Neural Fuzzy Motion Estimation and Compensation

Hyun Mun Kim and Bart Kosko

***Abstract*—Neural fuzzy systems can improve motion estimation and compensation for video compression. Motion estimation and compensation are key parts of video compression. They help remove temporal redundancies in images. But most motion estimation algorithms neglect the strong temporal correlations of the motion field. The search windows stay the same through the image sequences and the estimation needs heavy computation. A neural vector quantizer system can use the temporal correlation of the motion field to estimate the motion vectors. First- and second-order statistics of the motion vectors give ellipsoidal search windows. This algorithm reduced the search area and entropy and gave clustered motion fields. Motion-compensated video coding further assumes that each block of pixels moves with uniform translational motion. This often does not hold and can produce block artifacts. We use a neural fuzzy system to compensate for the overlapped block motion. This fuzzy system uses the motion vectors of neighboring blocks to map the prior frame's pixel values to the current pixel value. The neural fuzzy system used 196 rules that came from the prior decoded frame. The fuzzy system learns and updates its rules as it decodes the image. The fuzzy system also improved the compensation accuracy. The Appendix derives both the fuzzy system and the neural learning laws that tune its parameters.**

I. MPEG Standards for Video Compression

THIS PAPER presents new schemes for motion estimation and compensation based on neural fuzzy systems. Motion estimation and compensation help compress video images because they can remove temporal redundancies in the image data. Motion estimation schemes often neglect the strong temporal correlations of the motion field. The search windows remain the same through the image sequences and the estimation may need heavy computation. We designed an unsupervised neural system that uses the temporal correlation of the motion field to estimate the motion vectors and to reduce the entropy of source coding.

Motion-compensated video coding uses the motion of objects in the scene to relate the intensity of each pixel in the current frame to the intensity of some pixel in a prior frame. It predicts the value of the entire current block of pixels as the value of a displaced block from the prior frame. It also assumes that each block of pixels moves with uniform translational motion. This assumption often does not hold and can produce block artifacts. We designed a neural-fuzzy system that uses motion vectors of neighboring blocks to improve the compensation accuracy.

Manuscript received February 7, 1996; revised April 2, 1997. The associate editor coordinating the review of this paper and approving it for publication was Dr. Michael Zervakis.

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

Publisher Item Identifier S 1053-587X(97)07359-5.

Fig. 1 shows the typical structure of the Moving Picture Experts Group (MPEG) encoder. The MPEG standard depends on two basic algorithms. Motion-compensated coding uses block-based motion vector estimation and compensation to remove temporal redundancies. Block discrete cosine transforms reduce spatial redundancy. The MPEG standard defines and forms the bit-stream syntax to achieve interoperability among different blocks. Standards improve interoperability among video systems and help speed the development of high-volume low-cost hardware and software solutions [7].

Most current research in video compression seeks new algorithms or designs high-performance encoders that work with existing standards. These standards give a bit-stream syntax and a decoder and thus allow some flexibility in how one designs a compatible encoder. The MPEG standards do not give a motion estimation algorithm or a rate-control mechanism. This leaves manufacturers free to use the flexibility of the syntax.

Our neural quantizer system uses the first- and second-order statistics of the motion vectors to give ellipsoidal search windows. This method reduces the search area and gives clustered motion fields. It reduces the computation for motion estimation and decreases the entropy that the system needs to transmit the entropy-coded motion vectors.

We also propose a neural fuzzy overlapped block motion compensation (FOBMC) scheme for motion compensation. Fuzzy systems use a set of if-then rules to map inputs to outputs. Neural fuzzy systems learn the rules from data and tune the rules with new data. The FOBMC estimates each pixel intensity using the block-based motion vectors available to the decoder. The fuzzy system uses the motion vectors of neighboring blocks to map the prior frame's pixel values to the current pixel value. The 196 rules come from the prior decoded frame. The neural fuzzy system tunes its rules as it decodes the image. The fuzzy system defined a nonlinear "black box" function approximator that improved the compensation accuracy. The Appendix derives the supervised learning laws that tune the parameters of the fuzzy system.

II. Motion Estimation and Compensation

This section briefly reviews the standard techniques of motion estimation and compensation.

A. Motion Estimation

Motion estimation occurs in many areas of image processing. Video coding schemes often exploit the high temporal redundancy between successive frames in a sequence by predicting the current frame from the prior frame based on an estimated motion field. Then, the schemes code and

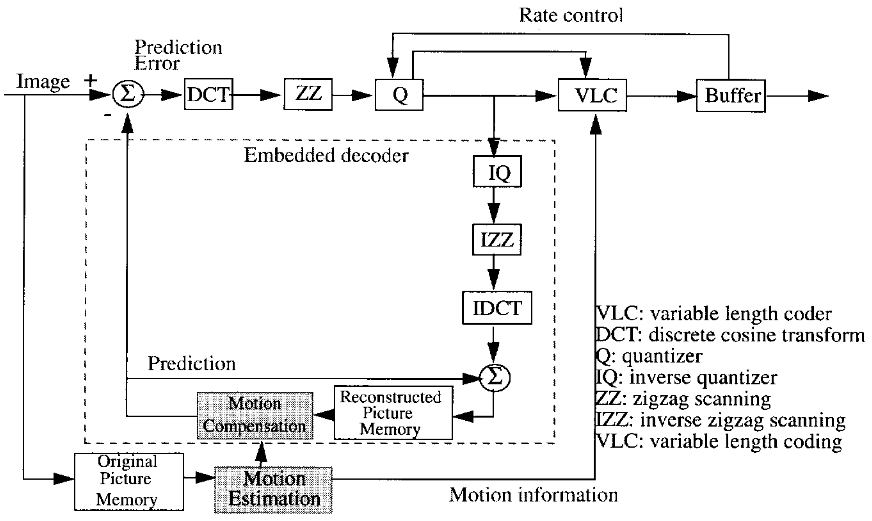


Fig. 1. Block diagram of the Moving Picture Experts Group (MPEG) encoder.

transmit the prediction error image. The schemes may also need to transmit the motion field if the motion estimation algorithm uses information that the receiver does not have. The prediction error often contains much less information than does the original frame if the motion estimates are accurate.

The MPEG standard uses three types of pictures that depend on the mode of motion prediction. The intra (I) picture serves as the reference picture for prediction. Block discrete cosine transforms (DCT's) code the intra pictures, and no motion estimation prevents long range error propagation. Coding the predicted (P) pictures uses forward prediction of motion. We divide each image into macroblocks of size 16×16 pixels and search blocks of the same size in the prior reference I frame or P frame.

A second type of picture is the bidirectional interpolated (B) picture. We perform both forward and backward motion prediction with respect to the prior or future reference I or P frames. Averaging these two predictions gives the interpolation. Bidirectional interpolation can handle just covered or uncovered areas since the system cannot predict an area just uncovered from the past reference. The system can still predict the areas from the future reference frame. The one that has the smallest mean-square error among the forward, backward, and interpolated prediction gives the best motion prediction. The encoding and decoding orders of video sequences can differ from that of the original sequences due to the three types of frames. Therefore, the $(J+1)$th and $(J+2)$th frames follow the $(J+3)$th frame as in Fig. 2. The decoder needs to reorder the frames to display them.

The two main types of motion estimation use pel-recursive algorithms or block matching algorithms [23]. Pel-recursive algorithms predict the motion field at the decoder based on how neighboring pixels decoded in the current frame relate to pixels in the prior frame. Block-based motion estimation derives from the need for relatively accurate motion fields while keeping low the side information one needs to represent the motion vectors. Image sequence coding often uses full-search block matching among the block-based motion estimation techniques. This scheme is simple and easy to implement in

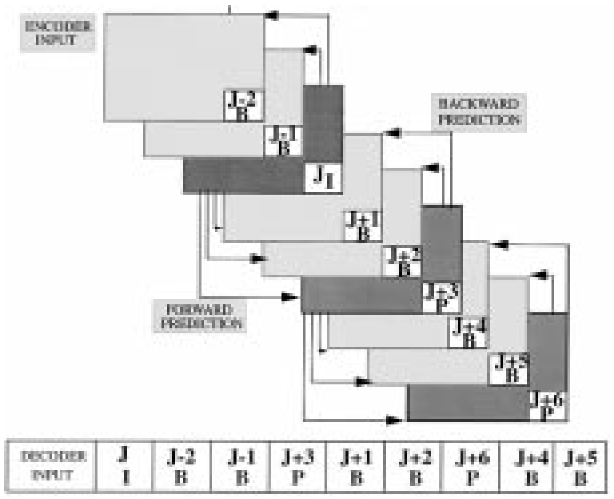


Fig. 2. Ordering of video sequences in MPEG.

hardware. Exhaustive search within a maximum displacement range leads to the absolute minimum for the energy of the prediction error and is optimal in this sense. This acts as a type of codeword search in vector quantization (VQ) [10]. VQ finds a codeword from the codebook that minimizes some criteria such as mean-squared error (MSE). It locates the minimum for the energy of the prediction error and tends to have a heavy computational load.

Accurate modeling of the motion field becomes more important under the constraint of a very low bit rate [21]. Here, however, full block search technique tends to produce noisy motion fields that do not correspond to the true 2-D motion in the scene. Noises in real video images can also affect the locations of the smallest distortion. Noise gives rise to a blocky effect in motion-compensated prediction images and has no physical meaning in terms of the estimated motion vectors. These artificial discontinuities lead to an increase of the side information to transmit the entropy-coded motion vectors. A decrease in this side information while keeping the same accuracy for the motion fields improves low bit rate applications [5]. Therefore, we propose a new adaptive scheme to estimate motion vectors that have spatial consistency. The scheme uses the temporal correlation of the motion field to reduce the computation and to give a clustered motion field.

B. Motion Compensation

Motion-compensated video coding relates the intensity of each pixel in the current frame to the intensity of some pixel in a prior frame. It links these pixels by predicting the motion of objects in the scene. However, the transmission overhead needed to inform the decoder of the true motion at every pixel in the image may far outweigh the gains of motion compensation. Motion compensation assigns only one motion vector to each square (often a 16×16-pixel) block in the frame. The encoder selects this motion vector to minimize the mean-squared prediction error. It predicts the value of the entire current block of pixels by the value of a displaced block from the prior frame. Therefore, it assumes that each block of pixels moves with uniform translational motion. This assumption often does not hold and can produce block artifacts.

Orchard and Sullivan [22], proposed overlapped block motion compensation (OBMC) to overcome this problem. This linear scheme estimates each pixel intensity using the block-based motion vectors available to the decoder. It predicts the current frame of a sequence by repositioning overlapping blocks of pixels from the prior frame. Then, it computes the coefficients of the linear estimator by solving the normal equations of least squares, but this scheme has at least two problems. The coefficients computed from the training sequences may not work well for the test sequences, and the coefficient calculation is computationally heavy and the decoder must store these values.

We propose a *fuzzy* overlapped block motion compensation (FOBMC). A fuzzy rule-based system estimates pixel intensities using the block-based motion vectors available to the decoder. Fuzzy systems $F\colon R^n \to R$ compute a model-free conditional mean $E[Y|X]$ [16], [20], [24] and thus compute a least-mean-square nonlinear estimate of the random variable Y based on our knowledge of the random vector X. The FOBMC system uses the conditional mean to predict each pixel intensity. It uses the motion vectors of neighboring blocks to map the prior frame's pixel values to the current pixel value. This has at least two advantages. The rules come from the prior decoded frame, and the neural fuzzy system tunes its rules as it decodes the image. Simulation results showed that the FOBMC improved the compensation accuracy. This method also shows how to insert expert knowledge into the compensation process.

III. Additive Fuzzy Systems and Learning

This section reviews the standard additive model (SAM) fuzzy system and how SAM's learn with and without supervision. The Appendix derives the ratio structure of the SAM and supervised learning laws that tune its parameters.

