ELSEVIER

Fuzzy Sets and Systems 114 (2000) 281–304

FUZZY sets and systems

www.elsevier.com/locate/fss

A neural fuzzy system for image motion estimation

C.T. Lin *, I.F. Chung, L.K. Martin Sheu

Department of Electrical and Control Engineering, National Chiao-Tung University, 1001 Ta Hseuh Road, Hsinchu, Taiwan

Received April 1998; received in revised form November 1998

Abstract

Many methods for computing optical flow (image motion vector) have been proposed while others continue to appear. Block-matching methods are widely used because of their simplicity and easy implementation. The motion vector is uniquely defined, in block-matching methods, by the best fit of a small reference subblock from a previous image frame in a larger, search region from the present image frame. Hence, this method is very sensitive to the real environments (involving occlusion, specularity, shadowing, transparency, etc.). In this paper, a neural fuzzy system with robust characteristics and learning ability is incorporated with the block-matching method to make a system adaptive for different circumstances. In the neural fuzzy motion estimation system, each subblock in the search region is assigned a similarity membership contributing different degrees to the motion vector. This system is more reliable, robust, and accurate in motion estimation than many other methods including Horn and Schunck's optical flow, fuzzy logic motion estimator (FME), best block matching, NR, and fast block matching. Since fast block-matching algorithms can be used to reduce search time, a three-step fast search method is employed to find the motion vector in our system. However, the candidate motion vector is often trapped by the local minimum, which makes the motion vector undesirable. An improved three-step fast search method is tested to reduce the effect from local minimum and some comparisons about fast search algorithms are made. In addition, a Quarter Compensation Algorithm for compensating the interframe image to tackle the problem that the motion vector is not an integer but rather a floating point is proposed. Since our system can give the accurate motion vector, we may use the motion information in many different applications such as motion compensation, CCD camera auto-focusing or zooming, moving object extraction, etc. Two application examples will be illustrated in this paper.

Keywords: Optical flow; Motion vector; Block matching; Membership function; Backpropagation; Affine motion

1. Introduction

The measurement of optical flow (or image motion vector) is a fundamental problem in the processing of image sequences. The goal is to compute an approximation to the 2-D motion field – a projection of the 3-D velocities of surface points onto the imaging surface. Several methods exist for computing optical flow: block-matching methods, differential techniques, etc. [3]. Regretably, however, these systems are usually

* Corresponding author.

0165-0114/00/$ - see front matter
PII: S0165-0114(99)00075-5

sensitive to noise, do not easily converge to the desired motion estimation, and have poor accuracy. This paper introduces neural fuzzy system to attack these problems.

Many methods for computing optical flow have been proposed and classified by Barron [3]. Differential techniques, phase-based and energy-based methods, and region-based matching techniques are the most widely used means for computing the 2-D motion field. Differential techniques compute velocity from spatiotemporal derivatives of image intensity or filtered versions of the image (using low-pass or band-pass filters) [8,16,26,27,30,31]. One requirement of differential techniques is that $I(\boldsymbol{x}, t)$ (image intensity at pixel location $\boldsymbol{x}$ and time t) must be differentiable. This implies that temporal smoothing at the sensor is needed to avoid aliasing and that numerical differentiation must be done carefully. Another problem of differential techniques is that, if more accurate results from the spatiotemporal patterns of image intensity are desired, then more than two spatiotemporal patterns are required. Regretably, having more than two patterns occupies much memory and takes much time. Image velocity is defined in terms of the phase behavior of band-pass filter outputs in phase-based methods. The zero-crossing techniques [6] are classified as phase-based methods because zero crossing can be viewed as level phase crossing. Energy-based techniques based on the output energy of velocity-tuned filters are also called frequency-based methods owing to the design of velocity-tuned filters in Fourier domain [9]. Region-based matching methods attempt to find the best fit of a small reference subblock from a previous image frame in a larger, search region from the present image frame [10,18,19]. More clearly, an image frame is segmented into two-dimensional small blocks of $N \times N$ pixels. Each block searches for the displacement which produces the best match among the candidate blocks in the present frame. Fast block-matching algorithms can be used to reduce search time [18,19]. However, the fast search algorithms are easily trapped by the local minimum and thus result in considerable error.

Because of noise or aliasing in the image acquisition process, finding a perfect match is impossible. Since fuzzy logic gives greater generality, higher expressive power, an enhanced ability to model real-world problems and, most importantly, a methodology for exploiting the tolerance for imprecision, it can be applied for image motion estimation. The fuzzy logic motion estimator (FME) proposed by Lipp [23] is a method based on block-matching algorithm linked with fuzzy logic. In the FME, for each subblock comparison, the fuzzy logic system classifies an $(M \times M)$ subblock in the search region as to its potential membership in the subblock being matched using fuzzy membership function (FMF). The FME has shown more accurate motion vector estimates with uniform and affine modeled frame-to-frame image motion of real-image data than those from Horn and Schunk's optical flow algorithm [16] or Netravali and Robbins' pel recursive algorithm [28]. In the FME system, however, the generation of membership for each subblock comparison is quite subjective; this subjectivity in turn harms the accuracy of the motion estimation even if the FME technique has better system performance than conventional methods do.

This paper integrates the conventional region-based matching method and a five-layered neural fuzzy network into a system to estimate image motion. The backpropagation (BP) learning rule is used for choosing proper membership functions in order to make this system adapt to different environments. Since the fast block-matching algorithm is a good choice for reducing search time, it was adopted in this system. However, since the local minimum problem appears in the searching process of the conventional fast block-matching algorithms and results in considerable error for the motion estimation, an improved fast block-matching method is proposed to lessen the local minimum problem. Some comparisons of fast search algorithms are made. This system is more reliable and accurate for image motion estimation when compared with other methods including Horn and Schunck's optical flow algorithm, best block-matching, Netravali and Robbins' (NR) pel recursive algorithm, FME, and three-step fast block-matching algorithm. As a summary, the significant characteristics of the proposed system include that it is more robust than several previous methods; it can estimate motion vector more accurately than several other methods; the system parameters are tunable through neural learning; each subblock in the search region will be assigned similarity membership contributing different degrees to the motion estimation; and it adopts a modified fast block-matching method which can reduce local minimum problem.

The architecture and learning algorithm of the proposed system are described in Section 2. Some performance comparisons are made in Section 3. In Section 4, the applications of the proposed system are illustrated. Conclusions are summarized in Section 5.

2. The architecture of the neural fuzzy motion estimator

Fig. 1 shows the system architecture of the proposed neural fuzzy motion estimator. It is a five-layered network structure. Fig. 2 is the functional block diagram of this system. First, two image sequences containing moving objects were processed by a region-based matching technique attempting to find a best fit of a small reference subblock from a previous image frame in a larger search region from the present image frame. Second, the outputs from region-based matching were fed into a neural fuzzy motion estimator which computed the motion field of the image sequences.

Nodes at layer one were input units representing inputs in linguistic variables. Nodes at layer two were input term nodes which acted as membership functions for performing the similarity measure of each subblock in the search region of the present image frame. Layer 3 represented the rule nodes. Layer 4 contained output term nodes and Layer 5 contained two output nodes producing the crisp image velocity value in the x_1 and x_2 directions, respectively. In the proposed system, the motion vector was not decided merely by the best fit of the reference subblock from a previous image frame in the larger search region of the present image frame. For each subblock matching, a fuzzy membership function (FMF) will give potential membership to each subblock in the search region. A potential membership represents the confidence measure of motion vector. This was performed over all subblocks in the search region. The matching degrees then passed through fuzzy reasoning process (Layers 3 and 4) and the defuzzification process (Layer 5) to obtain two crisp outputs; these outputs indicate the motion vector of the reference subblock in the search region. To train (or, more precisely, to calibrate) the proposed network, a known, uniformly displaced and/or affine modeled frame-to-frame image was used to get the desired motion vectors for learning the proper parameters of the FMFs. Moreover, a modified fast search scheme was used in this present system. In both learning or processing, the fast search scheme increased the learning rate or reduced the search time if it could find the global minimum of the matching criterion correctly in the region-based matching algorithm. Each part of this system will now be described in greater detail.

2.1. Block-matching Algorithm

A block-matching algorithm attempts to find a best fit of a small reference subblock from a previous image frame in a larger search region from the present image frame as depicted in Fig. 3. As the figure shows, this algorithm compares the $(M \times M)$ reference subblock with each of those having the same-sized portions within the $(N \times N)$ search region. In general, the size of the search region is larger than the reference's but much smaller than the image frame containing it. There are many choices for determining the best match, e.g., the *cross-correlation function*, *mean square error* (*MSE*), *mean absolute difference* (*MAD*) [5], etc. If one chooses the *mean square error* (*MSE*) as the matching criterion, one has

$$MSE(d_1, d_2) = \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} [I_s(i + d_1, j + d_2) - I_r(i, j)]^2, \tag{1}$$

where I_r is the reference subblock in the previous image frame, I_s is the search region in the present image frame, $N \times N$ is the size of search region, and $M \times M$ is the size of reference subblock. The displacement (d_1, d_2) which minimizes $MSE(d_1, d_2)$ is selected as the motion vector.

