Spatial Vision, Vol. 10, No. 1, pp. 31–50 (1996)
© VSP 1996.

Displacement field estimation and image segmentation using block matching enhanced by a neural network

YONG-SHENG CHEN and CHIOU-SHANN FUH*
Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan

Received 10 August 1995; accepted 30 October 1995

Abstract—The block-matching method plays an important role in displacement field estimation due to its simplicity, achievement of long-range motion, and robustness to noise. In this paper, a single-layer feedback neural network model is proposed that enhances block matching, estimates the displacement field, and simultaneously performs image segmentation from consecutive images. In this paper, image segmentation is defined as partitioning each image into a set of moving objects and the background. For any two consecutive images, a neural network is created that learns the connection relationship of the pixels in an object from the displacement field and stores the relationship in the network. A modified block matching is used to compute a more accurate displacement field by utilizing the segmentation information embedded in the neural network. The displacement vector at the edge of an object or occluding boundary is hard to estimate, but the proposed model performs satisfactorily because it learns and uses the connection information. Furthermore, a flood-fill algorithm is used to compute the dense displacement field more efficiently and correctly than the exhaustive search does. The most important aspect of this paper is that image segmentation is performed simultaneously with the displacement-field estimation by the neural-network model. The novel idea of the work is to embed the segmentation information (connection relations) in the neural network and to perform the displacement-field estimation and image segmentation simultaneously. Two methods for retrieving segmentation information from the neural network with any two consecutive images are also presented.

1. INTRODUCTION

1.1. Motivation

The abundance of attributes in information of time-varying images allows visual motion analysis to provide surface structure, 3D-translation and rotation parameters, and other useful information. With this ability, visual motion analysis can be applied to target tracking, video coding, passive navigation, automatic surveillance, remote

*To whom correspondence should be addressed. E-mail: fuh@csie.ntu.edu.tw sensing, and many other real-life applications. There are four major tasks in motion analysis as Fig. 1 shows. The first one is to estimate the optical flow field, which is the apparent motion of brightness patterns observed as the sensor moves relative to the objects being imaged. Second, we can estimate the motion parameters and surface structure from the optical flow field by writing a set of linear equations in the unknown parameters of the motion and surface structure. We can also divide the image into parts with coherent optical flow in order to extract object shapes by finding the rapidly varying part of the optical flow field. The optical flow field is thus used as an intermediary for recovery of successive motion and surface structure and for segmentation. The accuracy of the optical flow field hence determines the overall accuracy of the motion analysis.

There is, however, a dilemma concerning optical flow-field estimation and segmentation. If we can first divide the images well, we can extract the objects separately to reduce the complexity of the images and produce a better optical flow field. On the other hand, if we have a good optical flow field, we can locate the boundary of the segmentation where optical flow varies rapidly. The way to resolve this dilemma is to combine the estimation of the optical flow field and segmentation in an iterative manner as Fig. 2 shows; that is, segmenting the image from the previously estimated optical flow field and estimating the optical flow field according to the previous segmentation information.

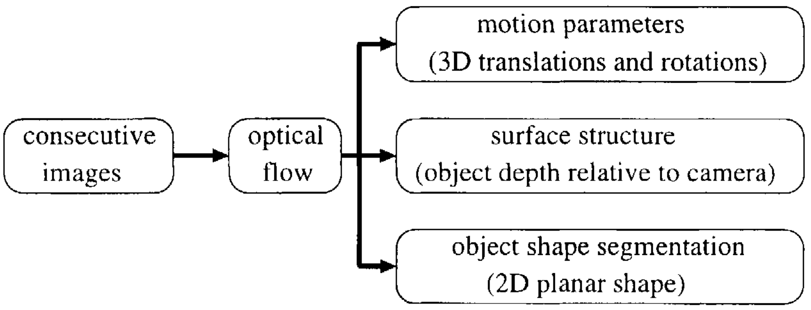

consecutive images
optical flow
motion parameters (3D translations and rotations)
surface structure (object depth relative to camera)
object shape segmentation (2D planar shape)

Figure 1. The major tasks of motion analysis.

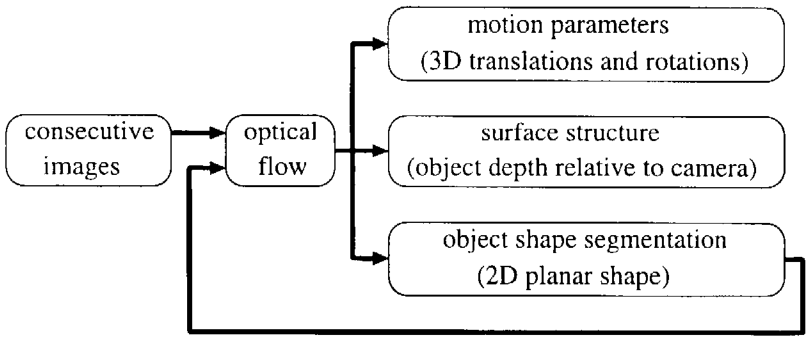

consecutive images
optical flow
motion parameters (3D translations and rotations)
surface structure (object depth relative to camera)
object shape segmentation (2D planar shape)

Figure 2. The iterative steps of motion analysis.

In this paper, we will concentrate on an iterative method for estimating the displacement field, that is, the optical flow field after being divided by the interframe time and latting the interframe time tend to zero, and obtaining image segmentation for regions in which all pixel displacement vectors are coherent. We create a single-layer feedback neural-network model that obtains information about segmentation from the displacement field using an enhanced block-matching method. This block matching in turn uses information about segmentation embedded in the neural network to enhance the accuracy of the displacement field.

Many methods have been proposed estimating the displacement field and performing image segmentation from the motion field. We briefly survey these methods in Section 1.2.

1.2. Background

There are three major groups of methods estimating the optical flow field or displacement field. These are the gradient methods, correspondence methods, and block-matching methods. Each group of methods has its own advantages, disadvantages, and limitations in different situations.

1.2.1. Gradient methods. Gradient methods depend on certain assumptions and constraints, such as the surface shape, the existence of rigid-body motion, constant intensity of corresponding pixels between consecutive images, and smoothness of optical flow over neighboring pixels.

Horn and Schunck (1981) define optical flow to be the apparent motion of brightness patterns. They assume that the intensity $I(x, y, t)$ of pixel (x, y) at time t will remain the same at time $t + \delta t$ at pixel $(x + \delta x, y + \delta y)$. That is,

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t). \tag{1}$$

After expanding Eqn (1) into a Taylor series and discarding the second- and higher-order terms, we can get the optical flow constraint equation:

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0, \tag{2}$$

where (u, v) is the optical flow of pixel (x, y) at time t.

We cannot solve for these two variables (u, v) uniquely from just this equation because there is only local information derived by each pixel. Schunck and Horn (1981) introduce a smoothness constraint containing global information which assumes neighboring pixels of the same object have similar optical flow: they minimize the change (magnitude of the derivative) of the optical flow along both the X and Y axes:

$$\iint \left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2\right] \mathrm{d}x\mathrm{d}y. \tag{3}$$

We can thus use the discrete first-order derivative to compute $\partial I/\partial x$, $\partial I/\partial y$, and $\partial I/\partial t$ of each pixel and iteratively over-estimate (u, v) at each pixel with these two constraint equations.

In general, gradient methods are effective and they can compute the dense optical flow field at subpixel accuracy. But they require derivatives, are sensitive to noise, and have difficulty with long-range optical flow. The smoothness constraint is not valid at the occluding boundary, so the optical flow field is blurred near the occluding boundary.

1.2.2. Correspondence methods. Correspondence methods extract the interesting parts of consecutive images to match and track them according to similarity, defined by the displacement field. The interesting part of an image depends on the application, and may be an isolated point, edge, corner, peak, valley, blob, or suchlike.

Using pixels at such interesting parts makes it hard to solve the correspondence problem, so Fuh and Maragos (1989, 1990, 1991c) used a region-based correspondence method. Four kinds or regions were extracted, matched, and tracked among consecutive images to estimate the displacement field via the criterion of centroid distance, region identity, area difference, and intensity difference.