A. Additive Fuzzy Systems

A fuzzy system $F\colon R^n \to R$ stores m rules of the word from "If $X = A_j$, then $Y = B_j$" or the patch form $A_j \times B_j \subset X \times Y = R^n \times R$. The if-part fuzzy sets $A_j \subset R^n$ and then-part fuzzy sets $B_j \subset R$ have set functions $a_j\colon R^n \to [0, 1]$ and $b_j\colon R \to [0, 1]$. The system can use the joint set function $a_j\colon R^n \to [0, 1]$ [11] or some factored form such as $a_j(x) = a_j^1(x_1) \cdots a_j^n(x_n)$, or $a_j(x) = \min[a_j^1(x_1), \cdots, a_j^n(x_n)]$, or any other conjunctive form for input vector $x = (x_1, \cdots, x_n) \in R^n$.

An additive fuzzy system [17]–[20] sums the "fired" then part sets B'_j

$$B = \sum_{j=1}^{m} B'_j = \sum_{j=1}^{m} a_j(x)B_j. \tag{1}$$

Fig. 3 shows the parallel fire-and-sum structure of the SAM system. The additive system is standard if the if-part value $a_j(x)$ scales the then-part set B_j to give the fired set $B'_j\colon B'_j = a_j(x)B_j$. These systems can uniformly approximate any continuous (or bounded measurable) function f on a compact domain [17].

Fig. 4 shows how three rule patches can cover part of the graph of a scalar function $f\colon R \to R$. The patch cover shows that all fuzzy systems $F\colon R^n \to R^p$ suffer from *rule explosion* in high dimensions. A fuzzy system F needs on the order of k^{n+p-1} rules to cover the graph and thus to approximate a vector function $f\colon R^n \to R^p$. Optimal rules can help deal with the exponential rule explosion. Lone or local mean-squared optimal rule patches cover the extrema of the approximand f [18]. They "patch the bumps." Better learning schemes move rule patches to or near extrema and then fill in between extrema with extra rule patches if the rule budget allows.

The scaling choice $B'_j = a_j(x)B_j$ leads to a *standard additive model* (SAM). Appendix A shows that taking the centroid of B in (1) gives [16]–[20] the SAM ratio

$$F(x) = \frac{\sum_{j=1}^{m} w_j a_j(x) V_j c_j}{\sum_{j=1}^{m} w_j a_j(x) V_j} \tag{2}$$

where

- w_j positive rule weight;
- V_j nonzero volume or area of then-part set B_j;
- c_j centroid of B_j or its center of mass.

Therefore, a SAM has the form of a convex sum $F(X) = \sum_{j=1}^{m} p_j(x)c_j$ for convex coefficients $p_j(x) = w_j a_j(x) V_j / \sum_{k=1}^{m} w_k a_k(x) V_k$ so that $p_j(x) \geq 0$ and $\sum_{j=1}^{m} p_j(x) = 1$ for each $x \in R^n$.

The SAM theorem (2) implies that the fuzzy structure of the then-part sets B_j does not matter for a first-order function approximator. The ratio depends on just the rule weight w_j and the volume V_j and location or centroid c_j of the then-part sets B_j. Our SAM has then-part sets of the same area and weight $V_1 = \cdots = V_m > 0$ and $w_1 = \cdots = w_m > 0$. Therefore, the volume terms V_j and the weight w_j cancel from (2). We need pick only the scalar centers c_j to define the B_j sets. The then-part structure does affect the second-order structure or uncertainty of the output $F(x)$ [19]–[20].

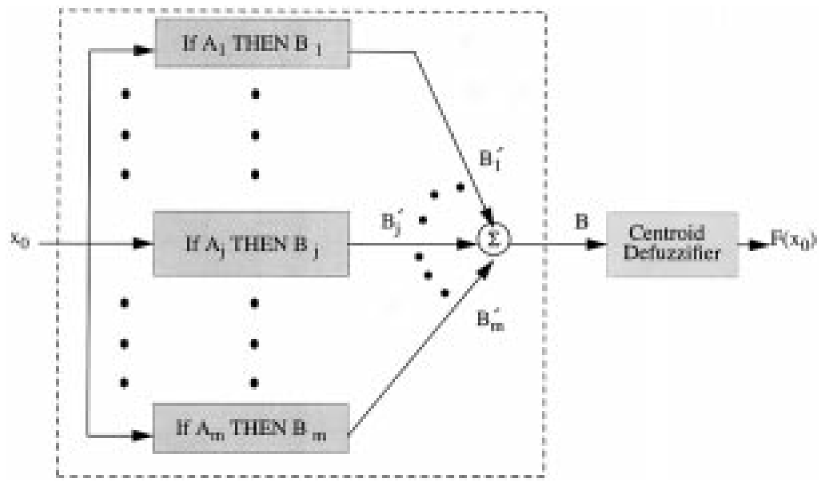


Fig. 3. Architecture of an additive fuzzy system F: $R^n \to R^p$ with m rules. Each input $x_0 \in R^n$ enters the system F as a numerical vector. At the set level x_0 acts as a delta pulse $\delta(x - x_0)$ that combs the if-part fuzzy sets A_j and gives the m set values $a_j(x_0)$ $a_j(x_0) = \int_{R^n} \delta(x - x_0) a_j(x)\,dx$. The set values "fire" the then-part fuzzy sets B_j to give B_j'. A standard additive model (SAM) scales each B_j with $a_j(x)$. Then the system sums the B_j' sets to give the output "set" B. The system output $F(x_0)$ is the centroid of B.

B. Learning in SAM's: Unsupervised Clustering and Supervised Gradient Descent

A fuzzy system learns if and only if its rule patches move or change shape in the input–output product space $X \times Y$. Learning might change the centers or widths of triangle or trapezoidal or bell-curve sets. These changing sets then change the shape or position of the Cartesian rule patches built out of them. The mean-value theorem and the calculus of variations show [18] that optimal lone rules cover the extrema or bumps of the approximand. Good learning schemes [3], [4] tend to quickly move rules patches to these bumps and then move extra rule patches between them as the rule budget allows. Hybrid schemes use unsupervised clustering to learn the first set of fuzzy rule patches in position and number and to initialize the gradient descents of supervised learning.

Learning changes system parameters with data. Unsupervised learning amounts to blind clustering in the system product space $X \times Y$ to learn and tune the m fuzzy rules or the sets that compose them. Then, k quantization vectors $q_j \in X \times Y$ move in the product space to filter or approximate the stream of incoming data pairs $[x(t), y(t)]$ or the concatenated data points $z(t) = [x(t)|y(t)]^T$. The simplest form of such *product space clustering* [15] centers a rule patch at each data point and, thus, puts $k = m$. In general, both the data and the quantizing vectors greatly outnumber the rules, and therefore, $k \gg m$.

A natural way to grow and tune rules is to identify a rule patch with the uncertainty ellipsoid [2], [3] that forms around each quantizing vector q_j. The ellipsoid stems from the inverse of the vector's positive definite covariance matrix K_j. Then, sparse or noisy data grows a patch larger and, thus, a less certain rule than does denser or less noisy data. Unsupervised competitive learning [15] can learn these ellipsoidal rules in three steps:

$$\|z(t) - q_j(t)\| = \min\left[\|z(t) - q_1(t)\|, \cdots, \|z(t) - q_k(t)\|\right] \quad (3)$$

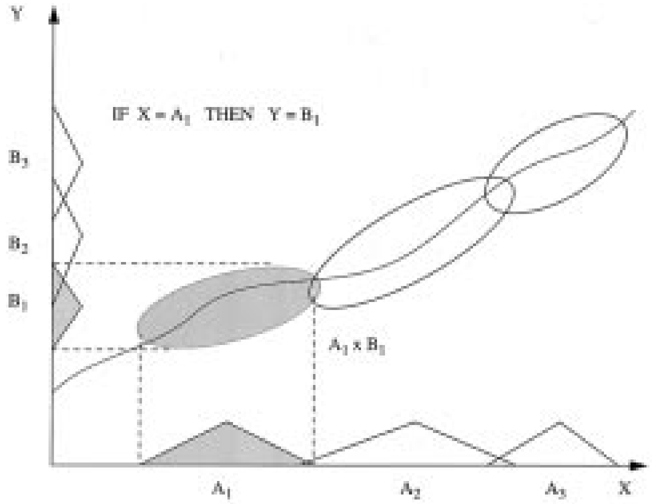


Fig. 4. Each fuzzy rule defines a Cartesian-product patch or fuzzy subset of the input–output state space. The fuzzy system approximates a function as it covers its graph with rule patches. Lone optimal rule patches cover extrema.

$$q_i(t+1) = \begin{cases} q_j(t) + \mu_t[z(t) - q_j(t)] & \text{if } i = j \\ q_i(t) & \text{if } i \neq j \end{cases} \quad (4)$$

$$K_i(t+1) = \begin{cases} K_j(t) + v_t[\{z(t) - q_j(t)\}T\{z(t) - q_j(t)\} - K_j(t)] & \text{if } i = j \\ K_i(t) & \text{if } i \neq j \end{cases} \quad (5)$$

for the Euclidean norm $\|z\|^2 = z_1^2 + \cdots + z_{n+p}^2$.

The first step (3) is the competitive step. It picks the nearest quantizing vector q_j to the incoming data vector $z(t)$ and ignores the rest. Some schemes may count nearby vectors as lying in the winning subset. We used just one winner per datum. This correlation matching approximates a great deal of the competitive dynamics of nonlinear neural networks. The second step updates the winning quantization or "synaptic" vector and drives it toward the centroid of the sampled data pattern class [14]. The third step updates the covariance matrix of the winning quantization vector. We initialize the quantization vector with sample data $[q_i(0) = z(i)]$ to avoid skewed groupings and to initialize the covariance matrix with small positive numbers on its diagonal to keep it positive definite. Then, projection schemes [2]–[4] can convert the ellipsoids into coordinate fuzzy sets. Other schemes can use the unfactored joint set function directly and preserve correlations among the input components [11]. Supervised learning can also tune the eigenvalue parameters of the rule ellipsoids.

Supervised learning changes SAM parameters with error data. The error at each time t is the desired system output minus the actual SAM output $\varepsilon_t = d_t - F(x_t)$. Unsupervised learning uses the blind data point $z(t)$ instead of the desired or labeled value d_t. The teacher or supervisor supervises the learning process by giving the desired value d_t at each training time t. Most supervised learning schemes perform stochastic gradient descent on the squared error and do so through iterated use of the chain rule of differential calculus. Appendix B derives the supervised learning laws that tune the parameters in (2). We do not know in advance which parameters to tune or which mix of parameters to tune.

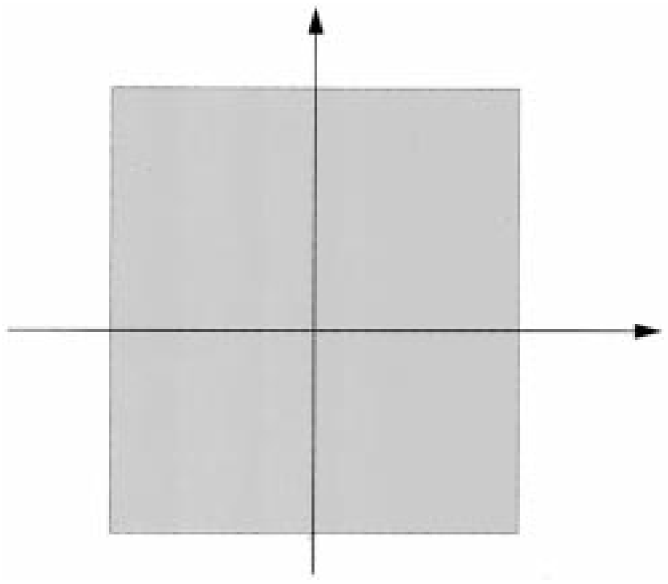
IV. Motion Estimation Using Adaptive Vector Quantization

Motion vector estimation removes the temporal correlation of video images. Block matching methods often neglect the fact that the motion field itself has strong spatial and temporal correlation. We propose an unsupervised neural system that uses the temporal correlation of the motion field to estimate the motion vectors. The neural system acts as an adaptive vector quantizer.

