Developments

Passive Ranging Using Illuminators of Opportunity

by Steven Zelubowski

A Pragmatic Overview of Object Oriented Programming

by Michael Mel

Contrasting Approaches to Noise Filtering: Kalman Filters and Artificial Neural Systems

by Paul Smith

ISSUE II - OCTOBER 1990

IDI Infotec Development Inc
3611 S. Harbor Blvd., Suite 260, Santa Ana, CA 92704 (714) 549-2182

The Technical Journal of Infotec Development Inc.

Contrasting Approaches to Noise Filtering: Kalman Filters and Artificial Neural Systems

by: **Paul C. Smith**
Infotec Development, Inc.
Costa Mesa, CA

Abstract

Kalman Filters are currently used to optimally estimate the value of a measurement in spite of noise associated with the sensors. The encoder configuration of an Artificial Neural Systems (ANS) also has the ability to filter noise from multiple input sources. There are some similarities, and many differences between the two techniques. This paper compares and contrasts the two techniques and suggests a method for incorporating features of multi-state Kalman filters to an organization of ANS encoders.

Contrasting Approaches to Noise Filtering: Kalman Filters and Artificial Neural Systems

Introduction

In the filtering world we talk about the Truth, which is the exact perfect solution, and measurements of the answer which aren't as accurate as we desire. The Filter combines measurements to produce a more accurate solution, closer to Truth.

For example, suppose we were traveling in a car at 55.345 MPH. Our speedometer cable has a kink in it, and wavers between 51 MPH and 59 MPH. Intuitively we estimate that we are traveling about 55 MPH. What have we done? We've developed an internal model of our speed. We don't believe the measurement device (the speedometer) entirely, but we incorporate its input to a certain extent.

That "certain extent" is called the gain. The gain is high if we tend to believe the measurements and low if we tend to believe our internal model.

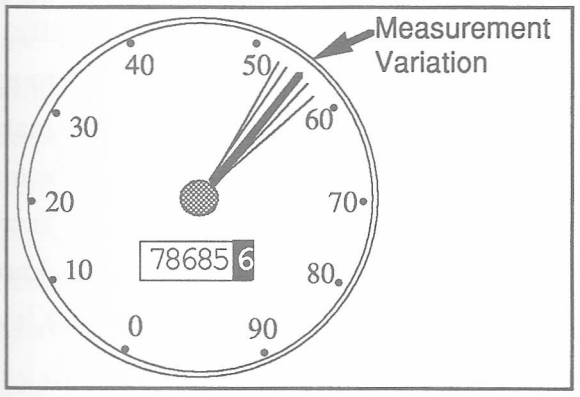

Figure 1. The speedometer cable has a kink. *An an example of a trivial filtering task.*

Now, continuing our speedometer example, suppose that we press on the accelerator. The speedometer eventually wavers between 71 and 79 MPH. We tend to believe the speedometer when it goes higher because we know that it should be higher. That is we have other knowledge in our model: the gain of the measurements increases because we know of the relationship between the accelerator depression and increased speed.

Filters typically are classified as having either fixed or adaptive gain. In a fixed gain filter, we update our internal model by a predetermined percentage of periodic measurements. In an adaptive gain filter we update the model by a varying percentage which correlates with our knowledge of how one variable (example: acceleration) affects another (example: speed). Why is this important? Because the adaptive gain filter will be more accurate during the period when things change.

The above example suggests that the higher levels of the neural cortex can act as an adaptive filter. Is it possible that other biological neural nets, or perhaps even some configurations of artificial neural nets, are adaptive filters?

Other Biological Filters: Cerebellar Hypothesis

Paulin has suggested [1] that the cerebellum acts as a neural analog of a Kalman-Bucy filter. The cerebellum controls gross and fine movement of our motor system. It integrates proprioceptor inputs from the muscles and tendons-- all of which must disagree with each other in both accuracy and phase--to stimulate

muscular performance in a very precise manner. Paulin's assertion is more precise [1 pp 1]:

" ...the cerebellum is directly involved in certain sensory tasks such as predicting trajectories and analyzing the mechanical properties of objects".

