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, B).

Fuzzy controllers are fuzzy systems. A finite fuzzy set A is a point [1] in a unit hypercube \( I^* = [0, 1]^n \). A fuzzy system \( F : I^* \rightarrow I^* \) is a mapping between unit hypercubes. \( I^* \) contains all fuzzy subsets of the domain space \( X = \{x_1, \ldots, x_n\} \). \( I^* \) is the fuzzy power set \( F(2^n) \) of \( X \). \( I^* \) contains all the fuzzy subsets of the range space \( Y = \{y_1, \ldots, y_n\} \). Element \( x \in X \) belongs to fuzzy set \( A \) to degree \( m_A(x) \). The \( 2^n \) non-fuzzy subsets of \( X \) correspond to the \( 2^n \) corners of the fuzzy cube \( I^* \). 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, B), 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
'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. 1** Target-tracking system

**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 decreased 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 \( n_t \). \( n_t \) has zero mean and variance \( E[n_t] = o_t \).

The radar sends azimuth and elevation co-ordinates to the tracking system at the end of each time interval. We calculate the current error \( e_t \) in platform position and the change in error \( \delta e_t \). 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 \( v_t \) gives the estimated angular velocity required during the next time interval. In principle, a hardware system must convert the angular velocity \( v_t \) into a voltage or current. For a controller output \( v_t \), the platform turns \( T(\psi_t + \psi_t) \), where \( T \) denotes the sampling interval in seconds, and \( \psi_t \) and \( \psi_t \) have units of degrees/second. We model the uncertainty \( \psi_t \) as white Gaussian noise with zero mean and variance \( E[\psi_t] = o_t \).

![Fig. 3 FAM inference procedure depends on FAM rule encoding procedure](image)

a Correlation-minimum encoding  b Correlation-product encoding

3 Fuzzy controller

We restrict the output angular velocity \( v_t \) 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

\[
|v_t| = \frac{9.0}{T} \text{ degrees/second azimuth}
\]

\[
|v_t| = \frac{4.5}{T} \text{ degrees/second elevation}
\]

![Fig. 4 Correlation-product inferences followed by centroid defuzzification](image)

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

INTELLIGENT SYSTEMS ENGINEERING AUTUMN 1992
• $SP =$ 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 ith fit or fuzzy unit [2] equals the value of the ith fuzzy set at the input value. In other words, the ith fit measures the degree to which the input belongs to the ith 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

$$1 \rightarrow (0 \ 0 \ 0 \ 0.7 \ 0.7 \ 0 \ 0)$$
$$-4 \rightarrow (0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0)$$
$$3.8 \rightarrow (0 \ 0 \ 0 \ 0 \ 0.1 \ 1 \ 0)$$

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

$$m_{sp}(3.8) = \delta(y - 3.8) \ast m_{sp}(y) = 0.1$$  \hspace{1cm} (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. 5** Output fuzzy set $O$

Similarly, the position error $\epsilon_i$ 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 $10^\circ$ azimuth and $5^\circ$ 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 $\epsilon_i$, $\delta_i$ and $v_{i-1}$. We define seven fuzzy levels by the following library of fuzzy-set values of the fuzzy variables $\epsilon_i$, $\delta_i$, and $v_{i-1}$:

- $LN =$ Large Negative
- $MN =$ Medium Negative
- $SN =$ Small Negative
- $ZE =$ Zero

---

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

6 INTELLIGENT SYSTEMS ENGINEERING AUTUMN 1992
(binary unit) vector \( B \) of length \( n \). A single 1 element in the \( i \)-th slot represents the 'crisp' input value \( x_i \). Similarly, we represent the \( k \)-th 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 comprises \( e_{i_1}, e_{i_2}, \ldots, e_{i_k} \), fuzzy-set values. For example,

\[
\text{IF } e_{i_1} = \text{MP AND } e_{i_2} = \text{SN AND } e_{i_k} = \text{ZE}, \text{ THEN } v_i = \text{SP}
\]

We abbreviate this to \((\text{MP, SN, ZE, SP})\).

