line of the other segment:

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

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.

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 (3), (4) and (5) above. For each x S ≡ {c, d, θ, h} 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 S we compute a Kalman Gain vector as:

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

A new estimated vector is then obtained by

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

A new estimated uncertainty is obtained from

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

The confidence of tokens is again represented with an confidence state labeled with an integer value. The confidence of tokens, as well as the inclusion and elimination are managed as described above in section 4.5.

The token tracking process has become a standard tool in our systems. It has been included in a process which uses an extended Kalman filter to reconstruct a 3-D model of a scene from a single moving camera [23]. It has been used to construct a vertical line stereo system which operates at 10 hz [21] described in the following section. It has also been used in an integrated active vision system described in section 5.5 below.

5.3 Vertical Line Stereo System

The edge segment tracking process described above in section 5.2 has been used to provide input to a system for world modeling for a mobile robot based on stereo matching of vertical edge segments. The components of vertical line stereo system are shown in figure 7.

The vertical line stereo system is organized as a pipeline of relatively simple modules, as illustrated in figure 7. The first module in the system is concerned with detecting vertical edges. A cascade of simple filters is used to first smooth the image and then approximate a first vertical derivative. Filters in this cascade are composed of binomial kernel for smoothing, and a first difference kernel for calculating the derivative.

The second module in the system is responsible for edge chaining and straight line approximations. Raster scan based chaining algorithms are well suited to real time implementation. However, such algorithms are normally greatly complicated by the presence of near horizontal line segments. The restriction to detecting vertical edges yields a very simple one-pass chaining algorithm. The resulting chains are expressed as segments by a recursive line splitting algorithm.