A. Competitive AVQ Algorithm for Local Means and Covariances

First- and second-order statistics of the motion vectors combine to pick the search windows. An *adaptive vector quantization* (AVQ) determines the local means and covariances of the motion vectors for the prior frame. The covariance matrix again defines an ellipsoid [2]. The ellipsoids from the motion vectors of the prior frame pick the search windows for the current frame based on the temporal correlation in the motion field. The ellipsoidal windows give a clustered motion field that has lower computational load than that of the full search method. Fig. 5 shows the search windows for full-search block matching and the ellipsoidal method. The ellipsoidal window uses the spatial consistency of the motion field.

Define each frame of an image sequence on a 2-D rectangular lattice S of pixels with members $s = (x, y)^T$. Let $I_k(s)$ stand for the intensity at pixel s of frame k of the sequence. Let $\tilde{I}_k(s)$ stand for the pixel intensity of the corresponding decoded frame. Let $v = (\Delta x, \Delta y)^T$ stand for a motion vector and v_s^k stand for a motion vector for pixel s of frame k. We call a set of motion vectors $\{v_s^k\}_{s \in S}$ for all pixels in the lattice a pixel motion field and write the entire field as V_S^k. Let B stand for a partition of S into a lattice of blocks of width W and height H. Let $b = (x'', y'')^T$ stand for a block from B,


(a)

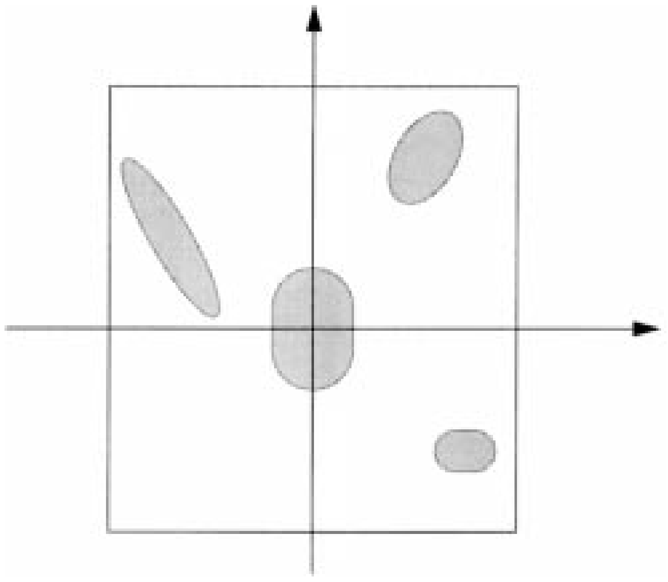

(b)

Fig. 5. (a) Search window for full-search block matching. Such window involves heavy computation and tends to produce noisy motion fields. (b) Ellipsoidal search window gives a smooth and meaningful motion field. The search window hops about more in the presence of noisy motion vectors and, thus, has a larger "error" ball. The covariance matrix K_j measures this error ball.

and let $s' = (x', y')^T$ stand for a pixel position within a block. Define a 2×2 diagonal matrix A with diagonal elements W and H such that $s = s' + Ab$ with $x' = x(\text{modulo})W$ and $y' = y(\text{modulo})H$. Let v_b^k stand for a motion vector for block b in frame k. We call the set of motion vectors $\{v_b^k\}_{b \in B}$ a block motion field for frame k and write it as V_B^k.

Vector quantizers can use competitive learning to estimate the local mean and conditional covariance matrix for each pattern class [16]–[20]. Motion vector $(\Delta x, \Delta y)$ drives the competitive learning process. We first form the concatenated vector $v = (\Delta x, \Delta y)^T$ in the product space $R \times R$. Then we assign p quantization vectors m_j to the same product space.

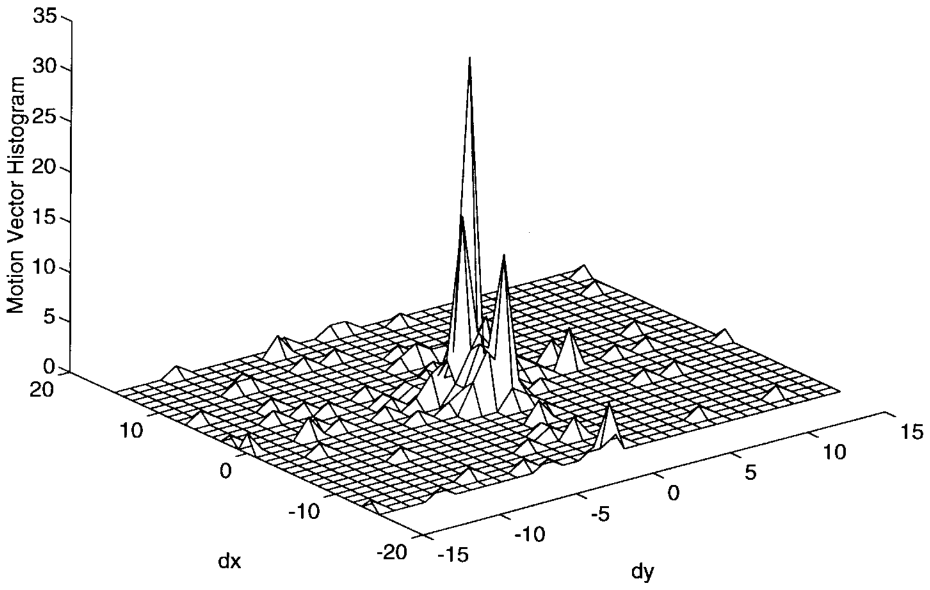


Fig. 6. Motion vector histogram from the full-search algorithm for the "Miss America" image. We use these motion vectors to estimate the local means and covariances.

These vectors learn or adapt as they code for the local sample statistics and give rise to an AVQ. The AVQ vectors m_j act as the synaptic fan-in columns of a connection matrix in a neural network with p competing neurons [14]. The points m_j learn if and only if they move in the product space. The points m_j track the distribution of the incoming motion vectors v's and tend to be sparse or dense where the v's are sparse or dense.

Each AVQ vector m_j converges to the local centroid of the motion vectors v's. Therefore the AVQ vectors m_j estimate the local first-order moments of some unknown probability density $p(v)$ that generates the motion vectors v's. The AVQ algorithm is a form of nearest-neighbor pattern matching or K-means clustering. The m_j are random estimators since the motion vectors v's are random. The AVQ point m_j hops about more in the presence of noisy or sparse data and thus has a larger "error" ball. The covariance matrix K_j measures this error ball. The competitive AVQ algorithm in (7)–(9) below updates the positive-definite matrix K_j. The inverse matrix K_j^{-1} defines an ellipsoid E in the product space $R \times R$ as the locus of all points v that obey

$$e^2 = (v - m_j)^T K_j^{-1}(v - m_j) \tag{6}$$

for centering constant $e > 0$. Then the jth ellipsoid defines the jth search window.

The following scheme describes the competitive adaptive vector quantization algorithm for local means and covariances.

1) Initialize cluster centers from sample data $m_i(0) = v(i)$ for $i = 1, \cdots, p$.
2) Find the "winning" or closest synaptic vector $m_j(k)$ to sample motion vector $v(k)$

$$\|m_j(k) - v(k)\| = \min_i \|m_i(k) - v(k)\| \tag{7}$$

where $\|\cdot\|$ denotes the Euclidean norm $\|v\|^2 = \Delta x^2 + \Delta y^2$.

3) Update the winner $m_j(k)$

$$m_j(k+1) = \begin{cases} m_j(k) + c_k[v(k) - m_j(k)] & \text{if the } j\text{th neuron wins} \\ m_j(k) & \text{if the } j\text{th neuron loses} \end{cases} \tag{8}$$

and update its covariance estimate $K_j(k)$ as shown in (9), at the bottom of the page.

The sequences of learning coefficients $\{c_k\}$ and $\{d_k\}$ should decrease slowly [20] in the sense of $\sum_{t=1}^{\infty} c_k = \infty$ but not too slowly in the sense of $\sum_{t=1}^{\infty} c_k^2 < \infty$. In practice, $c_k \approx 1/k$. The covariance coefficients obey a like constraint as in our choice of $d_k = 0.2[1 - k/(1.2N)]$ where N is the total number of data points.

B. Motion Estimation Using Adaptive Vector Quantization

This section shows how we obtain a smooth and meaningful motion field. The fact that the motion field has strong spatial and temporal correlation [12] motivates our algorithm. Fig. 6 shows the motion vector histogram we obtain if we use the full-search algorithm. It shows the scattered motion field. The full search with the MSE criterion locates just the minimum of the prediction error.

We assume that the objects in motion move with an almost constant velocity within a few consecutive frames. Therefore, we model the motion vector field v_B^k in frame k using the motion vector field v_B^{k-1} in frame $k-1$

$$v_B^k = v_B^{k-1} + n_B^{k-1} \tag{10}$$

$$K_j(k+1) = \begin{cases} K_j(k) + d_k[\{v(k) - m_j(k)\}\{v(k) - m_j(k)\}^T - K_j(k)] & \text{if the } j\text{th neuron wins} \\ K_j(k) & \text{if the } j\text{th neuron loses.} \end{cases} \tag{9}$$

where n_B^{k-1} is the noise field in frame $k-1$. The noise may come from an acceleration or deceleration of the block of pixels in consecutive image frames. We assume it has a zero mean and finite variance. The noise may result from many sources: poor camera handling, misfocus, nonideal film materials, camera zooming, scene cut, or sudden light intensity changes [12].

AVQ estimates the motion vector field v_B^{k-1}. The area of the search window depends on this motion vector field so that learning can reduce the search windows with centers that shift and window sizes that vary. However, the full-search method always chooses a fixed-size search window with the center at the origin. It simply locates the energy minimum of the prediction error and does so without outside information.

C. Unsupervised Weighting for AVQ Algorithm

We can also add an unsupervised weighting scheme to the AVQ algorithm. We need to minimize the calculation overhead since we want to decrease the complexity of the full-search algorithm. We add a simple weighting scheme to the AVQ algorithm to help achieve this.

The search window hops about more in the presence of noisy motion vectors and, therefore, has a larger "error" ball. This presents a problem since noisy motion vectors will have a larger search window than smaller and more certain motion vectors have. We give more weight to the window that has the closet center to the zero point.

The unsupervised weighting scheme has a simple form. We use $e = 2$ in (6) for the window that has the closet center to the zero point and use $e = 1$ in (6) for the other windows.

D. Complexity Analysis

The AVQ algorithm has less complexity than the full-search technique. Consider the number of additions and multiplications. We need $256 \times 2 = 512$ additions for each search point. Therefore, we need $512 \times 961 = 492\,032$ additions for each macroblock using the full block search technique if we assume a 31×31 window size. This gives $492\,032 \times 256 = 125\,960\,192$ additions for the full block search technique.

Suppose we have five ellipsoids for the AVQ algorithm. We need $3 \times 5 + 4 = 19$ additions and $2 \times 5 = 10$ multiplications to find the winning vector in (7). We need four additions and two multiplications to update $m_j(k)$ and eight additions and eight multiplications to update $K_j(k)$. There are 256 motion vectors if we assume an 256×256 image. We first use five motion vectors to initialize the mean and covariance and then use the other 251 motion vectors to update them. Therefore we need $31 \times 251 = 7781$ additions and $20 \times 251 = 5020$ multiplications to find the local means and covariances. We also need to compute (6) to pick the search points. This takes eight additions and 12 multiplications. Therefore we need $8 \times 961 = 7688$ additions and $12 \times 961 = 11\,532$ multiplications to pick search points inside the window. Therefore the extra calculations using AVQ come to 15 469 additions and 16 552 multiplications. However, we use this search window for the whole image sequence until intraframe coding resets it. Therefore if we divide the overhead by three then this gives 5156 additions and 5517 multiplications. The number of additions in the AVQ is just $125\,960\,192 \times 0.25 = 31\,490\,048$. This involves an overhead of 5156 additions and 5517 multiplications if we search only 25% of the window.