The scalar activation value \( \omega_i \) of the \( i \)-th 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, \( m_i(x_i) \) denotes the degree to which \( x_i \) belongs to the fuzzy set \( A_i \):

\[
\begin{align*}
\text{LN} & \quad \text{MN} & \quad \text{SN} & \quad \text{ZE} & \quad \text{SP} & \quad \text{MP} & \quad \text{LP} \\
\epsilon_i & = 2.6 & \rightarrow & 0 & 0 & 0 & 1 & 0 & 0 \\
\epsilon_i & = -2.0 & \rightarrow & 0 & 0 & 1 & 0 & 0 & 0 \\
v_i & = 1.8 & \rightarrow & 0 & 0 & 0 & 0 & 0 & 0 \\
m_{LU}(x_i) & = 0.4 \\
m_{MN}(x_i) & = 1 \\
m_{SN}(x_i) & = 0.1 \\
\omega_i & = \min(0.4, 1, 0.1) = 0.1
\end{align*}
\]

Thus, the system activates the consequent fuzzy set \( SP \) to the \( \omega_i \) degree = 0.1.

The output fuzzy set's shape depends on the FAM-rule encoding scheme used. With correlation-product encoding, we clip the consequent fuzzy set \( L \), in the library of fuzzy-set values to the \( \omega_i \) degree with pointwise minimum

\[
m_i(y) = \min(\omega_i, m_i(y))
\]

With correlation-product encoding, we multiply \( \omega_i \) by \( m_i(y) \)

\[
m_i(y) = \omega_i m_i(y)
\]

or equivalently

\[
O_i = \omega_i L_i
\]

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 \( O = 0.1 SP \), where \( L = SP \) denotes the fuzzy set of small but positive angular velocity values.

The fuzzy system activates each FAM rule consequent to a different degree. For the \( i \)-th FAM rule, this yields the output fuzzy set \( O_i \). The system then sums \( O_i \) to form the combined output fuzzy set \( O \)

\[
O = \sum_{i=1}^{N} O_i
\]

or equivalently

\[
m_i(y) = \sum_{i=1}^{N} m_i(y)
\]

Fig. 8 Fuzzy control system as a parallel FAM bank with centroidal output
The control output $v_i$ equals the fuzzy centroid of $O$

$$v_i = \frac{\int y \cdot m_i(y) \, dy}{\int m_i(y) \, dy}$$

(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

$$v_i = \frac{\sum_{j=1}^{p} y_j \cdot m_i(y_j)}{\sum_{j=1}^{p} m_i(y_j)}$$

(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 $v_i$ from local FAM-rule centroids. Theorem 2 states that $v_i$ 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. 9** $e_i = ZE$ cross-section of the fuzzy control system's FAM bank

Each entry represents one FAM rule with $e_i = ZE$ as the first antecedent term. The shaded FAM rule is 'IF $e_i = ZE$ AND $i = SP$ AND $v_i = SN$, THEN $v_i = ZE$', 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.

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

- a 20.8%
- c 6.3%
- b 23.3%
- d None

We plotted the control output $v_i$ against $e_i$ and $v_{i-1}$ along the west and south borders, respectively.
Fig. 11  Realisations of the Kalman-filter random control surface with \( \sigma^2 = 0 \) for two different values of \( \sigma^2 \), the variance of the unmodelled-effects noise process

\[ a: \sigma^2 = 0 \quad b: \sigma^2 = 4.0 \]

**Theorem 1**

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

\[
v_i = \frac{\sum_{j=1}^{I} w_i \cdot c_i}{\sum_{j=1}^{I} w_i} \quad \text{(9)}
\]

**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, \( m_d(x) = 1 \). Define \( I_i \) and \( c_i \) as the area and centroid of the \( i \)-th FAM rule's consequent set \( I_i \), respectively.

\[
I_i = \int m_i(y) \, dy \quad \text{(10)}
\]

\[
c_i = \frac{\int y \, m_i(y) \, dy}{\int m_i(y) \, dy} \quad \text{from eqn. 10. Hence}
\]

\[
\int y \, m_i(y) \, dy = c_i I_i \quad \text{(11)}
\]

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

\[
\int y \, m_i(y) \, dy = \int y \, w_i m_i(y) \, dy = w_i \int y \, m_i(y) \, dy = w_i c_i I_i \quad \text{(12)}
\]

substituting from eqn. 11. Similarly

\[
\int y \, m_i(y) \, dy = \int y \, w_i m_i(y) \, dy = w_i \int y \, m_i(y) \, dy = w_i c_i I_i \quad \text{(13)}
\]

substituting from eqn. 10.