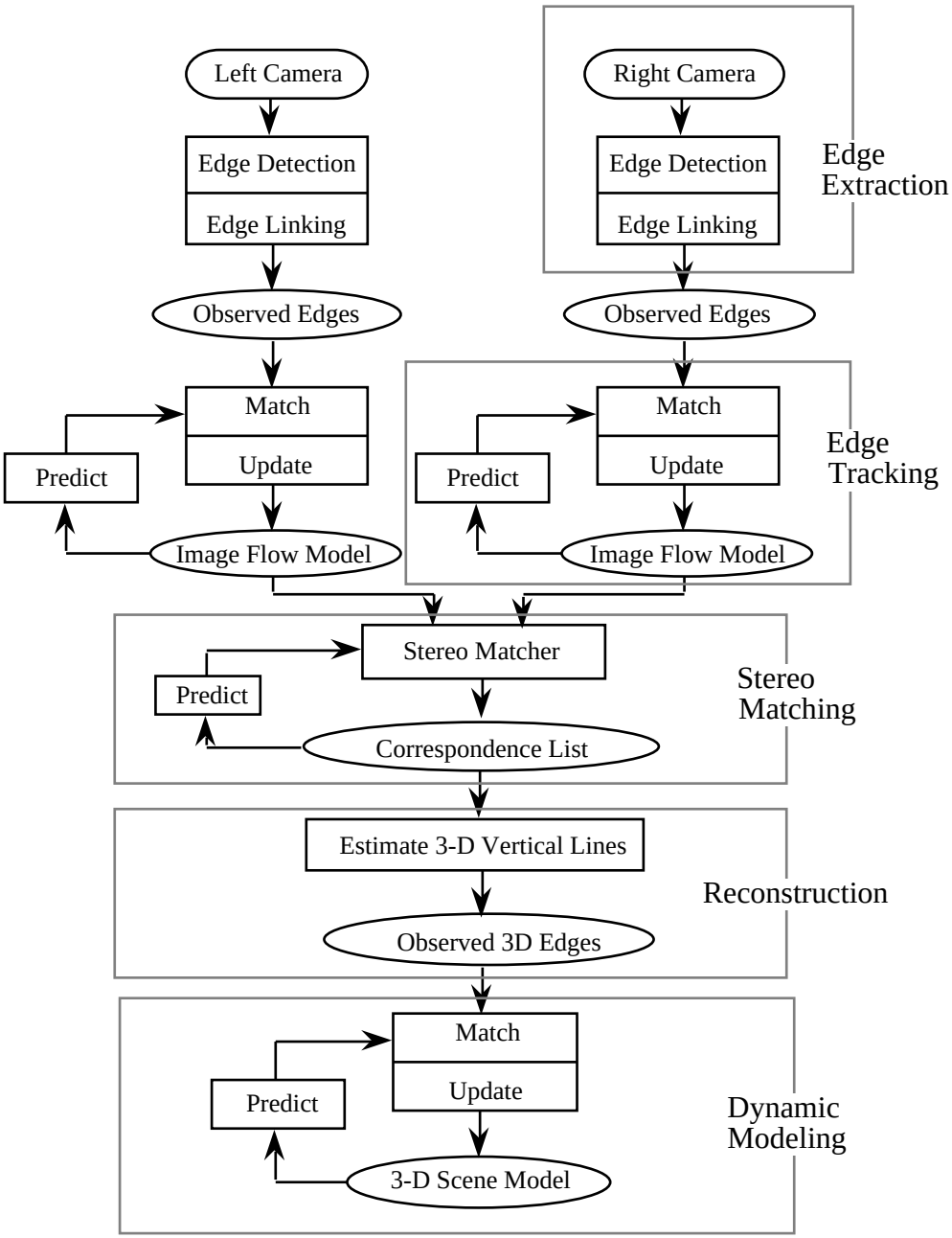

Figure 7 The System Organization for the Vertical line stereo system (from [21]).

The third module in the system describes the maintenance of an image flow model by tracking edge lines [18]. Tracking allows us to preserve the correspondence between an observed edge and information in the 3-D scene model.

Stereo matching is performed by a single pass of a dynamic programming algorithm over the entire image. Dynamic programming is based on matching the ordered sequence of edge lines in the left and right flow model. The algorithm determines all possible costs for matching the two image sequences in order. A cost measure is used based on the Mahalanobis distance for edge direction, overlap and distance from a default disparity. Matches are saved in a match list and used to generate the default disparity for subsequent cycles. The result of matching is a vertical 3-D

segment. Vertical edge segments may be used alone, or they may be fused with the model provided by ultrasonic range sensors as described in the next section.

5.4 World Modeling Using Ultrasound and Vertical Line Stereo

The vertical line stereo system described above has been combined with the world modeling system described in section 5.1 to provide a perception system for a mobile robot in an indoor environment. The architecture of this system is described in figure 8.

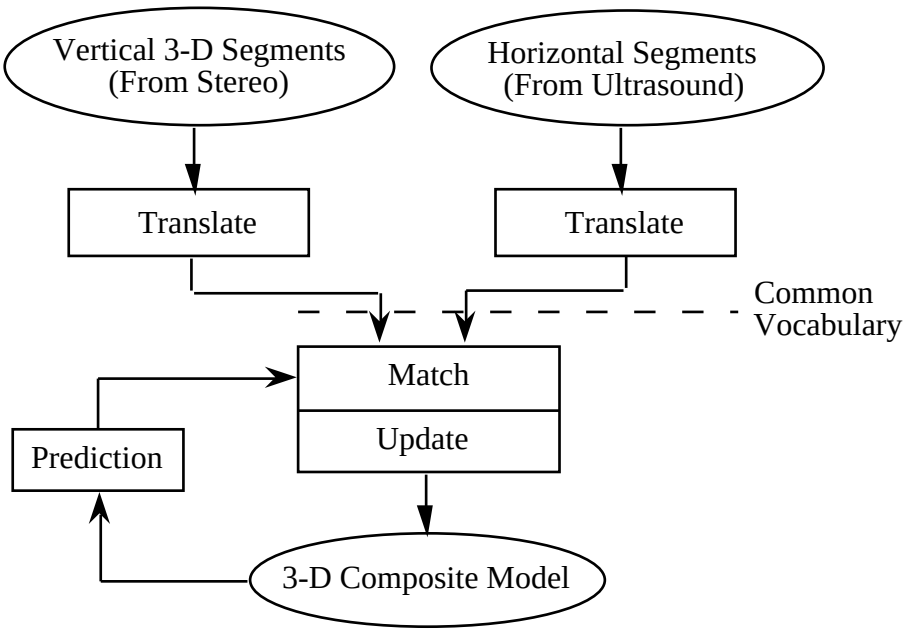

Figure 8 A System which Combines Ultrasonic Data and Vertical Line Stereo.

