scope: *control* **(p)**, *sensor interpretation* **(s)**; contribution: *principles*

Adaptive fuzzy systems for target tracking

by Peter J. Pacini and Bart Kosko

In this paper, we compared fuzzy and Kalman-filter control systems for real-time target tracking. Both systems performed well in the presence of additive measurement noise. In the presence of mild process (unmodelled-effects) noise, the fuzzy system exhibited finer control. We tested the robustness of the fuzzy controller by removing random subsets of fuzzy associations or 'rules', and by adding destructive or 'sabotage' fuzzy rules to the fuzzy system. We tested the robustness of the Kalman tracking system by increasing the variance of the unmodelled-effects noise process. The fuzzy controller performed well until we removed over 50% of the fuzzy rules. The Kalman controller's performance quickly depreciated as the unmodelled-effects variance increased. We used unsupervised neural-network learning to adaptively generate the fuzzy controller's fuzzy-associative-memory structure. The fuzzy systems did not require a mathematical model of how system outputs depended on inputs.

1 Fuzzy and maths-model controllers

Fuzzy controllers differ from classical maths-model controllers. Fuzzy controllers do not require a mathematical model of how control outputs functionally depend on control inputs. Fuzzy controllers also differ in the type of uncertainty they represent and how they represent it. The fuzzy approach represents ambiguous or fuzzy system behaviour as partial implications and approximate 'rules of thumb'—as fuzzy associations (A_i, B_i).

Fuzzy controllers are fuzzy systems. A finite *fuzzy set* A is a *point* [1] in a unit hypercube $I^n = [0, 1]^n$. A *fuzzy system* F: $I^n \rightarrow I^p$ is a *mapping* between unit hypercubes. I^n contains all fuzzy subsets of the domain space $X = \{x_1, \ldots, x_n\}$. I^n is the *fuzzy power set* $F(2^X)$ of X. I^p contains all the fuzzy subsets of the range space $Y = \{y_1, \ldots, y_p\}$. Element $x_i \in X$ belongs to fuzzy set A to degree $m_A(x_i)$. The 2^n non-fuzzy subsets of X correspond to the 2^n corners of the fuzzy cube I^n. The fuzzy system F maps fuzzy subsets of X to fuzzy subsets of Y. In general, X and Y are continuous, rather than discrete, sets.

Maths-model controllers usually represent system uncertainty with probability distributions. The most common probability model, the linear Gaussian model, describes system behaviour with first-order and second-order statistics, by conditional means and covariances. It describes unmodelled effects and measurement imperfections with additive 'noise' processes.

Mathematical models of the system state and measurement processes facilitate a mean-squared-error analysis of system behaviour. In general, we cannot accurately describe such mathematical models. This greatly restricts the range of real-world applications. In practice, we often use linear or quasi-linear (Markov) mathematical models.

Mathematical state and measurement models also make it difficult to add non-mathematical knowledge to the system. Experts may pronounce such knowledge, or neural networks may adaptively infer it from sample data. In practice, once we have stated the maths model, we use human expertise only to estimate the initial state and covariance conditions.

Fuzzy controllers consist of a bank of *fuzzy-associative-memory* (FAM) 'rules' or associations (A_i, B_i), operating in parallel and operating to different degrees. Each FAM rule is a set-level implication. It represents ambiguous expert knowledge or learned input/output transformations. A FAM rule can also summarise the behaviour of a specific mathematical model. The system non-linearly transforms exact or fuzzy state inputs to a fuzzy-set output. This output fuzzy set is usually

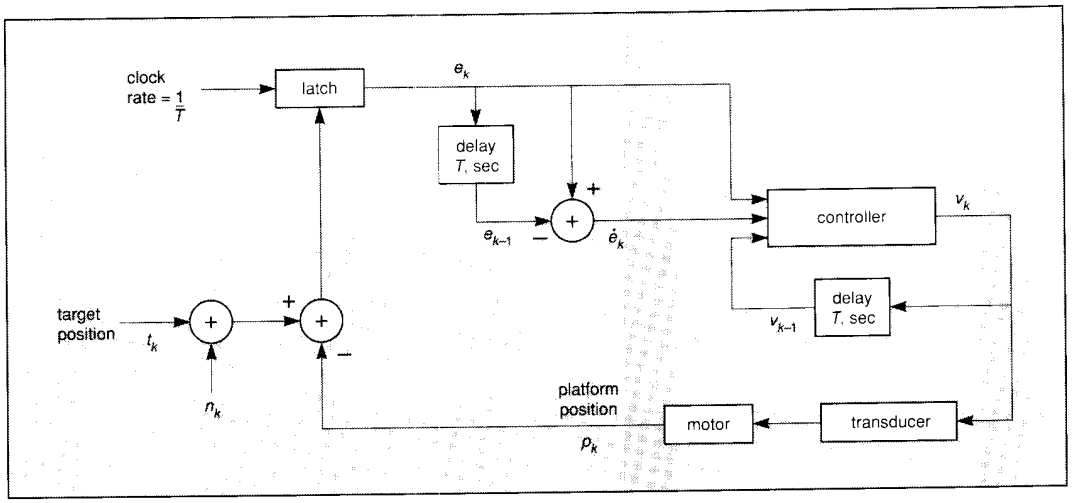

Fig. 1 Target-tracking system

'defuzzified' with a centroid operation to generate an exact numerical output. In principle, the system can use the entire fuzzy distribution as the output. We can easily construct, process and modify the FAM bank of FAM rules in software or in digital VLSI circuitry.

Fuzzy controllers require that we articulate or estimate the FAM rules. The fuzzy-set framework provides more expressiveness than, for example, traditional expert-system approaches, which encode bivalent propositional associations. However, the fuzzy framework does not eliminate the burden of knowledge acquisition. We can use neural network systems to estimate the FAM rules. Nevertheless, neural systems also require an accurate (statistically representative) set of articulated input–output numerical samples. Below we use unsupervised competitive learning to adaptively generate target-tracking FAM rules.

Experts can 'hedge' their system descriptions with fuzzy concepts. Although fuzzy controllers are numerical systems, experts can contribute their knowledge in natural language. This is especially important in complex problem domains, such as economics, medicine and history, where we may not know how to mathematically model system behaviour.

Below we compare a fuzzy controller with a Kalman-filter controller for real-time target tracking. This problem admits a simple and reasonably accurate mathematical description of its state and measurement processes. We chose the Kalman filter as a benchmark because of its many optimal linear-systems properties. We wanted to see whether this 'optimal' controller remains optimal when compared with a fuzzy controller in different uncertainty environments.

We indirectly compared the sensitivity of the two controllers by varying their system uncertainties. We randomly removed FAM rules from the fuzzy controller. We also added 'sabotage' FAM rules to the controller. Both techniques modelled less structured control environments. For the Kalman filter, we varied the noise variance of the unmodelled-effects noise process.

Both systems performed well for mildly uncertain target environments. They depreciated differently as the system uncertainty increased. The fuzzy controller's performance depreciated when we removed more than

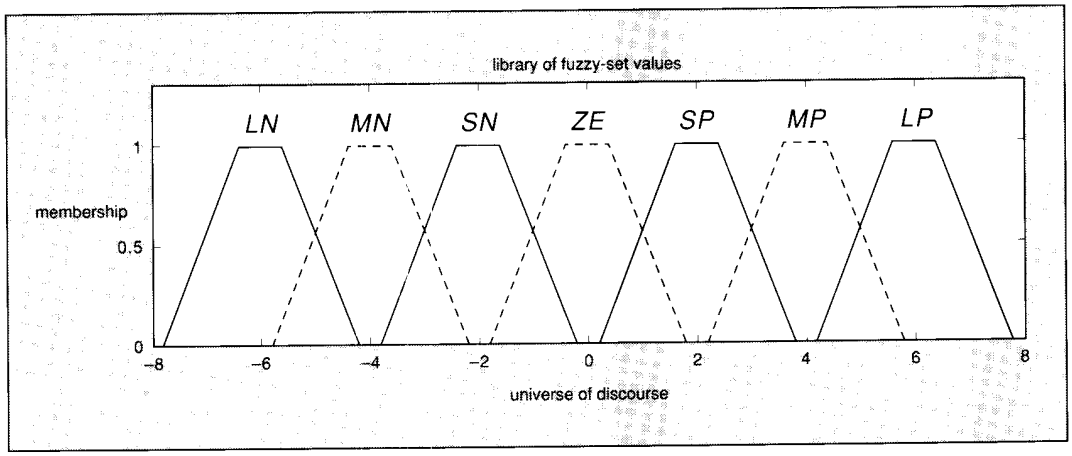

Fig. 2 Library of overlapping fuzzy-set values defined on a universe of discourse

half of the FAM rules. The Kalman-filter controller's performance quickly depreciated when the additive state noise process increased in variance.

2 Real-time target tracking

A target-tracking system maps azimuth-elevation inputs to motor control outputs. The nominal target moves through azimuth-elevation space. Two motors adjust the position of a platform to continuously point towards the target.

The platform can be any directional device that accurately points towards the target. The device may be a laser, video camera or high-gain antenna. We assume we have available a radar or other device that can detect the direction from the platform to the target, with an uncertainty modelled by an additive white Gaussian noise process n_k. n_k has zero mean and variance $E[n_k^2] = \sigma_n^2$.

The radar sends azimuth and elevation co-ordinates to the tracking system at the end of each time interval. We calculate the current *error* e_k in platform position and the *change in error* $\dot{e}_k$. A fuzzy or Kalman-filter controller then determines the control outputs for the motors, one each for azimuth and elevation. The control outputs reposition the platform.

We can independently control movement along the azimuth and elevation if we apply the same algorithm twice. This reduces the problem to that of matching the target's position and velocity in only one dimension.

Fig. 1 shows a block diagram of the target-tracking system. The controller output v_k gives the estimated angular velocity required during the next time interval. In principle, a hardware system must convert the angular velocity v_k into a voltage or current. For a controller output v_k, the platform turns $T(v_k + y_k)^\circ$, where T denotes the sampling interval in seconds, and v_k and y_k have units of degrees/second. We model the uncertainty y_k as white Gaussian noise with zero mean and variance $E[y_k^2] = \sigma_y^2$.

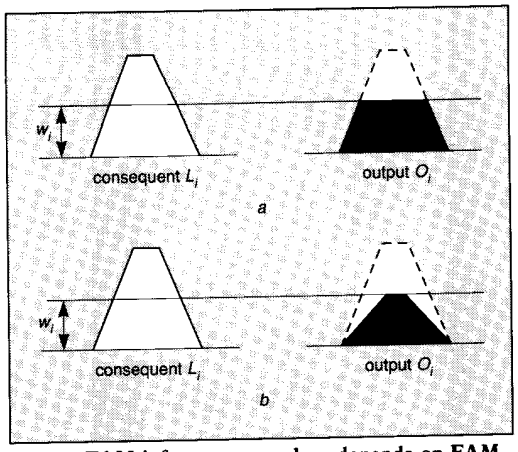

Fig. 3 FAM inference procedure depends on FAM rule encoding procedure

a Correlation-minimum encoding *b* Correlation-product encoding

3 Fuzzy controller

We restrict the output angular velocity v_k of the fuzzy controller to the interval $[-6, 6]$. Thus, we must insert a gain element before the voltage transduction. This gain must equal one-sixth of the platform's maximum angular velocity. The simulation used output gains of 1.5/T azimuth and 0.75/T elevation. Thus, the output angular velocity obeyed

$$|v_k| \leq \frac{9.0}{T} \text{ degrees/second azimuth}$$

$$|v_k| \leq \frac{4.5}{T} \text{ degrees/second elevation}$$

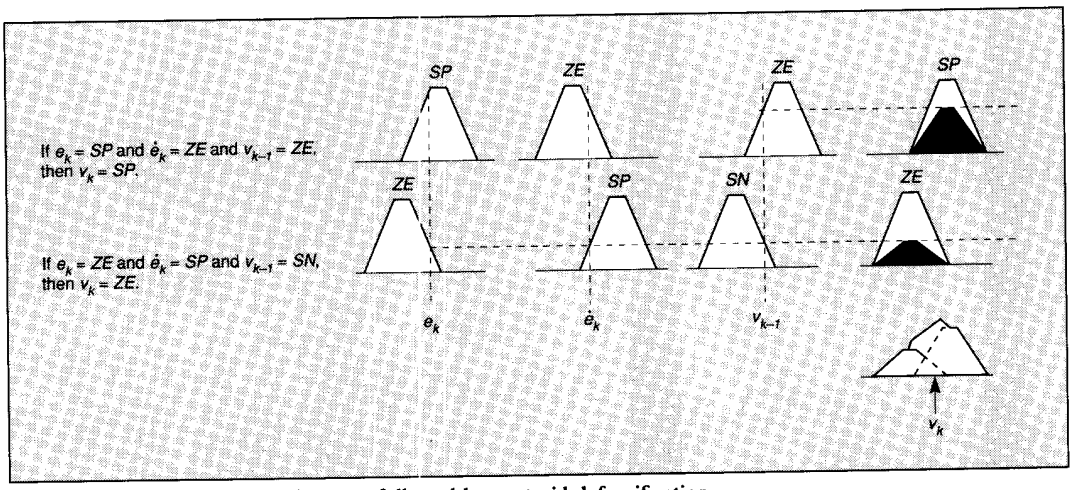

Fig. 4 Correlation-product inferences followed by centroid defuzzification

FAM rule antecedents combined with AND use the *minimum* fit value to activate consequents. Those combined with OR use the *maximum* fit value.