Although this method can only compute a sparse displacement field, it can achieve long-range displacement while eliminating the disturbance due to noise, so this method is useful in target tracking and can be applied to the correspondence problem in stereo images. But it is difficult to solve the correspondence problem and there is another consistency problem: the extracted interesting parts may have inconsistencies over the consecutive images.

1.2.3. Block-matching methods. The simplest block-matching method is to find in two images the corresponding pair of pixels and their surrounding blocks that minimize the sum of squares of intensity differences. This method is feasible if there are only translations but no rotations. Fuh and Maragos (1991a, b) developed a 2-D affine model to estimate the displacement field allowing affine intensity transformations and shape deformations. Dufaux and Kunt (1992) used a multi-grid block matching refined by an adaptive local mesh to obtain a more accurate motion field efficiently. Seferidis and Chanbari (1992) proposed a generalized block-matching method allowing affine, perspective, and bilinear transformations.

Although block-based methods are simple and easy to implement, there are two drawbacks that diminish the accuracy of estimation. First, there may be some regions appearing or disappearing between the two corresponding blocks of the two consecutive images, as in Fig. 3. The block-matching error using the sum of the squares of the intensity differences will depend on the intensity differences between the appearing and disappearing regions. Second, if there are several similar regions appearing in one image, there may be several blocks in the second image similar to some blocks in the first image. Using an inefficient and exhaustive block matching may lead to incorrect displacement vectors.

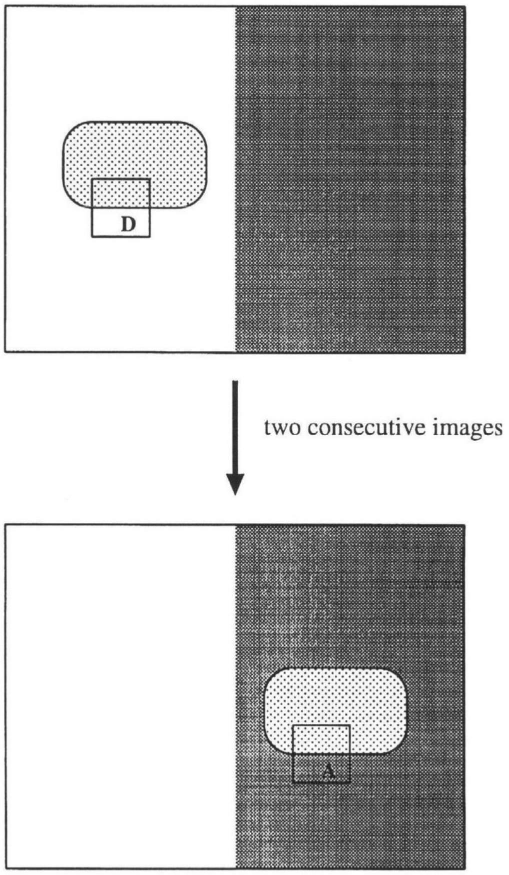


Figure 3. Region D is the disappearing region and Region A is the appearing region in these two corresponding blocks. The intensities of Region D and A are different.

These two drawbacks reveal themselves in the choice of the size of the matching block. If we choose a small block size, the result is more accurate near the boundary of the object, due to the small size of the appearing and disappearing regions, as in Fig. 3. But small block size increases the probability of ambiguity and reduces the total accuracy. Although large block size may reduce the ambiguity, it causes inaccuracy near the boundary.

Block matching can produce a long-range, dense displacement field and it is robust to noise. But it cannot estimate displacement vectors at the subpixel level, and it has difficulty in accurately estimating displacement vectors near the occluding boundary. Choosing the size of matching block is also a problem.

1.2.4. Image segmentation. Image segmentation is an important preliminary step in pattern recognition and scene analysis. Segmentation is based on similarity and discontinuity. In this section, we survey some methods for segmenting images into regions in which pixels have coherent and similar optical flow. The discontinuity in the optical flow field will define the boundary between two different neighboring regions.

François and Bouthemy (1990) used a likelihood test to divide the motion field into areas containing coherent motion. Thompson *et al.* (1985) used an edge-detection algorithm to locate the discontinuities in the optical flow, that is, the occluding boundaries. Irani *et al.* (1994) used temporal integration to divide and track the moving objects. Pal and Pal (1993) have reviewed some other segmentation techniques based on gray level, color, and range images.

Using optical flow field as an intermediary to divide images will inherit the noise and error of the optical flow field although this method modularizes and simplifies the problem of motion analysis.

1.2.5. Single-layer feedback neural network. In the past four decades, neural networks have attracted the attention of engineers and scientists from a variety of disciplines. Among the varieties of neural networks, a discrete-time, recursive, and single-layer feedback network can be treated as a recurrent network; that is, it processes the initial-condition information and moves through a sequence of states over time in a synchronous or asynchronous manner. This kind of neural network can provide associations or classifications, optimal solutions, restoration of patterns, mapping functions, and model dynamic systems. There is, however, a drawback inherent in modelling dynamic systems; it is difficult to explain how the solution is derived and to trace it back to its causes in view of the many random factors contributing to its evolution.

1.3. Computational model

In this paper, we put forward a single-layer feedback neural network model that estimates the displacement field and simultaneously performs image segmentation for each pair of consecutive images. The novel idea of the work is to embed the segmentation information (as a connection relation) in the neural network and to perform the displacement-field estimation and image segmentation simultaneously. Recall that the dilemma of motion analysis is whether we should first get a good estimate of the displacement field and then use it to divide an image, or first divide an image and then estimate the displacement field. We solve the problem by using this neural-network model for both estimation of the displacement field and segmentation. Every time we compute a displacement vector of some pixel, we modify the segmentation result in the neural network. According to the most recent segmentation result, we divide the image and compute the next displacement vector of another pixel with correspondingly less influence from occluding boundary.

In the following sections, we introduce the neural model, the method of manipulation of the synapses between neighboring neurons, and how the modified error function for block matching is made robust at the occluding boundary. Then, a flood-fill algorithm is used instead of exhaustive search to search for the matching block. The method of simultaneous segmentation is also presented. Some experimental results with real-world image sequences are also given.

2. DISPLACEMENT-FIELD ESTIMATION

2.1. Neural model

In this section, we create a single-layer feedback neural network to learn and record the segmentation information in each image. This neural network contains one neuron for each pixel in the image. Using the error function described in Section 2.2, each neuron contains information about the displacement vector, from which the block-matching error is computed. Each neuron also has eight synapses connected to the eight neighboring neurons. The construction of the neurons and synapses is shown in Fig. 4. The eight synapses of each neuron are ordered from 0 to 7 as the figure shows.

The weight of synapse stands for the strength of the connection, so if two neighboring pixels belong to the same object, the weight of the synapse between the two corresponding neurons should be large. As Fig. 5 shows, if the matching error $\varepsilon(r, c)$ of neuron $\mathrm{N}(r, c)$ using displacement vector $\mathbf{d}(r, c)$ is calculated, the feedback connection will adjust the weights of the synapses connecting neuron $\mathrm{N}(r, c)$ and its eight neighboring neurons to reveal the new state of the connection strengths.

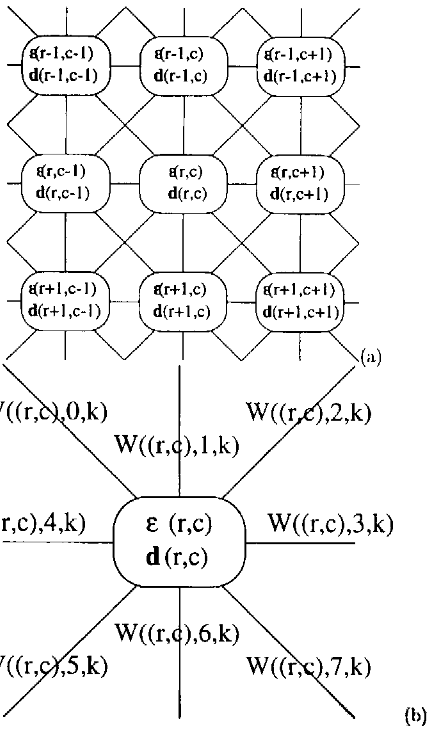


Figure 4. (a) The eight synapses of the neuron $\mathrm{N}(r, c)$ with the number on the synapses at the kth iteration. This order of synapses is just for the convenience of implementation. (b) The neuron $\mathrm{N}(r, c)$ contains the information of block matching error $\varepsilon(r, c)$, calculated by using the displacement vector $\mathbf{d}(r, c)$ and eight synapses connected to its neighbors.