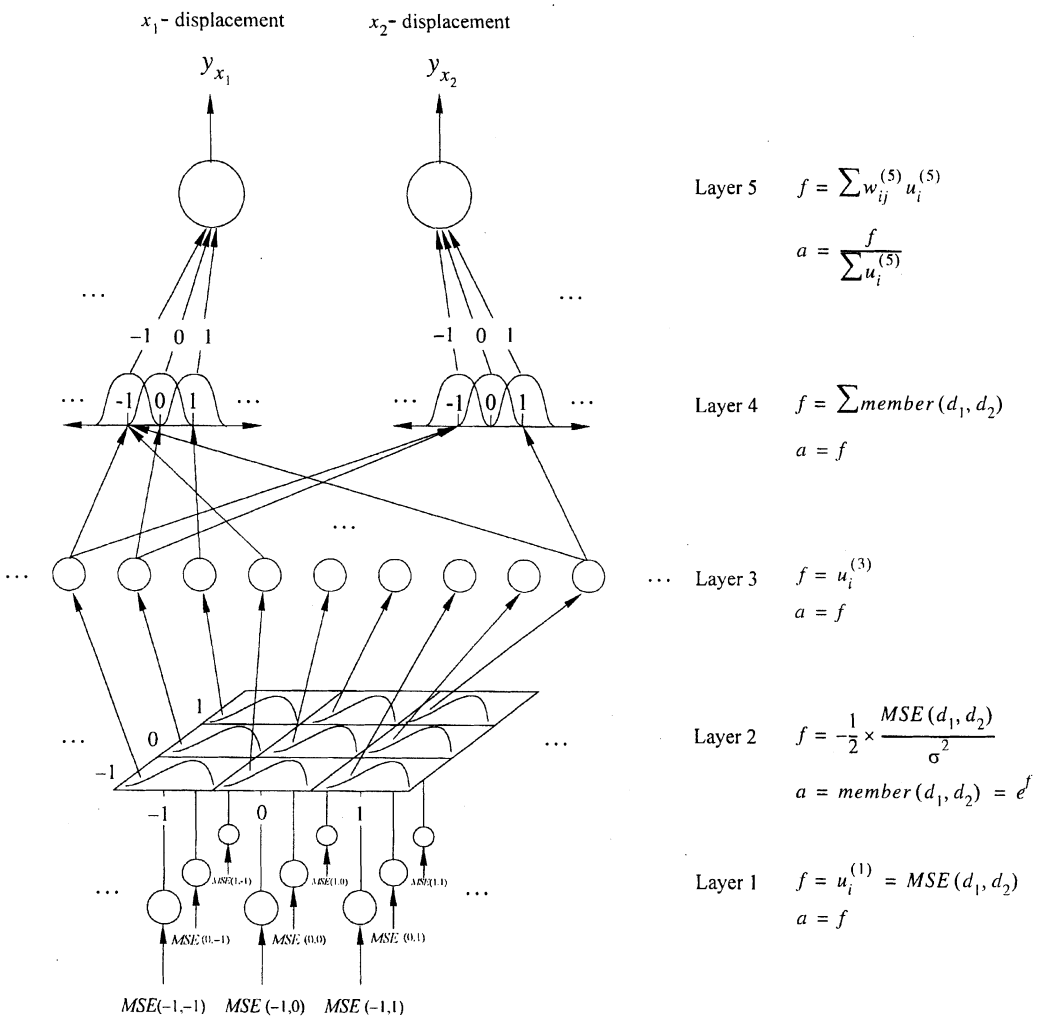

Fig. 1. System architecture of the proposed neural fuzzy motion estimator (NFME).

Layer 1 (Input layer): The nodes in this layer just transmit input values to the next layer directly. That is,

$$f = u_i^{(1)} \quad \text{and} \quad a = f, \tag{2}$$

where $u_i^{(1)} = MSE(d_1, d_2)$ and i means the ith compared subblock corresponding to displacement (d_1, d_2) in the search region.

Layer 2 (Input term nodes): A single node was used to act as a membership function and a bell-shaped Gaussian function was selected as the input membership function,

$$B(x, m) = \mathrm{e}^{-1/2 \cdot (x-m)^2/\sigma^2}, \tag{3}$$

where x is the input to the node from Layer 1, and m and σ are the center and width of the bell-shaped function, respectively. In this layer, the center of the bell-shaped function was set to zero and the width was

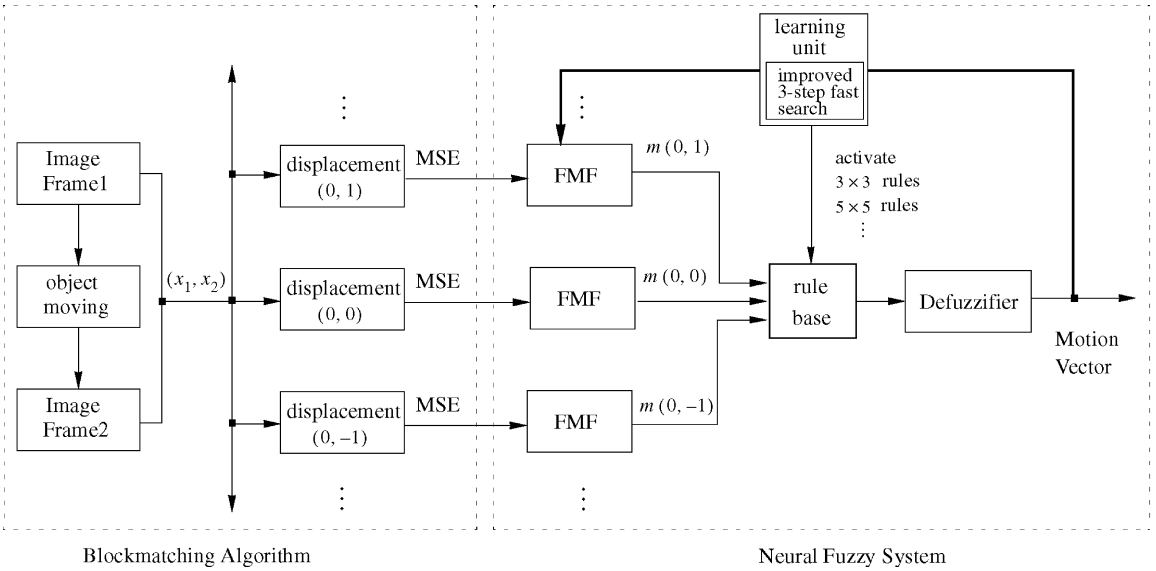

Fig. 2. The functional diagram of the proposed neural fuzzy system for image motion estimation.

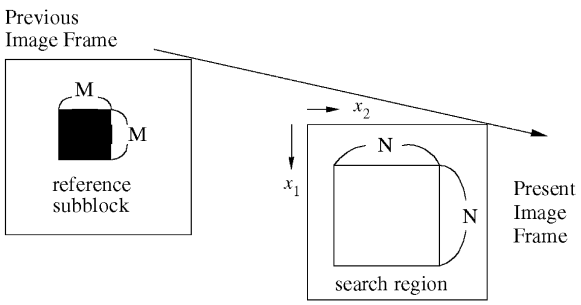

Fig. 3. Block-matching algorithm.

tuned in the learning procedure to fit different environments. Hence, we have

$$f = -\frac{1}{2} \cdot \frac{MSE(d_1, d_2)}{\sigma^2} \quad \text{and} \quad a \equiv member(d_1, d_2) = e^f, \tag{4}$$

where $member(d_1, d_2)$ is the membership value representing the confidence measure of each subblock which displaces d_1 and d_2 pixels in horizontal and vertical axes, respectively. If $member(d_1, d_2)$ is close to 1, then the subblock's motion vector probably equals (d_1, d_2); if $member(d_1, d_2)$ is close to 0, then the subblock probably does not have a motion vector equal to (d_1, d_2).

Layer 3 (Rule nodes): The fuzzy inference rules for generating fuzzy outputs are of this form:

If $member(d_1, d_2)$ then $y_{x_1} = D_1$ and $y_{x_2} = D_2$,

where $member(d_1, d_2)$ is given in Eq. (4), D_i is the fuzzy number of d_i (e.g., if $d_1 = 1$, then D_1 is the fuzzy number "1"); y_{x_1} and y_{x_2} are the output linguistic variables representing the displacement (in pixels/time-step) in the horizontal and vertical axes, respectively. The output membership functions defined as uniformly spaced *Gaussian functions* with overlap were used in this present system. The membership functions for

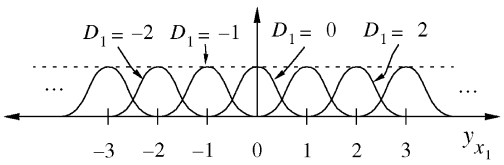

Fig. 4. The output membership functions.

output linguistic variable, y_{x_1}, are depicted in Fig. 4. The membership functions for y_{x_2} are exactly the same. From the above fuzzy rule, there is only one input for each rule node in Layer 3 (see Fig. 1). Hence, the (rule) node in this layer just transmits the input values to the next layer directly. That is,

$$f = u_{\mathrm{i}}^{(3)} \quad \text{and} \quad a = f. \tag{5}$$

In total, there are $(N-M)^2$ rules for all possible displacements ($disp_1$, $disp_2$) of the reference subblocks in the search region. In other words, there are $(N-M)^2$ rule nodes in Layer 3.

Layer 4 (Output term nodes): The links at layer four should perform the fuzzy OR operation to integrate the fired rules which have the same consequence. Hence, we have

$$f = \sum member(d_1, d_2) \quad \text{and} \quad a = f \tag{6}$$

and the link weight is $w_{\mathrm{i}}^{(4)} = 1$.

Layer 5 (Output nodes): The node at this layer performs the defuzzification process. The following function can be used to approximate the COA defuzzification method:

$$f = \sum w_{ij}^{(5)} u_{\mathrm{i}}^{(5)} \quad \text{and} \quad a = \frac{f}{\sum u_{\mathrm{i}}^{(5)}}, \tag{7}$$

where $w_{ij}^{(5)}$ is the link weight assigned as displacement value d_i.

2.2. Learning unit

Gaussian function is used as the fuzzy membership function (FMF) in Layer 2 (see Figs. 1 and 2) with parameter σ to be determined adaptively for various environments. The *backpropagation* (BP) learning rule is used to find the parameter's proper value. Consequently, training data indicating desired motion vectors are required for learning. The training data are obtained from a real-image synthetically displaced with affine motion, which is defined by

$$\begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} 1-\cos(\theta_{\mathrm{r}}) & \sin(\theta_{\mathrm{r}}) \\ -\sin(\theta_{\mathrm{r}}) & 1-\cos(\theta_{\mathrm{r}}) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \tag{8}$$

where $[d_1, d_2]^{\mathrm{T}}$ is the motion vector of the synthetic image, θ_{r} is the clockwise angle of rotation, and u_1, u_2 are the uniform components of the motion in the x_1 (vertical), x_2 (horizontal) directions, respectively. The detailed derivation of the learning algorithm is shown below.