TABLE I
COMPLEXITY ANALYSIS OF MOTION ESTIMATORS

	Full block search	AVQ search
number of additions	125,960,192	31,490,048
search overhead	none	5,156 additions and 5,517 multiplications
unsupervised weighting overhead	none	3 additions and 4 multiplications

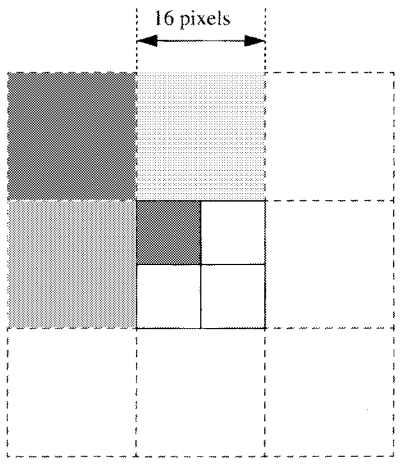


Fig. 7. Eight nearest neighbors of b assigned for optimal motion vectors for the center block b. The pixels in the second quadrant of b have four motion vectors selected for the shaded blocks.

Unsupervised learning can decide which window has the closet center to the zero point. It needs nine additions and ten multiplications if we assume five search windows. If we divide this by three, then we get only three additions and four multiplications. Table I summarizes these results.

V. FUZZY OVERLAPPED BLOCK MOTION COMPENSATION

This section shows how a "fuzzy" conditional mean can predict each pixel intensity. Standard motion compensation views the sent motion vectors as fixing a motion field for the decoder. This assumes that each block of pixels moves with uniform translational motion. This assumption often does not hold and produces block artifacts.

Orchard and Sullivan [22] have viewed the sent motion vectors as giving information about an underlying random motion field. They described this information with an inferred probability density that models the decoder uncertainty about the true motion at each pixel given the sent data. Then, motion compensation predicts the intensity of each pixel with respect to this inferred probability density.

Motion-compensated coding somehow represents a motion to create a prediction $\hat{I}_k(s)$ for each pixel s and then encodes

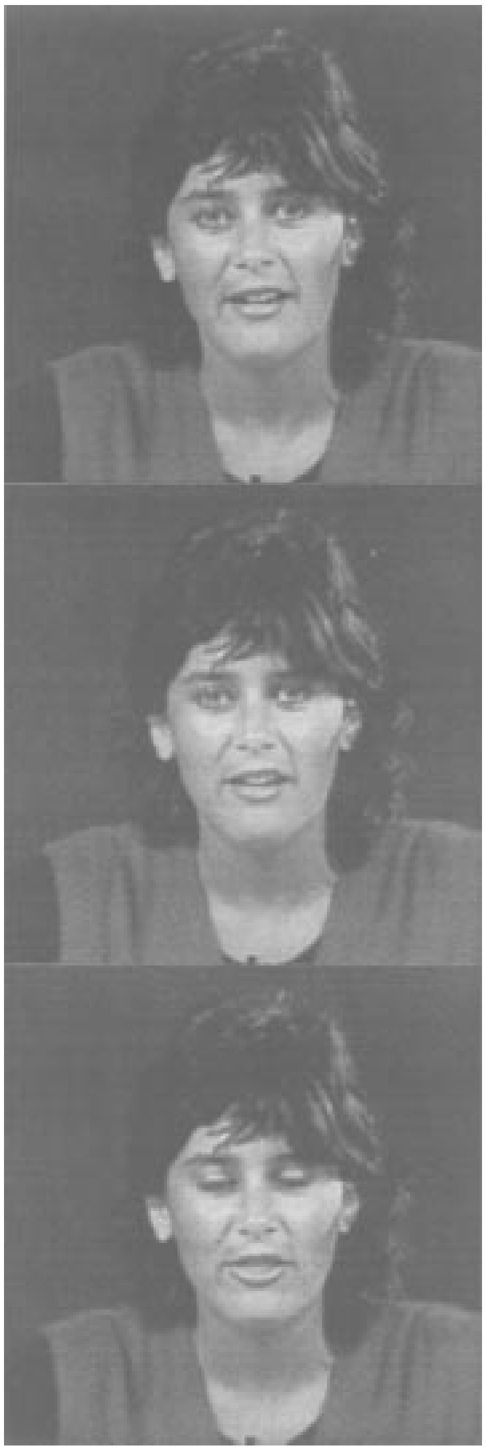


Fig. 8. "Miss America" image. Top: First frame. Middle: Fourth frame. Bottom: Seventh frame.

the compensated frame difference (CFD)

$$\mathrm{CFD}_k(s) = I_k(s) - \hat{I}_k(s). \tag{11}$$

We often send only a block motion field V_B^k rather than an exact pixel motion field V_S^k. The standard blockwise motion compensation approach forms the prediction for all pixels s in a block $b \in B$ by using the same encoded motion vector v_b^k

$$\hat{I}_k(s) = \tilde{I}_{k-1}(s + v_b^k). \tag{12}$$

Estimation theory can view any data D_k that the decoder received prior to the decoding of frame k as a source of information about the true motion field. This information allows us to define an *a posteriori* probability density function $f_s(v|D_k)$ that gives the probability that v is the correct vector to apply at s given all information D_k that the decoder received. Block motion compensation deals with the case $D_k = V_B^k$. Orchard and Sullivan [22] define ideal motion compensation as an optimal estimator of pixel intensity with respect to the probability density $f_s(v|D_k)$. Therefore the minimum mean-squared error estimate is the conditional expected value [22]

$$\hat{I}_k(s) = \int_v f_s(v|D_k)\tilde{I}_{k-1}(s + v)\, dv \tag{13}$$

but a general optimal solution to (13) may be quite complex. Orchard and Sullivan proposed an optimal *linear* solution called overlapped block motion compensation (OBMC) to reduce this complexity [22].

A. Overlapped Block Motion Compensation

Orchard and Sullivan proposed OBMC to simplify the solutions to (13). They used linear filtering (Wiener filter) for (13) and an ordered set of integer displacements $M_s^k = \{v_s^k(i)\}_{i \in I}$:

$$\hat{I}_k(s) = \sum_{i \in I} h_s(i, D_k)\tilde{I}_{k-1}[s + v_s^k(i)] \tag{14}$$

where $\{h_s(i, D_k)\}_{i \in I}$ is a set of weights for the displacements $M_s^k = \{v_s^k(i)\}_{i \in I}$.

Rule-based motion compensation often restricts M_s^k to consist of just one vector $v_s^k(0) = v_b^k$ and $h_s(0, D_k) = 1$. This gives a probability density function that assigns probability one to v_b^k and probability zero to all other vectors. It assumes the correct vector applied at pixel s is v_b^k with certainty. The sum (14) improves over (12) since the new model the does not assume that the coded block motion vector field can pick the true pixel-by-pixel motion vector field.

OBMC defines M_s^k to include motion vectors from blocks in some neighborhood of s instead of restricting M_s^k to consist of the single vector v_b^k. It assigns a set of motion vectors M_s^k for blocks in the neighborhood of b to predict $\hat{I}_k(s' + Ab)$. The OBMC scheme computes the mean-squared-error prediction of weight vector $h_{s'}$ by estimating the relevant cross correlations. The weight vector $h_{s'}$ consists of the elements $\{h_s(i, D_k)\}_{i \in I}$. The cross correlation vector for pixel location s' between the prediction pixel intensities and the true pixel intensity has the form

$$\gamma_{s'}(i) = E\{I_k(s' + Ab)\tilde{I}_{k-1}[s' + Ab + v_s^k(i)]\} \tag{15}$$

for each $v_s^k(i) \in M_s^k$ and where $E\{\cdot\}$ stands for the expectation operator. The autocorrelation matrix has the form

$$\Phi_{s'}(i, j) = E\{\tilde{I}_{k-1}[s' + Ab + v_s^k(i)]\tilde{I}_{k-1}[s' + Ab + v_s^k(j)]\} \tag{16}$$

with the finite variance

$$\theta_{s'}(i) = E\{I_k^2(s' + Ab)\}. \tag{17}$$

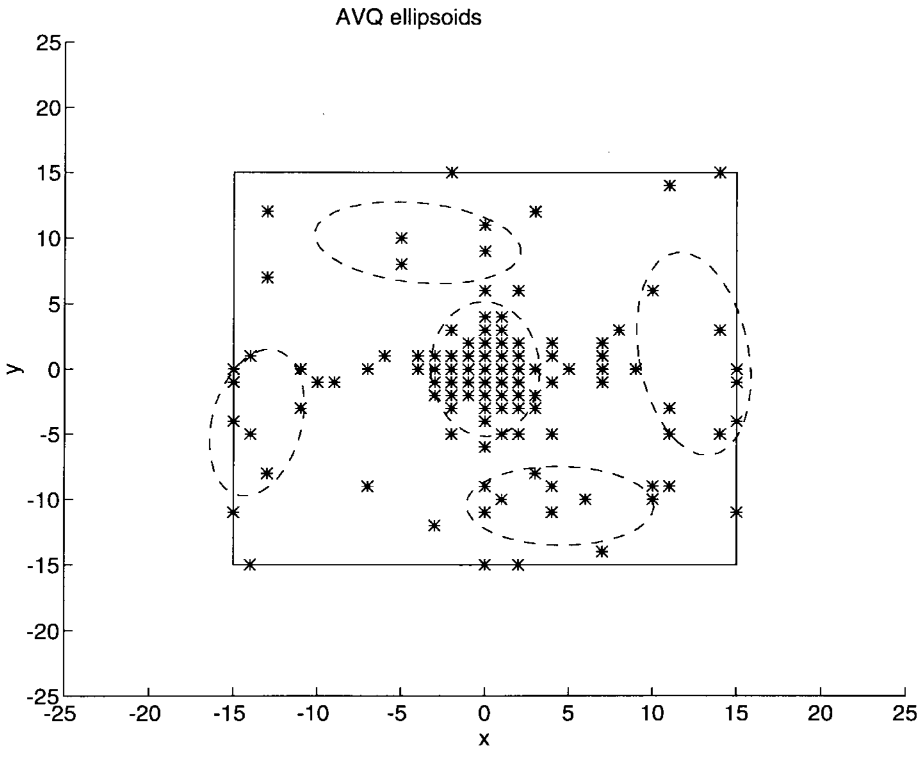


Fig. 9. Ellipsoids created from the local means and covariances of the motion vectors from "Miss America." These ellipsoids act as search windows for the next frame.

The term $\theta_{s'}(i)$ is the variance of the desired output $I_k(s'+Ab)$ that we assume has zero mean. Then the mean-squared-error of prediction has the form [22]

$$e_{s'}^2 = \theta_{s'} - 2h_{s'}^T\gamma_{s'} + h_{s'}^T\Phi_{s'}h_{s'}. \tag{18}$$

Solving (18) with orthogonality conditions gives [29]

$$h_{s'}^* = \Phi_{s'}^{-1}\gamma_{s'}. \tag{19}$$

This is just the Wiener weight vector.

OBMC assumes symmetric motion to form windows that are symmetric with respect to the location of pixels in a block quadrant. It also assumes that how the system predicts each quadrant of a block should be similar except for a mirroring of the weight vector.