Supporting evidence comes from a variety of animals, including the coordination and senses of teleosts (common bony fishes) and frogs. For example, the predation reflexes of some sharks is based on bio-electric phenomena. Frogs, which have a poorly differentiated cerebellum, have an amazingly accurate tongue used to snare prey.

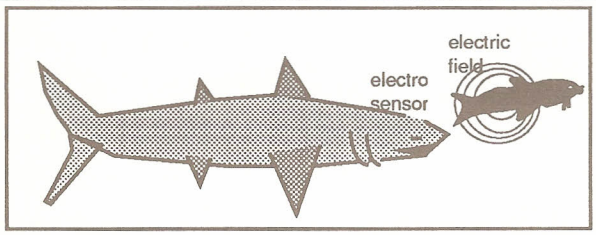

Figure 2. Lower invertebrates use a very coarse electrostatic sense that higher invertebrates lack to sense prey. The coarseness of the sensory apparatus is compensated for in the cerebellum.

The cerebellum is the dominant portion of the brain in these lower invertebrates. The granularity of the sensory input isn't accurate enough to produce the fine motor control that these species exhibit.

Like our speedometer example, the cerebellum must somehow average the inputs to create an answer that is more accurate than the inherent accuracies of the many measurement sensors.

Paulin then shifts to a review of optimal state estimation theory. He introduces the Kalman-Bucy filter, which is a [1 pp 4]

"continuous-time extension of the Kalman Filter, which estimates states of systems governed by difference equations".

Paulin also discusses--using appropriate experimental evidence--on the notion that training allows the cerebellum to create relationships between the different variables that model the equations of motion. The model, then, is the sensory product of cerebellar activity. The experimental evidence suggests that responses changing to stimuli seem to vary in a manner consistent with an adaptive filter.

	position	velocity
position	1	.8
velocity	.8	1

Figure 3. A simple covariance matrix

The key to Paulin's linkage of the Kalman-Bucy Filter to the cerebellar sensor hypothesis is the covariance feature of the filter for producing an adaptive gain. The covariance, or state(s) of awareness, is implemented as a matrix in digital computing as shown in Figure 3 and as intuitively understood in Figure 4. Digitally, the covariance changes from cycle to cycle through a set of equations known as a transition matrix. Biologically, the transition matrix would be made by neurological connections. The current thinking is that a set of nerve extensions known as climbing fibers interconnect at various levels of the cerebellum to provide this function.

I will now discuss another aspect of the filter that modulates the gain: the measurement noise. This is like the wavering speedometer indicator in Figure 4. It is a supplementary technique to the calibration function provided by observation of several related variables.

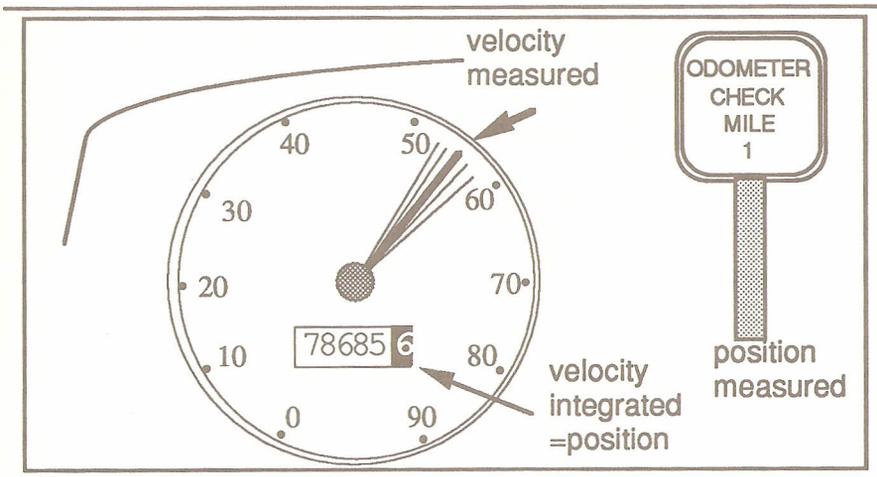

Figure 4. An intuitive understanding of a covariance matrix. *Knowledge about one variable can be used to calibrate the accuracy of another. The transition equation for the above situation is position = initial position plus delta time.*

A Simple Kalman Filter

Keith Brodie [2] designed (and I coded) a simple Kalman Filter that is used to estimate the difference between two observations of the same phenomena. In the application, the phenomena is the pseudo-range from a GPS receiver to a GPS satellite. The psuedo range measurement is different on the L1 and L2 channels of the GPS receiver because the signal suffers a phase delay as it travel through the ionosphere as shown in Figure 5.