The output error is defined as

$$E = \tfrac{1}{2}[y_{x_1} - \hat{y}_{x_1}]^2 + \tfrac{1}{2}[y_{x_2} - \hat{y}_{x_2}]^2, \tag{9}$$

where y_{x_1}, y_{x_2} are the desired outputs in the x_1 and x_2 directions, respectively, and $\hat{y}_{x_1}, \hat{y}_{x_2}$ are the corresponding actual outputs, which are given by

$$\hat{y}_{x_1} = \frac{\sum_i \sum_j m_i w a_{ij}}{\sum_i \sum_j w a_{ij}}, \tag{10}$$

$$\hat{y}_{x_2} = \frac{\sum_i \sum_j m_j w a_{ij}}{\sum_i \sum_j w a_{ij}}, \tag{11}$$

$$a_{ij} = \mathrm{e}^{-(1/2)x_{ij}^2/\sigma_{ij}^2}, \tag{12}$$

$$x_{ij}^2 = MSE(i,j), \tag{13}$$

where $MSE(i,j)$ is the mean square error for determining the matching criterion defined in Eq. (1), w is the width of output membership function, and m_i, m_j are the centers of output membership functions in the x_1 and x_2 directions, respectively. The backpropagation learning rule then gives

$$\sigma'_{ij} = \sigma_{ij} + \Delta\sigma_{ij}, \tag{14}$$

$$\Delta\sigma_{ij} = -\eta \frac{\partial E}{\partial \sigma_{ij}} \tag{15}$$

$$= -\eta \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \sigma_{ij}} \tag{16}$$

$$= -\eta \left[\frac{\partial E}{\partial \hat{y}_{x_1}} \frac{\partial \hat{y}_{x_1}}{\partial \sigma_{ij}} + \frac{\partial E}{\partial \hat{y}_{x_2}} \frac{\partial \hat{y}_{x_2}}{\partial \sigma_{ij}} \right] \tag{17}$$

$$= \Delta\sigma_{ij}^1 + \Delta\sigma_{ij}^2, \tag{18}$$

where $\Delta\sigma_{ij}^1 = -\eta \partial E/\partial \hat{y}_{x_1} \cdot \partial \hat{y}_{x_1}/\partial \sigma_{ij}$ and $\Delta\sigma_{ij}^2 = -\eta \partial E/\partial \hat{y}_{x_2} \cdot \partial \hat{y}_{x_2}/\partial \sigma_{ij}$ which can be calculated as follows:

1. $\dfrac{\partial E}{\partial \hat{y}_{x_1}}$

$$\frac{\partial E}{\partial \hat{y}_{x_1}} = -[y_{x_1} - \hat{y}_{x_1}], \tag{19}$$

2. $\dfrac{\partial \hat{y}_{x_1}}{\partial \sigma_{ij}}$

$$\frac{\partial \hat{y}_{x_1}}{\partial \sigma_{ij}} = \frac{\partial \hat{y}_{x_1}}{\partial a_{ij}} \cdot \frac{\partial a_{ij}}{\partial \sigma_{ij}}$$

$$= [y_{x_1} - \hat{y}_{x_2}] \sum_{i=1}^{M} \sum_{j=1}^{M} \left[\frac{\left(\sum_i \sum_j m_i \right) \cdot \sum_i \sum_j a_{ij} - \sum_{i=1}^{M} \sum_{j=1}^{M} m_i a_{ij}}{\sum_{i=1}^{M} \sum_{j=1}^{M} a_{ij}}^2 \times \mathrm{e}^{-(1/2)x_{ij}^2/\sigma_{ij}^2} \times \frac{x_{ij}^3}{\sigma_{ij}^3} \right]. \tag{20}$$

In Eq. (20), the term $\sum_i \sum_j m_i = 0$, because the output membership functions are chosen as *Gaussian* functions symmetrical with respect to the origin. Hence, $\Delta\sigma_{ij}^1$ becomes

$$\Delta\sigma_{ij}^1 = \eta [y_{x_1} - \hat{y}_{x_1}] \sum_{i=1}^{M} \sum_{j=1}^{M} \left[\frac{-\sum_{i=1}^{M} \sum_{j=1}^{M} m_i a_{ij}}{\sum_{i=1}^{M} \sum_{j=1}^{M} a_{ij}}^2 \times \mathrm{e}^{-(1/2)x_{ij}^2/\sigma_{ij}^2} \times \frac{x_{ij}^3}{\sigma_{ij}^3} \right]. \tag{21}$$

Similarly, $\Delta\sigma_{ij}^2$ can be derived as

$$\Delta\sigma_{ij}^2 = \eta[y_{x_2} - \hat{y}_{x_2}] \sum_{i=1}^{M}\sum_{j=1}^{M} \left[\frac{-\sum_{i=1}^{M}\sum_{j=1}^{M} m_j a_{ij}}{\sum_{i=1}^{M}\sum_{j=1}^{M} a_{ij}}^2 \times e^{-(1/2)x_{ij}^2/\sigma_{ij}^2} \times \frac{x_{ij}^3}{\sigma_{ij}^3} \right]. \tag{22}$$

The final update rule is

$$\Delta\sigma_{ij} = \Delta\sigma_{ij}^1 + \Delta\sigma_{ij}^2. \qquad \square \tag{23}$$

Eq. (23) is an update rule for the width of FMFs. Since objects can move arbitrarily in image sequences, it is reasonable to let all the FMFs have the same width, σ. The common width parameter, σ, is updated by the average value of all $\Delta\sigma_{ij}$ for each epoch of learning.

2.3. Fast search algorithm

In the proposed neural fuzzy motion estimator, the motion vectors in image sequences are determined by all the confidence measures of candidate locations in the search region, and especially by those locations with greater membership values. The greatest membership will fall in the location whose MSE value is the global minimum in the search region. Moreover, it is reasonable to assume that the significant memberships are at the locations around that with the greatest membership. Hence, when the global minimum of MSE is found in the search region, we can use $3\times 3, 5\times 5, \ldots$ MSEs centered at the global minimum of MSE to calculate the approximate motion vector, which will be close to the desired motion vector. If the global minimum of MSE can be found without full search, the search time can be reduced and the learning rate can be increased. Hence, a modified fast search algorithm is incorporated into the learning unit of this present system to speed up both the learning and processing.

To reduce the heavy computational cost resulting from the massive number of candidate locations, the three-step fast block search algorithm [19] searches for the best motion vector in a coarse-to-fine manner. Fig. 5 illustrates the procedure of the three-step search with an example of motion vector $(-7, 5)$. In the first step, nine sparsely located candidates are evaluated and the one with a minimal MSE is picked out. In the second step, the search focuses on the area centered at the winner of the previous step, but distances between candidate locations are shortened by one-half. In the same manner, the third step compares the MSE's of the nine locations around the winner found in the second step and then gives the final motion vector. The three-step search algorithm commonly uses the range of $d_1 = d_2 = 7$. In this manner, the number of search locations will decrease to $1/9$ of the number of the full search approach.

In practice, many fast block-matching algorithms are trapped by the local minimum, the three-step search algorithm included. This paper thus proposes a modified fast search algorithm. In the second and third steps, coarse-to-fine search is replaced by full search to reduce the extent to which that searching process is trapped by local minimum. With this modification, the number of search locations will decrease to $1/5$ of the number of the exhaustive search. Some comparisons of the full search, the original 3-step search, and the modified 3-step search are made in the following.

2.4. Experimental results of learning

The target motion vectors of the image sequences from the affine motion equation (Eq. (8)) can be found for updating the parameter, σ, of the fuzzy membership functions. The update rule is given by Eq. (23). Three cases are considered to show the learning ability of the proposed neural fuzzy motion estimator:

- Learning an image with different motion parameters: The training data are made using a real-image synthetically displaced with affine motion equation (Eq. (8)). Different rotation angles θ_r, $4^\circ, 8^\circ, -4^\circ, -8^\circ$ and

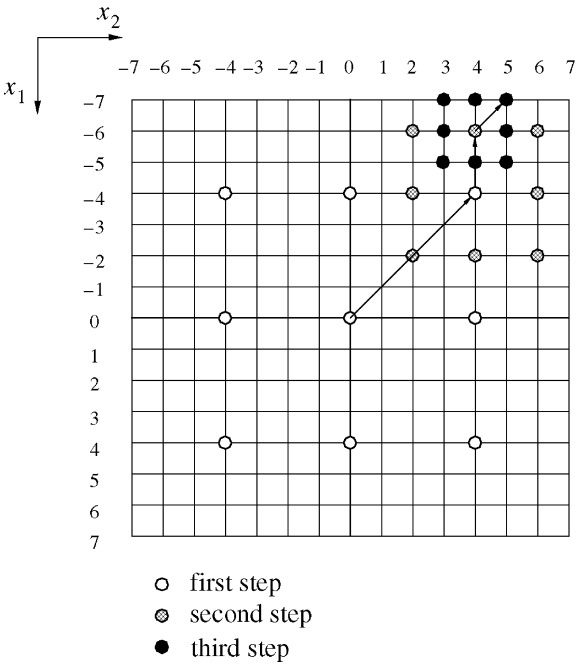

Fig. 5. Three step fast block search algorithm.

Fig. 6. (a), (b) Two different images for learning. (c), (d) Two rotated images for learning.

different uniform translation components (u_1, u_2), $(2,2)$, $(-2,2)$, $(2,-2)$, $(-2,-2)$ are used to produce training patterns representing the general motion of moving objects when they move arbitrarily in the image.

- Learning two different images: replacing the above image with a new one, redoing the same experiment, and observing the value of σ.
- Learning an image using different numbers of rules: Different numbers of rules, $3\times 3, (1\times 1$ is the best blockmatching), 5×5, 7×7, 9×9, are used for an image to see the value of σ.

Two different images and two rotated ones for learning are shown in Fig. 6. Some training patterns represented as the motion field flow map are depicted in Fig. 7. The final values of σ updated by different training patterns made by affine motion from two different images are listed in Table 1.

