Correspondence

New Techniques on Deformed Image Motion Estimation and Compensation

Chin-Teng Lin, Gin-Der Wu, and Shih-Chieh Hsiao

***Abstract*— In this paper, new techniques for deformed image motion estimation and compensation using variable-size block-matching are proposed, which can be applied to an image sequence compression system or a moving object recognition system. The motion estimation and compensation techniques have been successfully applied in the area of image sequence coding. Many research papers on improving the performance of these techniques have been published; many directions are proposed, which can all lead to better performance than the conventional techniques. Among them, both generalized block-matching and variable-size block-matching are successfully applied in reducing the data rate of compensation error and motion information, respectively. These two algorithms have their merits, but suffer from their drawbacks. Moreover, reducing the data rate in compensation error is sometimes increasing the data rate in motion information, or vice versa. Based on these two algorithms, we propose and examine several algorithms which are effective in reducing the data rate. We then incorporate these algorithms into a system, in which they work together to overcome the disadvantages of individual and keep their merits at the same time. The proposed system can optimally balance the amount of data rate in two aspects (i.e., compensation error and motion information). Experimental results show that the proposed system outweighs the conventional techniques. Since we propose a recovery operation which tries to recover the incorrect motion vectors from the global motion, this proposed system can also be applied for the moving object recognition in image sequence.**

***Index Terms*— Block-matching, data rate, genetic algorithm, image motion estimation, moving object recognition.**

I. Introduction

For the purpose of reducing temporal redundancy in image sequence, motion estimation and compensation techniques have been successfully applied [1]–[3]. Motion estimation and compensation techniques can be regarded as determining a coordinate transformation that is applied to every pixel of the frame. The coefficients of transformation are chosen by minimizing the distortion measure. Assume the $I_t(x,\ y)$ and $I_{t-1}(x,\ y)$ represent the pixel values of two consecutive frames, respectively, and $\mathbf{R}$ is a reference block in I_t. We can express the block-matching motion estimation as finding the motion vector $(d_x,\ d_y)$ which satisfies the following requirement:

$$\min \sum_{(x,\ y)\in\mathbf{R}} (I_t(x,\ y) - I_{t-1}(x + d_x,\ y + d_y))^2. \qquad (1)$$

In block-matching motion estimation, the assumption that an image is composed of rigid objects in pure translational motion is employed. In reality, however, motion is a complex combination of translation and rotation. In order to cope with rotation as well as other nonlinear deformations, a general approach to block-matching motion estimation was proposed in [4]. A geometric transformation instead of pure translation is employed and (1) is generalized to

$$\min \sum_{(x,\ y)\in\mathbf{R}} (I_t(x,\ y) - I_{t-1}(f(x,\ y),\ g(x,\ y)))^2 \qquad (2)$$

where $f(x,\ y)$ and $g(x,\ y)$ are the functions of transformation applied to coordinate values of x and y, respectively. Geometric transformations include: *affine*, *perspective*, and *polynomial* transformation. In all these transformations, translation is a special case.

Variable-size block-matching motion compensation is proposed in [5]–[7] to tackle the problem of nonuniform motion in search blocks of block-matching algorithm. In the algorithm, an image is adaptively divided into blocks of variable size to meet the assumption of uniform motion for all blocks. The algorithm is a split or merge segmentation scheme based on regular decomposition of an image into blocks of varying sizes, each of which has more or less uniform motion parameters. In comparison with traditional fixed size block-matching schemes, better results are obtained and fewer overhead bits are required.

The generalized block-matching algorithms can estimate the motion information more correctly than the conventional block-matching methods; thus, reduce the data rate in compensation error. However, more bits are required to represent the motion information, which means data rate in motion information is increased. Similarly, variable-size block-matching techniques effectively reduce the data rate in one aspect (motion information or compensation error), but increase the data rate in the other.

In this paper, we propose a system which can effectively reduce total data rate consumed in off-line image sequence compression. The proposed system incorporates merits of generalized block-matching and of variable-size block-matching, and eliminates drawbacks of both techniques. The system makes every effort to reduce and balance the amount of data rate in both aspects optimally, and thus reduces the total data rate. Since we propose a recovery operation which tries to recover the incorrect motion vectors from the global motion, this proposed system can also be applied for the moving object recognition in image sequence. The simplified architecture of the proposed system is shown in Fig. 1. The functions of the proposed system are described in brief below. In the first layer, which is shown in the bottom of Fig. 1, the predicted frame is partitioned into blocks of smallest size. In the next layer, blocks are classified into several clusters of variable size. Consequently, pixels of the same cluster are transformed to a deformed region by the output function of that cluster, f_{ij}. Eventually, a region in the reference frame is thus obtained that is the best fit of block in the predicted frame.

This paper is organized as follows. In Section II, learning of output function is introduced. In Section III, partitioning of the input space is discussed. The proposed system is then described in Section IV. Simulations are conducted in Sections II-E and IV-B, which show that the performance of the proposed system is superior to that of the conventional block-matching motion compensation.

II. Learning of Output Function

In this section, we are engaged in the learning of output function. First, we formulate the mathematical basis on determining the coefficients of a transformation. Consequently, we propose two learning algorithms, recursive least-square (RLS) and genetic algorithm (GA)

Manuscript received June 19, 1998; revised August 22, 1999. This work was supported by the National Science Council, R.O.C, under Grant NSC 88-2622-E-009-006. This paper was recommended by Associate Editor M. S. Obaidat.

The authors are with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan, R.O.C.

Publisher Item Identifier S 1083-4419(99)09307-3.

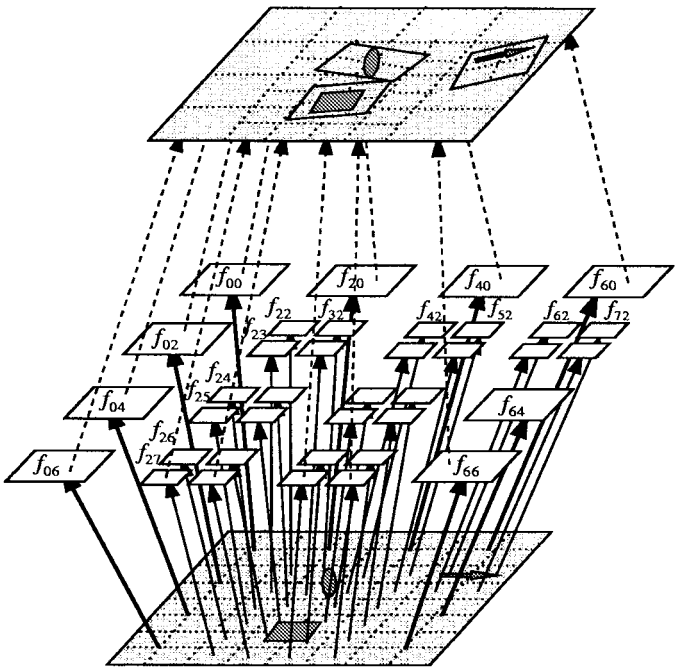


Fig. 1. Architecture of the proposed system.

learning algorithms. Finally, we conclude the section with simulations on the proposed learning algorithms.

A. Determination of Transformation Coefficients

In this section, we introduce how to solve for coefficients of a transformation, which is the basis of the proposed learning algorithms in the following sections.

An affine transformation that has six degrees of freedom, relating directly to coefficients $a_{11}, a_{21}, a_{31}, a_{12}, a_{22}, a_{32}$, is of the form

$$\begin{aligned} x &= a_{11}u + a_{21}v + a_{31} \\ y &= a_{12}u + a_{22}v + a_{32}. \end{aligned} \tag{3}$$

These six coefficients may be determined by specifying the coordinate correspondence of three noncollinear points in both images. Let (u_k, v_k) and (x_k, y_k) for $k = 0, 1, 2$ be three pairs of points in the reference and predicted frames, respectively. Affine mapping can be determined by solving the system of six linear equations as follows:

$$\begin{bmatrix} u_0 & v_0 & 1 & 0 & 0 & 0 \\ u_1 & v_1 & 1 & 0 & 0 & 0 \\ u_2 & v_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_0 & v_0 & 1 \\ 0 & 0 & 0 & u_1 & v_1 & 1 \\ 0 & 0 & 0 & u_2 & v_2 & 1 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{12} \\ a_{22} \\ a_{32} \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ y_0 \\ y_1 \\ y_2 \end{bmatrix}. \tag{4}$$

Let the system of equations given above be denoted as

$$\mathbf{WA} = \mathbf{X} \tag{5}$$

solving for the coefficients in terms of the known (u_k, v_k) and (x_k, y_k) pairs, we have

$$\mathbf{A} = \mathbf{W}^{-1}\mathbf{X}. \tag{6}$$

The constraint on $\mathbf{W}$ to consist of noncollinear points serves to ensure that $\mathbf{W}$ is nonsingular, i.e., $\det(\mathbf{W}) \neq 0$. Consequently, the inverse $\mathbf{W}^{-1}$ is guaranteed to exist.

A perspective transformation that has eight degrees of freedom is described by

$$\begin{aligned} x &= \frac{a_{11}u + a_{21}v + a_{31}}{a_{13}u + a_{23}v + 1} \\ y &= \frac{a_{12}u + a_{22}v + a_{32}}{a_{13}u + a_{23}v + 1}. \end{aligned} \tag{7}$$

The eight coefficients can be determined by establishing correspondence between four points in the reference and those in the predicted frame. Let (u_k, v_k) and (x_k, y_k) for $k = 0, 1, 2, 3$ be four pairs of points in the two frames, respectively. Moreover, (7) can be rewritten as

$$\begin{aligned} x &= a_{11}u + a_{21}v + a_{31} - a_{13}ux - a_{23}vx, \\ y &= a_{12}u + a_{22}v + a_{32} - a_{13}uy - a_{23}vy. \end{aligned} \tag{8}$$

Applying (8) to the four pairs of correspondence points yields the following 8×8 system of equations:

$$\begin{bmatrix} u_0 & v_0 & 1 & 0 & 0 & 0 & -u_0x_0 & -v_0x_0 \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -v_1x_1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -v_2x_2 \\ u_3 & v_3 & 1 & 0 & 0 & 0 & -u_3x_3 & -v_3x_3 \\ 0 & 0 & 0 & u_0 & v_0 & 1 & -u_0y_0 & -v_0y_0 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1y_1 & -v_1y_1 \\ 0 & 0 & 0 & u_2 & v_2 & 1 & -u_2y_2 & -v_2y_2 \\ 0 & 0 & 0 & u_3 & v_3 & 1 & -u_3y_3 & -v_3y_3 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{12} \\ a_{22} \\ a_{32} \\ a_{13} \\ a_{23} \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}. \tag{9}$$

We can thus determine coefficients by solving the linear system. This yields a solution to the general quadrilateral-to-quadrilateral problem.