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

\[
\int y \, m_i(y) \, dy = \int y \, \left[ \sum_{i=1}^{N} m_i(y) \right] \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy = \sum_{i} \int y \, m_i(y) \, dy}
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 $v_i$ from only seven samples of the combined output fuzzy set $O$

$$v_i = \frac{\sum_{j=1}^{7} m_i(y_j) y_j J_j}{\sum_{j=1}^{7} m_i(y_j) J_j}$$

(17)

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

**Proof**

Define $O_i$ as a fit vector of length 7, where the fit value corresponding to the $i$th consequent set has the value $w_i$ and the other entries equal zero. If all the fuzzy sets are symmetric and unimodal, the $j$th fit value of $O_i$ is a sample of $m_i$ at the centroid of the $j$th fuzzy set. The combined output fit vector is

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

(18)

Since

$$m_i(y) = \sum_{j=1}^{N} m_i(y_j)$$

the $j$th fit value of $\mathcal{O}$ is a sample of $m_i$ at the centroid of the $j$th fuzzy set. Equivalently, the $j$th fit value of $\mathcal{O}$ equals the sum of the output activations $w_i$ from the FAM rules with consequent fuzzy sets equal to the $j$th 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 $j$th 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

$$J = \text{diag}(J_1, \ldots, J_7)$$

(19)

where $J_j$ denotes the area of the $j$th fuzzy-set value. If the $i$th FAM rule's consequent fuzzy set equals the $j$th fuzzy-set value, the $j$th fit value of $\mathcal{O}$ increases by $w_i, s_i = y_j$, and $f = J_j$. Thus

$$\mathcal{O} J Y^T = \sum_{i=1}^{N} m_i(y_j) y_j J_j = \sum_{i=1}^{N} w_i s_i f$$

(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

\[
\hat{\mathbf{f}}_i = \sum_{j=1}^{N} m_{ij}(y) \mathbf{f}_j = \sum_{i=1}^{N} \omega_i \mathbf{f}_i
\]  

(21)

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

\[
\omega_i = \frac{\sum_{j=1}^{N} m_{ij}(y) \mathbf{f}_j}{\sum_{j=1}^{N} m_{ij}(y) \mathbf{f}_j}
\]

(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 \( \hat{O} \). QED.

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

<table>
<thead>
<tr>
<th>i</th>
<th>( \omega_i )</th>
<th>consequent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
<td>MP</td>
</tr>
<tr>
<td>2</td>
<td>0.2</td>
<td>SP</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>ZE</td>
</tr>
<tr>
<td>4</td>
<td>0.4</td>
<td>SN</td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td>SP</td>
</tr>
<tr>
<td>6</td>
<td>0.8</td>
<td>ZE</td>
</tr>
<tr>
<td>7</td>
<td>0.6</td>
<td>SN</td>
</tr>
</tbody>
</table>

Fig. 5 shows the combined output fuzzy set \( \hat{O} \), 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 \( \mathbf{O} \) equals \((0, 0, 1.0, 1.8, 0.3, 0, 0)\). From eqn. 22 we get

\[
\omega_k = \frac{-2 \times 1 + 0 \times 1.8 + 2 \times 0.3}{1 + 1.8 + 0.3} = -0.452.
\]

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

\[
\omega_i = \min(m_{MP}(e_k), m_{SN}(\dot{e}_k), m_{ZE}(v_{k-1}))
\]

\[
= \min(0.4, 1, 0.1)
\]

\[
= 0.1
\]

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

\[
m_{SP}(y) = m_{ZE}(y - 2)
\]

INTELLIGENT SYSTEMS ENGINEERING AUTUMN 1992
Define $\epsilon$, $\hat{\epsilon}$, and $\psi$ as the centroids of the corresponding antecedent fuzzy sets in the example above. Thus, $\epsilon = 4$, $\hat{\epsilon} = \frac{-3}{2}$ and $\psi = 0$. The output activation then equals

$$w_i = \min\{m_{ze}(\epsilon_i - \epsilon), m_{ze}(\hat{\epsilon} - \epsilon), m_{ze}(\psi - \psi)\}$$
$$= \min\{m_{ze}(-1.4), m_{ze}(0), m_{ze}(1.8)\}$$
$$= \min(0.4, 1, 0.1)$$
$$= 0.1$$