The average value of σ is obtained by averaging all the values learned from different training patterns, and this average will serve as the width parameter of input membership functions. Two average values of σ for

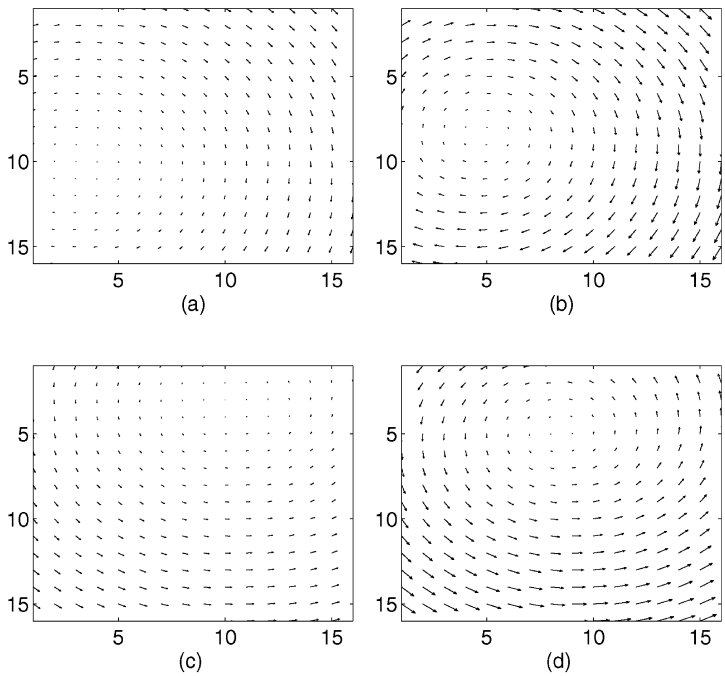

Fig. 7. The desired motion vectors. (a) $(u_1, u_2) = (2, 2)$, $\theta_r = 4°$. (b) $(u_1, u_2) = (2, 2)$, $\theta_r = 8°$. (c) $(u_1, u_2) = (2, 2)$, $\theta_r = -4°$. (d) $(u_1, u_2) = (2, 2)$, $\theta_r = -8°$.

Table 1
The learned values of the width parameter σ

θ_r / (u, v)	Image 1				Image 2			
	4°	8°	−4°	−8°	4°	8°	−4°	−8°
(2, 2)	6.025	7.339	5.601	5.304	2.718	2.332	1.602	3.002
(2, −2)	5.657	8.766	6.454	4.748	2.228	1.978	1.843	2.261
(−2, 2)	5.488	6.313	4.972	5.822	2.443	2.339	2.102	2.410
(−2, −2)	6.142	7.155	5.444	5.842	1.762	2.002	2.403	2.302
Final value	6.067				2.233			

two different images are observed in Table 1, indicating that the parameter of fuzzy membership function must be chosen carefully. Hence, one should tune the parameters of input membership functions to find a proper value for fitting different circumstances in practical applications.

As observed from Table 2, different numbers of rules result in different values of σ. That is, if one wants to produce crisp values of motion estimation using different number of rules, then the value of σ must be chosen properly.

2.5. Comparisons of various fast block-matching algorithms

The performance of fast search algorithms is affected by local minima. Hence, an algorithm with a bigger search range than the original three-step search algorithm was proposed to reduce the error. Using the modified

Table 2
The learned values of the width parameter σ when learning using different number of rules

Rule_no.	Image 1	Image 2
3×3	5.341	5.92
5×5	7.597	6.01
7×7	19.91	6.61
3×3	19.85	8.43

three-step search algorithm described previously, the number of search locations increased from 1/9 to 1/5 of the number of the exhaustive approach, compared with the original 3-step search algorithm. Two synthetically uniform translated images in Fig. 6 were used to examine the performance of the fast search and improve fast search algorithms. The full search algorithm may be regarded as having the perfect performance (i.e., 0% error rate) in the searching process, then the error percentage of three-step and modified three-step methods with respect to the full search algorithm was obtained. The original three-step search scheme had a 54.32% error rate, while the modified method with bigger search range had about 20.26% error. The local minimum problem still existed in the improved method. Hence, the learning rate could be increased effectively if the fast search algorithm worked well in the search process.

3. Test and comparisons of synthetic image sequences

This section examines the performance of the proposed and some existing image motion estimation techniques on synthetic image sequences for which 2-D motion fields have been known. Before discussing the experimental results, however, it is essential to describe the image sequences used for comparison and the measures of error.

3.1. Synthetic image sequences

The main advantage of using synthetic inputs is that the 2-D motion fields and scene properties can be controlled and tested in a methodical way. One may use the known motion vectors to quantify the performance of a specific algorithm. On the other hand, it must be kept in mind that such inputs are usually clean signals (involving no specularity, shadowing, transparency, occlusion, etc.) and therefore this measure of performance should be taken as an optimistic bound on the expected errors with real-image sequences. The synthetic images are made by means of affine motion (shown in Eq. (8)), and the synthetic image sequences include

- *Translation sequences*: Results of the case with velocity $v=(-3,2)$ are reported (see Fig. 8). Hence, the desired motion vectors will fall in location $(-3,2)$ of the 2-D plane if one represents the motion vectors using Cartesian coordinate system. The better the motion estimation is, the closer the estimated motion vectors will be to the point $(-3,2)$. This phenomenon can be observed from the scatter map of the motion vectors.
- *Rotation sequences*: For this experiment, the image is rotated 6° clockwise using affine modeled motion to produce the present image frame (see Fig. 8).

3.2. Image motion estimation techniques for comparisons

Synthetic image sequences of moving brightness patterns as mentioned above have been processed by many different methods including Horn and Schunck's, FME, Netravali and Robbins' pel recursive algorithm, best-fit block-matching, and this paper's neural fuzzy motion estimator. These methods are described as follows.

(a) (b) (c)

Fig. 8. Testing images. (a) Original image. (b) Image after translation, $v=(-3,2)$. (c) Image after rotation, $\theta_r=6°$ clockwise.

- Horn and Schunck (HS) [16]: Horn and Schunck [16] combined the gradient constraint with a global smoothness term to constrain the estimated velocity field $\boldsymbol{v}=(u(\boldsymbol{x},t),v(\boldsymbol{x},t))$ in minimizing the error equation. The gradient constraint is

$$\bigtriangledown I(\boldsymbol{x},t)\cdot v+I_t(\boldsymbol{x},t)=0, \tag{24}$$

where $I_t(\boldsymbol{x},t)$ denotes the partial time derivative of $I(\boldsymbol{x},t)$, $\bigtriangledown I\cdot v$ denotes the usual dot product, and $\bigtriangledown I(\boldsymbol{x},t)=(I_x(\boldsymbol{x},t),I_y(\boldsymbol{x},t))^{\mathrm{T}}$. The total error to be minimized is

$$\int_D[(\bigtriangledown I\cdot v+I_t)^2+\alpha^2(||\bigtriangledown u||_2^2+||\bigtriangledown v||_2^2)]\,\mathrm{d}\boldsymbol{x} \tag{25}$$

defined over a domain D, where the magnitude of α reflects the influence of the smoothness term. This study used $\alpha=50$ because it produced better results in most test cases. Iterative equations are used to minimize Eq. (25) to obtain motion vectors:

$$u_{k+1}=\bar{u}^k-\frac{I_x[I_x\bar{u}^k+I_y\bar{v}^k+I_t]}{\alpha^2+I_x^2+I_y^2}, \tag{26}$$

$$v_{k+1}=\bar{v}^k-\frac{I_y[I_x\bar{u}^k+I_y\bar{v}^k+I_t]}{\alpha^2+I_x^2+I_y^2}, \tag{27}$$

where k denotes the iteration number, u^0 and v^0 denote initial velocity estimates (here set to zero), and $\bar{u}^k$, $\bar{v}^k$ denote neighborhood averages of u^k and v^k. At least 100 iterations were used in this present study to obtain better results. The method with spatiotemporal aliasing was implemented and the subsequent derivative estimates [16] were also improved.

- Fuzzy logic motion estimator (FME) (Lipp [23]): Based on the block-matching algorithm, this technique calculates the displaced frame distortion (DFD) using

$$DFD(d_1,d_2)=\sum_{i=1}^{M}\sum_{j=1}^{M}[I_s(i+d_1,j+d_2)-I_r(i,j)]^2, \tag{28}$$

where (d_1,d_2) is the motion vector, M is the reference block size, $I_r(i,j)$ is the reference subblock in the previous image frame and $I_s(i,j)$ is the search region in the present image frame (with origin corresponding to the upper left-hand corner of the subblock in the previous image). The $DFD(0,0)$ is used as

a normalization factor in defining the following membership value, which plays the same role as that in Eq. (4):

$$member(d_1,d_2)=\begin{cases}1-\dfrac{5DFD(d_1,d_2)}{DFD(0,0)} & \text{if } \dfrac{5DFD(d_1,d_2)}{DFD(0,0)}<1,\\ 0 & \text{otherwise.}\end{cases} \tag{29}$$

In defuzzification, uniformly spaced triangular membership functions with little overlap (similar to those shown in Fig. 4) are used for the output membership functions. The outputs are combined using fuzzy centroid defuzzification to produce sharp outputs for directions x_1 (vertical) and x_2 (horizontal).

- Netravali and Robbins' Pel recursive algorithm (NR) [28]: This motion estimation method attempts to minimize recursively a certain quantity (function of the motion estimation error). If the changes in successive image frames are due to translation of an object, then the algorithm iterates in a gradient or steepest descent direction such that the consecutive estimates converge to an estimate of translation. It is noted that since the NR approach considers only the point-to-point displacement, it relates to image translation only rather than to rotation. Assume $I(\boldsymbol{x},t)$ and $I(\boldsymbol{x},t-\tau)$ are the intensity values of the two successive frames as a function of spatial location $\boldsymbol{x}$ (a two-dimensional vector) at time t. The time between the two frames is τ. If an object moves in translation, then in the moving area one has

$$I(\boldsymbol{x},t)=I(\boldsymbol{x}-\boldsymbol{D},t-\tau), \tag{30}$$

where $\boldsymbol{D}$ is the translation vector of the object during the time interval $[t-\tau,t]$. The frame difference at spatial position $\boldsymbol{x}$ is given by

$$FDIF(\boldsymbol{x})=I(\boldsymbol{x},t)-I(\boldsymbol{x},t-\tau) \tag{31}$$

$$=I(\boldsymbol{x},t)-I(\boldsymbol{x}+\boldsymbol{D},t). \tag{32}$$

In the recursive estimation scheme, the displaced frame difference $DFD(\boldsymbol{x},\hat{\boldsymbol{D}}^{i-1})$ analogous to $FDIF(\boldsymbol{x})$ is defined by

$$DFD(\boldsymbol{x},\hat{\boldsymbol{D}}^{i-1})=I(\boldsymbol{x},t)-I(\boldsymbol{x}-\hat{\boldsymbol{D}}^{i-1},t-\tau), \tag{33}$$

$$\hat{\boldsymbol{D}}^{i}=\hat{\boldsymbol{D}}^{i-1}+\boldsymbol{U}^{i}, \tag{34}$$

where $\hat{\boldsymbol{D}}^{i}$ is the ith displacement estimate, $\hat{\boldsymbol{D}}^{i-1}$ is an initial estimate of $\hat{\boldsymbol{D}}^{i}$, and $\boldsymbol{U}^{i}$ is the update of $\hat{\boldsymbol{D}}^{i-1}$ for making it more accurate (i.e., the estimate of $\boldsymbol{D}-\hat{\boldsymbol{D}}^{i-1}$). Then the estimator can be derived as

$$\hat{\boldsymbol{D}}^{i}=\hat{\boldsymbol{D}}^{i-1}-\varepsilon DFD(\boldsymbol{x}_a,\hat{\boldsymbol{D}}^{i-1})\bigtriangledown I(\boldsymbol{x}_a-[\hat{\boldsymbol{D}}^{i-1}],t-\tau), \tag{35}$$

where $\bigtriangledown$ is the gradient with respect to $\boldsymbol{x}$, ε is a positive scalar constant, and a pixel at location $\boldsymbol{x}_a$ is predicted with displacement $\hat{\boldsymbol{D}}^{i-1}$.