In addition to affine and perspective transformation used in [4], we introduce polynomial transformation in this paper. The polynomial transformation is originally intended to account for sensor-related spatial distortions and external image distortions. The form of bivariate polynomial transformation is

$$\begin{aligned} x &= \sum_{i=0}^{N}\sum_{j=0}^{N-i} a_{ij}u^i v^j \\ y &= \sum_{i=0}^{N}\sum_{j=0}^{N-i} b_{ij}u^i v^j. \end{aligned} \tag{10}$$

Note that a first-degree ($N = 1$) bivariate polynomial defines the mapping function that is exactly given by an affine transformation.

In polynomial transformation, the number of coefficients is related to the degree of polynomial. A polynomial transformation of degree N has $2K$ coefficients, K for a_{ij} and K for b_{ij}, where

$$K = \sum_{i=0}^{N}\sum_{j=0}^{N-i} 1 = \frac{(N+1)(N+2)}{2}. \tag{11}$$

If we provide K pairs of correspondence points, we can solve the following system of equations and determine the coefficients of a polynomial of degree N:

$$\begin{bmatrix} 1 & u_0 & v_0 & u_0v_0 & u_0^2 & v_0^2 & \cdots & u_0^N & v_0^N \\ 1 & u_1 & v_1 & u_1v_1 & u_1^2 & v_1^2 & \cdots & u_1^N & v_1^N \\ 1 & u_2 & v_2 & u_2v_2 & u_2^2 & v_2^2 & \cdots & u_2^N & v_2^N \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & u_K & v_K & u_Kv_K & u_K^2 & v_K^2 & \cdots & u_K^N & v_K^N \end{bmatrix}$$

$$\cdot \begin{bmatrix} a_{00} \\ a_{10} \\ a_{01} \\ a_{11} \\ a_{20} \\ a_{02} \\ \vdots \\ a_{N0} \\ a_{0N} \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_K \end{bmatrix} \tag{12}$$

and

$$\begin{bmatrix} 1 & u_0 & v_0 & u_0v_0 & u_0^2 & v_0^2 & \cdots & u_0^N & v_0^N \\ 1 & u_1 & v_1 & u_1v_1 & u_1^2 & v_1^2 & \cdots & u_1^N & v_1^N \\ 1 & u_2 & v_2 & u_2v_2 & u_2^2 & v_2^2 & \cdots & u_2^N & v_2^N \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & u_K & v_K & u_Kv_K & u_K^2 & v_K^2 & \cdots & u_K^N & v_K^N \end{bmatrix}$$

$$\cdot \begin{bmatrix} b_{00} \\ b_{10} \\ b_{01} \\ b_{11} \\ b_{20} \\ b_{02} \\ \vdots \\ b_{N0} \\ b_{0N} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_K \end{bmatrix}. \tag{13}$$

B. Supervised Learning of Output Function

In the previous section, concept of geometric transformations and methods to solve for their coefficients are introduced. Solving a system of equations, we are required to provide sufficient number of noncollinear point pairs in both frames. In an image sequence, however, it is not easy to estimate the coordinate of the corresponding point in another frame, when a particular point in the current frame is given. In other words, it is almost impossible to provide accurate point pairs in both frames. Exhaustive search algorithm used in [4] can find the coefficients, but it takes a very long computation time even the search region is small, which is not practical.

In this section, we propose a recursive least-square (RLS) learning algorithm to find the coefficients of geometric transformation. Let the ith row vector of matrix $\mathbf{W}$ defined in (5) be w_i and the ith element of $\mathbf{X}$ defined in (5) be x_i. Then $\mathbf{A}$ can be calculated using the following recursive least-square formula:

$$\mathbf{a}_{i+1} = \mathbf{a}_i + \mathbf{S}_{i+1}\mathbf{w}_{i+1}^T(x_{i+1} - \mathbf{w}_{i+1}\mathbf{a}_i) \tag{14}$$

$$\mathbf{S}_{i+1} = \mathbf{S}_i - \frac{\mathbf{S}_i\mathbf{w}_{i+1}^T\mathbf{w}_{i+1}\mathbf{S}_i}{1 + \mathbf{w}_{i+1}\mathbf{S}_i\mathbf{w}_{i+1}^T}, \qquad i = 0, 1, \cdots, p-1 \tag{15}$$

$$\mathbf{A} = \mathbf{a}_p \tag{16}$$

with initial conditions of

$$\mathbf{a}_0 = 0 \quad \text{and} \quad \mathbf{S}_0 = \gamma\mathbf{I} \tag{17}$$

where γ is a positive big number and $\mathbf{I}$ is the identity matrix of dimension $K \times K$, and K is the number of columns in $\mathbf{W}$.

Since the derivative of (7) shows that RLS method has poor convergence accuracy in nonlinear transformation, we cannot solve the system in (9) using the RLS method directly. In this paper, an alternative approach is proposed as follows according to the above observation. First, the correspondence pair of points are shifted to zero-mean by the same amount of displacement (s_x, s_y). Second, those shifted data are arranged as row vectors. Consequently, the row vectors are applied to the RLS method sequentially, and learning result, $\mathbf{A}'$, is obtained. Finally, the exact solution, $\mathbf{A}$, can be recovered from $\mathbf{A}'$ using the following equation:

$$\begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{12} \\ a_{22} \\ a_{32} \\ a_{13} \\ a_{23} \end{bmatrix} = \begin{bmatrix} (a'_{11} + a'_{13}s_x)/\text{div} \\ (a'_{21} + a'_{23}s_x)/\text{div} \\ (a'_{31} - a'_{11}s_x - a'_{21}s_y + s_x \cdot \text{div})/\text{div} \\ (a'_{12} + a'_{13}s_y)/\text{div} \\ (a'_{22} + a'_{23}s_y)/\text{div} \\ (a'_{32} - a'_{12}s_x - a'_{22}s_y + s_y \cdot \text{div})/\text{div} \\ a'_{13}/\text{div} \\ a'_{23}/\text{div} \end{bmatrix} \tag{18}$$

where $\text{div} = 1 - a'_{13} - a'_{23}$, and a_{ij} and a'_{ij} are elements of $\mathbf{A}$ and $\mathbf{A}'$, respectively. In simulations, coefficients of perspective transformation are determined correctly with the proposed approach.

In the RLS learning of polynomial transformation, the transformation function in (10) is also nonlinear. Similarly, we can shift the correspondence pair of points to zero-mean by amount of displacement (s_x, s_y), apply RLS to obtain a'_{ij} and b'_{ij}, and recover coefficients, a_{ij} and b_{ij}, of polynomial transformation from a'_{ij} and b'_{ij} using (19) and (20), shown at the bottom of the page.

The supervised learning algorithm is summarized as follows.

- *Step 1:* Select K points, randomly or not. Note that the number of noncollinear points in the selected points must be equal to or greater than that of points required to properly determine the selected transformation.
- *Step 2:* Apply full-search block-matching algorithm to the selected points to obtain their motion vectors.
- *Step 3:* Coordinate of each point is used as input, and the sum of each point's coordinate and its associated motion vector serves as desired output. They form a pair of training data.
- *Step 4:* Apply training data sequentially to the RLS algorithm to learn the coefficients of transformation.

$$a_{d_xd_y} = \begin{cases} \displaystyle\sum_{\substack{i \ge d_x, j \ge d_j \\ i+j \le N}} a'_{ij} \binom{i}{d_x} \binom{j}{d_y} (-s_x)^{i-d_x}(-s_y)^{j-d_y} & \text{if } d_x,\ d_y \ne 0 \\ \displaystyle\sum_{\substack{i \ge 0, j \ge 0 \\ i+j \le N}} a'_{ij}(-s_x)^i(-s_y)^j + s_x & \text{if } d_x = d_y = 0 \end{cases} \tag{19}$$

and

$$b_{d_xd_y} = \begin{cases} \displaystyle\sum_{\substack{i \ge d_x, j \ge d_j \\ i+j \le N}} b'_{ij} \binom{i}{d_x} \binom{j}{d_y} (-s_x)^{i-d_x}(-s_y)^{j-d_y} & \text{if } d_x,\ d_y \ne 0 \\ \displaystyle\sum_{\substack{i \ge 0, j \ge 0 \\ i+j \le N}} b'_{ij}(-s_x)^i(-s_y)^j + s_y & \text{if } d_x = d_y = 0 \end{cases} \tag{20}$$

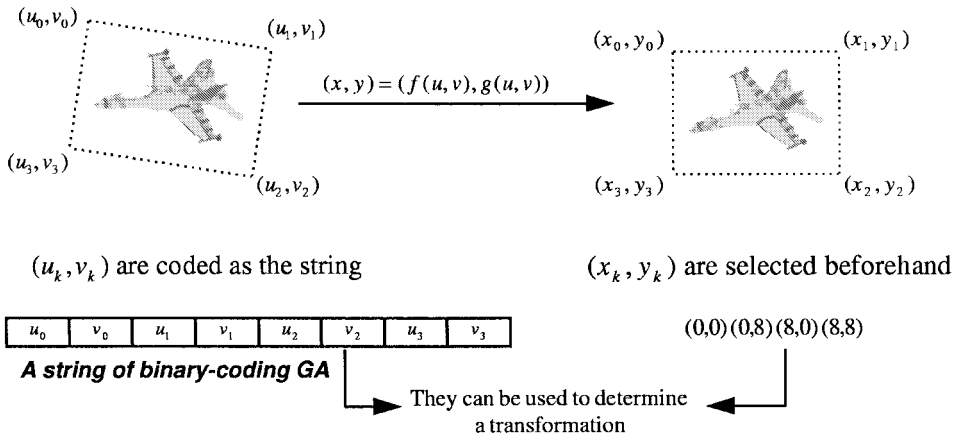


Fig. 2. Coding of the string in the binary-coding GA.

