

UC Berkeley

Research Reports

Title

Intelligent Sensor Validation And Fusion For Vehicle Guidance Using Probabilistic And Fuzzy Methods

Permalink

<https://escholarship.org/uc/item/2nq1f8n9>

Authors

Agogino, Alice
Goebel, Kai
Alag, Sanam

Publication Date

1997

CALIFORNIA PATH PROGRAM
INSTITUTE OF TRANSPORTATION STUDIES
UNIVERSITY OF CALIFORNIA, BERKELEY

Intelligent Sensor Validation and Fusion for Vehicle Guidance Using Probabilistic and Fuzzy Methods

**Alice Agogino, Kai Goebel
Satnam Alag**
University of California, Berkeley

**California PATH Research Report
UCB-ITS-PRR-97-31**

This work was performed as part of the California PATH Program of the University of California, in cooperation with the State of California Business, Transportation, and Housing Agency, Department of Transportation; and the United States Department of Transportation, Federal Highway Administration.

The contents of this report reflect the views of the authors who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the State of California. This report does not constitute a standard, specification, or regulation.

July 1997

ISSN 1055-1425

**INTELLIGENT SENSOR VALIDATION AND FUSION
FOR VEHICLE GUIDANCE
USING PROBABILISTIC AND FUZZY MEANS**

**Final Report
PATH Project
MOU-157**

**“Intelligent Sensor Validation and Fusion for Reliability and
Safety Enhancement in Vehicle Control”**

November 15, 1996

Principal Investigator

Alice Agogino
Tel.: (510) 642-6450
Fax.: (510) 643-8982
aagogino@euler.ME.Berkeley.EDU

Graduate Research Assistants
Kai Goebel
Satnam Alag

Department of Mechanical Engineering
University of California at Berkeley
Berkeley, CA 94720

Intelligent Sensor Validation and Fusion for Vehicle Guidance Using Probabilistic and Fuzzy Means

*Alice M. Agogino, Kai Goebel, and Satnam Alag
Dept. of Mechanical Engineering
UC Berkeley*

I. Acknowledgements

Funding for this project was made available by PATH grant MOU-157. The methodology shown in this report is part of the dissertation by Satnam Alag (chapters **3-6**) and Kai Goebel (chapters 2 and **4**). We acknowledge the help of David Bellm, and Mitchell Suarez who worked on this project as part of their MS project. We also appreciate the help of Dr. Seibum Choi.

11. Keywords

Sensor Validation, Sensor Fusion, Data Fusion, Tracking, Supervisory Control, Management of Uncertainty, Reliability, Safety, Bayes Networks, Fault Detection, Diagnosis, Influence Diagrams, Fuzzy Fusion

11.1. Abstract

Vehicles in an IVHS system rely heavily on information obtained from sensors. So far, most control systems make the implicit assumption that sensor information is always correct. However, in reality, sensor information is always corrupted to some degree by noise which varies with operating conditions, environmental conditions, and other factors. In addition, sensors can fail due to a variety of reasons. To overcome these shortcomings, sensor validation is needed to assess the integrity of the sensor information and adjust or correct as appropriate. In the presence of redundant information, sensor data must be fused, accommodating the findings from the validation process.

In this report, several methods to accomplish the validation and fusion are developed. One method uses probabilistic techniques. A vector dynamic belief net (VDBN) was developed and several fusion algorithms are shown in this context. These algorithms are all based on the Kalman filter and exhibit different properties as to how well they can filter noise and how robust they are. The key is the assumption of Gaussian noise which may be modeled differently for different sensors.

Another method uses fuzzy techniques to express a confidence in sensor measurements. Non-symmetric validation curves are used to validate the sensor reading which allow one to model Gaussian as well as non-Gaussian noise. Sensor data fusion is accomplished by evaluating the system state, using fuzzy time series prediction, and weighting the sensor information by their confidence in a process called FUSVAF (Fuzzy Sensor Validation and Fusion).

A comparison of the two approaches shows that they each have their advantages in certain environmental condition. The probabilistic approach performs better in the presence of

Gaussian noise alone while the fuzzy approach behaves better in the presence of some non-Gaussian noise.

The report also includes the findings of experiments carried out to evaluate sensor performance of radar and sonar distance sensors currently used. Several environmental modes such as rain, fog, obstruction, and heavy vibration were considered.

IV. Summary

Presently, control systems in IVHS assume that data obtained from sensors are correct. However, sensor data are uncertain because of the presence of noise and sensor failure. This may result in control actions which are unsafe for passengers of the IVHS. We propose validation and fusion method, based on probabilistic and fuzzy techniques, which express a confidence in the sensor data, taking into account environmental factors and the state of the system. Sensor data fusion uses the confidence assigned to each sensor reading and integrates them into one reading. Noise and failure are filtered from the data in many cases and should result in a safety improvement of the IVHS.

Table of Contents

1. Introduction.....	5
2. Methodology	10
2.1 Fuzzy Approach	10
2.1.1 Fuzzy Sensor Validation and Fusion	10
2.1.1.1 The FUSVAF Algorithm	11
2.1.1.2 Simulations.....	16
2.1.1.3 Sensitivity Analysis	25
2.1.1.4 Recommendation and Summary	29
2.1.2 Fuzzy Sensor Validation and Fusion for IVHS	31
2.2 Sensor Validation and Fusion Using Bayesian Techniques	40
2.2.1 Methodolgy for Probabilistic Sensor Validation and Fusion.....	40
2.2.1.1 Graphical Representation	41
2.2.1.1.1 Nodes and Arcs	42
2.2.1.1.2 Modeling of Time Gauss-Markov Process	43
2.2.1.2 Inference	44
2.2.1.2.1 Inference using Message Propagation Decentralized Approach.....	44
2.2.1.2.2 Inference Using Topology Transformation.....	50
2.2.1.3 Rules for Vector Gaussian Distribution	51
2.2.1.4 Probabilistic Data Association Filter (PDAF)	52
2.2.1.5 Sensor-Data Fusion	56
2.2.1.5.1 Architectures	56
2.2.1.5.2 Sensor Fusion and Inference	58
2.2.1.5.2.1 Independent Opinion Pool	59
2.2.1.5.2.2 Independent Likelihood Pool	59
2.2.1.5.2.3 Remarks.....	59
2.2.1.5.3 Fusion Methods.	60
2.2.1.5.3.1 Using A Priori Information	60
2.2.1.5.3.2 Using A Posteriori Information	60
2.2.1.5.3.3 Fusion Using Learning.....	62
2.2.1.5.4 Comparison of Fusion Algorithms via Monte Carlo Runs	64
2.2.2 Probabilistic Sensor Validation and Fusion Applied to IVHS	77
2.2.2.1 Methodology for Intelligent Sensor Validation and Fusion for the Longitudinal Sensors	77
2.2.2.1.1 State Evolution Model	77
2.2.2.1.1.1 One State Nearly Constant Distance Model	78
2.2.2.1.1.2 Two State Nearly Constant Velocity Model	79
2.2.2.1.1.3 Three State Nearly Constant Acceleration Model	80
2.2.2.1.2 Lead Vehicle	81
2.2.2.2 Tracking Performance of Various Algorithms for Platoon Follower Case	82
2.2.2.2.1 Simulation	82
2.2.2.2.1.1 Kalman Filter VDBN with Continuous Nodes.....	83
2.2.2.2.1.2 Discrete Nodes in VDBN	86
2.2.2.2.1.3 Multiple Estimates	89

2.2.2.3 Split and Merge Maneuver	93
2.2.2.4 Lead Vehicle	97
2.2.2.5 Summary	99
2.3 Comparison and Conclusion	100
2.3.1 Comparative Simulations.....	100
2.3.2 Recommendation and Summary	105
3 Sensor Evaluation	107
3.1 Vibration Experiments	108
3.1.1 Sonar Sensor	109
3.1.2 Radar Sensor	111
3.2 Obstruction Experiments	114
3.2.1 Plastic Debris	114
3.2.2 Dirt Debris	115
3.3 Fog Experiments	116
3.4 Rain Experiment	117
4. Summary and Conclusions	118
References	119

Final Report

PATH MOU157

Intelligent Sensor Validation and Fusion for Vehicle Guidance Using Probabilistic and Fuzzy Means

*Alice M. Agogino, Kai Goebel, and Satnam Alag
Dept. of Mechanical Engineering
UC Berkeley*

1. Introduction

The Intelligent Vehicle Highway System (IVHS) envisions significant increases in safety and highway capacity through the integration of control, communication and computing technologies (Varaiya 1991; Varaiya and Shladover 1991). An early analysis indicates that with a completely automated vehicle control system, freeway lane capacity could double or even triple under improved safety conditions. In the IVHS paradigm, closely spaced automated vehicles will travel at high velocities in their respective lanes. In order to perform its basic functions (such as longitudinal control, lateral control, platooning, maneuvering techniques, e.g. lane change, automated lane exit), the IVHS requires a large number of sensors for control at the coordination layer, the engine layer, for sensing, and for communication between the vehicles and the IVHS main controller. For all subsystems to work well and reliably the IVHS system requires high sensor data fidelity. However, most generally, sensor readings are uncertain because internal and external sources add noise to the readings or cause a malfunction of the sensor altogether. Relationships between the sensor readings and the system being monitored are non-deterministic. No sensor will deliver accurate information at all times. Consequently, the safety of the IVHS system is affected. It is therefore desirable to find a way to avert the negative effect of the shortcomings of the sensors. An inconsistent sensor reading could be a result of a system failure, process failure, or sensor failure, and it is important to distinguish between these failure types. It is also important to account for sources of uncertainty and propagate them to the final diagnosis of the system state.

A framework for real-time monitoring and diagnosis of the automated vehicle in IVHS was developed in MOU-132 (Agogino et al. 1995). This framework consists of five modules within the automated vehicle control system (AVCS) hierarchy. They can be found on two layers, the regulation layer and the coordination layer. This intermediate supervisory controller combines the advantages of having access to the data of the regulation layer as well as the information from the coordination layer. It operates in every vehicle and serves the purpose of real-time monitoring and diagnoses of the components in the vehicle and between the vehicle and its environment (other objects in its neighborhood). It predicts incipient failures and recommends suitable corrective actions. This research builds on the framework and the preliminary validation and fusion methods developed in MOU-132. Therefore, we will briefly review the supervisory control methodology in the IVHS context and introduce the two new approaches taken by us for MOU-157 to build the supervisory controller.

Since safety is of prime importance in an IVHS system, it is imperative to first validate and fuse the uncertain readings obtained from the numerous sensors. Therefore, the first task to be carried out by the supervisory controller is to validate the sensor readings and get an estimate for the various parameters to be used in the monitoring and fault diagnosis part. Our experience in past work (Kim et al. 1992) has shown that real-time fault diagnosis for complex systems benefits from a

hierarchical information processing structure with selecting faults on one layer, focusing in more detail on these candidate faults at a higher layer, and finally looking for facts to confirm the ultimate diagnosis and make repair/recovery recommendations at yet a higher layer. The approaches taken include the integration of heuristics and model-based reasoning, procedures for fusing qualitative and quantitative data for developing assessments for estimation of probabilities or fuzzy memberships, and explicit reasoning about the time constraints inherent in real-time processing of large amounts of data.

We therefore proposed a multi-layer architecture for real-time monitoring and diagnosis of the automated vehicle, which consists of modules for sensor validation, sensor fusion, fault diagnosis, hazard analysis, and a safety decision advisor. Before the different modules of the system are introduced, a brief discussion will illustrate how the supervisory controller proposed here is integrated into the AVCS. Figure 1-1 shows the outline of the complex hierarchical structure of the AVCS system control architecture which – in addition to the shown link, coordination, regulation, and physical layer – consists of the network layer at the top.

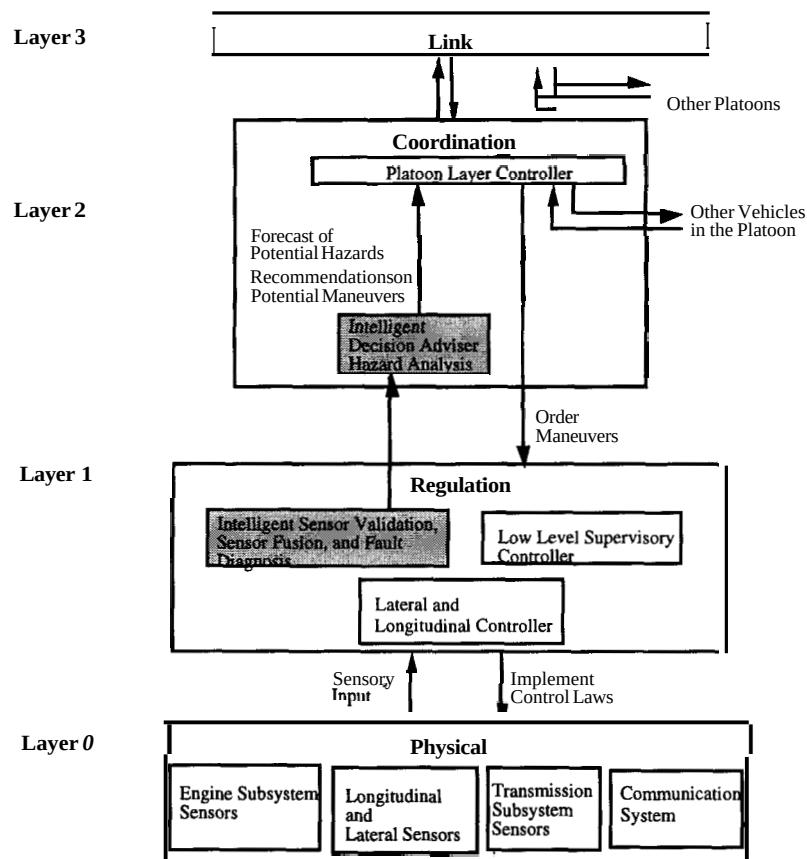


Fig. 1-1: Position of Intelligent Sensor Validation, Sensor Fusion, Fault Diagnosis, Hazard Analysis, and Intelligent Decision Advisor in the AVCS Control Hierarchy

We propose a supervisory decision advisor which considers the uncertainties in sensor readings to form a link between the coordination layer controller and the regulation layer controller and to rectify aberrant sensor readings by taking into account the information of several partly redundant sources.

Sensor validation and sensor fusion take place at the regulation layer (shaded box on the regulation layer in Fig. 1-1). Input are data from the sensors of the physical layer. The fault diagnosis of the

various subsystems is also located at the regulation layer. Some of the reasons for uncertainty in sensory information are measuring device error, environmental noise, and flaws or limitations in the data acquisition and processing systems. Extracting information from raw data is often difficult because of noise, missing data or occlusions. Phenomena may show up at disparate locations and can have a variety of time scales, from low frequency signals to high frequency vibrations. Therefore, the regulation layer seems to be the appropriate place for sensor validation and sensor fusion as it permits access to various sensor values at the same time. On the coordination layer, output from the sensor validation and fusion module as well as from the fault diagnosis module are used to perform hazard analysis and intelligent decision making because this central location within the control architecture allows for integration of all relevant information.

The modules are displayed isolated in Fig. 1-2. The input and output for each module are shown on the right hand side. On the lowest layer is the sensor validation module. It is responsible for detecting sensor failures and sensor faults. After the validation, in the case of multiple sensors, or a group of sensors measuring a set of related quantities, sensor fusion takes place. Redundancies of the sensors as well as correlation of processes measured by different sensors are utilized to find fused or corrected sensor values. In **MOU-132** we investigated the suitability of the Probabilistic Data Association Filter (PDAF). Based on the results of the sensor validation and fusion modules, a fault diagnosis module looks at potential failures of the various subsystems and calculates their respective probability or fuzzy likelihood. Here, subsystem influence diagrams are used to capture the influences of various failures on subsystem parameters. This information (either Bayesian or fuzzy) will then be used in a hazard analysis module to compute the likelihood of various hazards. Finally, an intelligent decision advisor is proposed which provides recommendations on potential maneuvers and other actions to the coordination layer controller. This decision advisor needs to arrive at the optimal decision in real-time. Since reaction time has to be small, optimization techniques which are generally very slow are trained off-line to obtain optimal responses for various scenarios. These can be implemented by means of look-up tables and pattern recognition systems and then used on-line in real-time. In this way a link between the vehicle sensors and the coordination layer is achieved. This will improve the operation of the system in diverse and adverse conditions.

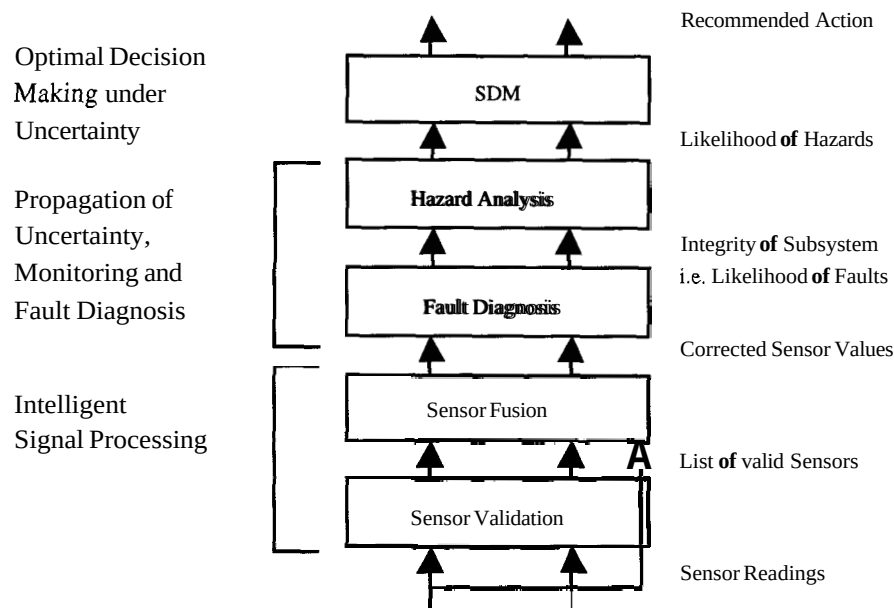


Fig. 1-2: Framework of 5 modules for sensor validation, fusion, fault diagnosis, hazard analysis, and the safety decision maker

For MOU-157, we developed new algorithms using Bayesian Kalman filtering techniques, and the Fuzzy Sensor Validation and Fusion (FUSVAF) algorithm. The effectiveness of the simulation study is greatly dependent on how reliably the longitudinal sensors are modeled. For this purpose various tests, both under static and dynamic conditions, were carried out to characterize the various longitudinal sensors (Agogino et al. 1995). These tests were carried out to model the longitudinal distance sensors used in the simulation. Figure 1-3 shows the spacing error during the maneuver for the first follower vehicle in the platoon, when there was no noise in the sensors and when the radar sensor was used to obtain the longitudinal distance. As can be seen the addition of noise leads to poor tracking performance. The sum squared error without noise was 0.67 while in the presence of noise was 196.58. Noise also affects the actuator response. A higher degree of control activity is required in the presence of sensor noise (Garg 1995). This higher degree of control activity has a negative effect on passenger comfort and safety. Therefore, it is very important to develop an effective methodology for estimating the longitudinal distance which obviates the negative effects of sensor noise and sensor failure.

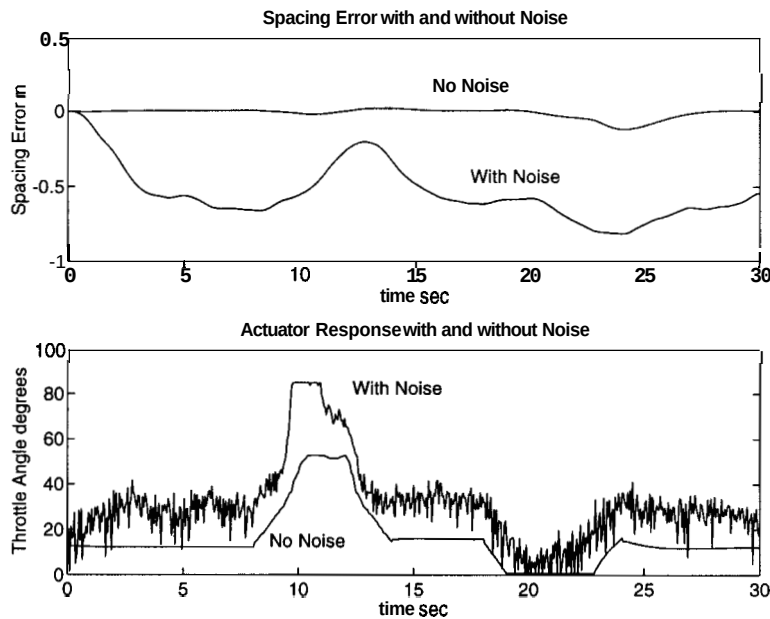


Fig. 1-3: Spacing error and actuator response for the first follower vehicle in the platoon with and without noise

2. Methodology

This section introduces the methodology used for sensor validation and fusion. To begin, a fuzzy approach is introduced in Section 2.1 followed by a probabilistic approach in Section 2.2. The two methods are compared and recommendations for their use are made in Section 2.3.

2.1 Fuzzy Approach

Section 2.1.1 introduces the new algorithm for fuzzy sensor validation and fusion (Goebel 1996). The fusion algorithm uses confidence values obtained for each sensor reading from validation curves and performs a weighted average fusion. With increasing distance from the predicted value, readings are discounted through a non-linear validation function. The predicted value uses Exponential Weighted Moving Average (EWMA) time series predictor with adaptive coefficients. These coefficients are governed by fuzzy membership functions which are learned via machine learning techniques using genetic algorithms (GA) with experimental data as training data. Simulations motivate the use of the chosen fuzzy rules. A sensitivity analysis shows the robustness of the chosen parameters.

2.1.1 Fuzzy Sensor Validation and Fusion

The FUSVAF algorithm is described (Goebel and Agogino 1996). This algorithm makes use of a Fuzzy Exponential Weighted Moving Average (FEWMA) time series predictor, dynamic validation curves which are determined by sensor characteristics, and a fusion scheme which uses confidence values for the measurements, the predicted value, the measurements, and the system state. Inputs to the FUSVAF algorithm include the raw sensors measurements. The output is the corrected value which can be used for the machine level controller as well as for supervisory control tasks (Agogino, Alag, and Goebel 1995). Additional information regarding the performance of individual sensors or actuators can be provided for use in a diagnostic module. Fig. 2.1.1-1 shows the architecture of the FUSVAF algorithm.

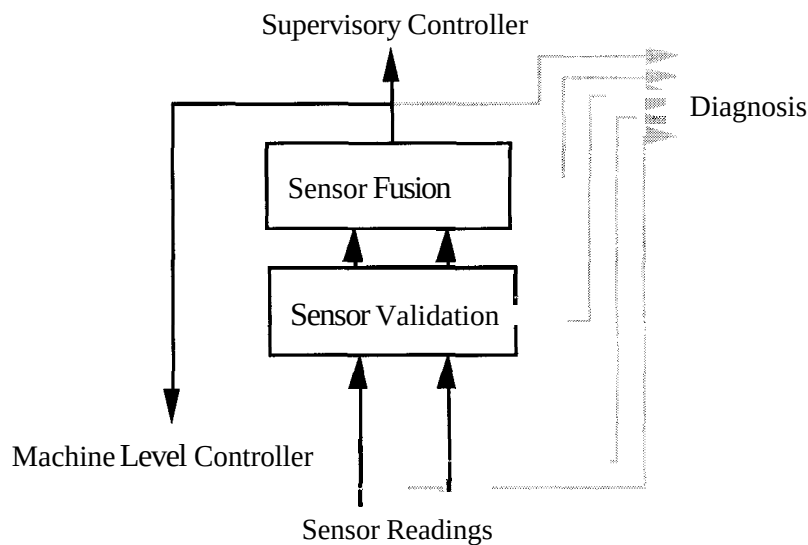


Fig. 2.1.1-1: Sensor Validation and Sensor Fusion Architecture

After the operation principle of the FUSVAF algorithm is introduced, calculating confidence values is explained. This is followed by the fusion algorithm. Next, the FEWMA algorithm is explained followed by a series of simulations motivating the use of the FEWMA algorithm. Then learning of the parameters is presented using genetic algorithms. A number of simulations shows the strengths and weaknesses of the algorithm. A sensitivity analysis points towards a feasible space for the parameters. Section 2.1.I concludes with a discussion about the algorithm.

2.1.1.1 The FUSVAF Algorithm

The validation and fusion algorithm works in the following manner: incoming sensor readings are validated using the validation gate and the old fused value. This fused value is then used to assess the state of the system expressed by $\mathbf{a}$. It is also used for prediction which in turn is necessary to perform the validation of the next time step. The fused value is also used for the machine level controller and supervisory control tasks. The algorithm is displayed in Fig. 2.1.1.1-1, where z^{-1} denotes an unit delay.

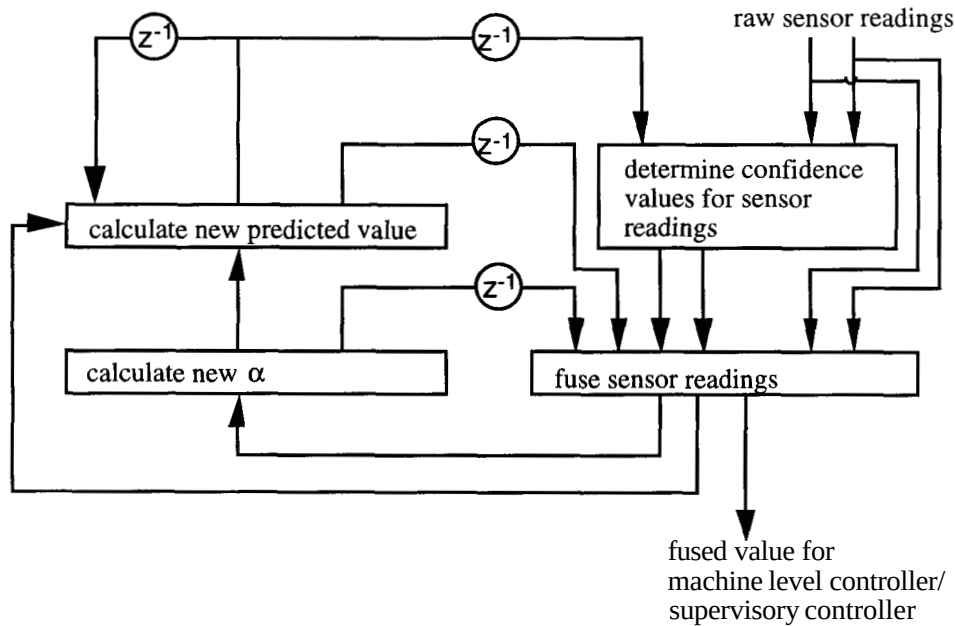


Fig. 2.1.1.1-1: Fuzzy sensor validation and fusion Algorithm

We begin by explaining how to obtain the confidence values which are assigned to all sensor measurements. This confidence value depends on the specific sensor characteristics, the predicted value, and the physical limitations of the sensor value. The assignment takes place in a validation gate which is bounded by the physically possible changes of the system it can undergo in one time step. This is an application of the **ASV** (Kim et al. 1992) which proposes limit checking for new sensor values. Due to the inertia of the system, it can only change within certain physical limits. If the sensor reading shows a change beyond that limit, it cannot be a correct value. In the context of validation gates, these limits are the boundaries for the validation gate. Sensor readings outside the validation gate are assigned a confidence value of 0 since they do not make sense. Within the region, a maximum value of 1 will be assigned to readings which coincide with the predicted value. The curve between the maximum and the two minima is dependent on the sensor behavior. Generally, this is a non-symmetric curve which is wider around the maximum value if the sensor is known to have little variance and narrower if the sensor exhibits noisy behavior. The curves change dynamically with the operating conditions which allows to capture the change in behavior

of the sensor over its operating span. Other effects resulting in changing sensor performance can be integrated into the validation curves as well. If, for example, environmental conditions, such as a change in temperature or humidity, are known to affect the sensor, the information can be easily integrated either in a crisp manner or through a fuzzy logic rule based approach utilizing information about the external effects. One such rule could for example be

IF humidity large THEN use narrow base for validation curve for sensor X.

An example for a functional relation which captures the notion of changing confidence with increasing range is

$$a_i(z) = \frac{a_{i_0}}{z}$$

where

a_i determines the width of the validation curve for which examples are listed below.

a_{i_0} is a constant width of the validation curve which captures the characteristic of the sensor in question and is determined empirically.

z is the measurement

If (through calibration) a sensor has been established to be more trustworthy than another sensor, i.e., its variance and bias are lower than another sensor used, parameter a should be larger and vice versa. A choice for validation curves $v(z)$ for a particular situation could be a piecewise bell curve of the form

$$v(z) = e^{-\left(\frac{\hat{x}-z}{a_w}\right)^2}$$

where parameters a have to be chosen separately for the left and right curves. The curves then need to be scaled from 0 and 1 between the validation border and the predicted value $\hat{x}$. The resulting assignment for confidence values is

$$\sigma = \begin{cases} 0 & z < v_{\text{left}} \\ \frac{e^{-\left(\frac{\hat{x}-z}{a_{\text{left}}}\right)^2} - e^{-\left(\frac{\hat{x}-v_{\text{left}}}{a_{\text{left}}}\right)^2}}{e^{-\left(\frac{\hat{x}-v_{\text{left}}}{a_{\text{left}}}\right)^2} - e^{-\left(\frac{\hat{x}-v_{\text{right}}}{a_{\text{right}}}\right)^2}} & v_{\text{left}} \leq z \leq \hat{x} \\ \frac{1 - e^{-\left(\frac{\hat{x}-z}{a_{\text{right}}}\right)^2}}{1 - e^{-\left(\frac{\hat{x}-v_{\text{right}}}{a_{\text{right}}}\right)^2}} & \hat{x} < z \leq v_{\text{right}} \\ 0 & z > v_{\text{right}} \end{cases}$$

where

σ is the confidence value for a particular sensor

v_{right} and v_{left} are the right and left validation gate borders, respectively

a_{right} and a_{left} are the parameters for the left and right validation curve

z is the sensor reading

$\hat{x}$ is the predicted value calculated by the Fuzzy Exponential Weighted Moving Average (FEWMA) predictor (explained below).

Other curves are of course acceptable such as the generalized bell curve (Jang 1993)

$$v(z) = \frac{1}{1 + \left| \frac{\hat{x} - z}{a_w} \right|^{2\omega}}$$

with the same scaling requirements. However, because the piecewise bell curve allows modification of the left and right side in a more flexible manner than the generalized bell curve it is the function of our choice (Goebel, 1996).

A validation gate is shown in Fig. 2.1.1.1-2.

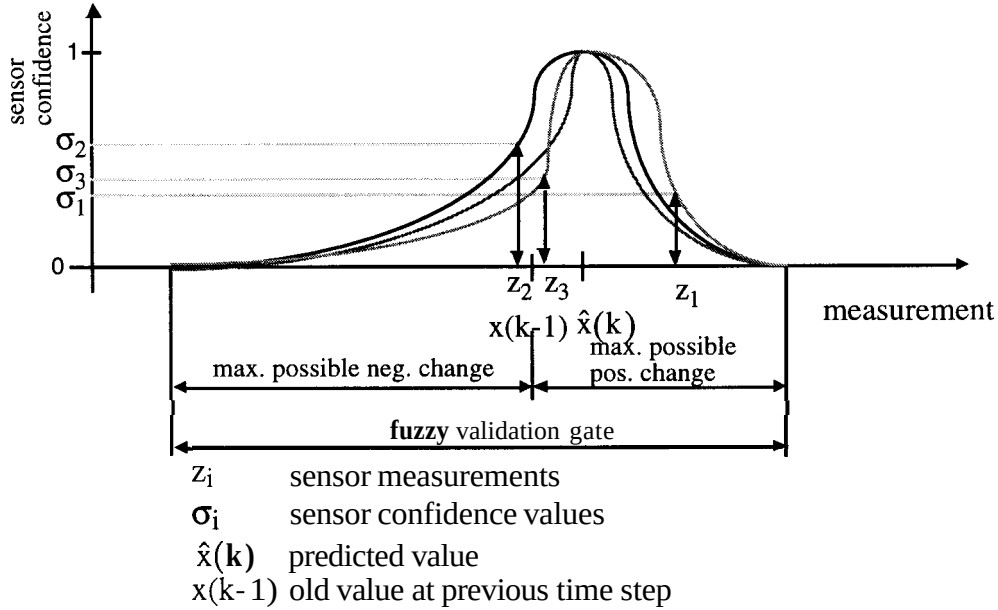


Fig. 2.1.1.1-2 Validation gate for confidence value assignment

The fusion is performed through a weighted average of confidence values and measurements plus a term which includes the predicted value weighted by $\mathbf{a}$ and a constant scaling factor ω . The sum of the confidence values times the measurements rewards measurements closest to the predicted value the most, depending on the validation curve which expresses a trust in the operation of each sensor through the design of its shape. Measurements further away are discounted. The confidence value drops to 0 at the border of the validation so that measurements beyond the gate borders are not taken into account. The term including the predicted value permits the calculation of fused values even when all sensors have values outside the validation region. Furthermore, it includes information about the state of the system through the use of the adaptive parameter $\mathbf{a}$. If the system is in a steady state, $\mathbf{a}$ is large and past history should be integrated more. When the system is in a transient state, however, $\mathbf{a}$ is small and the predicted value has less weight thus reducing lag induced by the term. The calculation of $\mathbf{a}$ will be explained with the FEWMA algorithm later. Parameter ω is a scaling factor which has to be tuned to the system at hand. Since $\mathbf{a}$ will be used in calculating the predicted value without the use of ω , the two parameters are not integrated into one value. The fusion algorithm is

$$\hat{x}_f = \frac{\sum_{i=1}^n z_i \sigma(z_i) + \frac{\alpha \hat{x}}{\omega}}{\sum_{i=1}^n \sigma(z_i) + \frac{\mathbf{a}}{\omega}}$$

where

$\hat{x}_f$: fused value

z_i : measurements

σ : confidence values

$\mathbf{a}$: adaptive parameter representing the system state

ω : constant scaling factor

$\hat{x}$: expected value.

Note that if all sensors lie on one side of the predicted value, the fused value will also be pulled to the same side. This ensures that evidence from the sensors is closely followed yet discounted the further it gets away from the predicted value.

The motivation for using a time series predictor with adaptive parameter stems from the trade off in the design of time series prediction with fixed parameter $\mathbf{a}$ between responsiveness, smoothness, stability, and lag of the predictor. The state equations of the system model used for the FEWMA predictor are described by

$$x(k+1) = x(k) + w(k)$$

$$z(k) = x(k) + v(k)$$

where

$w(k)$ describes the system perturbation

$v(k)$ represents the observation noise.

The standard EWMA predictor has the form

$$\hat{x}(k+1) = \alpha \hat{x}(k) + (1-\alpha)y(k).$$

Note the connection to the 1-state Kalman filter which has the same form

$$\hat{x}(k+1) = \hat{x}(k) + W(y(k) - \hat{x}(k))$$

$$= (1-W)\hat{x}(k) + Wy(k).$$

Let $\alpha = 1 - W$

$$\Rightarrow \hat{x}(k+1) = \alpha \hat{x}(k) + (1-\alpha)y(k).$$

If the parameter $\mathbf{a}$ is set to a fixed value the degree to which new information from sensor readings is used to update the system state is also fixed. This means that the predictor will usually lag behind the true state to some degree when $\mathbf{a}$ is too large or that the predictor follows the data so closely that noise is not filtered out when $\mathbf{a}$ is too small. Generally there are two states of the system for which a different design for $\mathbf{a}$ would be desirable. If the system is in a more or less steady state, then noise filtering should be the primary task. If, on the other hand, the system is in some sort of transient state, it is more important to track the system change. Therefore, $\mathbf{a}$ should be adaptive according to the system change. A later section will further motivate the reasoning behind the choice for $\mathbf{a}$. To summarize, $\mathbf{a}$ should be large when the system is in a steady state and

it should be small when the system is in a transient state (Khedkar and Keshav 1992). These linguistic statements can be interpreted as fuzzy rules and cast in the framework of fuzzy logic. The predictor has to rely on the measurements alone and has no extra help which will tell it whether the system is in a steady or a transient state. Conclusions about the state of the system have to be made solely on the basis of past and current observations. Assuming that noise is smaller than the change of the system, one can infer the state of the system from changes in the measurements. This means that if the changes are small, the system is likely to be in a steady state. Otherwise, the system is in some dynamic (or transient) state. Since we would like to have a large α when the system is in a steady state and a small α otherwise, the following fuzzy rules can be formed:

- IF change of readings small THEN α large
- IF change of readings medium THEN α medium
- IF change of readings large THEN α small.

In other words if the system is steady, the change in the readings is due more to noise and less to a system change; therefore more weight should be given to the past history and less to the new reading which is likely corrupted by noise (Khedkar and Keshav 1992). If on the other hand the system is in a transient state, then the change of readings will be due more to the change of the state and less due to noise. As a result, more weight should be given to the incoming reading and less to the past history to allow good responsiveness and little lag.

The membership functions “small”, “medium”, and “large” have to be designed. Using the standard assumption of triangular shaped functions and maximum overlap for the functions (Khedkar and Keshav 1992), only two parameters have to be specified: one for the fuzzification and one for the defuzzification. The functions are shown in Fig. 2.1.1.1-3.

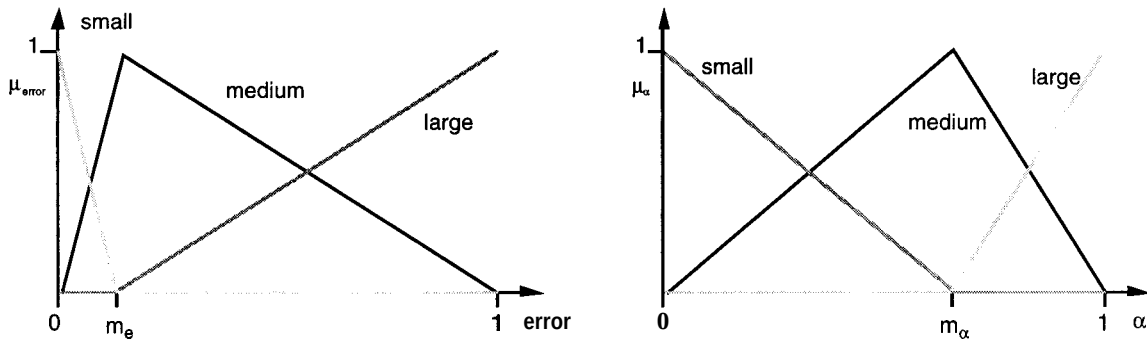


Fig. 2.1.1.1-3: Membership functions for fuzzy rules

The parameter m_e determines the vertex of the triangular membership function “medium error”. This parameter also sets the point where membership functions “small error” and “large error” change their shape, i.e., where the membership function for “small error” reaches 0 and changes course to a straight line with value 0. The membership function for “large error” changes from 0 to an increasing line. Similarly, the parameter m_α defines the shape of the membership functions for small, medium, and large α . Tuning of parameters m_e and m_α is done through a suitable optimization technique. We chose genetic algorithms because of its ease of use (Goldberg 1989). It should be noted that the choice of m_e and m_α which will lead to a functioning system can come from a very wide array. Further details can be found in the next section. Learning data have to be conditioned to the environment in which the system is supposed to operate, taking into account all possible scenarios. Experimental data are desirable, but in practice not always obtainable. In that

case, the sensor behavior has to be simulated as accurately as possible which we did using the sensor models introduced earlier.

2.1.1.2 Simulations

This section shows simulations dealing with the FEWMA algorithm, further motivating the use of an adaptive time series predictor by first looking at several algorithms with fixed α and later at various designs for the FEWMA. Fig. 2.1.1.2-1 shows a predictor with α fixed to 0.9. Measurements are assumed to be perfect. The predictor joins the data (which were held at a steady value) relatively slowly in an exponential fashion.

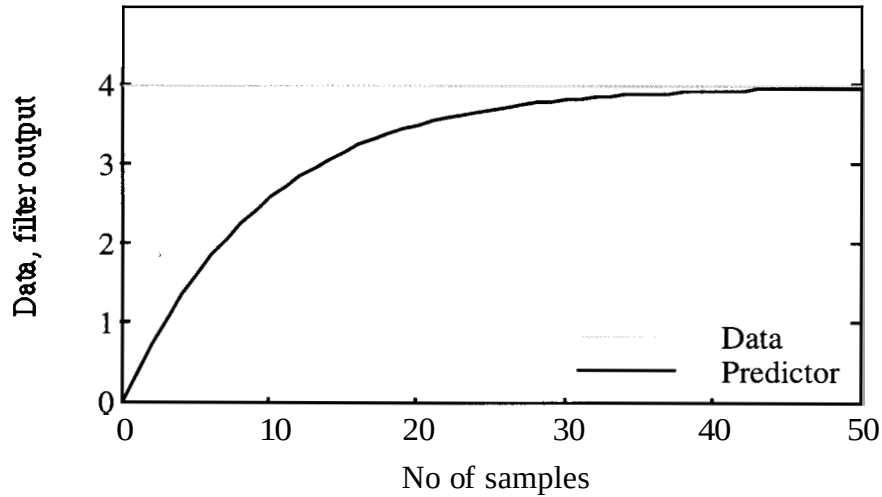


Fig. 2.1.1.2-1: Predictor with α fixed at 0.9; SSE=84.2083

Fig. 2.1.1.2-2 shows a predictor with fixed value set to $\alpha=0.3$. The predictor approaches the curve much faster than the one shown in Fig 2.1.1.2-1.

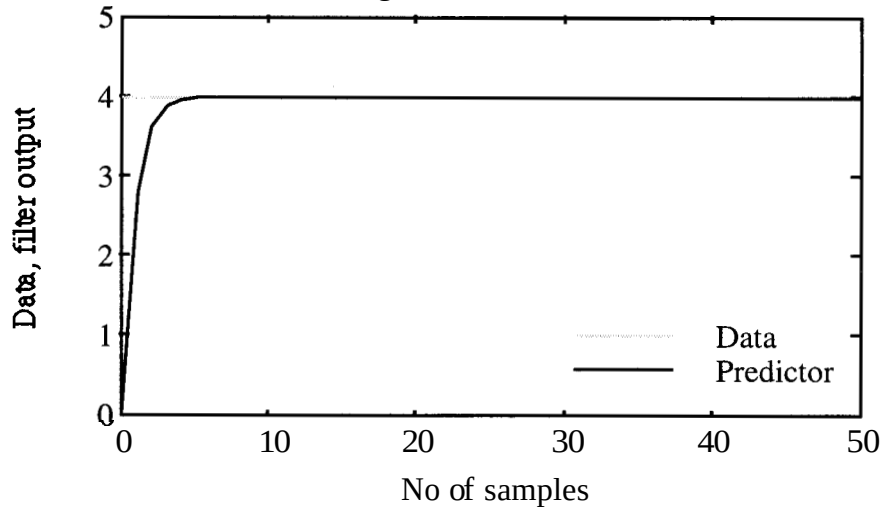


Fig. 2.1.1.2-2: Predictor with α fixed at 0.3; SSE=17.5824

From this one could conclude -prematurely- that a small α is always desirable. However, if the data are subject to noise, which is the case in all real systems, the same predictor with $\alpha=0.3$ is

very responsive to that noise as shown in Fig. 2.1.1.2-3. This responsiveness to noise is undesirable in most systems.

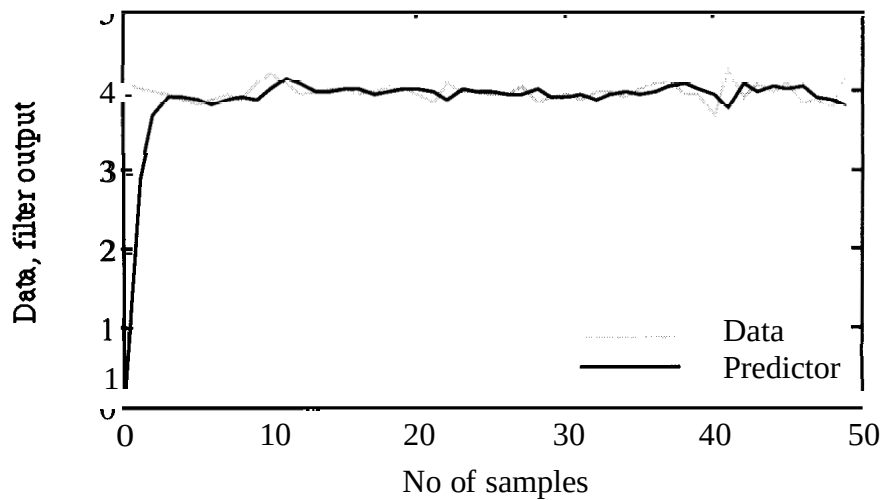


Fig. 2.1.1.2-3: Predictor with α fixed at 0.3 in the presence of noise; SSE=17.5674

Using a higher α will filter the noise much better in the region where the predictor has caught up with the measurements as shown in Fig. 2.1.1.2-4.