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 $\psi$,

$$m_{ze}(y) = w_i m_{ze}(y - \psi)$$  \hspace{1cm} (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 models and 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

$$x_{k+1} = \Phi x_k + \Gamma u_k + \Psi n_k$$
$$z_k = H x_k + r_k$$  \hspace{1cm} (24)

where $\nu_k$ denotes Gaussian white noise with the covariance matrix $R$. 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

$$e_k = \hat{e} + \nu_k$$
$$e_k = e_k_{\text{actual}} + \nu_k$$  \hspace{1cm} (25)

where $e_k_{\text{actual}}$ denotes the actual pointing error in degrees. Similarly
\[ \dot{x}_k = x_k - x_{k-1} \]
\[ = \dot{x}_{k, \text{actual}} + n_k - x_{k-1, \text{actual}} - n_{k-1} \]
\[ = \dot{x}_{k, \text{actual}} + n_k - n_{k-1} \] \hspace{1cm} (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 \( \hat{x}_{k+1|k} = \theta_k \). For zero target acceleration and zero platform velocity uncertainty in the \( (k, k+1) \) time interval
\[ T_{x_{k+1}} = T_{x_k} + \dot{x}_{k, \text{actual}} + \dot{x}_{k, \text{actual}} \] \hspace{1cm} (27)

\( T_{x_k} \) denotes the angular distance traversed by the platform between time \( k-1 \) and time \( k \). \( T_{x_k} + \dot{x}_{k, \text{actual}} \) denotes the distance the target traverses during that same interval. Adding the current position error \( x_{k, \text{actual}} \) gives the distance \( T_{x_{k+1}} \) 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
\[ \sigma_i^2 = E[a_i^2] \] \hspace{1cm} (28)

Then the complete state equation is
\[ T_{x_{k+1}} = T_{x_k} + \dot{x}_{k, \text{actual}} + \dot{x}_{k, \text{actual}} + T(y_k + a_k) \]
\[ = T_{x_k} + (\dot{x}_k - n_k) + (\dot{x}_k - n_k - n_{k-1}) + T(y_k + a_k) \]
\[ = T_{x_k} + \dot{x}_k + T(y_k + a_k) - 2n_k + n_{k-1} \]
\[ x_{k+1} = x_k + \frac{1}{T}(\dot{x}_k + \dot{x}_k) + y_k + a_k - \frac{2}{T} n_k + \frac{1}{T} n_{k-1} \]
\[ = x_k + \frac{1}{T}(\dot{x}_k + \dot{x}_k) + w_k \] \hspace{1cm} (29)

where
\[ w_k = y_k + a_k - \frac{2}{T} n_k + \frac{1}{T} n_{k-1} \] \hspace{1cm} (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
\[ E[w_k] = 0 \]
\[ Q_k = E[w_k^2] = \sigma_i^2 + \sigma_j^2 + \frac{5}{T^2} \sigma_i^2 \] \hspace{1cm} (31)

Comparing eqn. 29 with the general state equation (eqn. 24) gives
\[ \Phi_{k+1} = \Gamma_{k+1} = 1 \text{ for all } k \]
\[ \Psi_{k+1} = \frac{1}{T} \text{ for all } k \] \hspace{1cm} (32)
We assume that we can accurately measure the platform’s angular velocity. Thus,

\[ H_k = 1 \text{ for all } k \]  

(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 \( P_{k|k-1} \) becomes singular and remains singular [3]. The measurement equation is then

\[
\begin{align*}
\hat{x}_k &= x_k + r_k \\
\hat{x}_{k|k-1} &= \hat{x}_{k-1} + r_k \\
\hat{x}_{k|k-1} &= \hat{x}_{k-1} + r_k
\end{align*}
\]  

(34)

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

\[
R'_k = E[r'_k r'_k] = E[\hat{x}_{k|k-1} - \hat{x}_{k-1}]^2 = P_{k|k-1} + R_k
\]  

(35)