C. Genetic Algorithm Learning of Output Function

Genetic algorithm (GA) is a general purpose optimization algorithm with a probabilistic component that provide a means to search poorly understood, irregular spaces. Holland [8] developed the GA to simulate some of the processes observed in natural evolution. In this section, we propose two GA learning schemes of output functions.

1) Binary Coding Genetic Algorithm: In Section II-B, it is pointed out that if we apply a number of point pairs (three in affine transformation, four in perspective transformation, and K of (11) in polynomial transformation), the coefficients can be determined by solving a system of equations [(4), (9), (12), and (13) for the three transformations, respectively]. In other words, a transformation is determined by a group of point pairs (u_k, v_k) and (x_k, y_k).

Applying binary-coding GA to learn the transformation coefficients, we need to generate the strings by the coordinates of points, which is described as follows. We fix points (x_k, y_k) in the compensated frame, and vary points (u_k, v_k) in the reference frame. The variable points (u_k, v_k) are coded as strings in GA. A sufficient number of (u_k, v_k) and (x_k, y_k) can represent a transformation; hence, a transformation is thus coded. For example, in the perspective transformation case, four pairs of points are required to solve for coefficients. Therefore, four fixed points in the compensated frame, e.g., $(0, 0)$, $(8, 0)$, $(8, 8)$, and $(0, 8)$, are selected beforehand, and four correspondence points, (u_0, v_0), (u_1, v_1), (u_2, v_2), and (u_3, v_3), are coded as a string of eight bytes. To decode the string, a system of equations is constructed by the four pairs of points and the coefficients are obtained by solving the system of equations. The obtained coefficients (i.e., the obtained transformations, f and g) are evaluated by applying them to (2). The smaller the value of (2) is, the better the obtained coefficients (transformations) are. The coding of the string in binary-coding GA is illustrated in Fig. 2.

The following steps are employed to generate and handle a set of strings (i.e., a population) in the GA learning [9].

- *Step 1*: *Initialization:* We first generate an initial population containing N_{pop} strings, where N_{pop} is the number of strings in each population.
- *Step 2*: *Fitness Function:* In this step, each string is decoded by an evaluator into an objective function value called fitness, FIT. A fitness value is assigned to each individual in the population. According to (2), the fitness function is defined by

$$FIT = \sum_{(x,y)\in R} |I_t(x, y) - \tilde{I}_t(x, y)|, = \sum_{(x,y)\in R} |I_t(x, y) - I_{t-1}(f(x, y), g(x, y))| \quad (21)$$

  where f and g are the functions of transformation which are determined by the coded string. The purpose of the GA learning in our case is to search for the transformation which minimizes the above fitness function.
- *Step 3*: *Reproduction:* Reproduction is a process in which individual strings are copied according to their fitness values, i.e., based on the principle of survival of the fittest.
- *Step 4*: *Crossover:* We use the N-point crossover operator, in which a string is N bytes in length. At first, two strings from the reproduced population are mated at random, and N crossover points (N bits position, where one crossover point for each byte) are selected according to the type of transformation we are learning. Then the strings are crossed and separated at these points to produce two new strings.
- *Step 5*: *Mutation:* Mutation is the random alteration of bits in the string which assists in keeping diversity in the population. As a source of new bits, mutation is introduced and is applied with a probability p_m. In addition, a variation operation in GA is adopted, which enforces preserving the best string.
- *Step 6*: If the stopping condition is not met (e.g., the error is above a predefined level), return to Step 2. Otherwise, the GA is terminated.

Our experiments show that the learning curve of the binary-coding GA on a block of uniform motion falls in the first several generations, and remains unchanged in the following generations. In our experience, the binary-coding GA typically converges in several generations except in blocks of nonuniform motion due to unmask of background or variation of illumination, which takes more generations to reach convergence.

2) Floating-Point Genetic Algorithm: In the previous section, we code the coordinates of points (pixels) as a string, instead of coding coefficients themselves. Because of the restriction on binary coding, we can use only integer grid points. It is possible to extend the

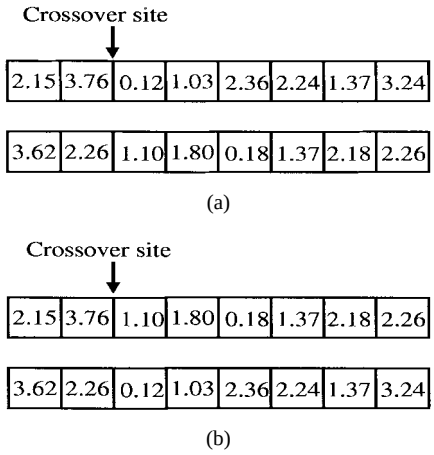


Fig. 3. Illustration of crossover operation for floating-point strings: (a) before crossover and (b) after crossover.

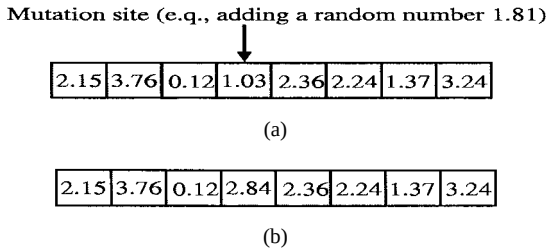


Fig. 4. Illustration of mutation operation for a floating-point string: (a) before mutation and (b) after mutation.

point to sub-subpixel accuracy. However, strings will be very long if a infinitesimal-subpixel is desired. Moreover, the computation time of decoding a string into coefficients, which occurs in the stage of evaluating fitness value, occupies a significant portions of execution time in the GA learning.

In order to increase the accuracy of the estimated coefficients and lessen the computation load, floating-point GA [10] is adopted. In the floating-point GA, we can code the coefficients themselves as strings of real values. Thus, the computation load of decoding is eliminated. Initially, the GA randomly generates a population of floating-point strings. An interpreter takes this floating-point string and uses it to set the coefficients of transformation. In this way, according to a defined fitness function, a fitness value is assigned to each string in the population. GA can look for a better set of strings to form a new population as the next generation. The crossover operation for encoding is demonstrated in Fig. 3. The steps to generate and handle a set of strings in the floating-point GA learning is similar to those in the binary-coding GA except for the mutation process, in which a random number $\Delta(t, i)$ is added to the string. The range of $\Delta(t, i)$ is variable according to the tth generation and the ith element in a string. Fig. 4 shows an example illustrating the mutation operation. Typically, the learning curve of the floating-point GA on a block of uniform motion falls quickly in the first several generations, and in the following generations the curve is slightly downward or stays on the same value.

D. Infinitesimal-Subpixel Compensation

In this section, we propose a quarter compensation algorithm (QCA). Note that the conventional motion compensation method operates only on the integer grids. We cannot apply the motion parameters estimated by generalized block-matching to the conventional motion compensation technique directly, because the obtained coordinate value of a point is not restricted to an integer. QCA is thus proposed to tackle this problem.

Suppose that the origin point of an image is on the left-top corner, and the positive directions are rightward and downward; moreover, the motion vector $d = (d_x, d_y)$ on the location (x, y) of the predicted frame is evaluated, in which both d_x and d_y are floating values. Without loss of generality, it is supposed that both d_x and d_y are positive, i.e., the point on the location (x, y) are compensated from a pixel of right-bottom direction in the reference frame. One distinguishes the integer part and decimal part of the motion vector as follows:

$$\begin{aligned} d_x &= d_{xi} + d_{xf} \\ d_y &= d_{yi} + d_{yf} \end{aligned} \tag{22}$$

where (d_{xi}, d_{yi}) and (d_{xf}, d_{yf}) are the integer and decimal parts of (d_x, d_y), respectively. The square, which is in location (x, y), is now reconstructed from the four locations, $(x + d_{xi}, y + d_{yi})$, $(x + d_{xi}+1, y+d_{yi})$, $(x+d_{xi}, y+d_{yi}+1)$, and $(x+d_{xi}+1, y+d_{yi}+1)$ in the reference frame. At these four locations, a reference square is thus divided into four parts, with the area of each part decided by the decimal part, (d_{xf}, d_{yf}), of the motion vector (d_x, d_y). The area of each part is viewed as a weighting factor in deciding the new gray value. More precisely, we assign the gray value of the pixel, which is in location (x, y), from the weighted gray values of the four locations (pixels).

Hence, for any location (x, y) in the compensated frame, with associated motion vector (d_x, d_y), the compensated gray value $I_c(x, y)$ is decided by the following equation:

$$I_c(x, y) = \frac{\sum_{(m,n)\in \mathbf{D}} w_{(m,n)-(x', y')} \cdot I_r(m, n)}{\sum_{(m,n)\in \mathbf{D}} w_{(m,n)-(x', y')}} \tag{23}$$

where

$(x', y') \triangleq (x + d_x, y + d_y)$: source pixel of compensated point,

$\mathbf{D} \triangleq \{(m, n)|$ one of the four points that are nearest to the point $(x', y')\}$,

$w_{(m,n)-(x', y')}$: the area (weighting factor) of the block of (m, n) that falls on on the unit-square of (x', y').

For any location (x, y) in the compensated image frame, its gray value can be obtained by (23). In (23), the numerator denotes the sum of product of the weighting factor and the gray value of points that belong to $\mathbf{D}$, and the denominator is the sum of all weighting factors, which is used as the normalization factor in the reconstruction process.