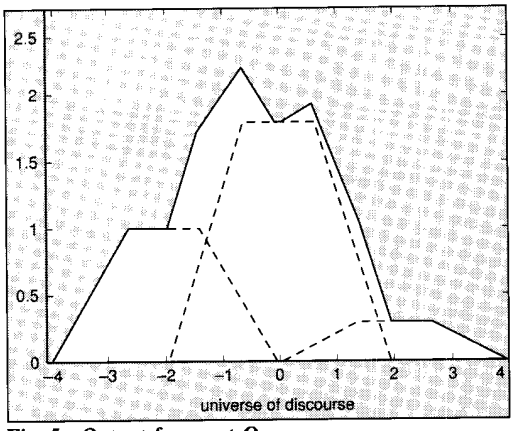

Fig. 5 Output fuzzy set *O*

Similarly, the position error e_k must be scaled so that the maximum error maps to a scaled value of 6. We scaled and limited/tested all error values outside the range $[-6, 6]$. The product of this error scale factor, the output gain and the sampling interval provides a design parameter, the 'gain' of the fuzzy controller. It equals the ratio of maximum platform angle in one time interval to the maximum threshold error. In the simulation, the maximum errors were 10° azimuth and 5° elevation, yielding a gain of 0.9.

The fuzzy controller uses heuristic control set-level 'rules' or *fuzzy-associative-memory* (FAM) associations, based on quantised values of e_k, $\dot{e}_k$ and v_{k-1}. We define seven fuzzy levels by the following library of fuzzy-set values of the fuzzy variables e_k, $\dot{e}_k$, and v_{k-1}:

- LN = Large Negative
- MN = Medium Negative
- SN = Small Negative
- ZE = Zero
- SP = Small Positive
- MP = Medium Positive
- LP = Large Positive

We do not quantise inputs in the classical sense of assigning each input to exactly one output level. Instead, each linguistic value equals a fuzzy set that overlaps with adjacent fuzzy sets. The fuzzy controller uses trapezoidal fuzzy-set values, as shown in Fig. 2. The lengths of the upper and lower bases provide design parameters that we must calibrate for satisfactory performance. A good rule of thumb is that *adjacent fuzzy-set values should overlap by approximately 25%*. Below we discuss examples of calibrated and uncalibrated systems. The fuzzy controller attained its best performance with upper and lower bases of 0.8 and 3.6, respectively, −20.8% overlap. Different target scenarios may require more or less overlap.

We assign each system input to a fit vector of length 7, where the *i*th *fit* or *fuzzy unit* [2] equals the value of the *i*th fuzzy set at the input value. In other words, the *i*th fit measures the degree to which the input belongs to the *i*th fuzzy-set value. For instance, we apply the input values 1, −4 and 3.8 to the seven fuzzy sets in the library to obtain the fit vectors

$$\mathbf{1} \rightarrow (\mathbf{0}\ \ \mathbf{0}\ \ \mathbf{0}\ \ \mathbf{0.7}\ \ \mathbf{0.7}\ \ \mathbf{0}\ \ \mathbf{0})$$

$$\mathbf{-4} \rightarrow (\mathbf{0}\ \ \mathbf{1}\ \ \mathbf{0}\ \ \mathbf{0}\ \ \mathbf{0}\ \ \mathbf{0}\ \ \mathbf{0})$$

$$\mathbf{3.8} \rightarrow (\mathbf{0}\ \ \mathbf{0}\ \ \mathbf{0}\ \ \mathbf{0}\ \ \mathbf{0.1}\ \ \mathbf{1}\ \ \mathbf{0})$$

We determine the above fit values by convolving a Dirac delta function centered at the input value with each of the seven fuzzy sets.

$$m_{SP}(3.8) = \delta(y - 3.8) * m_{SP}(y) = 0.1 \qquad (1)$$

If we use a discrete universe of discourse, we use a Kronecker delta function instead. Equivalently, for the discrete case n-dimensional universe of discourse $X = \{x_1, \ldots, x_n\}$, a control input corresponds to a *bit*

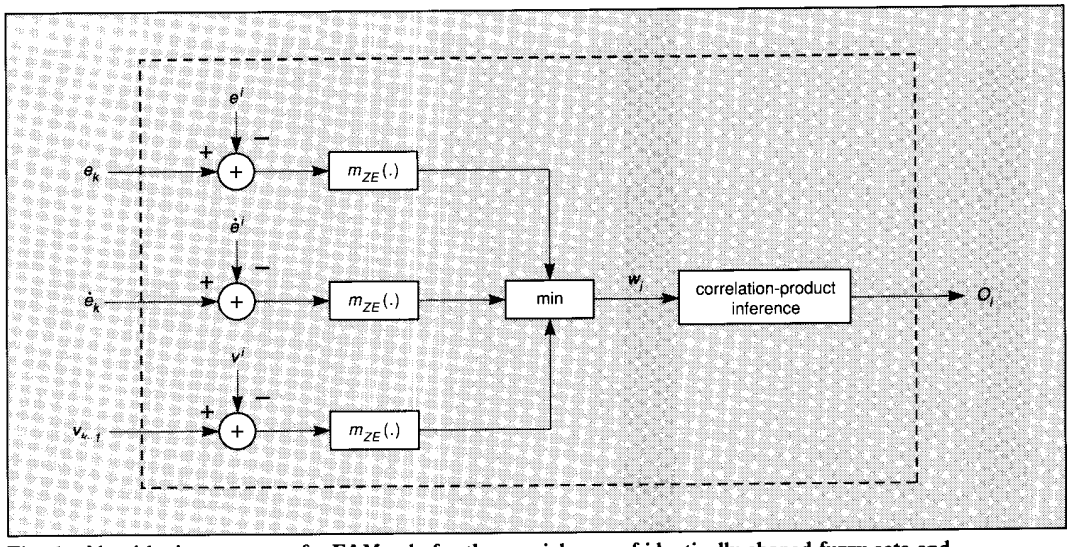

Fig. 6 Algorithmic structure of a FAM rule for the special case of identically shaped fuzzy sets and correlation-product inference

(binary unit) vector B of length n. A single 1 element in the ith slot represents the 'crisp' input value x_i. Similarly, we represent the kth library fuzzy set by an n-dimensional fit vector A_k, which contains samples of the fuzzy set at the n discrete points within the universe of discourse X. The degree to which the crisp input x_i activates each fuzzy set equals the inner product $B \cdot A_k$ of the bit vector B and the corresponding fit vector A_k.

We formulate control FAM rules by associating output fuzzy sets with input fuzzy sets. The antecedent of each FAM rule conjoins e_k, $\dot{e}_k$, and v_{k-1} fuzzy-set values. For example,

IF $e_k = MP$ AND $\dot{e}_k = SN$ AND $v_{k-1} = ZE$, THEN $v_k = SP$

We abbreviate this to $(MP, SN, ZE; SP)$.

The scalar activation value w_i of the ith FAM rule consequent equals the *minimum* of the three antecedent conjuncts' values. If, alternatively, we combine the antecedents disjunctively with OR, the activation degree of the consequent would equal the *maximum* of the three antecedent disjuncts' values. In the following example, $m_A(e_k)$ denotes the degree to which e_k belongs to the fuzzy set A:

		LN	MN	SN	ZE	SP	MP	LP
$e_k = 2.6$	$\rightarrow$	(0	0	0	0	1	0.4	0)
$e_k = -2.0$	$\rightarrow$	(0	0	1	0	0	0	0)
$v_{k-1} = 1.8$	$\rightarrow$	(0	0	0	0.1	1	0	0)

$m_{MP}(e_k) = 0.4$

$m_{SN}(\dot{e}_k) = 1$

$m_{ZE}(v_{k-1}) = 0.1$

$w_i = \min(0.4, 1, 0.1) = 0.1$

Thus, the system activates the consequent fuzzy set SP to the w_i degree $= 0.1$.

The output fuzzy set's shape depends on the FAM-rule encoding scheme used. With *correlation-minimum* encoding, we clip the consequent fuzzy set L_i in the library of fuzzy-set values to the w_i degree with pointwise minimum

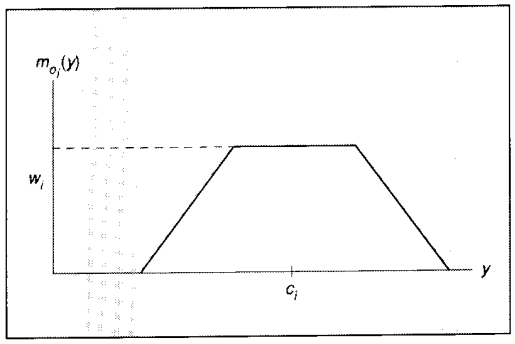

Fig. 7 Trapezoidal output fuzzy set O_i

$$m_{O_i}(y) = \min(w_i, m_{L_i}(y)) \tag{2}$$

With *correlation-product* encoding, we multiply L_i by w_i

$$m_{O_i}(y) = w_i m_{L_i}(y) \tag{3}$$

or equivalently

$$O_i = w_i L_i \tag{4}$$

Fig. 3 illustrates how both inference procedures transform L_i to scaled output O_i. For the example above, correlation-product inference gives output fuzzy set $O_i = 0.1SP$, where $L_i = SP$ denotes the fuzzy set of small but positive angular velocity values.

The fuzzy system activates each FAM rule consequent set to a different degree. For the ith FAM rule, this yields the output fuzzy set O_i. The system then sums O_i to form the combined output fuzzy set O

$$O = \sum_{i=1}^{N} O_i \tag{5}$$

or equivalently

$$m_O(y) = \sum_{i=1}^{N} m_{O_i}(y) \tag{6}$$

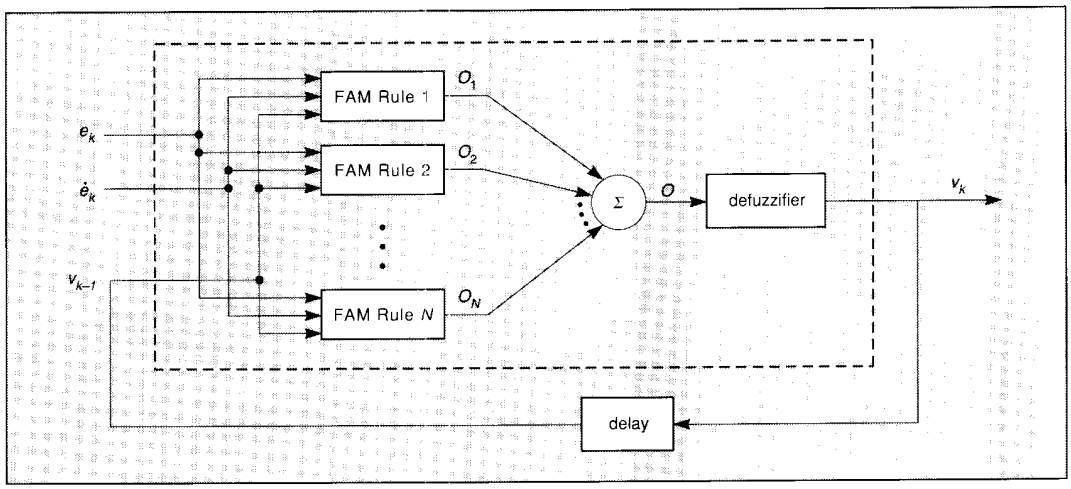

Fig. 8 Fuzzy control system as a parallel FAM bank with centroidal output

$\dot{e}_k$ \ v_{k-1}	LN	MN	SN	ZE	SP	MP	LP
LN	LN	LN	LN	LN	MN	SN	ZE
MN	LN	LN	LN	MN	SN	ZE	SP
SN	LN	LN	MN	SN	ZE	SP	MP
ZE	LN	MN	SN	ZE	SP	MP	LP
SP	MN	SN	ZE	SP	MP	LP	LP
MP	SN	ZE	SP	MP	LP	LP	LP
LP	ZE	SP	MP	LP	LP	LP	LP

Fig. 9 $e_k = ZE$ cross-section of the fuzzy control system's FAM bank

Each entry represents one FAM rule with $e_k = ZE$ as the first antecedent term. The shaded FAM rule is 'IF $e_k = ZE$ AND $\dot{e}_k = SP$ AND $v_{k-1} = SN$, THEN $v_k = ZE$', abbreviated as $(ZE, SP, SN; ZE)$. Note the ordinal anti-symmetry of this FAM-bank matrix. The six other cross-section FAM-bank matrices are similar. We can eliminate many FAM rule entries without greatly perturbing the fuzzy controller's behaviour.