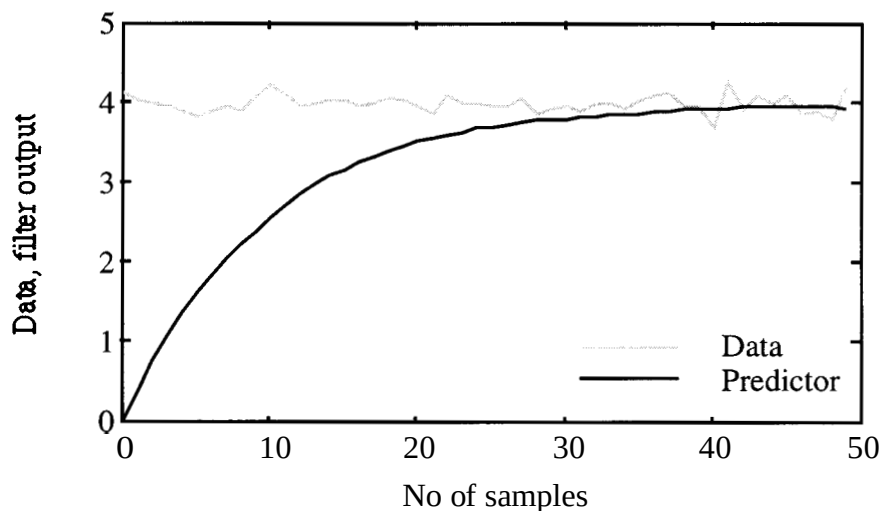


Fig. 2.1.1.2-4: Predictor with α fixed at 0.9 in the presence of noise; SSE=84.1979

Once the predictor reaches the line, the curve remains very smooth. This is a desirable feature for most control purposes. However, it already shows one of the trade offs between a fast and smooth response. Moreover, if the system is subject to change as for example shown in Fig. 2.1.1.2-5, the need for fast response becomes even clearer. Here the system input was changed from a steady period to a sinusoidal curve and back to steady. The predictor again uses an α of 0.9 while the measurements are assumed to be perfect. The predictor behaves well in the stationary portion after it has caught up with the measurements but exhibits an undesirable lag in the dynamic part of the system. The sum squared error (SSE) of the predictor is 98.2122.

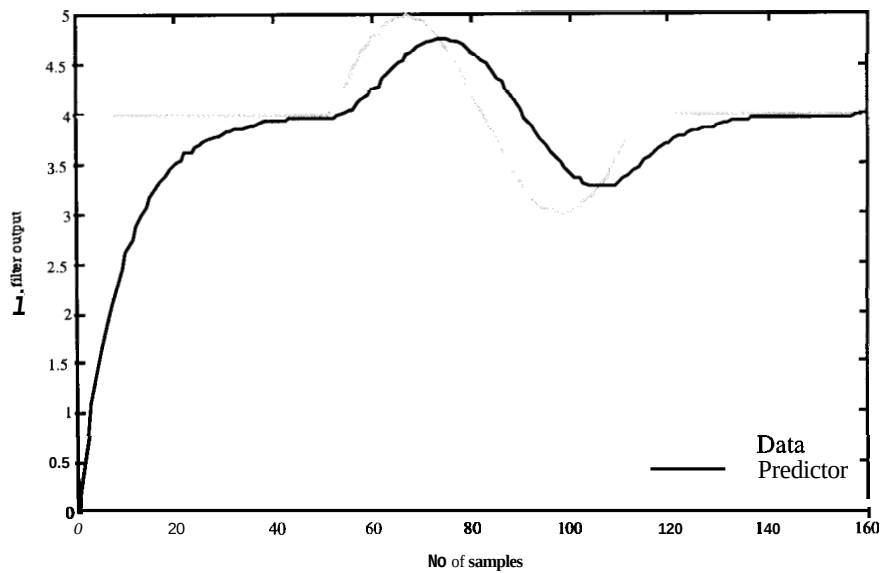


Fig. 2.1.1.2-5: Predictor with α fixed at 0.9, data input varying; SSE=98.2122

Setting $\alpha=0.3$ again allows fast tracking of the system as shown in Fig. 2.1.1.2-6. The SSE was 18.2044, i.e., much smaller than with larger α .

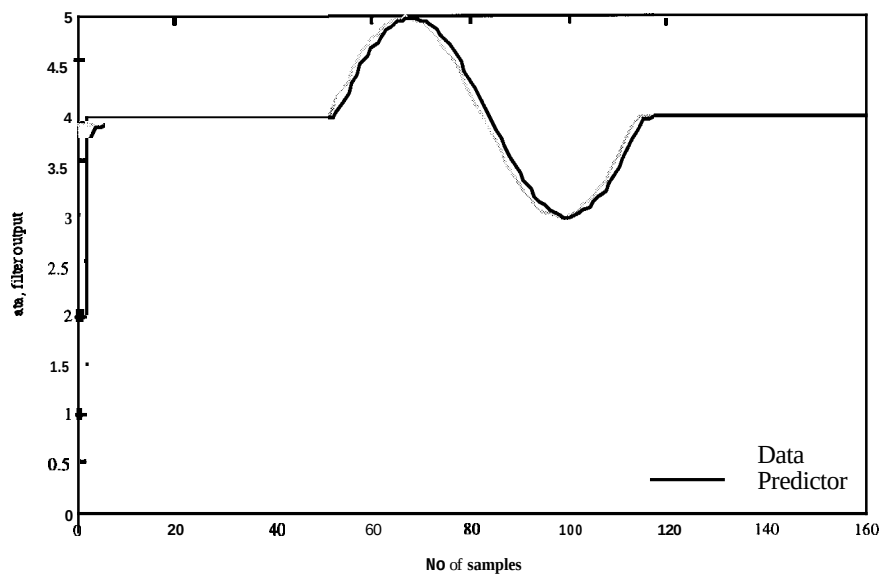


Fig. 2.1.1.2-6: Predictor with α fixed at 0.3, data input varying; SSE=18.2044

However, although the predictor now follows the true state more closely, it is much more receptive to noise as shown in Fig. 2.1.1.2-7. Although the SSE is still very small (SSE=19.4408), the predictor essentially follows the measurements even where better (i.e., smoother) filtering is desired.

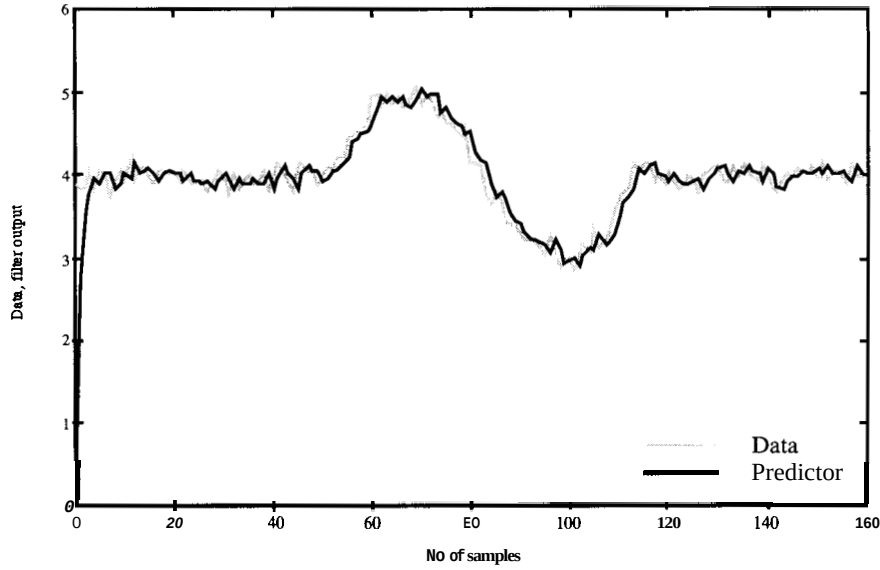


Fig. 2.1.1.2-7: Predictor with α fixed at 0.3; superimposed noise; SSE=19.4408

From these observations, the following strategy can be concluded about a better predictor: the predictor should be able to filter out noise when the system is in a steady state and to make it more receptive when it is in a transient state. It should also be able to move quickly towards the measurements, at least initially. This means that α has to be flexible as discussed earlier (Khedkar and Keshav 1992). Fig. 2.1.1.2-8 shows the input data, the response of the fuzzy predictor, and the change of α . The predictor line moves rapidly to the steady data line and also responds quickly to the system change. Also shown is the change of α which starts with a low value to catch up with the steady line and then assumes a large value while the system remains in the steady phase. It decreases when the system moves from steady to transient and reverts to a large value when the system resumes its steady path. Parameter α is small where the system has saddle points because the controller believes that the system will settle to a steady state since system changes are smaller. However, it reacts quickly after the system has changed its directions. The SSE is 0.5088 which is much smaller than the error with fixed α . The parameters chosen for Fig. 2.1.1.2-9 were $m_c=0.1$ and $m_o=0.99$. The SSE is here 0.5088. Parameter tuning is an issue which was addressed by machine learning techniques using genetic algorithms to learn the parameters of the system.

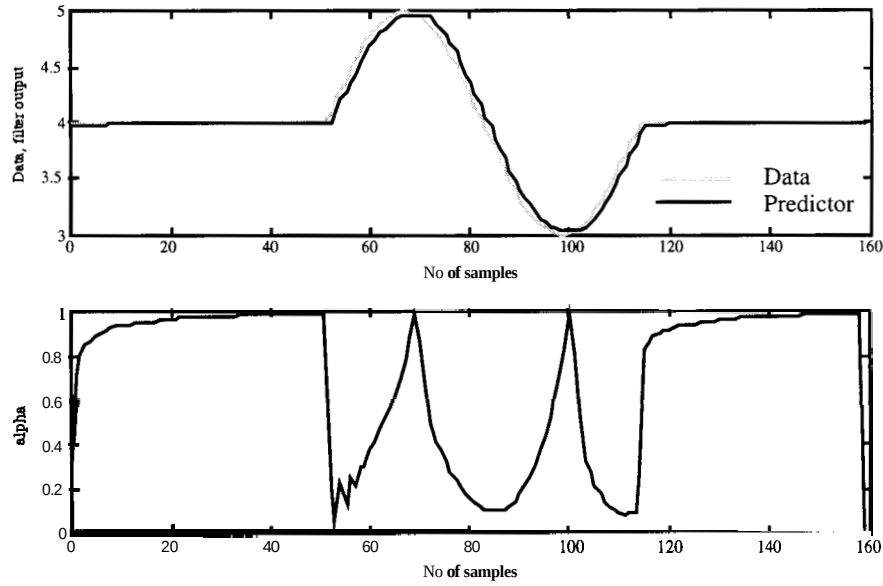


Fig. 2.1.1.2-8: Predictor with flexible α ; SSE=0.5088

Fig. 2.1.1.2-9 shows data for a non-linear system which was subject to both noise and outliers with a proper choice of the parameters. The SSE is 0.7361 which allows the system to respond quickly yet eliminate outliers and filter out noise.

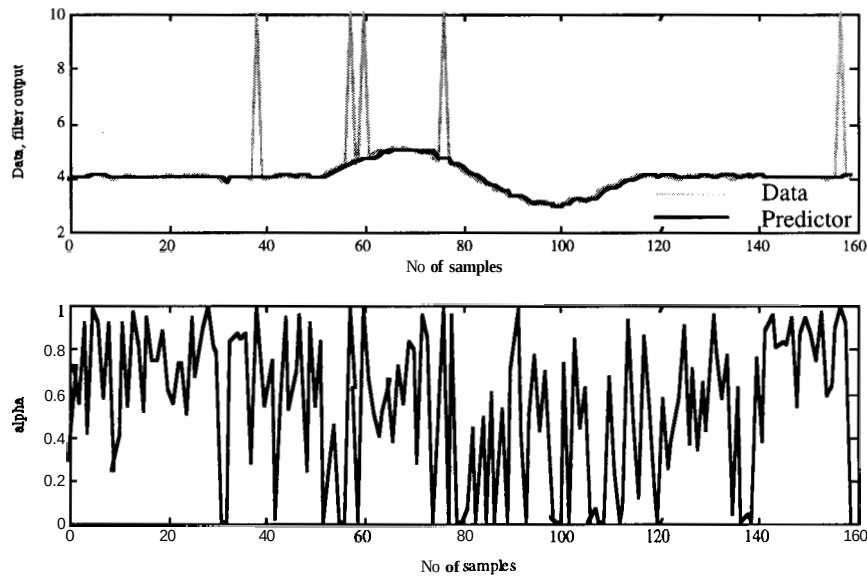


Fig. 2.1.1.2-9: Predictor with flexible α ; superimposed noise and outliers ($m_e=0.1$; $m_\alpha=0.99$; SSE = 0.7361)

As mentioned before, we used genetic algorithms for fine tuning of the parameters. Genetic algorithms are general purpose task independent adaptive search procedures (Holland, 1975; Grefenstette, 1990). They can be used as optimizers which provide in iterative procedures candidate solutions. Their particular strength is employment for complex objective functions where other approaches are not feasible. Such a situation is present here. The error function is (Goebel, 1996)

$$e(x, \alpha) = \sum_{i=1}^n (x_i - t_i)^2 + \frac{s}{\sum_{i=1}^n \alpha_i}$$

where

x is the fused value

t is the true datum

s is a fudge factor to scale the error of α

n is the number of steps during the experiment.

The terms of this error function are comprised of the **SSE** plus the inverse of the sum of the values for α over the entire range. The motivation is to penalize the aberration from the true data and to reward smooth filtering which can be obtained through a large α .

m_α and m_e and ω are found to be optimal at $m_\alpha=0.58$, $m_e=0.03$, and $\omega=930.60$. The response to the system with these parameters is shown in Fig. 2.1.1.2- 10. The parameters for the genetic algorithm are shown in Table 2.1.1.2- 1.

The results of the four independent experiments are summarized in Table 2.1.1.2- 1. The optimal value for m_e was found to be 0.03 while the optimal value for m_α lies in the range of 0.58 to 0.70. Since greater smoothness is desired, the lower value for m_α is chosen, giving the best results in terms of error and smoothness.

experiment	m_α	m_e	error
1	0.58	0.03	1.2026
2	0.67	0.03	1.2349
3	0.70	0.03	1.2325
4	0.70	0.03	1.2325

Table 2.1.1.2-1: GA experiments results

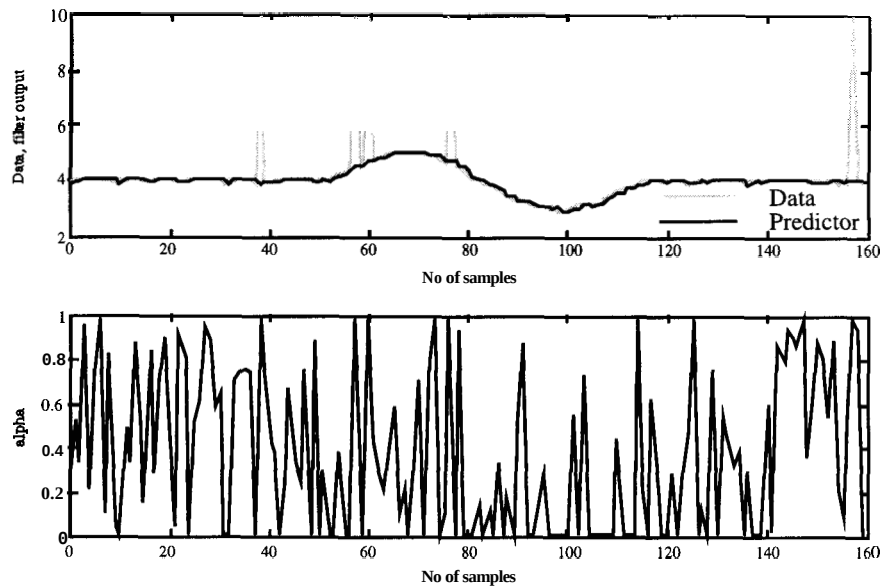


Fig. 2.1.1.2-10: Predictor response optimized through genetic algorithms, SSE=0.7056

The choice for the nonlinear validation functions $v(x)$ can be achieved through piecewise sigmoidal functions of the form

$$v(x) = \frac{1}{1 + e^{-a(x-c)}}.$$

Fig. 2.1.1.2-11 shows an example of a sigmoidal function where $a=1$, $c=0$. Parameter a determines the slope of the curve and parameter c places the curve along the x -axis.

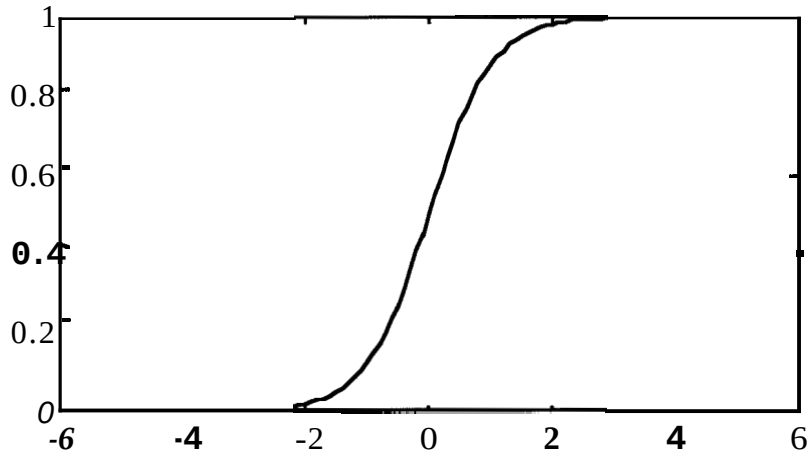


Fig. 2.1.1.2-11: Sigmoidal function ($a=1$, $c=0$)

Fig. 2.1.1.2-12 shows piecewise sigmoidal functions with different parameters a . They are centered around the value $c = 4$. The functions with declining slope are of the form

$$v(x) = 1 - \frac{1}{1 + e^{-a(x-c)}}$$

The sigmoidal functions have to fully cover the range from 0 to 1 between the validation gate boundary and the centerpoint (expected next value). Therefore, the functions are scaled over that range.

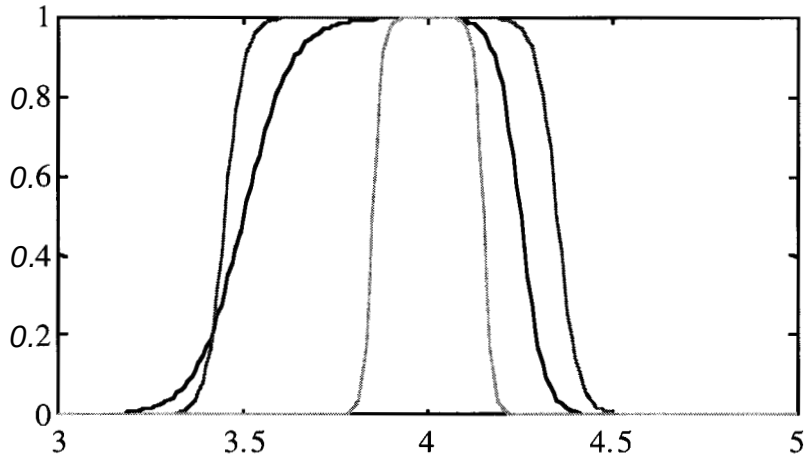


Fig. 2.1.1.2-12: Piecewise sigmoidal functions using different parameters a_i

Fig. 2.1.1.2-13 shows the fused values without the use of a predictor in the presence of measurements from steady state value **4** subject to Gaussian noise. The **SSE** is 0.053. The lower half of the graph shows the confidence values which vary with each reading and sensor.

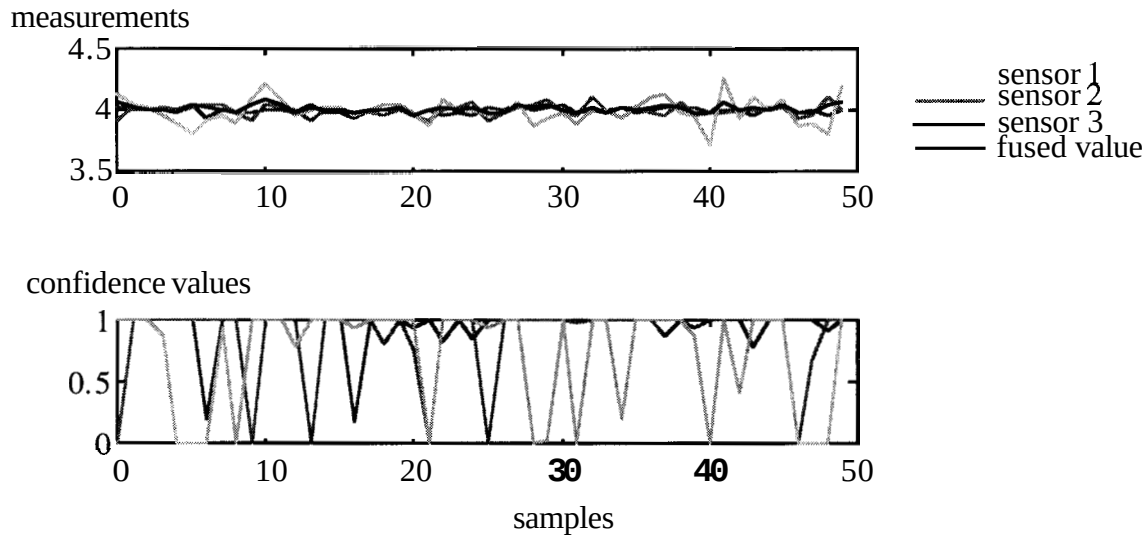


Fig. 2.1.1.2-13: Fusion using confidence values (without predictor)

In Fig. 2.1.1.2-14, the predictor is used to find the next expected reading and center the validation gate around it. The SSE drops to 0.0306. The lower half of the graph shows the single value for a .

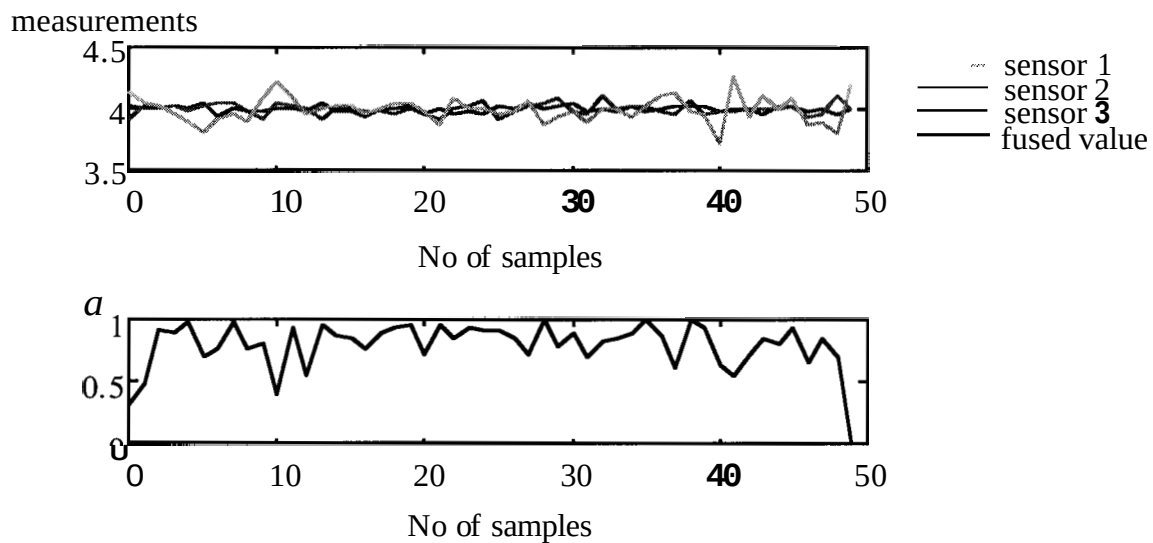


Fig. 2.1.1.2-14: Fusion using confidence values (with predictor)

In Fig. 2.1.1.2-15, outliers are superimposed to the readings. The fuzzy fusion algorithm filters out the outliers using validation gates. The SSE is **0.3490**.

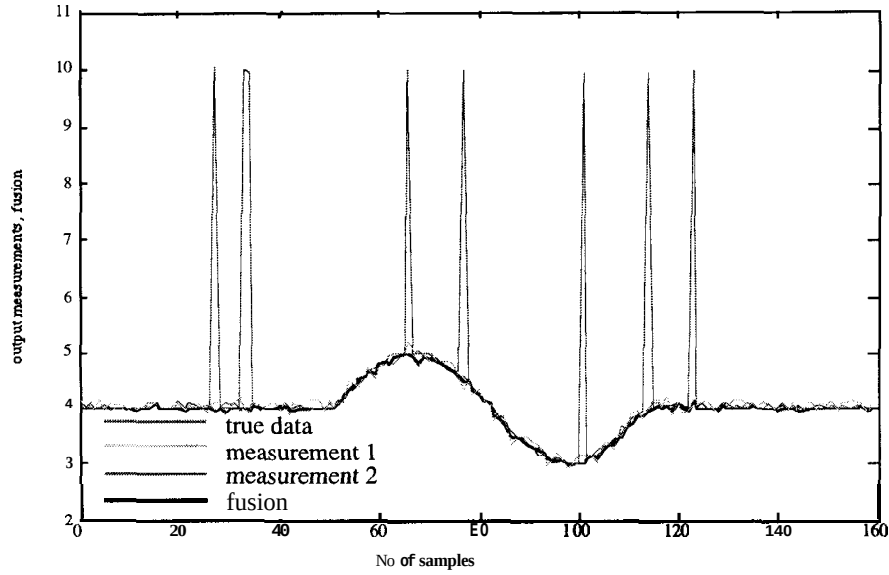


Fig. 2.1.1.2-15: Fuzzy fusion for nonlinear data with outliers

If a bias is introduced to the measurements of one sensor starting at sample no. **80**, the fuzzy filter easily disregards the newly introduced offset. Fig. 2.1.1.2-16 shows this scenario. The SSE is 0.8643.

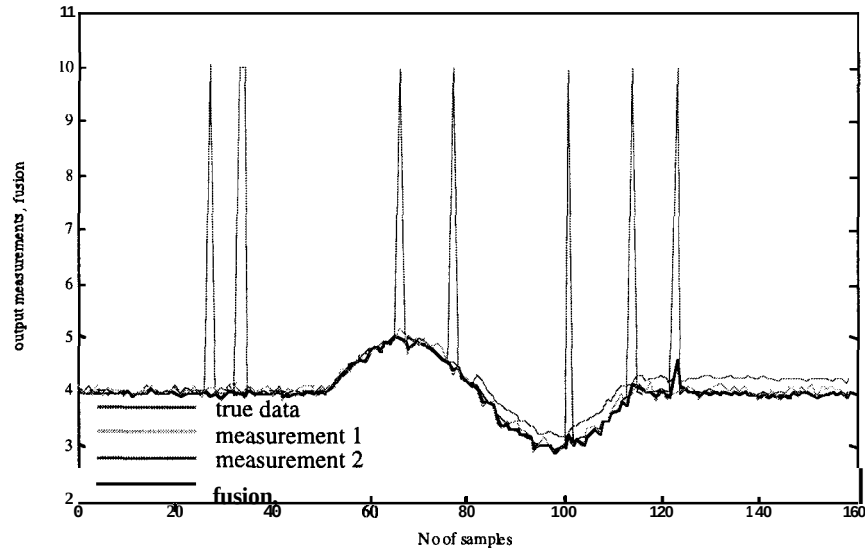


Fig. 2.1.1.2-16: Fuzzy fusion for nonlinear data with sudden bias

2.1.1.3 Sensitivity Analysis

The parameters m_e and m_α which determine the shape of the membership functions and ultimately the behavior of the time series predictor were optimized using GAs in a previous step. A sensitivity analysis following shows how the error changes with the parameters and motivate initial parameters. Fig. 2.1.1.3-1 shows the sensitivity analysis of m_e vs. m_α for Gaussian noise only. Parameter m_α is more receptive to change than m_e . In particular, m_α is found to be best at low values. If it is too large, the defuzzification method chosen produces values for $\mathbf{a}$ which are too large to properly respond to system changes. The best values for m_e are found at higher values

although m_e is not very receptive to changes. Fig. 2.1.1.3-2 shows the sensitivity analysis of m_e vs. m_α in the presence of outliers and clutter and Fig. 2.1.1.3-3 shows the sensitivity analysis for m_e vs. m_α in the presence of Gaussian noise, clutter, random and steady bias, and outliers. The shape of the error profile is essentially the same except that the acceptable m_e and m_α region shrinks. This is due to the fact that the system must be more responsive, i.e., it needs a smaller α .

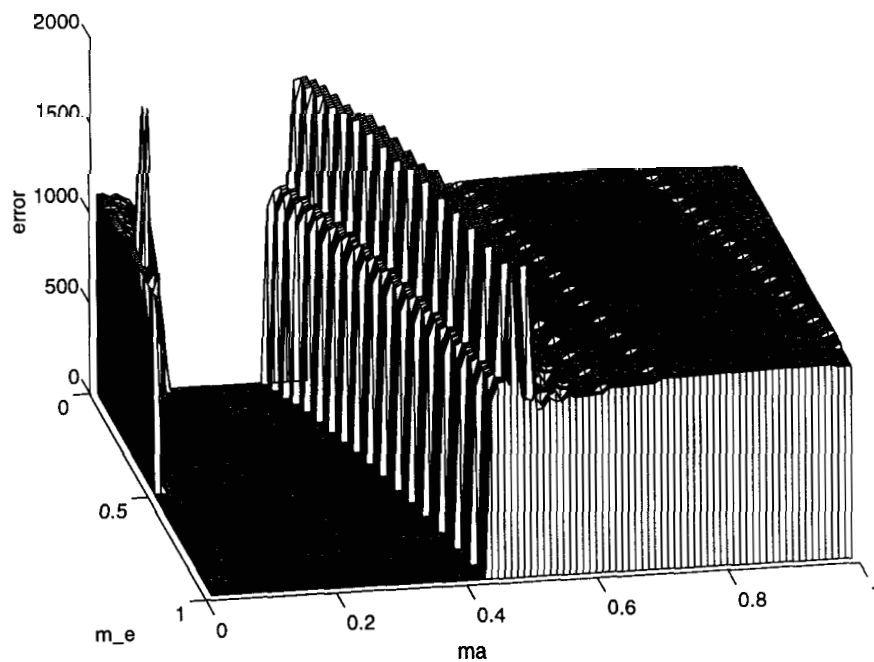


Fig. 2.1.1.3-1: Sensitivity analysis for m_e/m_α ; Gaussian noise only

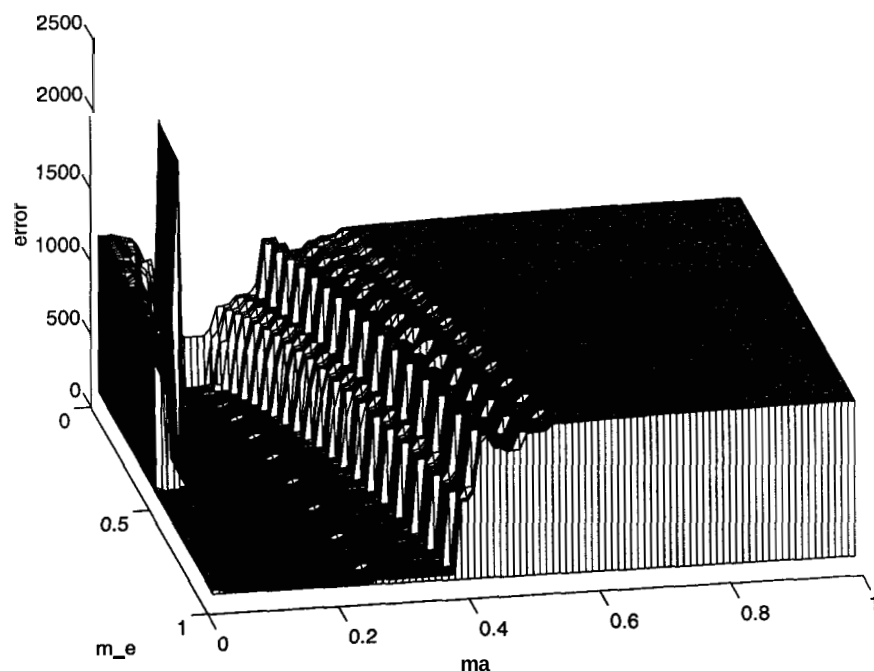


Fig. 2.1.1.3-2; Sensitivity analysis for m_e/m_α ; system subject to outliers and clutter

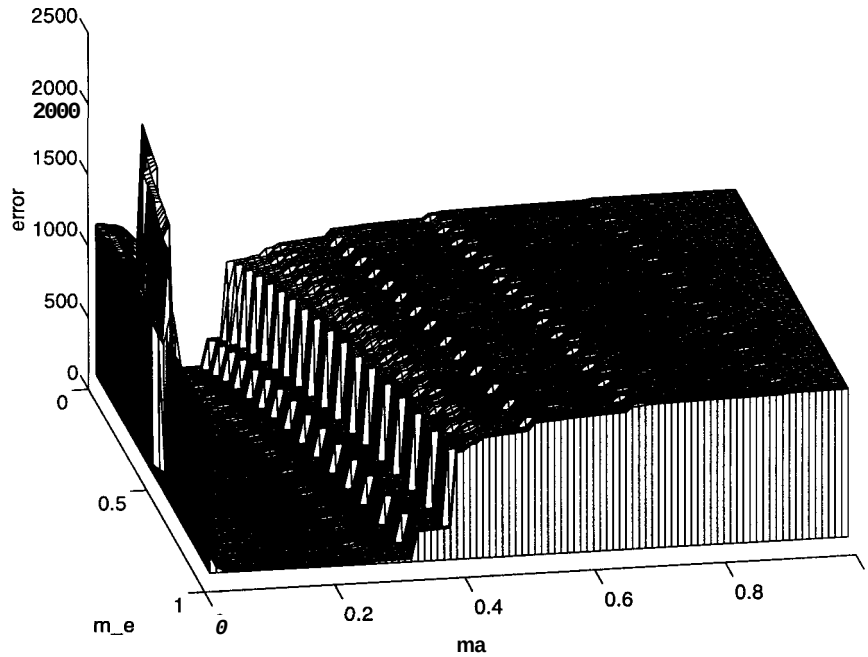


Fig. 2.1.1.3-3 Sensitivity analysis for m_e/m_α ; system subject to outliers, bias (random and steady), clutter, and Gaussian noise

The response of the system in the range of high error ($m_\alpha=0.5$, $m_e=0.5$) is seen in Fig. 2.1.1.3-4 for an example when the system is subject to all types of noise (Gaussian, random bias, steady bias, clutter, and outliers). The filter response is unstable in the sense that it does not allow large aberrations from the steady state. Here, the algorithm favors a constellation where the defuzzification is performed such that $\mathbf{a}$ is not too large. In this case, $\mathbf{a}$ is too large compared to the system dynamics, therefore overemphasizing past values without acknowledging the transient state. That in turn results in an algorithm which has periods of steady output, i.e., it does not consider incoming readings.

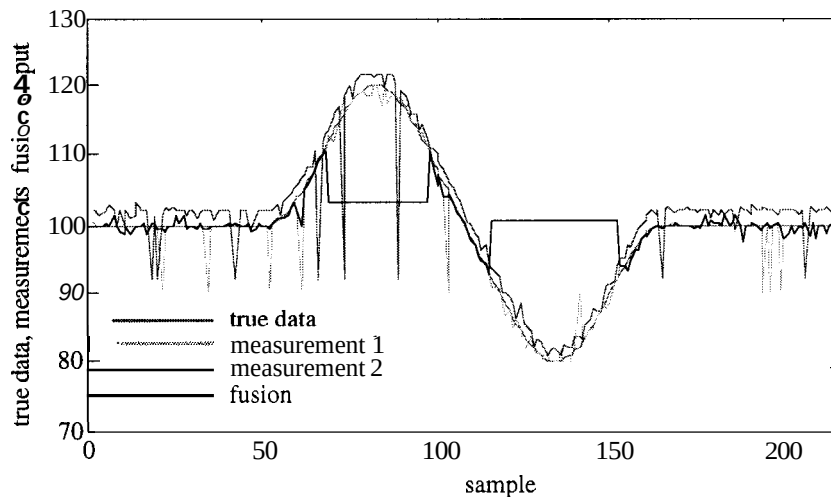


Fig. 2.1.1.3-4: Example of unstable choices for m_α and m_e

The sensitivity of the scaling factor ω , which determines to which degree the old value is integrated

into the new estimate (recall: $\hat{x}_f = \frac{\sum_{i=1}^n z_i \sigma(z_i) + \frac{\alpha \hat{x}}{\omega}}{\sum_{i=1}^n \sigma(z_i) + \frac{\alpha}{\omega}}$), is displayed in Fig. 2.1.1.3-5.

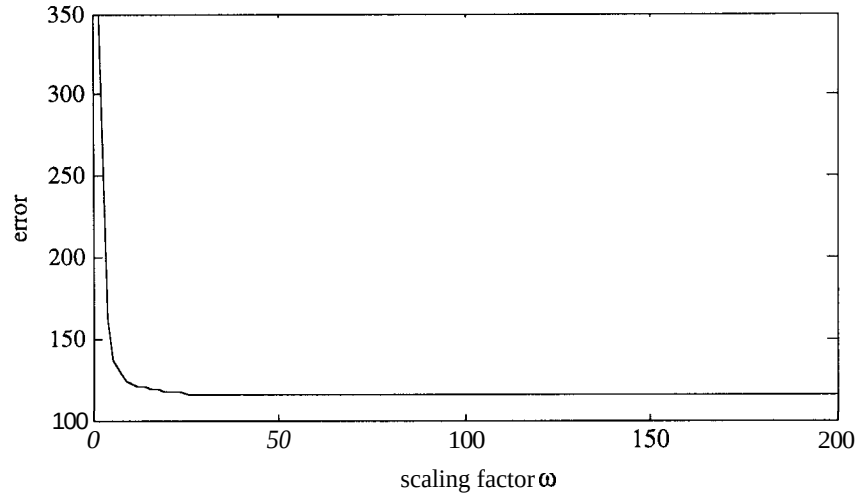


Fig. 2.1.1.3-5: Sensitivity analysis of the scaling factor ω

The scaling factor produces large errors for small values, drops to a minimum, and increases very slowly. Not visible on this curve is the flattening of the increase for large values of ω . For initial values, a large scaling factor is recommended. If too small values are used, new incoming information is discarded, resulting in a behavior as displayed in Fig. 2.1.1.3-6. Here a value of $\alpha=1$ was chosen. It can be seen how a lag is introduced in the fused value because the old information is integrated too much into the new value. Further reduction of ω results in periods of unstable algorithm behavior. In that case, there are periods of constant output because no new value comes close to the old ones and the validation gate discards new information.

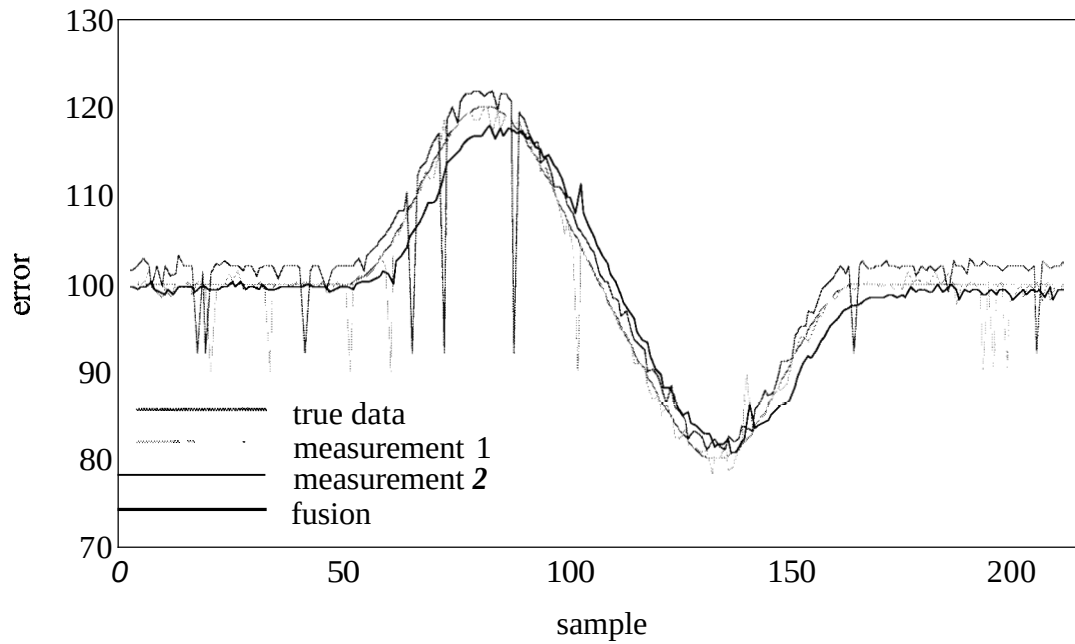


Fig. 2.1.1.3-6: Faulty behavior with small values for scaling factor $\omega=1$

Fig. 2.1.1.3-7 shows the sensitivity analysis for a_1 and a_2 . The algorithm is unstable only for very small values of a_1 and a_2 . This means that any “reasonably” large value is a good starting value from which an optimal value can be found. Small values for a_1 and a_2 mean a small basis of the validation curve. If they are too small, the incoming readings will be assigned very small degrees of memberships which means that the old readings are over evaluated. In other words, disregarding new information results in an unstable algorithm.

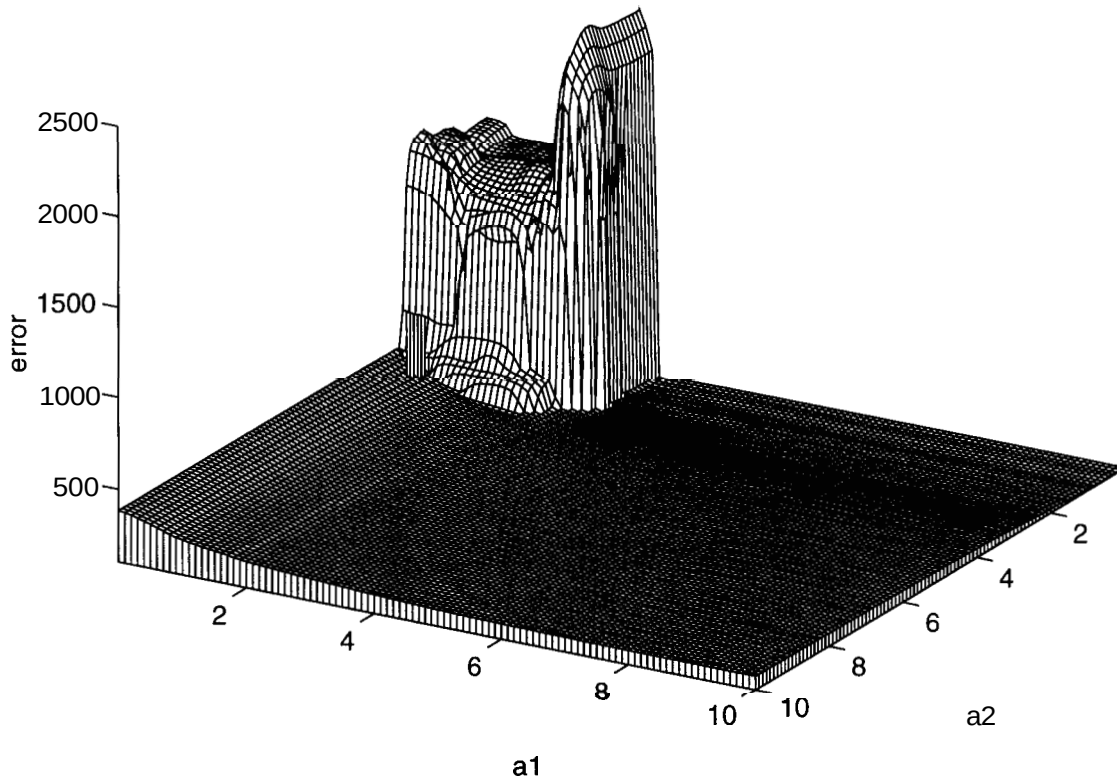


Fig. 2.1.1.3-7: Sensitivity analysis for a_1 and a_2

2.1.1.4 Recommendation and Summary

The FUSVAF algorithm is a tool for sensor validation and fusion using fuzzy techniques, exponential weighted moving average time series prediction, and heuristic knowledge about the sensor and system behavior. The algorithm assesses sensor confidence in validation regions which are modeled as dynamic non-linear curves according to sensor behavior. Several sensor readings are fused using a weighted average of sensor confidence, the sensor reading, and knowledge about the system state. This knowledge is expressed in the parameter $\mathbf{a}$ which captures the state of the system. Parameter $\mathbf{a}$ is also used in the time series prediction which in turn is used for the fusion and placement of the validation curves. The parameters of the fuzzy membership curves, validation curves, and fusion algorithm are learned through genetic algorithm learning techniques and it is shown in sensitivity analyses that they occupy a wide range of acceptable values. The algorithm is shown to filter out various kinds of Gaussian and non-Gaussian noise. Note that the fusion also accepts virtual measurements, e.g. values which are arrived at through functional redundancy. This feature makes the algorithm very stable against sensor failure as is shown in the application in Section 2.1.2. Further advantages of the validation and fusion methods are that time consuming numerical calculations are offloaded from the controller and that the measurement bus bandwidth requirements are reduced because no raw data need to be transferred to the controller.

Currently, the FEWMA algorithm assumes a fixed number of rules and shapes. Future research should explore the optimal number of rules and defuzzification techniques and automate this procedure for maximum performance. Also, it would be worthwhile to look into the ability of the

algorithm to include heuristic knowledge about sensor behavior, such as environmental conditions, in the design of the validation curves. This should be a straight forward extension of this work.