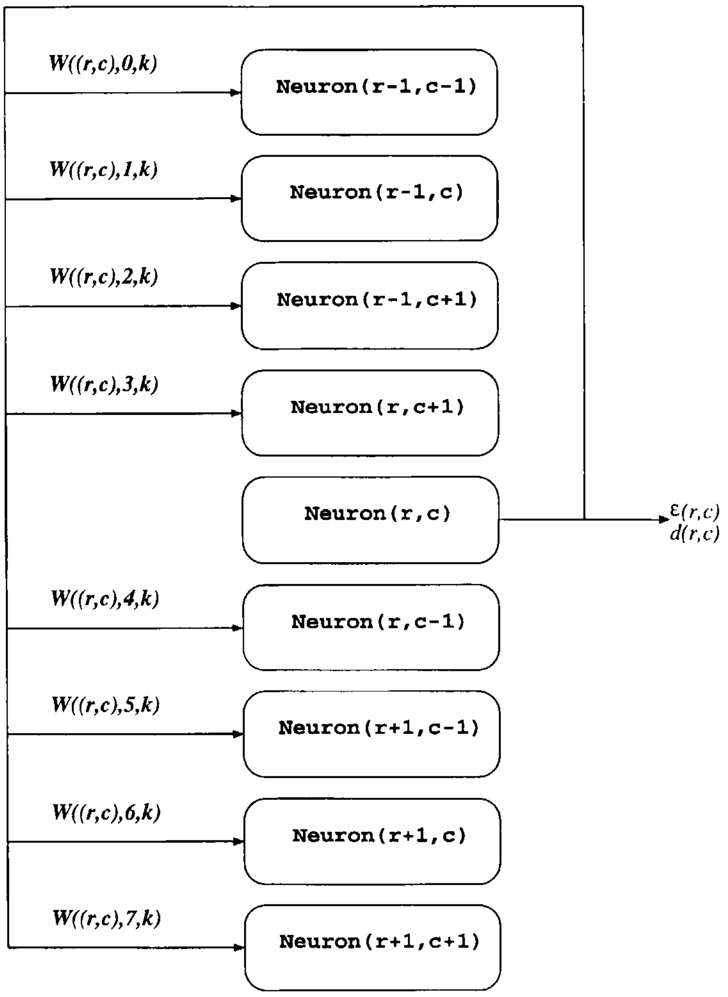


Figure 5. Single-layer feedback neural network model for the feedback connection of Neuron(r, c) only.

According to the smoothness assumption of Horn and Schunk (1981), the displacement field is continuous almost everywhere, except at the occluding boundaries of objects in the images. In other words, neighboring pixels of the same object have similar displacements but those pixels on opposite sides of the occluding boundary have different displacements. Thus we can expect that if $\mathbf{p}_1$, $\mathbf{p}_2$ are two pixels on opposite sides of an occluding boundary, then displacement vectors $\mathbf{d}(\mathbf{p}_1)$, $\mathbf{d}(\mathbf{p}_2)$ should be different and the matching error $\varepsilon(\mathbf{p}_1, \mathbf{d}(\mathbf{p}_2))$ of neuron $\mathrm{N}(\mathbf{p}_1)$ using the displacement vector $\mathbf{d}(\mathbf{p}_2)$ and matching error $\varepsilon(\mathbf{p}_2, \mathbf{d}(\mathbf{p}_1))$ of neuron $\mathrm{N}(\mathbf{p}_2)$ using the displacement vector $\mathbf{d}(\mathbf{p}_1)$ are both larger than $\varepsilon(\mathbf{p}_1, \mathbf{d}(\mathbf{p}_1))$ and $\varepsilon(\mathbf{p}_2, \mathbf{d}(\mathbf{p}_2))$.

The relationship presented above can be transformed into the weight of the synapse. According to the assumption above, if the matching error $\varepsilon(\mathbf{p}_1, \mathbf{d}(\mathbf{p}_2))$ is large, then $\mathbf{p}_1$, $\mathbf{p}_2$ should belong to different objects and the weight of the synapse between neurons $N(\mathbf{p}_1)$ and $N(\mathbf{p}_2)$ should be small. If, on the contrary, $\varepsilon(\mathbf{p}_1, \mathbf{d}(\mathbf{p}_2))$ is small, and $\mathbf{d}(\mathbf{p}_1)$ and $\mathbf{d}(\mathbf{p}_2)$ are coherent and similar, then $\mathbf{p}_1$, $\mathbf{p}_2$ should belong to the same object and the weight of synapse between neurons $N(\mathbf{p}_1)$ and $N(\mathbf{p}_2)$ should be large. The ith goal weight $W_g(\mathbf{p}_1, i, k)$ of neuron $N(\mathbf{p}_1)$ at the kth iteration is defined as the reciprocal of exponential error $\varepsilon(\mathbf{p}_1, \mathbf{d}(\mathbf{p}_2))$ scaled by λ:

$$W_g(\mathbf{p}_1, i, k) = e^{-\lambda\varepsilon(\mathbf{p}_1, \mathbf{d}(\text{neighbor}_i(\mathbf{p}_1)))}, \tag{4}$$

where $\mathbf{p}_2 = \text{neighbor}_i(\mathbf{p}_1)$ is the ith neighboring neuron of $\mathbf{p}_1$, $0 \leqslant i \leqslant 7$, as Fig. 4 shows, and the scaling factor λ depends on whether $\mathbf{d}(\mathbf{p}_1)$ is similar to $\mathbf{d}(\mathbf{p}_2)$ or not. Weight W_g will increase to 1 if $\varepsilon(\mathbf{p}_1, \mathbf{d}(\mathbf{p}_2))$ approaches 0 signifying that $N(\mathbf{p}_1)$ and $N(\mathbf{p}_2)$ belong to the same object, since $\mathbf{d}(\mathbf{p}_1)$ is similar to $\mathbf{d}(\mathbf{p}_2)$. In this case, a smaller value of λ is used to speed up the increase in W_g. Weight W_g will decrease to 0 if $\varepsilon(\mathbf{p}_1, \mathbf{d}(\mathbf{p}_2))$ is large, which means the displacement vectors $\mathbf{d}(\mathbf{p}_1)$ and $\mathbf{d}(\mathbf{p}_2)$ are different and that $N(\mathbf{p}_1)$ and $N(\mathbf{p}_2)$ may belong to different objects. In this case, a larger value of λ is used to speed up the decrease in W_g. From our experiments, we choose 0.0003 and 0.003 for the values of λ in both cases.

The connection relationship of the pixels is learned iteratively. At the kth iteration, the ith weight $W(\mathbf{p}, i, k)$ of neuron $N(\mathbf{p})$ is determined by an adaptive algorithm:

$$W(\mathbf{p}, i, k) = rW_g(\mathbf{p}, i, k) + (1 - r)W(\mathbf{p}, i, k - 1), \tag{5}$$

where k is the number of the iteration and i is the direction, $0 \leqslant i \leqslant 7$, as Fig. 4 shows. Consider pixel $\mathbf{p}$ at iteration $k - 1$: using the learning rate $r, 0 \leqslant r \leqslant 1$, we want to upgrade the weight of the ith synapse of the neuron $N(\mathbf{p})$ at iteration k toward W_g based on the weight of the ith synapse of the neuron $N(\mathbf{p})$ at iteration $k - 1$. When r is 1, $W(\mathbf{p}, i, k)$ will be $W_g(\mathbf{p}, i, k)$ at every iteration and may produce a bounce effect. If r is 0, $W(\mathbf{p}, i, k)$ will stay constant at the initial value, $W(\mathbf{p}, i, 0)$. From the analysis in Table 1 and our experiments, 0.4 was chosen as a stable learning factor.