The 3D composite model is expressed in terms of a 3D line segment which is a composition of the vertical and horizontal line segments from the two sub-systems. These 3D composite line segments are defined in a reference frame which assume an (x-y) ground plane and notion of vertical (z). Model elements in this representation are expressed vector composed of the following parameters:

Mid-point:	$P_m = (x, y, z)$
Horizontal Orientation:	θ
Vertical Orientation (tilt):	γ
Vertical Half Length:	h_z
Horizontal Half Length:	h_{xy}
Variance of Mid-point:	$\mathbf{C}_D$
Co-Variance in horizontal orientation:	C_θ
Variance of vertical orientation:	$\mathbf{C}_\gamma$

Redundant Parameters:

Surface Normal	$N = (A, B, C)$

Distance from the Origin: D (D = –Ax – By – Cz)
Corner Points: P_1, P_2, P_3, P_4

Labels:

Filled: {True, False}
Type: {Movable, Fixed, Unknown}

CF: Confidence Factor (For surfaces in model)

This system illustrates the use of different sensors to develop partial descriptions of the perceptual primitives. Parameters that are not measured by each sensor are given a default value and uncertainty. Primitives from stereo and ultrasound are matched based on overlapping 3D position. Parameters are merged using a simple form of recursive updating rule based on the Kalman filter.

5.5 An Integrated Active Vision System

In collaboration with the partners of the ESPRIT basic research consortium BR 3038/7108 "Vision as Process", we have recently demonstrated an integrated active vision system [17]. Our system is composed of a number of independent processes that dynamically maintain a description of a limited region of a scene at different levels of abstraction. This section discusses the structure of this system and uses the object recognition components to illustrate the application of our framework to the problem of symbolic fusion.

The SAVA active vision system is composed of distributed processes, as shown in figure 9. This system includes independent processes for camera control, image acquisition and tracking, 3D modeling, symbolic interpretation, and system supervision. Each module is a continuously running cyclic process, implemented as a separate Unix process.