- BF (Best-fit block-matching): This technique is widely used for the motion estimation because of its simplicity and coding efficiency for motion information. A detailed description of this technique is in Section 2.
- Neural fuzzy motion estimator (NFME): The authors propose this new technique introduced in Section 2 for image motion estimation. This system can learn proper system parameters for different circumstances and has been tuned using various training patterns obtained from affine motion equation (Eq. (8)). The rotation angles θ_r, 4°, 8°, -4°, -8° and the uniform translation components $(u_1,u_2)^T$, $(2,2)$, $(-2,2)$, $(2,-2)$, $(-2,-2)$ were used to find the proper value of σ as described in Section 2.3. From these training data,

the value $\sigma = 6.067$ was obtained as the parameter of input fuzzy membership functions. Although the synthetical images for comparisons were displaced uniformly $(-3, 2)$ and rotated 6° clockwise, which differed from training patterns, one will see the more accurate motion estimation resulted from this system in the following comparisons. The comparisons also show that the proposed system is more robust than the others.

3.3. Comparison criteria and results

Some criteria must be defined to compare the performance of different methods. The five criteria used here are the root-mean-square velocity difference in the x_1-direction ($err1$), the root-mean-square velocity difference in the x_2-direction ($err2$), the average angle difference (θ_{err}) of motion vector, the root-mean-square error of velocity ($RMSE$), and the average motion vector $(\bar{V}_{x_1}, \bar{V}_{x_2})$ in translation test. These five criteria are defined as follows:

$$err1 = \sqrt{\frac{1}{n_1 \cdot n_2} \sum \sum [D\_V_{x_1} - V_{x_1}]^2}, \tag{36}$$

$$err2 = \sqrt{\frac{1}{n_1 \cdot n_2} \sum \sum [D\_V_{x_2} - V_{x_2}]^2}, \tag{37}$$

$$\theta_{\text{err}} = \frac{1}{n_1 \cdot n_2} \sum \sum \left| \tan \frac{D\_V_{x_2}}{D\_V_{x_1}} - \tan \frac{V_{x_2}}{V_{x_1}} \right|, \tag{38}$$

$$RMSE = \sqrt{\frac{1}{n_1 \cdot n_2} \sum \sum \left[\sqrt{D\_V_{x_1}^2 + D\_V_{x_2}^2} - \sqrt{V_{x_1}^2 + V_{x_2}^2} \right]^2}, \tag{39}$$

$$(\bar{V}_{x_1}, \bar{V}_{x_2}) = \left(\frac{1}{n_1 \cdot n_2} \sum_i \sum_j V_{x_1}(i,j), \ \frac{1}{n_1 \cdot n_2} \sum_i \sum_j V_{x_2}(i,j) \right), \tag{40}$$

where $D\_V_{x_1}$ and $D\_V_{x_2}$ are desired motion vectors, V_{x_1} and V_{x_2} are actual motion vectors, and $n_1 \times n_2$ is the size of an image frame. Three types of performance representation are used: (i) motion vector flow map – represents motion vector at every location (x_1, x_2) by a small arrow whose length and direction are proportional to the motion vector's magnitude and angle; (ii) motion vector scatter map – represents motion vectors on the Cartesian coordinate plane; and (iii) error table – lists of all the errors in Eqs. (36)–(40) for comparison.

Fig. 9(a) and (b) show the flow map and scatter map, respectively, of the desired motion vectors when the image was uniformly displaced by -3 pixels in the vertical direction and 2 pixels in the horizontal direction. Fig. 9(c) is the flow map computed by the HS method, and Fig. 9(d) shows its scatter map. The motion estimation had $err1 = 0.2283$ pixels, $err2 = 0.7839$ pixels, $\theta_{\text{err}} = 20.095$ degrees, $RMSE = 0.7191$, and $(\bar{V}_{x_1}, \bar{V}_{x_2}) = (-1.43, 1.35)$ as shown in Table 3. Figs. 9(e) and (f) indicate the motion estimation computed from the FME method, which had $err1 = 0.3952$ pixels, $err2 = 0.6802$ pixels, $\theta_{\text{err}} = 12.448$ degrees, $RMSE = 0.7421$, and $(\bar{V}_{x_1}, \bar{V}_{x_2}) = (-2.51, 2.02)$. Figs. 10(g) and (h) are the flow map and scatter map representing the motion estimation of the NR recursive algorithm, which had $err1 = 0.5009$ pixels, $err2 = 0.3555$ pixels, $\theta_{\text{err}} = 6.258$ degrees, $RMSE = 0.6877$, and $(\bar{V}_{x_1}, \bar{V}_{x_2}) = (-2.91, 2.35)$. These errors are listed in Table 3. Figs. 10(i) and (j) show the motion estimation resulting from the BF method. The corresponding errors are listed in Table 3. Fig. 10(j) and Table 3 indicate that the motion estimation computed from the BF method was error-free in the translation test. This flawlessness is due to the fact that the signal made from an integer-translated image is clean (with no shadowing, specularity, transparency, occlusion, etc.); consequently, BF finds a motion

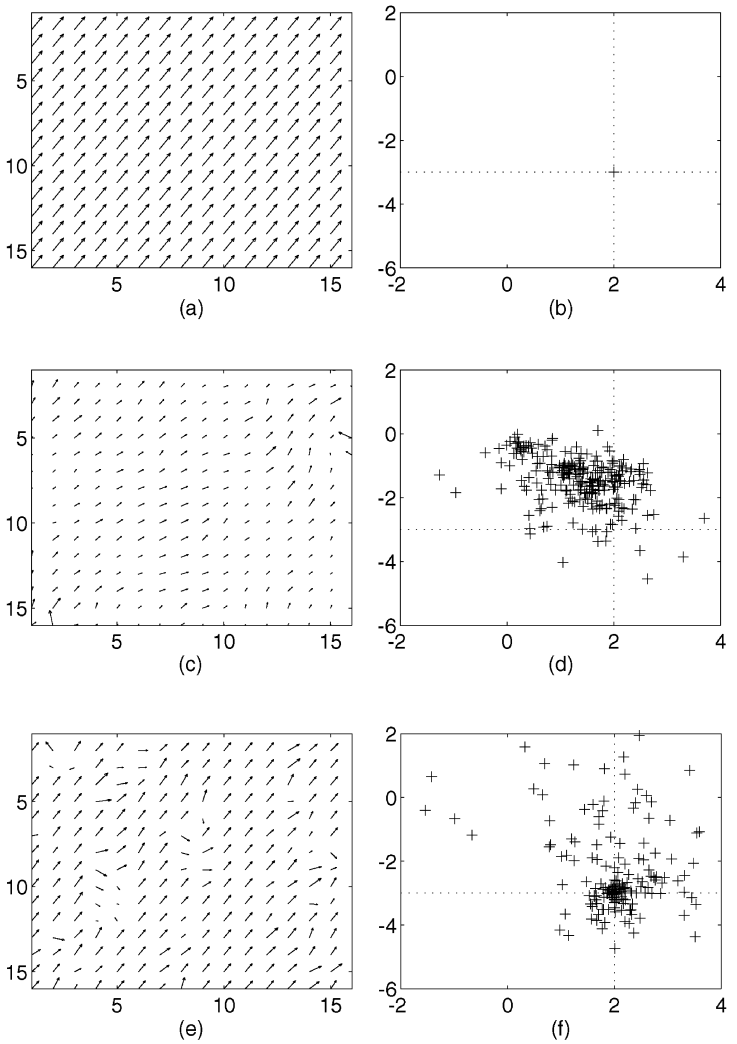

Fig. 9. Translation test. (a) Desired flow map. (b) Desired scatter map. (c) HS flow map. (d) HS scatter map. (e) FME flow map. (f) FME scatter map.

vector which is a perfect match in the search region from the present image. As will be see later, BF showed its shortcoming when it could not find a perfect match in the rotation test. Figs. 10(k) and (l) are the flow map and scatter map representing the motion estimation of the NFME method, which had $err1 = 0.0044$ pixels, $err2 = 0.0042$ pixels, $\theta_{\text{err}} = 0.065$ degrees, $RMSE = 0.0085$, and $(\bar{V}_{x_1}, \bar{V}_{x_2}) = (-3, 1.99)$. This result was very close to the desired one.