2.2. *Modified error function for block matching*

Block-matching methods use criteria such as minimizing the sum of squared or absolute intensity differences or maximizing an intensity cross-correlation to match blocks

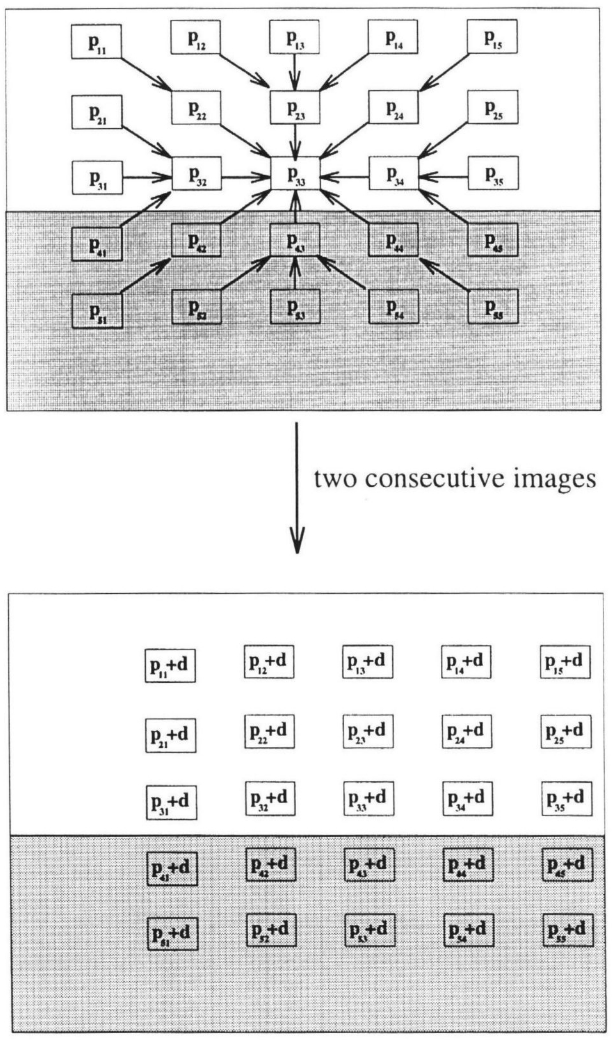
Table 1.
The value of $W(\mathbf{p}, i, k)$ for the first five iterations of the weight updated from $W(\mathbf{p}, i, 0) = 1.0$ to $W(\mathbf{p}, i, k) = 0.0$ via an adaptive method with the learning factor r from 0.0 to 1.0.

learning factor	0.0	0.2	0.4	0.6	0.8	1.0
iteration 0	1.0	1.0	1.0	1.0	1.0	1.0
iteration 1	1.0	0.8	0.6	0.4	0.2	0.0
iteration 2	1.00	0.64	0.36	0.16	0.04	0.00
iteration 3	1.000	0.512	0.216	0.064	0.008	0.000
iteration 4	1.0000	0.4096	0.1296	0.0256	0.0016	0.0000
iteration 5	1.00000	0.32768	0.07776	0.01024	0.00032	0.00000

in the next image against blocks in the current image. The displacement between these two matched blocks in the two images is the displacement vector of the matched block in the current image.

As an example, in Fig. 6, we use the criterion of minimizing the sum of squared intensity differences to calculate the matching error of pixel $\mathbf{p}_{33}$ in the first image I_1 and the corresponding pixel $\mathbf{p}_{33} + \mathbf{d}(\mathbf{p}_{33})$ in the second image I_2 with candidate displacement vector $\mathbf{d}(\mathbf{p}_{33})$. The matching error is

$$\varepsilon(\mathbf{p}_{33}, \mathbf{d}(\mathbf{p}_{33})) = \sum_{\mathbf{p} \in \text{block}(\mathbf{p}_{33})} \left(I_1(\mathbf{p}) - I_2(\mathbf{p} + \mathbf{d}(\mathbf{p}_{33}))\right)^2, \tag{6}$$



P_{11} P_{12} P_{13} P_{14} P_{15}
P_{21} P_{22} P_{23} P_{24} P_{25}
P_{31} P_{32} P_{33} P_{34} P_{35}
P_{41} P_{42} P_{43} P_{44} P_{45}
P_{51} P_{52} P_{53} P_{54} P_{55}
two consecutive images
$P_{11}+d$ $P_{12}+d$ $P_{13}+d$ $P_{14}+d$ $P_{15}+d$
$P_{21}+d$ $P_{22}+d$ $P_{23}+d$ $P_{24}+d$ $P_{25}+d$
$P_{31}+d$ $P_{32}+d$ $P_{33}+d$ $P_{34}+d$ $P_{35}+d$
$P_{41}+d$ $P_{42}+d$ $P_{43}+d$ $P_{44}+d$ $P_{45}+d$
$P_{51}+d$ $P_{52}+d$ $P_{53}+d$ $P_{54}+d$ $P_{55}+d$

Figure 6. Block matching error of pixel $\mathbf{p}_{33}$ in image I_1 and pixel $\mathbf{p}_{33} + \mathbf{d}(\mathbf{p}_{33})$ in image I_2 where $\mathbf{d}$ is $\mathbf{d}(\mathbf{p}_{33})$.

where block($\mathbf{p}_{33}$) is the block centered at $\mathbf{p}_{33}$. The estimated displacement vector $\mathbf{d}(\mathbf{p}_{33})$ minimizes the matching error $\varepsilon(\mathbf{p}_{33}, \mathbf{d}(\mathbf{p}_{33}))$. In this case, pixels $\mathbf{p}_{41}, \mathbf{p}_{42}, \ldots, \mathbf{p}_{55}$ are inside the disappearing region in image I_1; pixels $\mathbf{p}_{41}+\mathbf{d}(\mathbf{p}_{33}), \mathbf{p}_{42}+\mathbf{d}(\mathbf{p}_{33}), \ldots, \mathbf{p}_{55}+\mathbf{d}(\mathbf{p}_{33})$ are inside the appearing region in image I_2. Their intensities are different and $\varepsilon(\mathbf{p}_{33}, \mathbf{d}(\mathbf{p}_{33}))$ is large. So there may be some other displacements $\mathbf{d}'(\mathbf{p}_{33})$ satisfying $\varepsilon(\mathbf{p}_{33}, \mathbf{d}'(\mathbf{p}_{33})) < \varepsilon(\mathbf{p}_{33}, \mathbf{d}(\mathbf{p}_{33}))$ that lead to an incorrect displacement vector.

To solve the problem of estimating the displacement vector at the occluding boundary, as mentioned above, one approach is initially to divide the images into regions containing only one object and to use the matching block bounded by regions to exclude the disturbance of appearing and disappearing regions.

Instead of setting a threshold on the weights, the weights of the synapses are used as weighting values for the squares of the intensity differences because it is difficult to select a correct threshold for every application. Using fuzzy set theory, the weights of the synapses stand for the degree of the connection between the two neighboring pixels instead of the discretized true or false result. The degree of connection can be represented by the square of the intensity differences. If two pixels are on opposite sides of an occluding boundary, the synaptic weight between them will be small and lead to a small weighted square of the intensity differences, thus eliminating their disturbance on each other.

Therefore, the matching error becomes the weighted sum of the squares of the intensity differences:

$$\varepsilon(\mathbf{p}_{33}, \mathbf{d}(\mathbf{q})) = \sum_{\mathbf{p} \in \mathrm{block}(\mathbf{p}_{33})} \Big(\big(I_1(\mathbf{p}) - I_2(\mathbf{p} + \mathbf{d}(\mathbf{q}))\big)^2 W_{\mathrm{path}}(\mathbf{p}, \mathbf{p}_{33}) \Big), \tag{7}$$

where $\mathbf{q}$ is one of eight neighbors of $\mathbf{p}_{33}$. Weight $W_{\mathrm{path}}(\mathbf{p}, \mathbf{p}_{33})$ is the product of the weights of the synapses along the shortest path from $\mathbf{p}$ to the center pixel $\mathbf{p}_{33}$ of the block as shown in next equation; thus, pixels on the other side of the occluding boundary will have little influence on the error.