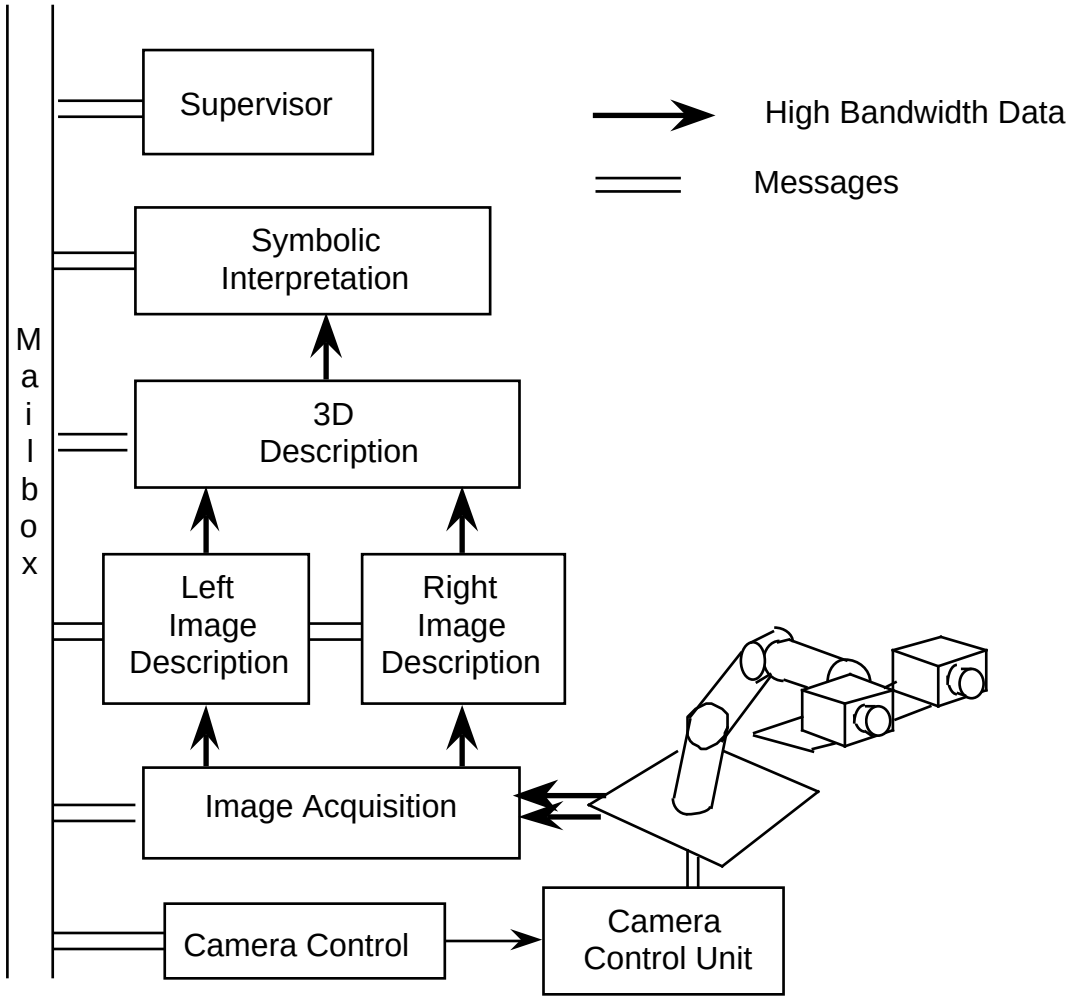

Figure 9. Components of an Integrated Active Vision System.

Descriptions of image information are maintained for both the left and right stereo cameras by tracking edge-segments from a multi-resolution region of interest. The 3-D scene description module uses information from the 2-D model to infer the 3-D form and position of structures in the scene. The symbolic scene interpretation module use information from the 2-D image description and the 3-D image description to describe the scene in terms of known objects.

The modules for maintaining a 2-D description, for maintaining a 3-D description and for maintaining a symbolic interpretation are all constructed using the predict-match-update cycle described above. We have developed a standard architecture for such modules based on this cycle.

In each module, observations are continuously acquired and integrated into a description (or model). The integration process operates by predicting the state of the model at the time of the observation. The predicted and observed information are then matched, and the result is used to update the model. In such a process the prediction from the previous model serves both to accelerate the description of additional observations, and to reduce the errors in the description.

The standard module is composed of a number of procedures that are called in sequence by a scheduler. Between each procedure call, the scheduler tests a mail box to see if any messages have arrived. Such messages may change the procedures that are used in the process, change the parameters that are used by the procedures, or interrogate the current contents of the description that is being maintained.

At the end of each cycle, the scheduler calls a set of demon procedures. Demons are responsible for event detection, and play a critical role in the control of perception. Some of the demon procedures, such as motion detection, operate by default, and may be explicitly disabled. Most of the demons, however, are specifically designed for detecting certain types of structures. These demons are armed or disarmed by recognition procedures in the interpretation module according to the current interpretation context.

We have defined a standard protocol for model interrogation based on the procedures "Find", "Verify", and "Get". In each module, these procedures provide a form of post-attentive perceptual grouping. Object recognition is based on perceptual grouping within the 2-D and 3-D descriptions using this protocol.

The objects that can be recognized by the system are described in terms of object schema. Each object schema is composed of:

Properties: A data structure composed of "slots" which can be used to represent properties of the object.

Procedures: A set of procedures that can detect instances of the object and to post an object hypothesis to the interpretation model.

A scene description is a list of object instances. Each object instance contains a list of properties for an object in the scene. The properties are dynamically updated by the recognition procedures associated with that object class. Object instances contain a unique ID and a confidence factor. Each object class is associated with a number of recognition procedures.