Fig. 11(a) is the flow map representing the desired motion vector when the image is rotated 6° clockwise. Figs. 11(b)–(f) are the estimated flow maps using various motion estimation methods. The corresponding error information is shown in Table 3. As observed from Table 3, the NFME method performs the best among all the compared methods. It was also found that the motion estimation obtained from the BF method was not error-free any more: this was because the perfect-match motion vectors did not always exist when pixels moved in the rotation style. From the comparison results shown as flow maps, scatter maps, and an

Table 3
Error table

	Translation $(-3,2)$					Rotation 6°			
	*err*1	*err*2	θ_{err}	*RMSE*	$(\bar{V}_x, \bar{V}_y)$	*err*1	*err*2	θ_{err}	*RMSE*
HS	0.2283	0.7839	20.095	0.7191	$(-1.43, 1.35)$	0.8136	1.0729	31.459	1.5315
FME	0.3592	0.6802	12.448	0.7421	$(-2.51, 2.02)$	0.8849	0.8789	34.703	1.0837
BF	0	0	0	0	(-3.2)	0.3251	0.3805	17.556	0.4155
NR	0.5009	0.3555	6.258	0.6877	$(-2.91, 2.35)$	2.7433	3.5037	63.68	1.9491
NFME	0.0044	0.0042	0.065	0.0085	$(-3, 1.99)$	0.2833	0.3152	12.648	0.3597
3-step	0.9648	0.7344	18.652	1.1971	$(-2.59, 1.85)$	0.8094	1.2068	33.657	1.2785
Modified 3-step	0.5926	0.5360	12.346	0.7539	$(-2.53, 1.48)$	0.6646	0.8984	27.100	0.9192

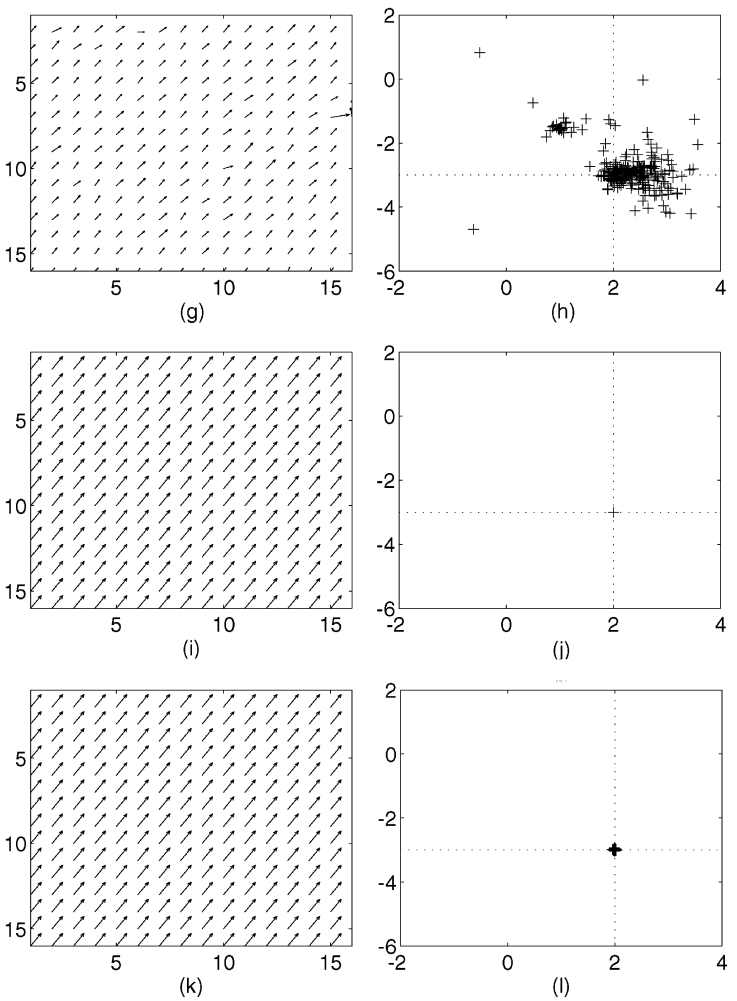

Fig. 10. Translation test (continued). (g) NR flow map. (h) NR scatter map. (i) BF flow map. (j) BF scatter map. (k) NFME flow map. (l) NFME scatter map.

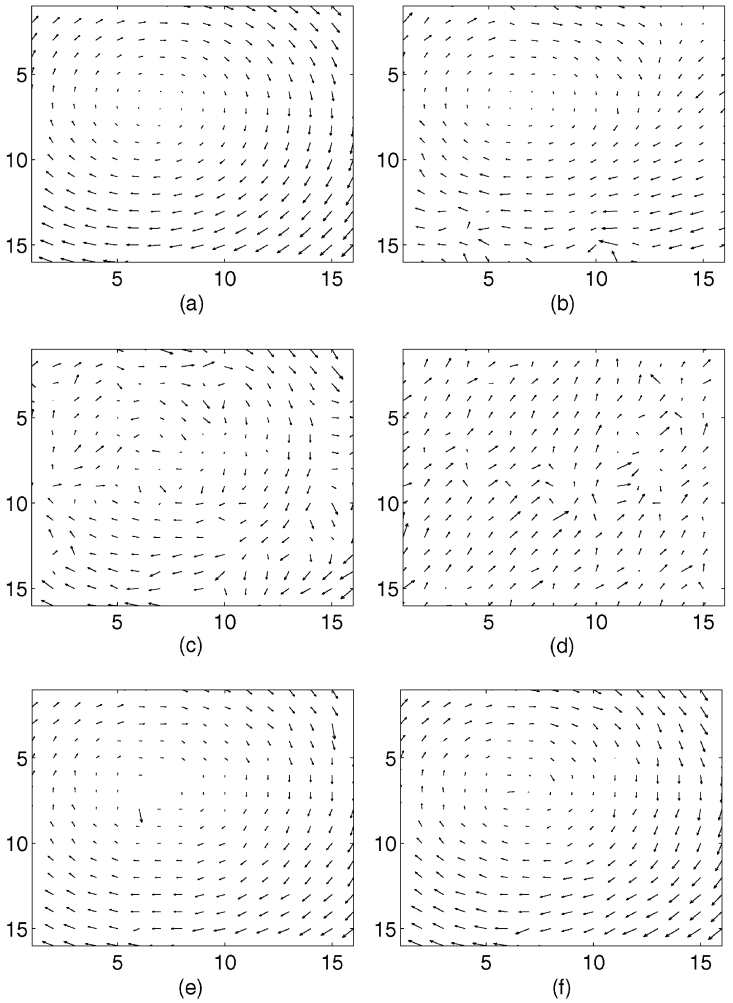

Fig. 11. Rotation test. (a) Desired flow map. (b) HS flow map. (c) FME flow map. (d) NR flow map. (e) BF flow map. (f) NFME flow map.

error table in Figs. 9–11, and Table 3, one can infer that the proposed NFME method is very robust and accurate in both the translation and rotation tests.

Performance comparisons of the three-step fast search and modified three-step fast search algorithms are shown in Table 3 and Figs. 12(a)–(f). A modified three-step algorithm could reduce the error from the local minimum. The number of search locations were 1/5 of those from the exhaustive search approach.

Affine motion with a large rotation angle, $\theta_r = 12°$, was also tested in this study. It is noted that this rotation angle is outside the range of the training data used for training the NFME system in Section 2. The parameter, reference block size 11×11, and search region 31×31 were used to examine the performance of various methods described in Section 3.2. The results are shown in Fig. 13 and Table 4. The same conclusion was still reached: the NFME system provides the most accurate estimation when the image is rotated with a large angle. The average time took by each motion estimation algorithm for computing the motion vectors of an image is listed in Table 5, where the programs were run on a PC486-66. The table shows that the proposed

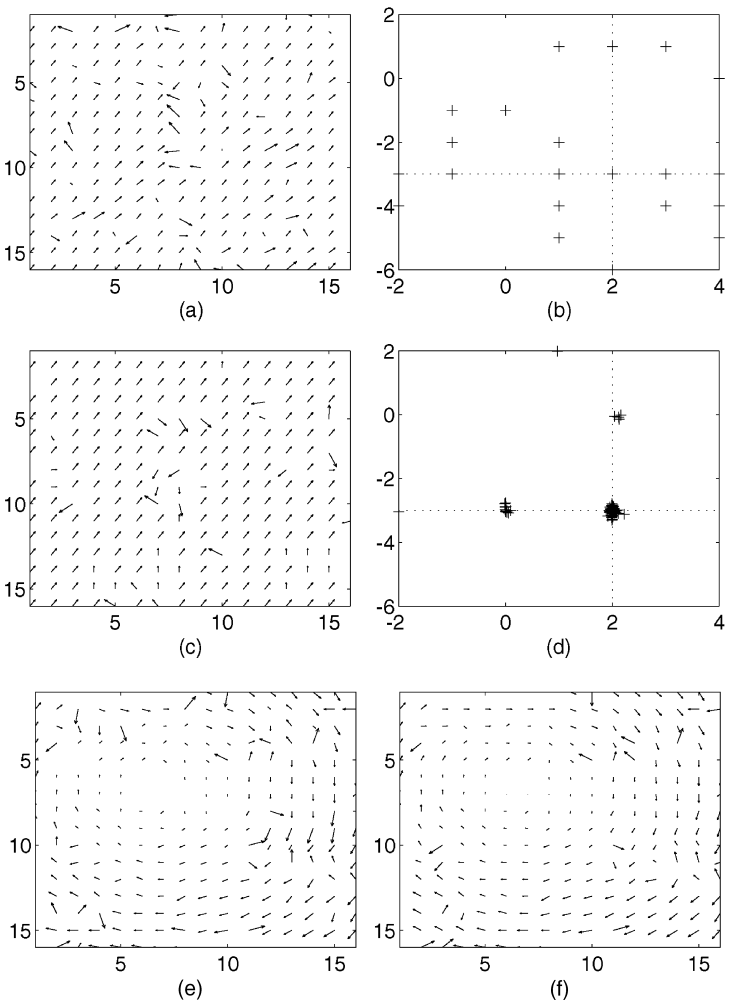

Fig. 12. Tests of fast search algorithms. (a) 3-step flow map. (b) 3-step scatter map. (c) Modified 3-step flow map. (d) Modified 3-step scatter map. (e) 3-step flow map (rotation test). (f) Modified 3-step flow map (rotation test).

NFME system takes the same time as FME and BF methods, longer time than HS and NR methods, but it reaches the highest accuracy.