As in the example of Fig. 6, the error of iteration k is

$$\begin{aligned}
\varepsilon(\mathbf{p}_{33}, \mathbf{d}(\mathbf{q})) = {} & \big(I_1(\mathbf{p}_{11}) - I_2(\mathbf{p}_{11} + \mathbf{d}(\mathbf{q}))\big)^2 W(\mathbf{p}_{11}, 7, k-1) W(\mathbf{p}_{22}, 7, k-1) \\
& + \big(I_1(\mathbf{p}_{12}) - I_2(\mathbf{p}_{12} + \mathbf{d}(\mathbf{q}))\big)^2 W(\mathbf{p}_{12}, 7, k-1) W(\mathbf{p}_{23}, 6, k-1) \\
& \vdots \\
& + \big(I_1(\mathbf{p}_{32}) - I_2(\mathbf{p}_{32} + \mathbf{d}(\mathbf{q}))\big)^2 W(\mathbf{p}_{32}, 3, k-1) \\
& + \big(I_1(\mathbf{p}_{33}) - I_2(\mathbf{p}_{33} + \mathbf{d}(\mathbf{q}))\big)^2 \\
& + \big(I_1(\mathbf{p}_{34}) - I_2(\mathbf{p}_{34} + \mathbf{d}(\mathbf{q}))\big)^2 W(\mathbf{p}_{34}, 4, k-1) \\
& \vdots \\
& + \big(I_1(\mathbf{p}_{54}) - I_2(\mathbf{p}_{54} + \mathbf{d}(\mathbf{q}))\big)^2 W(\mathbf{p}_{54}, 0, k-1) W(\mathbf{p}_{43}, 1, k-1) \\
& + \big(I_1(\mathbf{p}_{55}) - I_2(\mathbf{p}_{55} + \mathbf{d}(\mathbf{q}))\big)^2 W(\mathbf{p}_{55}, 0, k-1) W(\mathbf{p}_{44}, 0, k-1).
\end{aligned}$$

Because $\mathbf{p}_{41}, \mathbf{p}_{42}, \ldots, \mathbf{p}_{45}$ and $\mathbf{p}_{31}, \mathbf{p}_{32}, \ldots, \mathbf{p}_{35}$ are at opposite sides of the occluding boundary, $W(\mathbf{p}_{41}, 2, k-1)$, $W(\mathbf{p}_{42}, 2, k-1)$, $W(\mathbf{p}_{43}, 1, k-1)$, $W(\mathbf{p}_{44}, 0, k-1)$, $W(\mathbf{p}_{45}, 0, k-1)$ will approach zero and thus $\mathbf{p}_{41}, \mathbf{p}_{42}, \ldots, \mathbf{p}_{55}$ have little influence on $\varepsilon(\mathbf{p}_{33}, \mathbf{d}(\mathbf{q}))$. Therefore, the errors of the differences between $\mathbf{p}_{41}, \ldots, \mathbf{p}_{55}$ in image I_1 and $\mathbf{p}_{41} + \mathbf{d}(\mathbf{q}), \mathbf{p}_{42} + \mathbf{d}(\mathbf{q}), \ldots, \mathbf{p}_{55} + \mathbf{d}(\mathbf{q})$ in image I_2 will not affect $\varepsilon(\mathbf{p}_{33}, \mathbf{d}(\mathbf{q}))$ because they have to be multiplied by $W(\mathbf{p}_{41}, 2, k-1), W(\mathbf{p}_{42}, 2, k-1), \ldots,$ or $W(\mathbf{p}_{45}, 0, k-1)$. Using this we can compute a more accurate displacement vector at the occluding boundary.

2.3. *Flood-fill algorithm*

In the standard block-matching method, the dense displacement field is obtained by exhaustively matching all the possible corresponding blocks within a predefined range. This kind of matching is very inefficient and may not compute a smooth displacement field because every pixel is estimated separately. In this section, a flood-fill algorithm is used to estimate the dense displacement field efficiently. Another implementation issue is for the neural network computation which is performed in parallel on a dense mesh of computing neurons and synapses. Instead of monitoring all the neurons and synapses to determine if they have to be updated or not, the flood-fill algorithm maintains and processes through a queue of neurons and synapses which have to be adjusted. All the affected neurons and synapses are appended to the queue.

However, the iterative displacement-field estimation and segmentation procedure requires an initial value. When matching any two consecutive images of sequence, we assume all pixels are stationary, that is, their displacements are all zero, and the initial matching error using zero displacements are calculated. Further, we regard the whole image as one object, that is, the weights of the synapses of the first neural network are all 1. A recursive flood-fill search algorithm is used to produce a dense displacement field:

Step 1: All pixels in the two consecutive images are set to be unmarked.
Step 2: Randomly choose an unmarked pixel $\mathbf{p}_1$ in I_1. Exhaustively search an area surrounding $\mathbf{p}_1$ in I_2. Choose the pixel $\mathbf{p}_2$ which gives the minimum matching error $\varepsilon(\mathbf{p}_1, \mathbf{p}_2 - \mathbf{p}_1)$. Let $\mathbf{p} = \mathbf{p}_1$, $\mathbf{d}(\mathbf{p}) = \mathbf{p}_2 - \mathbf{p}_1$.
Step 3: For every pixel $\mathbf{p}'$ of the neighbors of pixel $\mathbf{p}$, calculate the matching error $\varepsilon(\mathbf{p}', \mathbf{d}(\mathbf{p}_1))$ and modify the weight of the synapse between $\mathbf{p}$ and $\mathbf{p}'$ by displacement $\mathbf{d}(\mathbf{p})$.
Step 4: If $\min_{-1 \leqslant i \leqslant 1, -1 \leqslant j \leqslant 1} \varepsilon(\mathbf{p}', \mathbf{d}(\mathbf{p}_1) + (i, j))$ is less than the previous matching error of $\mathbf{p}'$, let $\mathbf{d}(\mathbf{p}') \leftarrow \mathbf{d}(\mathbf{p}_1) + (i, j)$; mark $\mathbf{p}'$; let $\mathbf{p} = \mathbf{p}'$; and go to *Step 3*.
Step 5: If there are unmarked pixels, go to *Step 2*, or else stop.

Every object in the image will have a displacement seed to flood fill the whole region of the object, according to the assumption that neighboring pixels have similar displacements within objects and pixels across occluding boundaries have dissimilar displacements.

3. IMAGE SEGMENTATION

3.1. Introduction

Image segmentation is usually the first step of image analysis, after some preprocessing or enhancement. The resultant regions are typically matched with, recognized or interpreted with respect to a knowledge base to obtain the descriptions of objects in the image. The result of segmentation plays an important role in the overall success of an imaging problem. If the segmentation result is good, the complexity and difficulty is reduced because the whole image is partitioned into regions and each region contains only one object.

Segmentation algorithms are based on two properties: similarity and discontinuity. Pixels in the images are grouped together if they have some similar attributes. Boundaries of the regions are located at the discontinuous part according to these attributes. Most segmentation algorithms work on intensity and spatial attributes such as intensity, similarity, and discontinuity, detecting regions including isolated points, lines, edges, regions with peaks or valleys in intensity, and texture. Gradient operators, the Laplacian of a Gaussian operator, zero-crossing detection, thresholding, and region growing are the most often used segmentation algorithms. Some corner-detection algorithms trace the line which is detected by another segmentation algorithm and detect the corner by using the attribute of curvature.

In this work, image segmentation is defined as partitioning each image into a set of moving objects and background and we use an iterative displacement-field estimation and segmentation algorithm to do these two operations simultaneously. All the segmentation information is already embedded in the neural network. The remaining problem is to retrieve and represent the segmented regions from the neural network. One segmentation algorithm is based on region growing to aggregate all the neurons connected by strong synapses. This algorithm uses the similarity of activation criterion property because the neurons connected by strong synapses have similar displacement vectors. Another approach to retrieving segmentation information is to locate the boundary at the weak synapses. This algorithm uses the discontinuity property because the neurons connected by weak synapses have different displacement vectors. We present these two segmentation algorithms in Sections 3.2 and 3.3.