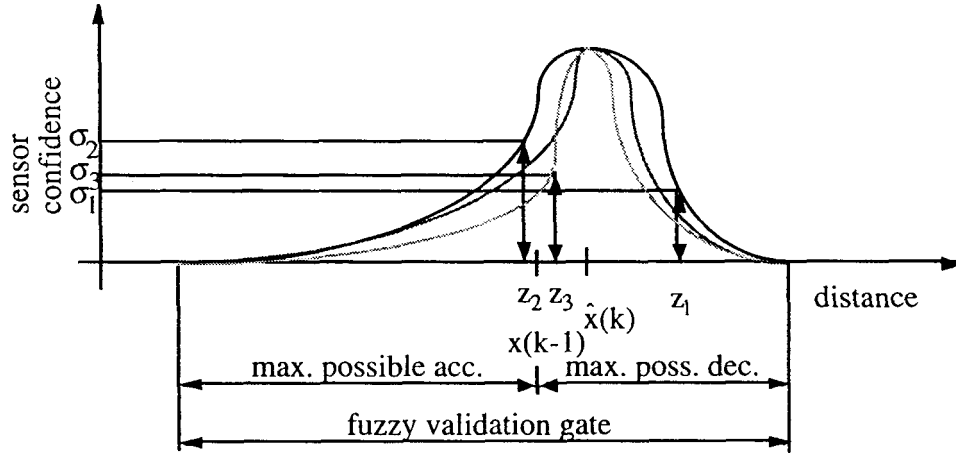
The FUSVAF algorithm has certain advantages over probabilistic algorithms in specific situations. However, it does not operate optimally in all cases. In the presence of Gaussian noise alone, Kalman filter based approaches perform slightly better since they are designed for optimal performance in that special situation. In the presence of non-Gaussian noise, the FUSVAF algorithm performs better by a large degree. If knowledge about sensor behavior exists, i.e., if regions or environmental conditions are known where the sensor acts in a Gaussian fashion, and the system is in a steady state, the Kalman filter should be chosen. If, on the other hand, the conditions change unpredictably or the noise is known to be of non-Gaussian nature, the FUSVAF algorithm should be used since it performs more robustly over a wide range of non-Gaussian as well as Gaussian noise (Goebel 1996). Each filter has to be tuned to perform in an optimal way. Ideally, sensor data exist to allow tuning for the parameters. If such data does not exist, a choice of the parameters should be used as outlined earlier in this chapter. Furthermore, the strength of the FUSVAF algorithm is its ability to incorporate knowledge about sensor characteristics and external conditions through the use of appropriate validation curves. These curves can be updated dynamically. The FUSVAF algorithm is a computationally less expensive validation and fusion procedure than the Kalman Filter based approach, the Probabilistic Data Association Filter (PDAF).

2.1.2 Fuzzy Sensor Validation and Fusion for IVHS

This section shows how real-time sensor validation and fusion using means from fuzzy logic is applied to vehicle following tasks for automated highways. Multiple sensor readings are validated and fused using the FUSVAF algorithm consisting of a fuzzy time series prediction model, fuzzy validation gates, and a weighted average fusion scheme. The collaboration of these methods allows the assignment of degrees of confidence to each sensor reading. This is achieved through the use of validation gates which are areas in which the reading is expected to lie. They take into account the specific properties of each sensors as well as the physical limitations of the system considered. The placement of the validation gates in turn is dependent of the fuzzy time series prediction which uses fused past values and sensor readings as input. Experiments done for PATH (Partners for Automated Transportation Highways) California show that this method works successfully under a variety of operating conditions.

Probabilistic approaches such as the PDAF introduced in Section 2.2 commonly assume zero mean, Gaussian noise distributions. This assumption is often invalid as our experiments have shown (Agogino, Goebel and Alag 1995). Rather, sensors often exhibit non-Gaussian behavior which is due to the underlying physics of the sensor or environmental conditions which result in sub-optimal performance (Agogino et al. 1995, Bellm 1995). In addition, the variance of the system perturbation and the noise variables must be known in advance for the PDAF or, if estimated on-line, require additional computation time. The fuzzy approach does not depend on such presuppositions. As shown in Section 2.1.1 the FUSVAF algorithm is a robust method which performs well for a wide range of data subject to Gaussian and non-Gaussian noise.

The changes in the validation gate boundaries are determined using a form of Algorithmic Sensor Validation (ASV) (Kim et al. 1992). More specifically, the boundaries are set by the change from the old separation distance at the previous time step to what could be achieved by maximum acceleration of the follower car and the maximum deceleration of the lead car on the one side and maximum deceleration of the follower car and maximum acceleration of the lead car on the other side. The characteristics of the sensors set the shape of the curves in the validation gate and are dynamically adjusted throughout the operating range. The sonar sensor, for example, is known to give good readings when the sensor works within its bounds. Therefore, the final shape will have a wide base around the predicted value. The optical sensor, on the other hand, has bigger variance throughout its operation range. Therefore, the shape of the curve in the validation region narrows around the predicted value. The performance of the radar sensor lies somewhere between the two other sensors. Most notably, it experiences a non-Gaussian offset which has been attributed to a quantization effect (Agogino et al. 1995). In the area of this quantization phenomenon, the shape of the validation curve will narrow down on the right side as the region is approached because the sensor will give too large values in that region; it is wider on the left side. Fig. 2.1.2-1 shows the validation gate.



- radar sensor
- sonar sensor
- optical sensor
- z_i sensor measurements
- σ_i sensor confidence values
- $\hat{x}(k)$ predicted value
- $x(k-1)$ old value at time $k-1$

Fig. 2.1.2-1: Validation gate for the distance sensors

The parameters for the **FUSVAF** algorithm were learned using **GA** algorithms. Data using sensor models obtained from experiments for the longitudinal sensors provided the learning set for the **GA**. The experiments involved the sonar, radar, and optical sensors and were carried out in a static and dynamic manner. The optical sensor was unavailable for some experiments. For the static test, the distance was measured between the bumpers of the vehicles in increments of 1 meter over a distance from 1-65 meters. Measurements were taken in the center of the vehicle as well as with an offset to the side in increments of 10cm until no further correct readings could be recorded. This methodology established sensor footprints. The dynamic tests involved two vehicles. The front vehicle was kept at rest while the rear vehicle moved with constant velocity towards and away from the first. These patterns were performed driving straight at the center of the vehicles as well as driving at specified offsets and angles. Further experiments included testing when the vehicles were subject to vibration as well as other external disturbances such as rain, fog, and partial and complete masking of the sensors with dirt and plastic debris (Agogino et al. 1995, Bellm 1995).

The sensor model for the sonar sensor was thus established to be:

$$s(x) = \begin{cases} x + r_G \sigma_s & \text{for } x < 4 \\ x + r_G \sigma_s & \text{for } 4 \leq x < 8 \text{ and } r_u \leq \frac{8-x}{4} \\ 15 & \text{for } 4 \leq x < 8 \text{ and } r_u > \frac{8-x}{4} \\ 15 & \text{for } x \geq 8 \end{cases}$$

where

$s(x)$ is the modeled sonar measurement given the true distance x
 r_G is a random number with Gaussian distribution
 r_u is a random number with uniform distribution
 σ_s is the variance for the sonar sensor.

The sensor model for the radar sensor is:

$$r(x) = \begin{cases} x + r_G(\sigma_r + \sigma_d) + \rho & \text{for } x < 3.8 \\ 11x - 38 + r_G(\sigma_r + \sigma_d) + \rho & \text{for } 3.8 \leq x < 3.95 \\ 20.27 - 3.714x + r_G(\sigma_r + \sigma_d) + \rho & \text{for } 3.95 < x < 4.3 \\ x + r_G(\sigma_r + \sigma_d) + \rho & \text{for } 4.3 \leq x < 8.5 \\ 6.67x - 48.167 + r_G(\sigma_r + \sigma_d) + \rho & \text{for } 8.5 \leq x < 8.8 \\ 31.62 - 2.4x + r_G(\sigma_r + \sigma_d) + \rho & \text{for } 8.8 \leq x < 9.3 \\ x + r_G(\sigma_r + \sigma_d) + \rho & \text{for } x \geq 9.3 \end{cases}$$

where

$r(x)$ is the modeled sonar measurement given the true distance x
 σ_r is the variance for the radar sensor
 σ_d is the distance related variance

ρ is a non-Gaussian spike of small order described by $\rho = \begin{cases} 0.1 & \text{for } r_G > 2 \\ -0.1 & \text{for } r_G < -2 \\ 0 & \text{elsewhere} \end{cases}$

The results of the GA learning are summarized in Table 2.1.2-1. Recall the fusion function as

$$\hat{x}_f = \frac{\sum_{i=1}^n z_i \sigma(z_i) + \frac{a}{\omega}}{\sum_{i=1}^n \sigma(z_i) + \frac{a}{\omega}}$$

where

$\hat{x}_f$: fused value
 z_i : measurements
 σ : confidence values
 a : adaptive parameter of FEWMA time series predictor
 ω : constant scaling factor
 $\hat{x}$: expected value

and the validation curve as:

$$\sigma = \begin{cases} 0 & z < v_{\text{left}} \\ \frac{e^{-\left(\frac{\hat{x}-z}{a_{\text{left}}}\right)^2} - e^{-\left(\frac{\hat{x}-v_{\text{left}}}{a_{\text{left}}}\right)^2}}{1 - e^{-\left(\frac{\hat{x}-v_{\text{left}}}{a_{\text{left}}}\right)^2}} & v_{\text{left}} < z \leq \hat{x} \\ \frac{e^{-\left(\frac{\hat{x}-z}{a_{\text{right}}}\right)^2} - e^{-\left(\frac{\hat{x}-v_{\text{right}}}{a_{\text{right}}}\right)^2}}{1 - e^{-\left(\frac{\hat{x}-v_{\text{right}}}{a_{\text{right}}}\right)^2}} & \hat{x} < z \leq v_{\text{right}} \\ 0 & z > v_{\text{right}} \end{cases}$$

where

σ is the confidence value for a particular sensor

v_{right} and v_{left} are the right and left validation gate borders, respectively

a_{right} and a_{left} are the parameters for the left and right validation curve.

Parameter	
$a_{\text{left}_{\text{radar}}}$	55.527
$a_{\text{right}_{\text{radar}}}$	1.672
$a_{\text{left}_{\text{sonar}}}$	11.641
$a_{\text{right}_{\text{sonar}}}$	32.052
ω	185.6

Table 2.1.2-1: Parameters obtained from GA learning for IVHS application

Experiments were also carried out on a test lane on 1-15 near San Diego in collaboration with Caltrans. Here the platoons were driven under automatic control. Data were sampled and stored throughout these tests. Figs. 2.1.2-2 and 2.1.2-3 show the separation distance between the lead vehicle and the first follower car for straight following performed on 1-15. Two sensors for longitudinal distance sensing were used for this experiment. The sensors show a bias which depends on the calibration and tuning of the parameters and on environmental conditions. The sampling time was 20ms. The sonar sensor has one outlier towards the end of the sequence. The fused value follows the radar sensor more than the sonar and filters out the outlier. Fig. 2.3.2-3 shows a Kalman filter for the same experiment using the parameters chosen during simulation.

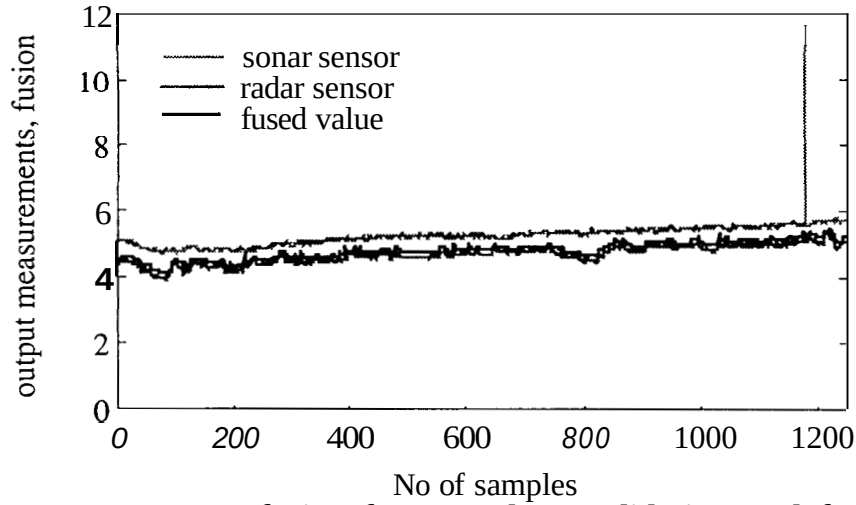


Fig. 2.1.2-2: Fuzzy fusion for open loop validation and fusion

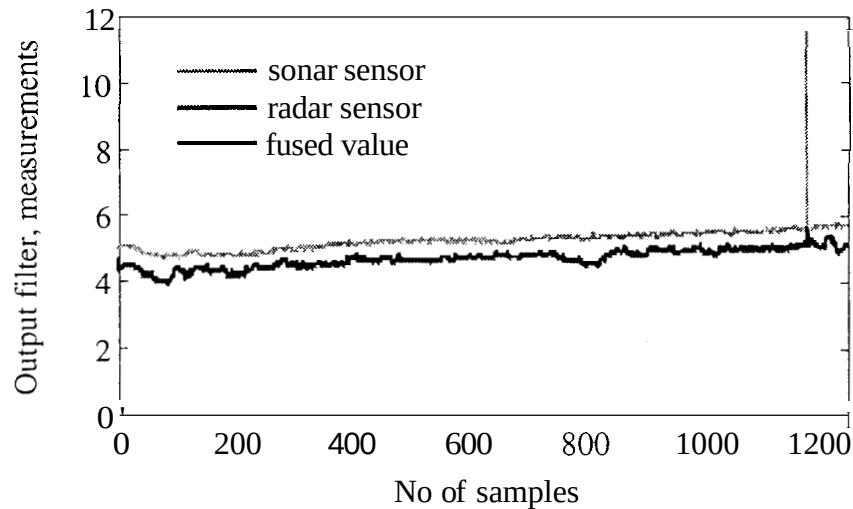


Fig. 2.1.2-3: Kalman filter for open loop validation and fusion

Fig. 2.1.2-4 shows the same sequence of data. Here, data obtained from the follower and lead vehicle wheel speed sensor and data from the follower closing rate sensor were used to establish functional redundancy through integration of the data to derive two more distance readings. These four readings were fused into one value representing the true distance between the cars. This redundant scheme leads to smoother control input values and very robust data return. Sensor failure and its inherent safety loss can be overcome. If for example all forward looking sensors such as the radar and sonar sensor are blocked, for example by some piece of cardboard flying up from the road in between vehicles (as happened during one experiment), a centralized power failure to the sensor grid, or an unusually high riding vehicle for which sensor contact is lost temporarily, the wheel speed sensors and the information about the speed of the first car transmitted via radio communication still allow proper control although the primary sensors are essentially blind. This constitutes a significant safety improvement of the control scheme. Fig. 2.1.2-5 shows this scenario for the same data but with the three active forward looking sensors (radar sensor, sonar sensor, and closing rate sensor) knocked out starting at sample no. 500. The virtual distance sensors obtained from the wheel speed sensor and the vehicle speed of the preceding vehicle transmitted to the follower vehicle via radio communication are able to provide a proper sensor reading for the controller.

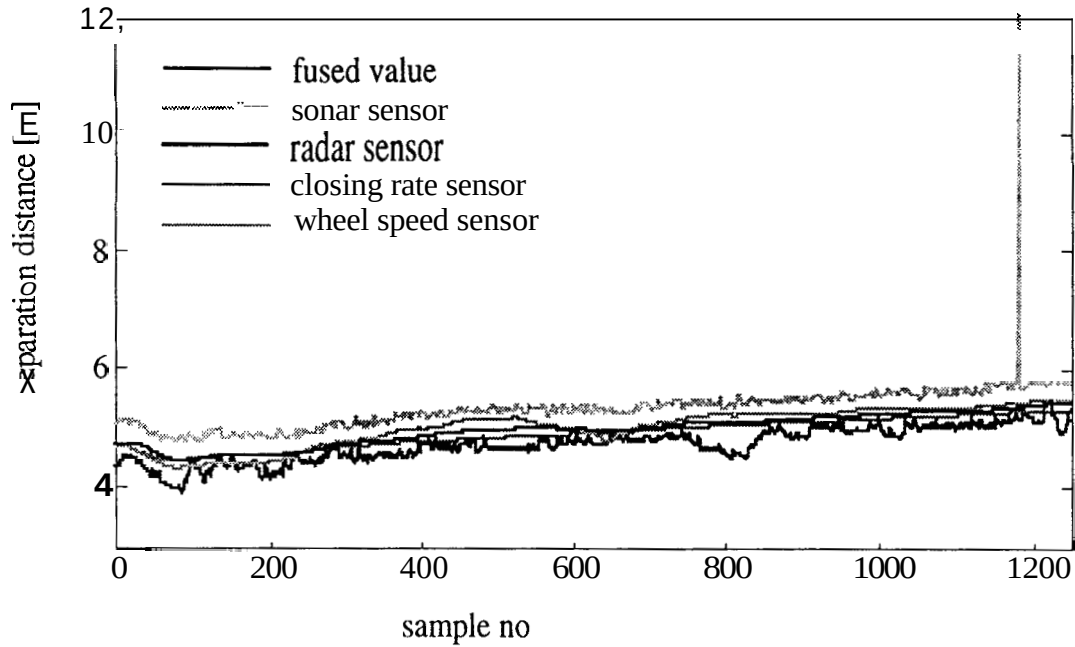


Fig. 2.1.2-4: Filter using information from radar, sonar, closing rate and wheel speed sensors

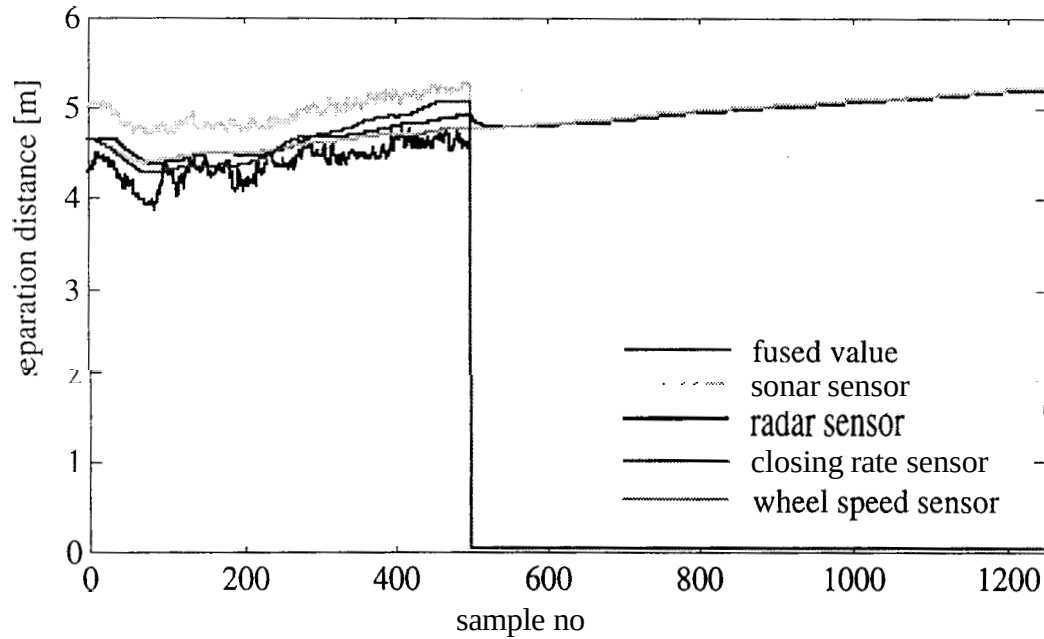


Fig. 2.1.2-5: Filter maintaining proper control input despite failure of all forward looking sensors (radar distance, radar closing rate, and sonar)

A simple split and merge experiment was carried out in Figs. 2.1.2-6 and 2.1.2-7. The radar sensor had little variance throughout the experiment but experienced “bumps” around 4.5m and 9m which have been attributed to a quantization error. The sonar sensor showed the smallest variance throughout its operating region but exhibited outliers which showed up above 4m and increased with distance between the follower vehicle and the lead vehicle. Above 8m no good readings were found. The optical sensor had the highest variance of all sensors which increased with growing distance between follower and lead vehicle but otherwise show no adverse effects. The fused value

filters out the spikes of the sonar sensor, the bumps of the radar sensor, and the noise of all sensors. The Kalman filter, on the other hand, performs well except where the sensors behave in a non-Gaussian manner, although a validation gate was also used for the Kalman filter to filter out outliers.

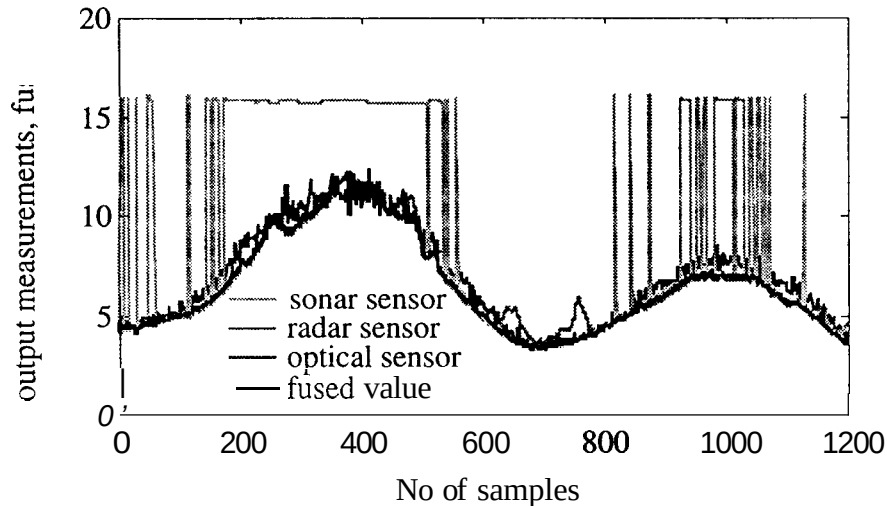


Fig. 2.1.2-6: Fuzzy fusion for split/join maneuver; open loop validation and fusion of three longitudinal sensors

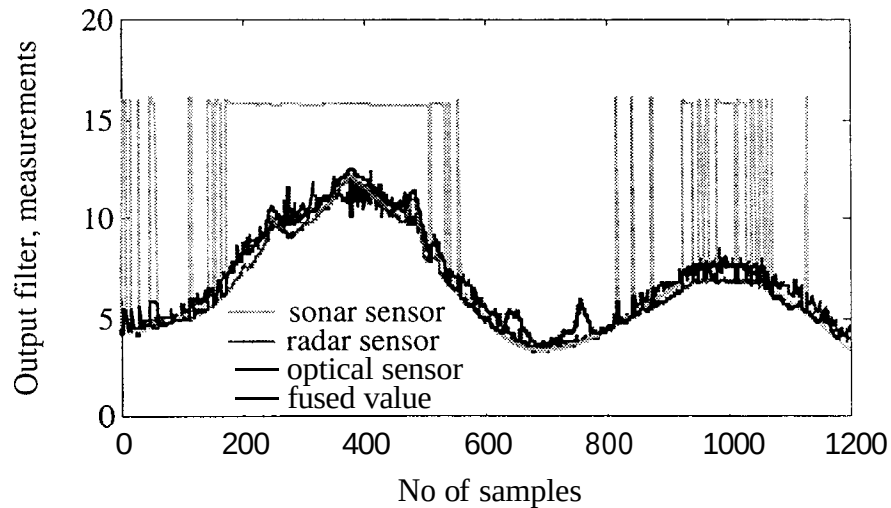


Fig. 2.1.2-7: Kalman filter for split/join maneuver; open loop validation and fusion of three longitudinal sensors

While the fused values shown in Fig. 2.1.2-6 were used in open loop fashion, Fig. 2.1.2-9 shows the effect of feedback of the fused value to the machine level controller which affects the throttle angle. For this, simulation software release 1.0 from the Vehicle Dynamics lab of UC Berkeley was used. Fig. 2.1.2-8 shows the velocity and acceleration profile for the simulated maneuver. For comparison, Kalman and Probabilistic Data Association Filters (PDAF) were used for fusion (Alag, Goebel, and Agogino 1995) with preset gains. Over a time period of 30 seconds, the spacing error was summed up. The sum squared error (**SSE**) for perfect information, i.e. no noise, was 0.6693. When non-Gaussian noise was introduced, the **SSE** was 186.5875. A Kalman filter reduced the **SSE** to 1.9186, the PDAF to 1.3901, the fuzzy filter alone to 0.8638, and the fuzzy fusion to 0.8454.

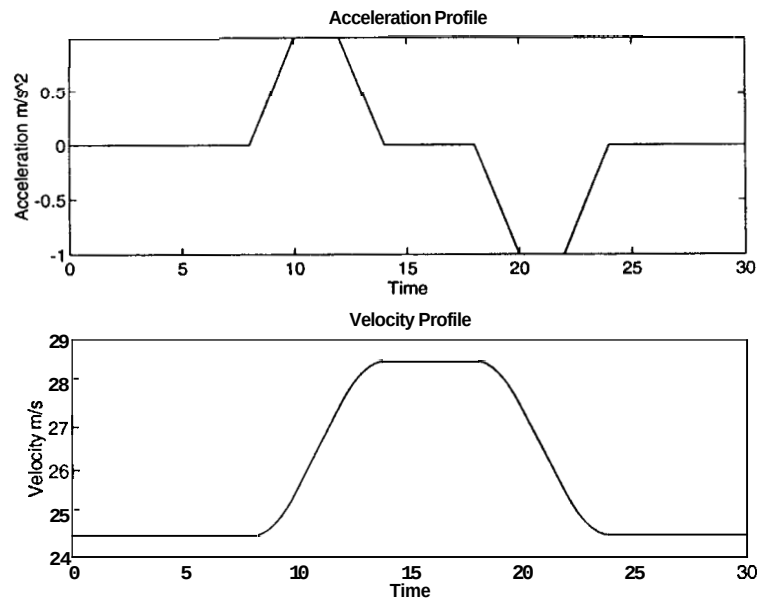


Fig. 2.1.2-8: Simulated maneuver velocity and acceleration profiles

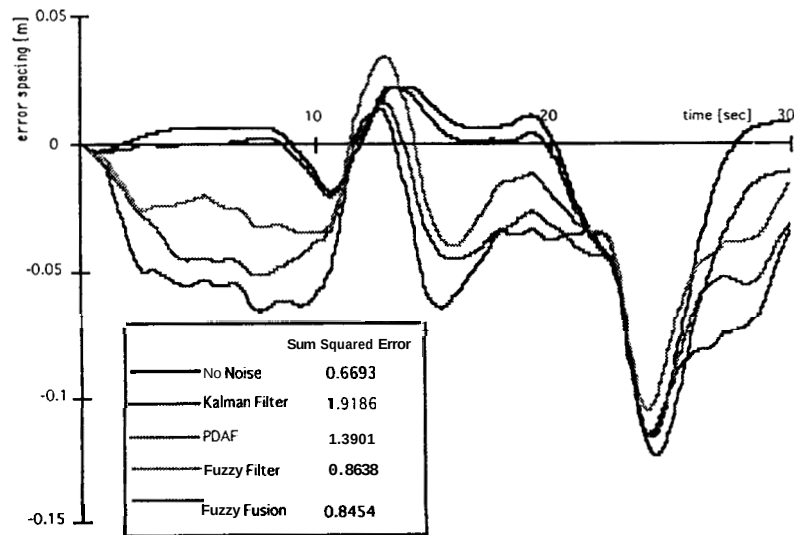


Fig. 2.1.2-9: Error spacing of follower car for closed loop sensor validation and fusion in the presence of non-Gaussian noise

2.2 Sensor Validation and Fusion Using Bayesian Techniques

This section shows how sensor validation and fusion are performed using vector dynamic belief networks. After the theory is introduced in Section 2.2.1, applications in IVHS are shown in Section 2.2.2.

2.2.1 Methodology for Probabilistic Sensor Validation and Fusion

This section develops a single unified framework, which is the synthesis of techniques used in artificial intelligence and in ‘modern control theory’, with probability as the common link between them. By combining these two approaches into a broad and integrated framework, we provide a new perspective, that not only draws from but also builds on previous work. This framework provides a rich foundation for developing Bayesian estimation algorithms. Readers are referred to Alag (1996) for the representation and development of other estimation algorithms and their application to extant practical systems.

Estimation is the process of obtaining an approximation to the time history of a process’s behavior from noisy measurements. The representation scheme introduced is based on Gaussian networks¹, – directed acyclic graphs that encode multivariate Gaussian probabilistic relationships between variables. The network representation consists of nodes, representing random variables, and arcs – representing inter-relationships between the variables. We also develop rigorous calculi for carrying out inference (the process of estimating the states of the random variables represented in the network when the values for some of the variables are known) within these networks. These rules for inference have been developed using two different methods that lead to the development of estimation algorithms that can be implemented in either a centralized (single processor) or a decentralized (multiple processor or information filter) manner. The framework provides a rich foundation for developing Bayesian estimation algorithms. By using this framework one can convert the tasks associated with monitoring and estimation: sensor validation, sensor data-fusion, and fault detection to a problem in decision-theory. Bayesian estimation algorithms such as the Kalman filter, probabilistic data association filter (Bar-Shalom and Tse 1975; Bar-Shalom and Fortmann 1988), multiple model algorithms (Bar-Shalom and Li 1993), as well as new algorithms can be easily represented and derived using this network representation scheme and the rules for inference (Alag 1996).

Kalman (1960) solved the problem of sequential least squares estimation of state vectors for discrete-time linear system with noise. It is well-known that the Kalman filter can be derived by the application of Bayes’ rule to the measurement model of observations from a dynamic system (Bryson and Ho 1975). The use of networks to represent probabilistic information began early in the twentieth century with the work of Wright (1921). The decision network representation for decision problems, which incorporated a directed acyclic graph representation for random variables was used in decision analysis in the late 1970s, but no topological evaluation algorithms were developed for decision networks until the development of the topology transformation method by Olmstead (1984). Kenley (1986) showed how the discrete-time filtering process can be represented by means of Gaussian networks. He extended Olmstead’s method of topology transformation to Gaussian networks and showed how the Kalman filter can be derived in its centralized form. Pearl (1988) developed an alternate inference scheme for Gaussian networks consisting of scalar nodes (a node represents a single random variable) based on the method of message propagation. In this section, we formalize the graphical representation framework by making use of vector nodes (a node represents multiple random variable) and extend Pearl’s message propagation scheme for Gaussian networks with vector nodes. This enables us to represent and derive Bayesian estimation algorithms in their centralized and decentralized form.

¹ Gaussian networks are a subset of probabilistic or Bayesian belief networks.

A fully decentralized system is an interconnected network of intelligent sensors each with its own transducers and processing capabilities. Locally obtained results are communicated to the other sensors for further processing, allowing each sensor to achieve a global result. Such a system has many benefits in terms of modularity, speed and survivability (Rao and Durrant-Whyte 1993). As we will see one can derive the inter-nodal messages in a decentralized architecture using the message propagation scheme. The decentralized algorithm is ideal for parallel processing implementation. A centralized system consists of a central processor where global data fusion takes place or a central communication medium through which all messages must pass. For details on centralized and decentralized Kalman filtering and sensor management, readers are referred to Hashemipour (1988) and Manyika and Durrant-Whyte (1994).

Estimation Problem

Consider a discrete time linear stochastic system described by the following vector difference equation

$$x(k+1) = F(k)x(k) + G(k)u(k) + v(k) \quad k = 0, 1, \dots$$

where vector x of dimension n_z is the state of the system. The state transition matrix F and the discrete time gain G through which the input, assumed to be constant over a sampling period, enters the system. We have introduced time through the index k . In this equation, $u(k)$ is an n_z -dimensional known input vector, and $v(k)$, $k=0, 1, \dots$, is the sequence of zero-mean white Gaussian process noise (also n_z dimension vector) with covariance

$$E[v(k)v(k)^T] = Q(k).$$

The system is monitored by a group of sensor with the following sensor model

$$z(k) = H(k)x(k) + w(k) \quad k = 1, \dots$$

where H is the measurement matrix and $w(k)$ the sequence of zero-mean white Gaussian measurement noise with covariance

$$E[w(k)w(k)^T] = R(k).$$

The observations could be some simple subset of the parameters, or the complete parameter vector. It is assumed that the state evolution model (matrices $F(k)$, $G(k)$, and $Q(k)$) as well as the sensor model (matrices $H(k)$, $R(k)$) are known.

The initial state of the random variable $x(0)$ is generally unknown and the probability density function is modeled as a normal distribution with a known mean and variance. The two noise sequences and the initial state are assumed mutually independent.

2.2.1.1 Graphical Representation

We begin by describing the meaning of nodes and arcs. This is followed by the representation of time. A Gaussian network is a sub-class of probabilistic or belief networks. A belief network is a directed acyclic graph representing random variables. Each node in the network represents a variable which could be either a random variable, a constant, or a deterministic variable. The links represent causal influences among the random variables.

2.2.1.1.1 Nodes and Arcs

Consider a system x , of n state (continuous) variables $x_1, \dots, x_n$. The joint probability density function for x is a multivariate nonsingular normal distribution

$$\Pr(\bar{x}) = N(x, P, \bar{x}) = \frac{1}{(2\pi)^{n/2} |P|^{1/2}} \exp\left(-\frac{1}{2}(x - \bar{x})^T P^{-1} (x - \bar{x})\right)$$

where $N(\cdot)$ denotes the normal pdf with argument $\mathbf{x}$, and $\bar{\mathbf{x}} = E[\mathbf{x}]$ and $\mathbf{P} = E[(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T]$ are respectively, the mean and covariance matrix².

This joint distribution of the random variables can also be written as a product of conditional distributions each being an independent normal distribution, namely

$$\Pr(\mathbf{x}) = \prod_{i=1}^n \Pr(\mathbf{x}_i | \mathbf{x}_1, \dots, \mathbf{x}_{i-1})$$

$$\Pr(\mathbf{x}_i | \mathbf{x}_1, \dots, \mathbf{x}_{i-1}) = N\left(\mathbf{x}_i, \mathbf{m}_i + \sum_{j=1}^{i-1} \mathbf{b}_{ij} (\mathbf{x}_j - \mathbf{m}_j) / \mathbf{v}_i\right)$$

where $\mathbf{m}_i$ is the unconditional mean of $\mathbf{x}_i$, $\mathbf{v}_i$ is the conditional variance of $\mathbf{x}_i$ given values for $\mathbf{x}_1, \dots, \mathbf{x}_{i-1}$, and $\mathbf{b}_{ij}$ is a linear coefficient reflecting the strength of the relationship between $\mathbf{x}_i$ and $\mathbf{x}_j$. One can interpret a multivariate normal distribution as a belief network, where there is an arc from $\mathbf{x}_j$ to $\mathbf{x}_i$ whenever, $\mathbf{b}_{ij} \neq 0, j < i$. Due to the assumption that the variables are normally distributed the complete distribution can be specified with the help of just two parameters - the mean and the variance.

In order to develop a graphical representation for the estimation process we extend these continuous Gaussian networks to their vector form. Fig. 2.2.1.1.1-1 shows a generic form of a vector Gaussian network. Here, the variables ($\mathbf{U}_i, \mathbf{X}, \mathbf{Y}_i$) represented by the nodes are vectors, e.g., $\mathbf{X} \equiv [\mathbf{x}_1, \dots, \mathbf{x}_n]^T$, where $\mathbf{x}_i$ are Gaussian random variables.

The arc between the variables represents the following relationship

$$\mathbf{X} = \sum \mathbf{F}_i \cdot \mathbf{U}_i + \mathbf{v}; \quad \mathbf{Y}_i = \mathbf{H}_i \mathbf{X} + \mathbf{w}_i$$

where $\mathbf{U}_i$ and $\mathbf{Y}_i$ are n_{U_i} and n_{Y_i} dimensional vectors, $\mathbf{F}_i$ is a $n_n \times n_{U_i}$ matrix, $\mathbf{H}_i$ is a $n_{Y_i} \times n_n$ matrix, $\mathbf{v}$ is a n_n vector, $\mathbf{Q} = E[\mathbf{v}\mathbf{v}^T]$ is the covariance matrix for the noise term representing the correlation between the parent variables, and $\mathbf{w}_i$ is a n_{Y_i} vector.

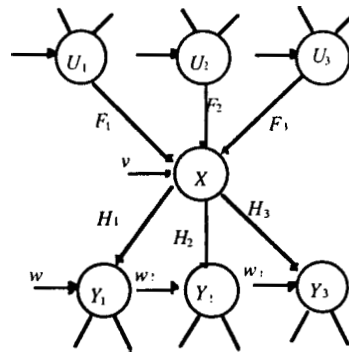


Fig. 2.2.1.1.1-1: Generic form of a vector Gaussian belief network

² Throughout the section we will use the symbol **Pr** to represent probability, and the symbol $N(\cdot)$ to represent the normal distribution. The inverse of the covariance matrix, $\mathbf{P}^{-1}$ is also known as the precision or the information matrix.

2.2.1.1.2 Modeling of Time: Gauss-Markov Process

In a monitoring system as time progresses new information is obtained by the sensors, which needs to be incorporated into the estimation process. For this purpose we will build a network where nodes at a particular time are duplicated at each time step. Nodes over time are connected by means of a state evolution model, i.e., we model the process as a Markov process.

Fig. 2.2.1.1.2-1 illustrates the process for a generic network. Here, the evolution of the states is modeled by the conditional probability distribution $\Pr(X_t|X_{t-1})$, i.e., the sequence of X_t values are determined solely by the previous one. The network consists of a sequence of time slices where nodes within time slice t are connected to nodes in time slice $t+1$ as well as to other nodes within slice t (Dean and Kanazawa 1989; Kjaerulff 1992). In a dynamic network, if t is the current time step, then we have evidence from the sensor nodes up to and including time t . The network must calculate the probability distribution for the state at time t and determine how the state will evolve into the future: the probability distribution for $State_{t+1}$ etc. This task which is called probabilistic projection is carried out in two operations, the prediction and the estimation phases. Each process cycle consists of the following steps (Russell and Norvig 1995):

- i. Prediction: Let the two slices of the network be $t-1$ and t . Let, $Bel(X_{t-1})$ be the belief in the state incorporating all the evidence up to and including the evidence at time $t-1$. By using the state evolution model, we use probabilistic projection to calculate $Bel(X_t|E_{t-1})$.
- ii. Roll up: The slice $t-1$ is removed. This requires adding a prior probability table for the state variable at time t . The prior is just $Bel(X_t|E_{t-1})$.
- iii. Estimation: we use the new evidence E_t to update our current belief, i.e., calculate $Bel(X_t|E_t)$. The time slice for $t+1$ is added and the network is ready for the next cycle.

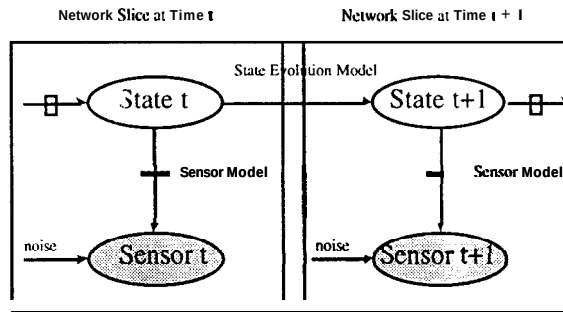


Fig. 2.2.1.1.2-1: Generic structure of a dynamic probabilistic network. In an actual network there may be many state and sensor variables in each time slice

2.2.1.2 Inference

The basic task of the inference process is to compute the posteriori probability distribution - the belief, i.e., the expectation - for the variables given exact values for some evidence variables³. We extend Pearl's message propagation scheme (Pearl 1988) to vector Gaussian networks. This leads to the development of an inference scheme by which decentralized form of estimation algorithms

³ Within the estimation framework the sensor nodes are the evidence nodes.

can be derived. We review Olmstead's topology transformation scheme for inference and adapt it to our vector Gaussian networks.

2.2.1.2.1 Inference using Message Propagation: Decentralized Approach

Consider a typical fragment of a singly connected network, consisting of an arbitrary node X , the set of all X 's parents, $\mathbf{U} = \{U_1, U_2, \dots, U_n\}$, and the set of all X 's children, $\mathbf{Y} = \{Y_1, Y_2, \dots, Y_m\}$. Let $\mathbf{e}$ be the total evidence obtained, $\mathbf{e}_x^-$ be the evidence connected to X through its children ($\mathbf{Y}$), and $\mathbf{e}_x^+$ be the evidence connected to X through its parents ($\mathbf{U}$). Here, the links correspond to

$$x = \sum_i F_i \cdot u_i + v$$

$$y_i = H_i x + w_i \quad ,$$

Belief Update

To compute the belief of X , i.e., $BEL(x)$, we divide the evidence $\mathbf{e}$ into two components, $\mathbf{e}_x^+$ and $\mathbf{e}_x^-$, representing data in the sub networks above X and below X , respectively.

We calculate the belief for node X , as follows

$$BEL(x) = f(x|\mathbf{e}_x^+, \mathbf{e}_x^-) = \alpha \cdot f(x|\mathbf{e}_x^+) f(\mathbf{e}_x^-|x) = \alpha \cdot \pi(x) \cdot \lambda(x)$$

$$\pi(x) = f(x|\mathbf{e}_x^+) = \int \cdots \int_{U_1 \dots U_n} f(x|\mathbf{e}_x^+, u_1, \dots, u_n) f(u_1, \dots, u_n|\mathbf{e}_x^+) du_1 \cdots du_n$$

$$= \int \cdots \int_{U_1 \dots U_n} f(x|u_1, \dots, u_n) \prod_{i=1}^n f(u_i|e_i^+) du_1 \cdots du_n$$

$$= \int \cdots \int_{U_1 \dots U_n} N\left(x; Q, \sum_{i=1}^n B_i u_i\right) \prod_{i=1}^n N(u_i; P_i^+, \bar{u}_i^+) du_1 \cdots du_n$$

Using Rule 9, we get

$$\pi(x) = N\left(x; \sum_{i=1}^n B_i P_{u_i} B_i^T + Q, \sum_{i=1}^n B_i \bar{u}_i^+\right)$$

$$= N[x; P_\pi, \bar{x}_\pi]$$

Similarly, we now compute $\lambda(x)$. Let Φ denote the set of child nodes of X , let

$\Omega = \{j \in \Phi | \mathbf{e}_j^- \neq \mathbf{O}\}$, and let m be the number of elements in Ω . Next, we relabel the child nodes so that nodes Y_1 through Y_m , correspond to the nodes with $\mathbf{e}_j^- \neq \mathbf{O}$. If $m = 0$ then $\lambda(x) \triangleq 1$. If $m = 1$ then $\lambda(x) \triangleq \lambda_{Y_1}(x)$. For $m \geq 2$, we compute $\lambda(x)$ as follows

$$\lambda(x) = f(\mathbf{e}_x^-|x) = f(\mathbf{e}_1^-, \mathbf{e}_2^-, \dots, \mathbf{e}_m^-|x) = \prod_j f(\mathbf{e}_j^-|x) = \prod_j \lambda_j(x)$$

$$\begin{aligned} \lambda(x) &= \prod_j N(H_j x; R_j, \bar{y}_j) \\ &= a \cdot N\left(x; \left[\sum_j H_j^T R_j^{-1} H_j\right]^{-1}, \left[\sum_j H_j^T R_j^{-1} H_j\right]^{-1} \left[\sum_j H_j^T R_j^{-1} \bar{y}_j\right]\right) \\ &= a \cdot N(x; P_\lambda, \bar{x}_\lambda) \end{aligned}$$

where again a is some constant the exact form of which is not important. **As** we will soon see it cancels out during the belief update process. P_λ^{-1} , the inverse of the covariance matrix is called the information matrix. We also define a transformed state vector $z_\lambda \stackrel{\Delta}{=} P_\lambda^{-1} x_\lambda$ and $\therefore \bar{z}_\lambda = P_\lambda^{-1} \bar{x}_\lambda$. It is important to note that P_λ^{-1} is really the covariance of the information state vector z , $P_\lambda^{-1} x_\lambda$. Hence, $N(x; P_\lambda, \bar{x}_\lambda) = a_1 \cdot N(z; P_\lambda^{-1}, \bar{z})$, where again a , is some constant.

If $\mathbf{e}_j^- = 0$ then $\lambda(x) \stackrel{\Delta}{=} 1$.

Combining these two results, we obtain

$$\begin{aligned} BEL(x) &= f(x|e_x^+, e_x^-) \\ &= \frac{f(e_x^-|x, e_x^+) \cdot f(x|e_x^+)}{\int_x f(e_x^-|x, e_x^+) \cdot f(x|e_x^+) \cdot dx} \\ &= \frac{a \cdot N(x; P_\pi, \bar{x}_\pi) N(x; P_\lambda, \bar{x}_\lambda)}{a \cdot \int_x N(x; P_\pi, \bar{x}_\pi) N(x; P_\lambda, \bar{x}_\lambda) dx} \\ &= \frac{N\left(x; [P_\pi^{-1} + P_\lambda^{-1}]^{-1}, [P_\pi^{-1} + P_\lambda^{-1}]^{-1} [P_\pi^{-1} \bar{x}_\pi + P_\lambda^{-1} \bar{x}_\lambda]\right) \cdot N(\bar{x}_\pi, P_\pi + P_\lambda, \bar{x}_\lambda)}{N(\bar{x}_\pi, P_\pi + P_\lambda, \bar{x}_\lambda)} \\ &= N\left(x; [P_\pi^{-1} + P_\lambda^{-1}]^{-1}, [P_\pi^{-1} + P_\lambda^{-1}]^{-1} [P_\pi^{-1} \bar{x}_\pi + P_\lambda^{-1} \bar{x}_\lambda]\right) \\ &= N\left(x; P_\pi - P_\pi [P_\pi + P_\lambda]^{-1} P_\pi, \bar{x}_\pi + P_\pi [P_\pi + P_\lambda]^{-1} (\bar{x}_\lambda - \bar{x}_\pi)\right) \end{aligned}$$

All constants associated with $\lambda(x)$ and $\pi(x)$ cancel out. Hence, the exact form of these constants is unimportant. Therefore, in the remaining part of this section, we will neglect the constants.