E. Simulation Results

In this section, we compare the performance of a total of ten algorithms: conventional block-matching (BM), and affine, perspective, polynomial transformations with three different learning algorithms (RLS, binary-coding GA, floating-point GA), respectively. We synthesize nine test images of different transformations to evaluate the learning algorithms. Among the nine test images, image 1 is the resulted image of rotation; image 2 is that of scaling; images 3 and 4 are resulted images of shearing in different directions. These images mentioned above are affine-transformed images. Images 5

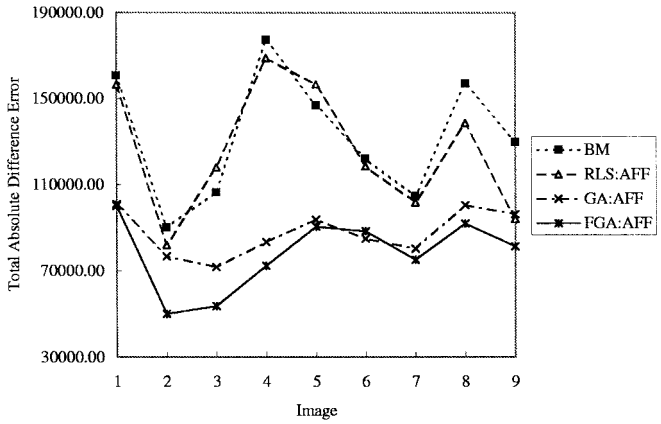


Fig. 5. Compensation error comparison between BM and RLS, GA, FGA affine transformations.

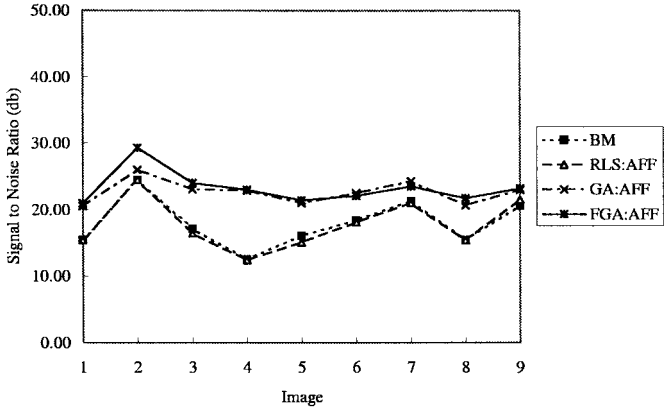


Fig. 6. SNR comparison between BM and RLS, GA, FGA affine transformations.

and 6 are perspective-transformed images. Images 7 and 8 are more complex images of affine and perspective transformations, respectively. Image 9 is created to test the subpixel prediction ability of learning algorithms. The quality evaluation between the original image and reconstructed one is performed under some objective quality criteria. These criteria include total absolute difference error, SNR, and data entropy, which are defined as follows.

- total absolute difference error

$$\text{Error} = \sum |I_t(x,\, y) - \tilde{I}(x + d_x,\, y + d_y)|; \tag{24}$$

- signal to noise ratio (SNR)

$$\text{SNR}_{\text{db}} = 10 \log_{10} \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} I^2(i,j)}{\sum_{i=1}^{M}\sum_{j=1}^{N} (I(i,j) - \tilde{I}(i,j))^2}; \tag{25}$$

- estimated entropy of compensation error

$$H = -\sum_{i=0}^{255} p_i \log_2 p_i \tag{26}$$

where p_i is the probability of pixel value i in the difference image.

Note that I and $\tilde{I}$ are the desired image frame and the reconstructed image frame, respectively. The absolute difference error reflects the ability of learning algorithms in motion estimation and compensation. SNR indicates the visual quality of the reconstructed images. The estimated entropy of compensation error is included to show the approximate data rate.

In Fig. 5, we compare the performance in terms of compensation error between BM and affine transformation of generalized block-matching with RLS, GA (binary-coding GA), FGA (floating-point GA) learning algorithms. The RLS predicts the frame more accurate than the BM except for two frames. In addition, the performances of both binary-coding and floating-point GA's outweigh the RLS and BM, and the floating-point GA is superior to the binary-coding one.

Fig. 6 shows the quality of the reconstructed images on the basis of signal-to-noise ratio. As expected, the performance of GA's is superior to the others. The estimated entropy of compensation error is shown in Fig. 7. The GA's outweigh the other algorithms again. Note that the performance of RLS in terms of SNR and entropy is poor when compared with BM, because the RLS is unable to predict the test images correctly and produces more frame difference errors on

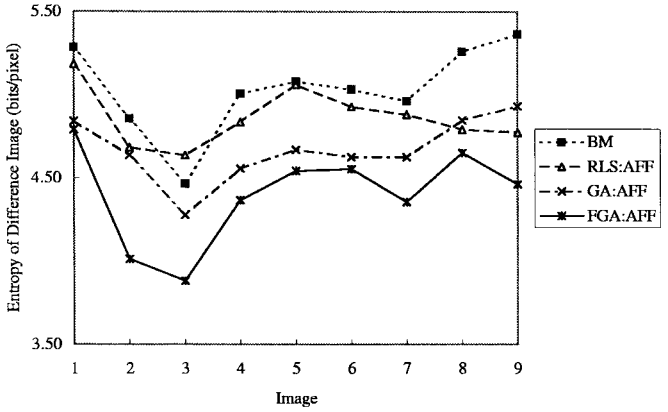


Fig. 7. Entropy comparison between BM and RLS, GA, FGA affine transformations.

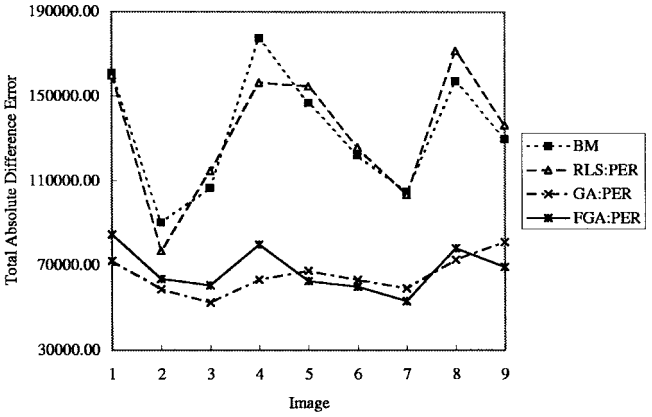


Fig. 8. Compensation error comparison between BM and RLS, GA, FGA perspective transformations.

border blocks where motion vectors are prone to error. Compensation errors on border blocks greatly deteriorate the performance of RLS.

Fig. 8 shows the test result in terms of compensation error between BM and perspective transformation, instead of affine transformation, of generalized block-matching with RLS, GA, and FGA learning. The compensation errors of GA's are again lower than those of the other two algorithms. However, compensation errors on the binary-coding GA and floating-point GA are approximate in this case. The results of comparisons on SNR and entropy are shown in Figs. 9 and 10, respectively.

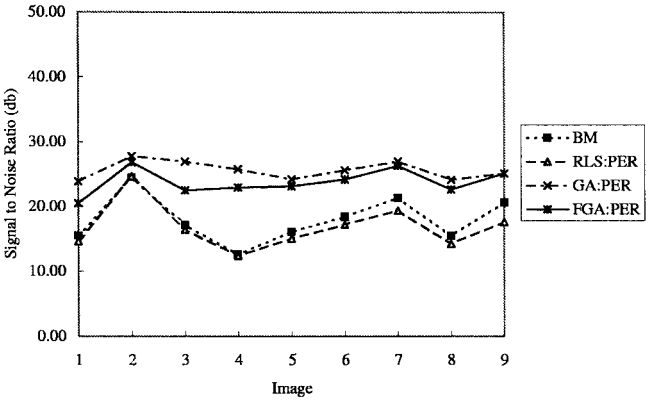


Fig. 9. SNR comparison between BM and RLS, GA, FGA perspective transformations.

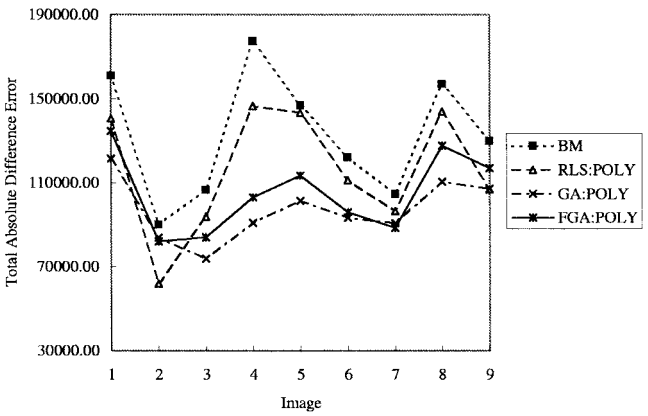


Fig. 11. Compensation error comparison between BM and RLS, GA, FGA polynomial transformations.

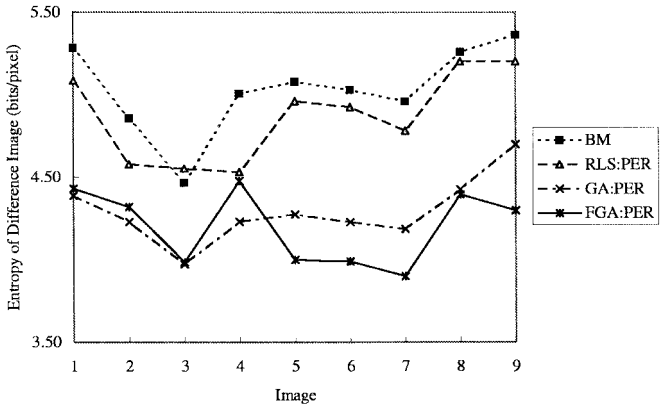


Fig. 10. Entropy comparison between BM and RLS, GA, FGA perspective transformations.

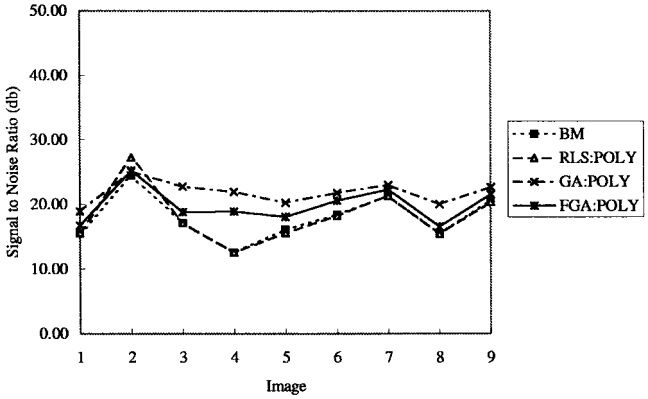


Fig. 12. SNR comparison between BM and RLS, GA, FGA polynomial transformations.