The general form of the recursive Kalman-filter equations is

\[
\begin{align*}
\hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k [z_k - H_k \hat{x}_{k|k-1}] \\
K_k &= P_{k|k-1} H_k^T [H_k P_{k|k-1} H_k^T + R_k]^{-1} \\
\hat{x}_{k+1|k} &= \Phi_{k+1|k} \hat{x}_{k|k} + \Psi_{k+1|k} u_k
\end{align*}
\]  

(36)

\[
\begin{align*}
P_{k|k-1} &= (I - K_k H_k) P_{k|k-1} \\
Q_{k} &= \text{Var}(w_k) = E[w_k w_k^T], \\
P_{k|k} &= (I - K_k H_k) P_{k|k-1}
\end{align*}
\]

where \( Q_k \) is defined in eqn. 32. Substituting eqns. 32–35 and the definition of \( v_k \) into eqn. 36, we get the following one-dimensional Kalman filter:

\[
\begin{align*}
\hat{x}_{k|k} &= \hat{x}_{k-1} + K_k r'_k \\
K_k &= \frac{P_{k|k-1}}{R'_k} \\
v_k &= \hat{x}_{k|k} + \frac{1}{T} (z_k - \hat{x}_{k|k}) \\
P_{k|k} &= (1 - K_k) P_{k|k-1}
\end{align*}
\]  

(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:

\[
\begin{align*}
|v_a| &\leq 9.0 \frac{\text{deg}}{T} \text{ degrees/second azimuth} \\
|v_e| &\leq 4.5 \frac{\text{deg}}{T} \text{ degrees/second elevation}
\end{align*}
\]  

(38)
The Kalman filter in eqn. 37 is optimal only when \( \epsilon_i \) 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 \( v_i \) 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 \( \epsilon_i \), for example. Each control surface characterizes 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 \( \epsilon_i = ZE \). Each entry in this linguistic matrix represents one FAM rule with \( \epsilon_i = ZE \) as the first antecedent term.

The entire FAM bank (including cross-sections for \( \epsilon_i \) 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 \( v_i \) for any set of crisp input values \( \epsilon_i, \epsilon_i, \) and \( \epsilon_{i-1} \).

Fig. 10 shows the control surfaces of the fuzzy controller for \( \epsilon_i = 0 \). 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 \( v_i \) against \( \epsilon_i \) and \( \epsilon_{i-1} \). 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. 10d 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

\[
v_i = \hat{v}_{i-1} + \frac{1}{T}(\epsilon_i + \hat{\epsilon_i})
\]

\[
\hat{\epsilon}_{i-1} = \epsilon_{i-1} + K_i r_i
\]

where \( r_i \) denotes Gaussian noise with variance given by eqn. 35. Combining these two equations gives the equation for the random control surface

\[
v_i = \bar{v}_{i-1} + \frac{1}{T}(\epsilon_i + \hat{\epsilon_i}) + K_i r_i
\]

(39)

At time \( k \), the noise term \( K_i r_i \) has variance

\[
\sigma_i^2 = K_i R_i
\]

(40)

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

\[
v_i = \bar{v}_{i-1} + \frac{1}{T}(\epsilon_i + \hat{\epsilon_i}) + \sigma_i r_i
\]

(41)

where \( r_i^2 \) 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 \( \sigma_i^2 \). For constant error \( \epsilon_i \), we can interpret eqn. 41 as a smooth control surface in \( R^1 \) defined by
$v_k = v_{k-1} + \frac{1}{T}(e_k + \dot{e}_k)$  \hspace{1cm} (42)

and perturbed at time $k$ by Gaussian noise with variance $\sigma^2$.

The model assumes fixed variance values

$R_k = \bar{R}$

$Q_k = \bar{Q} = \sigma^2 + \frac{1}{T^2}\sigma^2$  \hspace{1cm} (43)

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

$\lim_{k \to \infty} \sigma^2 = \sigma^2 = \frac{\bar{P}}{\bar{P} + \bar{R}}$  \hspace{1cm} (44)

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

$\psi_k = v_{k-1} + \frac{1}{T}(e_k + \dot{e}_k)$  \hspace{1cm} (45)

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

$P_{kk} = P_{kk-1} + Q_k = (1 - K_k)P_{kk-1} + Q_k = \frac{R_kP_{kk-1}}{P_{kk-1} + R_k} + Q_k$  \hspace{1cm} (46)