During implementation, we begin with the leaf nodes or nodes with no children. Here, λ messages are sent by a node to its parent nodes. This λ propagation stops at the root nodes. Then, starting from the root nodes, a node calculates its belief and sends π messages to each of its children. The propagation stops on reaching a leaf node.

To prescribe how the influence of new information will spread through the network, we need to specify how a typical node, say X, will compute its outgoing messages $\lambda_x(u_i)$, $i=1, \dots, n$, and $\pi_{y_j}(\mathbf{x})$, $j=1, \dots, m$, from the incoming messages $\lambda_{y_j}(\mathbf{x})$, $j=1, \dots, m$ and $\pi_x(u_i)$, $i=1, \dots, n$. This is done next.

Top Down Propagation: Message to Children

Consider the message $\pi_{y_j}(\mathbf{x})$, which node X sends to its jth child Y_j ($j=1, 2, \dots, m$), we note that it is conditioned on all data except a subset $\mathbf{e}_j$ of variables that connect to X via Y_j . Therefore,

$$\begin{aligned}\pi_{y_j}(x) &= f(x | \mathbf{e} - \mathbf{e}_j^-) = BEL(x | \mathbf{e}_j^- = \phi) \\ &= N(x; P_{y_j}^+, \bar{x}_{y_j}^+) \\ \pi_{y_j}(x_j) &= N(y_j; H_j P_{y_j}^+ H_j^T + R_j, H_j^T \bar{x}_{y_j}^+)\end{aligned}$$

So, $\lambda_{y_j}(\mathbf{x})$ can be computed by the method of the last section with the assumption that $\lambda_{y_j}(\mathbf{x}) = 1$. Hence,

$$\begin{aligned}P_{y_j}^+ &= \left[P_\pi^{-1} + \sum_{k \neq j} H^T R_k^{-1} H \right]^{-1} \\ &= P_\pi - P_\pi \left[P_\pi + \sum_{k \neq j} H^T R_k^{-1} H \right]^{-1} P_\pi \\ \bar{x}_{y_j}^+ &= \left[P_\pi^{-1} + \sum_{k \neq j} H^T R_k^{-1} H \right]^{-1} \left[P_\pi^{-1} \bar{x}_\pi + \sum_{k \neq j} H^T R_k^{-1} \bar{y}_k \right] \\ &= \bar{x}_\pi + \left[\sum_{k \neq j} H^T R_k^{-1} H \right]^{-1} \left[P_\pi + \sum_{k \neq j} H^T R_k^{-1} H \right]^{-1} \left(\sum_{k \neq j} H^T R_k^{-1} \bar{y}_k - \bar{x}_\pi \right)\end{aligned}$$

Bottom Up Propagation: Message to Parents

Consider the message $\lambda_x(u_i)$, which node X sends to its ith parent U_i . We divide the evidence $\mathbf{e}$ into its disjoint components $\mathbf{e}_i^+$, $i=1, \dots, n$ and $\mathbf{e}_j^-$, $j=1, \dots, m$, and condition $\lambda_x(u_i)$ on all parents of X. For notational convenience we temporarily denote U_i by U and b_i by b, and let the other parents be indexed by $\mathbf{k}$, ranging from 1 to some n:

$$\begin{aligned}\lambda_x(u) &= f(\mathbf{e} - \mathbf{e}_U^+ | u) \\ &= \int \cdots \int_{u_1 \dots u_n} f(e_1^+, \dots, e_n^+, e_1^-, \dots, e_m^- | u_1, \dots, u_n, x, u) \cdot f(u_1, \dots, u_n, x | u) dx \cdot du_1 \cdots du_n\end{aligned}$$

Consider the first distribution in the integrand

$$\begin{aligned}
f(e_1^+, \dots, e_n^+, e_1^-, \dots, e_m^- | u_1, \dots, u_n, x, u) &= f(e_1^-, \dots, e_m^- | x) \cdot f(e_1^+, \dots, e_n^+ | u_1, \dots, u_n, x, u) \\
&= \prod_j \lambda_{\gamma_j}(x) \cdot \prod_k f(\mathbf{e}_k^+ | u_k) \\
&= \lambda(x) \prod_k \frac{f(u_k | \mathbf{e}_k^+) f(\mathbf{e}_k^+)}{\int_{\mathbf{e}_k^+} f(u_k | \mathbf{e}_k^+) f(\mathbf{e}_k^+) d\mathbf{e}_k^+}
\end{aligned}$$

Next, consider the second distribution in the integral

$$\begin{aligned}
f(u_1, \dots, u_n, x | u) &= f(x | u, u_1, \dots, u_n) f(u_1, \dots, u_n) \\
&= f(x | u, u_1, \dots, u_n) \prod_k f(u_k) \\
&= f(x | u, u_1, \dots, u_n) \prod_k \int_{\mathbf{e}_k^+} f(u_k | \mathbf{e}_k^+) f(\mathbf{e}_k^+) d\mathbf{e}_k^+
\end{aligned}$$

Substituting

$$\begin{aligned}
\lambda_x(u) &= \int_{u_1} \cdot \int_{u_n} \int_x \lambda(x) \prod_k f(u_k | \mathbf{e}_k^+) f(\mathbf{e}_k^+) f(x | u, u_1, \dots, u_n) dx \cdot du_1 \cdot \dots \cdot du_n \\
&= \int_{u_1} \cdot \int_{u_n} \int_x \lambda(x) \prod_k \pi_x(u_k) f(x | u, u_1, \dots, u_n) dx \cdot du_1 \cdot \dots \cdot du_n
\end{aligned}$$

where

$$\begin{aligned}
\lambda(x) &= N(x; P_\lambda, \bar{x}_\lambda) = a \cdot N(z; P_\lambda^{-1}, \bar{z}) \quad \pi_x(u_k) = N(u_k; P_{u_k}^+, \bar{u}_k) \\
f(x | u, u_1, \dots, u_n) &= N\left(x; Q, Bu + \sum_k B_k u_k\right)
\end{aligned}$$

Using the properties for vector normal distributions and integrating with respect to x

$$\begin{aligned}
\lambda_x(u) &= \int_u \cdot \int_{u_n} \int_x N(x; P_\lambda, \bar{x}_\lambda) \prod_k N(u_k; P_{u_k}^+, \bar{u}_k) N\left(x; Q, Bu + \sum_k B_k u_k\right) dx \cdot du_1 \cdot \dots \cdot du_n \\
\lambda_x(u) &= N\left(Bu; P_\lambda + Q + \sum_k B_k P_{u_k}^+ B_k^T, \bar{x}_\lambda - \sum_k B_k \bar{u}_k\right) \\
&= N\left(u; P_x(u) = \left[B^T \left(P_\lambda + Q + \sum_k B_k P_{u_k}^+ B_k^T \right)^{-1} B \right]^{-1}, P_x(u) \left[B^T \left(P_\lambda + Q + \sum_k B_k P_{u_k}^+ B_k^T \right)^{-1} \left(\bar{x}_\lambda - \sum_k B_k \bar{u}_k \right) \right] \right)
\end{aligned}$$

$$= N\left(P_x^{-1}(u) \cdot u; P_x^{-1}(u) = B^T \left(P_\lambda + Q + \sum_k B_k P_{u_k}^+ B_k^T \right)^{-1} B \left(P_\lambda + Q + \sum_k B_k P_{u_k}^+ B_k^T \right)^{-1} \left(\bar{x}_\lambda - \sum_k B_k \bar{u}_k \right) \right)$$

Alternate Form for Update

We can simplify the belief update process for the parent nodes u_i

$$P_\pi(u_i) = N(u_i; P_{u_i}, \bar{u}_i) = N(B_i u_i; B_i P_{u_i} B_i^T, B_i \bar{u}_i)$$

$$P_\lambda(u_i) = N\left(B_i u_i; P_\lambda + Q + \sum_{k \neq i} B_k P_{u_k} B_k^T, \bar{x}_\lambda - \sum_{k \neq i} B_k \bar{u}_k \right)$$

Combining the above two equations to update the belief in u_i results in

$$N(B_i u_i; B_i P_{u_i}^{new} B_i^T, B_i \bar{u}_i^{new})$$

$$B_i P_{u_i}^{new} B_i^T = (B_i P_{u_i} B_i^T) - (B_i P_{u_i} B_i^T) \left[P_\lambda + Q + \sum_k B_k P_{u_k} B_k^T \right]^{-1} (B_i P_{u_i} B_i^T).$$

$$B_i \bar{u}_i^{new} = B_i \bar{u}_i + (B_i P_{u_i} B_i^T) \left[P_\lambda + Q + \sum_k B_k P_{u_k} B_k^T \right]^{-1} \left(\bar{x}_\lambda - \sum_k B_k \bar{u}_k \right)$$

Therefore,

$$N(u_i; P_{u_i}^{new}, \bar{u}_i^{new})$$

$$= N\left(u_i; P_{u_i} - P_{u_i} B_i^T \left[P_\lambda + Q + \sum_k B_k P_{u_k} B_k^T \right]^{-1} B_i P_{u_i}, \bar{u}_i + P_{u_i} B_i^T \left[P_\lambda + Q + \sum_k B_k P_{u_k} B_k^T \right]^{-1} \left(\bar{x}_\lambda - \sum_k B_k \bar{u}_k \right) \right)$$

When the above formula is used we need $P_\lambda = \left[\bigcap_j H_j^T R_j^{-1} H_j \right]^{-1}$. We can remove this requirement by noting

$$P_\lambda^{-1} = \sum_{j=1}^m H_j^T R_j^{-1} H_j \stackrel{A}{=} H^T R^{-1} H$$

$$H = [H_1^T, \dots, H_m^T]^T \quad R = \text{blockdiag}\{R_1, \dots, R_m\}$$

$$\therefore R = H P_\lambda H^T \quad P_\lambda^{-1} = H^T [H P_\lambda H^T]^{-1} H$$

where m is the number of children nodes of x through which there is evidence. But

$$\left[P_\lambda + Q + \sum_k B_k P_{u_k} B_k^T \right]^{-1} = H^T \left[H \left(Q + \sum_k B_k P_{u_k} B_k^T \right) H^T + R \right]^{-1} H$$

Further,

$$y = [y_1^T, \dots, y_m^T]^T \text{ and } \bar{y} = [\bar{y}_1^T, \dots, \bar{y}_m^T]^T.$$

Using this

$$P_\lambda^{-1} \bar{x}_\lambda = \sum_{j=1}^m H^T R_j^{-1} \bar{y}_j = H^T R^{-1} \bar{y}$$

$$H^T R^{-1} H \bar{x}_\lambda = H^T R^{-1} \bar{y}$$

$$\therefore H \bar{x}_\lambda = \bar{y}$$

$$N(u_i; P_{u_i}^{new}, \bar{u}_i^{new}) = N \left(u_i; P_{u_i} - P_{u_i} B_i^T H^T \left[H \left(Q + \sum_k B_k P_{u_k} B_k^T \right) H^T + R \right]^{-1} H B_i P_{u_i}, \right. \\ \left. \bar{u}_i + P_{u_i} B_i^T H^T \left[H \left(Q + \sum_k B_k P_{u_k} B_k^T \right) H^T + R \right]^{-1} \left(\bar{y} - H \sum_k B_k \bar{u}_k \right) \right)$$

is an alternate form for updating the beliefs. The above alternative form is particularly useful in deriving the decentralized form of the Kalman filter.

2.2.1.2.2 Inference Using Topology Transformation

Another method for doing inference in a belief consists of eliminating nodes and transforming the diagram through a series of transformations (Olmsted 1984; Rege and Agogino 1988). For vector Gaussian continuous networks we require two basic transformations, which we will now develop.

Parent Node Removal

The process of removing a parent node, i.e., node propagation in the direction of the arrow, corresponds to taking the expectation with respect to that parent variable. We make use of the generalized sum rule

$$\Pr(X|\xi) = \int_y \Pr(X, Y|\xi) dy.$$

Fig. 2.2.1.2.2-1 shows the change in topology of the vector Gaussian belief network due to this transformation. Let,

$$x = Fu + \sum_i F_i u_i + v.$$

correspond to the relationship between x and its parent nodes. Let, Q be the covariance of the noise in the relationship. Then after the transformation we have the relationship

$$x = F\bar{u} + \sum_j F_j u_j + v'.$$

where the distribution for u was given by $N(u; P_u, \bar{u})$, the covariance of noise v' is given by $Q + FP_u F^T$.

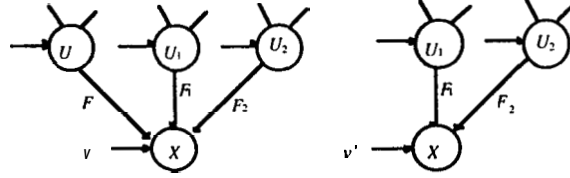


Fig. 2.2.1.2.2-1: Topology transformation after parent node removal

Arc Reversal and Propagation

The process of arc reversal, and node propagation corresponds to applying Bayes' rule, which for Gaussian random variables (for a proof see Bar-Shalom and Li 1993; pp 43-44), is given by

$$\text{Mean } E[x|y] = \hat{x} = \bar{x} + P_{xy} P_{yy}^{-1} (y - \bar{y})$$

$$\text{Variance: } \text{Cov}[x|y] = P_{xx|y} = T_{xx}^{-1} = P_{xx} - P_{xy} P_{yy}^{-1} P_{yx}$$

which is the conditional probability of x given y . Here, y corresponds to the value of the variables in the child node, which is to be removed from the graph. After the application of Bayes Rule the node is removed from the graph as shown in Fig. 2.2.1.2.2-2.

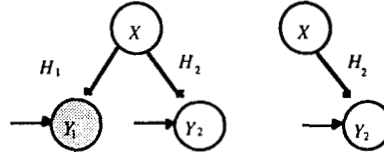


Fig. 2.2.1.2.2-2: Topology transformation after arc reversal and propagation

2.2.1.3 Rules for Vector Gaussian Distribution⁴

1. $N(x; P, \bar{x}) = |2\pi P|^{-1/2} \exp\left(-\frac{1}{2}(x - \bar{x})^T P^{-1} (x - \bar{x})\right)$
2. $N(x; P, \bar{x}) = N(\bar{x}, P, x)$
3. $N(y = Ax + B; P_y, \bar{y}) = a \cdot N\left(x; P_x = [A^T P_y^{-1} A]^{-1}, P_x [A^T P_y^{-1} (\bar{y} - B)]\right)$ where a is some constant.
4. $N(x; P_1, \bar{x}_1) \cdot N(x; P_2, \bar{x}_2) = a \cdot N\left(x; [P_1^{-1} + P_2^{-1}]^{-1}, [P_1^{-1} + P_2^{-1}]^{-1} [P_1^{-1} \bar{x}_1 + P_2^{-1} \bar{x}_2]\right)$ where the constant a is given by $a = N(\bar{x}_1, P_1 + P_2, \bar{x}_2)$
5. $N(x; P_1, \bar{x}_1) \cdot N(x; P_2, \bar{x}_2) = a \cdot N\left(x; P_2 - P_2 [P_1 + P_2]^{-1} P_2, \bar{x}_2 + P_2 [P_1 + P_2]^{-1} (\bar{x}_1 - \bar{x}_2)\right)$ here again a is some constant.
6. $\prod_i N(Ax; P_i, \bar{y}_i) = a \cdot N\left(x; \left[\sum_i A^T P_i^{-1} A\right]^{-1}, \left[\sum_i A^T P_i^{-1} A\right]^{-1} \left[\sum_i A^T P_i^{-1} \bar{y}_i\right]\right)$ where again a is some constant.
7. $\int N(y; P_1, \bar{x}_1) N(y; P_2, \bar{x}_2) dy = N(x; P_1 + P_2, \bar{x}_1)$
8. $\int N(x; P_x, y) N(y; P_y, \bar{y}) dy = N[x; P_x + P_y, \bar{y}]$

⁴ Refer to Alag (1996; Chapter 2) for **detailed** proofs.

$$9. \int \cdots \int \prod_{i=1}^n N(u_i, P_i, \bar{u}_i) \cdot N\left(\sum_j B_j u_j, Q, x\right) du_1 \cdots du_n = N\left(x; Q + \sum_j B_j P_j B_j^T, \sum_j B_j \bar{u}_j\right)$$

Matrix Inversion Lemma

$$(P^{-1} + H^T R^{-1} H)^{-1} = P - P H^T (H P H^T + R)^{-1} H P$$

where P is n x n, H is m x n, and R is m x m.

2.2.1.4 Probabilistic Data Association Filter (PDAF)

The probabilistic data association filter (**PDAF**), (Bar-Shalom and Tse 1975) is used for tracking when the source of measurement data is uncertain. The following assumptions (Bar-Shalom and Fortmann 1988) are made.

1. There is only one target of interest.
2. Of the many validated measurements, only one can be target originated if the target was detected. The remaining measurements are assumed due to false alarms or “residual clutter” and are modeled as independent identically distributed random variables with uniform spatial distributions.
3. Undesirable returns occur independently in time and space and the location, nature, or number of these returns cannot be inferred from past data.
4. The probability of each return being correct, conditioned on the past data, is the same, i.e., no target signature information is used.
5. The state of the system is assumed to be normally distributed (Gaussian) according to the latest estimate and covariance matrix.

The PDAF is restricted to the case of tracking a single target (geometric feature in vision, robotics problems). It provides a state estimate based on the weighted sum of all the measurements in the validation region of the track. Obviously this algorithm is suboptimal because measurements that should not have been associated with the track are used to update its state estimate. Consequently, there is a risk that the state estimate can diverge from the actual state under certain circumstances and result in a track loss. The advantage is that it requires fixed and finite computational resources (Cox and Leonard 1994). The algorithm is adaptive to the environment in the sense that the actual number of observed returns will affect the confidence in terms of the covariance, of the estimate. The joint probability data association filter (JPDAF) extends the PDAF to a fixed known number of targets.

The corresponding Vector Dynamic Belief Network (VDBN) is shown in Fig. 2.2.1.4-1. The VDBN consists of multiple z nodes corresponding to the multiple measurements. There is a discrete node H corresponding to the hypothesis that only one of the measurements is target originated.

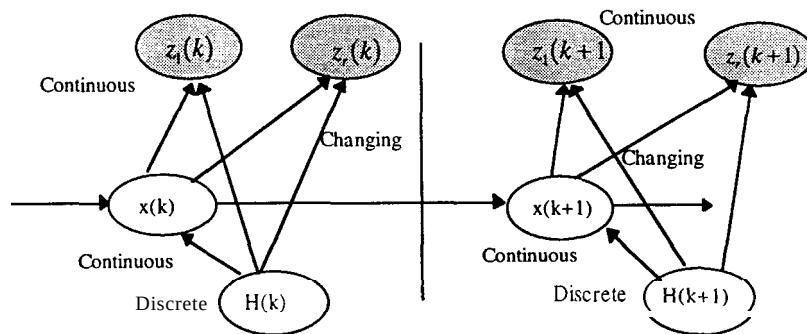


Fig. 2.2.1.4-1: The VDBN for the Probabilistic Data Association Filter

The discrete node H corresponds to the hypothesis

$\theta_i^A = i$ th reading i is target originated and others are due to false alarms $i = 1, \dots, m_k$

$\theta_0^A =$ all the measurements are due to false alarms

Note that there are no temporal transitions of the discrete node as per assumption 3, which states that no inference on the number of incorrect returns can be made from past data.

Initialization

The process is the same as in the multiple model algorithms and the Kalman filter.

Prediction

We begin at time k , where all the evidence up to time k has been taken into account. Our estimate for the state is given by $N(x(k); P(k|k), \hat{x}(k|k))$.

The VDBN for this stage is shown in Fig. 2.2.1.4-2. To update the belief in the node $x(k+1)$ we have

$$\pi(x(k+1)) = N(x(k+1); F(k+1)P(k|k)F(k+1)^T + Q; F(k+1)\hat{x}(k|k))$$

$$\lambda(x(k+1)) = 1$$

$$Bel(x(k+1)) = \pi(x(k+1)) = N(x(k+1); P(k+1|k), \hat{x}(k+1|k))$$

Fig. 2.2.1.4-2: The VDBN at the prediction stage for the PDAF algorithm.

Roll Up

The slice at time k is removed and the prior beliefs in all nodes are simply the projected beliefs.

Estimation

Fig. 2.2.1.4-3 shows the VDBN at the beginning of the estimation stage.

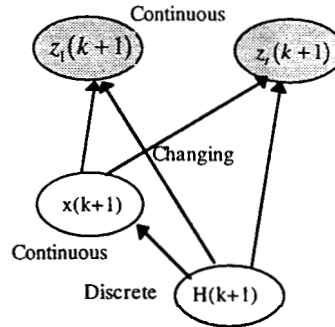


Fig. 2.2.1.4-3: The VDBN at the beginning of the estimation phase of the PDAF algorithm

Validation

Fig. 2.2.1.4-4 shows the VDBN during the validation process. For the formation of the validation gate we need to solve the inverse problem. Here we assume that no evidence is available at $z(k+1)$, and we need to estimate the distribution for $z(k+1)$ and compare it with the sensor output.

The expected distribution for the sensor node is obtained by observing

$$\Pr(z(k+1)|k) = N(z(k+1); S(k+1), \hat{z}(k+1|k))$$

$$= N(z(k+1); H(k+1)P(k+1|k)H(k+1)^T + R(k+1), H(k+1)x(k+1|k))$$

The validation region is obtained by defining

$$v(k+1) = z(k+1) - \hat{z}(k+1|k)$$

Accept the reading if

$$v^T(k+1)S^{-1}(k+1)v(k+1) \leq \gamma$$

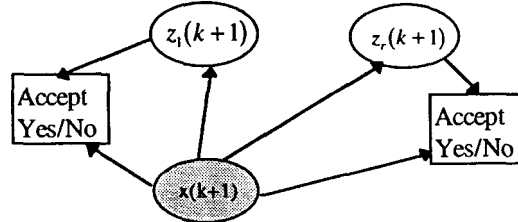


Fig. 2.2.1.4-4: The validation process for the PDAF algorithm.

Estimating the a priori probabilities

The a priori probabilities for the $H(k+1)$ node are assumed to be diffuse, i.e., all the states are equally likely. A Poisson distribution can also be assumed for this. Hence, based on the number of measurements, the a priori probabilities are approximated by

$$\Pr(\theta_i(k)|m_k) = \gamma_i(m_k) = \begin{cases} P_D P_G / m_k, & i = 1, \dots, m_k \\ 1 - P_D P_G, & i = 0 \end{cases}$$

where P_D is the target detection probability, P_G is the probability that the reading will lie in the validation region when it is correct, and m_k is the number of measurements that lie in the validation region for the k th sample.

Estimation: Node $x(k+1)$

Note that the VDBN at this stage is not connected singly. We will use conditioning to break the loop. For this we instantiate the node $H(k+1)$, use the propagation rules to update the distribution for $x(k+1)$, and combine the various estimates using the likelihood of each instantiation (Fig. 2.2.1.4-5).

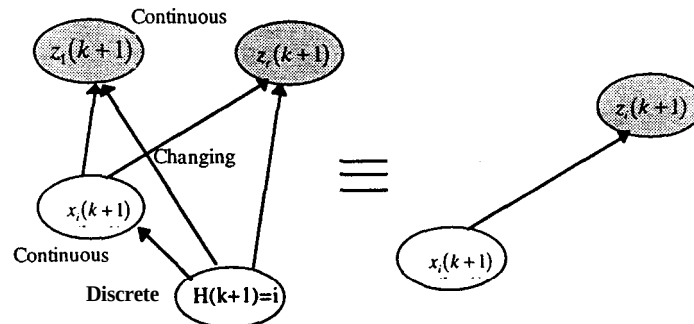


Fig. 2.2.1.4-5: The conditioned VDBN for the PDAF algorithm.

$$\pi(x(k+1)) = N(x(k+1); P(k+1|k), \hat{x}(k+1|k))$$

$$\lambda(x(k+1)) = N \left[x(k+1), \left[H(k+1)^T R^{-1}(k+1) H(k+1) \right]^{-1}, \right. \\ \left. \left[H(k+1)^T R^{-1}(k+1) H(k+1) \right]^{-1} \left[H^T(k+1) R^{-1}(k+1) z_i(k+1) \right] \right]$$

Therefore, we obtain the new estimate in the information form first

$$P_j^{-1}(k+1|k+1) = P_j^{-1}(k+1|k) + H(k+1)^T R^{-1}(k+1) H(k+1) \\ P_j^{-1}(k+1|k+1) \hat{x}_j(k+1|k+1) = P_j^{-1}(k+1|k) \hat{x}_j(k+1|k) + H(k+1)^T R^{-1}(k+1) z(k+1)$$

Second, in the Kalman filter form

$$N(x(k+1); P_j(k+1|k+1), \hat{x}_j(k+1|k+1)) \\ = N \left(x(k+1); P_j(k+1|k) - W_j(k+1) S(k+1) W_j(k+1)^T, \right. \\ \left. \hat{x}_j(k+1|k) + W_j(k+1) (z_j(k+1) - \hat{z}(k+1|k)) \right)$$

$$W_j(k+1) \stackrel{\Delta}{=} P_j(k+1|k) H(k+1)^T S(k+1)$$

Next, we estimate the conditional probabilities from the node $H(k+1)$ to the nodes $z_i(k+1)$. The i th state corresponds to the hypothesis that the i th reading is target originated, while the others are false alarms. In the case of a false alarm it is assumed that the density of a measurement is uniform in the validation region. For each of the conditioned cases we have the following probabilities

$$\Pr(z_1(k+1), \dots, z_m(k+1) | \theta_i) = V_k^{-m_k+1} P_G^{-1} N(z_i(k+1); S(k+1), \hat{z}(k+1)), \quad i = 1, \dots, m_k \\ = V_k^{-m_k} \quad i = 0$$

Therefore, updating the probabilities for the discrete node $H(k+1)$, we have

$$\Pi(H(k+1)) = [1 - P_D P_G, P_D P_G / m_k, \dots, P_D P_G / m_k] \\ \lambda(H(k+1)) = [V_k^{-m_k}, V_k^{-m_k+1} P_G^{-1} N(v_1(k+1); S(k+1), 0), \dots, V_k^{-m_k+1} P_G^{-1} N(v_{m_k}(k+1); S(k+1), 0)] \\ Bel(H(k+1)) = \alpha \left[\begin{aligned} & (1 - P_D P_G) V^{-1} \frac{P_D}{m_k} |2\pi S(k+1)|^{-1/2} \exp\left(-\frac{1}{2} v_1(k+1)^T S^{-1}(k+1) v_1(k+1)\right), \dots, \\ & \frac{P_D}{m_k} |2\pi S(k+1)|^{-1/2} \exp\left(-\frac{1}{2} v_{m_k}(k+1)^T S^{-1}(k+1) v_{m_k}(k+1)\right) \end{aligned} \right] \\ = \alpha \left[\begin{aligned} & (1 - P_D P_G) V^{-1} m_k |2\pi S(k+1)|^{1/2}, \exp\left(-\frac{1}{2} v_1(k+1)^T S^{-1}(k+1) v_1(k+1)\right), \dots, \\ & \exp\left(-\frac{1}{2} v_{m_k}(k+1)^T S^{-1}(k+1) v_{m_k}(k+1)\right) \end{aligned} \right]$$

where α is the corresponding normalizing constant. Hence,

$$Bel(H(k+1)) \stackrel{\Delta}{=} [\beta_0(k+1), \beta_1(k+1), \dots, \beta_{m_k}(k+1)].$$

State Update

The conditioned states, along with their corresponding probabilities are combined together to update the state. This corresponds to combining the Gaussian mixture,

$$\begin{aligned}\hat{x}(k+1|k+1) &= \sum_{j=0}^{m_k} \beta_j(k+1|k+1) \hat{x}_j(k+1|k+1) \\ P(k+1|k+1) &= \sum_{j=0}^{m_k} \beta_j(k+1|k+1) P_j(k+1|k+1) \\ &+ \sum_{j=0}^{m_k} \beta_j(k+1|k+1) \hat{x}_j(k+1|k+1) \hat{x}_j(k+1|k+1)^T - \hat{x}(k+1|k+1) \hat{x}(k+1|k+1)^T\end{aligned}$$

The above expression can be simplified to

$$\begin{aligned}\hat{x}(k+1|k+1) &= \hat{x}(k+1|k) + W(k+1) \left[\sum_{j=1}^{m_k} \beta_j(k+1|k+1) (z_i(k+1) - \hat{z}(k+1)) \right] \\ P(k+1|k+1) &= \beta_o(k+1|k+1) P(k+1|k) + (1 - \beta_o(k+1|k+1)) [P(k+1|k) - W(k+1) S(k+1) W(k+1)^T] \\ &+ W(k+1) \left[\sum_{j=1}^{m_k} \beta_j(k+1|k+1) v_i(k+1) v_i(k+1)^T - \iota(k+1) v(k+1)^T \right] W(k+1)^T \\ v(k+1) &= \sum_{j=1}^{m_k} \beta_j(k+1|k+1) v_j(k+1)\end{aligned}$$

2.2.1.5 Sensor-Data Fusion

Data refers to the actual sensor measurements. A sensor is a device that reacts to physical or chemical stimuli by measuring, or recording these physical phenomena. Fusion is the process of combining these data or information in such a way that the combination provides more information than the sum of the individual parts. A number of definitions for sensor-data fusion have been employed (Manyika and Durrant-Whyte **1994**). Among them, the one by Abidi and Gonzales (1992, Chapter 1) is the most comprehensive as it includes the notion of Sensor Synergy: "Data fusion deals with the synergistic combination of information made available by various knowledge sources such as sensors, in order to provide a better understanding of a given scene."

2.2.1.5.1 Architectures

Architecture refers to the way sensing agents are inter-connected in order to facilitate information fusion. Data fusion architectures can be broadly divided into centralized and decentralized architectures (Hashemipour et al. **1988**; Rao et al. **1993a**; Manyika and Durrant-Whyte **1994**; Clark and Yuille **1990**). In a centralized structure (Fig. 2.2.1.5.1-1) all the observations of the sensors are passed back to a central processing facility which fuses the data, or there is some level of local embedded processing capability in each sensing node. However, there is still a reliance on a central processing facility to perform the global data fusion. Both these methods require a central processing unit to carry out the data fusion. The need to relieve the computational burden at the central processor leads to hierarchical systems. Whilst ideal for coordination and control, hierarchical architectures are vulnerable to processor failure, computational bottlenecks and inflexibility. The need to overcome these shortcomings and the recent trend towards autonomous systems have led to the development of distributed architectures (Iyenger et al. **1991**). A refinement of these are the decentralized architectures described by Hashemipour et al. (1988), Durrant-Whyte et al. (1990), and Manyika and Durrant-Whyte (1994). In decentralized data fusion (Fig. 2.2.1.5.1-2) there is no central processing facility. Each node communicates local information to each other so each one arrives at the common global consensus.

Clark and Yuille (1990) divide centralized fusion algorithms into three parts as shown in Fig. 2.2.1.5.1-1. In the first type the central fused estimate is a weighted combination of the measurements at two or more sensor nodes. An example of this is the fusion of two or more directly redundant measurement readings. In this case fusion decreases the uncertainty in the estimate. Usually, the weightings of one information source with respect to another are derived from measure of the relative reliabilities of the two measurement sources. In the second fusion class, two or more sources of information are needed to arrive at a centralized estimate. All the measurements are required to give an estimate. An example of this is the process by which functional redundancy is created by using nonlinear regression or neural networks. The third fusion class is a combination of the above two methods.

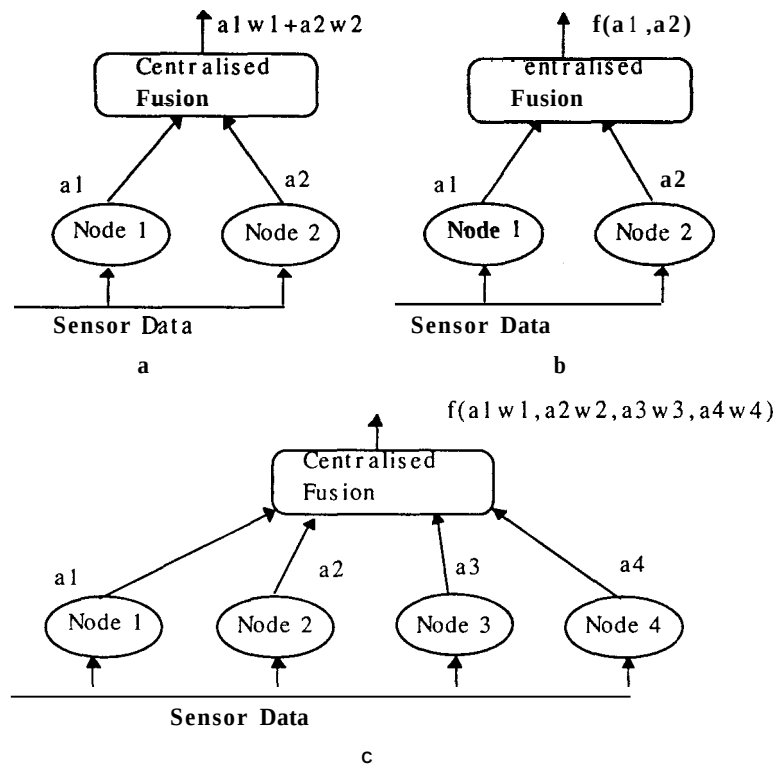


Fig. 2.2.1.5.1-1: Centralized Architecture for data fusion: (a) Weighted combination of the outputs (b) Two or more sources of information combine to give an estimate (e.g. Neural Network estimate) (c) Combination of the first two methods

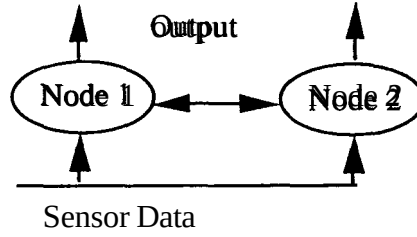


Fig. 2.2.1.5.1-2: Decentralized Architecture for Data Fusion

Decentralized systems have several advantages over centralized systems, some of them are (Manyika and Durrant-Whyte 1994):

- **Modularity:** This is a result of the fact that local sensing and global data fusion takes place at the sensor node itself. This is facilitated by each sensor node being fully autonomous with its own sensing, processing and communication facilities.
- **Scalability and Flexibility:** Scaling the system is merely a matter of adding or removing sensors, as all the functionality is localized.
- **Survivability:** Due to the absence of a central processor, the system can withstand the loss of nodes and performance is gracefully degraded.

2.2.1.5.2 Sensor Fusion and Inference

Much work has been done in developing methods for combining information from different sensors. The basic approach has been to pool the information using what are essentially “weighted averaging” techniques of varying degrees of complexity.

We briefly review some of the previous work in this area. For more details readers are referred to Manyika and Durrant-Whyte (1994). Stone (1961) proposes pooling of various estimates based on a probabilistic representation of information. There are several variations on this theme which are discussed by Berger (1985). An initial discussion on sensor probabilistic fusion can be found in Durrant-Whyte (1987). Geometric methods, which are a slight variation on the theme yielding similar results has been proposed for sensor fusion by Nakamura (1992) and Lee et al. (1990). Non-probabilistic methods such as Dempster Shafer evidential reasoning have also been used for fusion.

Inferring the state of nature given a probabilistic representation is in general a well understood problem in classical statistical estimation. Representative are methods such as Bayesian estimation (Berger 1985; Pearl 1988), Least Squares estimation, Kalman filtering, etc. Application of the Kalman filter and its derivatives have been widely explored in the literature.

Alternative methods for inferring the state using measurements from disparate sources have also been developed and include methods which make use of neural networks and expert Systems. Kim (1992) considers the process of sensor fusion as a process of estimation of an unknown quantity by a decision maker based on advice or multiple inputs (probability distributions) from experts (sensors).

An intuitive method for sensor fusion, assuming sensors which are direct measurements x_j^T of the state x , is to simply obtain a weighted average,

$$\hat{x} = \sum_j w_j x_j^T$$

where $\sum_j w_j = 1$. The main drawback of this rather simplistic approach is that it does not take into

account the uncertainty inherent in sensor observations and so the resulting estimate $\hat{x}$ does not incorporate any notion of optimality in a statistical sense. We will consider fusion in a probabilistic sense.

We define a set of all observations made by the set of sensors up to time-step k as

$$\{Z^K\} = \cup_i Z_i^k$$

$$\text{where } Z_i^k = \{z_i(1), z_i(2), \dots, z_i(k)\}$$

and Z_i^k is the i th set of observations up to the time-step k .

2.2.1.5.2.1 Linear Opinion Pool

In the linear opinion pool (Stone 1961) the posteriors from each information source are combined linearly,

$$\Pr(x|\{Z^K\}) = \sum_j w_j \Pr(x|Z_j^k)$$

w_j is a weight such that, $0 \leq w_j \leq 1$ and $\sum_j w_j = 1$. The weight w_i reflects the significance attached to the information source i . The weights can also be used to model the trustworthiness of an information source. Though a general methodology for obtaining the weights w_i has not been forthcoming, problem specific methods have been developed (Manyika and Durrant-Whyte 1994).

2.2.1.5.2.2 Independent Opinion Pool

The independent opinion pool makes the assumption that the information obtained conditioned on the observation set is independent. The independent opinion pool is defined by the product

$$\Pr(x|\{Z^K\}) = \alpha \prod_j \Pr(x|Z_j^k)$$

where α is a normalizing constant. The independent opinion pool places an unwarranted reinforcement of opinion when prior information at each node is common. It is appropriate only when the priors are obtained independently on the basis of subjective priors at each information source.

2.2.1.5.2.3 Independent Likelihood Pool

Here we assume that for each information source i , the likelihood $\Pr(Z_i^k|x)$ is independent. Hence,

$$\Pr(x|\{Z^K\}) = \frac{\Pr(x) \prod_j \Pr(Z_j^k|x)}{\Pr(Z_1^k, Z_2^k, \dots, Z_N^k)}$$

which can be written recursively as

$$\Pr(x|\{Z^K\}) = \alpha \Pr(x|\{Z^{k-1}\}) \left[\prod_{i'} \underbrace{\Pr(z_i(k)|x)}_{\text{likelihood}} \right]$$

where α is a normalizing constant independent of x .

2.2.1.5.2.4 Remarks

The independent opinion pool and the independent likelihood pool more accurately model the situation in multi-sensor systems where the conditional distributions of the observations can be shown to be independent. The choice between the two depends on the origin of the prior information. The independent likelihood pool is more appropriate in most sensing systems because the prior information in these systems tends to be from the same origin. The linear opinion pool is useful if there are dependencies between information sources and the dependencies are known a priori.

2.2.1.5.3 Fusion Methods

In this section, we will look at some of the methods for data-fusion which are based on the combination of multiple estimates. The first method using a priori information is based on the well known Kalman filter, while the remaining algorithms are sub-optimal.

2.2.1.5.3.1 Using A Priori Information

The best way to understand this fusion process is to consider the various estimates: measurements from various sensors, and the predicted estimate as consisting of normal distributions, characterized by a mean and a variance. The reliability of the sensor- the sensor noise covariance used in the estimation process, corresponds to the a priori information that is used in the fusion process. Each estimate is combined by a weight which is inversely proportional to its spread- its noise covariance. This process of fusion is optimal in the statistical sense and was addressed in Alag (1996). Fig. 2.2.1.5.3.1- 1 shows the VDBN transformation during the fusion process.

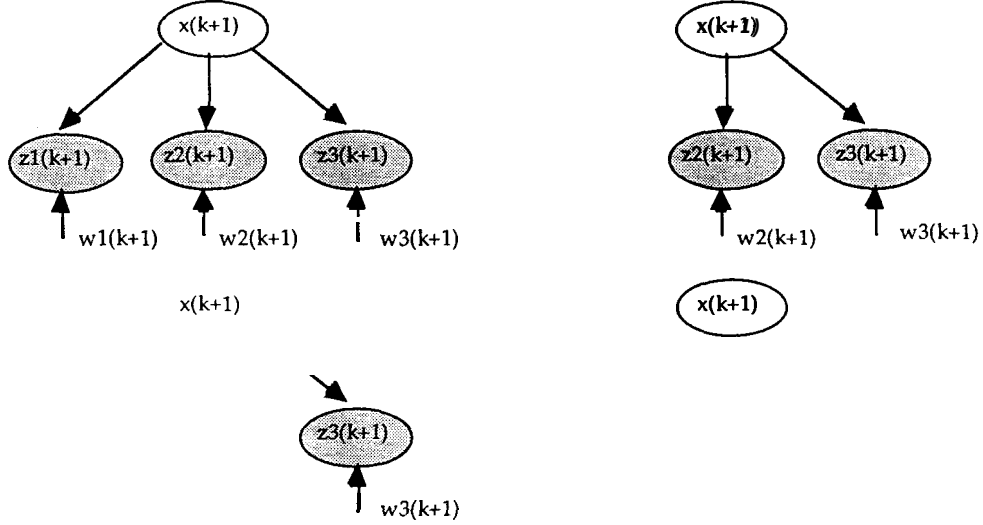


Fig. 2.2.1.5.3.1-1: The topology transformation using a priori information for sensor-data fusion.

2.2.1.5.3.2 Using A Posteriori Information

This is an example of the linear opinion pool method, where the weights for the fusion process are obtained using the predicted sensor reading distributions. This is a particularly useful means of combining information when the sources of information have similar a priori reliabilities- similar noise covariance and in the presence of clutter. This method of fusion is sub-optimal in the presence of Gaussian noise as it combines various estimates into one, assuming that only one reading best describes the expected distribution.

This algorithm is very similar to the PDAF algorithm. The VDBN is the same as for the PDAF algorithm (Fig. 2.2.1.5.3.2-1), except that the discrete node has the hypothesis that the distribution from the i th sensor best approximates the real distribution

This approach is very similar to combining estimates from various distributions (sensor readings) using a reference distribution (predicted reading).

The prediction, roll up, and the validation steps are similar to that of a Kalman filter. However, during the estimation process the different estimates are combined together using their expected distribution and sensor readings. As shown in Fig. 2.2.1.5.3.2-1, the likelihood of each of the readings is obtained and used to obtain the fusion weights. This likelihood is dependent on the expected distribution and the sensor reading.

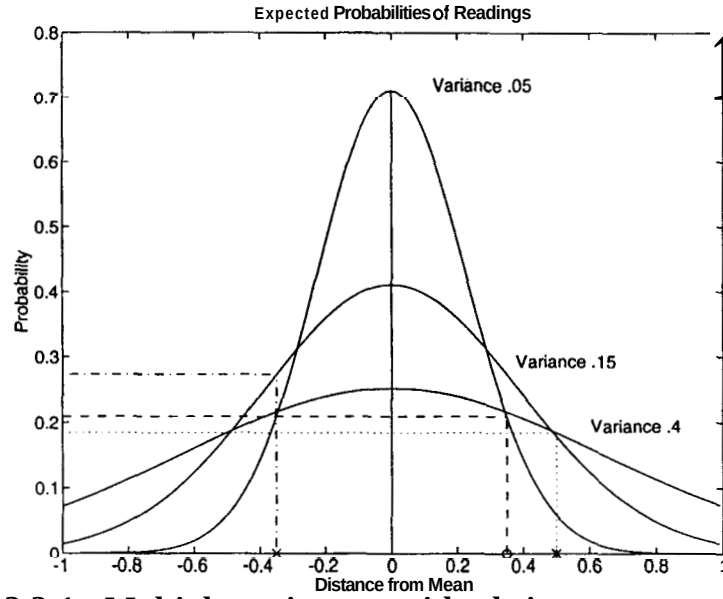


Fig. 2.2.1.5.3.2-1: Multiple estimates with their corresponding expected distributions. The fusion weights are dependent on the likelihood of the estimate

The fusion process consists of the following procedure. Let the predicted estimate be given by, $N(x(k+1); P(k+1|k), i(k+1|k))$. Then, the expected distribution for each of the sensor readings is given by $N(z_i(k+1); S_i(k+1), \hat{z}_i(k+1))$

where

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k) + G(k)u(k)$$

$$\hat{z}(k+1|k) = H(k+1)\hat{x}(k+1|k)$$

$$S_i(k+1) = H(k+1)P(k+1|k)H(k+1)^T + R_i(k+1)$$

The relative weight for each of the reading is obtained by

$$\Pr(\hat{z}(k+1)|\hat{x}(k+1|k)) \triangleq \pi_i$$

$$= |2\pi S_i(k+1)|^{-1/2} \exp\left(-\frac{1}{2}(\hat{x}(k+1|k) - \hat{z}_i(k+1))S_i^{-1}(k+1)(\hat{x}(k+1|k) - \hat{z}_i(k+1))^T\right)$$

and the fusion weights are given by

$$\beta_i = \frac{\pi_i}{\sum_j \pi_j}$$

This corresponds to consolidating the various estimates, given by the following distribution

$$z(k+1) = \sum_j \beta_j z_j(k+1)$$

$$S(k+1) = \sum_i \beta_i S_i(k+1) + \sum_i \beta_i z_i(k+1)z_i^T(k+1) - z(k+1)z^T(k+1)$$

Lastly, this 'combined reading is fused with the predicted distribution to get the estimate for the state. Note that in obtaining the relative weighting of the multiple readings it is also possible to compare the predicted distribution with the sensor estimate. Here, two distributions are compared to get the corresponding fusion weights. This approach of using a distance measure for obtaining the fusion weights was also used by Kim (1992).