A recognition procedure is composed of three parts:

Trigger: A test that causes execution of the procedure.

Condition: A set of measurements based on interrogation of one or more of the description modules.

Action: A set of actions taken by the procedure, including posting an object hypothesis, updating an object's properties or changing the current context.

The organisation of the interpretation procedure is illustrated in figure 10.

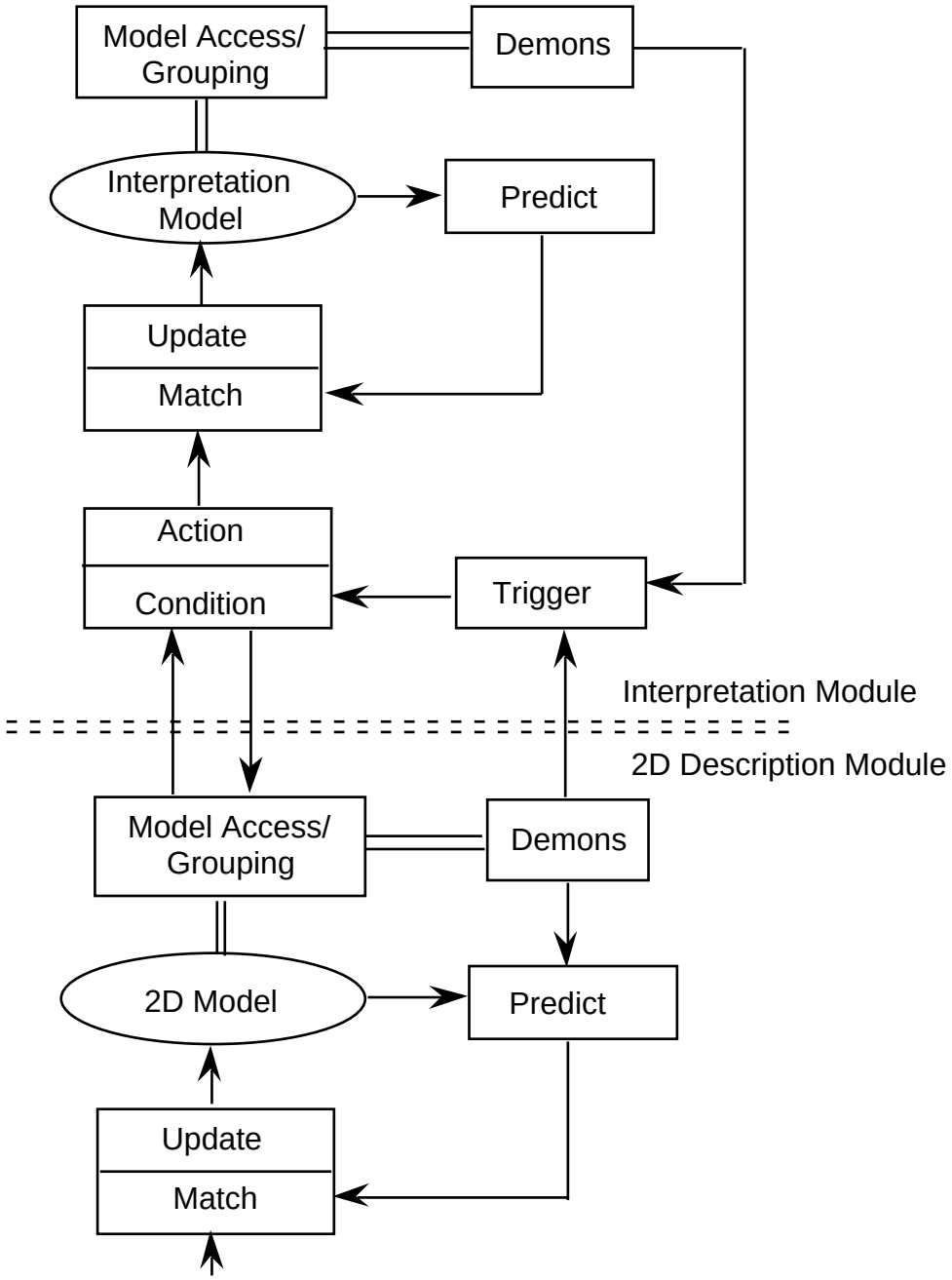

Figure 10. The interaction between the recognition and 2-D Tracking processes. Recognition procedures are triggered by demons. When triggered, a recognition procedure uses perceptual grouping to interrogate the 2D image description.