A little thought spurs one to conclude that this same process must happen in biological neural systems. For example, two (or more) proprioceptors measure, say, the movement of the tendons controlling my right index finger; the measured difference could indicate the curvature of my right index finger at a particular moment. But each proprioceptor has noise associated with it. And the cumulative accuracy is probably not enough to finely control my finger unless their inputs are averaged somehow.

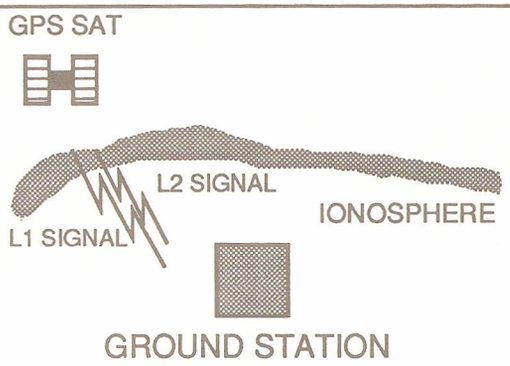

Figure 5. Two measurements of the same thing. *In this case, the L1 and L2 signals traverse the ionosphere. The travel time of these signals depends upon the number and activity of the electrons in the ionosphere. The GPS receiver has a mechanism to determine what noise on the signal is. Knowledge of the noise creates better estimates of truth using filtering techniques.*

The following simple filter illustrates how knowledge of the measurement noise can be used to better estimate the true delay. What would the Kalman Filter look like for this [adapted from 2 pp 13-1,2,3] ?

(1) Initial State: X = S1 - S2 where S1 and S2 are the two measurements, and X is the difference between them.

(2) Initial Covariance: $P = \sigma 1^2 + \sigma 2^2$, where $\sigma 1$ and $\sigma 2$ are the initial measurement noise variances, and P is the covariance (in this case a 1 by 1 matrix)

On succeeding cycles the filter operates with the following five simplified Kalman filter equations:

(3) State Propagation: $X_n+1 = X_n$

(4) Covariance Propagation: $P_n+1 = P_n + Q$

where Q is the unmodelled process noise (the value of Q starts out as a guess by the filter designer and is tuned by subsequent simulations depending upon response time required for the filter). The net affect is that the covariance automatically gets increased every update cycle as a result of a constant Q component that tries to account for random inputs, much like random uncorrelated neuronal spikes.

(5) Kalman Gain Propagation: $K = P_n+1 / P_n+1 + \sigma 1^2 + \sigma 2^2$

where K is the Kalman Gain. Basically this just says that gain is reduced when the measurement noise is known to be high; i.e., the measurement is noisy.

(6) State Update: $X = K(S1 - S2) + (1 - K) X$

(7) Covariance Update: $P_n+1 = (1-K)P_n+1$

Here we have an adaptive filter that can adapt to different levels of noise in the sensor system, including transmission media.

In the more traditional, complex Kalman (as opposed to Kalman-Bucy and the simple filter shown above) the observed states, affect each other as follows:

(8) Covariance Propagation: $P = \phi P \phi^T + Q$

(9) Gain Computation: $K = P H^T/(HPH^T + R)$

(10) State Update: $X = X + K (Z - HX)$

where:
P is the covariance matrix.
ϕ is the transition matrix (unity in the simple filter),
Q is the process noise,
H is the observation matrix (translates filtered state to observed state... it was unity in the simple filter),
R is the measurement noise ($\sigma 1^2 + \sigma 2^2$ in the simple filter) and .
Z is the input (S1 and S2 in the simple filter).

For example, ϕ could define the effect of change of acceleration errors to velocity and position errors and P correlates the "certainty" of each state with every other state. During filter debugging we often look at P to determine what it "thinks" about the certainty of its model, much like the ANS designer looks at the weights to determine the intelligence trained into network.