B. Fuzzy Overlapped Block Motion Compensation

OBMC needs the normal equation of least squares to solve for the Wiener weight vectors but the estimation of the correlations in (15)–(17) has a high cost in computation. OBMC uses 215 frames of scene for training in [22]. OBMC may not work well if the correlations of the test sequences differ from those of the training sequences. We cannot assume the same correlations for the test sequences as those for the training sequences. We propose instead a fuzzy estimator. Samples give probabilities associated with pixel intensities between two consecutive frames that motion vectors link. The learned probability density functions serve as fuzzy rules in the fuzzy system [24]. The number of blocks in a frame picks the number of rules.

We seek rules that relate the current pixel value to the prior frame's pixel values. To find them we use the motion vectors of neighboring blocks rather than estimate the Wiener weight vector. OBMC limits the size of the set M_s^k by 4 to control the complexity. The optimal motion vector for b depends on motion vectors assigned to the eight nearest neighbors of b. We choose the four nearest motion vectors per pixel that depend on s', as in Fig. 7. Therefore the sample data will give rules of the form IF X_1 is A_j^1 and X_2 is A_j^2 and X_3 is A_j^3 and X_4 is A_j^4 THEN Y is B_j.

Then, the fuzzy system F defines the map F: $R^4 \rightarrow R$. The X_i's stand for the pixel values that the four motion vectors point to and Y stands for the predicted pixel value. We change these rules slightly to reduce their dimension. We compute the differences between X_1 and the other variables by taking X_1 as a reference. This gives the reduced fuzzy system F: $R^3 \rightarrow R$ with rules of the form IF $X_2 - X_1$ is A_j^1 and $X_3 - X_1$ is A_j^2 and $X_4 - X_1$ is A_j^3 THEN $Y - X_1$ is B_j. The fuzzy system F: $R^3 \rightarrow R$ uses 196 rules of this form.

We factor the joint set function a_j: $R^3 \rightarrow [0, 1]$ as the product $a_j(x) = a_j^1(x_1)a_j^2(x_2)a_j^3(x_3)$ for the vector input $x = (x_1, x_2, x_3)$. The combined output fuzzy set B becomes

$$\begin{aligned} b(x, y) &= \sum_{j=1}^{m} w_j a_j(x) b_j(y) \\ &= \sum_{j=1}^{196} w_j a_j^1(x_1)a_j^2(x_2)a_j^3(x_3)b_j(y). \end{aligned} \tag{20}$$

We use triangular scalar set functions

$$b(x, y) = \sum_{j=1}^{196} w_j k\left(\frac{x_1 - x_j^1}{\lambda}\right)k\left(\frac{x_2 - x_j^2}{\lambda}\right)$$

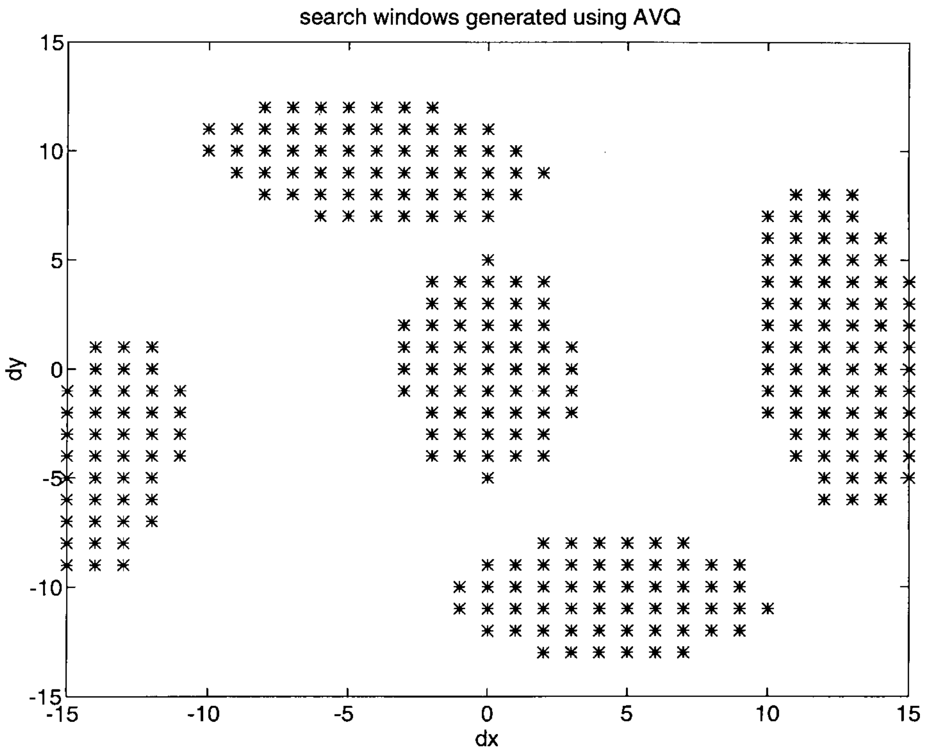


Fig. 10. Search windows obtained from the neural AVQ algorithm with unsupervised weighting for the “Miss America.” The system used these windows for the next frames until intra-frame coding reset them.

$$\cdot k\left(\frac{x_3 - x_j^3}{\lambda}\right) k\left(\frac{y - y_j}{\lambda}\right) \tag{21}$$

where $a[(x - x_j^i)/\lambda]$ stands for a triangular fuzzy set centered at x_j^i with base 2λ and height 1.

Parzen [26] showed how one may construct a family of estimates $\hat{f}_n$ of a probability density function f from n samples

$$\hat{f}_n(x) = \frac{1}{n\sigma} \sum_{i=1}^{n} k\left(\frac{x - x_i}{\sigma}\right). \tag{22}$$

Here, $x_1, \cdots, x_n$ are independent realizations of the random variable X. The estimator is consistent at all points x, where the density f is continuous. The weighting function k must obey Parzen's conditions

$$\sup_{-\infty < x < \infty} |k(x)| < \infty, \tag{23}$$

$$\int_{-\infty}^{\infty} |k(x)|\, dx < \infty, \tag{24}$$

$$\lim_{x \to \infty} |xk(x)| = 0, \tag{25}$$

and

$$\int_{-\infty}^{\infty} k(x)\, dx = 1. \tag{26}$$

Our set functions satisfy Parzen's conditions if we choose $\lambda = 1$. Parzen also proved that the estimate $\hat{f}$ is consistent in the mean-squared sense that

$$\lim_{n \to \infty} E\{|\hat{f}_n - f|^2\} = 0. \tag{27}$$

Cacoullos [1] has extended Parzen's results to cover the multivariate case

$$\hat{f}_n(x^1, x^2) = \frac{1}{n\lambda^2} \sum_{j=1}^{n} k\left(\frac{x^1 - x_j^1}{\lambda}\right) k\left(\frac{x^2 - x_j^2}{\lambda}\right). \tag{28}$$

This estimator uses n sampled data points (x_j^1, x_j^2) from random variables X_1 and X_2 and uses triangular weighting functions. Therefore, the combined output fuzzy “set” B with integrable set function b: $R \to [0, \infty)$ becomes the consistent estimator or joint density except for a scaling constant.

The conditional mean is the least-mean-square nonlinear estimate of the random variable Y in terms of the observed value x of the random variable X [25]

$$\phi(x) = E[Y|X = x] = \int_{-\infty}^{\infty} y f(y|x)\, dy. \tag{29}$$

Centroid defuzzification maps B to the conditional mean estimate $\hat{\phi}(x)$ [17]–[19]

$$\hat{\phi}(x) = \int_{-\infty}^{\infty} y \hat{f}(y|x)\, dy \tag{30}$$

$$= \int_{-\infty}^{\infty} y \frac{\hat{f}(x, y)}{\hat{f}(x)}\, dy \tag{31}$$

$$= \frac{\int_{-\infty}^{\infty} y \hat{f}(x, y)\, dy}{\int_{-\infty}^{\infty} \hat{f}(x, y)\, dy} \tag{32}$$

$$= \frac{\int_{-\infty}^{\infty} y b(x, y)\, dy}{\int_{-\infty}^{\infty} b(x, y)\, dy} \tag{33}$$

since

$$B(x) \longleftrightarrow b(x, y) \tag{34}$$

for all $y \in R$, even though $b(x, y) > 1$ may hold. Therefore the centroid defuzzifier plays the role of the conditional mean estimator. Putting (21) into (33) gives (35)–(39), shown at the bottom of the page, in the notation of the SAM in (2). Therefore, (39) can compute the conditional expected value in (13). Neural learning can tune the if-part sets A_j, the rule weights w_j, or the then-part set centroids c_j.

We do not need to store the weights as in the OBMC scheme since the rules come from the prior decoded frame. The neural fuzzy system also learns and tunes its rules as it decodes the image. This improves the prediction if the images are stationary over a small number of consecutive frames.

VI. Simulation Results

A. Motion Estimation

This section shows the simulation results we obtained for the neural AVQ motion estimation algorithm applied to the International Committee on Telegraph and Telephones (CCITT) test sequences "Miss America" and "mobile1." We applied this algorithm to forward prediction only. We can apply the same algorithm to backward and bidirectional predictions if we choose the proper reference frames. We set the frame rate at 10 Hz rather than the original 30 Hz. Therefore we used the first, fourth, seventh, $\ldots$ frames in the simulation. They correspond to I frames and P frames in Fig. 2.

Fig. 8 shows the first, fourth, and seventh frames for "Miss America." We computed the motion vectors from the first two frames of "Miss America" using the full-search block matching technique. Fig. 6 shows the motion vector histogram that the first two frames gave. We estimated the local means and covariances using AVQ based on these motion vectors. Fig. 9 shows the ellipsoids that arise from AVQ and the directions of motion vectors. The directions of motion vectors do not reflect the number of occurrences in that directions. The local means and covariances of the motion vectors choose these ellipsoids. We chose five ellipsoids for the simulation. The maximum displacement and the ellipsoids pick the common set of the boundary. This in turn picked the size and shape of the search windows for the next frames until intraframe coding reset them. Fig. 10 shows the windows the system grew from the next frames.

Simulation results compared the full-search block matching technique, AVQ window without unsupervised weighting, and AVQ window with unsupervised weighting. We used 16×16

$$\hat{\phi}(x) = \frac{\int_{-\infty}^{\infty} y \sum_{j=1}^{196} w_j a\left(\frac{x_1 - x_j^1}{\lambda}\right) a\left(\frac{x_2 - x_j^2}{\lambda}\right) a\left(\frac{x_3 - x_j^3}{\lambda}\right) a\left(\frac{y - y_j}{\lambda}\right) dy}{\int_{-\infty}^{\infty} \sum_{j=1}^{196} w_j a\left(\frac{x_1 - x_j^1}{\lambda}\right) a\left(\frac{x_2 - x_j^2}{\lambda}\right) a\left(\frac{x_3 - x_j^3}{\lambda}\right) a\left(\frac{y - y_j}{\lambda}\right) dy} \tag{35}$$

$$= \frac{\sum_{j=1}^{196} w_j a\left(\frac{x_1 - x_j^1}{\lambda}\right) a\left(\frac{x_2 - x_j^2}{\lambda}\right) a\left(\frac{x_3 - x_j^3}{\lambda}\right) y_j}{\sum_{j=1}^{196} w_j a\left(\frac{x_1 - x_j^1}{\lambda}\right) a\left(\frac{x_2 - x_j^2}{\lambda}\right) a\left(\frac{x_3 - x_j^3}{\lambda}\right)} \tag{36}$$

$$= \frac{\sum_{j=1}^{196} w_j a_j^1(x_1) a_j^2(x_2) a_j^3(x_3) y_j}{\sum_{j=1}^{m} w_j a_j^1(x_1) a_j^2(x_2) a_j^3(x_3)} \tag{37}$$

$$= \frac{\sum_{j=1}^{196} w_j a_j(x) y_j}{\sum_{j=1}^{m} w_j a_j(x)} \tag{38}$$

$$= \frac{\sum_{j=1}^{196} w_j a_j(x) c_j}{\sum_{j=1}^{196} w_j a_j(x)} \tag{39}$$