4. Applications

4.1. Moving image compression

There are many methods used to compress data for transmission or storage of images. Frame skipping is one of the simplest methods of data compression for interframe motion images. For simplicity, suppose only the alternate frames are skipped. With no knowledge of the motion trajectory of the pixels, a skipped frame is generally reproduced either by repeating the preceding frame or by interpolation between the preceding and

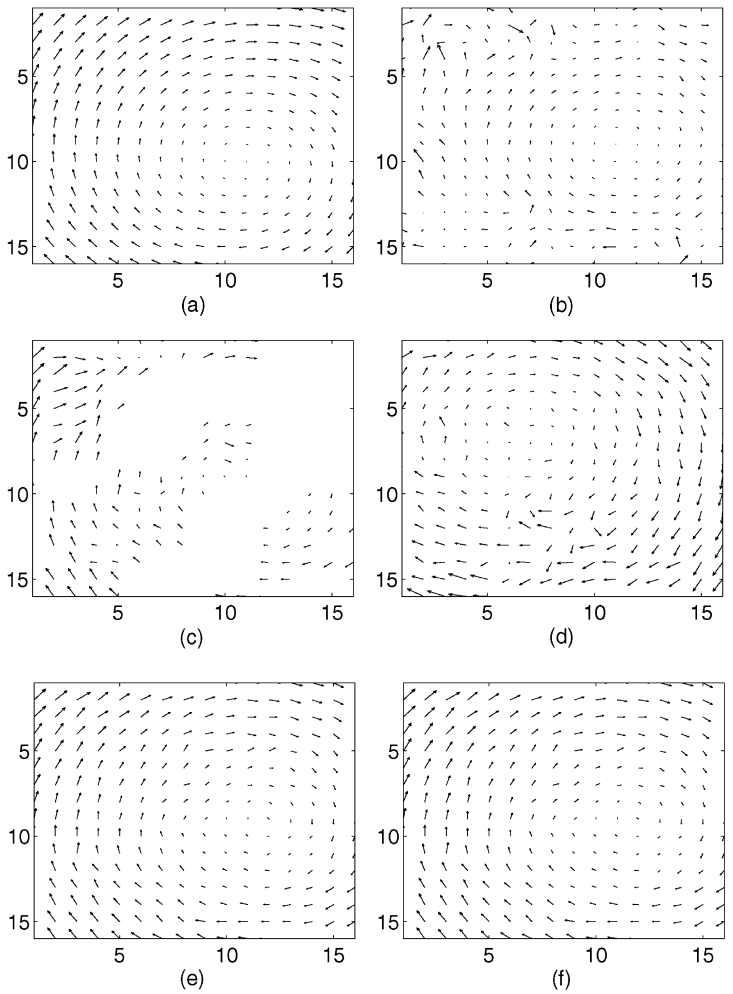

Fig. 13. Affine motion with large rotation angle. (a) Desired flow map. (b) HS flow map. (c) FME flow map. (d) NR flow map. (e) BF flow map. (f) NFME flow map.

the following frames. Both of these methods harm the quality of motion reproduction, however. The former results in jerkiness in the reproduction of the motion while the latter blurs the moving areas.

Let U_{2k} be the $(2k)$th frame where frames $2,4,\ldots,2k,\ldots$, have been skipped. Then U^*_{2k}, the reproduced value of U_{2k}, is obtained (without motion compensation) as follows:

Frame repetition:

$$U^*_{2k}(m,n) = U_{2k-1}(m,n). \tag{41}$$

Frame interpolation:

$$U^*_{2k}(m,n) = \tfrac{1}{2}\{U_{2k-1}(m,n) + U_{2k+1}(m,n)\}. \tag{42}$$

The disadvantages of frame repetition and interpolation can be overcome by predicting or interpolating the pixels of the skipped frame along its motion trajectory. Hence, with motion compensation, frame repetition

Table 4
The performance of various estimation methods on the affine motion with large rotation angle

	Translation $(-3,2)$		Rotation 12°	
	*err*1	*err*2	θ_{err}	*RMSE*
HS	2.0931	2.0834	38.616	3.1053
FME	2.0217	1.7948	33.401	3.2401
NR	1.9745	3.1130	49.448	2.6358
BF	0.7207	0.6481	18.618	0.6922
NFME	0.6606	0.6179	15.798	0.6231

Table 5
Average time took by each motion estimation algorithm for computing the motion vectors of an image

	HS	NR	FME	BF	NFME
Time (s)	15	25	75	75	75

and frame interpolation equations are replaced by

$$U^*_{2k}(m,n) = U_{2k-1}(m+q, n+l) \tag{43}$$

and

$$U^*_{2k}(m,n) = \tfrac{1}{2}\{U_{2k-1}(m+q, n+l) + U_{2k+1}(m+q', n+l')\}, \tag{44}$$

respectively, where (q,l) and (q',l') are the motion vector of U_{2k} relative to the preceding and the following frames, respectively. It is noted that q,l,q',l' in Eqs. (43) and (44) are all integer numbers in the image coordinates. How does one tackle such a problem if q,l,q',l' are not integer numbers as those produced by the proposed NFME method?

Coding and decoding systems (codecs) that use motion-compensation with fractional-pel accuracy have been reported in [7,12–15,20,24]. Typically, fractional-pel accuracy is achieved by simple bilinear interpolation which produces a spatially blurred prediction signal. Improvement gained in this way is referred to as the "filtering effect". Sinc-interpolation, bilinear interpolation, and Wiener filtering were compared at integer-pel, 1/2-pel, 1/4-pel, and 1/8-pel accuracies. Remarkably, for the neural fuzzy motion estimator (NFME), the motion vector's accuracy was at infinitesimal-subpixel. Hence, we propose a compensation method called the quarter compensation algorithm (QCA), an infinitesimal-subpixel compensation method, to compensate for inter-frame of images according to NFME's motion estimation.

Suppose the origin point of an image is in the left-top corner and the positive directions are rightward and downward. Assume the motion vector $v = (v\text{-}row, v\text{-}col)$ in some location (m,n) has been evaluated, and both $v\text{-}row$ and $v\text{-}col$ are floating numbers. Without loss of generality, it is assumed that both $v\text{-}row$ and $v\text{-}col$ are positive; i.e., the point in location (m,n) moves toward the lower-right corner. One distinguishes the integer part and decimal part of the motion vector as follows:

$$v\text{-}row = q + d\text{-}row, \tag{45}$$

$$v\text{-}col = l + d\text{-}col, \tag{46}$$

where (q,l) and $(d\text{-}row, d\text{-}col)$ are the integer and decimal part of $(v\text{-}row, v\text{-}col)$, respectively. The square which was in location (m,n) originally is now situated on the four locations $(m+q, n+l)$, $(m+q+1, n+l)$, $(m+q, n+l+1)$, and $(m+q+1, n+l+1)$. A moved square is thus divided into four portions, with the area of each portion decided by the decimal part, $(d_{\text{row}}, d_{\text{col}})$, of the motion vector v. The area of each portion is viewed as a weighting factor in deciding new gray values. In other words, the gray value of the pixel, which was originally in location (m,n), is distributed into the new gray values of the four locations (pixels) according to these weights.

Hence, for any location (x_1, x_2) in the compensated image frame, its gray value $U^*_{2k}(x_1, x_2)$ is decided by the following equation:

$$U^*_{2k}(x_1, x_2) = \frac{\sum_{(m,n)\in \boldsymbol{D}} w_{(m,n)\to(x_1,x_2)} \cdot U_{2k-1}(m, n)}{\sum_{(m,n)\in \boldsymbol{D}} w_{(m,n)\to(x_1,x_2)}}, \tag{47}$$

where (x_1, x_2) is the image coordinate, $D \triangleq \{(m,n) \mid$ one of the four portions of the moved points $U_{2k-1}(m,n)$ is located in the position $(x_1, x_2)\}$, $w_{(m,n)\to(x_1,x_2)}$ is the area (weighting factor) of the portion of $U_{2k-1}(m,n)$ that falls in the location (x_1, x_2), $\sum_{\boldsymbol{D}} w_{(m,n)\to(x_1,x_2)}$ is the normalization factor.

For any location (x_1, x_2) in the compensated image frame, its gray value can be recovered by Eq. (47). The numerator denotes the sum of product of the weighting factors and the gray values of the moved points which fall in the location (x_1, x_2), and the denominator is the sum of all weighting factors, which is used as the normalization factor in the recovering process.

Experiment. Two real-image frames containing moving objects were tested using the quarter compensation algorithm (QCA) for interframe motion compensation. The result is compared with that of the integer compensation method described by Eq. (43), and the performance is measured by *normalized mean square error* (*NMSE*):

$$NMSE = \frac{1}{N_1 \times N_2} \sum_{n_1} \sum_{n_2} [U_{2k}(n_1, n_2) - U^*_{2k}(n_1, n_2)]^2, \tag{48}$$

where $U_{2k}(n_1, n_2)$ is the original skipped image, $U^*_{2k}(n_1, n_2)$ the compensated image from $U_{2k-1}(n_1, n_2)$, and $N_1 \times N_2$ the dimension of image frame.

The results are shown in Fig. 14. By integrating NFME with the QCA to compensate for the interframe of image, better results than using the integer-compensated method can be found as observed from Fig. 14. Without motion compensation (i.e., the interframe is skipped), the error (*NMSE*) is up to 22.514. This error is reduced to 10.658 using the integer compensation method, to 7.342 using the bilinear interpolation scheme, and to 2.764 using the proposed QCA.

4.2. Multiple moving object extraction

Letting the computer find a specific item among many uncertain moving objects is a difficult task. Human beings can easily distinguish different moving objects and grab them even when the light is dim or the object's outline is vague. However, a computer must have more useful information, such as motion vector, to catch the moving objects and discard the redundancies in an image frame. It is possible that there is more than one moving object in an image frame, but only one is sought. How to extract it then becomes an important task in order to interpret or recognize the object. Obviously, the reasonable motion information can help to extract a moving object from image sequences. More accurate motion vectors will provide more useful information about the moving objects. Hence, the proposed NFME can be used to compute motion vectors and extract moving object because the NFME can provide accurate dynamic information. An experiment on multiple moving object extraction in this application was conducted.

Two image sequences with two moving objects contained in them are shown in Fig. 15. The motion estimation computed by the NFME system is shown in Fig. 16, where one can see clearly the two moving objects. As observed from this diagram, two different regions corresponding to the two different moving objects can be distinguished and extracted based on the combined information of moving angles and velocities transformed from the estimated motion vectors. Hence, a moving object toward the North and with a velocity of about 2 pixels/frame was extracted as shown in Fig. 15(d).