The control output v_k equals the *fuzzy centroid* of O

$$v_k = \frac{\int y\, m_O(y) dy}{\int m_O(y) dy} \tag{7}$$

where the limits of integration correspond to the entire universe of discourse Y of angular velocity values. Fig. 4 shows an example of correlation-product inference for two FAM rules followed by centroid defuzzification of the combined output fuzzy set.

To reduce computations, we can discretise the output universe of discourse Y to p values, $Y = \{y_1, \ldots, y_p\}$, which gives the discrete fuzzy centroid

$$v_k = \frac{\sum_{j=1}^{p} y_j m_O(y_j)}{\sum_{j=1}^{p} m_O(y_j)} \tag{8}$$

3.1 *Fuzzy centroid computation*

We now develop two discrete methods for computing the fuzzy centroid (eqn. 7). Theorem 1 states that we can compute the global centroid v_k from local FAM-rule centroids. Theorem 2 states that v_k can be computed from only seven sample points if all the fuzzy sets are symmetric and unimodal (in the broad sense of a trapezoid peak), although otherwise arbitrary. Both results reduce computation and favour digital implementation.

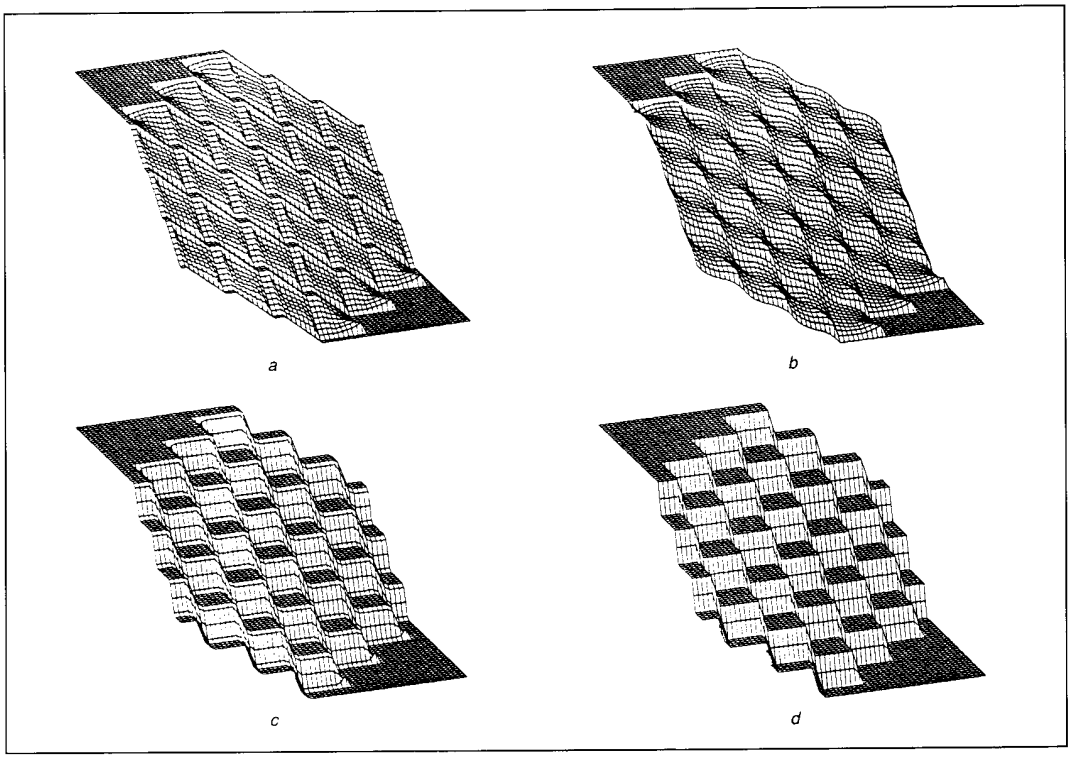

Fig. 10 Control surfaces of the fuzzy controller for constant error $e_k = 0$ and different amounts of overlap

a 20.8% *c* 6.3%
b 33.3% *d* None
We plotted the control output v_k against $\dot{e}_k$ and v_{k-1} along the west and south borders, respectively.

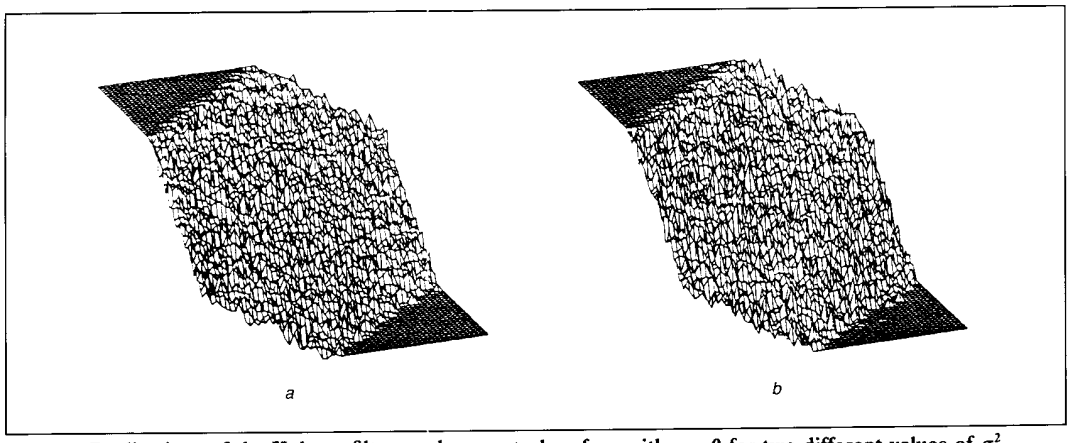

Fig. 11 Realisations of the Kalman-filter random control surface with $e_k = 0$ for two different values of σ_a^2, the variance of the unmodelled-effects noise process

a $\sigma_a^2 = 0$ *b* $\sigma_a^2 = 4.0$

Theorem 1

If correlation-product inference determines the output fuzzy sets, we can compute the global centroid v_k from local FAM-rule centroids

$$v_k = \frac{\sum_{i=1}^{N} w_i c_i I_i}{\sum_{i=1}^{N} w_i I_i} \tag{9}$$

Proof

The consequent fuzzy set of each FAM rule equals one of the fuzzy-set values shown in Fig. 2. We assume each fuzzy set includes at least one unity value, $m_A(x) = 1$. Define I_i and c_i as the area and centroid of the ith FAM rule's consequent set L_i, respectively.

$$I_i = \int m_{L_i}(y)dy \tag{10}$$

$$c_i = \frac{\int y\, m_{L_i}(y)dy}{\int m_{L_i}(y)dy} = \frac{\int y\, m_{L_i}(y)dy}{I_i}$$

substituting from eqn. 10. Hence

$$\int y\, m_{L_i}(y)dy = c_i I_i \tag{11}$$

Using eqn. 3 (the results of correlation-product inference), we get

$$\int y\, m_{O_i}(y)dy = \int y\, w_i m_{L_i}(y)dy = w_i \int y\, m_{L_i}(y)dy = w_i c_i I_i \tag{12}$$

substituting from eqn. 11. Similarly

$$\int m_{O_i}(y)dy = \int w_i m_{L_i}(y)dy = w_i I_i \tag{13}$$

substituting from eqn. 10.

We can use eqns. 12 and 13 to derive a discrete expression equivalent to eqn. 7

$$\int y\, m_O(y)dy = \int y\left[\sum_{i=1}^{N} m_{O_i}(y)\right]dy$$

substituting from eqn. 6

$$= \sum_i \int y\, m_{O_i}(y)dy = \sum_i w_i c_i I_i \tag{14}$$

from eqn. 12. Similarly

$$\int m_O(y)\, dy = \int \sum_{i=1}^{N} m_{O_i}(y)dy = \sum_i \int m_{O_i}(y)dy = \sum_i w_i I_i \tag{15}$$

from eqn. 13. Substituting eqns. 14 and 15 into eqn. 7, we derive a new form for the centroid

$$v_k = \frac{\sum_{i=1}^{N} w_i c_i I_i}{\sum_{i=1}^{N} w_i I_i} \tag{16}$$

which is equivalent to eqn. 9. Each summand in each summation of eqn. 16 depends on only a single FAM

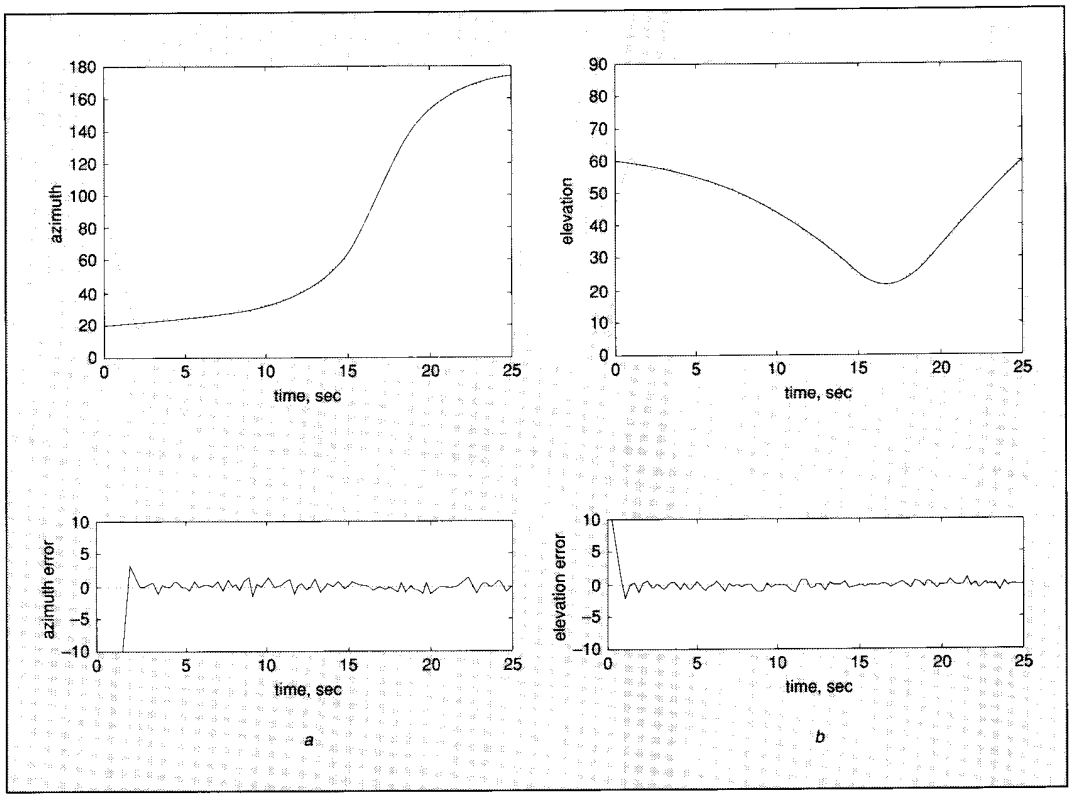

Fig. 12 Best performance of the fuzzy controller

a azimuth position and error *b* elevation position and error
Fuzzy set overlap = 20.8%.

rule. Thus, we can compute the global output centroid from local FAM-rule centroids. QED.