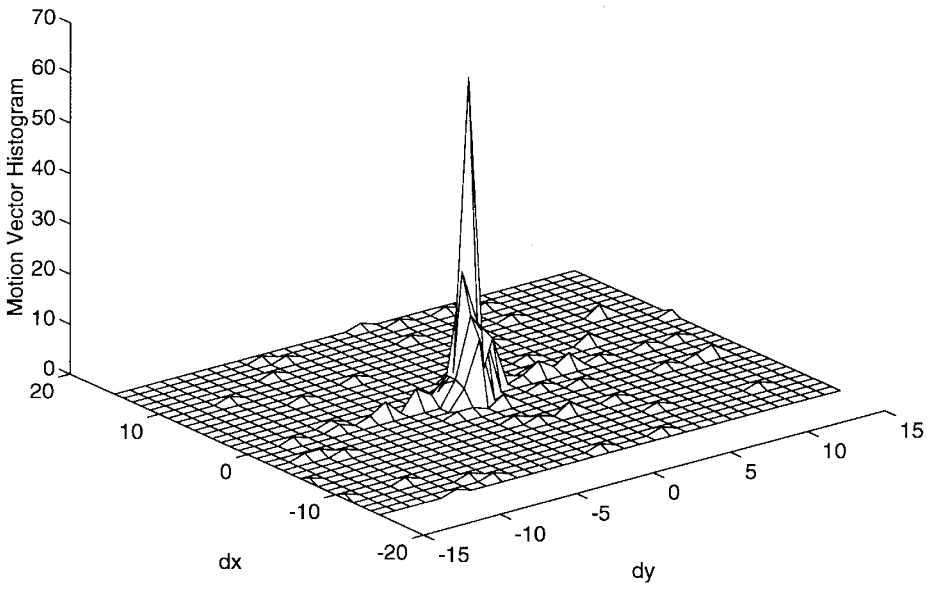


Fig. 11. Motion vector histogram obtained for the full-search algorithm for the seventh frame in the "Miss America" image. Note the randomness of the motion field.

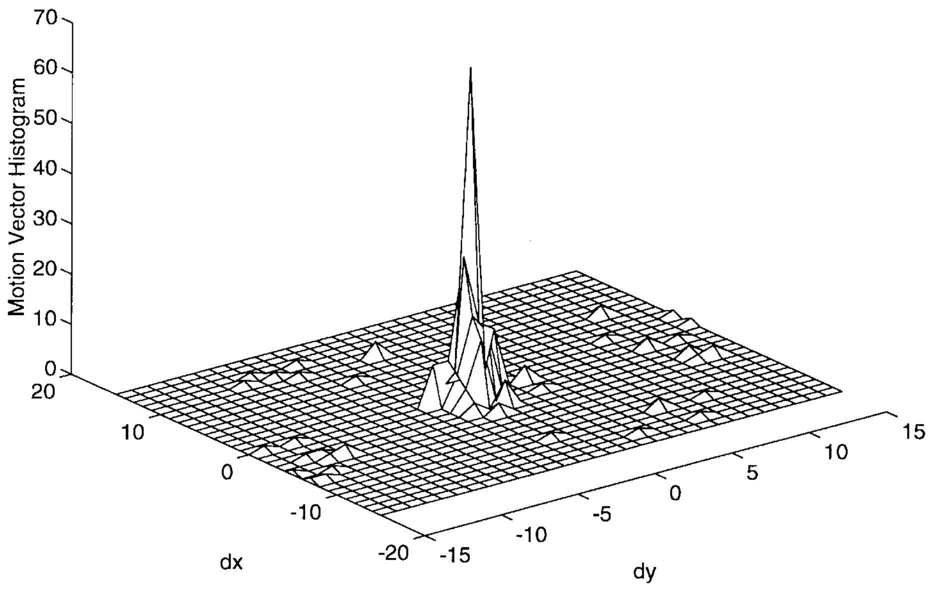


Fig. 12. Motion vector histogram obtained for the AVQ search algorithm for the seventh frame in the "Miss America" image.

pixels as a block size and ± 15 pixels for maximum displacement. The simulation ran for one period until intraframe coding reset it. Fig. 11 shows the motion vector histogram obtained from the full-search algorithm for the seventh frame. It shows the randomness of the motion field because it searches only for the least-error position. Fig. 12 shows the motion vector histogram obtained for AVQ search with unsupervised weighting for the seventh frame of "Miss America." This gave a smooth and clustered motion field with almost the same prediction error. We can also decrease the side information to transmit the entropy-coded motion vectors. We compared the average sum of absolute difference (SAD) and entropy for "Miss America." We compute the entropy $H(P)$ of the discrete probability density $P = (p_1, \cdots, p_n)$ as

$$H(P) = -\sum_{i=1}^{n} p_i \log_2 (p_i) \tag{40}$$

where p_i stands for the occurrence probability of the motion vectors. Table II shows the results. The SAD increased about 0.4 to 1.7% as a result of a 70 to 74% reduction of the search area and decreased entropy.

Next, we applied our algorithm to the "mobile1" image. Fig. 13 shows the first, fourth, and seventh frames for "mobile1." Fig. 14 shows the motion vector histogram obtained for the full-search algorithm for "mobile1."

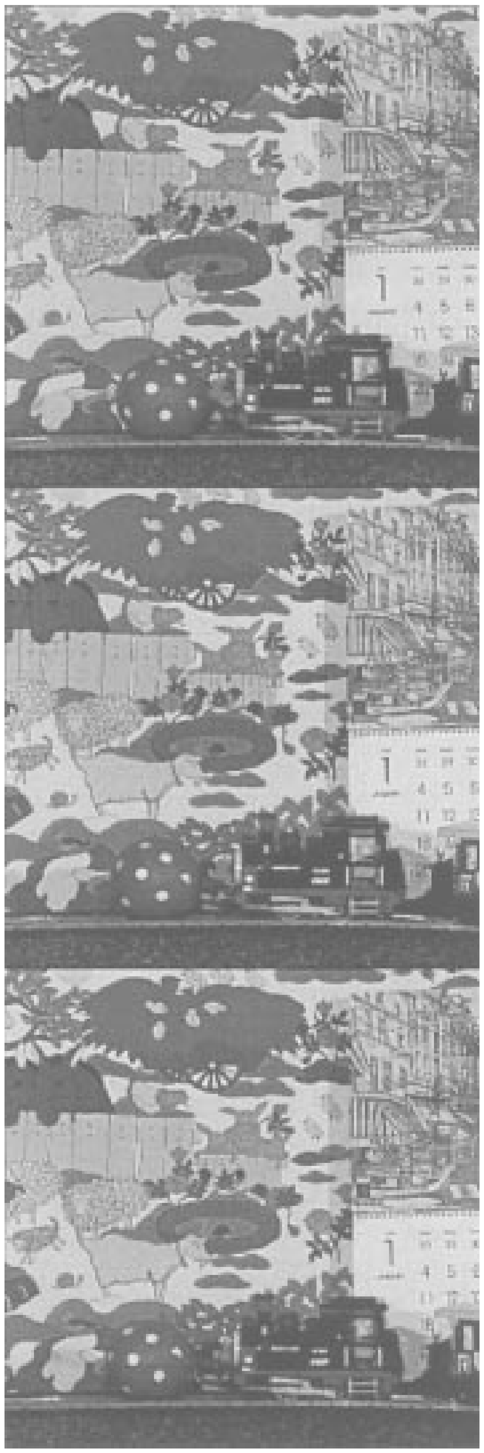


Fig. 13. "Mobile1" image. Top: First frame. Middle: Fourth frame. Bottom: Seventh frame.

Fig. 15 shows the search windows obtained for AVQ algorithm with unsupervised weighting for "mobile1." Table III shows the results. The SAD increased about 0.9 to 1.7% as a result of a 79 to 81% reduction of the search area and decreased entropy.

B. Fuzzy Motion Compensation and Supervised Learning

This section shows the simulation results we obtained when we applied the FOBMC to the CCITT test sequence "Miss America." All tests used a block size of 16×16 and reflect the results of only the motion-compensating original frames. The tests used integer-searched block motion fields. We excluded the border blocks from all test scores so that the measurements would not depend on how we treated the border regions. The test began with a ± 15 full-search integer pixel minimum-MSE block matching estimation.

Simulation results compared the standard motion compensation, FOBMC without SAM learning, and FOBMC with supervised SAM learning. We did not need to store the weights in the FOBMC system without learning since the rules come from the prior decoded frame. We used (64) in Appendix B to update the centroid c_j in the FOBMC. This further increased the compensation accuracy and the system complexity. We also tuned the weights w_j in (2) and then-part set centroids c_j using (57) and (64). Our fuzzy systems had the same then-part set volumes. Therefore $V_1 = \cdots = V_m > 0$. We did not tune the then-part set volumes. Tuning both w_j and c_j further increased the compensation accuracy but at a greater cost of computation. Table IV and Fig. 16 show the results. We also tuned the mean and "variance" of the triangles with (68) and (69) as explained in Appendix B. Tuning the if-part sets did not improve over tuning only the rule weights w_j and then-part centroids c_j but it did require much more computation.

VII. Conclusion

Neural fuzzy motion estimation and compensation offer new tools for video compression. They helped reduce the complexity for motion estimation using the temporal correlation of the motion field and decreased transmission entropy of motion vectors. Tuning the fuzzy compensator with supervised learning further increased the compensation accuracy without side information. The real advantage of the fuzzy system is that it computes a *model-free* nonlinear conditional mean.

We applied the neural-fuzzy motion estimation and compensation schemes to image sequences with fast motion. We had problems when we tried to estimate motion vectors with small search windows. Our scheme may best apply to systems like video conferencing where the motion is moderate and the bit budget for motion vectors is tight. The side information for motion vectors makes up a large portion of the total bit stream.

Very low-bit-rate coding requires improved motion estimation and compensation techniques since it creates annoying blocking artifacts as bit rates fall. We need to estimate more meaningful motion vectors than simple scalar optimization may provide. The mean-squared-error score may not estimate true motion vectors because it locates only the minimum of the prediction error.

Future fuzzy systems may estimate motion vectors using prior knowledge of the moving objects. Fuzzy systems allow the user to state knowledge in rules and insert these rules in a numerical function approximator. The same or new neural learning laws can tune these rules.

Future neural fuzzy systems for motion compensation may also use more complex learning laws to tune their sets and rules for better compensation accuracy. The supervised SAM

TABLE II
MEAN-ABSOLUTE ERROR (AND ENTROPY) FOR THE "MISS AMERICA" IMAGE

	Full-Search Window	AVQ Window without unsupervised weighting	AVQ Window with unsupervised weighting
Search Points	961	253	289
7th frame	754.6 (1.7939)	767.1 (1.6199)	758.3 (1.6925)
10th frame	699.3 (1.7583)	703.7 (1.6073)	702.3 (1.6622)
13th frame	670.4 (1.6936)	681.0 (1.5600)	673.5 (1.6194)

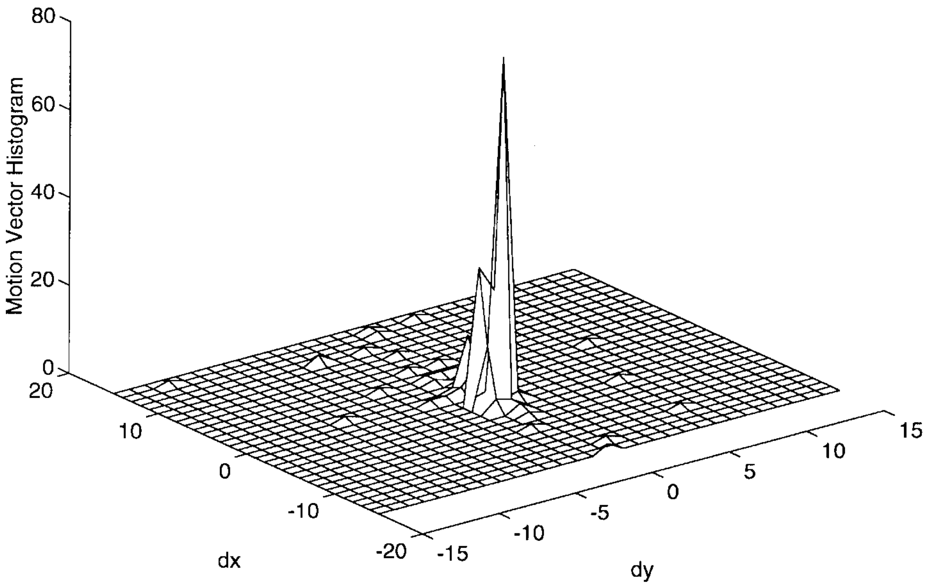


Fig. 14. Motion vector histogram obtained for the full-search algorithm for the "mobile1" image.