The system's knowledge base about objects is organised as a network of contexts. Each context is a list of object classes and their associated recognition procedures. For each recognition procedure, a message is sent to the appropriate demon in one of the description modules (2-D, 3-D or interpretation). When a demon detects one or more instances of its event, it sends a message to the interpretation module with information about the events. A demon continues to monitor the event in subsequent cycles without generating new messages. If the demon ceases detect its event, a new message is set to "de-activate" the recognition procedure.

The condition part of a recognition rule is composed of a set of calls to the relevant model using the primitives "Find", Verify" and "Get". These recognition rules interrogate the various description modules using the model interrogation and grouping procedures, and then create or update object hypotheses based on the result. The expressive power of the recognition procedures is based on a vocabulary of primitives for interrogation and grouping of the data in the 2D, 3D and descriptive modules.

6. Conclusions

In this paper we have presented a framework for perceptual fusion. We have then postulated a set of five principles for sensor data fusion. These principles are based on lessons learned in the construction of a sequence of systems.

The framework has been illustrated by briefly describing five systems., including:

1) A system for world modeling using ultrasonic ranging [19].

2) A system for tracking 2D edge Segments [18].

3) A system for dynamic 3D modeling using stereo [21].

4) A system for fusing vertical edges from stereo with horizontal edges from ultrasonic ranging, and

5) A system which integrates 2-D, 3-D and symbolic interpretation [22].

We have illustrated this framework by presenting techniques from estimation theory for the fusion of numerical properties. While these techniques provide a powerful tool for combining multiple observations of a property, they leave open the problem of how to manage the confidence in the existence of a vector which represents the an association of properties. Speculation that precision and confidence can be unified through a form of probability theory or minimum entropy theory have not yet been born out. These remain an important area for research.

We have then discuss the problem of the fusion of symbolic data. It is clear that there is not yet the equivalent of a "Kalman Filter" to provide a mathematical formulation for combining symbolic properties. Many people belive that a constraint based logic may provide such a formulation, but for the moment this has not yet been demonstrated convincingly. Much research remains to be done in the area of fusing properties which are represented by systems.

References

[1] N. Ayache, O. Faugeras, "Maintaining Representation of the Environment of a Mobile Robot". Proceedings of the International Symposium on Robotics Research, Santa Cruz, California, USA, August 1987.

[2] O. Boissier and Y. Demazeau, "A DAI View on General Purpose VIsion Systems, Decentralized AI 3, Werner & Demazeau, ed., Elsevier-North-Holland, June 1992.

[3] A. Blake, A. Zisserman, Visual Reconstruction, Cambridge MA, MIT Press, 1987.

[4] K. Brammer, G. Siffling, Kalman Bucy Filters, Artech House Inc., Norwood MA, USA, 1989.

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

[6] C. Brown, 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.

[7] L. Brownston, R. Farrell, E. Kant, N. Martin, Programming Expert Systems in OPS-5, Addison Wesley, Reading Mass, 1985.

[8] B. Buchanan, E. Shortliffe, Rule Based Expert Systems, Addison Wesley, Reading Mass, 1984.

[9] R. Bucy, "Optimum finite filters for a special non-stationary class of inputs", Internal Rep. BBD-600, Applied Physics Laboratory, Johns Hopkins University, 1959.

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

[11] R. Chatila, J. P. Laumond, "Position Referencing and Consistent World Modeling for Mobile Robots", Proc 2nd IEEE International Conf. on Robotics and Automation, St. Louis, March 1985.

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

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

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

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

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

[17] J.. L. Crowley, "Coordination of Action and Perception in a Surveillance Robot", IEEE Expert, Novembre 1987 (also appeared in the 1987 International Joint Conference on Artificial Intelligence).