We also test the algorithms with polynomial transformation, and the comparison of total absolute difference error is shown in Fig. 11. The performance of BM is the worst when compared with the other three algorithms. The binary-coding GA performs better than the floating-point one, but the RLS algorithm sometimes results in the lowest error. The outcome indicates that the implementation of GA's is not optimized for learning coefficients of polynomial transformation. The comparison of algorithms on SNR is shown in Fig. 12. The performance of GA's is better than that of RLS and BM. Learning algorithms are also compared in terms of coding entropy. As depicted in Fig. 13, the BM needs more bits than others. Note that the performance of RLS is superior to that of BM in terms of compensation error and estimated entropy. However, the performance of RLS is similar to that of BM in terms of SNR, because the RLS fails to predict the test images correctly on the border blocks.

In summary, the performance of GA's in terms of frame difference, SNR, and entropy is superior to the other algorithms. This is due to the global optimization capability of GA's. With the crossover and mutation operations, the GA, unlike the block-matching and RLS algorithms, is able to get out of a local minimum trap and find better local minima or even the global minimum. However, it is observed that the standard GA learning hardly reaches the global minimum, which means the standard GA lacks the ability on fine local tuning. Three possible reasons account for this situation. First, the GA is trapped into local minima when almost all the individuals in a generation look alike. Second, assume the global minimum is m_d, and the current located minimum is m_i. It is possible that the

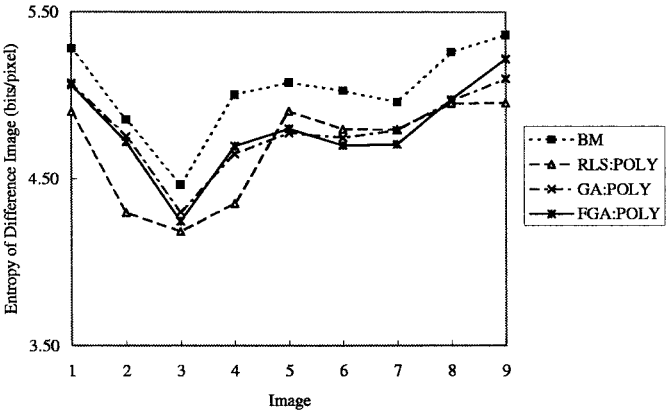


Fig. 13. Entropy comparison between BM and RLS, GA, FGA polynomial transformations.

continuous evolution of GA's cannot produce an individual (string) which is closer to m_d than m_i. Although there may be a chance to reach m_d, but it is not guaranteed to occur in a few generations. Third, in using floating-point GA, the global minimum m_d is harder to reach when m_i is closer to m_d due to the possible big change of floating-point strings applied by GA operations. If we fix the change of the strings to a small scale, then the number of generations to reach m_d will increase. The performance of standard GA can be improved with tailored mutation operation and fine local tuning. Tailored mutation

means that we adaptively change the mutation rate as well as the mutation operation properly after some generations according to the extend of the situations caused by the first and second reasons in the above. This is in contrast to using fixed mutation rate and operation in the standard GA. Fine tuning means that we adaptively change the updating scale (size) of GA operations according to the distance to desired solution or the number of generations. This is different from using fixed updating size in the standard GA, and can solve the problem caused by the third reason in the above.

III. Partitioning of Input Space

In this section, we shall propose several methods to reduce the data rate of motion information. There are two directions which can reduce the data rate of motion information effectively:

- reduce the number of blocks used in compensation;
- reduce the entropy of transformation coefficients.

Partitioning of input space must operate without increasing total frame difference error, or operate under the criteria which can optimally reduce the total data rate.

A. Partitioning of Input Space Using Parameter Similarity

As pointed out, an efficient way to reduce the data rate of motion information is using less blocks, thus less motion information, in motion compensation. Under the premise that the total frame error does not increase, we can achieve this goal by classifying those blocks which experience nearly the same transformation as a group. At first, we take the coefficients of transformation as elements of the pattern directly. In order to determine whether two transformations are similar or the same, we simply compare each corresponding coefficients of two transformations. If the difference of each corresponding coefficients is below a threshold value T_c, we can consider these two transformations as the same one. The algorithm of clustering is described in details as follows.

Step 1: Initialize the first cluster $C_1 = P_1$, where P_1 is the first pattern. Let elements of C_i and P_i be c_{ij} and p_{ij}, $j = 1, 2, \cdots N$, where N is the number of elements in a pattern.

Step 2: Set $i = 2$.

Step 3: Input pattern P_i.

Step 4: For each cluster, calculate distance D_{ik} between pattern P_i and cluster C_k:

$$D_{ik} = \sum_j |p_{ij} - c_{kj}|.$$

Step 5: Determine the c for which $D_{ic} < D_{ik}$ for all $k \neq c$.

Step 6: If D_{ic} is less than a predefined constant T_c, then pattern P_i is assigned to cluster c. Otherwise, a new cluster is created with pattern P_i chosen as the center of the new cluster.

Step 7: If there are more patterns, set $i = i + 1$ and go back to Step 3.

The approach that uses coefficients of transformation as content of a pattern directly suffers from two problems. First, the threshold value T_c is difficult to determine. We need to set T_c as large as possible in a reasonable range in order to make this method effective. However, the effect of the coefficients on the transformed point, (x, y), differs as the value of reference point, (u, v), changes. Second, in the perspective transformation, two transformations whose coefficients differ a lot may produce similar deformation, if their coefficients are proportional.

To tackle the problems mentioned above, in an alternative approach, we do not use the coefficients of transformation as elements of pattern directly; instead, we use as elements of pattern the points that are transformed from a set of fixed reference points by the transformation for clustering. If all corresponding transformed points of two transformations differ below a threshold value T_d, which means the effects of these two transformations are similar, then we can consider these two transformations are in the same cluster. In these two approaches, the first is fast but restricted, while the second is robust and effective.

B. Partitioning of Input Space Using Entropy Criterion

In this section, we shall introduce two operations: merge and split. These two operations can effectively reduce or increase the number of blocks and optimally balance the amount of data rate corresponding to the prediction error and motion parameters, because they operate under entropy criteria. Two entropy criteria for split are stated as follows.

- If the total absolute difference error (or another error measure) of the motion compensated block is above a preset threshold T, the block is split:

$$\text{Error}_{\text{nosplit}} > T \Longrightarrow \text{ split.} \tag{27}$$

- If the extra-cost to send additional motion parameters is worth the gain obtained on the frame difference side, then the block is split into four subblocks:

$$\begin{aligned} &n \cdot (H_{\text{FDnosplit}} - H_{\text{FDsplit}}) \\ &\quad > 4 \cdot H_{\vec{v}\,\text{split}} - H_{\vec{v}\,\text{nosplit}} \Longrightarrow \text{split} \end{aligned} \tag{28}$$

where n is the number of pixels in the block, H_{FDsplit} and $H_{\text{FDnosplit}}$ are their entropy (26) with or without split, respectively, and $H_{\vec{v}\,\text{split}}$ and $H_{\vec{v}\,\text{nosplit}}$ the entropy [(26)] of the motion vectors with or without split, respectively.

The merge operation operates in the sense of reducing total data rate. Unlike the combination process mentioned in the previous section, which keeps the compensated error unchanged or decrease in a small amount, the merge process that operates under the criteria

$$\Delta \text{ Error}_{\text{combine}} < T \Longrightarrow \text{combine} \tag{29}$$

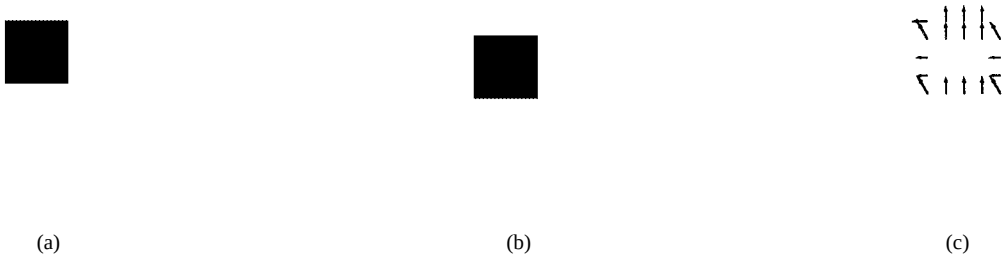
and

$$\begin{aligned} &n \cdot (H_{\text{FDcombine}} - H_{\text{FDnocombine}}) \\ &\quad < 4 \cdot H_{\vec{v}\,\text{nocombine}} - H_{\vec{v}\,\text{combine}} \\ &\Longrightarrow \text{combine} \end{aligned} \tag{30}$$

does not have such restriction. In brief, if merge of blocks, which reduces the number of motion parameters by one-fourth in the quadtree segmentation but increases error of compensation, can achieve a lower data rate, then the blocks are merged as one larger block.

The split operation is also proposed for the goal of reducing total data rate. However, it reversely splits a larger block to small ones and assigns each split blocks a set of motion parameters. The criteria mentioned in (27) and (28) can serve as criteria for split. The main purpose of split operation is to prevent premature combination. Because the estimated motion parameters may be incorrect, the combination of blocks with incorrect parameters corrupts the further learning of these blocks. The split operation is thus proposed to remedy the problem of combination with error motion information.

In comparison with combination methods proposed in the previous section, the merge and split operation perform more effectively in reduction of data rate, because the number of blocks is effectively reduced in the operations.



(a) (b) (c)

Fig. 14. A pair of consecutive images and its motion vectors: (a), (b) moving rectangle, and (c) the motion vectors of the moving rectangle in (a) and (b).

C. Quad-Tree Segmentation

In previous sections, methods of reducing the number of blocks partitioned in an image are introduced. In these methods, images are segmented adaptively into variable sizes, which introduces overhead of representing size and location of blocks in an image. When the partitions are fixed, there is no overhead. However, when the partition of an image is variable, the overhead occupies an amount of bit-rate. Therefore, the quad-tree segmentation is introduced to reduce the overhead.