TABLE III
MEAN-ABSOLUTE ERROR (AND ENTROPY) FOR THE "MOBILE1" IMAGE

	Full Search Window	AVQ Window without unsupervised weighting	AVQ Window with unsupervised weighting
Search Points	961	179	206
7th frame	3788.9 (1.3453)	3853.0 (1.3013)	3848.0 (1.3013)
10th frame	3636.5 (1.2364)	3671.7 (1.2077)	3671.7 (1.2077)
13th frame	3767.1 (1.1397)	3811.0 (1.1211)	3802.2 (1.1211)

learning laws in the Appendix may not allow real-time learning for large imaging problems. That real-time barrier recedes as computer systems become faster and more parallel. Customized fuzzy chips can improve the speed and on-line learning of the fuzzy motion compensators but perhaps not enough for fuzzy systems that use many input variables. Even the best fuzzy systems face a prohibitive exponential rule explosion in high enough dimensions. Real-time neural learning only compounds this complexity. New learning schemes may ease this burden and may tune the black-box approximators without gradient descent.

APPENDIX A
THE FUZZY STANDARD ADDITIVE MODEL (SAM) THEOREM

Theorem: Suppose the fuzzy system F: $R^n \rightarrow R^p$ is a *standard additive model*: $F(x) = \text{Centroid}[B(x)] = \text{Centroid}[\sum_{j=1}^{m} a_j(x)B_j)]$. Then $F(x)$ is a convex sum of

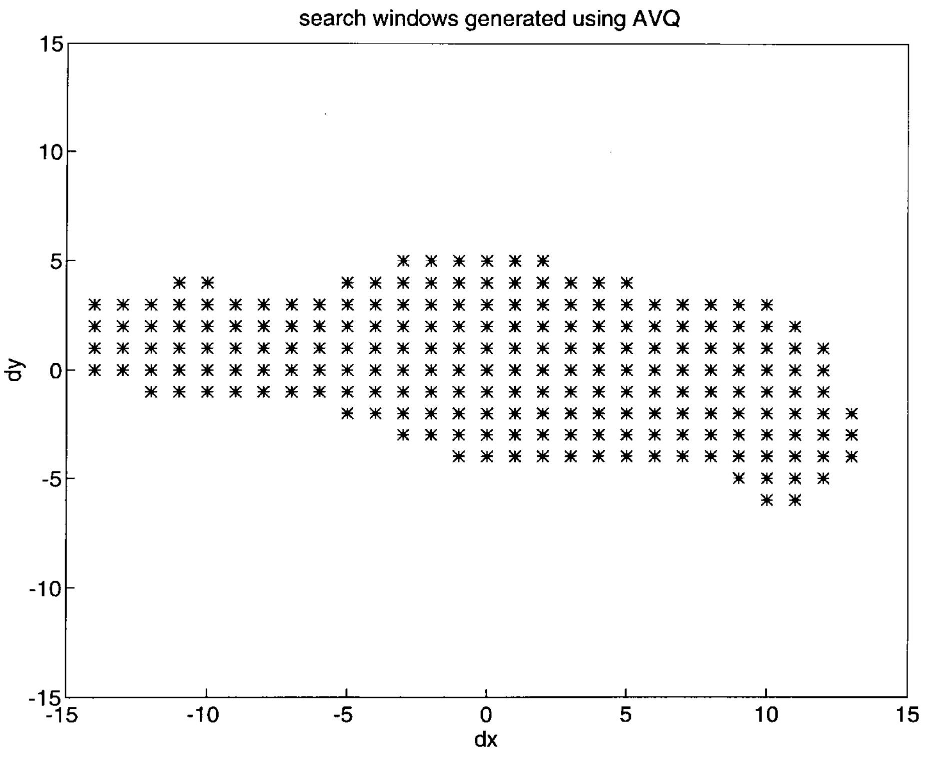


Fig. 15. Search windows obtained for the AVQ algorithm for the "mobile1" image. The system used these windows for the next frames until intra-frame coding reset them.

the m then-part set centroids

$$F(x) = \frac{\sum_{j=1}^{m} w_j a_j(x) V_j c_j}{\sum_{j=1}^{m} w_j a_j(x) V_j} = \sum_{j=1}^{m} p_j(x) c_j. \tag{41}$$

The convex coefficients or discrete probability weights $p_1(x), \cdots, p_m(x)$ depend on the input x through

$$p_j(x) = \frac{w_j a_j(x) V_j}{\sum_{k=1}^{m} w_k a_k(x) V_k} \tag{42}$$

where V_j is the finite positive volume (or area if $p = 1$) and c_j is the centroid of then-part set B_j

$$V_j = \int_{R^p} b_j(y_1, \cdots, y_p)\, dy_1 \cdots dy_p > 0 \tag{43}$$

$$c_j = \frac{\int_{R^p} y b_j(y_1, \cdots, y_p)\, dy_1 \cdots dy_p}{\int_{R^p} b_j(y_1, \cdots, y_p)\, dy_1 \cdots dy_p}. \tag{44}$$

Proof: There is no loss of generality to prove the theorem for the scalar-output case $p = 1$ when F: $R^n \to R^p$. This simplifies the notation. We need but replace the scalar integrals over R with the p-multiple or volume integrals over R^p in the proof to prove the general case. The scalar case $p = 1$ gives (43) and (44) as

$$V_j = \int_{-\infty}^{\infty} b_j(y)\, dy \tag{45}$$

$$c_j = \frac{\int_{-\infty}^{\infty} y b_j(y)\, dy}{\int_{-\infty}^{\infty} b_j(y)\, dy.} \tag{46}$$

Then, the theorem follows by expanding the centroid of B and invoking the SAM assumption $F(x) = \text{Centroid}[B(x)] = \text{Centroid}[\sum_{j=1}^{m} w_j a_j(x) B_j]$ to rearrange terms

$$F(x) = \text{Centroid}(B) \tag{47}$$

$$= \frac{\int_{-\infty}^{\infty} y b(y)\, dy}{\int_{-\infty}^{\infty} b(y)\, dy} \tag{48}$$

$$= \frac{\int_{-\infty}^{\infty} y \sum_{j=1}^{m} w_j b'_j(y)\, dy}{\int_{-\infty}^{\infty} \sum_{j=1}^{m} w_j b'_j(y)\, dy} \tag{49}$$

$$= \frac{\int_{-\infty}^{\infty} y \sum_{j=1}^{m} w_j a_j(x) b_j(y)\, dy}{\int_{-\infty}^{\infty} \sum_{j=1}^{m} w_j a_j(x) b_j(y)\, dy} \tag{50}$$

$$= \frac{\sum_{j=1}^{m} w_j a_j(x) \int_{-\infty}^{\infty} y b_j(y)\, dy}{\sum_{j=1}^{m} w_j a_j(x) \int_{-\infty}^{\infty} b_j(y)\, dy} \tag{51}$$

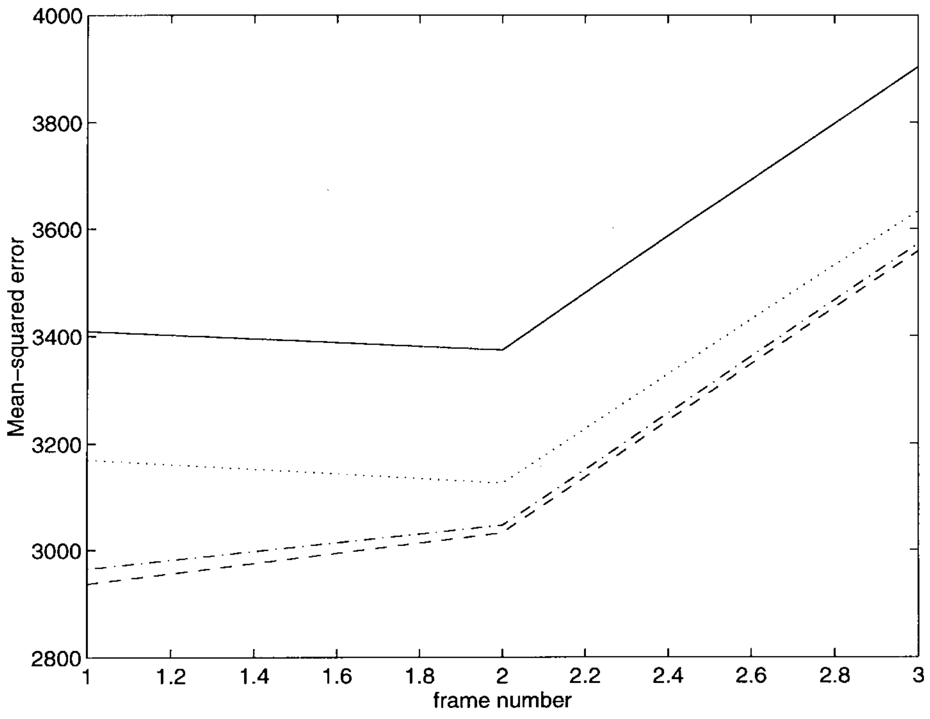


Fig. 16. Mean-squared error for neural-fuzzy motion compensation of the "Miss America" image. Solid line: Standard motion compensation. Dotted line: FOBMC without learning. Dashdot line: FOBMC with supervised SAM learning for then-part set centroids c_j. Dashed line: FOBMC with supervised learning for both rule weights w_j and then-part set centroids c_j. Further tuning the if-part set triangles A_j did not significantly improve the motion compensation.

$$= \frac{\sum_{j=1}^{m} w_j a_j(x) V_j \frac{\int_{-\infty}^{\infty} y b_j(y)\,dy}{V_j}}{\sum_{j=1}^{m} w_j a_j(x) V_j} \tag{52}$$

$$= \frac{\sum_{j=1}^{m} w_j a_j(x) V_j c_j}{\sum_{j=1}^{m} w_j a_j(x) V_j}. \tag{53}$$

Q.E.D.