In this case, we compare the predicted distribution with the fused estimate using a distance measure. A number of distance measures have been suggested over the years, most of which are based on the likelihood ratio. Jeffery's Divergence Measure (Jeffery 1946) and Bhattacharyya's Coefficient (Bhattacharyya 1943) are based on the likelihood ratio between two conditional density functions. The Kolmogorov Variational Distance, the Matusita Distance, the Kolmogorov-Smirnov Distance, and the Tie Statistic (Kim and Agogino 1991) have also been proposed. We use the Bhattacharyya coefficient due to its computational efficiency and empirical accuracy. As a distance measure it has the desirable property that it decreases or increases accordingly to the probability of error as defined by the Kolmogorov Distance (Kailath 1967). Given two Gaussian density functions with different means and standard deviations, we have the following explicit formula for the Bhattacharyya Coefficient

$$\rho = \sqrt{\frac{2\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2}} e^{-\frac{1}{4} \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}}$$

Using this we calculate the relative likelihood of each hypothesis by the following formula

$$\Pr(x(k+1)|k, Q(k+1)=j) = \pi_j^Q = \sqrt[4]{2 \cdot (F(k)P(k|k)F(k)^T + Q_j) \cdot P(k+1|k+1)}$$

$$\cdot (F(k)P(k|k)F(k)^T + Q_j + P(k+1|k+1))^{-1} \exp \left(-\frac{(\hat{x}(k+1|k+1) - \hat{x}(k+1|k))^2}{4 \cdot F(k)P(k|k)F(k)^T + Q_j + P(k+1|k+1)} \right)$$

2.2.1.5.3.3 Fusion Using Learning

This algorithm is similar to the one in the previous section, except that instead of equal a priori probabilities for the various weights, the prior weights are projected from the previous sampling period. The VDBN for this algorithm is shown in Fig. 2.2.1.5.3.3-1. Note that there is an arrow over time for the discrete node, due to which the discrete probabilities will be predicted over to the next sampling period.

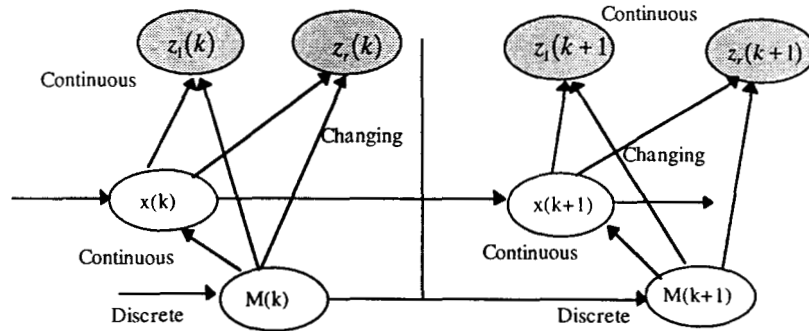


Fig. 2.2.1.5.3.3-1: The VDBN for the fusion algorithm with learning

The discrete node has the following hypothesis:

θ_i = The distribution from the i th sensor best approximates the real distribution.

Initialization

The state transition matrix $\Phi_M(0)$ for the discrete node M, along with the initial probabilities for the node M, are set. As usual, if such information is not available, a uniform distribution among the states and an identity matrix for the transition matrix are assumed. This transition matrix can be learnt on-line through the adaptation process.

Prediction

We begin at time $\mathbf{k}$, where all the evidence up to time $\mathbf{k}$ has been taken into account. Our estimate for the state is given by $N(x(k); P(k|k), \hat{x}(k|k))$. The probabilities for the discrete node are given by $\{m_1(k|k), \dots, m_{n_m}(k|k)\}$ where $\sum_j m_j(k|k) = 1$

Since there is no evidence of $x(k+1)$ and $M(k+1)$ in any of the children nodes, for these two nodes

$$\lambda(x) = 1 \quad \therefore Bel(x) = \pi(x).$$

Hence, for node $M(k+1)$

$$\Pr(M(k+1)=j|k) \stackrel{\Delta}{=} m_j(k+1|k) = \sum_i \Pr(M(k+1)=j|M(k)=i) \Pr(M(k)=i|k)$$

$$m_j(k+1|k) = [m_1(k|k), \dots, m_{n_m}(k|k)] \cdot \Phi_m(k)$$

Roll Up and Validation

The slice at time $\mathbf{k}$ is removed and the prior beliefs in all nodes are simply the projected beliefs. As usual the sensor readings are validated by the formation of the validation gate. The process is similar as in the case of the PDAF algorithm.

Estimation

Using the procedure similar to the method above to obtain the expected distributions for each of the sensor readings, we estimate the conditional probabilities for node M:

$$\begin{aligned} \Pr(\hat{z}(k+1)|\hat{x}(k+1|k)) &\stackrel{\Delta}{=} \pi_i \\ &= |2\pi S_j(k+1)|^{-1/2} \exp\left(-\frac{1}{2}(\hat{x}(k+1|k) - \hat{z}_j(k+1)) S_j^{-1}(k+1) (\hat{x}(k+1|k) - \hat{z}_j(k+1))^T\right) \end{aligned}$$

Hence for node M

$$\lambda(M(k+1)) = [\pi_1, \dots, \pi_{n_m}]$$

$$\Pi(M(k+1)) = [m_1(k+1|k), \dots, m_{n_q}(k+1|k)]$$

$$Bel(M(k+1)) = \alpha [\pi_1 q_1(k+1|k), \dots, \pi_{n_m} m_{n_m}(k+1|k)]$$

$$\alpha^{-1} = \sum_{j=1}^{n_q} \pi_j m_j(k+1|k)$$

The various readings are combined into one,

$$z(k+1) = \sum_j m_j(k+1|k+1) z_i(k+1)$$

This

$$S(k+1) = \sum_i m_j(k+1|k+1) S_i(k+1) + \sum_i m_j(k+1|k+1) z_i(k+1) z_i^T(k+1) - z(k+1) z^T(k+1)$$

is used to update the distribution for $\mathbf{x}$

$$\text{Mean: } \hat{x}(k+1|k+1) = \hat{x}(k+1|k) + W(k+1)(z(k+1) - \hat{z}(k+1|k))$$

$$W(k+1) \stackrel{\Delta}{=} P(k+1|k)H(k+1)^T S(k+1)^{-1}$$

$$\text{Variance: } P(k+1|k+1) = P(k+1|k) - H(k+1)^T S(k+1)^{-1} H(k+1)P(k+1|k)$$

$$\text{or in symmetric form } = P(k+1|k) - W(k+1)S(k+1)W(k+1)^T$$

2.2.1.5.4 Comparison of Fusion Algorithms via Monte Carlo Runs

In many situations, the relative performance of algorithms cannot be evaluated analytically⁵. In such a case, Monte Carlo simulations are carried out to evaluate the relative performance of the algorithms. To compare the relative performance of the algorithms we simulate a Wiener process (also called Brownian motion or a random walk), which is useful in describing the motion of particles under the influence of diffusion. The Wiener process is often used in such comparisons. The advantage of using this process is that no additional uncertainty is added to the system as all system parameters are known exactly. The process is described by the following one-state Markov process

$$x(k+1) = x(k) + w(k)$$

where the covariance of the process noise is $Q = E[w(k)w^T(k)]$. The sensor model is given by

$$z(k) = x(k) + v(k)$$

where the covariance of the sensor noise is $R = E[v(k)v^T(k)]$.

To compare various algorithms we look at the absolute error between the fused estimate and the real value. A larger number of Monte Carlo runs yields a smaller variability (error) for the resulting estimate. Denoting by $C_i, i=1, \dots, N$ the mean absolute error for each independent run, we obtain the sample average error

$$\bar{C} = \frac{1}{N} \sum_{i=1}^N C_i$$

and the associated standard deviation

$$\sigma_c = \sqrt{\frac{1}{N^2} \sum_{i=1}^N (C_i - \bar{C})^2}.$$

In order to compare the different algorithms we use the comparison technique used by Bar-Shalom and Li (1993) which converts the comparison process into a hypothesis testing problem:

$$H_0 : A = J^{(2)} - J^{(1)} \leq 0 \quad (\text{algorithm 1 not better than 2})$$

versus

$$H_1 : A = J^{(2)} - J^{(1)} > 0 \quad (\text{algorithm 1 better than 2})$$

subject to

$$\Pr\{\text{accept } H_1 | H_0 \text{ true}\} = \alpha \quad (\text{level of significance of hypothesis } H_1)$$

where

$$J^{(j)} = E[C^{(j)}] \quad j = 1, 2.$$

The decision whether to accept H_1 is made based upon the sample performance difference

$$\Delta_i = C_i^{(2)} - C_i^{(1)}.$$

Here, A , is independent of $\Delta_k, \forall k \neq i$.

⁵ Or as put by Bar-Shalom and Li (1993, page 74) "Those who can do. Those who cannot, simulate."

Then H_1 is accepted if

$$P = \frac{\bar{\Delta}}{\sigma_{\bar{\Delta}}} > P_0$$

where

$$\bar{\Delta} = \frac{1}{N} \sum_{i=1}^N \Delta_i \quad \text{Equation 2.2.1-61}$$

and

$$\sigma_{\bar{\Delta}} = \sqrt{\frac{1}{N^2} \sum_{i=1}^N (\Delta_i - \bar{\Delta})^2} \quad \text{Equation 2.2.1-62}$$

are the sample mean of the differences and the standard error of this sample mean. Assuming the error in $\bar{\Delta}$ to be normal, the threshold μ_0 is based on the upper tail of the normal distribution: $\mu_0 = 2.64$ for $\alpha = 5\%$, $\mu_0 = 2.33$ for $\alpha = 1\%$, and so on.

Seven different algorithms were tested.⁶ These were:

1. Kalman filter for multiple estimates.
2. Pure averaging.
3. Nearest Neighbor algorithm (Bar-Shalom and Fortman 1987). Here, the reading closest to the expected is taken for updating the state of the system. All other readings are rejected.
4. A posteriori method of sensor fusion, using the Gaussian distribution for estimating the conditional probabilities. This is similar to the PDAF algorithm.
5. Similar to case 4, except that the distance measure for comparing two distributions was used to estimate the conditional probabilities.
6. A posteriori method of fusion using learning. In this case the Gaussian distribution was used to estimate the conditional probabilities.
7. Similar to case 6, except that the distance measure for comparing two distributions was used to estimate the conditional probabilities.

Test Cases

A number of different test cases were simulated. Each case consisted of 500 Monte Carlo runs. Each run consisted of 100 steps. The same sensor readings were used by all the algorithms. The absolute error between the fused estimate and the actual state for each step was taken as the metric to compare the algorithms. The cases included: Gaussian noise with and without perfect information and clutter; added non-Gaussian noise, with zero mean to the sensor readings and randomized bias; and non-Gaussian noise added to one of the sensors with a non-zero mean.

Gaussian Noise

In these experiments, the sensor noise was Gaussian. Four different cases with different process and sensor noise covariance were simulated. In these cases, the fusion algorithms had perfect information about the noise characteristics. In the next two cases, the algorithms had imperfect and information about the noise characteristics. The cases were:

1. Equal uncertainty in process and sensor. $Q=1$, $R=[1, 1]$.
2. More accurate process model. $Q=0.25$, $R=[1, 13]$.
3. Less accurate process model. $Q=4$, $R=[1, 1]$.

⁶ For Algorithm 6 and Algorithm 7, a moving window average of the previous node probabilities was taken for $m_j(k+1|k)$. This is necessary since otherwise the a priori probability could be zero if the reading lay outside the validation region.

4. Very accurate sensors. $Q=1$, $R=[0.25, 0.25]$.

In the next two cases the algorithms had incorrect information about the process and the sensor noise characteristics. The process and the sensors were simulated using $Q=1$, $R=[1, 1]$.

5. The process was incorrectly modeled to be very accurate with $Q_{\text{modeled}}=0.25$, $R_{\text{modeled}}=[1, 1]$.

6. The sensors were incorrectly modeled to be very accurate with $Q_{\text{modeled}}=1$, $R_{\text{modeled}}=[0.25, 0.25]$.

Fig. 2.2.1.5.4-1 shows a typical run with the actual state and one of the sensor readings for Case 1 and Case 2. Fig. 2.2.1.5.4-2 shows a histogram for the mean absolute error at each step for the seven algorithms for the first four cases, while Fig. 2.2.1.5.4-3 shows the histogram of the absolute error for the fifth and the sixth cases. Fig. 2.2.1.5.4-4 shows the normalized absolute error for each of the seven algorithms for the first six cases. Here, the algorithm with the minimum error was taken as the base with a value of 100 and the errors from the other algorithms were correspondingly scaled. From these results one can conclude:

- As expected, when all assumptions of the Kalman filter (Gaussian noise, known noise characteristics) are satisfied (Cases 1-4) it is the optimal method for sensor fusion.
- The mean absolute error for the algorithms increases when the algorithms have wrong information about the process and sensor noise. This can be seen from the fact that the mean absolute error for the same process as in Case 1, increases from 0.4810 in Case 1 to 0.5648 in Case 5, and to 0.5165 in Case 6. It is interesting to note that for Case 5, when the algorithms thought (wrongly) that the process was very accurate, the mean absolute error by simple averaging (0.5621) was lower than that from the other algorithms. There is degradation in algorithm performance when they have wrong information about the noise characteristics.
- In the presence of perfect information the error was about 15-20% higher than the Kalman filter using the a posteriori methods of sensor fusion (Algorithms 4-7). This is to be expected since they are sub-optimal filters due to the assumption of only one sensor reading best representing the actual value.
- Algorithms 5 and 7, which compare distributions for the conditional probabilities had about 5% less error than algorithms 4 and 6, which use the Gaussian distribution for the conditional probabilities.
- There was no difference in performance between algorithms 4 and 6, and algorithms 5 and 7; i.e., no advantage of learning in this set of cases (Gaussian noise) since there was no change in behavior of the sensor noise over time.
- Since the performance of the algorithms deteriorates with incorrect information to the algorithms, adaptive algorithms which can estimate the noise characteristics are needed for cases where noise characteristics are poorly known or are time varying.

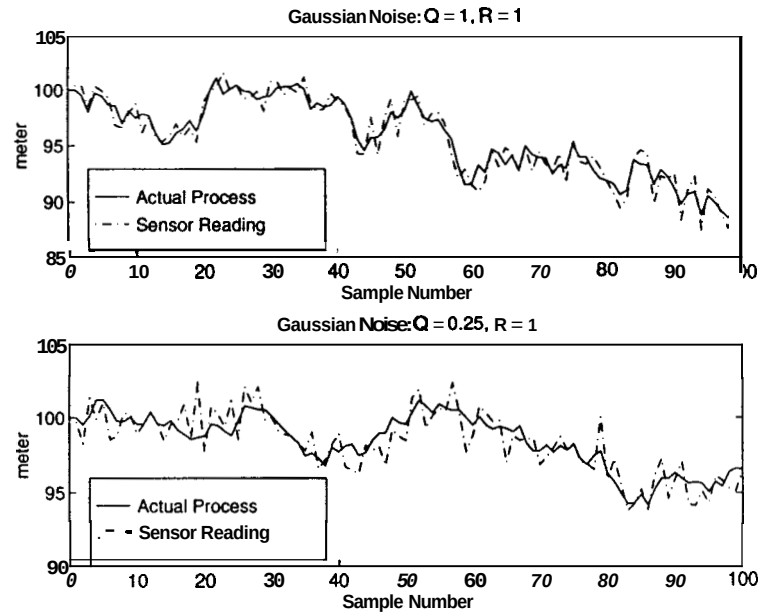


Fig. 2.2.1.5.4-1: Typical process and sensor noise reading for Cases 1 and 2

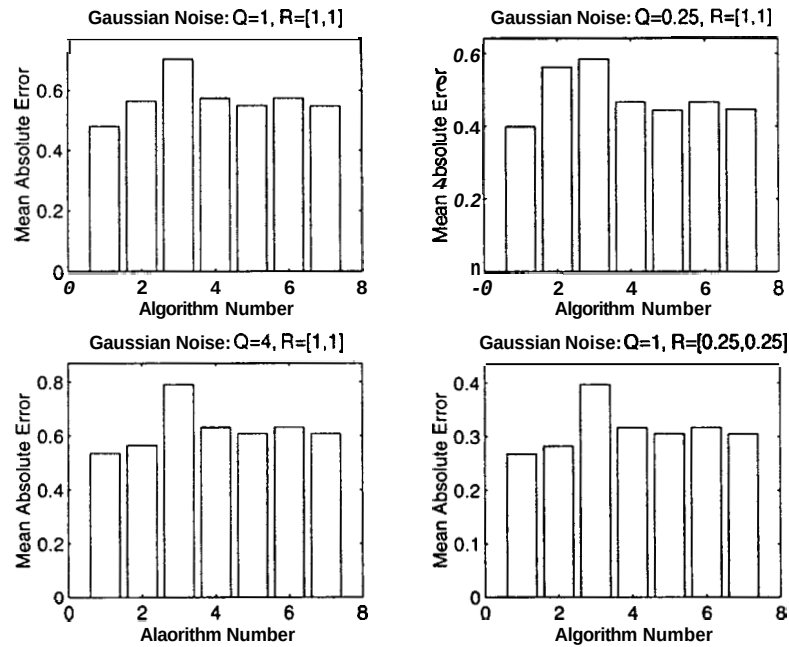


Fig. 2.2.1.5.4-2: Mean absolute error at each step for the seven algorithms for the first four cases, when the sensor noise was Gaussian and the algorithms had correct information about the noise characteristics

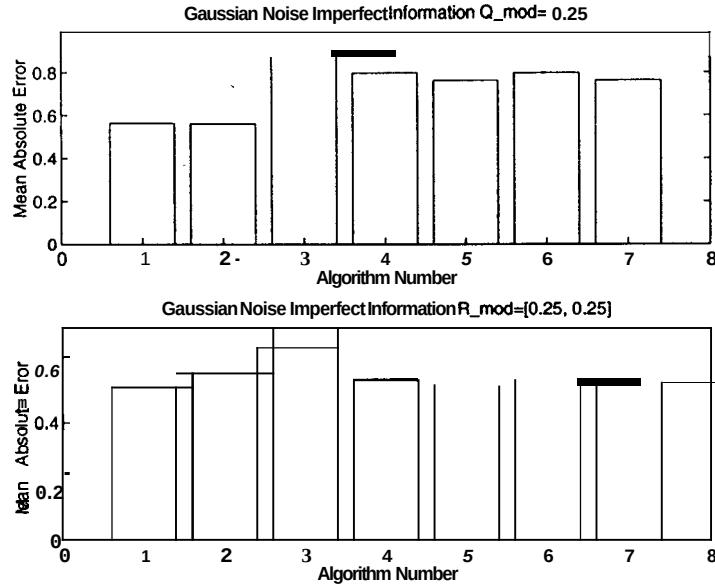


Fig. 2.2.1.5.4-3: Mean absolute error at each step for the seven algorithms for the fifth and sixth case. The sensor noise was Gaussian and the algorithms had wrong information about the noise characteristics

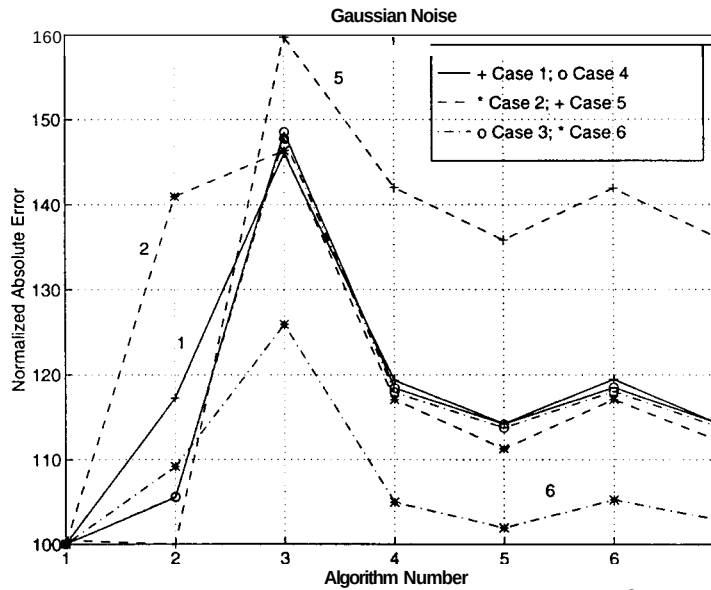


Fig. 2.2.1.5.4-4: Normalized absolute error (lowest error for each case was taken as 100) for the seven algorithms for the first six cases

Clutter

Clutter corresponds to spurious or aberrant readings. In this case clutter was added to the sensor readings. There are two random factors which determine clutter. First is the clutter density: the percentage of readings to which clutter was added and second, the clutter magnitude. Both the occurrence (dependent on the density) and magnitude of the aberrant reading were at random. To generate clutter a random number was generated to first determine if clutter is to be added to this reading. Next, if clutter was to be added, a random number was generated using a uniform distribution. The magnitude of the random number would then determine the magnitude of the

clutter to be added to the sensor noise. The expected value of the clutter was zero. The following cases were investigated:

7. No validation. Low clutter density = 0.25, Maximum magnitude = ± 5 .
8. Same as Case 7, but with validation.
9. No validation. High clutter density = 0.5, Maximum magnitude = ± 5 .
10. Same as Case 9, but with validation.
11. No validation. High clutter density = 0.5, Maximum magnitude = ± 10 .
12. Same as Case 11, but with validation.

In the next four cases the algorithms were provided with wrong information about the noise characteristics.

13. Same as Case 10, but algorithms were wrongly told that the sensors were very accurate
 $Q_{\text{modeled}} = 1, R_{\text{modeled}} = [0.25, 0.25]$.
14. Same as Case 10, but algorithms were wrongly told that the process was very accurate
 $Q_{\text{modeled}} = 0.25, R_{\text{modeled}} = [1, 1]$.
15. Same as Case 12, but algorithms were wrongly told that the process was very accurate
 $Q_{\text{modeled}} = 0.25, R_{\text{modeled}} = [1, 1]$.
16. Same as Case 12, but algorithms were wrongly told that the sensors were very accurate
 $Q_{\text{modeled}} = 1, R_{\text{modeled}} = [0.25, 0.25]$.

Fig. 2.2.1.5.4-5 shows a typical run with the actual process and one of the sensor readings for Cases 7 and 9. Fig. 2.2.1.5.4-6 shows the histograms for the mean absolute error for the seven algorithms for Cases 7 and 8. Similarly, Fig. 2.2.1.5.4-7 and Fig. 2.2.1.5.4-8 show the histograms for the mean absolute error for Cases 9 and 10, and Cases 11 and 12. Fig. 2.2.1.5.4-9 shows the histograms of the mean absolute error for Cases 13-16, where the algorithms had incorrect information about the noise characteristics. Fig. 2.2.1.5.4-10 shows the normalized mean error for all seven algorithms for Cases 7-16.

Based on the observations we can conclude the following:

- From Cases 7-12 we can infer that the suitability of carrying out validation is dependent on: the magnitude of the clutter and the clutter density. When the clutter magnitude was small, such that the reading lay within the validation region, there was an increase in the magnitude of the absolute error (Case 7-10). However, when the clutter magnitude is high (Cases 11 and 12) the utility of validation can be clearly seen (the error drops from 1.7602 to 1.3759 for the Kalman filter). Adding a validation gate can lead to the algorithm losing track, which happens when readings consistently fall outside the validation region. This is especially true when the algorithm has incorrect information about the noise characteristics, which leads to a faulty validation region.
- In the presence of clutter, a posteriori methods (algorithms 4-7) are superior to the rest. In Cases 14 and 15 when the algorithms were told that the process model was very accurate (hence (fortunately) the algorithm did not believe the sensor readings much) the Kalman filter had a lower error than the a posteriori methods. This was more due to the wrong information that the algorithms had than due to the nature of fusion carried out using the Kalman algorithm.
- Algorithms 5 and 7 which compare distributions to obtain the conditional probabilities had about 5-10% lower error than algorithms 4 and 6 which use the Gaussian distribution for the conditional probabilities.
- There was no difference in performance between algorithms 4 and 6, and algorithms 5 and 7, i.e., no advantage of learning in this case since there was no change in sensor noise behavior over time.

- Since the performance of the algorithms deteriorates with misinformation, adaptive algorithms which can estimate the noise characteristics are needed for cases where noise characteristics are poorly known or are time-varying.

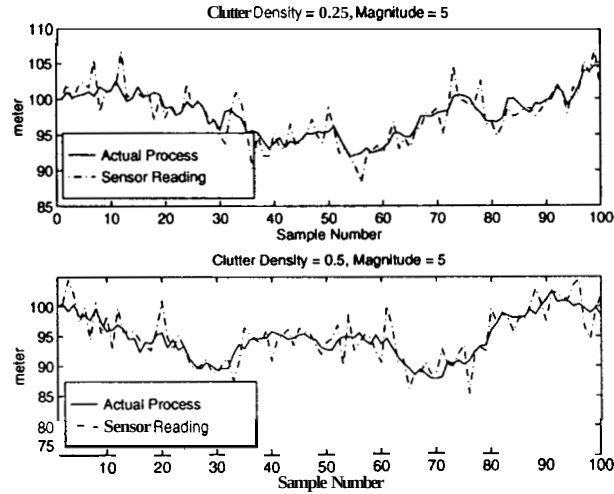


Fig. 2.2.1.5.4-5: Actual and sensor readings for a typical run for Cases 7 and 9

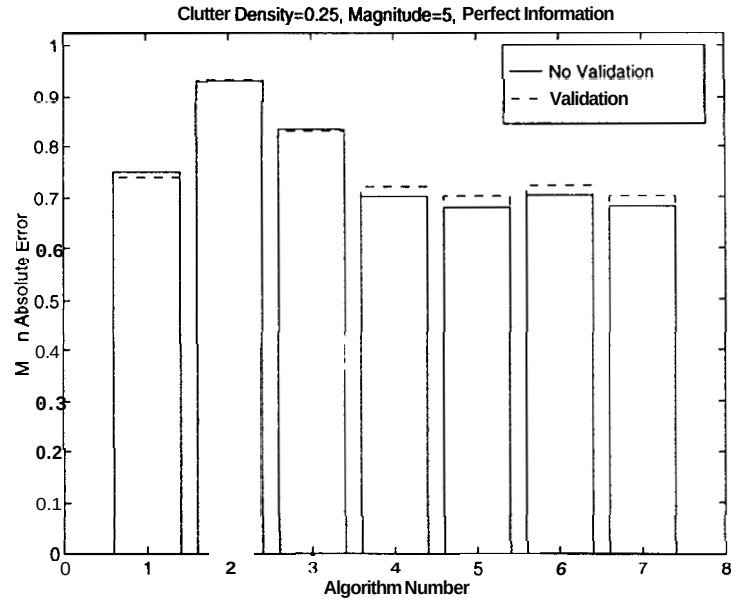


Fig. 2.2.1.5.4-6: Histogram of mean absolute error for the seven algorithms for Cases 7 and 8

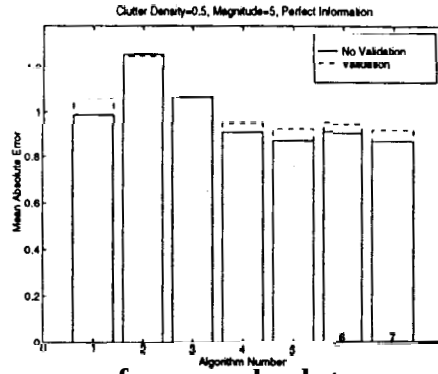


Fig. 2.2.1.5.4-7: Histogram of mean absolute error for Cases 9 and 10

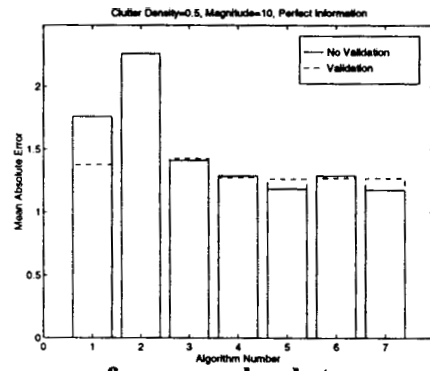


Fig. 2.2.1.5.4-8: Histogram of mean absolute error for Cases 11 and 12

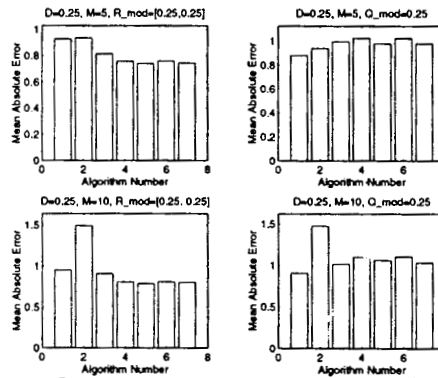


Fig. 2.2.1.5.4-9: Histogram of mean absolute error for the seven algorithms for Cases 13 to 16

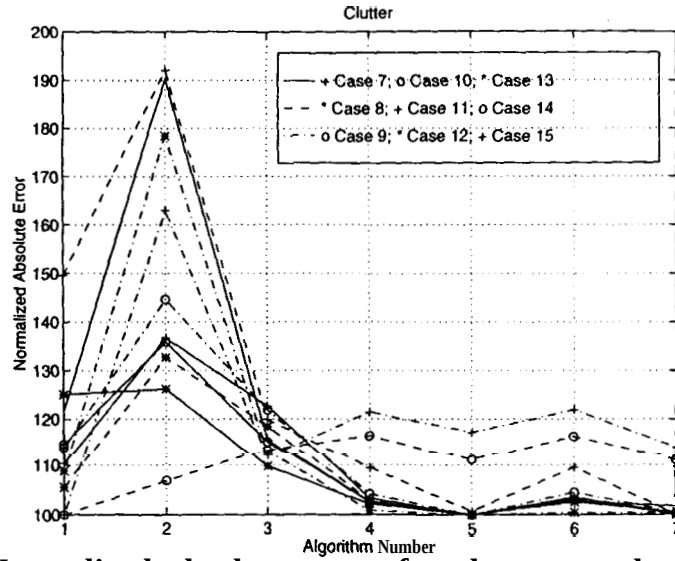


Fig. 2.2.1.5.4-10: Normalized absolute error for the seven algorithms for Cases 7-16 (case 16 straight line)

Randomized Bias

In this case randomized bias was added to one of the two sensors. A random number was generated from a uniform distribution for generating the randomized bias. The bias to be added to the sensor reading was then scaled using this random number magnitude. This differs from the clutter case in that the mean value of the added non-Gaussian noise is non-zero. The randomized bias was added to only the first sensor, the other sensor had only Gaussian noise. The following cases were investigated:

17. Bias magnitude 3 with validation.
18. Bias magnitude 5 with validation.
19. Bias magnitude 4 with validation, but sensors modeled to be very reliable $Q_{\text{modeled}} = 1$, $R_{\text{modeled}} = [0.25, 0.25]$.
20. Bias magnitude 4 with validation, but process modeled to be very reliable $Q_{\text{modeled}} = 0.25$, $R_{\text{modeled}} = [1, 1]$.
21. Bias magnitude 2 with validation, and clutter with density 0.35 and magnitude ± 5 .

Fig. 2.2.1.5.4-11 shows the actual process and the sensor noise for a typical run as in Cases 17 and 19. Fig. 2.2.1.5.4-12 shows the histogram for the mean absolute error for the seven algorithms for Cases 17 and 18. Similarly, Fig. 2.2.1.5.4-13 and Fig. 2.2.1.5.4-14 show the mean absolute error for Cases 19-21. Fig. 2.2.1.5.4-15 shows the normalized absolute error for the seven algorithms for Cases 17-21. Fig. 2.2.1.5.4-16 shows the normalized absolute error for algorithms 1, 5, 6, and 7 for all the 21 cases considered. From the results we can conclude:

- Algorithms 6 and 7 which learn the sensor behavior (that sensor 1 gives wrong estimates) over time perform the best. A posteriori methods (algorithms 4-7) perform better than the other algorithms.
- Algorithms 5 and 7, which compare two distributions for updating the conditional probabilities, were better than algorithms 4 and 6, which use the Gaussian distribution, by about 1-5%.
- The performance of the algorithms deteriorates when they have misinformation about the noise characteristics. This can be seen from Cases 19.

- Algorithm 7 works the best in the presence of non-Gaussian noise, while the Kalman filter is best when the noise is Gaussian and the noise characteristics are known.

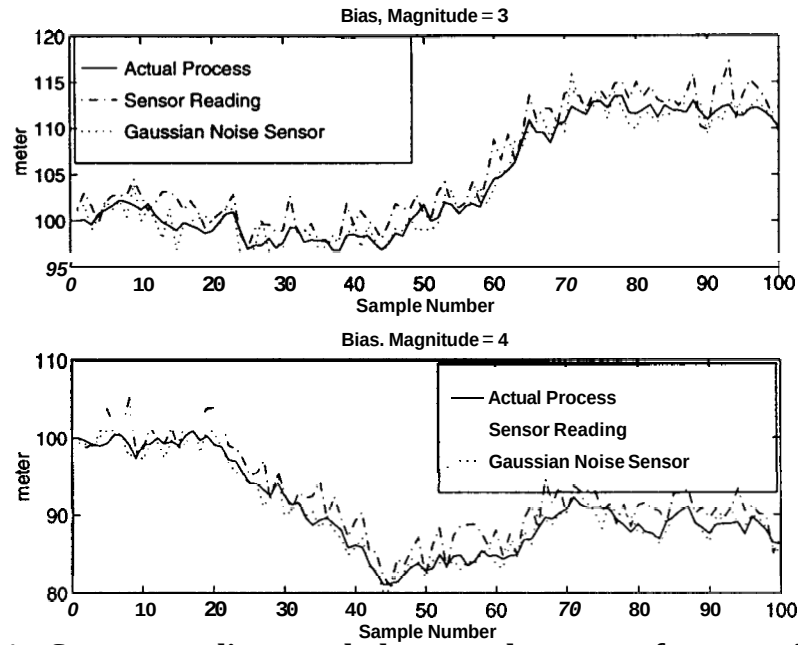


Fig. 2.2.1.5.4-11: Sensor readings and the actual process for a typical run for Cases 17 and 19

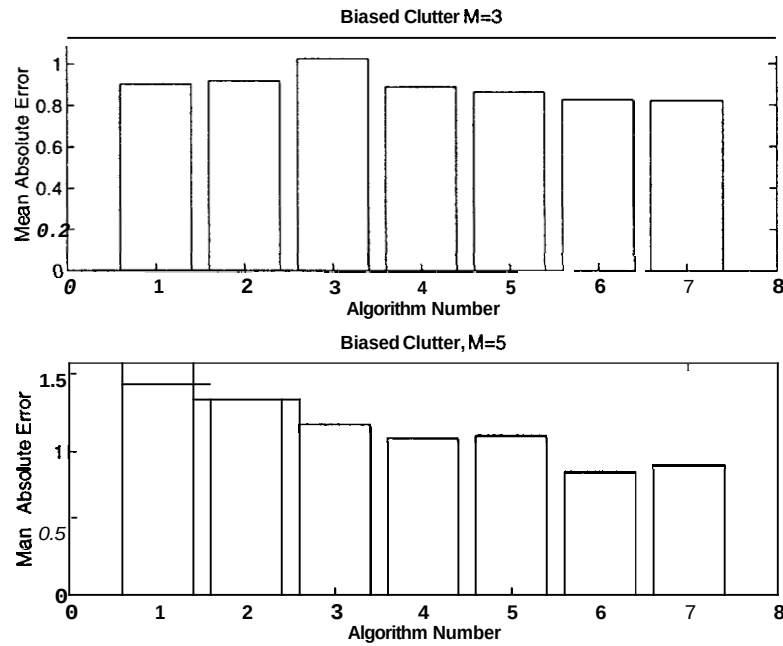


Fig. 2.2.1.5.4-12: Histogram of mean absolute error for the seven algorithms for Cases 17 and 18

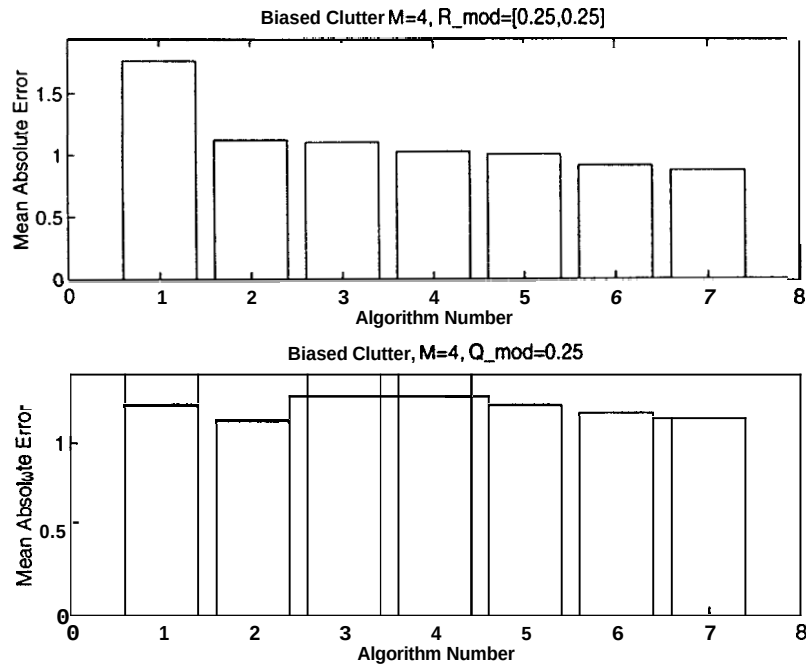


Fig. 2.2.1.5.4-13: Histogram of mean absolute error for the seven algorithms for Cases 19 and 20

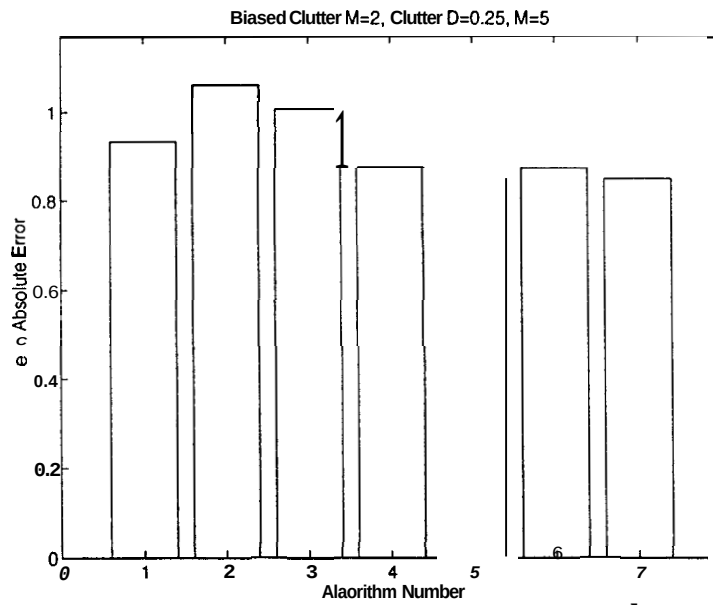


Fig. 2.2.1.5.4-14: Histogram of mean absolute error for the seven algorithms for Case 21

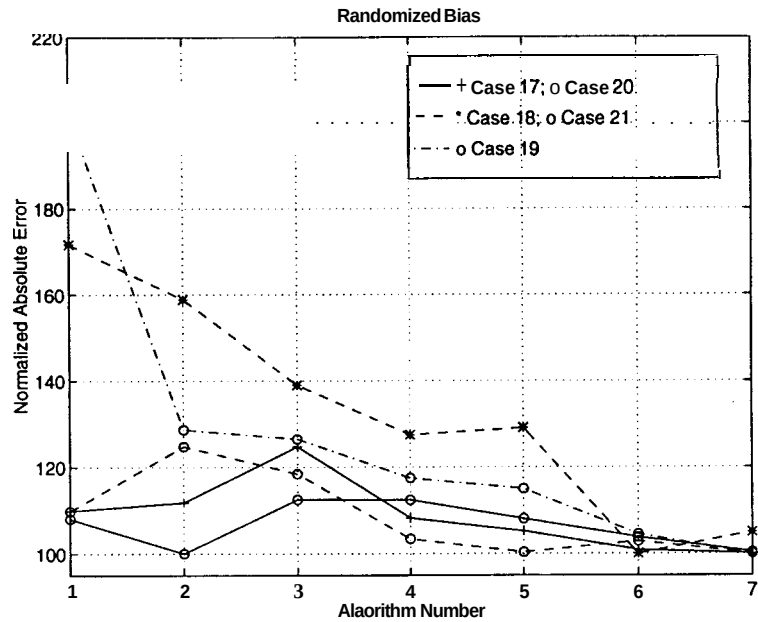


Fig. 2.2.1.5.4-15: Normalized absolute-error for the seven algorithms for Cases 17-21

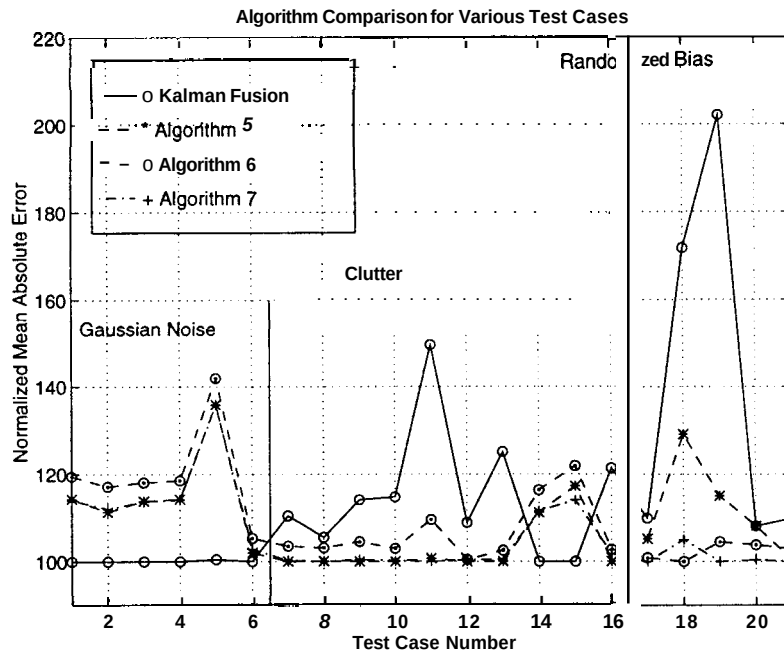


Fig. 2.2.1.5.4-16: Normalized absolute error for all 21 cases for Algorithms 1, 5, 6, and 7

2.2.2 Probabilistic Sensor Validation and Fusion Applied to IVHS

In this section, we apply concepts introduced in Section 2.2.1 to the problem of sensor validation and fusion for automated vehicles in IVHS by developing a framework for the supervisory control of automated vehicles. We introduce multiple state tracking models to be used for validation and fusion of readings from longitudinal sensors. These models correspond to the state transition model in our VDBN framework. We classify the vehicle states according to their tracking tasks. We investigate the efficacy of various algorithms for the tracking task by simulating platooning operations and applying them to real data obtained from normal platooning operations. For this we will use results from the Monte Carlo simulations carried out in the preceding section.

2.2.2.1 Methodology for Intelligent Sensor Validation and Fusion for the Longitudinal Sensors

For longitudinal control the automated vehicles in an IVHS require sensors to estimate the relative distance and velocity between vehicles. High data sensor fidelity is required to maintain the reliability and safety of the IVHS. Here we develop a methodology for validation and fusion of sensory readings obtained from the longitudinal sensors used in tracking automated vehicles.

For this purpose we will apply the Bayesian Decision-Theoretic framework that was introduced earlier. The VDBN shown in Fig. 2.2.2.1-1, corresponding to a Kalman filter, will be our basic inference engine. Additional uncertainty in the system will be modeled by means of discrete nodes.

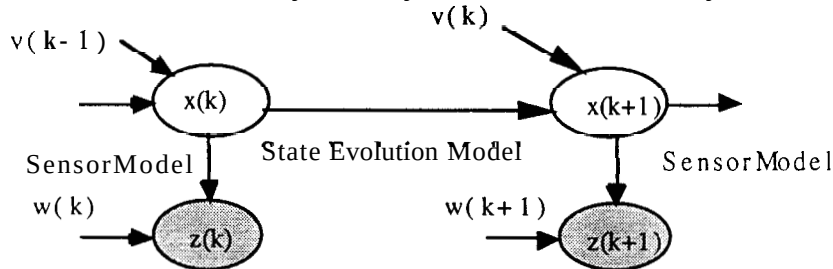


Fig. 2.2.2.1-1 : VDBN corresponding to a Kalman filter

In order to apply any of the algorithms developed in Section 2.2.1, we need a state evolution model for our system. We therefore first begin by developing the state evolution models that will be used for tracking purposes.