The multi-variable implementation of ϕ and H were the basis for Paulin's thesis. That is, the transition and observation matrices contain system information, that are programmed into a traditional Kalman filter but are trained via the climbing fibers in the cerebellum.

Overview of Artificial Neural Systems

Artificial neural systems (ANS) are like Kalman filters in that they apply a series of weights, W, to input signals to produce the output signals. Typically ANS are composed of at least three layers, an input layer, a middle, or hidden layer and an output layer. Whereas in a Kalman filter, the intelligence for adaptive weighting present in the covariance, transition and process noise matrixes, the intelligence of an ANS is more or less contained in the hidden layers.

A Kalman filter typically requires an analyst to determine the equations to govern the adaptive weighting. The equations contain "tuning" weights that result from months of simulations performed by the analyst. An ANS is taught, through many examples, how to adaptively weight the inputs based on the overall characteristics of a changing input signal. There are many ways to "teach" an ANS.

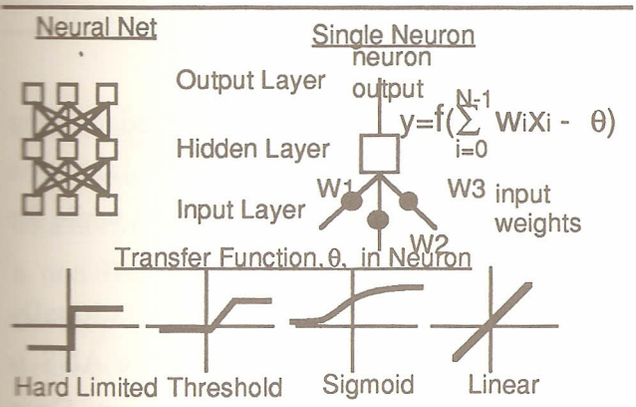

Figure 6. A neural net is composed of a combination of single neurons. *Neurons, or computational elements perform a weighted sum of the inputs. The sum is then passed through a transfer function that is typically non-linear.*

Two training methods often used are backward propagation or backprop [4] and a method similar to the annealing of a piece of metal (called a Boltzman machine). The more common method, backprop, is summarized later in this paper.

Since a neural net can be implemented as a set of parallel units, it has the capability to rapidly characterize a noise environment.

ANS Filtering Strategies

Klimasauskas [3] has summarized an ANS approach to noise filtering where back-propagation is used as an adaptive method for filtering out noise.

In this approach, a three layer ANS consisting of input, hidden, and output layers is organized as an encoder, as shown in Figure 7. The hidden units provide both data compression and feature detectors [3, pp 32]:

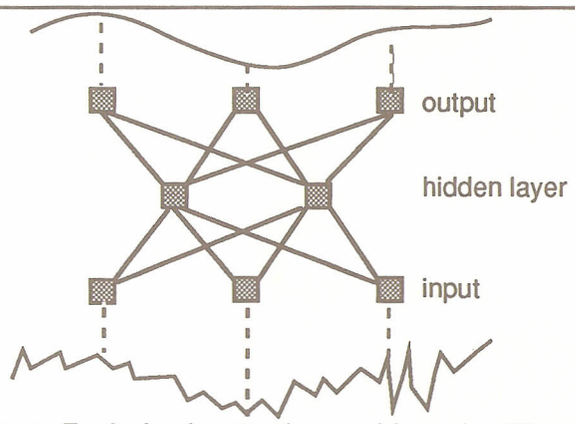

Figure 7. A simple encoder used for noise filtering. *In addtion to the layer to layer connections, feedforward connections that skip layers and feedback connections that connect output to input are sometimes used as well.*

"The compression process eliminates portions of the input data, which represent small or nonrecurring features. By selecting the number of encoders, you can vary the amount of detail retained in the transformation."