Appendix B
Learning in SAMs: Supervised Gradient Descent

Supervised gradient descent can learn or tune SAM systems [19], [20] by changing the rule weights w_j in (54), the then-part volumes V_j, the then-part centroids c_j, or parameters of the if-part set functions a_j: $R^n \rightarrow [0, 1]$. The rule weight w_j enters the ratio form of the general SAM system

$$F(x) = \frac{\sum_{j=1}^{m} w_j a_j(x) V_j c_j}{\sum_{j=1}^{m} w_j a_j(x) V_j} \tag{54}$$

in the same way as does the then-part volume V_j in the SAM Theorem. Both cancel from (54) if they have the same value—if $w_1 = \cdots = w_m > 0$ or if $V_1 = \cdots = V_m > 0$. Therefore, both have the same learning law if we replace the nonzero weight w_j with the nonzero volume V_j or V_j'

$$w_j(t+1) = w_j(t) - \mu_t \frac{\partial E_t}{\partial w_j} \tag{55}$$

$$= w_j(t) - \mu_t \frac{\partial E_t}{\partial F} \frac{\partial F}{\partial w_j} \tag{56}$$

$$= w_j(t) + \mu_t \varepsilon_t \frac{p_j(x_t)}{w_j(t)} [c_j - F(x_t)] \tag{57}$$

for instantaneous squared error $E_t = \frac{1}{2}[d_t - F(x_t)]^2$ with desired-minus-actual error $\varepsilon_t = d_t - F(x_t)$. The volumes then change in the same way if they do not depend on the weights (which they may in some ellipsoidal learning schemes)

$$V_j(t+1) = V_j(t) - \mu_t \frac{\partial E_t}{\partial V_j} \tag{58}$$

$$= V_j(t) + \mu_t \varepsilon_t \frac{p_j(x_t)}{V_j(t)} [c_j - F(x_t)]. \tag{59}$$

The learning law (57) follows since $\partial E_t / \partial F = -\varepsilon$ and since from the SAM Theorem we have

$$\frac{\partial F}{\partial w_j} = \frac{a_j(x) V_j c_j \sum_{i=1}^{m} w_i a_i(x) V_i - a_j(x) V_j \sum_{i=1}^{m} w_i a_i(x) V_i c_i}{\left[\sum_{i=1}^{m} w_i a_i(x) V_i\right]^2} \tag{60}$$

TABLE IV
MEAN-SQUARED ERROR FOR "MISS AMERICA" MOTION COMPENSATION

	Standard MC	FOBMC without learning	FOBMC with supervised learning for centroid c_j	FOBMC with supervised learning for weightw_j and centroid c_j
7th frame	3409	3170	2965	2937
10th frame	3376	3126	3046	3032
13th frame	3904	3633	3572	3558

$$= \frac{w_j a_j(x) V_j}{w_j \sum_{i=1}^{m} w_i a_i(x) V_i} \left[\frac{c_j \sum_{i=1}^{m} w_i a_i(x) V_i}{\sum_{i=1}^{m} w_i a_i(x) V_i} - \frac{\sum_{i=1}^{m} w_i a_i(x) V_i c_i}{\sum_{i=1}^{m} w_i a_i(x) V_i} \right] \tag{61}$$

$$= \frac{p_j(x)}{w_j} [c_j - F(x)] \tag{62}$$

The centroid c_j in the SAM Theorem has the simplest learning law

$$c_j(t+1) = c_j(t) - \mu_t \frac{\partial E_t}{\partial F} \frac{\partial F}{\partial c_j} \tag{63}$$

$$= c_j(t) + \mu_t \varepsilon_t p_j(x_t). \tag{64}$$

Therefore, the terms w_j, V_j, and c_j do not change when $p_j \approx 0$, and thus, when the jth if-part set barely fires, $a_j(x_t) \approx 0$.

Tuning the if-part sets involves more computation since the update law contains an extra partial derivative. Suppose if-part set function a_j is a function of l parameters $a_j = a_j(m_j^1, \cdots, m_j^l)$. Then we can update each parameter with

$$m_j^k(t+1) = m_j^k(t) - \mu_t \frac{\partial E_t}{\partial F} \frac{\partial F}{\partial a_j} \frac{\partial a_j}{\partial m_j^k} \tag{65}$$

$$= m_j^k(t) + \mu_t \varepsilon_t \frac{p_j(x_t)}{a_j(x_t)} [c_j - F(x_t)] \frac{\partial a_j}{\partial m_j^k}. \tag{66}$$

Exponential if-part set functions can reduce the learning complexity. They have the form $a_j = e^{f_j(m_j^1, \cdots, m_j^l)}$ and obey $\partial a_j / \partial m_j^k = a_j [\partial f_j(m_j^1, \cdots, m_j^l) / \partial m_j^k]$. Then, the parameter update (66) simplifies to

$$m_j^k(t+1) = m_j^k(t) + \mu_t \varepsilon_t p_j(x_t) [c_j - F(x_t)] \frac{\partial f_j}{\partial m_j^k}. \tag{67}$$

This can arise for independent exponential or Gaussian sets $a_j(x) = \prod_{i=1}^{n} \exp\{f_j^i(x_i)\} = \exp\left\{\sum_{i=1}^{n} f_j^i(x_i)\right\} = \exp\{f_j(x)\}$. The exponential set function $a_j(x) = \exp\left\{\sum_{i=1}^{n} u_j^i(v_j^i - x_i)\right\}$ has partial derivatives $\partial f_j / \partial u_j^k = v_j^k - x_k(t)$ and $\partial f_j / \partial v_j^k = u_j^k$.

The Gaussian set function $a_j(x) = \exp\left\{-\frac{1}{2} \sum_{i=1}^{n} [(x_i - m_j^i)/\sigma_j^i]^2\right\}$ has mean partial derivative $\partial f_j / \partial m_j^k = (x_k - m_j^k)/(\sigma_j^k)^2$ and variance partial derivative $\partial f_j / \partial \sigma_j^k = (x_k - m_j^k)^2/(\sigma_j^k)^3$. Such Gaussian set functions reduce the SAM model to Specht's [28] radial basis function network. We used these smooth update laws (67) in the motion compensation simulation to update the nondifferentiable triangles. We viewed their centers and widths as the Gaussian means and standard deviations and tuned them with the laws

$$m_j^k(t+1) = m_j^k(t) + \mu_t \varepsilon_t p_j(x_t) \left[\frac{c_j - F(x_t)}{\sigma_j^k} \right] \left[\frac{x_t^k - m_j^k}{\sigma_j^k} \right] \tag{68}$$

$$\sigma_j^k(t+1) = \sigma_j^k(t) + \mu_t \varepsilon_t p_j(x_t) \left[\frac{c_j - F(x_t)}{\sigma_j^k} \right] \left[\frac{x_t^k - m_j^k}{\sigma_j^k} \right]^2. \tag{69}$$

REFERENCES

[1] T. Cacoullos, "Estimation of a multivariate density," *Ann. Inst. Statist. Math. (Tokyo)*, vol. 18, no. 2, pp. 179–189, 1966.

[2] J. A. Dickerson and B. Kosko, "Fuzzy function learning with covariance ellipsoids," in *Proc. 2nd IEEE Int. Conf. Fuzzy Syst.*, Mar. 1993, pp. 1162–1167.

[3] ——, "Fuzzy function approximation with supervised ellipsoidal learning," in *Proc. World Congr. Neural Networks*, July 1993, vol. II, pp. 9–13.

[4] ——, "Fuzzy function approximation with ellipsoidal rules," *IEEE Trans. Syst., Man, Cybern.*, vol 26, pp. 542–560, Aug. 1996.

[5] F. Dufaux, I. Moccagatta, B. Rouchouze, T. Ebrahimi, and M. Kunt, "Motion-compensated generic coding of video based on a multiresolution data structure," *Opt. Eng.*, vol. 32, no. 7, pp. 1559–1570, July 1993.

[6] D. L. Gall, "MPEG: A video compression standard for multimedia applications," *Commun. ACM*, vol. 34, pp. 46–58, Apr. 1991.

[7] B. Girod, D. Gall, M. I. Sezen, M. Vitterli, and H. Yasuda, "Guest editorial: Introduction to the special issue on image sequence compression," *IEEE Trans. Image Processing*, vol. 3, pp. 465–468, Sept. 1994.

[8] A. K. Jain and J. R. Jain, "Displace measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COMM-29, pp. 1799–1808, Dec. 1981.

[9] S. Kappagantula and K. R. Rao, "Motion compensated interframe image prediction," *IEEE Trans. Commun.*, vol. COMM-33, pp. 1011–1015, Sept. 1985.

[10] J. Katto, J. Ohki, S. Nogoki, and M. Ohta, "A wavelet codec with overlapped motion compensation for very low bit-rate environment," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 328–338, June 1994.

[11] H. M. Kim and B. Kosko, "Fuzzy prediction and filtering in impulsive noise," *Fuzzy Sets Syst.*, vol. 77, no. 1, pp. 15–33, Jan. 1996.

[12] S. Kim and C.-C. J. Kuo, "A stochastic approach for motion vector estimation in video coding," in *Proc. SPIE 1994 Ann. Mtg.*, San Diego, CA, July 1994.

[13] S. Kong and B. Kosko, "Fuzzy subimage classification in image sequence coding," in *Proc. ICASSP'92*, Mar. 1992, vol. III, pp. 517–520.

[14] B. Kosko, "Stochastic competitive learning," *IEEE Trans. Neural Networks*, vol. 2, pp. 522–529, Sept. 1991.

[15] ——, *Neural Networks and Fuzzy Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1991.
[16] ——, *Neural Networks for Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1991.
[17] ——, "Fuzzy systems as universal approximators," *IEEE Trans. Comput.*, vol. 43, pp. 1329–1333, Nov. 1994; an earlier version appears in *Proc. 1st IEEE Int. Conf. Fuzzy Syst.*, Mar. 1992, pp. 1153–1162.
[18] ——, "Optimal fuzzy rules cover extrema," *Int. J. Intell. Syst.*, vol. 10, no. 2, pp. 249–255, Feb. 1995.
[19] ——, "Combining fuzzy systems," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Mar. 1995, vol. IV, pp. 1855–1863.
[20] ——, *Fuzzy Engineering.* Englewood Cliffs, NJ: Prentice-Hall, 1996.
[21] H. Li, A. Lundmark, and R. Forchheimer, "Image sequence coding at very low bitrates: A review," *IEEE Trans. Image Processing*, vol. 3, pp. 589–609, Sept. 1994.
[22] M. T. Orchard and G. J. Sullivan, "Overlapped block motion compensation: An estimation–theoretic approach," *IEEE Trans. Image Processing*, vol. 3, pp. 693–699, Sept. 1994.
[23] M. T. Orchard, "New pel-recursive motion estimation algorithms based on novel interpolation kernels," *SPIE Visual Commun. Image Processing*, vol. 1818, pp. 85–96, 1992.
[24] P. J. Pacini and B. Kosko, "Adaptive fuzzy frequency hopper," *IEEE Trans. Commun.*, vol. 43, pp. 2111–2117, June 1995.
[25] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 2nd ed. New York: McGraw-Hill, 1984.
[26] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statistics*, vol. 33, pp. 1065–1076, 1962.
[27] B. Ramamurthi and A. Gersho "Classified vector quantization of images," *IEEE Trans. Commun.*, vol. COMM-34, pp. 1105–1115, Nov. 1986.
[28] D. F. Specht, "A general regression neural network," *IEEE Trans. Neural Networks*, vol. 2, pp. 568–576, Feb. 1991.
[29] B. Widrow and S. D. Sterns, *Adaptive Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1985.

Hyun Mun Kim received the B.S. degree in control and instrumentation engineering from the Seoul National University, Seoul, Korea, the M.S. degree in electrical and computer engineering from North Carolina State University, Raleigh, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 1984, 1985, and 1995, respectively.

During 1986 to 1991, he worked with ETRI, Seoul, as a researcher. Since 1996 he has been working with LG Semicon Co. Ltd., Seoul. His research interests include neural and fuzzy techniques in signal and image processing.

Bart Kosko received Bachelor degrees in philosophy and economics from the University of Southern California (USC), Los Angeles, the M.S. degree in applied mathematics from the University of California, San Diego, and the Ph.D. degree in electrical engineering from the University of California, Irvine.

He is an Associate Professor of electrical engineering at USC and a past Director of USC's Signal and Image Processing Institute.

Dr. Kosko is an elected governor of the International Neural Network Society and has chaired and cochaired many neural and fuzzy systems conferences. He has authored the textbooks *Fuzzy Engineering, Neural Networks for Signal Processing*, and *Neural Networks and Fuzzy Systems* (Englewood, Cliffs, NJ: Prentice-Hall) and the popular books *Fuzzy Thinking* and *Nanotime*. He was the guest co-editor of the summer 1988 PROCEEDINGS OF THE IEEE special issue on Intelligent Signal Processing.