Quad-trees which represent the segmentation are typically constructed by top-down or bottom-up methods [11]. In the proposed system, a bottom-up construction is adopted. The procedure of merge in the proposed system is described in Sections III-A and III-B. The quad-tree structure can be represented by *treecode* [12]. The treecode is encoded by listing the nodes encountered by a depth-first traversal of the tree structure. If encountered node is not a leaf, then it is represented by "1"; otherwise it is represented by "0". The approach of treecode requires exactly one bit of overhead per node, which is used to distinguish between leaf nodes and internal nodes. Therefore, a small overhead rate is achieved with the structure of quad-tree segmentation.

D. Improvement of Partitioning with Motion Information of Neighbor Blocks

Several methods of partitioning input space are discussed in the previous sections. They rely on the assumption that all motion information of blocks are correct, which is not always true in image sequence. In fact, the motion information which minimizes the compensated error is not necessarily the real motion of objects in an image sequence. We demonstrate the statement by the following example.

In Fig. 14(c), motion vectors of a "moving rectangle" are shown. The real motion vector of the solid block is $(-4, -7)$. Because the block-matching algorithm finds a best fit of a small reference subblock, in which several candidates are possible, the estimated motion vector can be used to perform compensation perfectly though they may not be the actual motion vector.

The estimated motion information which differs from the actual one may be recovered from that of neighbor blocks. Because a moving object in an image sequence is larger than the block size of a minimal block in many occasions, motion information of neighbor blocks are usually the same as, or approximate to, current blocks. The concept of global motion is discussed in many researches on motion estimation or related interests. In [13], a method which reconstructs the frame with the aid of neighbor motion vector is successfully applied to motion compensation.

We propose a recovery operation which tries to recover the incorrect motion vector from the global motion. In the operation,

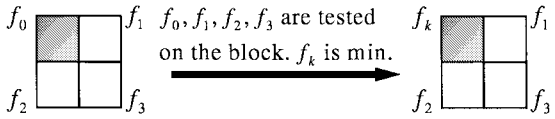


Fig. 15. The proposed operation that tries to recover the incorrect motion information from the global motion.

the parameters of the current block and that of neighbor blocks are tested with motion compensation, and their compensation errors are compared. After comparison, the parameters of this block are then replaced by the ones with the lowest error. This operation is illustrated in Fig. 15. The upper-left block whose motion information is f_0 is tested with motion compensation, but the motion information is temporarily replaced with f_1, f_2, or f_3, each for a time. After comparing the compensation error of each motion information, the one with the lowest error, f_k, becomes the new motion information of the current block, where k is 0, 1, 2, or 3.

IV. Proposed Deformed Image Motion Estimation System

We have discussed in the previous sections two aspects of reducing the amount of information required in a block-matching motion estimation and motion-compensation compression algorithm. In the past, incorporating techniques of two aspects simply means segmentation of images followed by block-matching or by generalized block-matching [14], which lacks the ability of determining as a whole the blocks in the same motion that is more complex than pure translation. In this paper, we propose a system in which those information reduction algorithms can cooperate well.

A. Description of the Proposed System

The architecture of the proposed system is already shown in Fig. 1. The block diagram of the proposed system in a motion estimation and compensation system is shown in Fig. 16. There are two main components in the proposed system. The component of output function and that of input partition are discussed in Sections II and III, respectively. Quad-tree segmentation is adopted in combining and splitting of blocks in the input space.

The operation procedure of the proposed system is described as follows.

- *Step 1:* Input space (compensated image) is partitioned into blocks of minimum size.
- *Step 2:* For each block, learn the associated output function (transformation).
- *Step 3:* The output functions of blocks are replaced and tested with neighbor output function, as described in Section III-D.

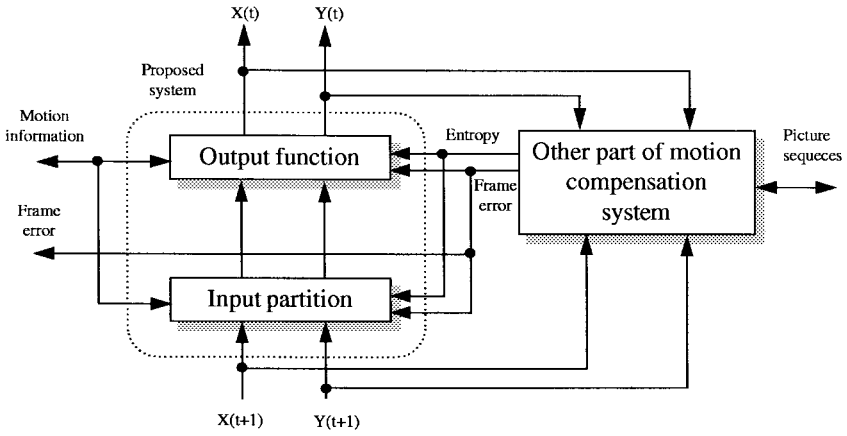


Fig. 16. Block diagram of the proposed deformed image motion estimation system.

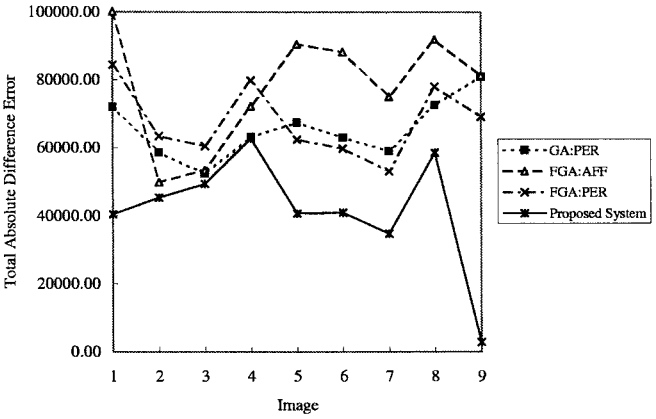


Fig. 17. Compensation error comparison among GA perspective, FGA affine, FGA perspective, and the proposed system.

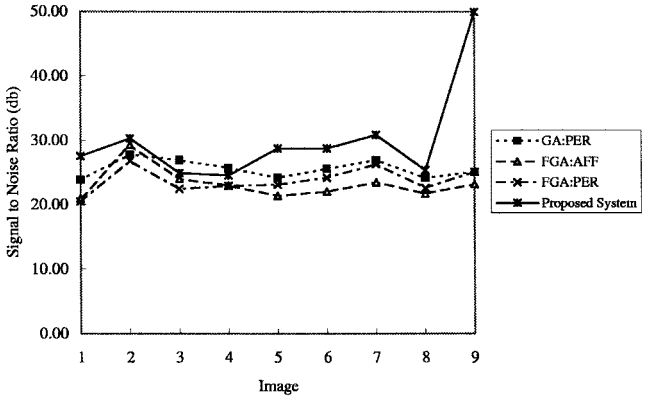


Fig. 18. SNR comparison among GA perspective, FGA affine, FGA perspective, and the proposed system.

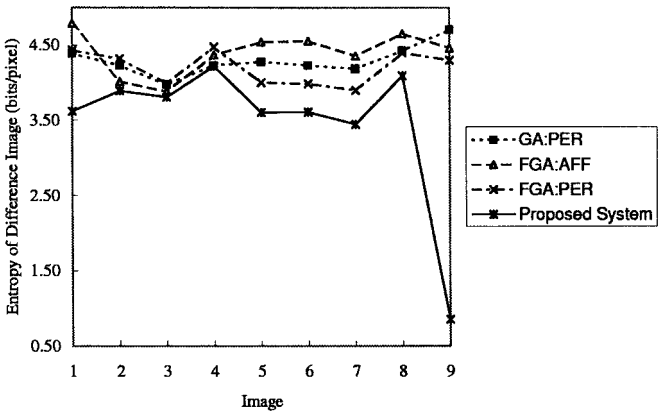


Fig. 19. Entropy comparison among GA perspective, FGA affine, FGA perspective, and the proposed system.

Step 4: Classify transformation parameters according to the methods described in Section III-A. If combination is possible on the basis of quad-tree segmentation, then do the combination.

Step 5: Split blocks according to (27) and (28).

Step 6: For each block, relearn the associated output function. Output functions of neighbor blocks and current block are used as initial parameters (for RLS) or initial population (for GA) of learning process.

Step 7: Combine blocks according to (29) and (30).

Step 8: If the ending condition is not reached, go back to Step 3.

Step 9: For each block, verify the total data rate of the block-matching algorithm when it is applied to the block and that of generalized block-matching algorithm. If the data rate of block-matching is lower, then replace the motion information of the block with the motion vector estimated by the block-matching algorithm.

We explain each step of the operation procedure of the proposed system. In Step 2, GA is adopted to precisely estimate the motion information. In order to prevent GA from being trapped in local minima, Step 3 is adopted to help GA escape from local minima with global motion; moreover, it can speed up the learning of GA in the future relearning. After learning of output function, Step 4 is introduced to reduce the data rate of motion information, including the data rate of representing output function and input partition. Step 5 prevents the premature combination of blocks, as described in Section III-B. Since the input partition and corresponding output function have been modified, the learning of output function is processed again in Step 6 to further reduce the data rate of compensation error. At this point, we can apply Step 7 to combine blocks in the input space. Step 7 is executed after other operations in that it degrades the accuracy of motion information in exchange with less data rate. The ending condition in Step 8 may be that the smallest fitness value falls below a preset threshold, or that the number of generations exceeds

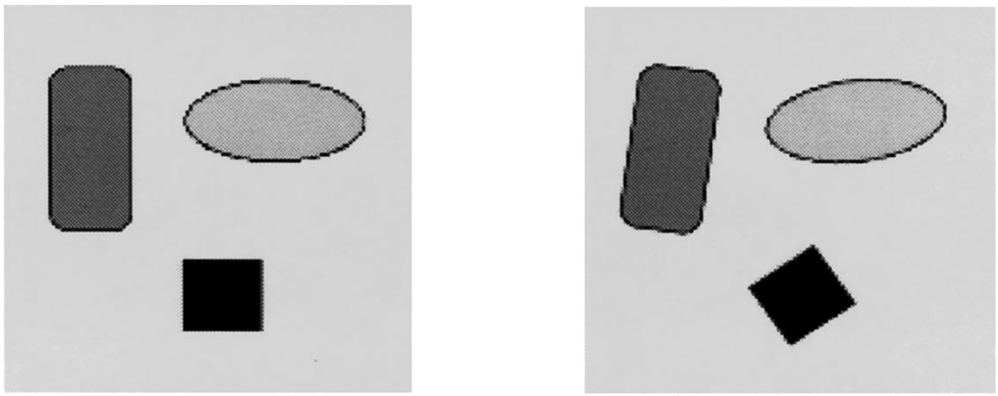


Fig. 20. Test images: three objects.