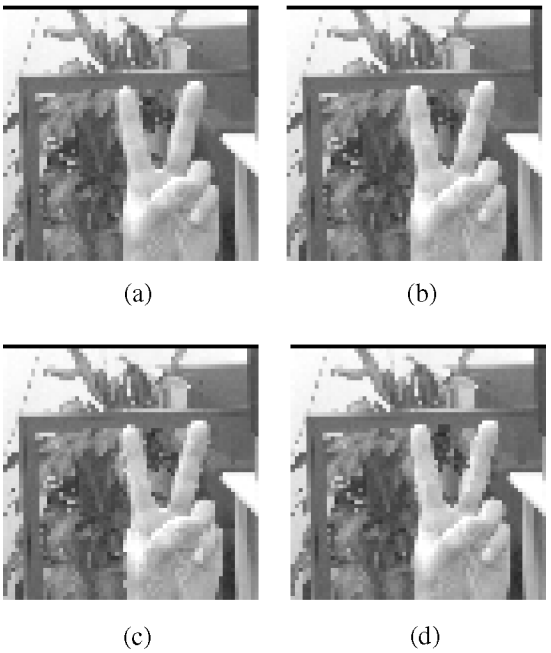

Fig. 14. Motion estimation and compensation. (a) The previous image. (b) The present image. (c) The compensated image using the integer compensation method. (d) The compensated image using the QCA.

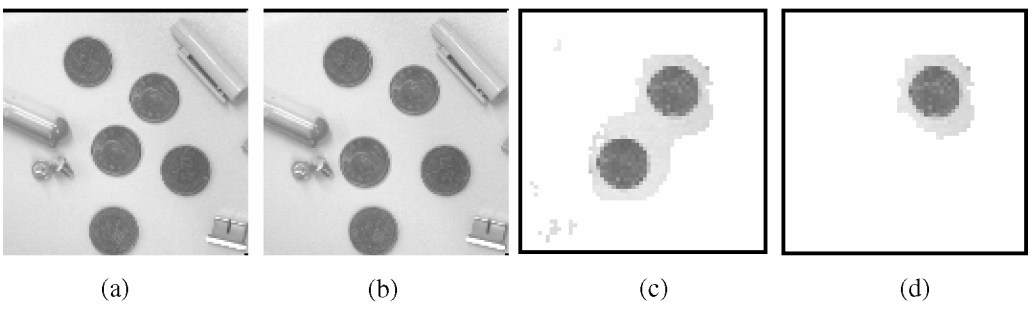

Fig. 15. The images for multiple moving objects extraction. (a) The previous image. (b) The present image. (c) The moving objects. (d) The extracted moving object.

5. Conclusions

The neural fuzzy motion estimator presented in this paper has been shown to provide accurate motion vector estimates for uniform and affine modeled object's motion. Based on block matching, each subblock in the search region is assigned a similarity membership contributing different degree to the estimated motion vector in the neural fuzzy motion estimator. This system is more reliable and robust in motion estimation than other methods such as Horn and Schunck's optical flow, fuzzy logic motion estimator (FME), NR, fast block-matching, etc. Motion estimation and compensation are an integral part for video compression since it enables the removal of naturally existing temporal redundancies. In this paper a neural motion compensation

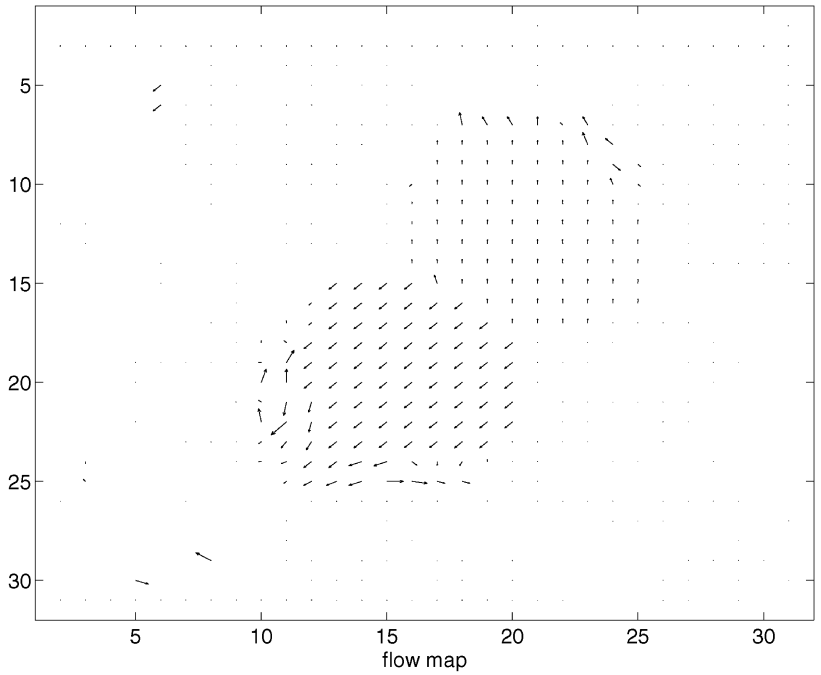

Fig. 16. The flow map of multiple moving objects.

method, the QCA, is proposed. The QCA can perform infinitesimal-subpixel interframe motion compensation. This algorithm provides better results than the conventional methods based on integer motion vectors for the interframe motion compensation. The proposed neural fuzzy motion estimator can be applied in various dynamic image-related applications where motion information is concerned. The proposed system is especially effective for extracting multiple moving objects. In this application, useful motion-related features were extracted from the estimated motion vectors. This is an important technique that can be processed at the earlier stages of vision, prior to higher level tasks such as segmentation, recognition, or interpretation. Motion information has become a basic knowledge for image understanding, image compression, vision-based control, etc. The accurate motion field estimated by the proposed system can provide useful motion information for making better decisions.

References

[1] P.K. Allen, A. Timcenko, B. Yoshimi, P. Michelman, Automated tracking and grasping of a moving object with a robotic hand-eye system, IEEE Trans. Robotics Automat. 9 (1993) 152–165.
[2] R. Baker, Waveform based very low rate video coding, Keynote address presented at the Internat. Note Workshop Very Low Bit Rate Video Compression, Urbana, IL, 1993.
[3] J.L. Barron, D.J. Fleet, S.S. Beauchemn, Systems and experiment performance of optical flow techniques, Internat. J. Comput. Vision 12 (1) (1994) 43–77.
[4] G.C. Buttazzo, B. Allotta, P. Fanizza, Mousebuster: a robot for real-time catching, IEEE Control Systems 14 (1) (1994) 49–56.
[5] K.W. Chun, J.B. Ra, An improved block matching algorithm based on successive refinement of motion vector candidates, Signal Processing: Image Com. 2 (1994) 115–122.
[6] J.H. Duncan, T.C. Chou, Temporal edges: the detection of motion and the computation of optical flow, Proc. 2nd Internat. Conf. Comput. Vis., Tampa, 1988, pp. 374–382.
[7] S. Ericsson, Fixed and adaptive predictors for hybrid predictive/transform coding, IEEE Trans. Commun. COM-33 (1985) 1291–1302.

[8] C. Fennema, W. Thompson, Velocity determination in scenes containing several moving objects, Comput. Graph. Image Process. 9 (1979) 301–315.
[9] D.J. Fleet, Measurement of Image Velocity, Kluwer Academic Publishers, Norwell, MA, 1992.
[10] H. Gharavi, M. Mills, Blockmatching motion estimation algorithm – new results, IEEE Trans. Circuits Systems 37 (1990) 649–651.
[11] B. Girod, D.J. Le Gall et al., Guest editorial: introduction to the special issue on image sequence compression, IEEE Trans. Image Process. 3 (1994) 465–468.
[12] B. Girod, F. Joubert, Motion-compensating prediction with fractional pel accuracy for 64 kbits/s coding of moving videl, Proc. Internat. Workshop 64kbit/s Coding Moving Video, Hannover, Germany, 1988.
[13] B. Girod, The deficiency of motion-compensating prediction for hybrid coding of video sequences, IEEE J. Select. Areas Commun. SAC-5 (1987) 1140–1154.
[14] B. Girod, Motion-compensating prediction with fractional-pel accuracy, IEEE Trans. Commun. 41 (1993) 604–612.
[15] M. Glige, A high quality videophone coder using hierachical motion estimation and structure coding of the prediction error, Proc. SPIE Conf. Visual Commun. Image Proc. '88 1001, Cambridge, MA, SPIE, 1988, pp. 864–874.
[16] B.K.P. Horn, B.G. Schunck, Determining optical flow, Artificial Intell. 17 (1981) 185–204.
[17] J. Hwang, Y. Ooi, S. Ozawa, An adaptive sensing system with tracking and zooming a moving object, IEICE Trans. Inform. Systems E76-D (1993) 926–934.
[18] J.R. Jain, A.K. Jain, Displacement measurement and its applications in interfame coding, IEEE Trans. Commun. COM-29 (1981) 1799–1808.
[19] H.M. Jong, L.G. Chen, T.D. Chiueh, Parallel architectures for 3-step hierachical search blockmatching algorithm, IEEE Trans. Circuits Systems Video Technology 4 (1994) 407–416.
[20] C.D. Kuglin, D.C. Hines, The phase correlation image alignment method, Proc. IEEE 1975 Internat. Conf. Cybernet. Soc. (1975) 163–165.
[21] C.T. Lin, C.S. Gorge Lee, Neural-network-based fuzzy logic control and decision system, IEEE Trans. Comput. 40 (1991) 1320–1366.
[22] H. Li, A. Lundmark, R. Forchheimer, Image sequence coding at very low bitrates: a review, IEEE Trans. Image Process. 3 (5) (1994) 589–609.
[23] J.I. Lipp, Frame-to-frame image motion estimation with a fuzzy logic system, Proc. Circ. and Syst. Conf., 1992, pp. 987–990.
[24] F. May, Model based movement compensation and interpolation for ISDN videotelephony, IEEE Int. Symp. Circuits Syst. (ISCAS 88) Espoo, Finland, 1988.
[25] A. Murat Tekalp, Digital Video Processing, Prentice-Hall, Englewood Cliffs, NJ, 1995.
[26] H.H. Nagel, Displacement vectors derived from second-order intensity variations in image sequences, Comp. Graph Image Process. 21 (1983) 85–117.
[27] H.H. Nagel, On the estimation of optical flow: relations between different approaches and some new results, Artificial Intell. 33 (1987) 299–324.
[28] A.N. Netravali, J.D. Robbins, Motion compensated television: Part I, Bell System Tech. J. 58 (1979) 631–670.
[29] R. Plompen, Motion video coding for visual telephony, PTT Research Neher Laboratories, 1989.
[30] O. Tretiak, L. Pastor, Velocity estimation from image sequences with second-order differential operators, Proc. 7th Internat. Conf. Patt. Recog., Montreal, 1984, pp. 20–22.
[31] S. Uras, F. Girosi, A. Verri, V. Torre, A computational approach to motion perception, Biol. Cybernet. 60 (1988) 79–97.