3.2. Region growing segmentation

In this section, a segmentation algorithm based on region growing is presented. Neurons are aggregated via the weights of the synapses connecting neighboring neurons. Neurons of similar high synaptic weight (see *Step 4* below) are aggregated together. Because the large weights of some synapses mean the neurons connected by this synapse have coherent displacement vectors, the resulting segmentation will partition the image into regions with the same motion. This algorithm is as follows.

Input: Neural network.
Output: Segmented regions.

Step 1: Initialize an empty queue.
Step 2: Append the queue with an ungrouped neuron in the neural network; assign a new region number for it.
Step 3: Retrieve the neuron N from the head of the queue.
Step 4: For each N_n ($0 \leqslant n \leqslant 7$) of the eight ungrouped neighboring neurons of N, if the weight of synapse W between neuron N_n and N is larger than $\frac{1}{3}$ of the 72 synapses of neuron N and its 8 neighbors, N_n is grouped by the same region number of N and is appended to the queue.
Step 5: If the queue is not empty, go to *Step 3*.
Step 6: If there is any ungrouped neuron, go to *Step 2*.
Step 7: Output the regions in which all the neurons in the same region have the same region number.

3.3. Boundary segmentation

Instead of aggregating neurons as regions, the segmentation algorithm presented in this section draws the boundaries using the connection information stored in the synapses of the neural network. If the weight of some synapse is small, the neurons connected by this synapse have different displacement vectors and should belong to two different objects and thus a boundary exists between these two neurons. There are six cases when we draw boundaries between any four neighboring pixels as shown in Fig. 7.

If the weights of the three or four synapses between the four neighboring neurons are all smaller than the threshold value, we draw the boundary indicated. This method cannot produce a clear-cut boundary but is a satisfactory indicator. The output of the boundary will depend on the choice of the threshold value, which is 0.75 in these experiments.

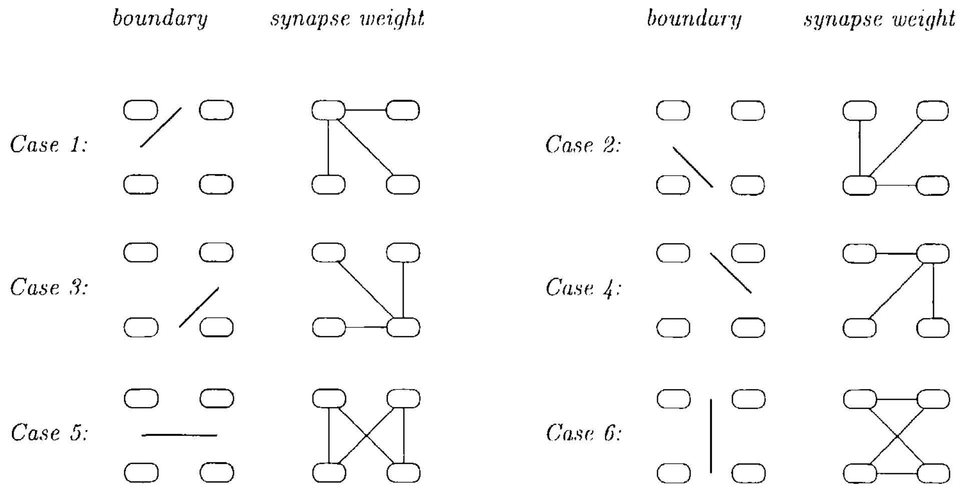


Figure 7. The six cases of drawing boundaries between the four neighboring neurons.

4. RESULTS OF EXPERIMENTS

In this section, we will demonstrate some results of the experiments using these methods with two image sequences. The first image sequence is a cube sequence consisting of a magic cube on a round plate, as shown in Fig. 8. The round plate

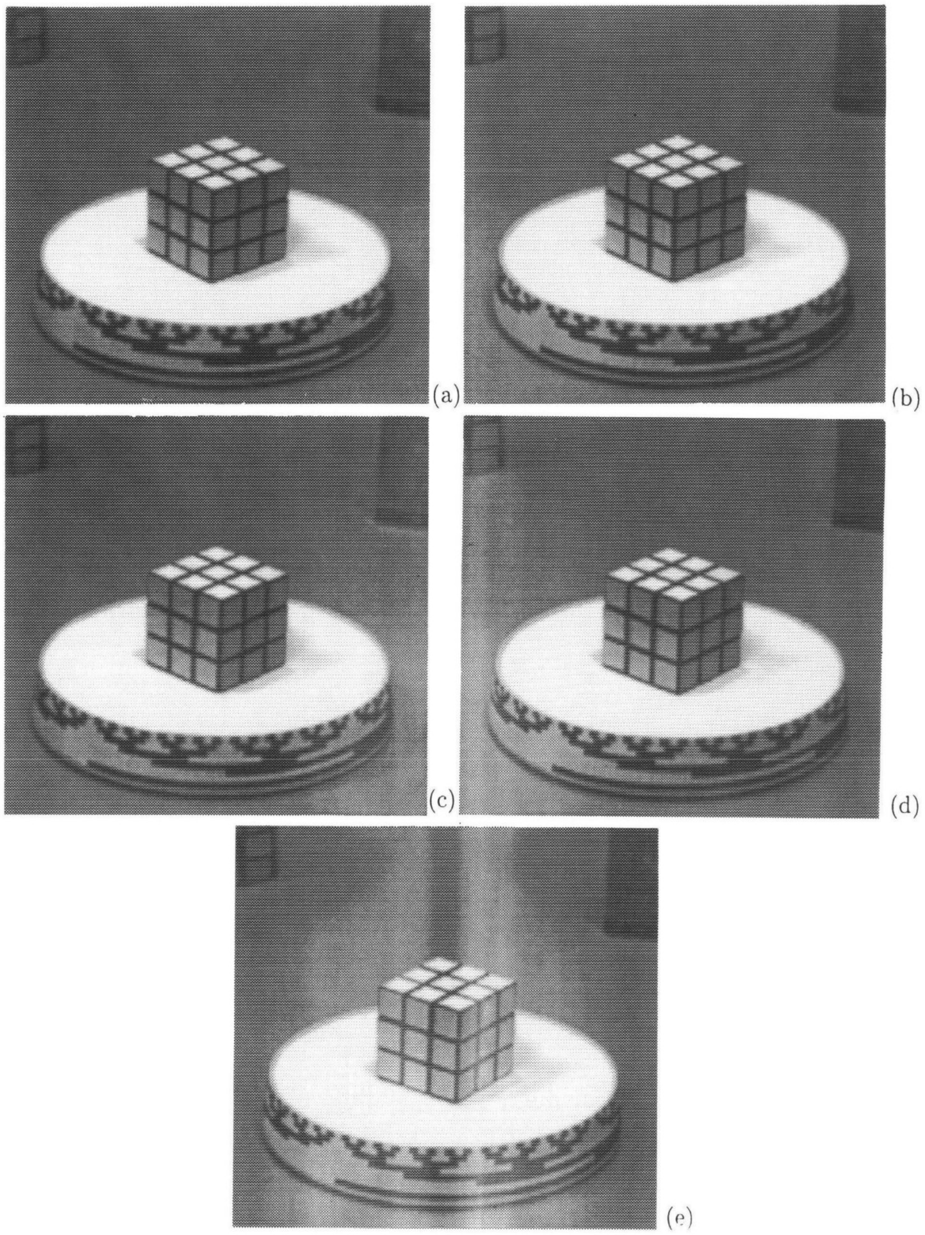


Figure 8. (a)–(e) Consecutive images of a rotating plate with a cube on it.