[18] J. L. Crowley, P. Stelmaszyk, C. Discours, "Measuring Image Flow by Tracking Edge-Lines", Proc. 2nd International Conference on Computer Vision, Tarpon Springs, Fla. 1988.

[19] J. L. Crowley, "Dynamic Modeling of Free-Space for a Mobile Robot", 1989 IEEE Conference on Robotics and Automation, Scottsdale, April 1989.

[20] J. L. Crowley, "Knowledge, Symbolic Reasoning and Perception", Proceedings of the IAS-2 Conference, Amsterdam, December, 1989.

[21] J. L. Crowley, P. Bobet et K. Sarachik , "Dynamic World Modeling Using Vertical Line Stereo", Journal of Robotics and Autonomous Systems, Elsevier Press, June, 1991.

[22] J. L. Crowley, "Towards Continously Operating Integrated Vision Systems for Robotics Applications", Scandinavian Conference on Image Analysis, August 1991.

[23] J. L. Crowley, P. Stelmaszyk, T. Skordas et P. Puget, "Measurement and Integration of 3-D Structures By Tracking Edge Lines", International Journal of Computer Vision, July 1992.

[24] J. Doyle, "A Truth Maintenance Systems", Artificial Intelligence, Vol 12(3), 1979.

[25] R. Duda, R. Hart, N. Nilsson, "Subjective Bayesian Methods for Rule Based Inference Systems", Proc. 1976 Nat. Computer Conference, AFIPS, Vol 45, 1976.

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

[27] O. Faugeras, N. Ayache, B. Faverjon, "Building Visual Maps by Combining Noisey Stereo Measurements", IEEE International Conference on Robotics and Automation, San Francisco, Cal., April, 1986.

[28] C. Forgy, "RETE: A Fast Algorithm for the Many Pattern Many Object Pattern Match Problem", Artificial Intelligence, 19(1), Sept. 1982.

[29] B. Hayes-Roth, "A Blackboard Architecture for Control", Artificial Intelligence, Vol 26, 1985.

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

[31] J. Hopfield, "Neural Networks and physical systems with emergent collective computational abilities", Proc. Natl. Acad. Sci., vol-79, USA, 1982, pp 2554-2558.

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

[33] E. Kant, Critique of Pure Reason, Translated by N. Kemp Smith, New York Random House, 1958 (original work published in 1781).

[34] R. Kalman, "A new approach to Linear Filtering and Prediction Problems", Transactions of the ASME, Series D. J. Basic Eng., Vol 82, 1960.

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

[36] C. Koch, J. Marroquin, A. Yuille, "Analog neural networks in early vision", AI Lab. Memo, N° 751, MIT Cambridge, Mass, 1985.

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

[38] S. Li, "Invariant surface segmentation through energy minimization with discontinuities", submitted to Intl. J. of Computer Vision, 1989.

[39] S. Li, "A curve analysis approach to surface feature extraction from range image", Proc. International Workshop on Machine Intelligence and Vision, Tokyo, 1989.

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

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

[42] M. R. Genesereth and N. Nilsson, Logical Foundations of Artificial Intelligence, Morgan Kaufmann Publishers, 1987.

[43] T. Poggio, C. Koch, "Ill-posed problems in early vision: from computational theory to analog networks", Proc. R. Soc. London, B-226, 1985, pp.303-323.

[44] G. A. Shafer, A Mathematical Theory of Evidence, Princeton N. J., Princeton University Press.

[45] R. Smith, P. Cheesemn the Estimation and Representation of Spatial Uncertainty", International Journal of Robotics Research 5 (4), Winter, 1987.

[46] D. Terzopoulos, "Regularization of inverse problems involving discontinuities", IEEE Trans PAMI, 8, 1986, pp.129-139.

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

[48] L. Zadeh, "A Theory of Approximate Reasoning", Machine Intelligence, J. E. Haynes, D. Mitchie and L. I. Mikulich, eds, John Wiley and Sons, NY, 1979.