2.2.2.1.1 State Evolution Model

In the IVHS a string of closely spaced vehicles called platoons will travel under automatic control at high velocities. Depending on their position on the automated highway, the vehicles can be divided into two types: those in the platoon which are called follower vehicles and those that lead the platoon which are called leader vehicles. Those that travel alone are equivalent to a platoon of length one. The tracking tasks for these two vehicle states are different. It is therefore necessary to distinguish between these two cases in our validation and fusion methodology.

To deal with the various situations, we further divide the operating conditions into several states (see Fig. 2.2.2.1.1-1). There are three states for the follower vehicles: steady state, transient, and hazardous states. The first two are desired states while the third is undesired. A vehicle under automatic control is defined to be in the steady state when it is in a platoon and is trying to follow the one before it at a fixed known distance. Whenever the vehicle carries out a maneuver such as a split (leave the platoon), merge (join the platoon), or lane change it is defined to be in a transient state. This state involves relative acceleration between the vehicles and includes the state of the lead vehicle in the platoon. The last state, the hazardous state, is defined as the state when the vehicle carries out emergency maneuvers to avoid an accident. The lead vehicle, on the other hand, has only two states: a desired state and a hazardous state. If the lead vehicle's distance to the next

object are large enough (e.g., the distance to the next object is greater than the minimum for safe distance or the object is out of range), then the state is a desired state. If the distance to the next object is too small, we say it is a hazardous state. Other hazardous states exist beyond the one outlined above (Hitchcock 1993).

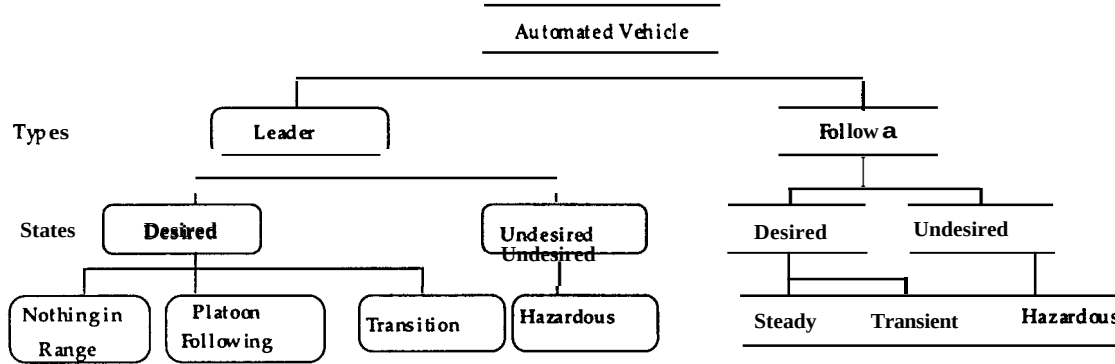


Fig. 2.2.2.1.1-1: Classification of states for the automated vehicle.

Three main state evolution models were developed. These include a one-state nearly constant distance, a two-state nearly constant velocity, and a three-state nearly constant acceleration model. The appropriate choice of the state evolution model would be dependent on the state of the vehicle.

For the purpose of developing these state evolution models consider a vehicle in the IVHS under automatic control moving in its lane as shown in Fig. 2.2.2.1.1-2. (*I*) is a fixed 1-dimensional inertial reference frame, while frame *i* is fixed to vehicle (*n-I*) and moves along with vehicle (*n-I*). Vehicle *n* is the target (either another vehicle or another arbitrary object) in its lane whose position, velocity and acceleration with respect to vehicle (*n-I*) we are trying to estimate, i.e.,

$$\Delta_x = X_n I - X_{(n-1)} I = (\ddot{x}_n - \ddot{x}_{(n-1)}) \tilde{i} = \ddot{\tilde{x}} \tilde{i}, \dot{x} \tilde{i}, x \tilde{i}$$

where

$\ddot{x}$ is the relative acceleration
 $\dot{x}$ is the relative velocity
 x is the relative position.

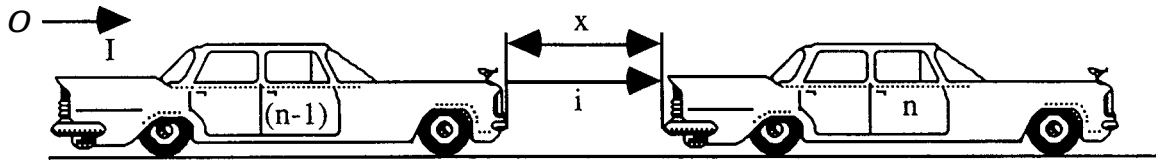


Fig. 2.2.2.1.1-2: Automated vehicle (*n-1*) behind an object in its lane.

2.2.2.1.1.1 One State Nearly Constant Distance Model

The one state nearly constant distance model is appropriate for the case when the vehicle is in the steady state. Steady state is defined as the state in which the automated vehicle in a platoon follows the one before it at a constant (zero) relative velocity, i.e. at $\dot{X} = 0$. However, in practice the velocity will undergo at least slight changes. These changes are modeled as a continuous time white noise $\tilde{v}$ as follows,

$$\dot{x}(t) = \tilde{v}(t)$$

$$E[\tilde{v}(t)] = 0$$

$$E[\tilde{v}(t)\tilde{v}(\tau)] = q(t)\delta(t - \tau)$$

where

$E[\tilde{v}(t)]$ is the expected value of $\tilde{v}(t)$

$q(t)$ is the covariance of $\tilde{v}(t)$.

The discrete time state equation is a Wiener process, i.e., a process driven by white noise

$$x(k+1) = x(k) + v(k)$$

$$Q = E[v(k)v^T(k)]$$

where

Q is the covariance of $v(k)$.

This one state Wiener model can also be used during the maneuvers.

Varaiya (1991) defines three kinds of maneuvering techniques for vehicles in IVHS: join, split, and lane change. Smooth trajectories have been designed for these procedures (Narendran 1994). These trajectories have been designed to keep the vehicle jerk (ride quality) and acceleration within acceptable limits. For example, for a lane change maneuver the trajectory is relatively simple as it requires a lateral position change of a fixed distance which is the distance between the centers of adjoining lanes and the final reference for trajectory design (i.e. center of target lane) is fixed. In general, for merge and split the desired spacing and desired relative velocities at the beginning and end (t_0 & t_f , respectively) are known. Using these four conditions a desired spacing profile can be generated which is of the form

$$sp_d(t) = c_0 + c_1t + c_2t^2 + c_3t^3$$

where

sp_d is the desired spacing profile

c_0, c_1, c_2, c_3 are constants.

Trajectories for various conditions (with and without platoon acceleration) have been developed by Narendran (1994), and will not be repeated here. The important point is that the desired relative distance and relative velocity of the vehicles during the maneuver are known and can be used for our sensor validation procedure. In this case, we model the residuals, i.e., the difference between the actual distance and the desired distance

$$r(t) = x(t) - x_d(t)$$

$$\dot{r}(t) = \dot{x}(t) - \dot{x}_d(t)$$

where

r is the residual

x_d is the desired distance.

As in the previous case, $\dot{r}(t)$ should ideally be zero. As before one can model the changes in $\dot{r}(t)$ as a continuous time white noise. In the absence of known trajectories, the multiple state models considered next can also be used.

2.2.2.1.1.2 Two State Nearly Constant Velocity Model

For the nearly constant velocity case we model the changes in the relative acceleration as a continuous time white noise $\tilde{v}$, $\ddot{x}(t) = \tilde{v}(t)$, where $\tilde{v}$ has the same properties regarding covariance and expected value as above. One can also define the process noise directly in discrete terms as

$$v(k) = \Gamma v(k)$$

where

$v(k)$ is a scalar valued zero-mean white noise sequence

$$\text{and } \Gamma = \begin{bmatrix} \frac{T^2}{2} & T \end{bmatrix}^T.$$

The two state evolution model is given by,

$$\mathbf{x}(k+1) = \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}} \end{bmatrix} + \Gamma \sigma_v$$

$$E[v(k)v(j)] = \sigma_v^2 \delta_{kj}$$

$$\mathbf{Q} = \Gamma \sigma_v^2 \Gamma^T = \begin{bmatrix} \frac{T^4}{4} & \frac{T^3}{2} \\ \frac{T^3}{2} & T^2 \end{bmatrix} \sigma_v^2$$

where

σ_v is the standard deviation

$\mathbf{Q}$ is the covariance of the process noise.

Here, the implicit assumption is that the relative acceleration between the vehicle and the target undergoes constant acceleration $\mathbf{v}(k)$ during sampling period k and that these accelerations are independent from period to period. Therefore, for one sampling period the change in relative velocity is $\tilde{\mathbf{v}}(t)T$ and the change of the relative distance is $\tilde{\mathbf{v}}(t)T^2/2$. The main difference between the continuous and discrete cases is that in the discrete case the assumption is piecewise constant white noise, while for the other case the assumption is continuous time white noise (Bar-Shalom and Li 1993).

2.2.2.1.1.3 Three State Nearly Constant Acceleration Model

This is for the case when constant relative acceleration between the two vehicles is nearly zero: $\ddot{\mathbf{x}}(t) = 0$. This model is most appropriate in case of emergency braking or maneuvering when no information about the maneuver is used.

In practice the acceleration will never be exactly constant. We model its changes by means of a zero-mean white noise as follows $\ddot{\mathbf{x}}(t) = \tilde{\mathbf{v}}(t)$. The smaller the variance of the noise the more constant is the acceleration. In this case the state vector and the continuous time state equation are:

$$\dot{\mathbf{x}}(t) = \mathbf{A} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} (t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tilde{\mathbf{v}}(t).$$

Similarly, the discrete time state equation with sampling time T is

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{v}(k)$$

$$\mathbf{F} = e^{\mathbf{A}T} = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{Q} = E[\mathbf{v}(k)\mathbf{v}'(k)] = \begin{bmatrix} T^5/20 & T^4/8 & T^3/6 \\ T^4/8 & T^3/3 & T^2/2 \\ T^3/6 & T^2/2 & T \end{bmatrix} q$$

where

$\mathbf{Q}$ is the covariance matrix for the process noise discretized from continuous time. As in other cases we can directly define the process noise in the discrete case. Thus, $\mathbf{v}(k) = \Gamma \mathbf{v}(k)$ where $E[\mathbf{v}(k)\mathbf{v}'(k)] = \sigma_v^2 \delta_{kj}$ and $\Gamma = [T^2/2 \quad T \quad 1]$. In this case, we assume that the acceleration is a discrete time Wiener process, i.e., non-stationary random process with mean zero and variance at . Furthermore, $\mathbf{v}(k)$ is the acceleration increment in a sampling time ("jerk"). The process noise covariance is

$$\mathbf{Q} = \Gamma \sigma_v^2 \Gamma^T = \begin{bmatrix} T^4/4 & T^3/2 & T^3/2 \\ T^3/2 & T^2 & T \\ T^2/2 & T & 1 \end{bmatrix} \sigma_v^2.$$

An estimate for the covariance matrix for the sensor noise $\mathbf{R}$ can be obtained by testing the sensors under test conditions and for various distances, i.e., obtaining accuracy versus sensor range. As we will see it can also be estimated by adding a discrete node to the VDBN in Fig. 2.2.2.1-1 (refer to Section 2.2.1).

The covariance for the process noise has one unknown which needs to be estimated. As the various operating states of the IVHS become more standardized the unknown parameter can be estimated. As a guideline, the changes in the velocity over a sampling period T are of the order of $\sqrt{Q_{22}} = qT$ for the constant velocity model. For the constant acceleration model, the changes in the acceleration over a sampling interval T are of the order of $\sqrt{Q_{33}} = \sqrt{qT}$. A particular range for choosing σ_v is $0.5\Delta a_M \leq \sigma_v \leq Aa$, where Aa is the maximum relative acceleration increment over a sampling period (Bar-Shalom and Li 1993).

2.2.2.1.2 Lead Vehicle

The lead vehicle in a platoon sets the velocity and the acceleration for the other vehicles in the platoon. This is determined by the conditions on the highway. A number of different scenarios are possible; for example there may be no target in the operating range of the longitudinal sensor, or the target may be a part of another platoon or the target could be a stationary target (e.g., a stalled car in the same lane), etc. The lead vehicle is more prone to misinterpreting longitudinal sensor data than the follower vehicle. The short distance range sensor readings will in most cases be out of range because the intraplatoon distance is too large. False readings can occur when the signal is reflected by roadside objects in which case it could be mistaken as an object on the lane. To tackle the problems of misinterpretation it is especially important to carry out the process of sensor data validation and data association as explained in Section 2.2.1.

2.2.2.2 Tracking Performance of Various Algorithms for Platoon Follower Case

In this section we will consider the suitability of the various algorithms developed in Section 2.2.1, for the case of a follower vehicle, when the lead vehicle goes through an acceleration and a deceleration profile. For more details on the vehicle model that is used for the simulation readers are referred to McMahan et al. 1992; Garg 1995; and Hedrick et al. 1995.

2.2.2.2.1 Simulation

Fig. 2.2.2.2.1-1 shows an acceleration and velocity profile for the lead vehicle during the simulation process. Throughout the simulations the desired spacing distance between the vehicles is four meters. The effectiveness of the simulation study greatly depends on how reliably the longitudinal sensors are modeled. For this purpose various tests, both under static and dynamic conditions were carried out to characterize the various longitudinal sensors (Agogino et al. 1995). These tests were carried out to model the sensors used in the simulation. Fig. 2.2.2.2.1-2 shows the spacing error during the maneuver for the first vehicle in the platoon, when there was no noise in the sensors and when the radar sensor was used to obtain the longitudinal distance. As can be seen clearly the addition of noise leads to poor tracking performance. The sum squared error without noise was **0.66926**, while in the presence of noise was **196.5826**. We will be using this sum squared error over the thirty second acceleration-deceleration profile as a metric for comparing the various algorithms developed in Section 2.2.1. Noise also effects the actuator response. A higher degree of control activity is required in the presence of sensor noise. This higher degree of control activity has a negative effect on passenger comfort. The effects of noise (poor tracking, higher control activity and poor ride quality) can be clearly seen from Fig. 2.2.2.2.1-2. Therefore, it is very important to develop an effective methodology for estimating the longitudinal distance, which obviates the negative effects of sensor noise and sensor failure. Fig. 2.2.2.2.1-3 shows the tracking performance for the first four vehicles in the platoon in the absence of any noise, i.e., with perfect sensor information to the controller.

In this section, we will systematically evaluate the relative performance of the different algorithms that were developed in Section 2.2.1. For this purpose we will compare the sum squared tracking error during the complete acceleration-deceleration profile.

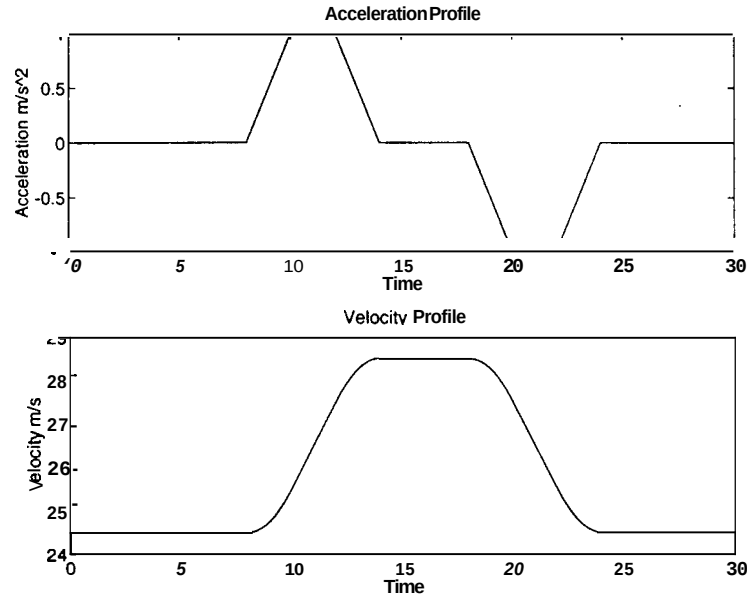


Fig. 2.2.2.2.1-1: Acceleration and velocity profile for the lead vehicle

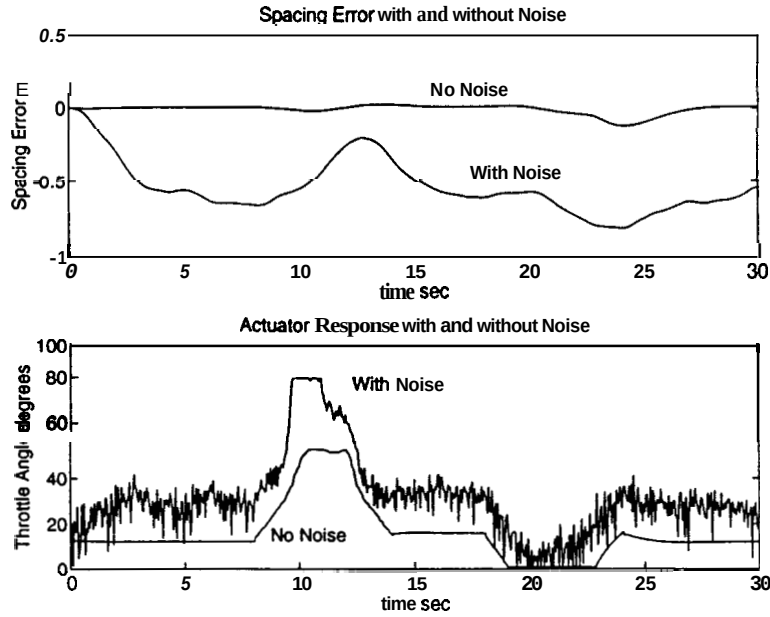


Fig. 2.2.2.2.1-2: Spacing error and actuator response for the first vehicle in the platoon with and without noise

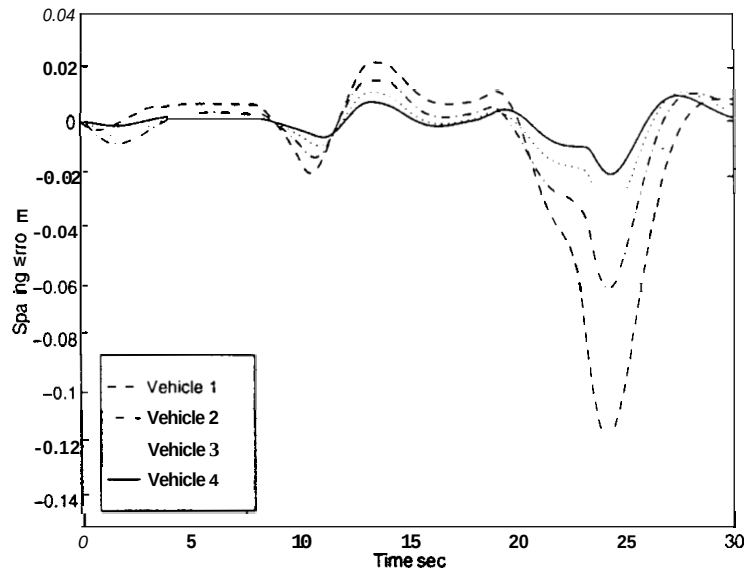


Fig. 2.2.2.2.1-3: Spacing error for the first four vehicles in the platoon. No sensor noise

2.2.2.2.1.1 Kalman Filter: VDBN with Continuous Nodes

We begin by using the VDBN network shown in Fig. 2.2.2.2.1.1-1. As explained in Section 2.2.1 this corresponds to the implementation of the Kalman filter. There are two parameters: the process noise covariance and the sensor noise covariance which need to be specified. These two parameters affect the overall performance of the algorithm, as their relative magnitude is used for generating the validation region and the relative weighting of the state evolution model with the incoming evidence. Readers are referred to Section 2.2.1 for details on the validation and fusion

process using this VDBN structure **and** Section 2.2.1 where the performance of the Kalman filter was investigated for different scenarios.

We briefly review the main steps of the algorithm. The VDBNs corresponding to the prediction, validation, and the inference stages are shown in Fig. 2.2.2.2.1.1-1.

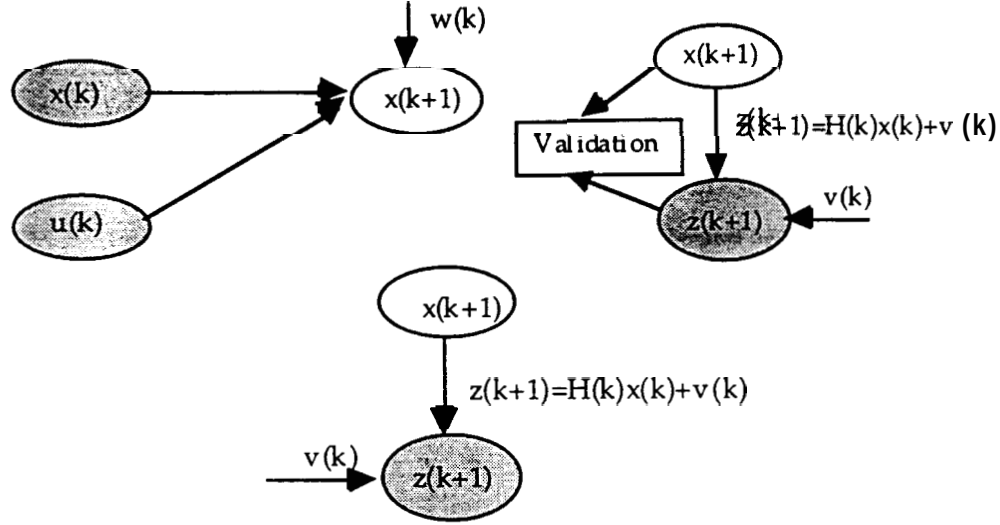


Fig. 2.2.2.2.1.1-1: VDBNs corresponding to the prediction, validation, and estimation process for the Kalman filter

Let our current estimate for the vector variables $x(k)$ after using all the evidence at time k be normally distributed with mean $\hat{x}(k|k)$, and variance $P(k|k)$, i.e., our knowledge about $x(k)$ is given by the following distribution $N(x(k), P(k|k), \hat{x}(k|k))$.

Prediction:

$$\begin{aligned}\pi(x(k+1)) &= N(x(k+1), P(k+1|k), \hat{x}(k+1|k)) \\ &= N(x(k+1); F(k)P(k|k)F(k)^T + Q, F(k)\hat{x}(k|k) + G(k)u(k))\end{aligned}$$

$$\lambda(x(k+1)) = 1$$

$$Bel(x(k+1)) = \pi(x(k+1))$$

Validation:

$$N(z_i(k+1); S_i(k+1), \hat{z}_i(k+1|k)) = N(z_i(k+1); H_i(k+1)P(k+1|k)H_i(k+1)^T + R_i, H_i(k+1)\hat{x}(k+1|k))$$

$$v_i \triangleq (z_i(k+1) - H_i(k+1)\hat{x}(k+1|k))$$

$$\tilde{V}_{k+1}(\gamma) \triangleq \{z_i(k+1) : [z_i(k+1) - \hat{z}_i(k+1|k)]S^{-1}(k+1)[z_i(k+1) - \hat{z}_i(k+1|k)] \leq \gamma\}$$

Estimation:

$$\begin{aligned}
\pi(x) &= N(x(k+1), P_\pi, \bar{x}_\pi) = N(x(k+1), P(k+1|k), \hat{x}(k+1|k)) \\
\lambda(x) &= N(x(k+1), P_\lambda, \bar{x}_\lambda) = \\
&N\left(x(k+1), [H(k+1)^T R^{-1}(k+1)H(k+1)]^{-1}, [H(k+1)^T R^{-1}(k+1)H(k+1)]^{-1} [H(k+1)^T R^{-1}z(k+1)]\right) \\
Bel(x) &= N(x(k+1), P(k+1|k+1), \hat{x}(k+1|k+1)) \\
&= N\left(x(k+1); [P^{-1}(k+1|k) + H(k+1)^T R^{-1}(k+1)H(k+1)]^{-1}, \right. \\
&\quad \left. P^{-1}(k+1|k+1)(P^{-1}(k+1|k)\hat{x}(k+1|k) + H(k+1)^T R^{-1}z(k+1))\right)
\end{aligned}$$

Since this is a case of platoon following, where the changes in the relative distance between vehicles in a sampling period is small, a one state nearly constant distance state evolution model is used, i.e.,

$$Distance(k+1) = Distance(k) + noise$$

$$x(k+1) = x(k) + w(k)$$

The covariance of the process noise $Q = E[w(k)w^T(k)] = q^2$, was varied keeping the sensor noise covariance fixed at $R = 0.16$. The sum squared spacing error over a range of modeled process noise is shown in Fig. 2.2.2.2.1.1-2. **As** can be seen there is a dramatic reduction in the sum of the squared error as compared to the case when no filtering is carried out. Tracking performance deteriorates whenever the modeled value for q deviates from its true value. In this case the process noise is due to the lead vehicle carrying out a non-uniform velocity. Similarly, Fig. 2.2.2.2.1.1-3 shows the sum squared error over a range of sensor noise covariance when the process noise covariance was kept at $Q = q^2 = .02^2$. Note that in both cases, nearly perfect tracking, i.e., tracking obtained in the presence of perfect information, can be obtained. The tracking performance is a function of how close the two modeled parameters are to their actual values (see also results from Monte Carlo test in Section 2.2.1). It is interesting to note the increase in the tracking error with the deviation of the filter gains from their optimal values. For most practical applications, the two parameters can be estimated off-line using a number of statistical methods and will remain close to their optimal values during implementation. However, a sudden change in the operating conditions, deterioration of the sensor, changes in the vehicle dynamics, etc. may change the characteristics of the process and the sensor noise. The algorithms provide feedback in the form of an estimate for process and sensor noise.

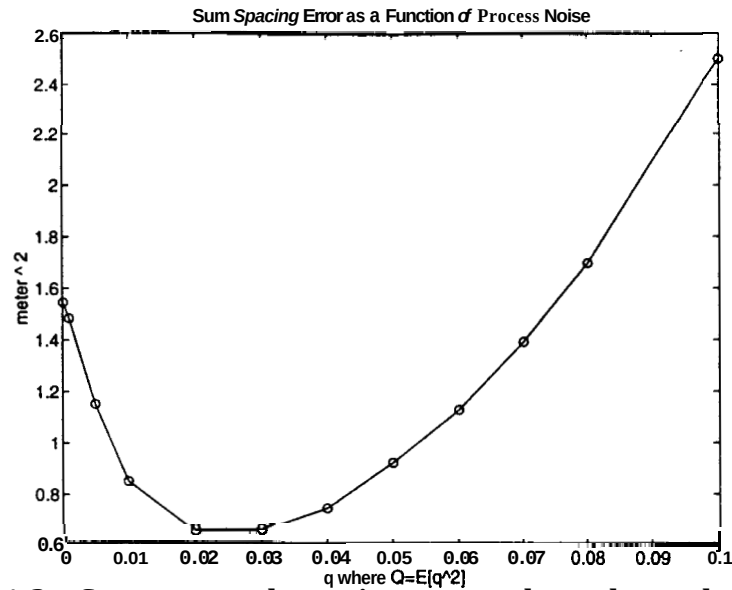


Fig. 2.2.2.2.1.1-2: Sum squared spacing error throughout the simulation for various process noise modeled in the VDBN

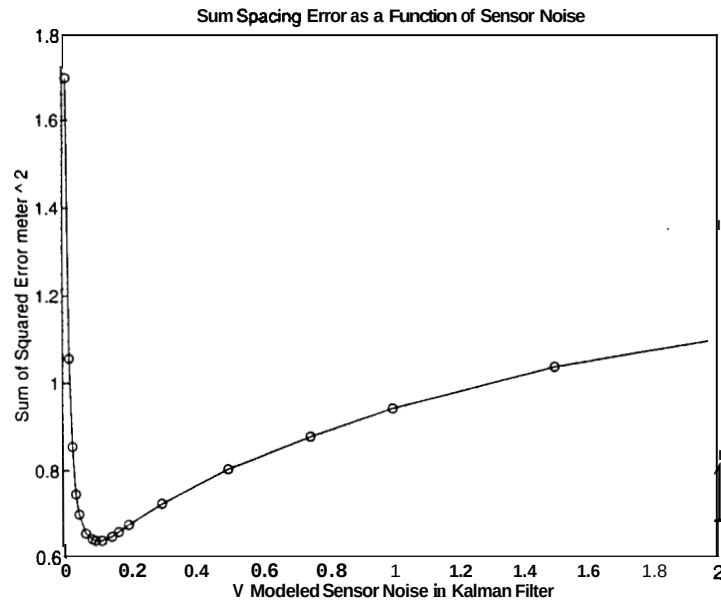


Fig. 2.2.2.2.1.1-3: Sum squared spacing error as a function of sensor noise modeled in the VDBN

2.2.2.2.1.2 Discrete Nodes in VDBN

The computationally least expensive adaptive estimation process uses the MDA (Multiple Discrete nodes Algorithm) for either the process or sensor noise. More computationally expensive Bayesian algorithms can in most cases distinguish between sensor and process noise, which the MDA cannot. This however is not a major constraint for the platooning follower case. Here, the deviation in the process (process noise) is negligible, due to which one can assume the process noise to be more or less constant and estimate the changes in sensor performance. Alternatively, in the case when there is a change in the process (emergency or maneuvering) the sensor noise covariance (assumption: sensor is not deteriorating during this instant) can be assumed to be more

or less constant. Of course both sensor and process noise can be estimated together, but this is more computationally expensive than just estimating sensor or process noise.

Fig. 2.2.2.2.1.2-1 shows the VDBN for the multiple model algorithm and for the algorithm to obtain feedback on the process noise. For the multiple model algorithm, the multiple models correspond to a different hypothesis in process noise, while all other parameters of the multiple models for this implementation were similar. The multiple model algorithm (generalized pseudo-Bayesian of first order) in this case simplifies to

Prediction

$$\Pr(M(k+1) = j|k) \stackrel{\Delta}{=} m_j(k+1|k) = \sum_i \Pr(M(k+1) = j|M(k) = i) \Pr(M(k) = i|k)$$

$$m_j(k+1|k) = [m_1(k|k), \dots, m_{n_m}(k|k)] \cdot \Phi_m(k)$$

$$\hat{x}_j(k+1|k) = F\hat{x}(k|k)$$

$$P_j(k+1|k) = FP(k|k)F^T + Q_j$$

$$\hat{x}(k+1|k) = F\hat{x}(k|k)$$

$$P(k+1|k) = FP(k|k)F^T + \sum_j m_j(k+1|k)Q_j$$

Validation

$$\Pr(z(k+1)|k) = N(z(k+1); S(k+1), \hat{z}(k+1|k))$$

$$= N(z(k+1); H(k+1)P(k+1|k)H(k+1)^T + R(k+1), H(k+1)\hat{x}(k+1|k))$$

$$v(k+1) = z(k+1) - \hat{z}(k+1|k)$$

Accept the reading if

$$v^T(k+1)S^{-1}(k+1)v(k+1) \leq \gamma$$

Estimation

$$\Pr(z(k+1)|k, M(k+1) = j) = N(z(k+1); S_j(k+1), \hat{z}(k+1|k, M(k+1) = j))$$

$$= N(z(k+1); H(k+1)P_j(k+1|k)H(k+1)^T + R(k+1), H(k+1)\hat{x}(k+1|k))$$

node $M(k+1)$

$$\Pi(M(k+1)) = [m_1(k+1|k), \dots, m_{n_m}(k+1|k)]$$

$$\lambda(M(k+1)) = [\pi_1^M, \dots, \pi_{n_m}^M]$$

$$Bel(M(k+1)) = \alpha [m_1(k+1|k)\pi_1^M, \dots, m_{n_m}(k+1|k)\pi_{n_m}^M]$$

Hence,

$$Bel(m(k+1|k+1) = i) = m_i(k+1|k+1) = \frac{\pi_i^M m_i(k+1|k)}{\sum_j \pi_j^M m_j(k+1|k)}.$$

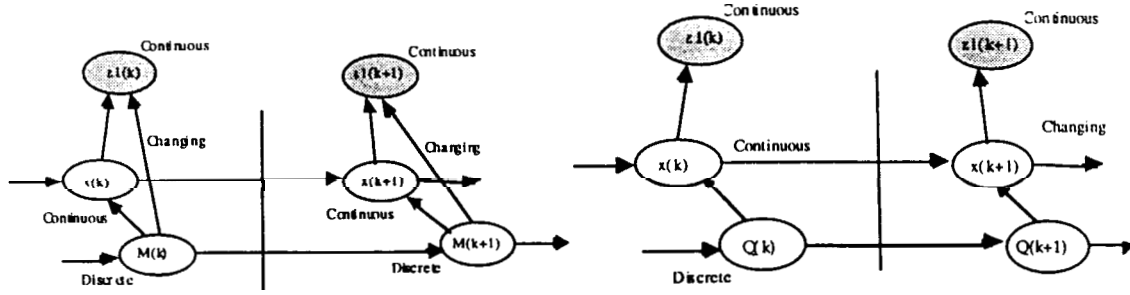


Fig. 2.2.2.1.2-1: Feedback on the process noise, through the addition of a discrete node. VDBNs corresponding to the multiple model algorithm and algorithm for estimating process noise

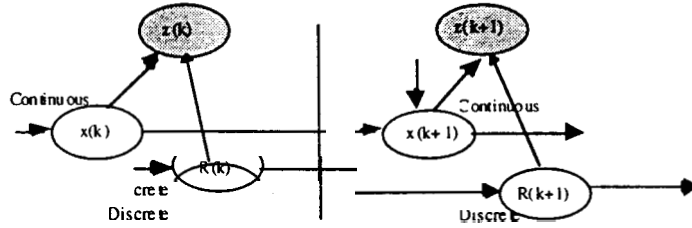


Fig. 2.2.2.1.2-2: VDBN for sensor noise feedback

Fig. 2.2.2.1.2-2 shows the VDBN for sensor noise feedback. Extensive testing of various adaptive algorithms was carried out in Section 2.2.1. Here, the mean of the prior probabilities over a sliding window was taken as the a priori probabilities. As a variant to that approach we investigated an alternative approach where the estimates for the noise characteristics were updated after a number of samples. An update of this kind however leads to non-uniform computational load. Fig. 2.2.2.1.2-3 shows the sum squared error when three clusters were used to obtain feedback on sensor noise. The three curves correspond to three ranges that the clusters modeled. The dash-dotted line corresponds to the hypothesis being at $[0.01 \ 0.1 \ 1.251]$, the dashed line corresponds to a smaller range $[0.05 \ .15 \ 1]$, and dotted line corresponds to the smallest range corresponding to $[0.1 \ 0.15 \ 0.21]$. As can be seen the sum squared error is a function of the number of samples, after which feedback was used to update the estimates, and the distance between the cluster centers were.

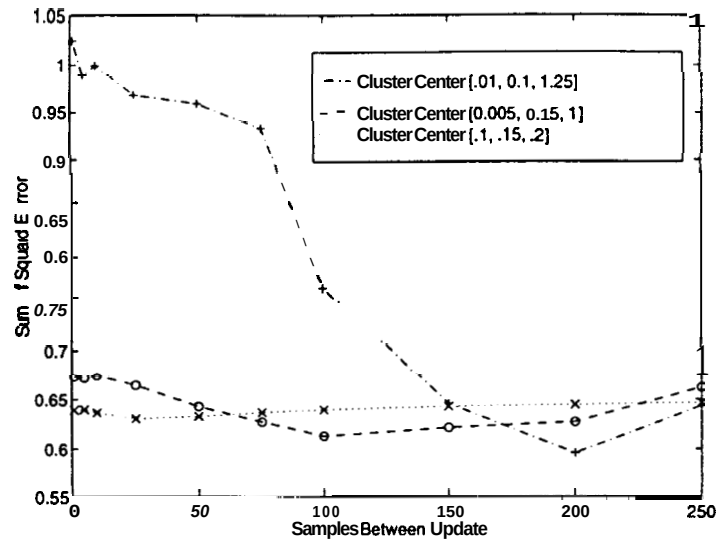


Fig. 2.2.2.2.1.2-3: Sum squared error using three different cluster centers to obtain feedback on the sensor noise

We investigated updating after a number of samples rather than taking a moving window average. Fig. 2.2.2.2.1.2-4 shows the sum of the squared error of the multiple model algorithm (dashed line), the algorithms in using λ message considering both the mean (dotted line) and distribution (dashed-dotted line) to estimate the process noise. Here, three Gaussian clusters were used to represent the standard deviation of the process noise of $q=[0.005, .025, 0.05]$. Note that the three hypotheses represented by this discrete node correspond to a very large range of possible process noise. The sum squared error is plotted versus the number of samples after which the probabilities were updated, i.e., after which feedback on the process noise was obtained. For this simulation case, all three algorithms seemed to perform equally well, which is not surprising because sensor noise was assumed to be constant.

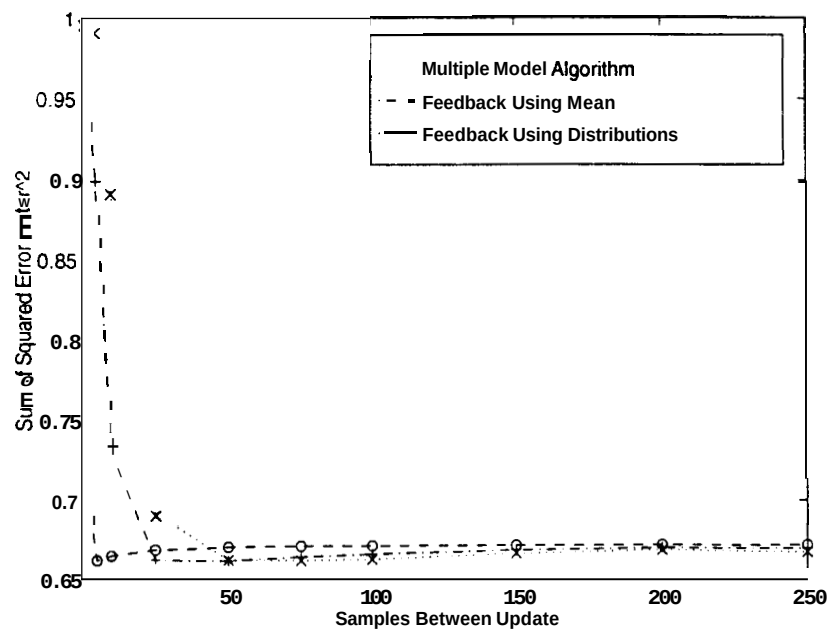


Fig. 2.2.2.2.1.2-4: Sum squared error for the simulation using a three state discrete node to estimate the process noise

Readers are referred to Section 2.2.1 which deals with using adaptive Bayesian algorithms to estimate process and sensor noise. One has to weigh the advantage gained through adaptation to the additional computation **task**. When operating conditions cause changes in the sensor (from that modeled) or the process (emergency conditions) it may be advisable to adapt for better tracking.

2.2.2.2.1.3 Multiple Estimates

In this section we will evaluate the results when readings from the radar and the sonar were used. The learning algorithm of Section 2.2.1 with a distance metric performed the best for clutter and randomized bias. The Kalman filter is the optimal filter when the noise is Gaussian with known characteristics.

To investigate the effects of incorrectly modeled noise characteristics we first consider the APDAF algorithm (which corresponds to the best filter of Section 2.2.1), whose VDBN is shown in Fig. 2.2.2.2.1.3-1. There are two parameters for the APDAF algorithm and the other Bayesian algorithms: the modeled process noise, and the modeled sensor noise. Fig. 2.2.2.2.1.3-2 shows the variation of the sum of squared error for various values of the process and sensor noise. As a general trend one can see the increase in the sum squared error with the deviation of these two parameters (rather their relative ratios) from their optimal values.

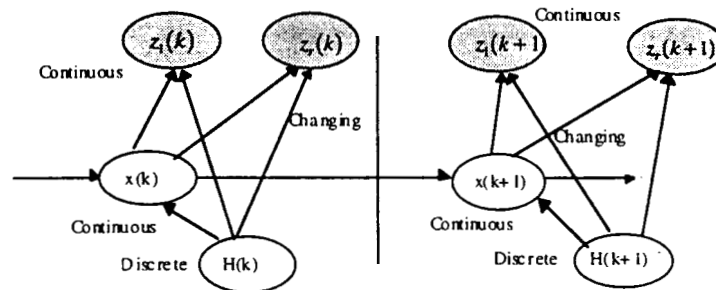


Fig. 2.2.2.2.1.3-1: VDBN corresponding to the APDAF algorithm

Three other fusion algorithms, presented in Section 2.2.1, were also implemented: (1) fusion using the Kalman filter (Fig. 2.2.2.2.1.3-3), (2) fusion using a posteriori weights (Fig. 2.2.2.2.1.3-4), and (3) fusion using a posteriori linear weights with learning (Fig. 2.2.2.2.1.3-5). The results with these algorithms were similar and the sum squared error for each of the algorithms was very close to the minimum, i.e., when there was no noise in the system.

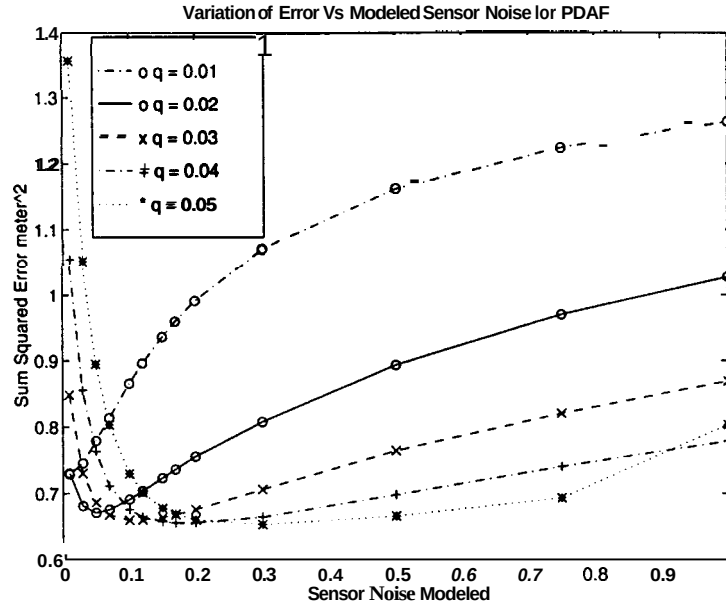


Fig. 2.2.2.2.1.3-2: Variation of sum squared error for the simulation as a function of the modeled process and sensor noise for the APDAF

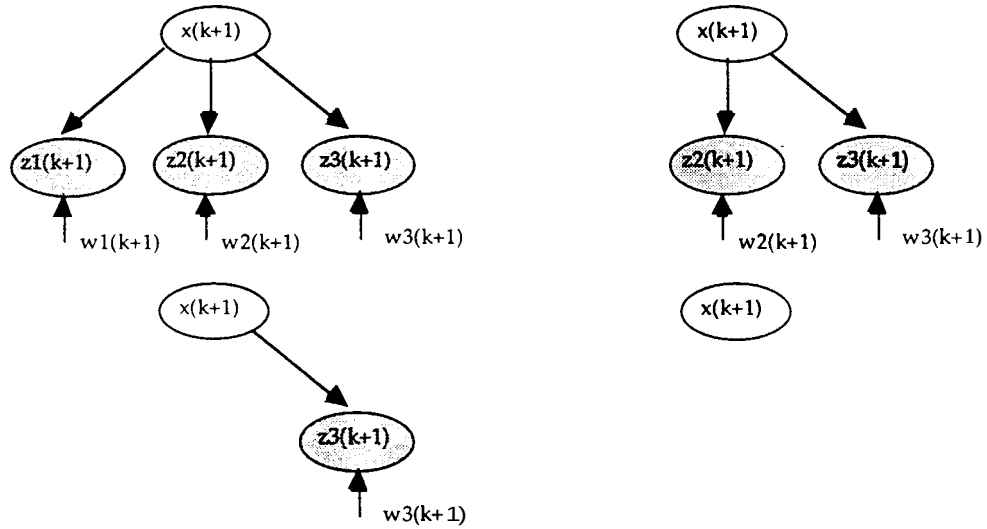


Fig. 2.2.2.2.1.3-3: Topology transformation corresponding to fusion using the Kalman filter

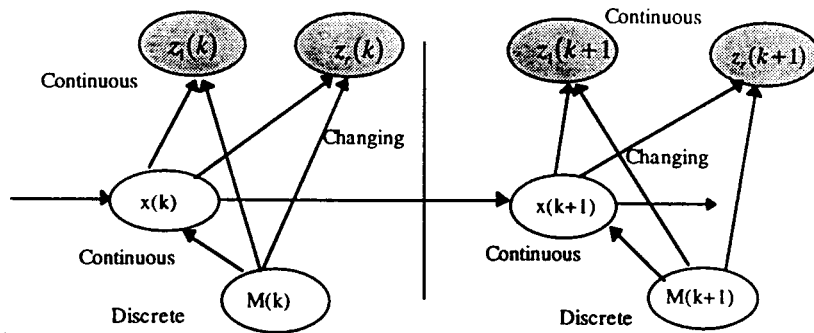


Fig. 2.2.2.2.1.3-4 : VDBN corresponding to fusion using a posteriori linear weighting

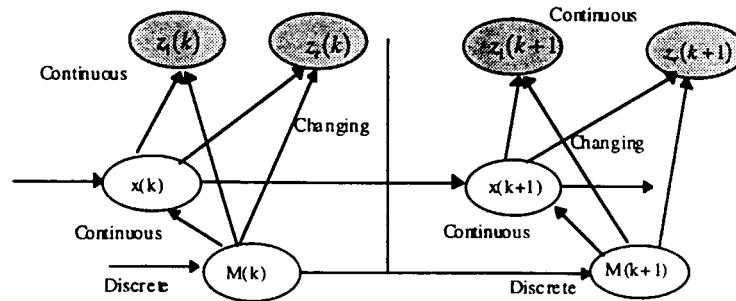


Fig. 2.2.2.2.1.3-5: VDBN corresponding to fusion using a posteriori information and learning

Fig. 2.2.2.2.1.3-6 shows the output from the radar and the sonar sensor during normal platooning operations that were carried out on the freeway. The true distance between the vehicles is about 4 meters. As can be clearly seen, there are a number of outliers in the sonar sensor and few in the radar sensor.