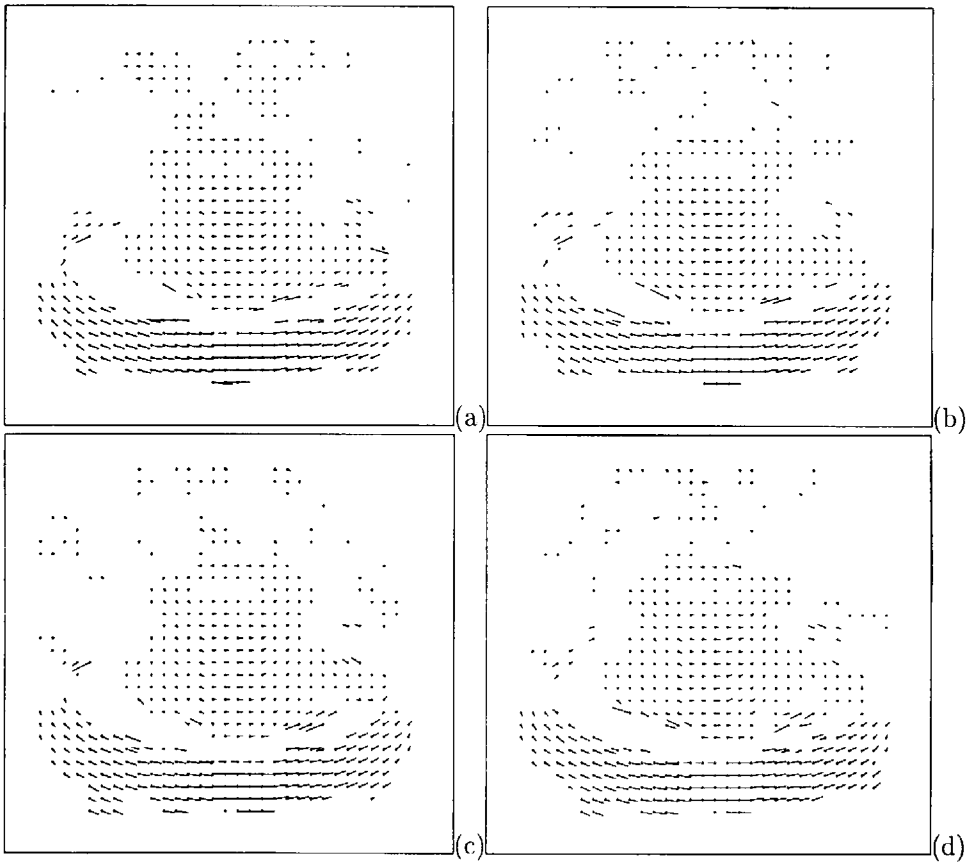


Figure 9. (a)–(d) Displacement fields between two consecutive images.

is rotating counter-clockwise with the magic cube. The displacement fields of this sequence are shown in Fig. 9 without showing the zero displacement vectors (e.g. inside the round plate). Three major partitions and the boundary of the first two images are shown in Fig. 10 by using the region-growing and boundary-segmentation algorithms. The magic cube and the round plate are grouped together because they have the same motion: rotation.

The next image sequence is the taxi sequence, as shown in Fig. 11. There are three cars moving in this image sequence: the white taxi turning right around the street corner, the black sedan at the left side of the road moving toward the right side, and the truck at the right side of the road moving toward the left. Except for the white taxi, the sedan and the truck are somewhat blurred and thus it is hard to estimate the displacement vectors and to divide them, as Fig. 12 shows. The segmentation result in Fig. 13 shows that the white taxi is partitioned into three parts as the taxi is turning right and the taxi displacement vectors are not all the same.

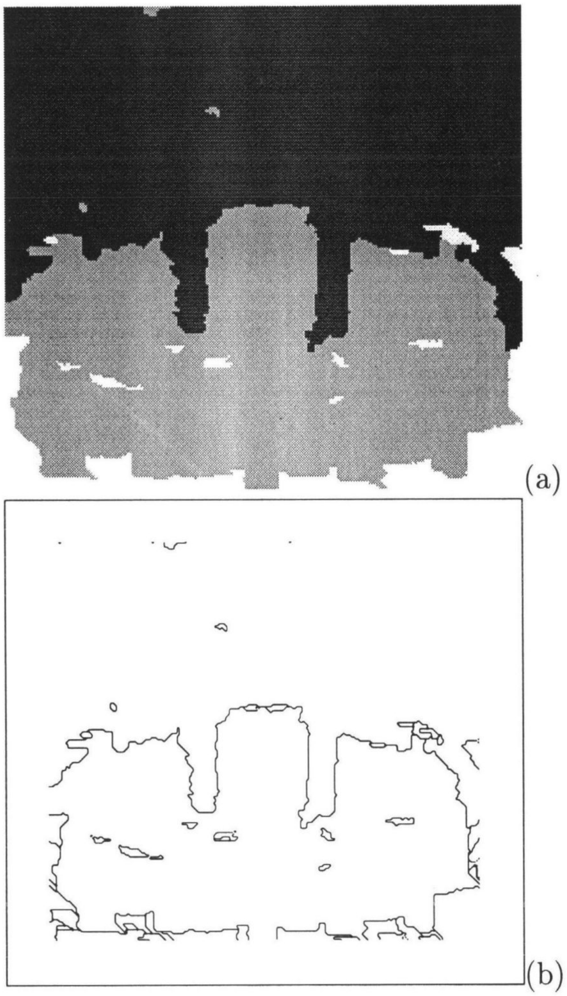


Figure 10. Segmentation result for the first two consecutive images produced by (a) the region-growing segmentation algorithm and (b) the boundary-segmentation algorithm.

5. CONCLUSION

In this paper, we put forward a block-matching method enhanced by a single-layer feedback neural network to estimate the motion-displacement field and segment the image simultaneously in an iterative manner. This neural network is trained by and maintains the segmentation information during estimation of the displacement field. The enhanced block matching uses the segmentation information embedded in the neural network to estimate the displacement vectors accurately. These algorithms were applied to various indoor and outdoor real-world image sequences; the experimental results showed that the displacement vectors and object boundaries were accurately located even near the occluding boundary. Indeed, using synaptic weights and the

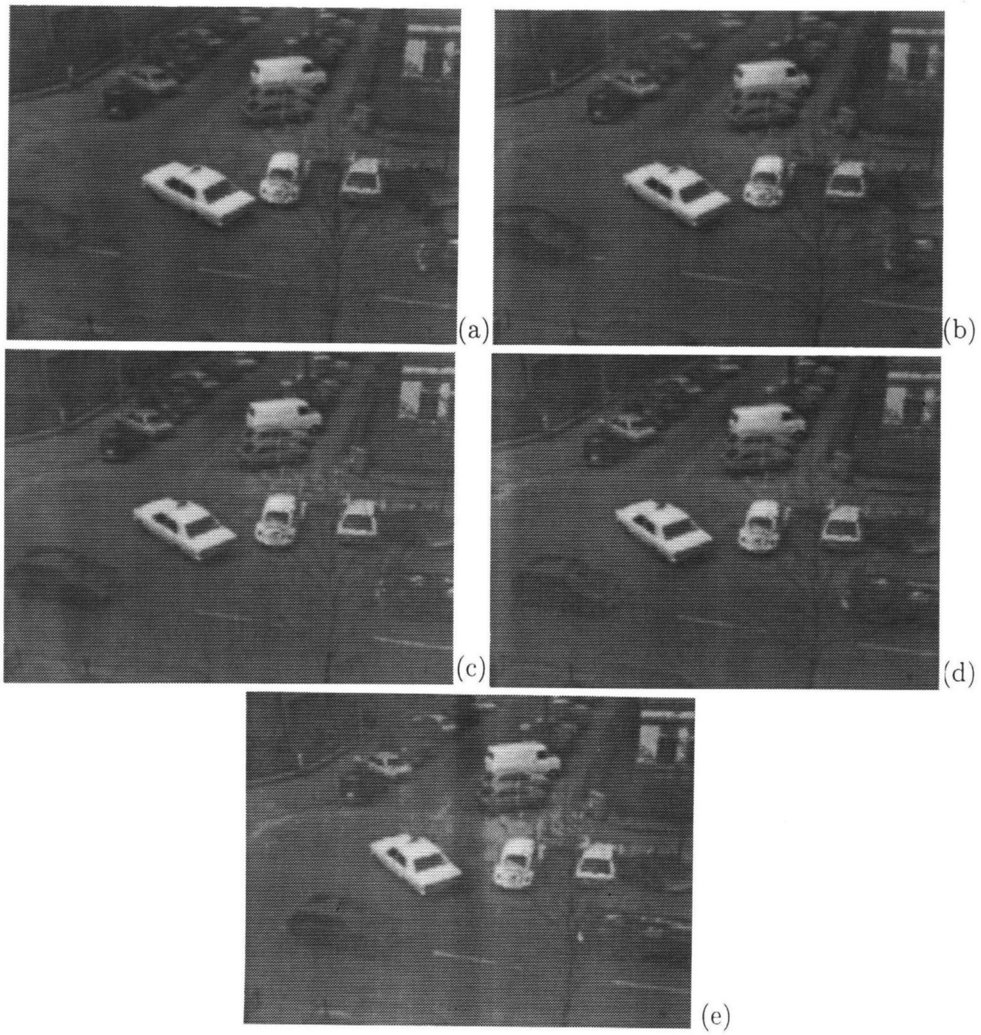


Figure 11. (a)–(e) Consecutive images of moving cars.

displacement fields from previous images as initial values improved displacement-field estimation and image segmentation.