Theorem 2

If the seven library fuzzy sets are symmetric and unimodal (in the trapezoidal sense) and we use correlation-product inference, we can compute the centroid v_k from only seven samples of the combined output fuzzy set O

$$v_k = \frac{\sum_{j=1}^{7} m_O(y_j) y_j \mathcal{J}_j}{\sum_{j=1}^{7} m_O(y_j) \mathcal{J}_j} \tag{17}$$

The seven sample points are the centroids of the output fuzzy-set values.

Proof

Define $\bar{O}_i$ as a fit vector of length 7, where the fit value corresponding to the ith consequent set has the value w_i and the other entries equal zero. If all the fuzzy sets are symmetric and unimodal, the jth fit value of $\bar{O}_i$ is a sample of m_{O_i} at the centroid of the jth fuzzy set. The combined output fit vector is

$$\bar{O} = \sum_{i=1}^{N} \bar{O}_i \tag{18}$$

Since

$$m_O(y) = \sum_{i=1}^{N} m_{O_i}(y)$$

the jth fit value of $\bar{O}$ is a sample of m_O at the centroid of the jth fuzzy set. Equivalently, the jth fit value of $\bar{O}$ equals the sum of the output activations w_i from the FAM rules with consequent fuzzy sets equal to the jth library fuzzy-set value.

Define the reduced universe of discourse as $Y = \{y_1, \ldots, y_7\}$, such that y_j equals the centroid of the jth output fuzzy set. In vector form

$$Y = (y_1, \ldots, y_7)$$
$$= (-6, -4, -2, 0, 2, 4, 6)$$

for the library of fuzzy sets in Fig. 2. Also define the diagonal matrix

$$\mathcal{J} = \text{diag}(\mathcal{J}_1, \ldots, \mathcal{J}_7) \tag{19}$$

where $\mathcal{J}_j$ denotes the area of the jth fuzzy-set value. If the ith FAM rule's consequent fuzzy set equals the jth fuzzy-set value, the jth fit value of $\bar{O}$ increases by w_i, $c_i = y_j$ and $I_i = \mathcal{J}_j$. Thus

$$\bar{O}\mathcal{J}Y^T = \sum_{j=1}^{7} m_O(y_j) y_j \mathcal{J}_j = \sum_{i=1}^{N} w_i c_i I_i \tag{20}$$

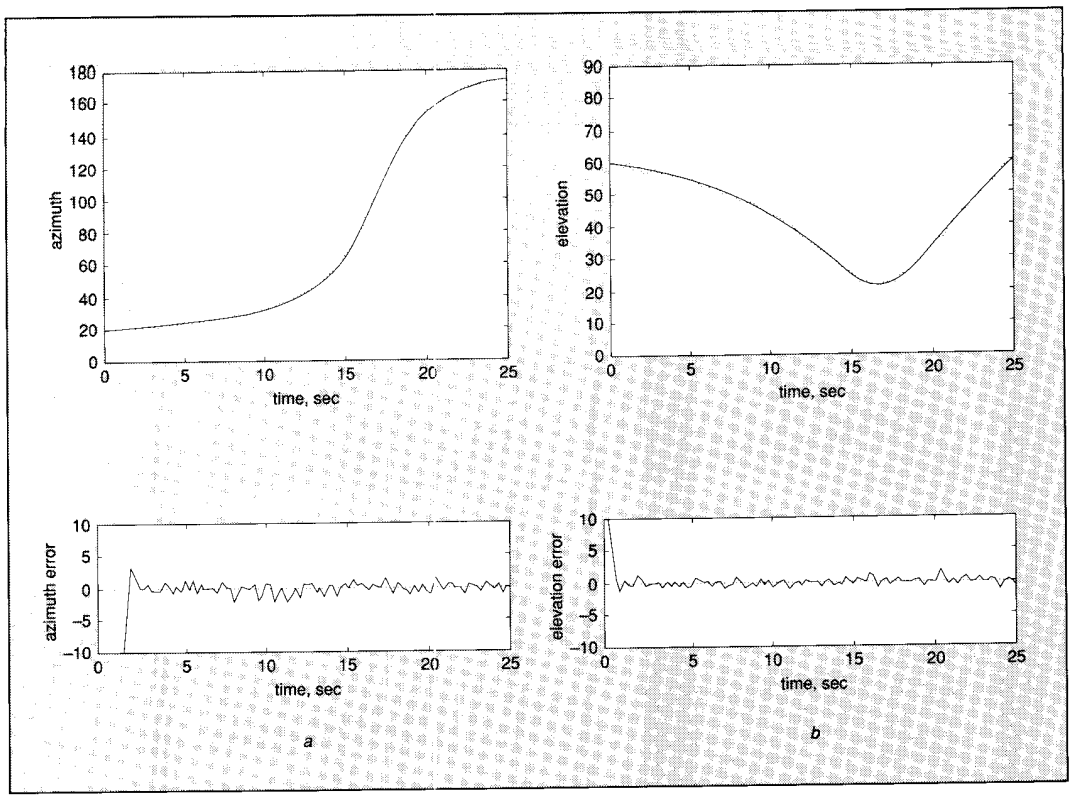

Fig. 13 Uncalibrated fuzzy controller

a azimuth position and error *b* elevation position and error
Fuzzy-set overlap = 33.3%. Too much overlap causes excessive overshoot.

Also

$$\bar{O}\mathcal{J}\mathbf{1}^T = \sum_{j=1}^{7} m_O(y_j)\mathcal{J}_j = \sum_{i=1}^{N} w_i I_i \tag{21}$$

where $\mathbf{1} = (1, \ldots, 1)$. Substituting eqns. 20 and 21 into eqn. 16 gives

$$v_k = \frac{\Sigma_{j=1}^{7} m_O(y_j) y_j \mathcal{J}_j}{\Sigma_{j=1}^{7} m_O(y_j)\mathcal{J}_j} \tag{22}$$

which is equivalent to eqn. 17. Therefore, eqn. 22 gives a simpler, but equivalent form of the centroid (eqn. 7) if all the fuzzy sets are symmetric and unimodal, and if we use correlation-product inference to form the output fuzzy sets O_i. QED.

Consider a fuzzy controller with the fuzzy sets defined in Fig. 2 and seven FAM rules with the following output

i	w_i	consequent
1	0.0	*MP*
2	0.2	*SP*
3	1.0	*ZE*
4	0.4	*SN*
5	0.1	*SP*
6	0.8	*ZE*
7	0.6	*SN*

Fig. 5 shows the combined output fuzzy set O, with the *SN*, *ZE* and *SP* components shown by dotted lines. Using eqn. 7, we get a velocity output of -0.452. Alternatively, the combined output fit vector O equals (0, 0, 1.0, 1.8, 0.3, 0, 0). From eqn. 22 we get

$$v_k = \frac{-2 \times 1 + 0 \times 1.8 + 2 \times 0.3}{1 + 1.8 + 0.3} = -0.452.$$

3.2 *Fuzzy controller implementation*

A FAM bank or 'rulebase' of FAM rules defines the fuzzy controller. Each FAM rule associates one consequent fuzzy set with three antecedent fuzzy-set conjuncts.

Suppose the ith FAM rule is (*MP*, *SN*, *ZE*; *SP*). Suppose the inputs at time k are $e_k = 2.6$, $\dot{e}_k = -2.0$ and $v_{k-1} = 1.8$. Then

$$\begin{aligned} w_i &= \min(m_{MP}(e_k), m_{SN}(\dot{e}_k), m_{ZE}(v_{k-1})) \\ &= \min(0.4, 1, 0.1) \\ &= 0.1 \end{aligned}$$

If all the fuzzy sets have the same shape, they correspond to shifted versions of a single fuzzy set *ZE*

$$m_{SP}(y) = m_{ZE}(y - 2)$$

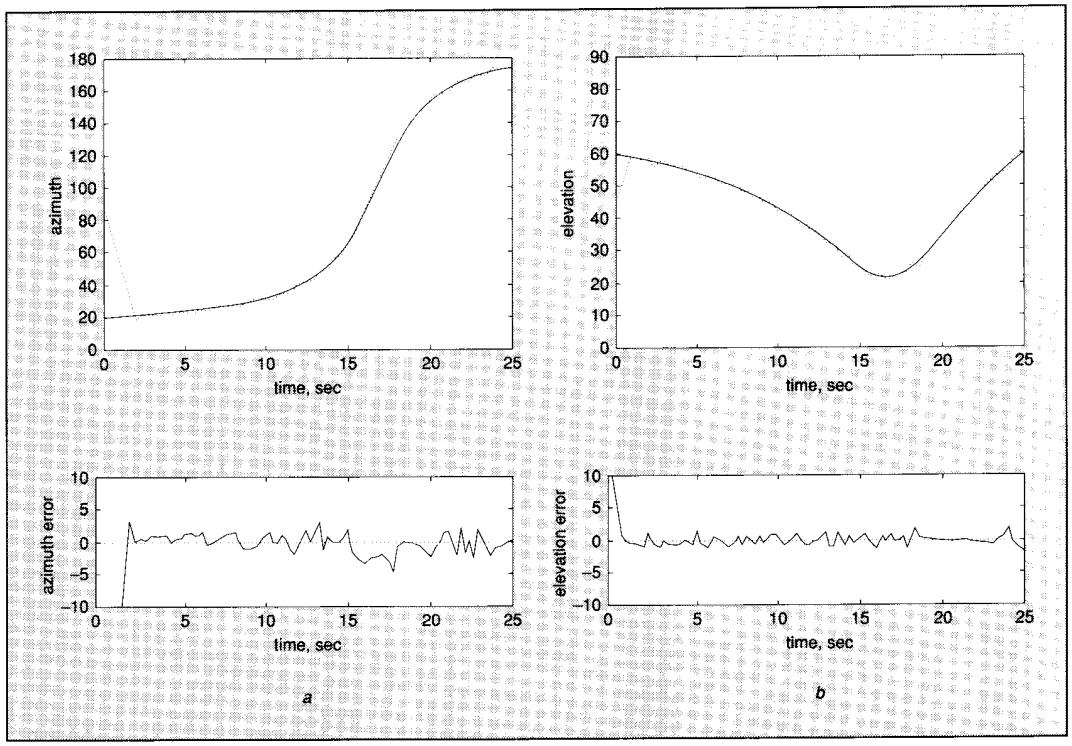

Fig. 14 Uncalibrated fuzzy controller

a azimuth position and error *b* elevation position and error
Fuzzy-set overlap = 6.3%. Too little overlap causes lead or lag for several consecutive time intervals.

Define e^i, $\dot{e}^i$, and v^i as the centroids of the corresponding antecedent fuzzy sets in the example above. Thus, $e^i = 4$, $\dot{e}^i = -2$ and $v^i = 0$. The output activation then equals

$$\begin{aligned} w_i &= \min(m_{ZE}(e_k - e^i),\ m_{ZE}(\dot{e}_k - \dot{e}^i),\ m_{ZE}(v_{k-1} - v^i)) \\ &= \min(m_{ZE}(-1.4),\ m_{ZE}(0),\ m_{ZE}(1.8)) \\ &= \min(0.4,\ 1,\ 0.1) \\ &= 0.1 \end{aligned}$$

as computed above. Fig. 6 schematises such a FAM rule when presented with crisp inputs.

The output fuzzy set O_i in Fig. 6 equals the fuzzy set ZE scaled by w_i and shifted by c_i

$$m_{O_i}(y) = w_i m_{ZE}(y - c_i) \tag{23}$$

Fig. 7 illustrates O_i.

The fuzzy control system activates a bank of FAM rules operated in parallel, as shown in Fig. 8. The system sums the output fuzzy sets to form the total output set O, which the system converts to a 'defuzzified' scalar output by computing its fuzzy centroid.

The structure in Fig. 8 describes a wide array of fuzzy controllers. As the problem dimensionality increases, computational delays for matrix-based maths-model controllers increase rapidly. In contrast, the parallel FAM bank's computational delay increases more slowly, because the system processes all rules in parallel. The defuzzifier delay does not depend on the number of rules (or the number of inputs). Fig. 6 shows that increasing the number of inputs only slightly increases the FAM rule delay, because all membership values feed into a *min* operation. Thus, for large FAM banks, the summation node in Fig. 8 represents the only computational bottleneck.