Significant to Klimasauskas' discussion is the ability to identify and recreate any frequency of noise (including a low frequency bias and

random non-stationary noise) and subtract it from the input to remove the noise from the signal to create a clean version of "Truth". The signal presented is a rapidly varying signal that is buffered and sampled multiple times to filter the noise. The sidebar by Steve Melnikof in Klimasauskas article describes the weight training scheme. Basically, backprop is implemented as follows:

- the input sequence is used to train the network to adjust the weights to recognize the time structure of the network
- every N value is used as a target value while the other N-1 inputs are used as training inputs
- input/output and input/error measurements are made (similar to what happens when you say a word and you realize that you didn't pronounce it correctly)
- errors at the output of each neuron are used to tweak the input weights on a layer by layer basis backprop; this continues until the input/output measurement error is minimized
- a network simulator (Cognitron™ in this case) is used to test feedback and feedforward connections (called a Jordon network) that could enhance the network's memory of the signal time structure. A network simulator is a key development tool.

KF vs ANS Approach to Noise Filtering

The following differences are apparent in the two filtering schemes, and are summarized.

- Sampling Window: The Kalman Filter treats the most recent inputs with more priority than older inputs, while the ANS approach can potentially use all points within the buffered window for feature recognition. See Figure 8.

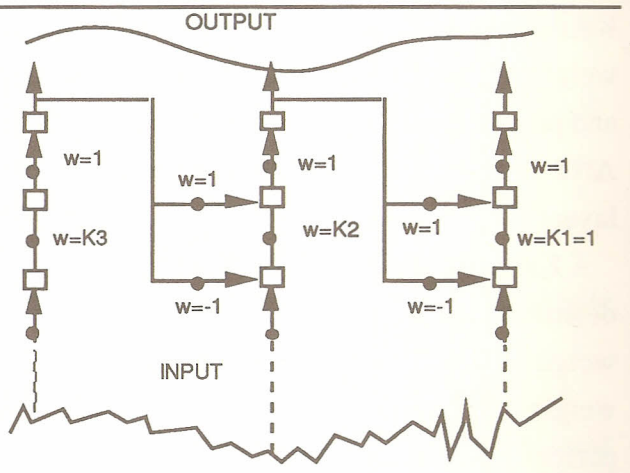

Figure 8. ANS implementation of a simple Kalman filter. *The diagram shows an implementation of the state update formula (10). The network shown is artificially organized with a hidden layer because the gain feature of the Kalman filter is analogous to the feature recognition weights present in the hidden layer of an encoder. In the network, the neurons are Widrow-Hoff, mean the transfer function is linear.*

- A Kalman Filter needs knowledge of the problem programmed into its transition matrix and observation matrices while the ANS gains this by training.
- A Kalman Filter is computationally expensive to execute while the ANS is computationally expensive to train. Once trained, however, a hardware version of an ANS could perform the sophisticated filtering of a Kalman Filter on high rate signals.
- The Kalman Filter is perhaps better suited to the problem of merging several inputs

to create a solution close to the truth. The encoder solution presented by Klimasauskas seemed tied to a single waveform.

- The Kalman Filter is linear (and in fact the bulk of the work in implementing a Kalman Filter is to make the inputs appear linear) while the ANS is non-linear.
- The Kalman Filter can learn-while-doing (to the extent that the P-matrix can change) to a greater extent that the current generation of ANS noise filters which are not easily weight-trained during operation. See Figure 9.

Kalman Filter Weight (Gain) Update:

Covariance Propagation: $P = \phi P \phi^T + Q$

Gain Computation: $K = P H^T / (HPH^T + R) = P/(P+R)$ if $H = 1$

Encoder BackProp Weight Update [4]:

$\Delta W_{ji}(n+1) = \eta(\delta_{pj} o_{pi}) + \alpha \Delta w_{ji}(n)$

where w is the weight

η is the gain associated with learning

o_{pi} is the actual output before presentation to the neuron non-linearity

$\delta_{pi} = (t_{pi} - o_{pj}) o_{pi}(1 - o_{pj})$ for an output unit

$= o_{pj}(1 - o_{pj}) \Sigma \delta_{pk} w_{ki}$ for a hidden unit

$o_{pj} = 1/(1 + e^{-\Sigma w_{ji} o_{pi} + \theta_j})$ o_{pi} after the neuron's non-linearity

$\alpha \Delta w_{ji}(n)$ is the momentum term

Figure 9. Comparison of weight update equations for the Kalman filter and the backprop method of neuron training are very different, yet there are some similarities: the process noise, Q, acts like the momentum term of back-prop.