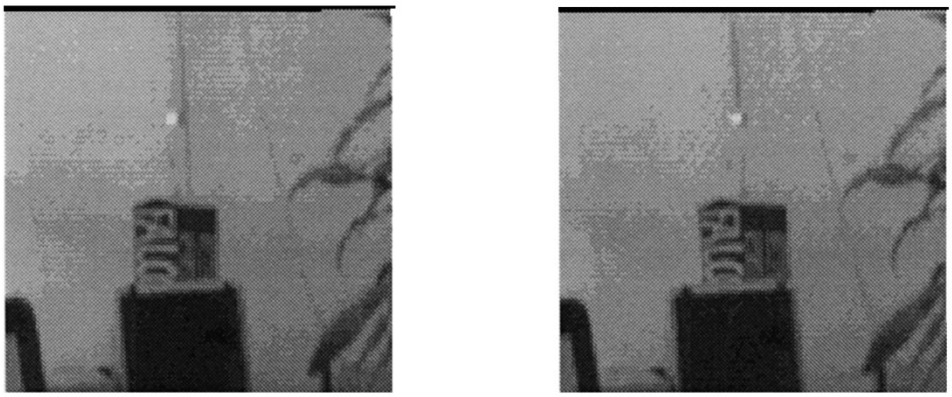


Fig. 21. Test images: the crane system.

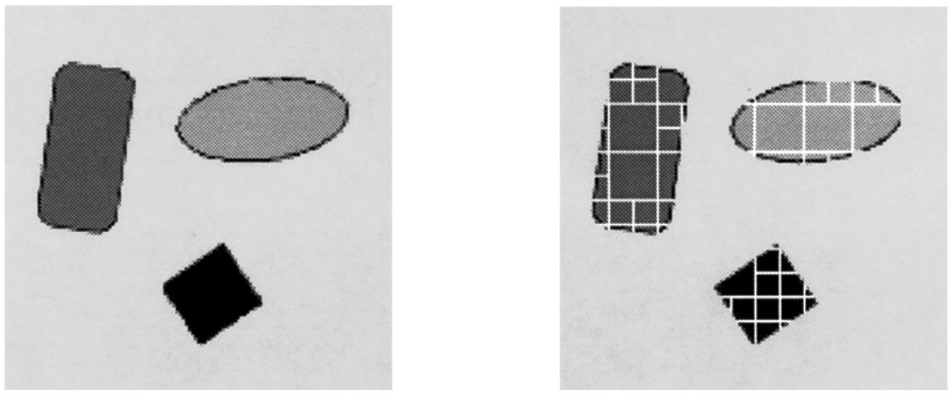


Fig. 22. Reconstructed image and partition map of "three objects."

a preset value. Finally, in Step 9 we further reduce the total data rate. Since the generalized block-matching trades more bits in motion information for less bits in compensation error, sometimes the bits saved in compensation error are less than that consumed in motion information. Thus Step 9 is proposed to tackle this problem.

In brief, the algorithms introduced in the previous two sections are effective in reducing data rate, but they have restriction or drawbacks. The proposed system integrates these algorithms and further improve the performance of each algorithm with the help of other algorithms. It is shown to be superior to the conventional techniques by simulations in the following section.

B. Simulation Results

Nine test images used in Section II-E are tested by the proposed system. In the test, the floating-point GA learning is adopted in the proposed system. Comparisons among the proposed system and GA perspective, floating-point GA affine, floating-point GA perspective learning algorithms are shown in Figs. 17–19. The proposed system is obviously superior to other algorithms.

We then test the proposed system with two pairs of consecutive images. The first pair of images is "three objects" in which three objects rotate in different directions and degrees as shown in Fig. 20. The second pair of image is "the crane system" shown in Fig. 21.

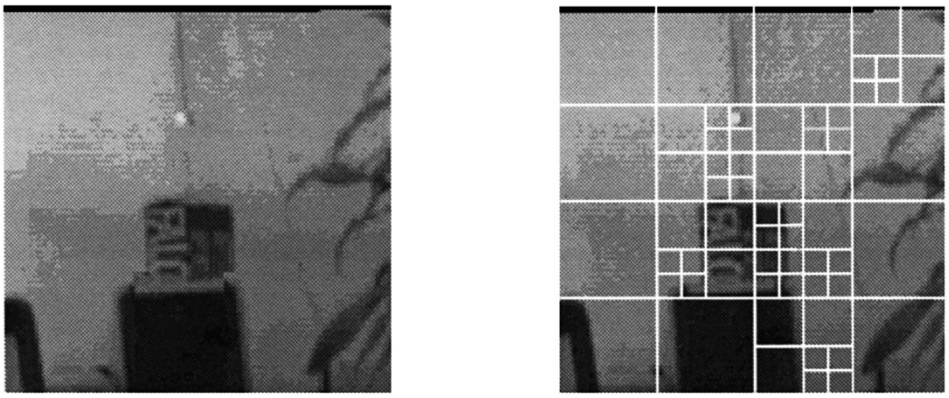


Fig. 23. Reconstructed image and partition map of "the crane system."

TABLE I
COMPENSATED RESULTS OF THE "THREE OBJECTS" IMAGE

	Total Absolute Difference Error	Entropy of Error Image	SNR	Computation Time (second)
BM	40115.00	0.55	22.32	179
RLS:AFF	39621.00	0.68	23.84	1211
GA:AFF	18646.00	0.53	29.48	5396
FGA:AFF	20553.00	0.56	28.35	5102
Proposed System	18232.00	0.52	29.58	5408

TABLE II
COMPENSATED RESULTS OF THE "THE CRANE SYSTEM" IMAGE

	Total Absolute Difference Error	Entropy of Error Image	SNR	Computation Time (second)
BM	36177.00	3.53	32.45	183
RLS:AFF	57803.00	3.87	22.04	1223
GA:AFF	33542.00	3.45	33.06	5402
FGA:AFF	29856.00	3.29	34.15	5211
Proposed System	29681.00	3.28	34.17	5498

TABLE III
DATA RATE COMPARISON BETWEEN THE PROPOSED SYSTEM AND CONVENTIONAL BLOCK-MATCHING TECHNIQUE ON TWO TEST IMAGE SEQUENCES

Test image	Algorithm	Data Rate of Error Image	Data Rate of Motion Information	Number of Blocks	Data Rate of Image Segmentation	Total Data Rate
Three Object	Block Matching	9075.29	2502.92	256.00	0.00	11578.22
	Proposed System	8597.50	2147.61	106.00	137.00	10882.11
Crane System	Block Matching	57819.14	1603.26	256.00	0.00	59422.40
	Proposed System	53704.79	1130.58	61.00	81.00	54916.37

The "three objects" images are artificial and the illumination of objects and background does not change. The other one is taken from real image sequences, and illumination of the contents in image varies. The reconstructed images and partition maps of these images by using the proposed system are shown in Figs. 22 and 23. The performance of the proposed system and BM and other affine transformation learning algorithms are compared, as given in Tables I and II. We can conclude that the proposed system is effective in reducing compensation error even the image sequences are not intensity-invariant. However, the proposed system needs more computation time than the others. If the system is applied to one minute of video image, it will take about 3 246 000 s by using Pentium II 266 MHz (Frame size = 256 × 256, Frame rate = 10).

The comparison in terms of data rate required between the block-matching algorithm and the proposed system is summarized in Table III. The data rate of error image and that of motion information is effectively reduced in the proposed system. The overhead of image segmentation is small due to the quad-tree segmentation. Though the motion information of one block in the proposed system requires more bits than that of block-matching, the total data rate in motion information decreases because the number of blocks is effectively reduced. In "mobile calendar," data rate of motion information is much higher than that of block-matching, because only a small

TABLE IV
SUMMARY OF THE FEATURES AND COMPARISON OF THE PROPOSED ALGORITHM

Algorithm	Feature	Advantages	Disadvantages
RLS method (Subsection 2.2)	Supervised learning with pre-chosen points and corresponding motion vectors estimated by the block-matching algorithm as input-output training pairs	◦ Easy implementation ◦ Fast computation	◦ Unpredictable error due to the possibly wrong motion vectors estimated by the block-matching algorithm
Binary-Coding GA (Subsection 2.3.1)	GA evolution by coding the estimated motion vectors (i.e., correspondence points in the neighbor frame) as a string	◦ Need not estimate the motion vectors by the block-matching algorithm ◦ Higher accuracy than the RLS method	◦ Longer computation time than the RLS method ◦ Cannot cover all transformation types due to the resolution constraint
Floating-point GA (Subsection 2.3.2)	GA evolution by coding the coefficients of transformation function as a string directly	◦ Need not estimate the motion vectors by the clock-matching algorithm ◦ Higher accuracy than the RLS method and binary-coding GA	◦ Longer computation time than the RLS method ◦ Not easy to reach global minimum ◦ Need to control the search range of the coefficients of perspective transformation
Quarter Compensation Algorithm (QCA) (Subsection 2.4)	Decide the gray value of a pixel in a recovered image from the weighted gray values of the four neighboring pixels	◦ More accurate than the conventional motion compensation method that operates only on the integer grids ◦ Can recover the image compressed by the subpixel motion estimation scheme	◦ Longer computation time than conventional integer-grid motion compensation methods
Input space partitioning Using parameter Similarity (Subsection 3.1)	Partition the input space according to the similarity of coefficients of the transformation functions	◦ Intuitive and direct ◦ Short computation time ◦ Reduce data rate	◦ Ignore the possibility that combining dissimilar transformation can reduce total data error
Input space partitioning Using entropy criterion (Subsection 3.2)	Combine two blocks into a single one, if this combination can reduce the data rate (including the increased error data and decreased transformation coefficient data)	◦ Achieve lower data rate than the method based on parameter similarity	◦ Long computation time (can be improved by using the method based on parameter similarity for preprocessing)
Quad-tree segmentation (Subsection 3.3)	Use Q-tree structure to represent the partitioning (formed blocks) of an image	◦ Reduced the data rate for representing the partitioning of an image	◦ The way that an image can be partitioned is restricted
Partitioning Improvement method (Subsection 3.4)	Recover incorrect motion vector from the motion information of neighbor blocks	◦ Find more accurate motion vectors ◦ Can reduce the time in finding the best transformation for each block	◦ Need extra operations
The proposed system (Section 4)	Integrate the algorithms presented in this paper into an efficient deformed image motion estimation system	◦ Efficiently reduce the data rate than conventional methods	◦ Take longer time in image compression than conventional methods ◦ Suitable for off-line coding

number of blocks is combined. However, the amount of data rate that decreases in error image exceeds that increases in motion information, which results in a reduced total data rate.