Throughout this work, we have assumed that the objects in the consecutive images only go through the motion of 2D translation in the $X-Y$ plane. If the objects in the images have motion such as rotation, scaling, or intensity change, our algorithm cannot achieve subpixel accuracy and has to be refined.

The problem of choosing the size of the matching block remains unsolved. Some systems use a variable size to adapt to different neighborhood intensity variations. One way to estimate the amount of variability in a region is to use the statistic entropy. If $P(i)$ denotes the probability of the appearance of intensity i, the information contained by the intensity i is $-\log P(i)$. The size of the matching blocks is chosen when

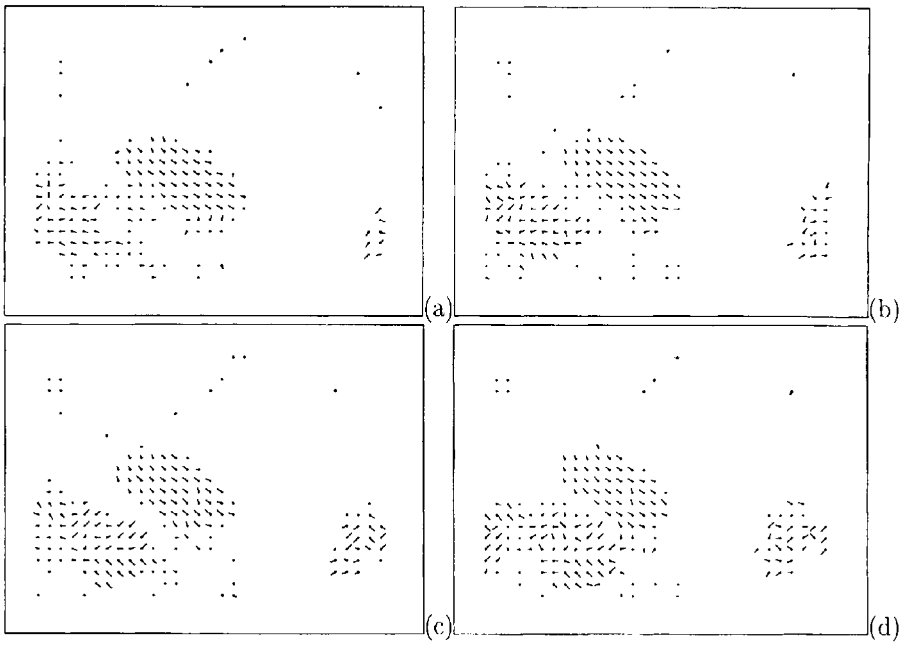


Figure 12. (a)–(d) Displacement fields between two consecutive images.

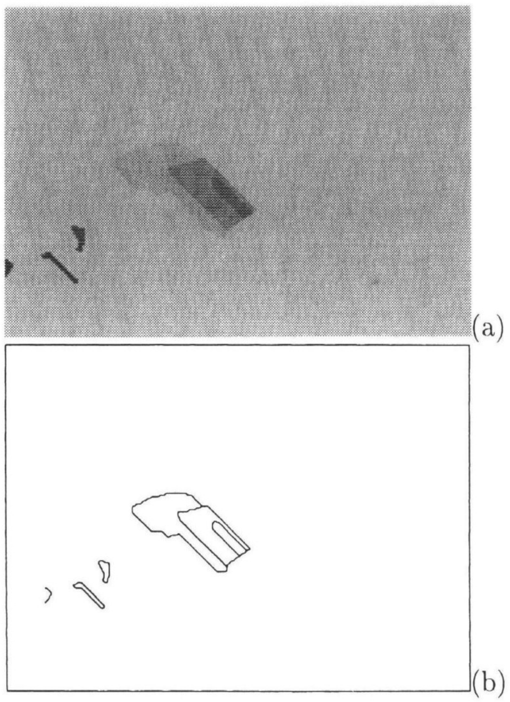


Figure 13. Segmentation result for the first two consecutive images produced by (a) the region-growing segmentation algorithm and (b) the boundary-segmentation algorithm.

the total information $-\sum_{i\in\text{block}}\log P(i)$ is large enough. Such issues need further exploration.

Acknowledgement

This research work was supported by National Science Council of Taiwan, under NSC Grants NSC 83-0422-E-002-010, NSC 84-2212-E-002-046, and NSC 85-2212-E-002-077, and by Cho-Chang Tsung Foundation of Education under Grant 84-S-26.

REFERENCES

Aggarwal, J. K., Davis, L. S. and Martin, W. N. (1981). Correspondence processes in dynamic scene analysis. *Proc. IEEE* **69**, 562–572.

Barnard, S. T. and Thompson, W. B. (1980). Disparity analysis in images. *IEEE Trans. Pattern Analysis and Machine Intelligence* **2**, 333–340.

Costa, M. S., Haralick, R. M. and Shapiro, L. G. (1990). Optimal affine invariant point matching. In: *Proc. Intl. Conf. on Pattern Recognition*. Atlantic City, pp. 233–236.

Dufaux, F. and Kunt, M. (1992). Multigrid block matching motion estimation with an adaptive local mesh refinement. In: *Proc. of Visual Communications and Image Processing*. Boston, pp. 97–109.

François, E. and Bouthemy, P. (1990). Derivation of qualitative information in motion analysis. *Image and Vision Computing* **8**, 279–288.

Fuh, C. S. and Maragos, P. (1989). Region-based optical flow estimation. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*. San Diego, pp. 130–135.

Fuh, C. S. and Maragos, P. (1990). Application of mathematical morphology to motion image analysis. In: *Proc. Electronic Imaging EAST Conference*. Boston, pp. 261–264.

Fuh, C. S. and Maragos, P. (1991a). Affine models for image matching and motion detection. In: *Proc. of International Conference on Acoustics, Speech, and Signal Processing*. Toronto, Canada, pp. 2409–2412.

Fuh, C. S. and Maragos, P. (1991b). Motion displacement estimation using an affine model for image matching. *Optical Engineering* **30**, 881–887.

Fuh, C. S., Maragos, P. and Vincent, L. (1991c). Region-based approaches to visual motion correspondence. Technical Report 91–98, Harvard Robotics Laboratory.

Horn, B. K. P. and Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence* **17**, 185–203.

Irani, M., Rousso, B. and Peleg, S. (1994). Computing occluding and transparent motions. *International Journal of Computer Vision* **12**, 5–16.

Lappe, M. and Rauschecker, J. P. (1993). A neural network for the processing of optic flow from ego-motion in man and higher mammals. *Neural Computation* **5**, 374–391.

Lippmann, R. P. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4–22.

Pal, N. R. and Pal, S. K. (1993). A review on image segmentation techniques. *Pattern Recognition* **26**, 1277–1294.

Schnörr, C. (1992). Computation of discontinuous optical flow by domain decomposition and shape optimization. *International Journal of Computer Vision* **8**, 153–165.

Schunck, B. G. and Horn, B. K. P. (1981). Constraints on optical flow computation. In: *Proc. Pattern Recognition and Image Processing Conference*. Dallas, pp. 205–210.

Seferidis, V. and Chanbari, M. (1992). Generalized block matching motion estimation. In: *Proc. of Visual Communications and Image Processing*. Boston, pp. 110–119.

Thompson, W. B., Mutch, K. M. and Berzins, V. A. (1985). Dynamic occlusion analysis in optical flow fields. *IEEE Trans. Pattern Analysis and Machine Intelligence* **7**, 374–383.