Other Approaches to Noise Filtering

Priebe and Sung [5] have studied the properties of a network that self-organizes its weights using a non-Backprop method. Their intent was to develop a system that could classify even in the presence of noise, but use the order of presentation as additional knowledge in the training process. In their words:

The temporal classification system produces Gaussian classifications that represent the statistics of the temporal data, and the system uses a learning scheme of moving mean and covariance to update self-developed classes.

The activation value that permits a given classification is a function of the covariance of an input with previous patterns.

Multi-State Neural Network

The ability to merge several distinct inputs streams, or difference several input streams to filter noise appears viable. One approach would be to create local filters for each input variable, like the equations in the Kalman transition matrix that model the state of an individual state.

For example, one could fabricate the simple difference filter described earlier using two

separate simple encoders like those shown in Figure 10 as noise filters and a differencing neuron at their output.

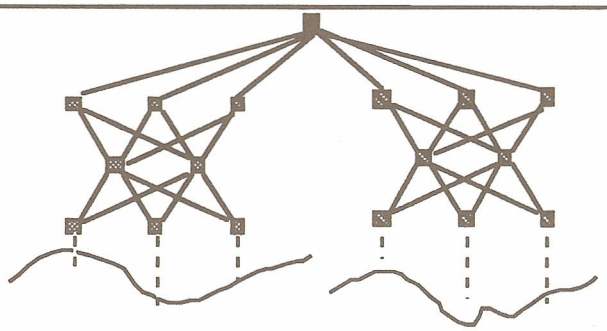

Figure 10. An ANS implementation of a multi-variable differncing filter. This configuration filters noise from each input stream separately.

However, if the two inputs are truly related to each other, and if the noise is common to both, the filtering scheme is not optimum. That is there is knowledge that should couple the two filtering nets. Figure 11 suggests another possibility to merge inputs at the start. In this scheme, the phase of the inputs would be tightly-coupled.

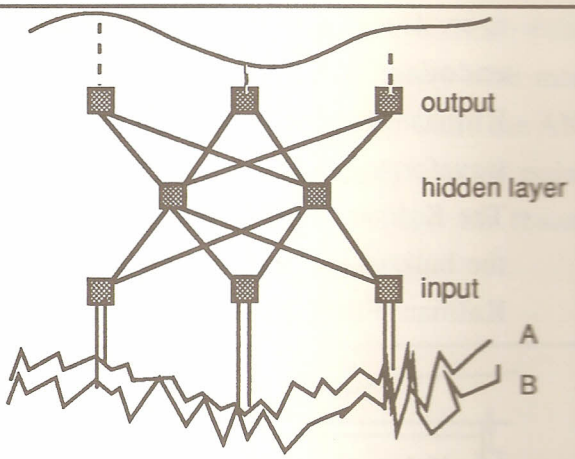

Figure 11. A simple encoder used for noise filtering of multiple input streams, where inputs are merged at the onset.

Conclusions

The network connections and weight updating of a Kalman Filter and Encoder Filter are distinctly different. However both methods encode or are programmed to encode system information, perhaps even temporal information. The correct organization of an encoder to perform a difference filter is unknown, and should be simulated to determine an optimal configuration. However it seems that the function of merging/differencing the inputs of different sensors that represent the same information is a natural ANS implementation.

Bibliography

[1] Paulin, "A Kalman Filter Theory of the Cerebellum", to appear in Competition and Cooperation in Neural Nets II, Springer-Verlag 1988.

[2] Brodie, "Extended GPS Receiver Program Kalman Filter Functional Design Description". Computing Applications Software Technology, 1986.

[3] Klimasauskas, "Neural Nets and Noise Filtering", Dr. Dobbs Journal, Jan. 1989.

[4] Rumelhart, Hinton, and Williams, "Learning Internal Representations By Error Propagation", Parallel Distributed Processing: Explorations in Microstructures of Cognition, Vol. 1. MIT Press pp 318 -362.

[5] Priebe and Sung, "Temporal Pattern Recognition", Technical Document 1331, Naval Ocean Systems Center, September 1988.

Other Recommended Readings

Lippmann, "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, April 1987.

Anderson an Rosenfeld, "Neurocomputing, Foundations of Research", MIT Press 1988.