4 Kalman-filter controller

We designed a one-dimensional **Kalman filter** to act as an alternative controller. The *state* and *measurement* equations take the general form

$$\begin{aligned} x_{k+1} &= \Phi_{k+1,k} x_k + \Gamma_{k+1,k} w_k + \Psi_{k+1,k} u_k \\ z_k &= H_k x_k + r_k \end{aligned} \tag{24}$$

where r_k denotes Gaussian white noise with the covariance matrix R_k. If r_k is coloured noise or if $R_k = 0$, the filtering-error covariance matrix $P_{k|k}$ becomes singular. The state x_k and the measurements z_k are jointly Gaussian. Mendel [3] gives details of this model.

From Fig. 1 we see that

$$\begin{aligned} e_k &= t_k + n_k - p_k \\ &= e_{k,\,actual} + n_k \end{aligned} \tag{25}$$

where $e_{k,\,actual}$ denotes the actual pointing error in degrees. Similarly

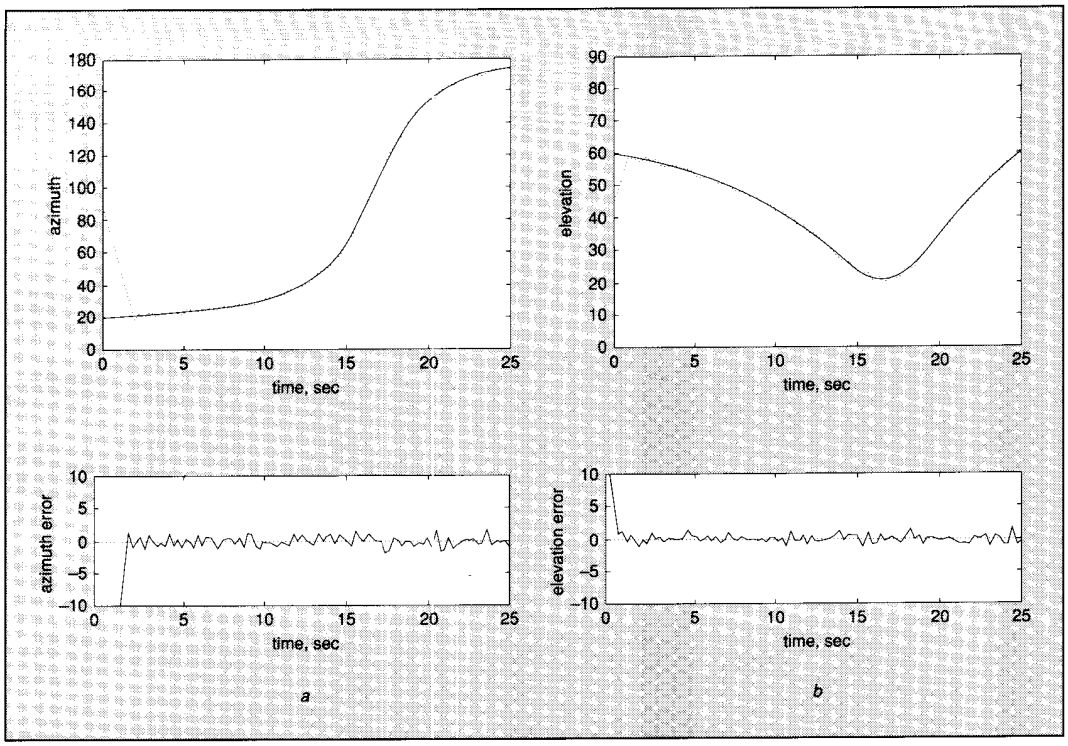

Fig. 15 Kalman-filter controller with unmodelled-effects noise variance $Var(a)=0$

a azimuth position and error *b* elevation position and error

$$\begin{aligned}\dot{e}_k &= e_k - e_{k-1}\\ &= e_{k,\,actual} + n_k - e_{k-1,\,actual} - n_{k-1}\\ &= \dot{e}_{k,\,actual} + n_k - n_{k-1}\end{aligned} \tag{26}$$

Let x_{k+1} denote the platform angular velocity, in degrees/second, required at time k to give zero position error at time $k+1$. Thus, the controller output at time k equals the 'predictive' estimate $\hat{x}_{k+1|k} = v_k$. For zero target acceleration and zero platform velocity uncertainty in the $(k,\ k+1)$ time interval

$$Tx_{k+1} = Tx_k + e_{k,\,actual} + \dot{e}_{k,\,actual} \tag{27}$$

Tx_k denotes the angular distance traversed by the platform between time $k-1$ and time k. $Tx_k + \dot{e}_{k,\,actual}$ denotes the distance the target traverses during that same interval. Adding the current position error $e_{k,\,actual}$ gives the distance Tx_{k+1} that the platform must turn during the next time interval to eliminate the position error at time $k+1$.

Let a_k denote zero-mean white Gaussian noise that models target acceleration and other unmodelled effects. Suppose a_k has variance

$$\sigma_a^2 = E[a_k^2] \tag{28}$$

Then the complete state equation is

$$\begin{aligned}Tx_{k+1} &= Tx_k + e_{k,\,actual} + \dot{e}_{k,\,actual} + T(y_k + a_k)\\ &= Tx_k + (e_k - n_k) + (\dot{e}_k - n_k + n_{k-1}) + T(y_k + a_k)\\ &= Tx_k + e_k + \dot{e}_k + T(y_k + a_k) - 2n_k + n_{k-1}\end{aligned}$$

$$\begin{aligned}x_{k+1} &= x_k + \frac{1}{T}(e_k + \dot{e}_k) + y_k + a_k - \frac{2}{T}n_k + \frac{1}{T}n_{k-1}\\ &= x_k + \frac{1}{T}(e_k + \dot{e}_k) + w_k\end{aligned} \tag{29}$$

where

$$w_k = y_k + a_k - \frac{2}{T}n_k + \frac{1}{T}n_{k-1} \tag{30}$$

Since y_k, a_k, n_k and n_{k-1} are mutually uncorrelated, zero-mean and Gaussian, w_k is also Gaussian, with statistics

$$E[w_k] = 0$$

$$Q_k = E[w_k^2] = \sigma_y^2 + \sigma_a^2 + \frac{5}{T^2}\sigma_n^2 \text{ for all } k \tag{31}$$

Comparing eqn. 29 with the general state equation (eqn. 24) gives

$$\Phi_{k+1,\,k} = \Gamma_{k+1,\,k} = 1 \text{ for all } k$$

$$\Psi_{k+1,\,k} = \frac{1}{T} \text{ for all } k \tag{32}$$

$$u_k = e_k + \dot{e}_k$$

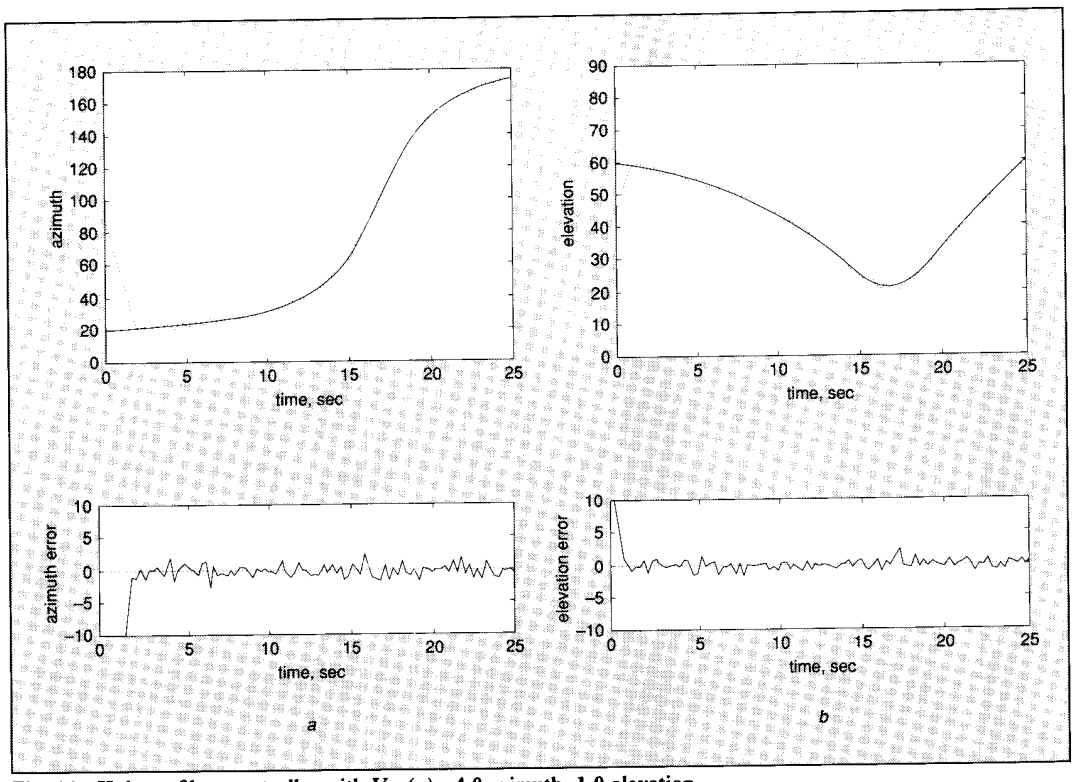

Fig. 16 Kalman-filter controller with $Var(a)=4.0$ azimuth, 1.0 elevation

a azimuth position and error *b* elevation position and error

We assume that we can accurately measure the platform's angular velocity. Thus,

$$H_k = 1 \text{ for all } k \tag{33}$$

However, even if we can measure the velocity perfectly, we must add some white Gaussian noise r_k to the measurement equation. Otherwise, the filtering-error covariance matrix $P_{k|k}$ becomes singular and remains singular [3]. The measurement equation is then

$$\begin{aligned} z_k &= x_k + r_k \\ &= \hat{x}_{k|k-1} + \tilde{x}_{k|k-1} + r_k \\ &= \hat{x}_{k|k-1} + r'_k \end{aligned} \tag{34}$$

Since we assume $\tilde{x}_{k|k-1}$ and r_k are uncorrelated, the variance of r'_k is

$$\begin{aligned} R'_k &= E[r'^2_k] \\ &= E[\tilde{x}^2_{k|k-1}] + E[r^2_k] \\ &= P_{k|k-1} + R_k \end{aligned} \tag{35}$$

The general form of the recursive Kalman-filter equations is

$$\begin{aligned} \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k[z_k - H_k\hat{x}_{k|k-1}] \\ K_k &= P_{k|k-1}H_k^T[H_kP_{k|k-1}H_k^T + R_k]^{-1} \\ \hat{x}_{k+1|k} &= \Phi_{k+1,k}\hat{x}_{k|k} + \Psi_{k+1,k}u_k \\ P_{k|k-1} &= \Phi_{k,k-1}P_{k-1|k-1}\Phi^T_{k,k-1} + \Gamma_{k,k-1}Q_{k-1}\Gamma^T_{k,k-1} \\ P_{k|k} &= [I - K_kH_k]P_{k|k-1} \end{aligned} \tag{36}$$

where $Q_k = Var(w_k) = E[w_kw_k^T]$. Substituting eqns. 32–35 and the definition of v_k into eqn. 36, we get the following one-dimensional Kalman filter:

$$\begin{aligned} \hat{x}_{k|k} &= v_{k-1} + K_kr'_k \\ K_k &= \frac{P_{k|k-1}}{R'_k} \\ v_k &= \hat{x}_{k|k} + \frac{1}{T}(e_k + \dot{e}_k) \\ P_{k|k-1} &= P_{k-1|k-1} + Q_{k-1} \\ P_{k|k} &= [1 - K_k]P_{k|k-1} \end{aligned} \tag{37}$$

Unlike the fuzzy controller, this Kalman filter does not automatically restrict the output v_k to a usable range. We must apply a threshold immediately after the controller. To remain consistent with the fuzzy controller, we set the following thresholds:

$$\begin{aligned} |v_k| &\le \frac{9.0}{T} \text{degrees/second azimuth} \\ |v_k| &\le \frac{4.5}{T} \text{degrees/second elevation} \end{aligned} \tag{38}$$

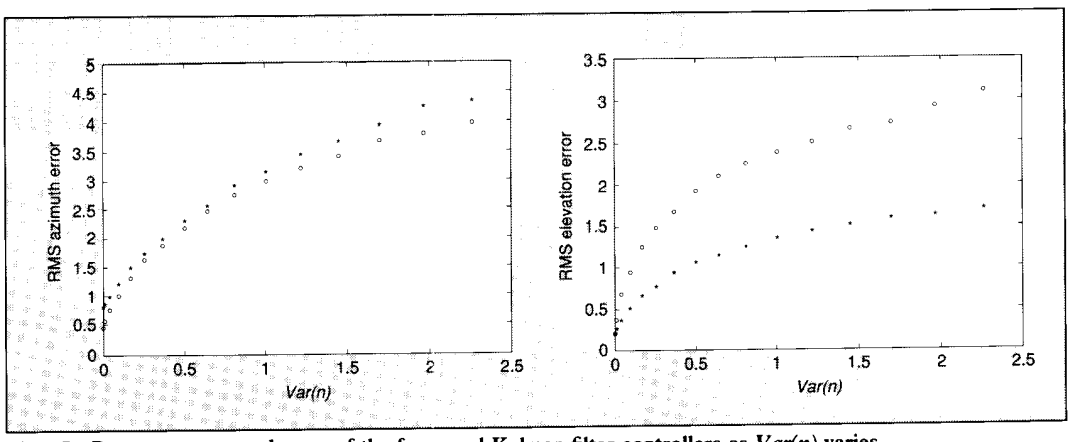

Fig. 17 Root-mean-squared error of the fuzzy and Kalman-filter controllers as *Var(n)* varies

Asterisks plot the fuzzy controller's RMS error. Circles indicate the Kalman-filter's RMS error

The Kalman filter in eqn. 37 is optimal only when v_k stays within the thresholds in eqn. 38. Thus, when comparing root-mean-squared errors for the two controllers, we ignore the first ten iterations, when the platform still moves at maximum velocity to intercept the target.