At steady-state, eqn. 46 is reduced to

$P = \frac{R\bar{P}}{\bar{P} + \bar{R}} + \bar{Q}$  \hspace{1cm} (47)

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

$\bar{Q} = \frac{-R\bar{P}}{\bar{P} + \bar{R}}$  \hspace{1cm} (48)
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 trapezoidal 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, \( R_k = P_{k-1} \) 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 \( \sigma^2_z \) to 0 for optimal performance. Fig. 16 shows the Kalman-filter controller's performance when \( \sigma^2_z = 4.0 \) 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 \( \sigma^2_z \), and increased the measurement uncertainty \( \sigma^2_n \). In Fig. 18, we did just the opposite, fixing \( \sigma^2_n \) and increasing \( \sigma^2_z \).

The Kalman-filter state equation (eqn. 29) contains the noise term \( \mathbf{w}_k \), whose variance we must assume. When \( \text{Var}(w) \) 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 \( \sigma^2_n \), the variance of the...
unmodelled-effects noise process. Recall from eqn. 31 that increasing $\sigma^2$ increases $Q_n = \text{Var}(e)$. 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 $\sigma^2$, and then gradually levels off.

We also tested 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 ($Z\overline{E}ZEZEZE$) 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 $F_1$. A manually designed FAM bank and 50 random target trajectories generated 19236 training vectors. Each product-space training vector $(e_1, e_2, n_1, n_2, n_3, n_4)$ defined a point in $R^4$.

Symmetry allowed us to reflect about the origin all sample vectors with negative errors $e$. We then trained 3000 synaptic quantisation vectors ($p = 3000$) 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 $F_1$ within-field synaptic connection strengths had diagonal elements $w_n = 2.9$ and off-diagonal elements $w_n = -0.1$. After training, we reflected the 3000 synaptic quantisation vectors about the origin to give 6000 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 $LN, LN$, $MN$, $MN$, $MN$, $MN$, $MN$, $MN$. 

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

a. azimuth position and error
b. elevation position and error
The sabotage rule ($Z\overline{E}ZEZEZE, LP$) replaces the steady-state FAM rule ($Z\overline{E}ZEZEZE$). The system quickly adjusts each time the sabotage rule is activated.
SN, ZE, SP, MP and LP. Therefore, the product space contained \( 7^5 = 2401 \) 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 antecedent 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. 22a shows the \( e_i = ZE \) cross-section of the original FAM bank used to generate the training samples. Fig. 22b shows the same cross-section of the DCL-estimated FAM bank. Fig. 23 shows the original and DCL-estimated control surfaces for constant error \( e_i = 0 \).

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 occurrence 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>
<thead>
<tr>
<th>Trajectory number</th>
<th>Azimuth original</th>
<th>Azimuth estimated</th>
<th>Elevation original</th>
<th>Elevation estimated</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.33</td>
<td>2.33</td>
<td>3.31</td>
<td>3.37</td>
</tr>
<tr>
<td>2</td>
<td>4.14</td>
<td>4.14</td>
<td>3.03</td>
<td>2.69</td>
</tr>
<tr>
<td>3</td>
<td>6.11</td>
<td>6.11</td>
<td>3.69</td>
<td>3.68</td>
</tr>
<tr>
<td>4</td>
<td>3.83</td>
<td>3.83</td>
<td>3.32</td>
<td>3.30</td>
</tr>
<tr>
<td>5</td>
<td>4.02</td>
<td>4.02</td>
<td>3.11</td>
<td>3.10</td>
</tr>
<tr>
<td>6</td>
<td>2.84</td>
<td>2.84</td>
<td>1.20</td>
<td>1.21</td>
</tr>
<tr>
<td>7</td>
<td>3.22</td>
<td>3.22</td>
<td>3.04</td>
<td>2.98</td>
</tr>
<tr>
<td>8</td>
<td>0.76</td>
<td>0.74</td>
<td>2.00</td>
<td>2.00</td>
</tr>
<tr>
<td>9</td>
<td>9.26</td>
<td>9.27</td>
<td>5.50</td>
<td>5.41</td>
</tr>
<tr>
<td>10</td>
<td>1.61</td>
<td>1.61</td>
<td>2.29</td>
<td>2.29</td>
</tr>
<tr>
<td>average</td>
<td>3.83</td>
<td>3.83</td>
<td>3.05</td>
<td>3.02</td>
</tr>
</tbody>
</table>