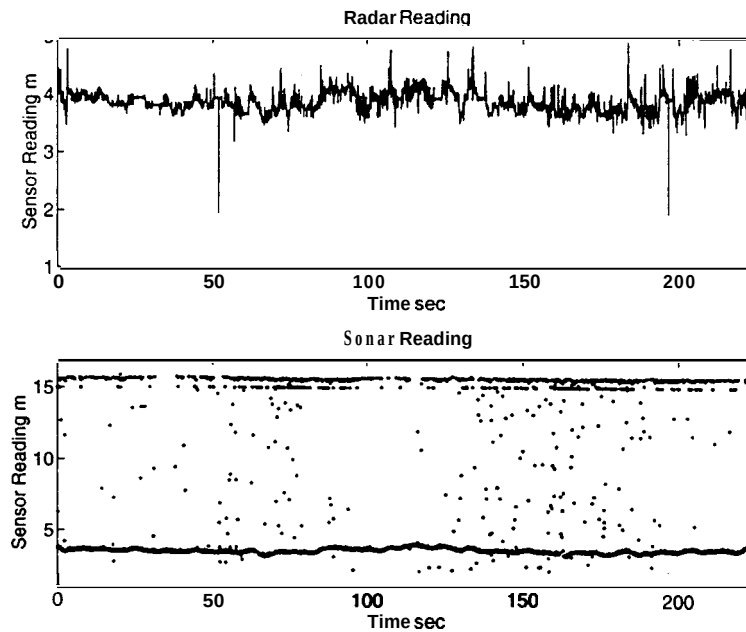


Fig. 2.2.2.2.1.3-6: Output from the radar and the sonar sensor during normal platooning operations carried out on the freeway. The true distance is about four meters. The outliers in the radar and the sonar sensor can be clearly seen

Fig. 2.2.2.2.1.3-7 shows the Kalman filtered estimate along with the radar sensor output during normal platooning operations carried out on the freeway. Fig. 2.2.2.2.1.3-8 shows the normalized innovation along with the validation gate during this filtering process.

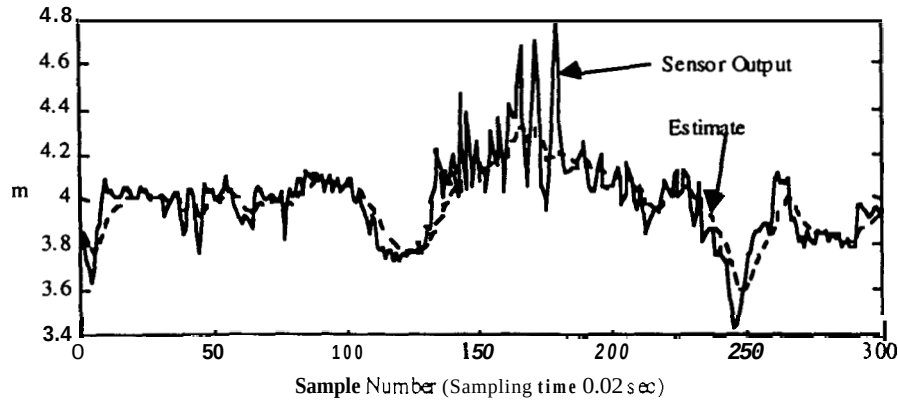


Fig. 2.2.2.2.1.3-7: Sensor output and the Kalman filter estimate during normal platooning operations on the freeway

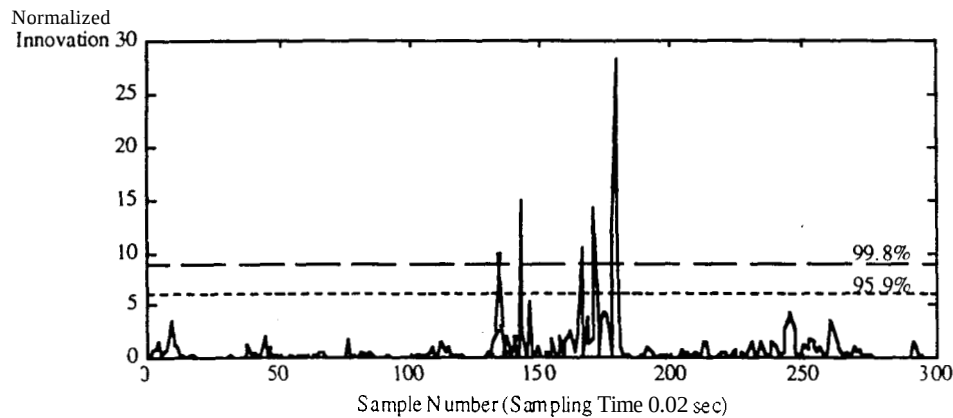


Fig. 2.2.2.2.1.3-8: Normalized Innovations used for sensor validation from the normal platooning operations on the freeway

2.2.2.3 Split and Merge Maneuver

In the previous section, we briefly considered how a one state transition model could be used for sensor data validation when an open-loop trajectory (Narendran 1994) for maneuvers was carried out. In this section, we will investigate the effect of sensor noise on carrying out the closed-loop maneuvers. For this purpose, we use the closed-loop trajectory designed by Frankel et al. (1994) and the control law developed by Connolly (1996).

The various algorithms investigated in the previous section were tried using one, two, and three state transition models. A simple one state transition model using Kalman filtering was adequate for removing sensor noise effects, which is illustrated by the following Figures. Fig. 2.2.2.3-1 shows the velocity of the i th vehicle during the join maneuver. Due to effect of sensor noise spacing error is created. There is a lag between the velocity of the vehicle when there is no noise and the velocity in the presence of noise. Fig. 2.2.2.3-2 shows the spacing error between two vehicles when one of them undergoes a maneuver. For the plot the desired spacing during the maneuver was taken as the final distance between the vehicles after the completion of the

maneuver. Fig. 2.2.2.3-3 shows the throttle angle during the join maneuver with and without noise, while Fig. 2.2.2.3-4 shows the throttle angle using a simple one state Kalman filter. As can be seen from Figs. 2.2.2.3-3 and 2.2.2.3-4, filtering removes throttle chatter and leads to a smoother ride. Similar, results can be seen for the split maneuver. Fig. 2.2.2.3-5 shows the velocity for the i th vehicle during the split maneuver. Fig. 2.2.2.3-6 shows the distance between the i th and the $(i-1)$ th vehicle during the maneuver. Fig. 2.2.2.3-7 shows the throttle angle during the split maneuver with and without sensor noise. Fig. 2.2.2.3-8 shows the throttle angle during the split maneuver using a one state Kalman filter.

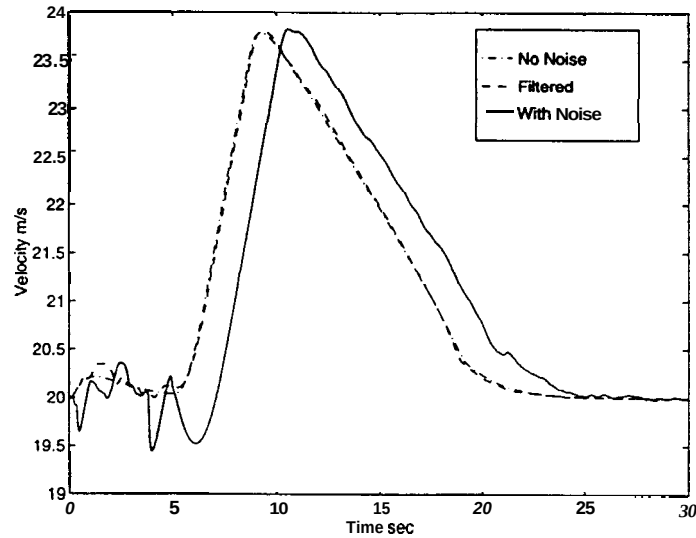


Fig. 2.2.2.3-1: Velocity of the i th vehicle during the join maneuver. A simple one state transition model for the Kalman filter was used

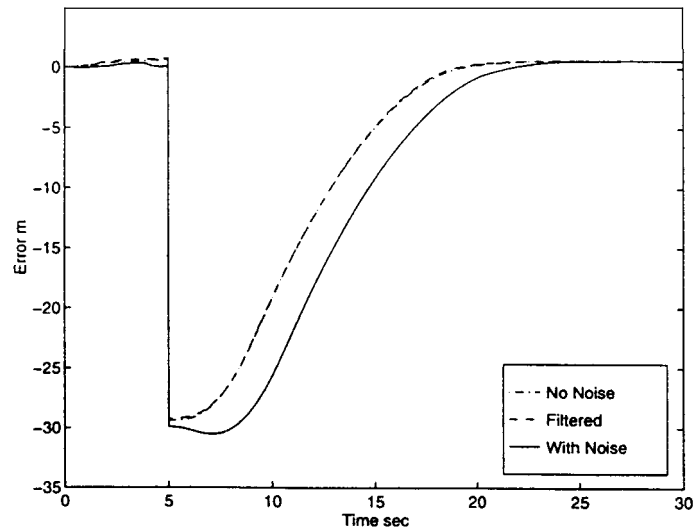


Fig. 2.2.2.3-2: Spacing error between the two vehicles. The desired spacing between the vehicles during the maneuver was taken as the final distance

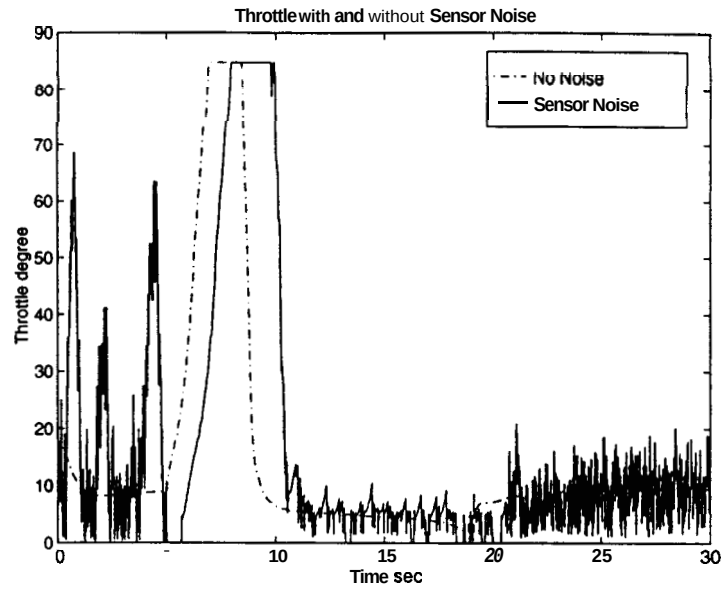


Fig. 2.2.2.3-3: Throttle angle during the join maneuver with and without sensor noise

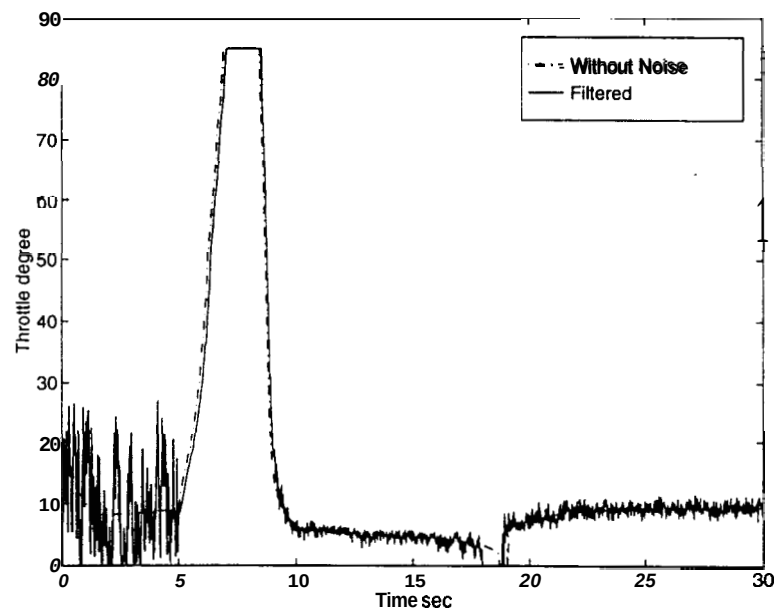


Fig. 2.2.2.3-4: Throttle angle during the join maneuver without noise and using a one state Kalman filter

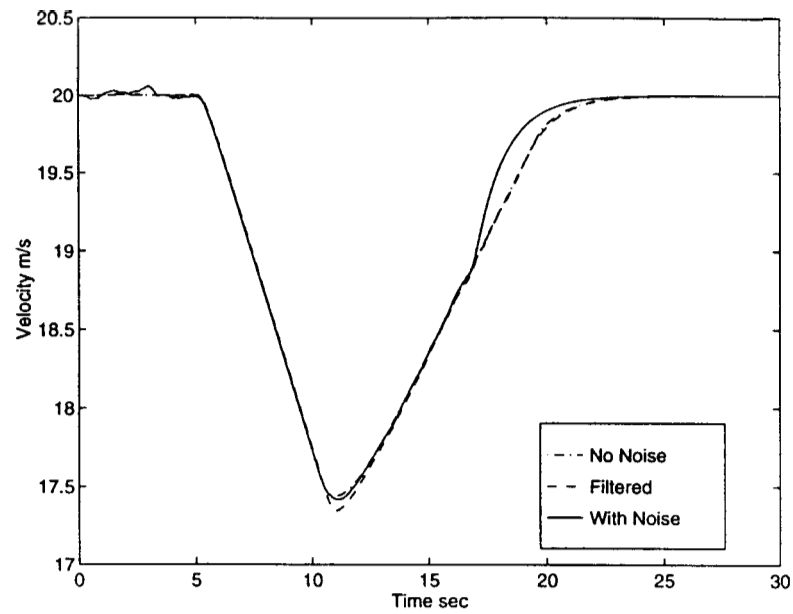


Fig. 2.2.2.3-5: Velocity of the i th vehicle during the split maneuver

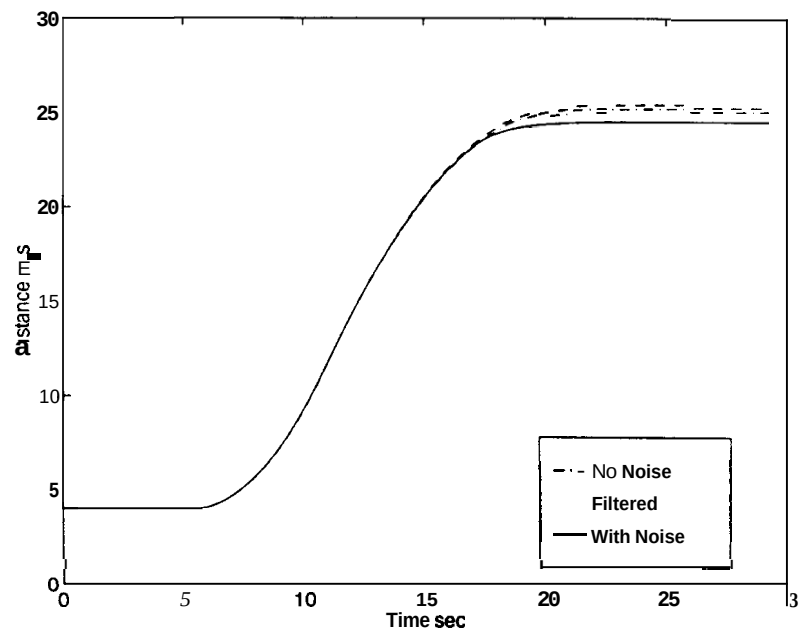


Fig. 2.2.2.3-6: Distance between the i th and the $(i-1)$ th vehicle during the split maneuver

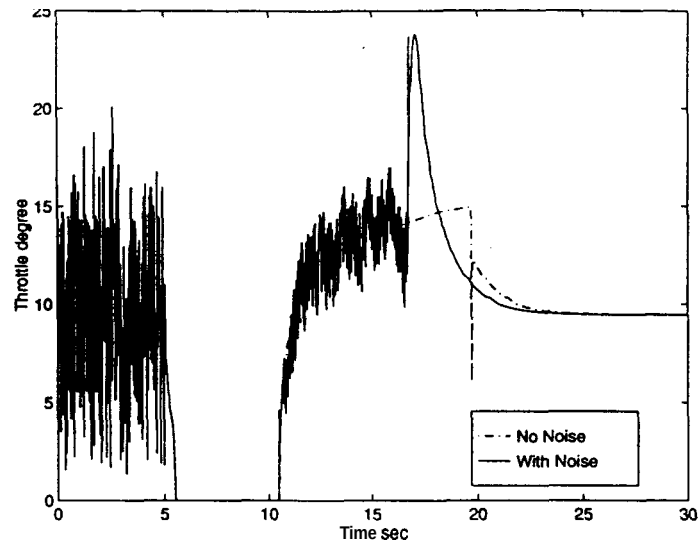


Fig. 2.2.2.3-7: Throttle angle during the split maneuver with and without noise

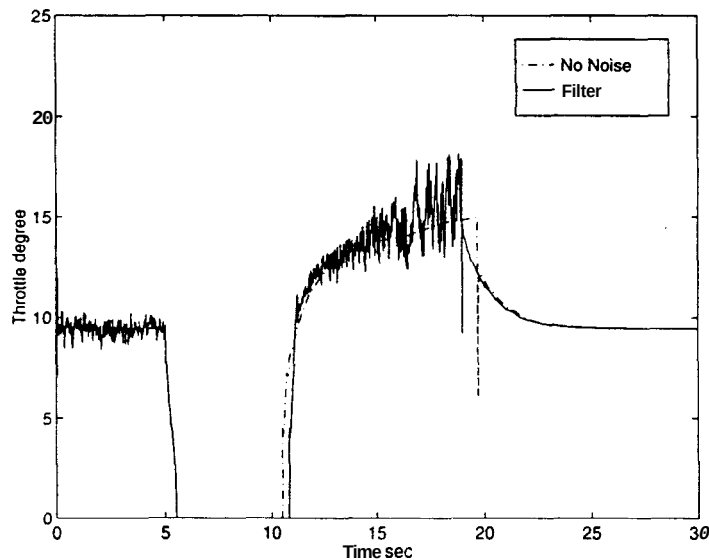


Fig. 2.2.2.3-8: Throttle angle during the split maneuver using a one state Kalman filter

2.2.2.4 Lead Vehicle

The lead vehicle in a platoon sets the velocity and acceleration for the other vehicles in the platoon. This is determined by highway conditions. A number of different scenarios are possible; for example there may be no target in the operating range of the longitudinal sensor, or the target may be a part of another platoon or the target could be a stationary target (e.g., a stalled car in the same lane), etc. The lead vehicle is more prone to misinterpret longitudinal sensor data than the follower vehicle. The lead vehicle differs from a follower vehicle in that it does not have access to a priori information to the advent of a vehicle in its sensors range, i.e., the sudden presence of a target in its sensors range. For this purpose the multiple model algorithms, namely the pseudo-Bayesian algorithm of the first order (Fig. 2.2.2.4-1) and the interacting multiple model (Fig. 2.2.2.4-2), are useful for modeling the multiple hypotheses, to determine a target is present or absent.

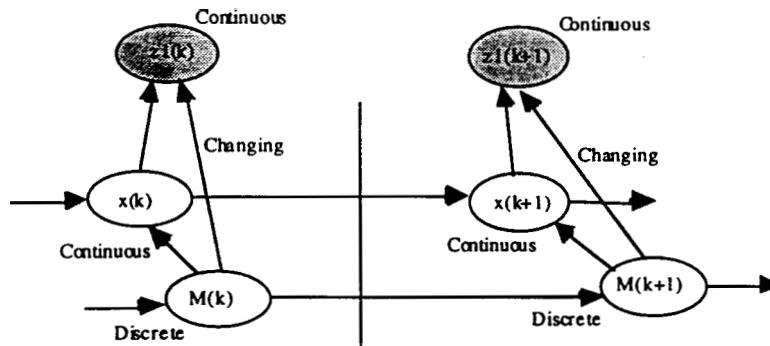


Fig. 2.2.2.4-1: VDBN corresponding to the pseudo-Bayesian algorithm of the first order

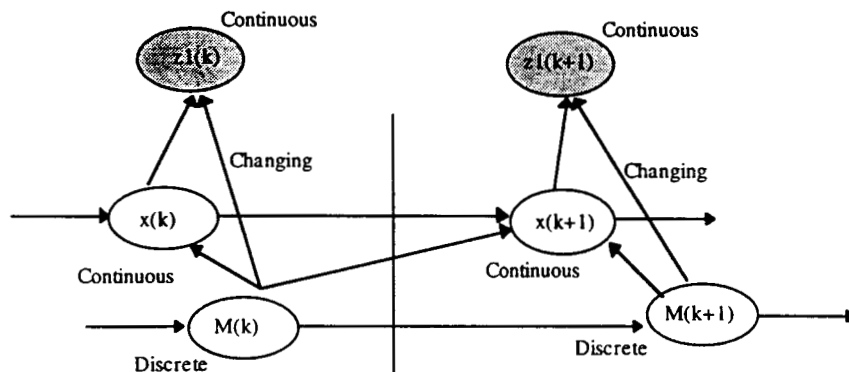


Fig. 2.2.2.4-2: VDBN for the interacting multiple model

As an example of tracking in the absence of information about the vehicle state (vehicle following, split, merge, etc.) a number of dynamic tests were carried out, one of which is shown in Fig. 2.2.2.4-3. Here, information from three sensors (radar, sonar, and an optical sensor,) were combined to get a fused estimate (Alag et al. 1995).

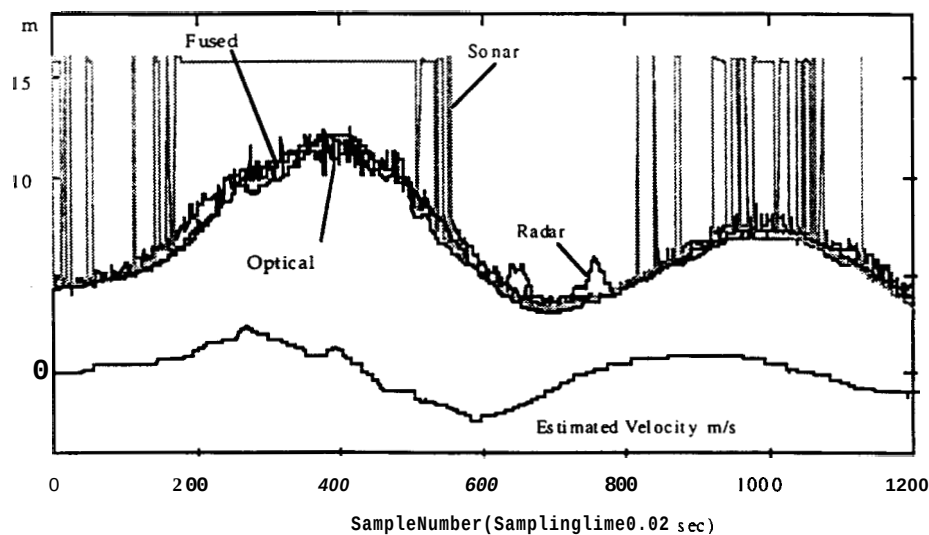


Fig. 2.2.2.4-3: Example of tracking in the absence of information about the vehicle state

2.2.2.5 Summary

In this section, we have applied concepts developed in Section 2.2.1 to the problem of intelligent sensor validation and fusion for automated vehicles in an intelligent vehicle highway system. We developed a framework for supervisory control of automated vehicles. We introduced multiple state tracking models to be used for validation and fusion of readings from the longitudinal sensors. These models correspond to the state transition model in our VDBN framework. We classified the vehicle states according to their tracking tasks. We investigated the efficacy of various algorithms developed in Section 2.2.1 for the tracking task by simulating platooning operations and applying it to real data obtained from normal platooning operations. **As** shown in this section the validation and fusion process will improve the safety of the **IVHS** and lead to better ride quality. Furthermore, the effects of sensor noise (when the noise characteristics are known) can be reduced by simple algorithms such **as** one state transition Kalman filter. However, in the presence of uncertainty in the process or sensor characteristics it is useful to use algorithms with a discrete node that can provide feedback on the noise characteristics. It is best to use the fusion algorithm which can learn the relative performance of the sensors over time. **As** in this case, it performs better than other algorithms in the presence of clutter.

Estimating noise characteristics and detecting sensor faults on-line lead to an increase in computation time. Using multiple sensors leads to an increase in the reliability and safety of the automated vehicle, as it provides a redundant estimate. However, this has to be weighed against the additional cost of incorporating a sensor and processing the data from the additional sensor. The same is true for developing intelligent self-learning algorithms. The gain in system performance by using these intelligent algorithms has to be weighed against the additional computational cost.

2.3 Comparison and Conclusion

This section compares the two approaches using a wide range of operating conditions and provides recommendations for choosing one method under various circumstances.

2.3.1 Comparative Simulations

The following graphs show the results for a series of experiments covering different types of noise and failures including random Gaussian noise, random bias, clutter, outliers, and steady bias. The fuzzy fusion was compared to a Kalman filter using probabilistic validation gates and the best algorithm (algorithm #7) which was introduced in Section 2.2 and is similar in design and performance to the PDAF. We will refer to it from here on as the Adaptive Probabilistic Data Association Filter (APDAF). The Kalman filter is designed to handle Gaussian noise in an optimal fashion while the APDAF is designed to achieve better results in the presence of clutter. The APDAF also learns the covariance of the noise for each sensor and adapts its covariance matrices accordingly. It is shown how both perfect and false information about the system affect the performance of the probabilistic approaches. The parameters of the fuzzy system were learned through machine learning techniques as explained earlier using training data with conditions similar to the test data. Two types of system behavior were investigated. First, a random walk was considered. Here the system can randomly jump from one point to the next within certain limits. Monte Carlo simulations of 100 runs over 100 steps each were carried out and compared. The second system under consideration was an inertia driven system which has more restriction on the behavior of the system. That is, it does not allow sudden jumps from one state to the other. Rather, smooth transitions are required between the states. The experiments were carried out in a systematic manner with the conditions as shown in Table 2.3-1.

#	Q _S	R _{S1}	R _{S2}	Q _A	R _{A1}	R _{A2}	γ	C	C _d	C _M	B	B _M	B _S	O
1	1	.01	.001	.9	.001	.01	9E99	0	0	0	0	0	0	0
2	1	.01	.001	1	.01	.001	9E99	0	0	0	0	0	0	0
3	1	.01	.001	.9	.001	.01	9	0	0	0	0	0	0	0
4	1	.01	.001	1	.01	.001	9	0	0	0	0	0	0	0
5	1	.01	.001	.9	.001	.01	9E99	1	.1	1	0	0	0	0
6	1	.01	.001	1	.01	.001	9E99	1	.1	1	0	0	0	0
7	1	.01	.001	.9	.001	.01	9	1	.1	1	0	0	0	0
8	1	.01	.001	1	.01	.001	9	1	.1	1	0	0	0	0
9	1	.01	.001	.9	.001	.01	9E99	0	0	0	1	.1	0	0
10	1	.01	.001	1	.01	.001	9E99	0	0	0	1	.1	0	0
11	1	.01	.001	.9	.001	.01	9	0	0	0	1	.1	0	0
12	1	.01	.001	1	.01	.001	9	0	0	0	1	.1	0	0
13	1	.01	.001	.9	.001	.01	9E99	1	.1	1	1	.1	0	0
14	1	.01	.001	1	.01	.001	9E99	1	.1	1	1	.1	0	0
15	1	.01	.001	.9	.001	.01	9	1	.1	1	1	.1	0	0
16	1	.01	.001	1	.01	.001	9	1	.1	1	1	.1	0	0
17	1	.01	.001	.9	.001	.01	9E99	0	0	0	0	0	2	0
18	1	.01	.001	1	.01	.001	9E99	0	0	0	0	0	2	0
19	1	.01	.001	.9	.001	.01	9	0	0	0	0	0	2	0
20	1	.01	.001	1	.01	.001	9	0	0	0	0	0	2	0
21	1	.01	.001	.9	.001	.01	9E99	1	.1	1	0	0	2	0
22	1	.01	.001	1	.01	.001	9E99	1	.1	1	0	0	2	0
23	1	.01	.001	.9	.001	.01	9	1	.1	1	0	0	2	0
24	1	.01	.001	1	.01	.001	9	1	.1	1	0	0	2	0

25	1	.01	.001	.9	.001	.01	9E99	0	0	0	1	.1	2	0
26	1	.01	.001	1	.01	.001	9E99	0	0	0	1	.1	2	0
27	1	.01	.001	.9	.001	.01	9	0	0	0	1	.1	2	0
28	1	.01	.001	1	.01	.001	9	0	0	0	1	.1	2	0
29	1	.01	.001	.9	.001	.01	9E99	1	.1	1	1	.1	2	0
30	1	.01	.001	1	.01	.001	9E99	1	.1	1	1	.1	2	0
31	1	.01	.001	.9	.001	.01	9	1	.1	1	1	.1	2	0
32	1	.01	.001	1	.01	.001	9	1	.1	1	1	.1	2	0
33	1	.01	.001	.9	.001	.01	9E99	0	0	0	0	0	0	1
34	1	.01	.001	1	.01	.001	9E99	0	0	0	0	0	0	1
35	1	.01	.001	.9	.001	.01	9	0	0	0	0	0	0	1
36	1	.01	.001	1	.01	.001	9	0	0	0	0	0	0	1
37	1	.01	.001	.9	.001	.01	9E99	1	.1	1	0	0	0	1
38	1	.01	.001	1	.01	.001	9E99	1	.1	1	0	0	0	1
39	1	.01	.001	.9	.001	.01	9	1	.1	1	0	0	0	1
40	1	.01	.001	1	.01	.001	9	1	.1	1	0	0	0	1
41	1	.01	.001	.9	.001	.01	9E99	0	0	0	1	.1	0	1
42	1	.01	.001	1	.01	.001	9E99	0	0	0	1	.1	0	1
43	1	.01	.001	.9	.001	.01	9	0	0	0	1	.1	0	1
44	1	.01	.001	1	.01	.001	9	0	0	0	1	.1	0	1
45	1	.01	.001	.9	.001	.01	9E99	1	.1	1	1	.1	0	1
46	1	.01	.001	1	.01	.001	9E99	1	.1	1	1	.1	0	1
47	1	.01	.001	.9	.001	.01	9	1	.1	1	1	.1	0	1
48	1	.01	.001	1	.01	.001	9	1	.1	1	1	.1	0	1
49	1	.01	.001	.9	.001	.01	9E99	0	0	0	0	0	2	1
50	1	.01	.001	1	.01	.001	9E99	0	0	0	0	0	2	1
51	1	.01	.001	.9	.001	.01	9	0	0	0	0	0	2	1
52	1	.01	.001	1	.01	.001	9	0	0	0	0	0	2	1
53	1	.01	.001	.9	.001	.01	9E99	1	.1	1	0	0	2	1
54	1	.01	.001	1	.01	.001	9E99	1	.1	1	0	0	2	1
55	1	.01	.001	.9	.001	.01	9	1	.1	1	0	0	2	1
56	1	.01	.001	1	.01	.001	9	1	.1	1	0	0	2	1
57	1	.01	.001	.9	.001	.01	9E99	0	0	0	1	.1	2	1
58	1	.01	.001	1	.01	.001	9E99	0	0	0	1	.1	2	1
59	1	.01	.001	.9	.001	.01	9	0	0	0	1	.1	2	1
60	1	.01	.001	1	.01	.001	9	0	0	0	1	.1	2	1
61	1	.01	.001	.9	.001	.01	9E99	1	.1	1	1	.1	2	1
62	1	.01	.001	1	.01	.001	9E99	1	.1	1	1	.1	2	1
63	1	.01	.001	.9	.001	.01	9	1	.1	1	1	.1	2	1
64	1	.01	.001	1	.01	.001	9	1	.1	1	1	.1	2	1

Table 2.3-1: Parameters used for experiments

where

is the batch number for 100 Monte Carlo experiments

Q_S is the modeled system noise covariance

R_{S1}, R_{S2} are the entries for the sensor covariance

Q_A is the perceived system noise covariance

R_{A1}, R_{A2} are the entries for the received sensor noise

γ is the validation gate for the probabilistic algorithms

C is a flag for the presence of clutter
 C_d is the variance for the clutter
 C_M is the magnitude for the clutter
 B is a flag for the presence of randomized bias
 B_M is the magnitude for randomized bias
 B_S is the magnitude for steady bias
 O is a flag for the presence of outliers

Details on the design of the probabilistic approaches can be found in Alag (1996) and Goebel (1996). The results of the two experiment types are summarized in Figs. 2.3-1 and 2.3-2. In the first set of experiments, a Monte Carlo simulation was performed for a random walk. 100 experiments each lasting 100 steps were carried out for each of the 64 conditions. Cases 1-2 investigate the case of little Gaussian noise with either perfect or imperfect information about system and noise variance for the probabilistic system. Cases 3-4 add validation gates to the APDAF and the Kalman filter. In cases 5-8, clutter was added while the first four conditions were repeated. Cases 9-16 add a small random bias, cases 17-32 add a steady bias, and for cases 33-64, outliers were added to the sensor readings. Each algorithm performs well under certain circumstances. The Kalman filter performs best in the presence of Gaussian noise alone as well as when the perfect or near perfect information about system and sensors is available. It performs very poorly without validation gate in the presence of non-Gaussian noise. The APDAF works best in the presence of clutter and in the presence of outliers when no steady bias is present. In the presence of a bias, the APDAF does not perform very well, although it does not exhibit the high mean error of the Kalman Filter. The fuzzy algorithm works best where the system and sensor behavior is not well known regardless of the conditions. It does not seem to be affected by any noise condition but stays at the same mean error for all cases. It is therefore the most stable algorithm, i.e., it stays within the smallest bounds of all algorithms.

The second set of experiments (Fig. 2.3-2) involved a similar scenario except that the random walk was replaced by an inertia driven system response taking into account the real behavior of many systems which cannot arbitrarily move from one state to another. Again 100 experiments each lasting 100 steps were performed for each of the 64 conditions. The results are somewhat different than the random walk experiments although the same parameters were used. Each algorithm shows weaknesses and strengths with the fuzzy filter performing in the smallest error bandwidth. However, where the Kalman filter and the APDAF are optimal, the fuzzy filter performs a little worse. The differences are not large, though, and the fuzzy filter performs better in certain scenarios than both of the probabilistic algorithms when the information about system and sensors is imperfect. It does not as well as the random walk experiments in the presence of clutter, probably because when the system is dynamically shifting, the parameters are very receptive to readings changes.

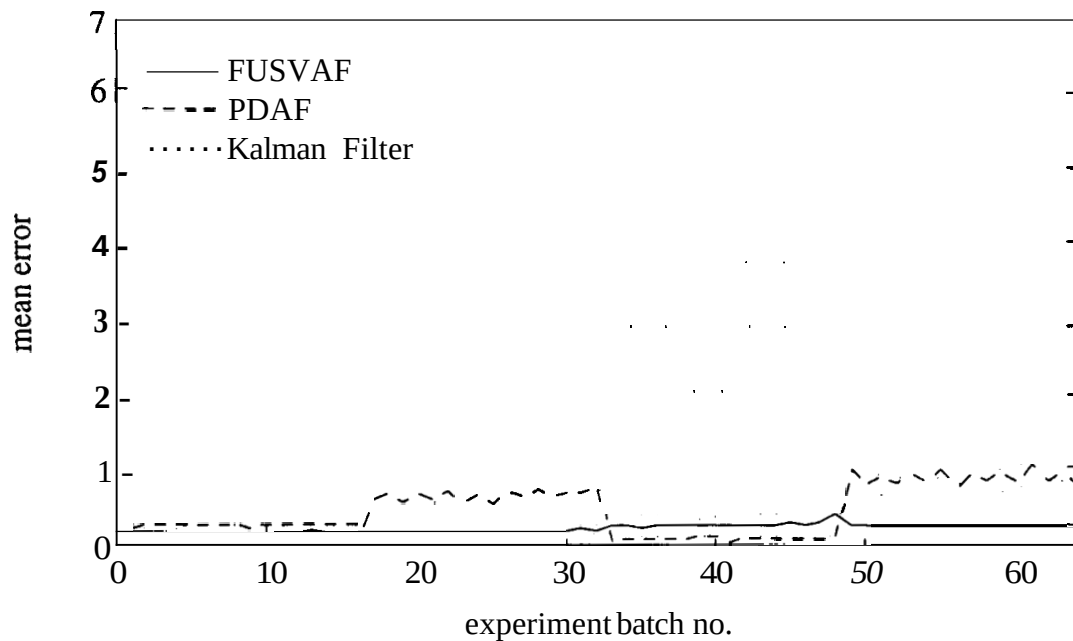


Fig. 2.3-1: Results of Monte Carlo runs for a random walk; comparison of fuzzy and probabilistic algorithms.

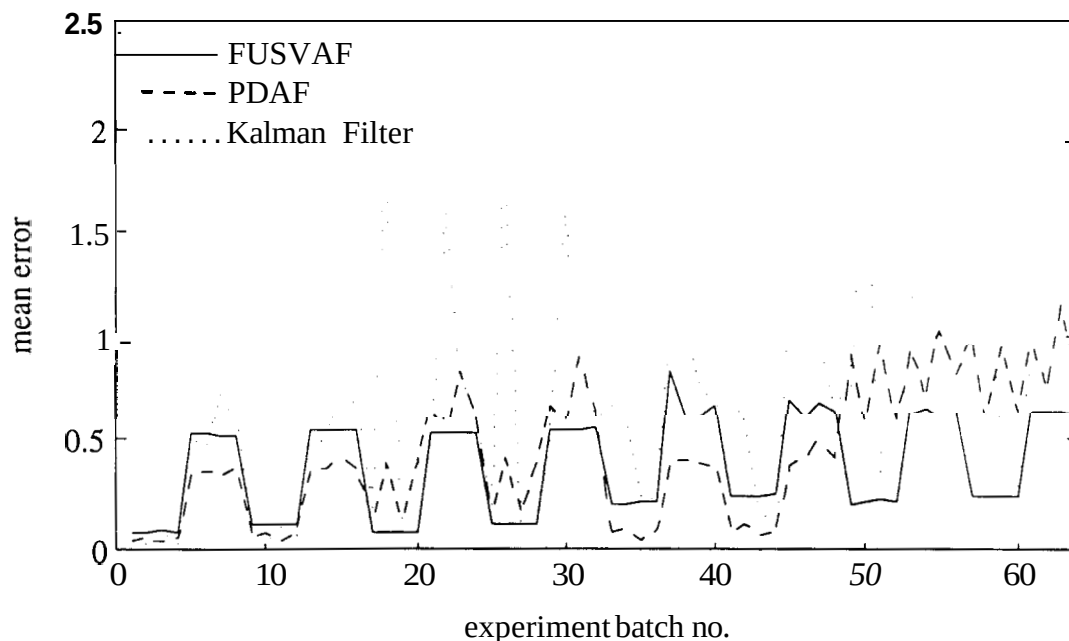


Fig. 2.3-2: Results of Monte Carlo runs for inertia driven system; comparison of fuzzy and probabilistic algorithms

The detailed results can be found in Goebel (1996). As an example of the output, consider Fig. 2.3-4. On the left side of the graph the mean error for 100 experiments are displayed. Each experiment is 100 steps long. The right side shows an example for how the algorithms behave. Only the Kalman filter and the fuzzy algorithm are shown. For this case only Gaussian noise was considered. The mean error for both algorithms is quite small with the Kalman filter performing better. Case 1 does not provide perfect information about the sensor noise while Case 2 does. Cases 3 and 4 have the same pattern as Cases 1 and 2 but also use a validation region. The fuzzy

algorithm is not affected by these changes **as** it does not access this type of information. With a validation gate, performance of the probabilistic algorithm actually degrades with some peaks recording worse than the fuzzy algorithm. This is due to the fact that the validation region for the probabilistic algorithm is kept at 3 times the variance which may not be the actual variance. Therefore, some readings are recorded as outliers while they are not really outliers. The algorithm then uses information only from previous readings and extrapolates from these which leads to poor performance. The line styles used to denote sensor readings, true data, Kalman filter and fuzzy filter are displayed in the legend in Fig. 2.3-3.

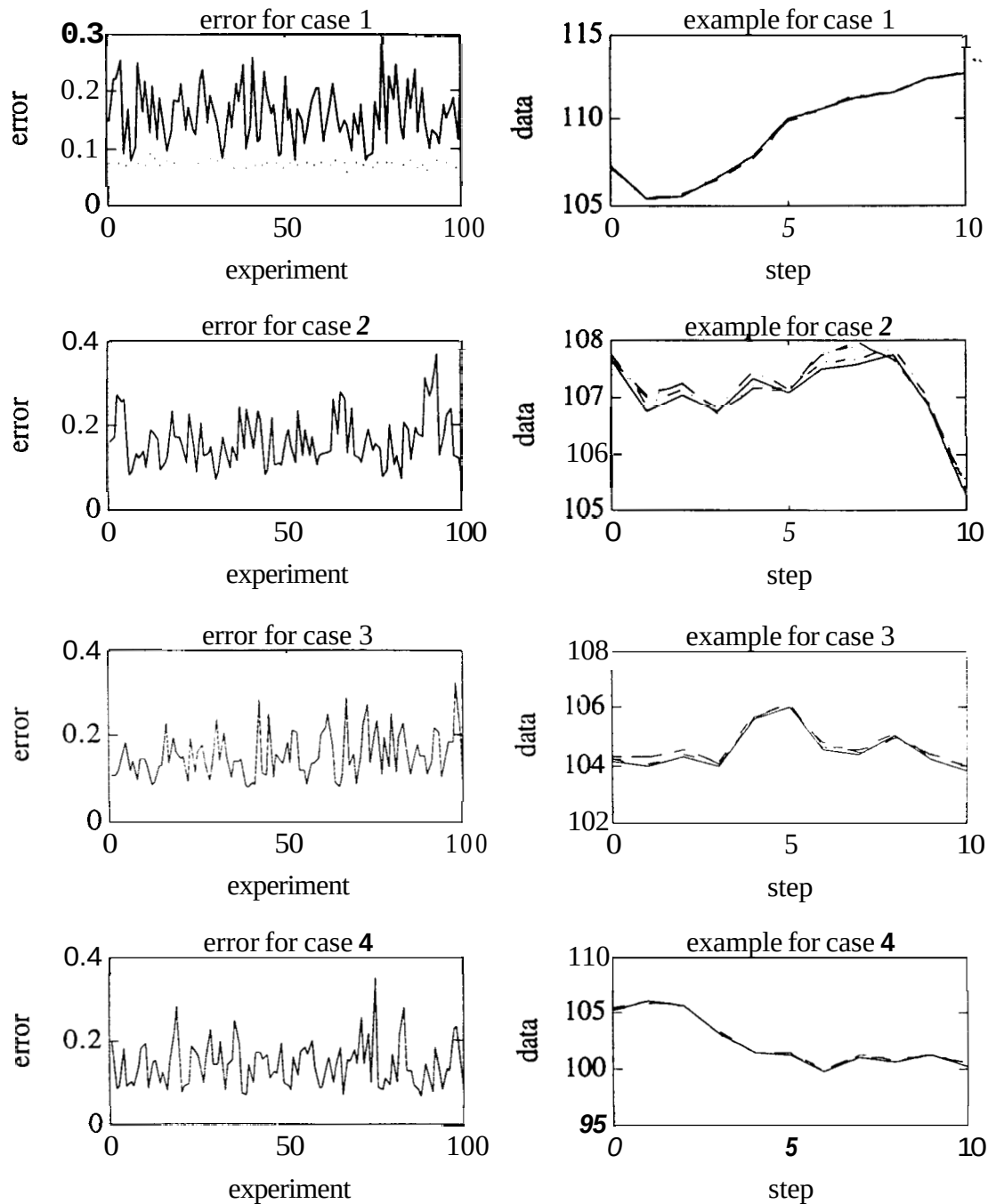


Fig. 2.3-4: Results of Monte Carlo runs, cases 1-4

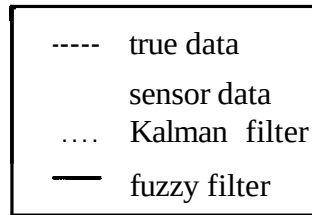


Fig. 2.3-3: Legend for line styles used

2.3.2 Recommendation and Summary

The FUSVAF algorithm for sensor validation and fusion provides a tool which can deal with Gaussian and non-Gaussian noise because the sensor characteristics can easily be modeled according to their characteristics. The parameters are tuned off-line using machine learning techniques. It also grants a better understanding of the system modeled. It performs the fusion task more satisfactorily in the presence of non-Gaussian noise than compared probabilistic approaches. The performance of the algorithm is rated according to smoothness of returned fused value, error spacing of the simulation tool, robustness in the presence of non-Gaussian noise and sensor failure, and computational expense. The FUSVAF algorithms perform better in all categories than the probabilistic approaches to which it was compared. This is due in part to the additional information which can be included in the shape of the validation regions. In the presence of Gaussian noise alone, the FUSVAF algorithm still performs very well with an error only slightly higher than the Kalman Filter based approaches. No algorithm is absolutely fool proof. There exist conditions at which every algorithm can be thrown off such as improper settings of the parameters or lack of proper sensor measurements. The latter problems can be alleviated through the use of sensors which use different techniques and physical configurations. Here, the wheel speed sensor of the follower vehicle and the preceding car are used to create a virtual sensor which is also used for fusion. When all forward looking sensors are knocked out, the control algorithm can still operate safely on the value provided by this indirect sensor. Other sensors which do not use the same principle should be integrated to provide a suite of sensors which operate independently from each other and provide the maximum amount of safety. Such sensors could use GPS information, but even sensors which are already on-board (such as a tachometer together with the information about gear ratio or accelerometers) are useful for this purpose. The integration of quantitative information from vision sensors is a straightforward extension. Thus the use of FUSVAF algorithms can lead to an improvement in both comfort and safety in IVHS.