5 Fuzzy and Kalman-filter control surfaces

Each control system maps inputs to outputs. Geometrically, these input/output transformations define *control surfaces*. The control surfaces are sheets in the input space (since the output velocity v_k is a scalar quantity). Three inputs and one output give rise to a four-dimensional control surface, which we cannot plot. Instead, for each controller we can plot a family of three-dimensional control surfaces indexed by constant values of the fourth variable, the error e_k, for example. Each control surface characterises the fuzzy system's fuzzy-set value definitions and its bank of FAM rules. Different sets of FAM rules yield different fuzzy controllers, and hence different control surfaces. Fig. 9 shows a cross-section of the FAM bank when $e_k = ZE$. Each entry in this linguistic matrix represents one FAM rule with $e_k = ZE$ as the first antecedent term.

The entire FAM bank (including cross-sections for e_k equal to each of the seven fuzzy-set values LN, MN, SN, ZE, SP, MP and LP) determines how the system maps input fuzzy sets to output fuzzy sets. The fuzzy-set membership functions shown in Fig. 2 determine the degree to which each crisp input value belongs to each fuzzy-set value. Therefore, both the fuzzy-set value definitions and the FAM bank determine the defuzzified output v_k for any set of crisp input values e_k, $\dot{e}_k$ and v_{k-1}.

Fig. 10 shows the control surfaces of the fuzzy controller for $e_k = 0$. All four plots use the same FAM rules, but with different amounts of overlap in their fuzzy-set value definitions. We plotted the control output v_k against $\dot{e}_k$ and v_{k-1}. Since we use the same algorithm for tracking in azimuth and elevation, the control surfaces for the two dimensions differ in scale only by a factor of two.

The controller in Fig. 10*d* has no overlap. Eliminating overlap removes all fuzziness from the controller. Only one rule can fire at any time, since only one combination of antecedents can yield a non-zero activation level. Removing all overlap from the fuzzy-set value definitions reduces the fuzzy controller to an expert system.

The Kalman filter has a random control surface that depends on a time-varying parameter. From eqn. 37, we see that

$$v_k = \hat{x}_{k|k} + \frac{1}{T}(e_k + \dot{e}_k)$$

$$\hat{x}_{k|k} = v_{k-1} + K_k r'_k$$

where r'_k denotes Gaussian noise with variance given by eqn. 35. Combining these two equations gives the equation for the random control surface

$$v_k = v_{k-1} + \frac{1}{T}(e_k + \dot{e}_k) + K_k r'_k \tag{39}$$

At time k, the noise term $K_k r'_k$ has variance

$$\sigma_k^2 = K_k^2 R'_k$$

$$= \frac{P_{k|k-1}^2}{R'_k} \text{ substituting from eqn. 37} \tag{40}$$

$$= \frac{P_{k|k-1}^2}{P_{k|k-1} + R_k}$$

substituting from eqn. 35. Combining eqns. 39 and 40 gives a new control surface equation

$$v_k = v_{k-1} + \frac{1}{T}(e_k + \dot{e}_k) + \sigma_k r''_k \tag{41}$$

where r''_k denotes unit-variance Gaussian noise. Therefore, the Kalman-filter control output equals the sum of the three scaled input variables, plus additive Gaussian noise with time-dependent variance σ_k^2. For constant error e_k, we can interpret eqn. 41 as a smooth control surface in R^3 defined by

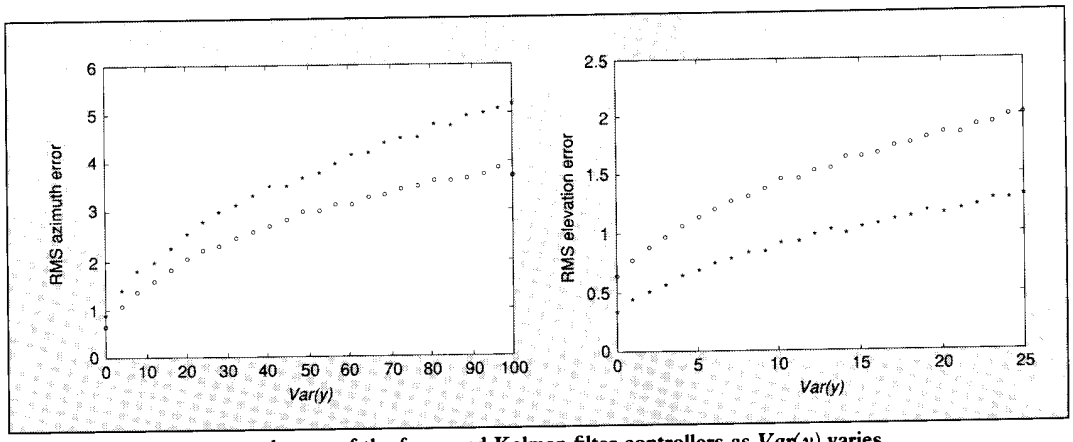

Fig. 18 Root-mean-squared error of the fuzzy and Kalman-filter controllers as *Var(y)* varies

Asterisks plot the fuzzy controller's RMS error. Circles indicate the Kalman-filter's RMS error

$$v_k = v_{k-1} + \frac{1}{T}(e_k + \dot{e}_k) \tag{42}$$

and perturbed at time k by Gaussian noise with variance σ_k^2.

The model assumes fixed variance values

$$R_k = \bar{R}$$

$$Q_k = \bar{Q} = \sigma_y^2 + \sigma_a^2 + \frac{5}{T^2}\sigma_n^2 \tag{43}$$

for all k. If $\lim_{k\to\infty} P_{k|k-1} = \bar{P}$, substituting eqn. 43 into eqn. 40 gives

$$\lim_{k\to\infty} \sigma_k^2 = \bar{\sigma}^2 = \frac{\bar{P}^2}{\bar{P}+\bar{R}} \tag{44}$$

When $P_{k|k-1}$ converges, eqn. 37 is reduced to a steady-state Kalman filter. Substituting eqn. 44 into eqn. 41 gives the steady-state control surface equation

$$v_k = v_{k-1} + \frac{1}{T}(e_k + \dot{e}_k) + \bar{\sigma} r_k'' \tag{45}$$

The recursive Kalman-filter equations (eqn. 37) imply the *matrix Riccati equation* relating $P_{k+1|k}$ to $P_{k|k-1}$

$$\begin{aligned} P_{k+1|k} &= P_{k|k} + Q_k \\ &= (1 - K_k)P_{k|k-1} + Q_k \\ &= \frac{R_k P_{k|k-1}}{P_{k|k-1} + R_k} + Q_k \end{aligned} \tag{46}$$

At steady-state, eqn. 46 is reduced to

$$\bar{P} = \frac{\bar{R}\bar{P}}{\bar{P}+\bar{R}} + \bar{Q} \tag{47}$$

Solving eqn. 47 for $\bar{Q}$ gives

$$\begin{aligned} \bar{Q} &= \bar{P} - \frac{\bar{R}\bar{P}}{\bar{P}+\bar{R}} \\ &= \frac{\bar{P}^2}{\bar{P}+\bar{R}} \\ &= \bar{\sigma}^2 \end{aligned} \tag{48}$$

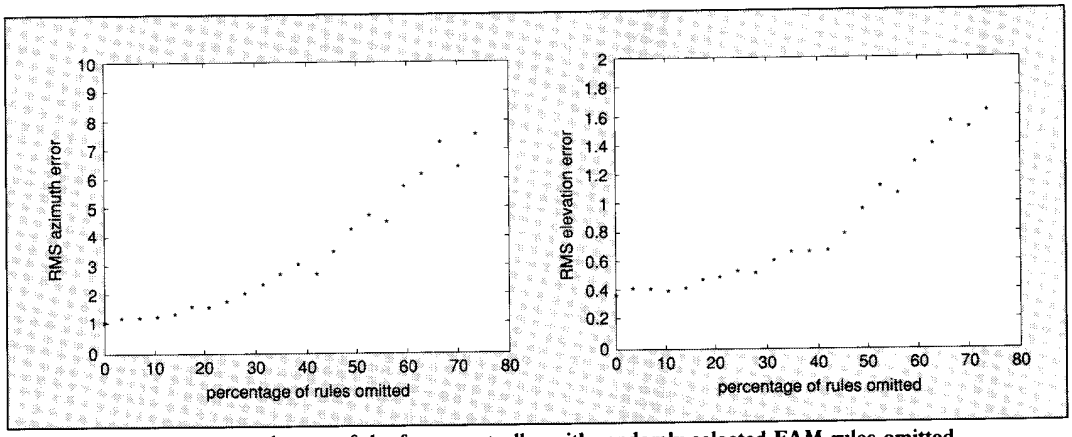

Fig. 19 Root-mean-squared error of the fuzzy controller with randomly selected FAM rules omitted

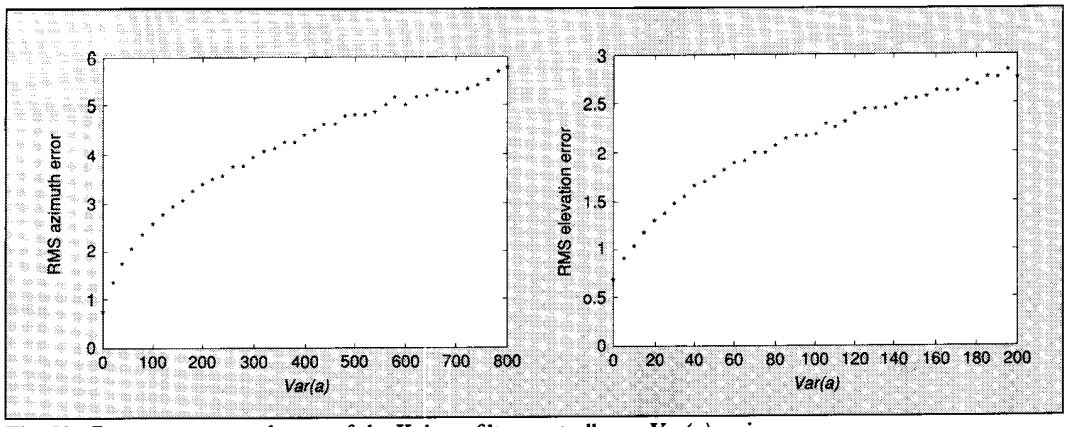

Fig. 20 Root-mean-squared error of the Kalman-filter controller as *Var(a)* varies

substituting from eqn. 44. Therefore the variance $\bar{\sigma}^2$ of the control-surface perturbation noise does not depend on $P_{0|0}$ and $\bar{R}$, which we chose arbitrarily.