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


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 quantisa-

(49)

\[ P(V) = \int_V p(x) \, dx \]

\[ = \text{number of } m_i \in V \]

\[ \frac{1}{p} \]

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

10.2 Stochastic competitive learning algorithms

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 \( j \)th synaptic vector \( m_j(t) \) is closest to \( x(t) \), the \( j \)th 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 \( x(i) \), find the closest or 'winning' synaptic vector \( m_j(t) \)

\[ ||m_j(t) - x(i)|| = \min_{i} ||m_i(t) - x(i)|| \]

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.

INTELLIGENT SYSTEMS ENGINEERING AUTUMN 1992
10.4 Differential competitive learning (DCL)

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

\[ m_j = S_j(y_j) [S(x) - m_j] \]

(52)

or in vector notation

\[ m_i = \sum_j S_j(y_j) [S(x) - m_j] \]

(53)

where \( S(x) = (S_1(x_1), \ldots, S_n(x_n)) \) and \( m = (m_1, \ldots, m_n) \). \( m_j \) denotes the synaptic value between the \( j \)th neuron in input field \( F_x \) and the \( j \)th 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 \( S_j(y_j) \). \( m_i \) and \( \dot{S}_i(y_i) \) denote the time derivatives of \( m_i \) and \( S_j(y_j) \), synaptic and signal velocities. \( S_j(y_j) \) measures the competitive status of the \( j \)th 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

\[ S_j(y_j) = \frac{1}{1 + e^{-\gamma}} \]

(54)

for a constant \( \gamma > 0 \). The \( j \)th 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]

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

if the \( j \)th neuron wins

\[ m_j(t+1) = m_j(t) \]

(55)

if the \( j \)th neuron loses

\( \Delta S_j(y_j(t)) \) denotes the time change of the \( j \)th neuron’s competition signal \( S_j(y_j) \) in the competition layer \( F_y \).

\[ \Delta S_j(y_j(t)) = S_j(y_j(t + 1)) - S_j(y_j(t)) \]

(56)

We define the signum operator \( \text{sgn}(x) \) as

\[ \text{sgn}(x) = \begin{cases} 
1 & \text{if } x > 0 \\
0 & \text{if } x = 0 \\
-1 & \text{if } x < 0 
\end{cases} \]

(58)

\( \{c_i\} \) denotes a slowly decreasing sequence of learning coefficients, such as \( c_i = 0.1(1 - i/2000) \) for 2000 training samples. Stochastic approximation [8] requires a decreasing gain sequence \( \{c_i\} \) to suppress random disturbances and to guarantee convergence to local minima of mean-squared performance measures. The learning coefficients should decrease slowly

\[ \sum_{i=1}^{\infty} c_i = \infty \]

(59)

but not too slowly

\[ \sum_{i=1}^{\infty} c_i^2 < \infty \]

(60)

Harmonic-series coefficients, \( c_i = 1/i \), satisfy these constraints.

We approximate the competitive signal difference \( \Delta S_j \) as the activation difference \( \Delta y_j \)

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

(61)

\[ \Delta y_j(t) \]

(62)

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

\[ m_j(t + 1) = m_j(t) + \gamma \Delta y_j(t)[x(t) - m_j(t)] \]

(63)

We update the \( F_y \) neuronal activations \( y_j(t) \) with the additive model

\[ y_j(t + 1) = y_j(t) + \sum_i S_i(x_i(t))m_i(t) \]

\[ + \sum_i S_i(x_i(t))w_i \]

(64)

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

\[ \sum_i S_i(x_i(t))m_i(t) \]

(65)

Positive learning then tends to occur \( (\Delta m_i > 0) \) when \( x \) is close to the \( j \)th synaptic vector \( m_j \).

Since a binary threshold function approximates the output signal function \( S_j(y_j) \), the second sum in eqn. 64 sums over just the winning neurons; \( \Sigma_i \) \( w_i \) for all winning neurons \( y_i \).

The \( p \times p \) matrix \( W \) contains the \( F_y \) within-field synaptic connection strengths. Diagonal elements \( w_i \) 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.

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.