The FUSVAF algorithm has certain advantages over the probabilistic algorithm in specific situations. However, it does not operate optimally in all cases. In the presence of Gaussian noise alone, the Kalman filter based approaches perform slightly better since they are designed for optimal performance in that special situation. In the presence of non-Gaussian noise, the FUSVAF algorithm performs better by a large degree. If knowledge about sensor behavior exists, i.e., if regions or environmental conditions are known where the sensor acts in a Gaussian fashion, and the system is in a steady state, the Kalman filter should be chosen. If, on the other hand, the conditions change unpredictably or the noise is known to be of non-Gaussian nature, the FUSVAF algorithm should be used since it performs more robustly over a wide range of non-Gaussian as well as Gaussian noise. Each filter has to be tuned to perform in an optimal way. Ideally, sensor data exist to allow tuning for the parameters. If such data do not exist, a choice of the parameters should be used as outlined earlier in this section. The strength of the FUSVAF algorithm is its ability to incorporate knowledge about sensor characteristics and external conditions through the use of appropriate validation curves. These curves can be updated dynamically. The FUSVAF algorithm is a computationally less expensive validation and fusion procedure than the APDAF.

3 Sensor Evaluation Under Adverse Environmental Conditions; A Preliminary Exploration

Since fault compensation and diagnosis of aberrant sensor behavior needs information about sensor characteristics, we performed experiments on the longitudinal distance sensors. The experiments aimed at evaluating performance under a wide array of environmental condition and tried to connect sensor characteristics with a particular event. These events ranged from heavy vibration to obstruction of the sensors with different types of debris, fog, or rain. Although we show numerical results, the value of these experiments is qualitative because we dealt with prototype sensor technology which undergoes an ongoing change.

However, since our goal is to show management of sensor uncertainty and malfunction; we can use those results to conceptually show the solution although the actual characteristics may have shifted. Evaluation of the final sensors will be necessary to tune the validation and diagnosis scheme to the specific sensors used.

Several statistical quantities were evaluated such as mean, variance, and power spectrum of a frequency analysis. The following sections show experimental setup and results for the different test settings. We start with the vibration tests in Section 3.1, followed by the obstruction experiments (Section 3.2), the fog experiments (Section 3.3), and the rain experiments (Section 3.4).

The experiments (Bellm 1995) used a two vehicle platoon consisting of a lead vehicle and a follower vehicle (Fig. 3-1). The lead and follower vehicles remained stationary during the tests. The vehicle separation distance is represented by the parameter D . The series of tests included separation distances from 0-15 meters. In addition, the vehicle separation range explores the outer edge of the performance envelope for the sonar sensor. The range of vehicle separation distance tested represents the lower half of the radar sensor's operational range.

The static tests (all tests but the vibration tests) recorded the sensor readings under motionless conditions. The sensors remained mounted on the vibration test stand for testing efficiency. The longitudinal sensors were positioned at the same height as the vehicle hard mount during the static tests. This ensured that mounting on the test stand under static conditions was equivalent to vehicle mounting. Fig. 3-2 shows the test configuration.

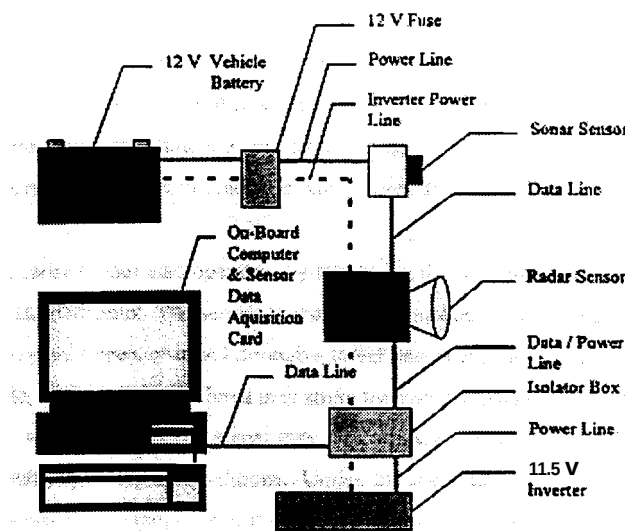


Fig. 3-1: Test ConFIGuration

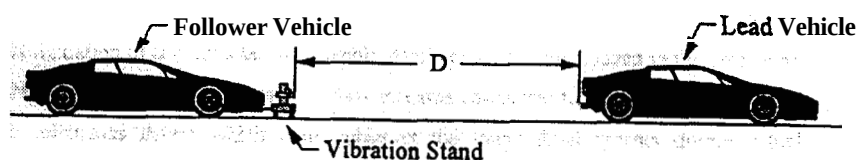


Fig. 3-2: Test ConFIGuration for Static and Vibration Tests

For a more detailed listing of the experiements, the reader is referred to Bellm (1995). ..

3.1 Vibration Experiments

The vibration test aimed at evaluating performance of sensors when subject to vertical displacement. This is the case constantly due to resonance from the engine and the road while the vehicle travels over uneven pavement. We simulate this environment with a test set up which features a sinusoidal vertical motion to allow repeatable results. The test stand is displayed in Fig. 3.1-1. It is equipped with a variable speed electric DC motor. This allowed versatility in the frequency of vibration selected. The DC motor is connected to the flywheel. The flywheel was manufactured with holes placed at various radii. These concentric holes allowed the amplitude of the vibration to be varied. The vibrations of the test stand were measured using an accelerometer. The vibrations occur at a frequency of approximately 0.8 Hz with a motor power setting of 10 and maximum amplitude ($\pm 0.15\text{m}$). The flywheel is connected to a linkage which translates the angular motion of the flywheel into vertical motion. The sonar and radar sensor are mounted to a hard point on the linkage.

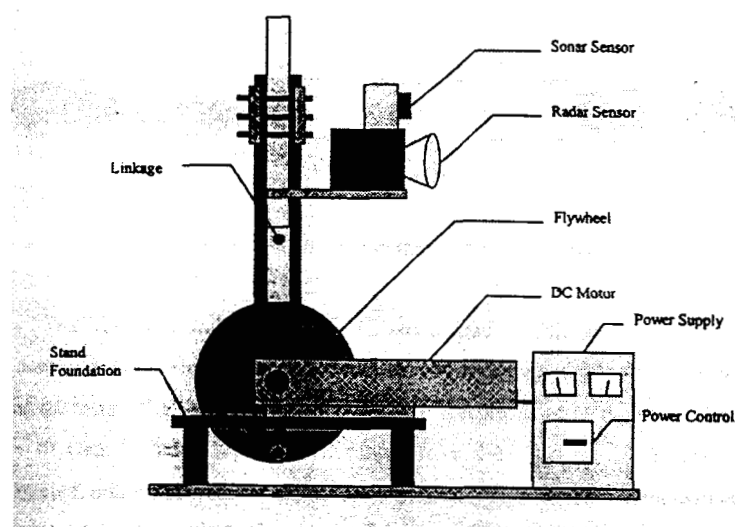


Fig. 3.1-1: Test stand for vibration experiments

A second series of vibration tests was conducted measuring acceleration to distinguish between sensor characteristics and test stand characteristics. The vehicles were arranged with a separation distance of 1 m. The on-board vehicle braking accelerometer was mounted on the vibration stand sensor mount in the vertical, horizontal and longitudinal directions. Sensor measurements were recorded under static and vibration conditions. Figs. 3.1-2 through 3.1-4 display the frequency spectra of the accelerometer which is an important metric used to identify peaks in the spectra due to the test set-up. The greatest peak magnitudes of the accelerometer signal are at frequencies of 0.8 Hz and higher order harmonics of that frequency. This is the frequency at which vibrations were

produced by the test stand. Almost all sensors *peak* magnitudes match those of the accelerometer signal.

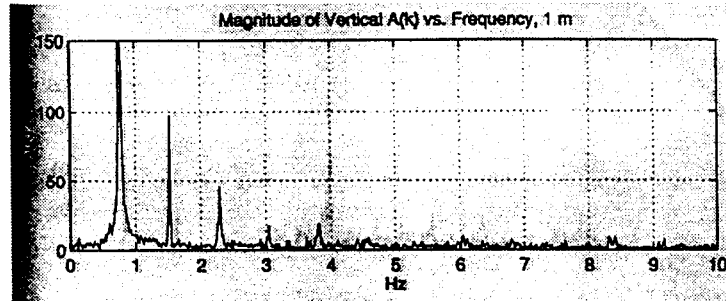


Fig. 3.1-2: Frequency spectrum for accelerometer on test stand, vertical direction

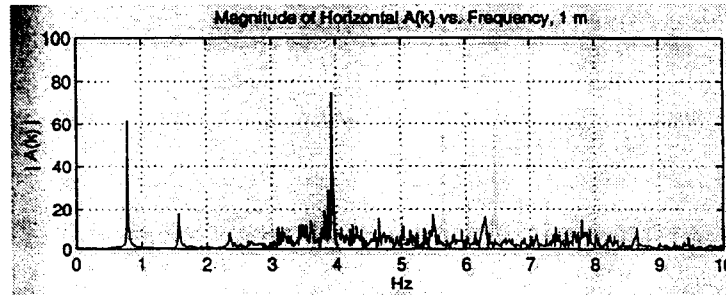


Fig. 3.1-3: Frequency spectrum for accelerometer on test stand, horizontal direction

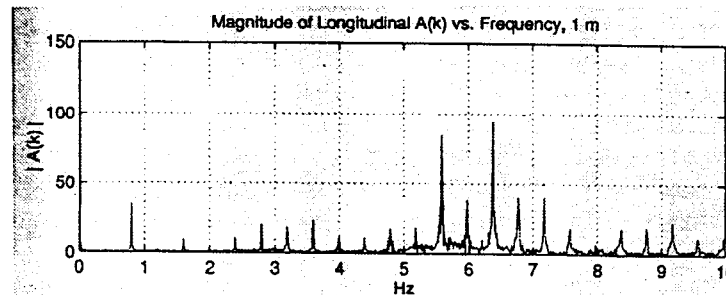


Fig. 3.1-4 Frequency spectrum for accelerometer on test stand, lateral direction

3.1.1 Sonar Sensor

The mean and standard deviation of the sonar sensor over the vehicle separation distance range of 10m are illustrated in Fig. 3.1.1-1. The data are shown on separate graphs for clarity. The dashed line represents the data of a "perfect" sensor capable of measuring the vehicle separation distance with complete accuracy. The sonar sensor performance during vibration is similar to static conditions over the entire range of vehicle separation distances. Note that the offset is assumed to be mostly due to a poorly calibrated sensor. **At** this stage in the project calibration was not a standard procedure at the Richmond Field station (**RFS**)PATH facilities.

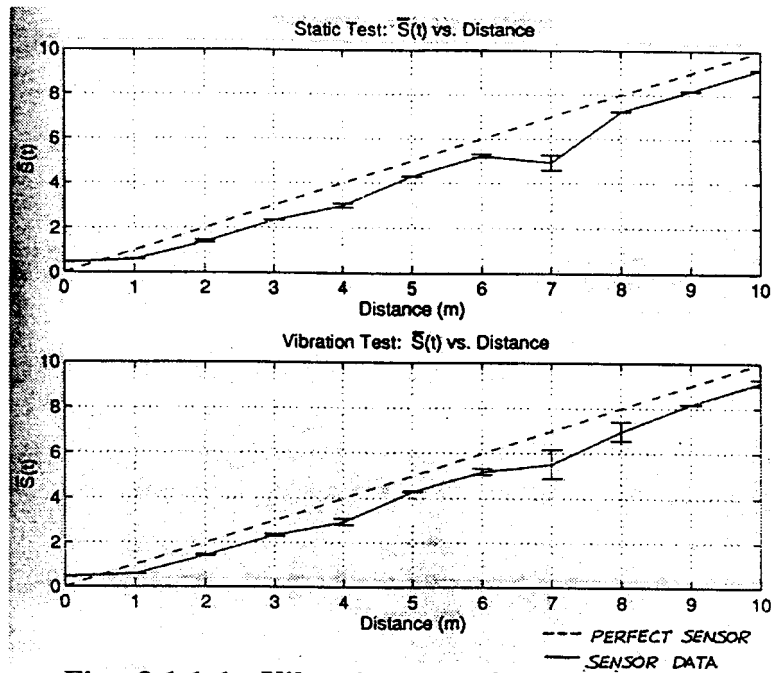


Fig. 3.1.1-1: Vibration test for sonar sensor

The variance of sonar sensor measurements increased only slightly over the operating range. One has to distinguish between valid readings and outliers. Outliers are not considered in the variance plot. The number of outliers increases steadily between 5 and 9 meters, beyond which point no proper readings can be found. The sonar sensor experienced an approximately 8 percent fault rate at a vehicle separation distance of 0 m under vibration conditions. The sensor measured accurately over the 1-6 m range, experiencing no faults. The number of faults began to increase slightly at separation distances greater than 6 m. The sonar sensor faults increased dramatically at distances greater than 8 m. The sonar sensor variation begins to increase in this upper range because there are numerous high variance readings which do not meet the sensor fault criteria. At a vehicle separation distance of greater than 9 m there are relatively few sensor readings which are not considered faults.

The magnitude scale of the frequency response plots was adjusted minimally over the vehicle separation distance range to illustrate the characteristics of the frequency spectra. The overall trend in the frequency response of the sonar sensor under static and vibration conditions is illustrated in Fig. 3.1.1-3. Peak magnitudes at multiple frequencies are evident during vibration. However, these peaks are attributed to the test stand and are not characteristic of the sensor as can be seen from the accelerometer readings as shown in Figs. 3.1-2 - 3.1-4 which display the same peaks as seen in the frequency spectra for the sensors. The overall magnitude of the frequency response does not increase significantly over the operating range except where it can be attributed to the test stand characteristics.

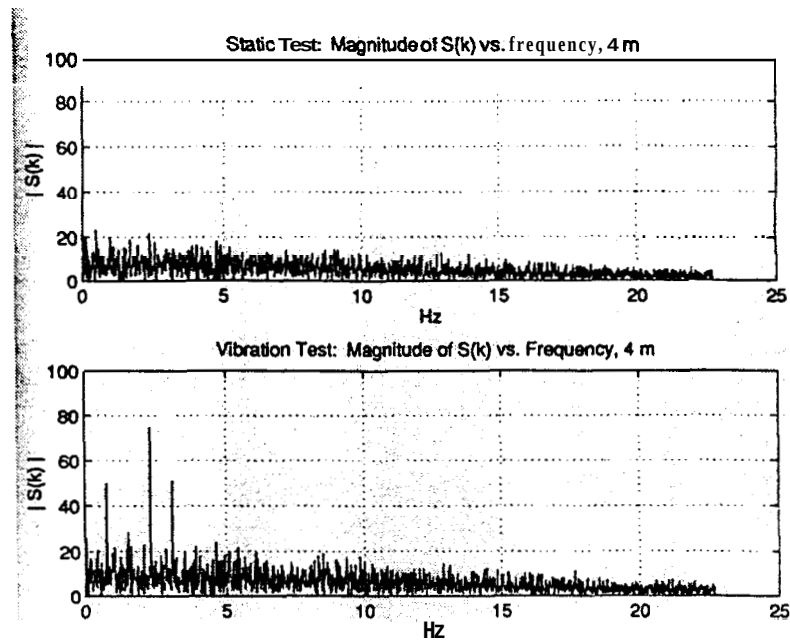


Fig. 3.1.1-3: Frequency spectrum for sonar sensor

3.1.2 Radar Sensor

The mean and standard deviation of the radar sensor over a vehicle separation distance range of 15 m are illustrated in Fig. 3.1.2-1. Again, we believe sensor performance is affected by erroneous calibration. Characteristic deviation in radar sensor measurements appeared in the **6-8 m** and 10-13 m distance ranges under both static and vibration test conditions. The radar sensor measures higher in these two target distance ranges. This has been attributed to the quantization effect, an inherent characteristic of the radar sensor (Agogino et al. 1995). The radar sensor performance was slightly affected by the vibration conditions. This is illustrated by an increase in radar sensor variance over the separation distance range. The variance of the radar sensor under static and vibration conditions is shown in Fig. 3.1.2-2. The variance is extremely high at a separation distance close to 0 m. This is due to the fact that although the majority of the radar sensor measurements were filtered out at this distance, measurements which did not meet the radar sensor fault criteria varied significantly. **As** a result, the variance of the radar sensor measurements increased significantly under vibration conditions.

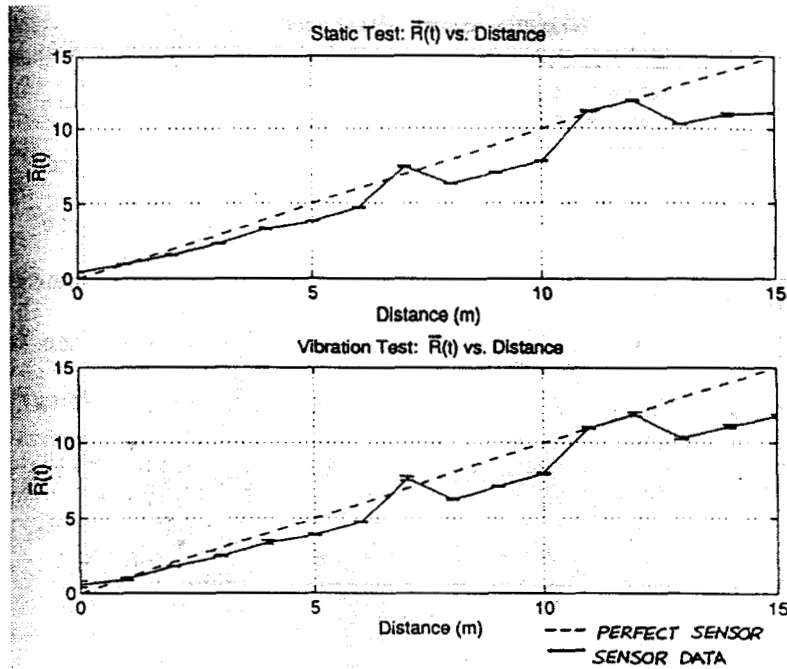


Fig. 3.1.2-1: Vibration test for radar sensor: mean and standard deviation

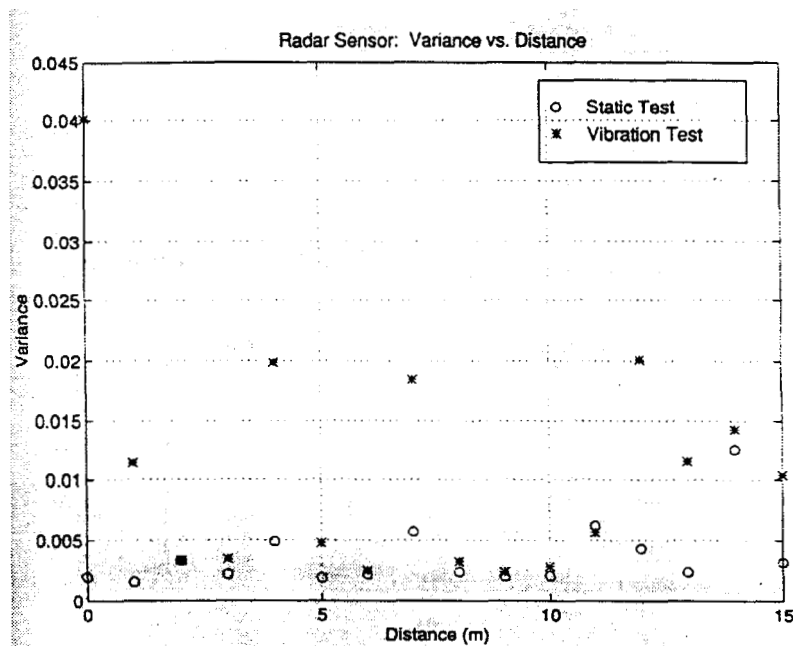


Fig. 3.1.2-2: Vibration test for radar sensor: variance vs. distance

Frequency spectrum plots for the radar sensor were obtained for vehicle separation distances from 0-15 m. The overall trend in the frequency response of the radar sensor under static and vibration conditions appears in Fig. 3.1.2-3. Peak magnitudes at multiple frequencies are evident during vibration. The output magnitude does increase as a function of measured distance. However, the trend is not as significant as the sonar sensor frequency response. The radar sensor output magnitude is consistently greater over all frequencies during vibration as well. The radar sensor

was tested in the lower half of its operation range. **As** a result, no dramatic increase in frequency response due to measurement outside the sensor's operation performance was experienced within this measurement range.

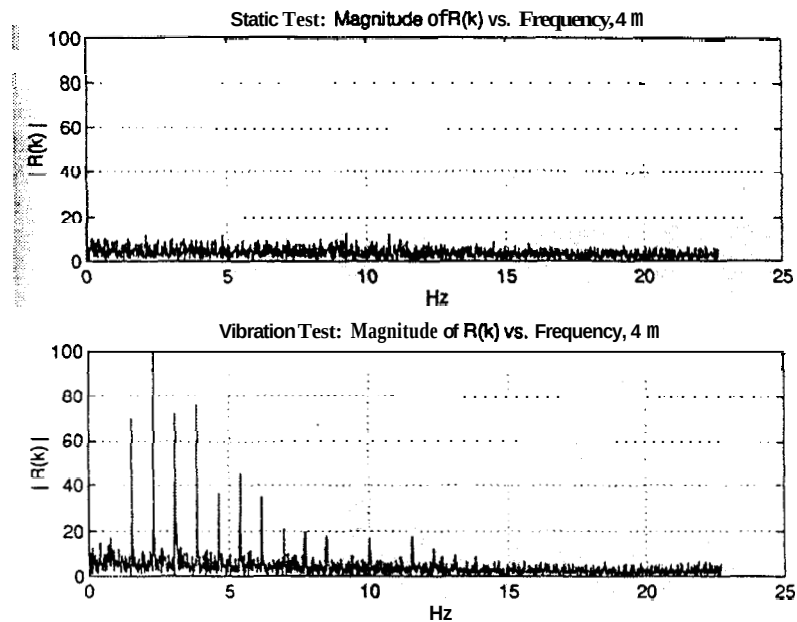


Fig. 3.1.2-3: Frequency spectrum for radar sensor

3.2 Obstruction Experiments

The longitudinal sensors are exposed to the environment in the current test vehicle configuration. As a result, debris could become lodged on the vehicle and obstruct the sensors. Plastic debris and dirt debris tests were conducted to simulate the presence of miscellaneous obstructions which could block the longitudinal sensors during vehicle operation in the IVHS. All debris tests rendered the sonar sensor inoperable. The performance of the radar sensor degraded as discussed below.

3.2.1 Plastic Debris

The experimental configuration of the plastic debris test was similar to that of the static and vibration tests. The longitudinal sensors were mounted in the same location on the vehicle for vehicle operation. Thin plastic (plastic bag material) was placed flush against the longitudinal sensors to simulate debris which could remain lodged in place by air flow during vehicle motion. The series of tests included vehicle separation distances from 0-15 meters. The sonar sensor did not work while obstructed. Results for the radar sensor can be seen in Fig. 3.2.1-1. There is a significant increase in the mean of the readings and the quantization region appears to be shifted. The variance appears to be unaffected as shown in Fig. 3.2.1-2.

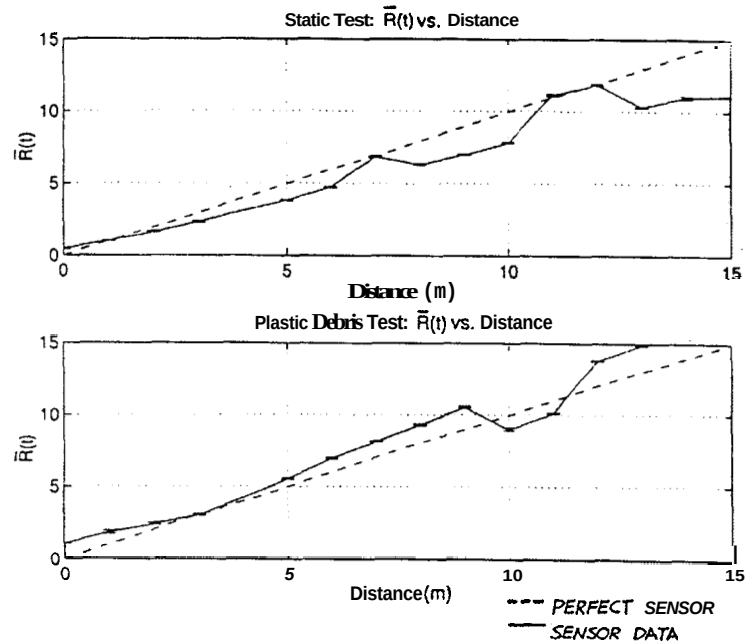


Fig. 3.2.1-1: Plastic Debris: Radar Sensor

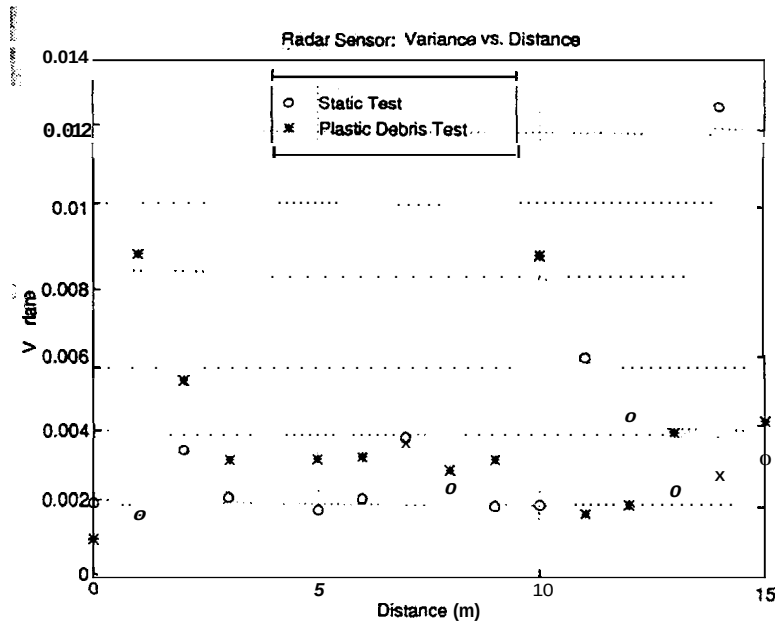


Fig. 3.2.1-2: Plastic Debris: Radar Sensor

3.2.2 Dirt Debris

The dirt debris test was conducted to simulate the effect of dirt and mud which may coat a motor vehicle during operation in adverse conditions. This effect is common for vehicles traveling in winter conditions or traveling on dirt roads, off-road ventures, and neglect of vehicle cleaning. A layer of dirt approximately 1 mm thick coated the exterior of the sonar and radar sensor for this series of tests.

The mean of the sensor readings is represented by a line with much lower slope than the real data (Fig. 3.2.2-1). In fact, only the characteristic bump in the quantization region is an indication of the functioning of the radar sensor. The variance of the radar sensor measurements (Fig. 3.2.2-2) was affected by the presence of the dirt debris resulting in an increase.

The frequency spectrum for the radar sensor signal over the vehicle separation distance range of 0-15 m was relatively similar for static open air and dirt debris conditions. However, the noise at low frequencies increased substantially as the vehicle separation distance increased.

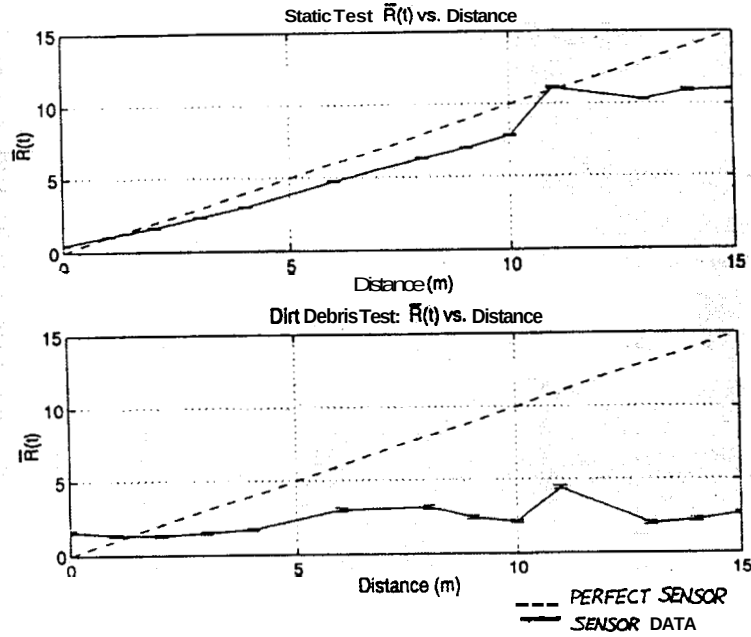


Fig. 3.2.1-1: Dirt Debris: Radar Sensor

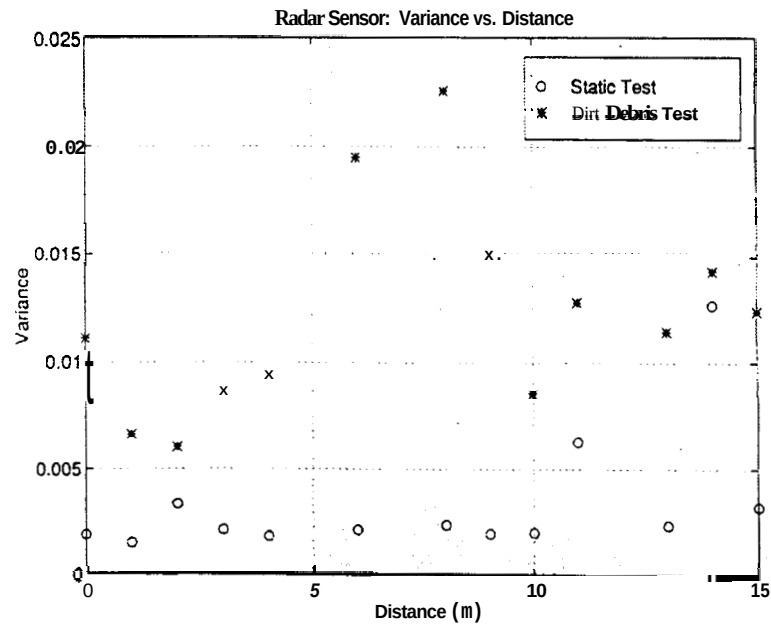


Fig. 3.2.1-2: Dirt Debris: Radar Sensor

3.3 Fog Experiments

The fog experiments did not affect the radar sensor. However, they significantly affected the performance of the sonar sensor. The sonar sensor signal in open air and fog environment is shown in Fig. 3.3-1. Conditions were such that a laser beam with 0.5W power could not penetrate the fog over 4m. The variance of the sonar sensor measurements increased under fog environment

conditions. Periodic outliers are identified over the observed period. The frequency spectrum changed dramatically under fog conditions. The noise level of the sonar signal increased dramatically over the frequency spectrum in fog.

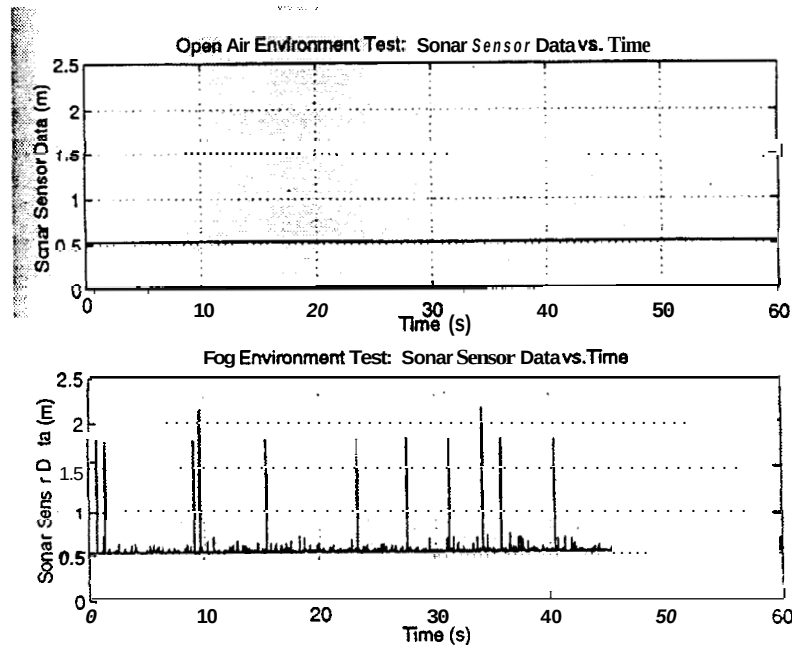


Fig. 3.2.1-1: Fog Experiment: Sonar Sensor

3.4 Rain Experiment

The sonar signal in open air and rain is shown in Fig. 3.4-1. The rain environment test increased the variance of the sonar signal while the mean remained essentially the same. The variance increased from 0.011 under open air conditions to 0.046 in the rain environment. The frequency response of the sonar sensor signal under open air and rain environment conditions was similar. There are outliers which are smaller than the actual distance, probably due to reflections of the signal from rain drops.

Radar sensor measurements were positively offset in the rain environment. Fig. 3.4-2 shows the signal was offset by approximately 0.2 m. The mean measurement increased from 4.09 m to 4.35 m. The variance of the signal increased from 0.005 under open air conditions to 0.021 in the rain environment. The frequency response of the radar sensor signal under open air and rain environment conditions was similar over most of the frequencies. However, the signal possessed some peak magnitudes at very low frequencies under rain conditions.

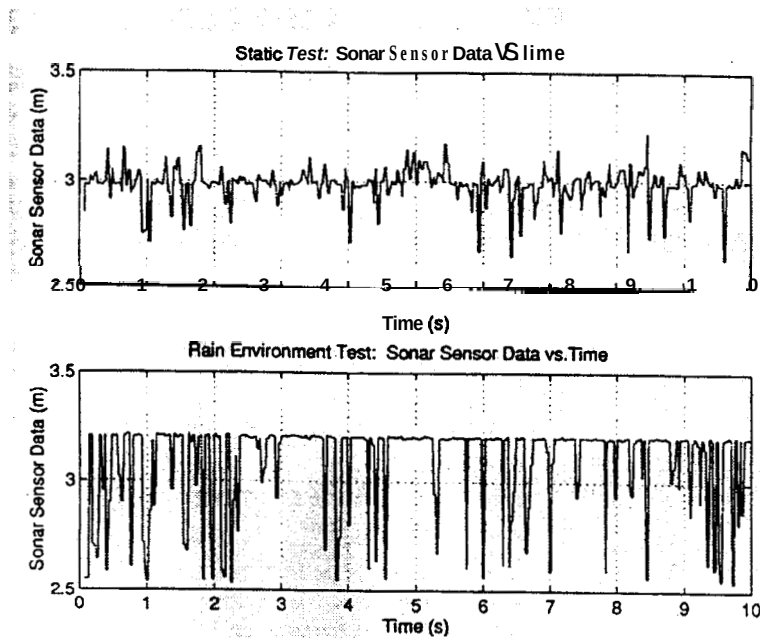


Fig. 3.4-1: Rain Experiment: Sonar Sensor

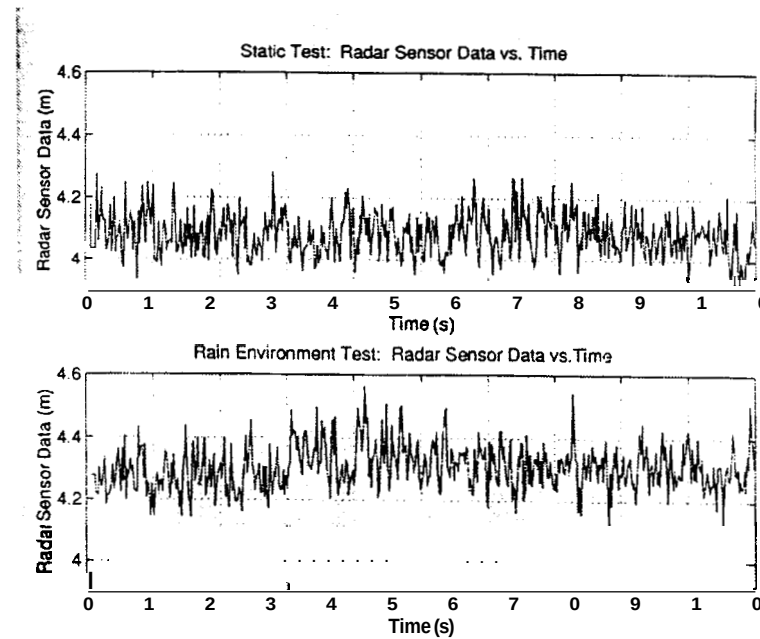


Fig. 3.4-2: Rain Experiment: Radar Sensor

3.5 Summary and Conclusions

The sensor experiments gave useful insight into the characteristics of sensors subject to a variety of environmental conditions. Apart from the sensor footprint, the sensor response was obtained for rain, fog, various kinds of obstructions, and vibrations. The sensors reacted with either an increase in variance or an underestimate in the measured distance. Changes in the frequency response were not significant to be attributed to a particular environmental condition. The insight gained is of both qualitative and quantitative nature. If sensor behavior is better understood under adverse conditions

then the **APDAF** and the **FUSVAF** algorithm can adjust for the changes in sensor behavior. Further experiments should explore other failure modes. Roadside information about environmental conditions such as temperature, humidity, visibility, etc. can help in allowing on-line recitification of aberrant sensor readings which were attributed to particular environmental conditions. It is also recommended to establish a procedure for sensor calibration before they are used in tests and experiments.

4. Conclusions and Recommendations

This report shows the development of several algorithms for sensor validation and sensor fusion. We conclude that the fusion of partly redundant sensor information allows to better estimate the state of the system and hence improve its safety. Knowledge about sensor characteristics and environmental conditions can be integrated into the validation and fusion process as shown to various degrees in the probabilistic and fuzzy algorithms developed. These characteristics can be obtained experimentally and need to be updated as the sensor prototypes change. The information from the corrected sensor readings improves the response of the control algorithms.

It is recommended that further experiments investigate other failure modes of the sensors through a structured failure mode or fault tree analysis. Future research should also address the relationships between sensor failure and hazards. This information should then be used to estimate the likelihood of hazards and to develop optimal decision making modules and an intelligent decision advisor to avert emergency situations. The use of other sensors such as vision sensors and GPS should be integrated into the sensor fusion scheme to provide a suite of sensors which has independent failure modes. It is also recommended that a standard calibration procedure be established before use of the sensors in testing on the vehicles.

References

- Agogino, A.M., Goebel, K., and Alag, S. 1995. Intelligent Sensor Validation and Sensor Fusion for Reliability and Safety Enhancement in Vehicle Control. MOU132, Final Report, UCB-ITS-PRR-95-40, *California PATH Research Report*.
- Agogino, A., Alag, S., Goebel, K. 1995. A Framework for Intelligent Sensor Validation, Sensor Fusion, and Supervisory Control of Automated Vehicles in IVHS. *Proceedings of the ITS America Annual Meeting*, Washington, D.C.
- Alag, S. 1996. *A Bayesian Decision-Theoretic Framework for Real-Time Monitoring and Diagnosis of Complex Systems: Theory and Application*. PhD Thesis, Department of Mechanical Engineering, University of California at Berkeley.
- Alag, S., Goebel, K., and Agogino, A. 1995. Intelligent Sensor Validation and Fusion used in Tracking and Avoidance of Objects for Automated Vehicles. *Proceedings of the ACC 1995*, Seattle.
- Bar-Shalom, Y. and Tse, E. 1975. Tracking in a Cluttered Environment with Probabilistic Data Association. *Automatica*, vol. 11, pp. 451-460.
- Bar-Shalom, Y. and Fortmann, T. E. 1988. Tracking and Data Association. Boston, MA: Academic.
- Bar-Shalom Y. and Li, X. 1993. Estimation and Tracking: Principles, Techniques, and Software. Boston, MA: Artech House.
- Bellm, D. 1995. Characterization of the Sonar and Radar Distance Sensing Devices Under Sub-Optimal Conditions for the California Partners for Advanced Transit and Highways. MS Report, Department of Mechanical Engineering, University of California at Berkeley.
- Bryson, A. E., and Ho, Y.-C. 1975. Applied Optimal Control: Optimization, Estimation, and Control. Hemisphere Publishing, Washington, D. C.
- Connolly, T. R. 1996. Transition Maneuvers in an Automated Highway System. MS Thesis, Department of Mechanical Engineering, University of California at Berkeley.
- Frankel, J. A., Horowitz, L. and Li, P. 1994. Robust Platoon Maneuvers for AVHS, *California PATH Technical Note 94-9*.

- Garg, V. 1995. Fault Detection in Nonlinear Systems: An Application to Automated Highway Systems. PhD Thesis, Department of Mechanical Engineering, University of California at Berkeley.
- Goebel, K., and Agogino, A. 1996. An Architecture for Fuzzy Sensor Validation and Fusion for Vehicle Following in Automated Highways. *Proceedings of the 29th ISATA*, Florence, Italy.
- Goebel, K. 1996. Management of Uncertainty for Sensor Validation, Sensor Fusion, and Diagnosis Using Soft Computing Techniques. Ph.D. Thesis, University of California at Berkeley, Berkeley.
- Goldberg, D.E. 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley, Reading, MA.
- Grefenstette, J., *A User's Guide to GENESZS*, Version **5.0**, 1990.
- Hashemipour, H. R., Roy, S. and Laub, A. J. 1988. Decentralized Structures for Parallel Kalman Filtering. *IEEE Trans. Automatic Control*, 33(1):88-96.
- Hedrick, J. K., McMahon, D., Swaroop, D., Garg, V., Gerdes, J., Maciucă, D., Blackman, T. and Yip, P. 1995. Longitudinal Control Development for NHS Fully Automated and Semi-Automated Systems-Phase 1. *California PATH Research Report, UCB-ITS-PRR-95-4*.
- Hitchcock, A. 1993. Intelligent Vehicle/Highway System Safety: Specification and Hazard Analysis. *Transportation Research Board No. 930435*.
- Holland, J. H., *Adaptation in Natural and Artificial Systems*, Univ. Michigan Press, Ann Arbor, 1975.
- Jang, J.-S.R. 1993. ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Transactions on Systems, Man and Cybernetics*, vol.23, (no.3):665-85.
- Kalman, R. E. 1960. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME, Series D*, vol. 82, pp. 35-60.
- Kenley, C. R. 1986. Influence Diagram Models with Continuous Variables. Doctoral Dissertation Stanford University, Department of Engineering-Economic Systems.
- Khedkar, P., and Keshav, S. 1992. Fuzzy Prediction of Time Series. *Proceedings of the IEEE International Conference on Fuzzy Systems*, San Diego, CA, USA, 8-12 March.
- Kim, Y.J., W.H. Wood, and A.M. Agogino. 1992. Signal Validation for Expert System Development. *Proceedings of the 2nd International Forum on Expert Systems and Computer Simulations in Energy Engineering*. Erlangen, Germany, pp. 9-5-1 to 9-5-6.
- Manyika, J. and Durrant-Whyte, H. F. 1994. Data Fusion and Sensor Management: a Decentralized Information-Theoretic Approach. Ellis Horwood, New York.
- McMahon, D. H., Narendran, V. K., Swaroop, D., Hedrick, J. K., Chang, K. S. and Devlin, P. 1992. Longitudinal Vehicle Controllers for IVHS: Theory and Experiment. *Proceedings of American Control Conference*. vol. 2.
- Narendran, V. K. Transition Maneuvers in Intelligent Vehicle Highway Systems. 1994. PhD thesis, Department of Mechanical Engineering, University of California, Berkeley.
- Olmsted, S. M. 1984. On Representing and Solving Decision Problems. Doctoral Dissertation, Department of Engineering-Economic Systems, Stanford University.
- Pearl, J. 1988. Probabilistic Reasoning in Intelligent Systems. San Mateo, Ca: Morgan Kaufmann.
- Rao, B. Y. S. and Durrant-Whyte, H. F. 1993. A Decentralized Bayesian Algorithm for Identification of Tracked Targets. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 6.
- Varaiya, P. 1991. Smart Vehicles on Smart Roads: Problems of Control, *PATH Technical Memorandum 91-5*, UC Berkeley.
- Varaiya, P., and Shladover, S. 1991. Sketch of an IVHS Architecture, *PATH Technical Report UCB-ITS-PRR-91-03*.

Wright, S. 1921. Correlation and Causation. *Journal of Agricultural Research*. Vol. 20, pp. 557-585.