In our simulations, the standard deviation σ_k converged after only two or three iterations. Fig. 11 shows two realisations of the Kalman-filter random control surface for $e_k = 0$. We used the same values for σ_n, σ_y and T as for the fuzzy azimuth controller; $\sigma_n = 0.5°$, $\sigma_y = 2.0°$/second and $T = 1/4$ second. σ_a^2, the variance of the unmodelled-effects process a_k, equals 0 in Fig. 11*a* and 4.0 in Fig. 11*b*. In both cases we used the azimuth output thresholds and unit initial conditions, $|v_k| \leq 9.0/T = 36°$/second and $R_k = P_{0|0} = 1.0$ for all k.

6 Simulation results

Our target-tracking simulations model several real-world scenarios. Suppose we have mounted the target-tracking system on the side of a vehicle, aircraft or ship. The system tracks a missile that cuts across the detection range on a straight flight path. The target maintains a constant speed of 1870 mph and comes within 3.5 miles of the platform at its closest approach. The platform can scan from 0 to 180° in azimuth at a maximum rate of 36°/second, and from 0 (vertical) to 90° in elevation at a maximum rate of 18°/second. The sampling interval is 1/4 second. The gain of the fuzzy controller equals 0.9. Thus, the maximum error considered is 10° azimuth and 5° elevation. We limited all error values above this level.

Fig. 12 demonstrates the best performance of the fuzzy controller for a simulated scenario. The solid lines indicate target position. The dotted lines indicate platform position. To achieve this performance, we calibrated the three design parameters: upper and lower trapezoid bases, and the gain. Figs. 13 and 14 show examples of uncalibrated systems. Too much overlap causes excessive overshoot. Too little overlap causes lead or lag for several consecutive time intervals. A gain of 0.9 suffices for most scenarios. We can fine-tune the fuzzy control system by altering the percentage overlap between adjacent fuzzy sets.

Fig. 15 demonstrates the best performance of the Kalman-filter controller for the same scenario used to test the fuzzy controller. For simplicity, $R_k = P_{0|0}$ for all values of k. For this study, we chose the values 1.0 (unit variance) for azimuth and 0.25 for elevation. This 1:4 ratio reflects the difference in the scanning range

0 to 180° azimuth

0 to 90° elevation

We set σ_a^2 to 0 for optimal performance. Fig. 16 shows the Kalman-filter controller's performance when $\sigma_a^2 = 4.0$ azimuth, 1.0 elevation.

6.1 *Sensitivity analysis*

We compared the uncertainty sensitivity of the fuzzy and Kalman-filter control systems. Under normal operating conditions, when the FAM bank contains all fuzzy control rules and when all noise sources have small variances, the controllers perform almost identically. The fuzzy controller has more error in azimuth and less error in elevation, where the target trajectory is more non-linear. Under more uncertain conditions, their performance differs.

We compared the sensitivity of the two controllers by alternately increasing the variance of the two white noise sources modelled in Fig. 1. Figs. 17 and 18 show *root-mean-squared errors* (RMSE) in degrees for both controllers, with each data point representing an average of ten runs. Asterisks plot the fuzzy controller RMS error and circles indicate the Kalman-filter RMS error. In Fig. 17, we fixed the output velocity uncertainty σ_y^2, and increased the measurement uncertainty σ_n^2. In Fig. 18, we did just the opposite, fixing σ_n^2 and increasing σ_y^2.

The Kalman-filter state equation (eqn. 29) contains the noise term w_k, whose variance we must assume. When *Var*(w) increases, the state equation becomes more uncertain. The fuzzy control FAM rules depend implicitly on this same equation, but without the noise term. Instead, the fuzziness of the FAM rules accounts for the system uncertainty. This suggests that we can increase the uncertainty of the implicit state equation by omitting randomly selected FAM rules. Figs. 19 and 20 show the effect on the root-mean-squared error in degrees when we omit FAM rules and increase σ_a^2, the variance of the

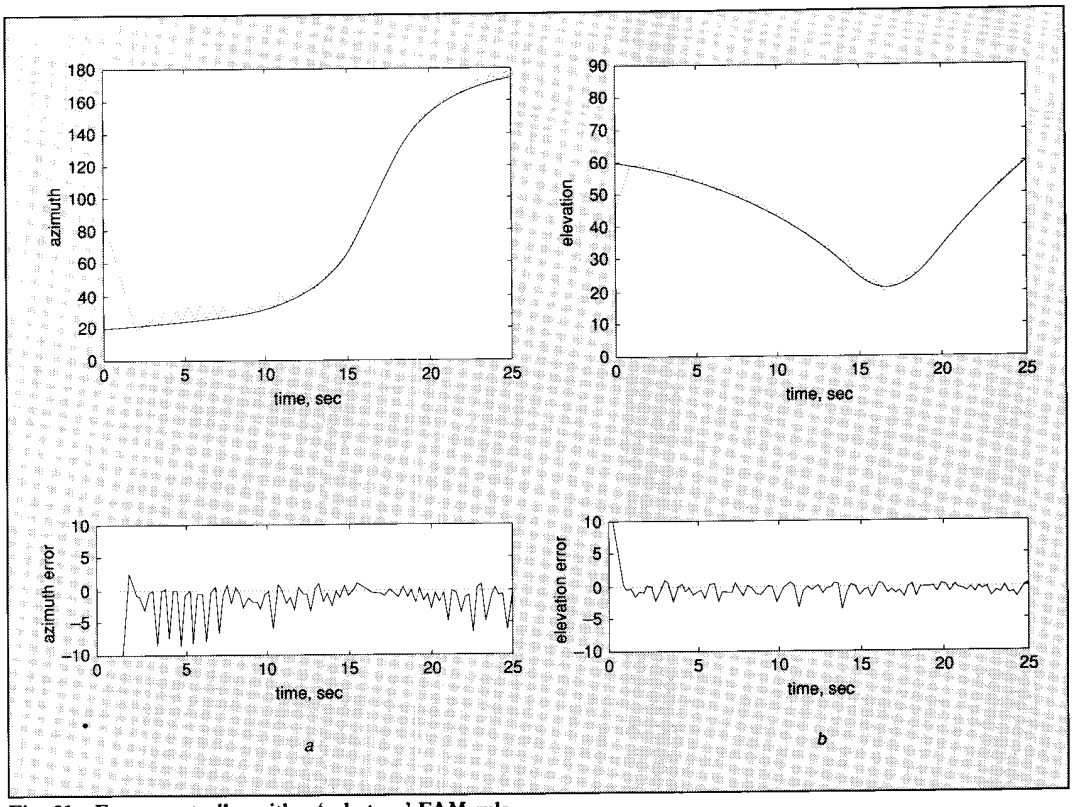

Fig. 21 Fuzzy controller with a 'sabotage' FAM rule

a azimuth position and error *b* elevation position and error
The sabotage rule (*ZE*, *ZE*, *ZE*; *LP*) replaces the steady-state FAM rule (*ZE*, *ZE*, *ZE*; *ZE*). The system quickly adjusts each time the sabotage rule is activated.

unmodelled-effects noise process. Recall from eqn. 31 that increasing σ_a^2 increases $Q_k = Var(w)$. Each data point again averages ten runs.

The controllers behave differently as uncertainty increases. The RMSE of the fuzzy controller increases only slightly until we omit nearly 60% of the FAM rules. The RMSE of the Kalman filter increases steeply for small values of σ_a^2, and then gradually levels off.

We also rested the fuzzy controller's robustness by 'sabotaging' the most vulnerable FAM rule. This could reflect a lack of accurate expertise or a highly unstructured problem. Changing the consequent of the steady-state FAM rule (*ZE*, *ZE*, *ZE*; *ZE*) to *LP* gives the following nonsensical FAM rule:

IF the platform points directly at the target
AND both the target and the platform are stationary,
THEN turn in the positive direction with maximum velocity.

Fig. 21 shows the fuzzy system performance when this sabotage FAM rule replaces the steady-state FAM rule. When the sabotage FAM rule is activated, the system quickly adjusts to again decrease the error. The fuzzy system is piecewise stable.

6.2 *Adaptive FAM (AFAM)*

We used unsupervised product-space clustering [4] to train an adaptive FAM (AFAM) fuzzy controller. Differential competitive learning (DCL) adaptively clustered input/output pairs. In the Appendix, we describe product-space clustering with DCL. For this study, there were four input neurons in F_X. A manually designed FAM bank and 80 random target trajectories generated 19 236 training vectors. Each product-space training vector $(e_k, \dot{e}_k, v_{k-1}, v_k)$ defined a point in R^4.

Symmetry allowed us to reflect about the origin all sample vectors with negative errors e_k. We then trained 3 000 synaptic quantisation vectors ($p = 3\,000$) in the positive error half-space. For each sample vector, we defined the ten closest synaptic vectors as 'winners' ($N = 10$). The matrix W of F_Y within-field synaptic connection strengths had diagonal elements $w_{ii} = 2.9$ and off-diagonal elements $w_{ij} = -0.1$. After training, we reflected the 3 000 synaptic quantisation vectors about the origin to give 6 000 trained synaptic vectors.

The product-space FAM cells uniformly partitioned the four-dimensional product space. Each FAM cell represented a single FAM rule. The four fuzzy variables could assume only the seven fuzzy-set values *LN*, *MN*,

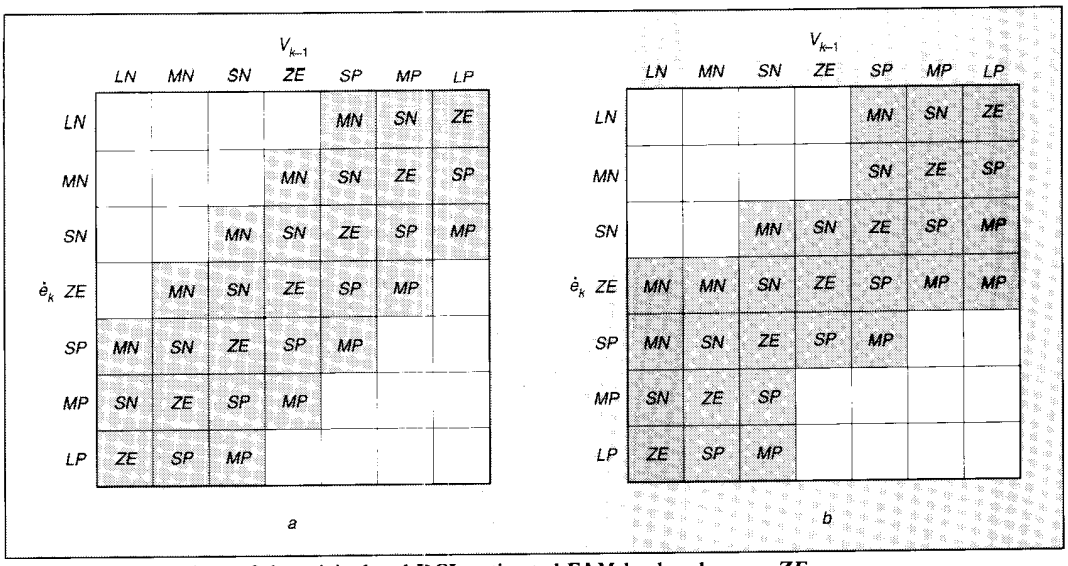

Fig. 22 Cross-sections of the original and DCL-estimated FAM banks when $e_k = ZE$

a original *b* DCL-estimated

SN, *ZE*, *SP*, *MP* and *LP*. Therefore, the product space contained $7^4 = 2401$ FAM cells.

At the end of the DCL training period, we defined a FAM cell as occupied only if it contained at least one synaptic vector. For some combinations of antecendent fuzzy sets, synaptic vectors occupied more than one FAM cell with different consequent fuzzy sets. In these cases, we computed the centroid of the consequent fuzzy sets weighted by the number of synaptic vectors in their FAM cells. We chose the consequent fuzzy set as that output fuzzy-set value with its centroid nearest the weighted centroid value. We ignored other FAM rules with the same antecedents but different consequent fuzzy sets.

Fig. 22*a* shows the $e_k = ZE$ cross-section of the original FAM bank used to generate the training samples. Fig. 22*b* shows the same cross-section of the DCL-estimated FAM bank. Fig. 23 shows the original and DCL-estimated control surfaces for constant error $e_k = 0$.

The regions where the two control surfaces differ correspond to infrequent high-velocity situations. Therefore, the original and DCL-estimated control surfaces yield similar results. Table 1 compares the controllers' root-mean-squared errors for ten randomly-selected target trajectories.