V. CONCLUSION

A system for deformed image motion estimation and compensation is proposed in this paper. The system is composed of several components and algorithms which try to reduce in different aspects the total data rate of image compression. Table IV summarizes the features of the proposed algorithms. The component concerned with learning of output function is on the purpose of reducing inter-frame redundancy. To learn the output function, we propose the RLS and GA learning of generalized block-matching. In supplement with generalized block-matching, we present a quarter compensation algorithm which can reconstruct the predicted frame according to the motion information of generalized block-matching. The component focused on partitioning of input space is on the purpose of reducing the data rate of motion information. Segmentation algorithms of variable-size block-matching are proposed to reduce the number of parameters. In addition, we introduce the concept of global motion which helps to find the motion of blocks and consequently improve the segmentation of image and reduce the data rate. The proposed system successfully integrates these two components into a whole system. We discuss the drawbacks of each algorithm, and explain how to eliminate the drawbacks with the help of other algorithms. Simulation results show that the proposed system effectively reduces the data rate in image compression.

REFERENCES

[1] G. Tziritas and C. Labit, *Motion Analysis for Image Sequence Coding.* Amsterdam, The Netherlands: Elsevier, 1994.
[2] A. N. Netravali and J. D. Robbins, "Motion compensated television coding part I," *Bell Syst. Tech. J.*, vol. 58, no. 3, pp. 629–668, 1979.
[3] A. N. Netravali and B. G. Haskell, *Digital Pictures Representation and Compression.* New York: Plenum, 1991.
[4] V. Seferidis and M. Ghanbari, "General approach to block-matching motion estimation," *Opt. Eng.*, vol. 32, no. 7, pp. 1464–1474, 1993.
[5] M. H. Chan, Y. B. Yu, and A. G. Constantinides, "Variable size block matching motion compensation with applications to video coding," *Proc. Inst. Elect. Eng.*, vol. 137, no. 4, pp. 205–212, 1990.
[6] F. Dufaux and M. Kunt, "Multigrid block matching motion estimation with an adaptive local mesh refinement," in *Proc. SPIE Visual Communication Image Processing'92*, Boston, MA, Nov. 1992, vol. 1818, pp. 97–109.
[7] F. Dufaux and F. Moscheni, "Motion estimation techniques for digital TV: A review and a new contribution," *Proc. IEEE*, vol. 83, pp. 858–875, June 1995.
[8] J. H. Holland, *Adaptation in Natural and Artificial Systems.* Cambridge, MA: MIT Press, 1992.
[9] D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning.* Reading, MA: Addison-Wesley, 1989.
[10] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs.* New York: Springer-Verlag, 1994.
[11] T. Pavalidis, *Algorithms for Graphics and Image Processing.* Rockville, MD: Computer Science, 1982.

[12] H. Samet and R. E. Webber, "Hierarchical data structures and algorithms for computer graphics part I: Fundamentals," *IEEE Comput. Graph. Applicat.*, pp. 48–68, May 1988.
[13] M. T. Orchard, "Predictive motion field segmentation for image sequence coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 54–70, Feb. 1993.
[14] V. Seferidis and M. Ghanbari, "Generalized block-matching motion estimation using quad-tree structured spatial decomposition," in *Inst. Elect. Eng. Proc. Vis. Image Signal Process*, 1994, vol. 141, no. 6, pp. 446–452.
[15] C. L. Liu, *Elements of Discrete Mathematics.* New York: McGraw-Hill, 1985.
[16] G. Wolberg, *Digital Image Warping.* Los Alamitos, CA: IEEE Comput. Soc. Press, 1990.
[17] J. Apostolopoulos and J. Lim, "Video compression for digital advanced television systems," in *Motion Analysis and Image Sequence Processing*, M. Sezan and R. Lagendijk, Eds. Norwell, MA: Kluwer, 1993, pp. 447–481.
[18] K. W. Chun and J. B. Ra, "An improved block matching algorithm based on successive refinement of motion vector candidates," *Signal Processing: Image Com.*, pp. 115–122, 1994.
[19] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks.* Reading, MA: Addison-Wesley, 1989.
[20] J. R. Jain and A. K. Jain, "Displacement measurement and its applications in interframe coding," *IEEE Trans. Commun.*, vol. COM-29, pp. 1799–1808, Dec. 1981.

POPFNN-AARS(S): A Pseudo Outer-Product Based Fuzzy Neural Network

C. Quek and R. W. Zhou

***Abstract*— A novel fuzzy neural network, the Pseudo Outer-Product-based Fuzzy Neural Network using the singleton fuzzifier together with the Approximate Analogical Reasoning Schema, is proposed in this paper. This network shall henceforth be referred to as the singleton fuzzifier POPFNN-AARS. The singleton fuzzifier POPFNN-AARS employs the Approximate Analogical Reasoning Schema (AARS) [13] instead of the commonly used Truth Value Restriction (TVR) method [19]. This makes the structure and learning algorithms of the singleton fuzzifier POPFNN-AARS simpler and conceptually clearer than those of the POPFNN-TVR model [20]–[22]. Different Similarity Measures (SM) and Modification Functions (FM) [23] for AARS are investigated. The structures and learning algorithms of the proposed singleton fuzzifier POPFNN-AARS are presented. Several sets of real-life data are used to test the performance of the singleton fuzzifier POPFNN-AARS and their experimental results are presented for detailed discussion.**

***Index Terms*— Approximate analogical reasoning schema (AARS), fuzzy rule identification, integrated fuzzy neural networks, modification functions, one-pass learning, pseudo outer-product learning, similarity measures, singleton fuzzifier POPFNN-AARS.**

I. Introduction

Zhou and Quek [20], [21] proposed the structure and learning algorithms of the pseudo outer-product based fuzzy neural network using the truth value restriction method (POPFNN-TVR). This novel fuzzy neural network has successfully been applied in an automatic off-line signature verification system [22]. The experimental results have successfully demonstrated the efficacy of POPFNN-TVR. In the TVR method, implication rules are used to derive the truth values of the consequences from the truth values of the conditions. This process is essentially based on the matrix operation. The effect of such matrix operations on the inference process of POPFNN-TVR is conceptually unclear. To resolve this problem, a variation of POPFNN-TVR is proposed. The novel fuzzy neural network called the pseudo outer-product based fuzzy neural network using the singleton fuzzifier together with the approximate analogical reasoning schema; henceforth referred to as the singleton fuzzifier POPFNN-AARS, uses the approximate analogical reasoning schema (AARS) as its fuzzy inference models. This avoids the computational complexity of the TVR method. A brief description of the AARS fuzzy inference model and its similarity measures (SM's) as well as modification functions (MF's) is given in the following section. Section III briefly contrasts the TVR and AARS fuzzy inference models. The structures and learning algorithms of the proposed singleton fuzzifier POPFNN-AARS's are presented in Sections IV and V, respectively. Section VI discusses the experimental results and analysis on the performance of the proposed fuzzy neural network. and Section VII concludes the work in this paper together with a brief description of the family of POPFNN architectures developed at the Intelligent Systems Laboratory.

Manuscript received August 7, 1997; revised March 1, 1999. This paper was recommended by Associate Editor L. O. Hall.

The authors are with the Intelligent Systems Laboratory, Nanyang Technological University, Singapore 639798.

Publisher Item Identifier S 1083-4419(99)08051-6.

II. Approximate Analogical Reasoning Schema (AARS)

The approximate analogical reasoning schema (AARS) was proposed by Turksen and Zhong [13] as an alternative to the commonly used compositional rule of inference (CRI) [18] and the truth value restriction (TVR) method [19]. It exhibits the advantages of fuzzy set theory and analogical reasoning in expert systems development.

Given an observed fact A' and a simple fuzzy rule "if A then B," the basic idea of AARS is to modify the consequence B of the fuzzy rule according to the closeness of the observed fact A' to the antecedent A. If they are close (similar) enough, in comparison to a threshold value, then the rule can be fired and the conclusion B' can be deduced using some modification techniques. Formally, their 'closeness' is expressed as a *similarity measure* (SM) that is in turn obtained from a *distance measure* (DM) [23]. Once the *similarity measure* $\mathrm{SM}(A, A')$ between A and A' exceeds the value of the threshold τ, the fuzzy rule is fired. A modification function (MF) is subsequently constructed and is used to modify the consequence B of the fuzzy rule to deduce a conclusion B', instead of using Zadeh's CRI. The whole fuzzy inference process of using AARS is shown in Fig. 1. Brief introductions on different SM's and MF's are presented in Sections II-A and II-B, respectively.

A. Similarity Measures

The notion of similarity plays a fundamental role in theories of knowledge and behavior. The theoretical analysis of similarity relations has been dominated by geometric models. These models represent objects as points in some coordinate spaces such that the observed dissimilarity among objects corresponds to the metric distance between the respective points. Many measures of similarity among fuzzy sets have been proposed in the literature, and some have been incorporated into linguistic approximation procedures. In Turksen and Zhong's paper [13], similarity measure (SM) between fuzzy sets is defined as a measurement transformed from a distant measure (DM) by using $\mathrm{SM} = 1/(1 + \mathrm{DM})$. In 1987, Zwick *et al.*