7 Conclusion

We developed and compared a fuzzy control system and a Kalman-filter control system for real-time target tracking. The fuzzy system represented uncertainty with continuous or fuzzy sets, with the partial occurence of multiple alternatives. The Kalman-filter system represented uncertainty with the random occurrence of an exact alternative. Accordingly, our simulations tested each system's response to a different family of uncertainty environments, one fuzzy and the other random. In general, representative training data can 'blindly' generate the governing FAM rules.

These simulations suggest that, in many cases, fuzzy controllers may be a robust computationally effective alternative to linear Kalman-filter (indeed to non-linear extended Kalman-filter) approaches to real-time system control, even when we can accurately articulate an input-output maths model.

8 Acknowledgment

This research was supported by the Air Force Office of Scientific Research (AFOSR-88-0236) and by a grant from the Rockwell Science Center.

Table 1 Root-mean-squared errors for ten randomly selected target trajectories

Trajectory number	*Azimuth*		*Elevation*	
	original	estimated	original	estimated
1	2.33	2.33	3.31	3.37
2	4.14	4.14	3.03	2.89
3	6.11	6.11	3.69	3.68
4	3.83	3.83	3.32	3.30
5	4.02	4.02	3.11	3.10
6	2.84	2.84	1.20	1.21
7	3.22	3.22	3.04	2.98
8	0.75	0.74	2.00	2.00
9	9.28	9.27	5.50	5.41
10	1.81	1.81	2.29	2.29
average	3.83	3.83	3.05	3.02

The original and DCL-estimated FAM banks yielded similar results, since they differed only in regions corresponding to infrequent high-velocity situations.

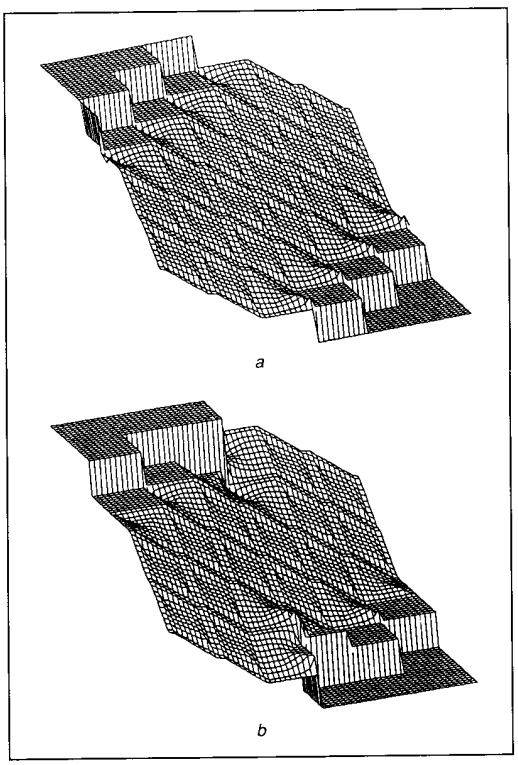

Fig. 23 Control surfaces for constant error $e_k = 0$

a original *b* DCL-estimated

9 References

[1] KOSKO, B.: 'Foundations of fuzzy estimation theory.' PhD Dissertation, Department of Electrical Engineering, University of California at Irvine, June 1987

[2] KOSKO, B.: 'Fuzzy entropy and conditioning,' *Inform. Sci.*, 1986, **40**, pp. 165–174

[3] MENDEL, J.M.: 'Lessons in digital estimation theory' (Prentice-Hall, 1987)

[4] KOSKO, B.: 'Unsupervised learning in noise,' *IEEE Trans.*, 1990, **NN-1**, (1), pp. 44–57

[5] KOSKO, B.: 'Stochastic competitive learning.' Proc. Summer 1990 Int. Joint Conf. on Neural Networks, June 1990, **II**, pp. 215–226

[6] KOSKO, B.: 'Neural networks and fuzzy systems: a dynamical system approach to machine intelligence' (Prentice-Hall, 1992)

[7] KONG, S.-G., and KOSKO, B.: 'Differential competitive learning for centroid estimation and phoneme recognition,' *IEEE Trans. Neural Networks*, 1991, (1)

[8] HUBER, P.J.: 'Robust statistics' (Wiley, 1981)

10 Appendix: Product-space clustering with differential competitive learning

10.1 Adaptive vector quantisation

Product-space clustering [4] is a form of stochastic adaptive vector quantisation. Adaptive vector quantisation (AVQ) systems adaptively quantise pattern clusters in R^n. Stochastic competitive-learning systems are neural AVQ systems. Neurons compete for the activation induced by randomly sampled patterns. The corresponding fan-in vectors adaptively quantise the pattern space R^n. The p synaptic vectors m_j define the p columns of the synaptic connection matrix M. M interconnects the n input or linear neurons in the input neuronal field F_X to the p competing non-linear neurons in the output field F_Y. Fig. 24 illustrates the neural network topology.

Learning algorithms estimate the unknown probability density function $p(x)$, which describes the distribution of patterns in R^n. More synaptic vectors arrive at more probable regions. Where sample vectors x are dense or sparse, synaptic vectors m_j should be dense or sparse. The local count of synaptic vectors then gives a non-parametric estimate of the volume density $P(V)$ for volume $V \subset R^n$

$$P(V) = \int_V p(x)\, dx \tag{49}$$

$$\approx \frac{\text{number of } m_j \in V}{p} \tag{50}$$

In the extreme case that $V = R^n$, this approximation gives $P(V) = p/p = 1$. For improbable subsets V, $P(V) = 0/p = 0$.

10.2 Stochastic competitive learning algoritms

The metaphor of competing neurons is reduced to nearest-neighbour classification. The AVQ system compares the current vector random sample $x(t)$ in Euclidean distance to the p columns of the synaptic connection matrix M and to the p synaptic vectors $m_1(t), \ldots, m_p(t)$. If the jth synaptic vector $m_j(t)$ is closest to $x(t)$, the jth output neuron 'wins' the competition for activation at time t. In practice, we sometimes define the nearest N synaptic vectors as winners. Some scaled form of $x(t) - m_j(t)$ updates the nearest or 'winning' synaptic vectors. 'Losers' remain unchanged; $m_i(t+1) = m_i(t)$. Competitive synaptic vectors converge to pattern-class centroids exponentially fast [5].

The following process describes the competitive AVQ algorithm, where the third step depends on which learning algorithm updates the winning synaptic vectors.

10.3 Competitive AVQ algorithm

- Initialise synaptic vectors, $m_i(0) = x(i)$, $i = 1, \ldots, p$. Sample-dependent initialisation avoids many pathologies that can distort nearest-neighbour learning.
- For random sample $x(t)$, find the closest or 'winning' synaptic vector $m_j(t)$

$$\|m_j(t) - x(t)\| = \min_i \|m_i(t) - x(t)\| \tag{51}$$

where $\|x\|^2 = x_1^2 + \cdots + x_n^2$ defines the squared Euclidean vector norm of x. We can define the N synaptic vectors closest to x as 'winners.'
- Update the winning synaptic vector(s) $m_j(t)$ with an appropriate learning algorithm.

10.4 *Differential competitive learning (DCL)*

Differential competitive 'synapses' learn only if the competing 'neuron' changes its competitive status [6]

$$\dot{m}_{ij} = \dot{S}_j(y_j)[S_i(x_i) - m_{ij}] \tag{52}$$

or in vector notation

$$\dot{m}_j = \dot{S}_j(y_j)[S(x) - m_j] \tag{53}$$

where $S(x) = (S_1(x_1), \ldots, S_n(x_n))$ and $m_j = (m_{1j}, \ldots, m_{nj})$. m_{ij} denotes the synaptic value between the ith neuron in input field F_X and the jth neuron in competitive field F_Y. Non-negative signal functions S_i and S_j convert the real-valued activations x_i and y_j into bounded monotone and non-decreasing signals $S_i(x_i)$ and $S_j(y_j)$. $\dot{m}_{ij}$ and $\dot{S}_j(y_j)$ denote the time derivatives of m_{ij} and $S_j(y_j)$, synaptic and signal velocities. $S_j(y_j)$ measures the competitive status of the jth competing neuron in F_Y. Usually, S_j approximates a binary threshold function. For example, S_j may equal a steep binary logistic sigmoid curve

$$S_j(y_j) = \frac{1}{1 + e^{-cy_j}} \tag{54}$$

for a constant $c > 0$. The jth neuron wins the laterally inhibitive competition if $S_j = 1$ and loses if $S_j = 0$.

For discrete implementation, we use the DCL algorithm as a stochastic difference equation [7]

$$m_j(t+1) = m_j(t) + c_t \Delta S_j(y_j(t))[S(x(t)) - m_j(t)]$$

$$\text{if the } j\text{th neuron wins} \tag{55}$$

$$m_i(t+1) = m_i(t) \text{ if the } i\text{th neuron loses} \tag{56}$$

$\Delta S_j(y_j(t))$ denotes the time change of the jth neuron's competition signal $S_j(y_j)$ in the competition layer F_Y

$$\Delta S_j(y_j(t)) = sgn[S_j(y_j(t+1)) - S_j(y_j(t))]. \tag{57}$$

We define the signum operator $sgn(x)$ as

$$sgn(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}. \tag{58}$$

$\{c_t\}$ denotes a slowly decreasing sequence of learning coefficients, such as $c_t = 0.1(1 - t/2\,000)$ for 2 000 training samples. Stochastic approximation [8] requires a decreasing gain sequence $\{c_t\}$ to suppress random disturbances and to guarantee convergence to local minima of mean-squared performance measures. The learning coefficients should decrease slowly

$$\sum_{t=1}^{\infty} c_t = \infty \tag{59}$$

but not too slowly

$$\sum_{t=1}^{\infty} c_t^2 < \infty \tag{60}$$

Harmonic-series coefficients, $c_t = 1/t$, satisfy these constraints.

We approximate the competitive signal difference ΔS_j as the activation difference Δy_j

$$\Delta S_j(y_j(t)) = sgn[y_j(t+1) - y_j(t)] \tag{61}$$

$$= \Delta y_j(t) \tag{62}$$

Input neurons in feedforward networks usually behave linearly; $S_i(x_i) = x_i$, or $S(x(t)) = x(t)$. Then we update the winning synaptic vector $m_j(t)$ with

$$m_j(t+1) = m_j(t) + c_t\, \Delta y_j(t)[x(t) - m_j(t)] \tag{63}$$

We update the F_Y neuronal activations y_j with the additive model

$$y_j(t+1) = y_j(t) + \sum_{i}^{n} S_i(x_i(t)) m_{ij}(t) + \sum_{k}^{p} S_k(y_k(t)) w_{kj} \tag{64}$$

For linear signal functions S_i, the first sum in eqn. 64 is reduced to an inner product of sample and synaptic vectors

$$\sum_{i}^{n} x_i(t) m_{ij}(t) = x^T(t) m_j(t) \tag{65}$$

Positive learning then tends to occur ($\Delta m_{ij} > 0$) when x is close to the jth synaptic vector m_j.

Since a binary threshold function approximates the output signal function $S_k(y_k)$, the second sum in eqn. 64 sums over just the winning neurons; $\Sigma_k\, w_{kj}$ for all winning neurons y_k.

The $p \times p$ matrix W contains the F_Y within-field synaptic connection strengths. Diagonal elements w_{ii} are positive and off-diagonal elements negative. Winning neurons excite themselves and inhibit all other neurons. Fig. 24 shows the connection topology of the laterally inhibitive DCL network.

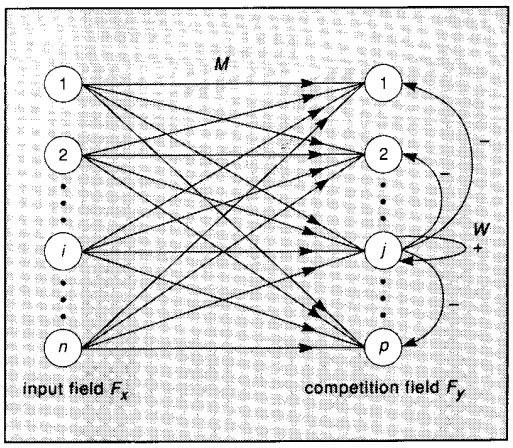

Fig. 24 Topology of the laterally inhibitive DCL network

The paper was received on 15 November 1991.

The authors are with the Department of Electrical Engineering, Signal and Image Processing Institute, University of Southern California, Los Angeles CA 90089-2564, USA.