In Proceedings of the Image Segmentation Workshop, Sydney, Australia, Dec. 1996, pages 13–19

Neural Network for Displacement Field Estimation and Image Segmentation Using Block Matching

Yong-Sheng Chen, Chiou-Shann Fuh[1] and *Yi-Ping Hung*
Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan

Abstract *In this paper, we propose a single-layer feedback neural network model to enhance the block matching, to estimate the displacement field, and simultaneously to perform image segmentation from consecutive images. For any two consecutive images, we create a neural network to learn the connection relationship of the pixels in an object from the displacement field and to store 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 our 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.*

1 Introduction

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 sensing, and many other real life applications.

Optical flow estimation and the image segmentation are the first two tasks in motion analysis. There are three major groups of methods to estimate the optical flow field or displacement field. They are the gradient methods, correspondence methods, and block matching methods. Each group of methods has its own advantages, drawbacks, and limitations for different situations.

Gradient methods are effective and they can compute the dense optical flow field at subpixel accuracy. But they need derivatives, which are sensitive to noise, and have difficulty with long range optical flow. Correspondence methods extract the interesting part of every consecutive images to match and track them by similarity, defined according to applications to compute the displacement field. The interesting part of an image depends on applications, such as isolated point, edge, corner, peak, valley, blob, or suchlike. Although this method can only compute sparse displacement field, it can achieve long range displacement while eliminating the disturbance of noise. 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.

The simplest block matching is to find in two images the corresponding pair of pixels and their surrounding blocks that minimize the sum of square of intensity differences. This method is feasible if there are only translations but no rotations. Fuh and Maragos (1991a, 1991b) develop a 2-D affine model to estimate the displacement field allowing affine intensity transformations and shape deformations. Dufaux and Kunt (1992) use a multigrid block matching refined by an adaptive local mesh to obtain more accurate motion field efficiently. Seferidis and Chanbari (1992) propose 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. The block matching error using the sum of the square of the intensity differences will depend on the intensity difference between appearing regions 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 the inefficient and exhaustive block matching may lead to incorrect displacement vectors.

Block matching can achieve long-range dense displacement field and is robust to noise. But it can not estimate the displacement vector at the subpixel level, and it is very difficult to accurately es-

[1]To whom all correspondence should be sent.
TEL: 886-2-3625336 ext 327
FAX: 886-2-3628167
Email: fuh@csie.ntu.edu.tw

timate the displacement vector near the occluding boundary. Choosing the size of matching block is also a problem.

Image segmentation is an important preliminary step for pattern recognition and scene analysis. Segmentation is based on similarity and discontinuity. François and Bouthemy (1990) use likelihood test to divide motion field into areas containing coherent motion. Thompson *et al.* (1985) use an edge detection algorithm to locate the discontinuities of optical flow, that is, the occluding boundaries. Irani *et al.* (1994) use temporal integration to divide and track the moving objects.

Using optical flow field as an intermediate result 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.

In this paper, we put forward an iterative method of the displacement field estimation and the image segmentation. We create a single-layer feedback neural network model to learn and keep the information of segmentation from the displacement field estimated by an enhanced block matching method. This block matching will use the information of segmentation embedded in the neural network to enhance the accuracy of the displacement field. This neural network model is trained by the pixels of the input sequence of images to produce accurate displacement field and segmentation.

2 Displacement Field Estimation

2.1 Neuron Model

In this section, we create a single-layer feedback neural network for each image to learn and record the segmentation information in the artificial synapses between neurons. This neural network contains one neuron for every pixel in the image. Using the error function described in Section 2.2, each neuron contains the information of the displacement vector and the block matching error is computed with this displacement vector. Each neuron also has eight synapses connected to the eight neighboring neurons, corresponding to the eight neighbors of the pixel. The construction of neurons and synapses is shown in Figure 1. The eight synapses of each neuron are ordered from number 0 to number 7 as Figure 1 shows.

The weight of synapse stands for the strength of the connection. That means if two neighboring pixels belong to the same object, the weight of synapse between the two corresponding neurons should be large. As Figure 2 shows, if the matching error $\epsilon(r,c)$ of neuron $\mathrm{N}(r,c)$ using displacement vector $\mathbf{d}(r,c)$ is calculated, the feedback connection will

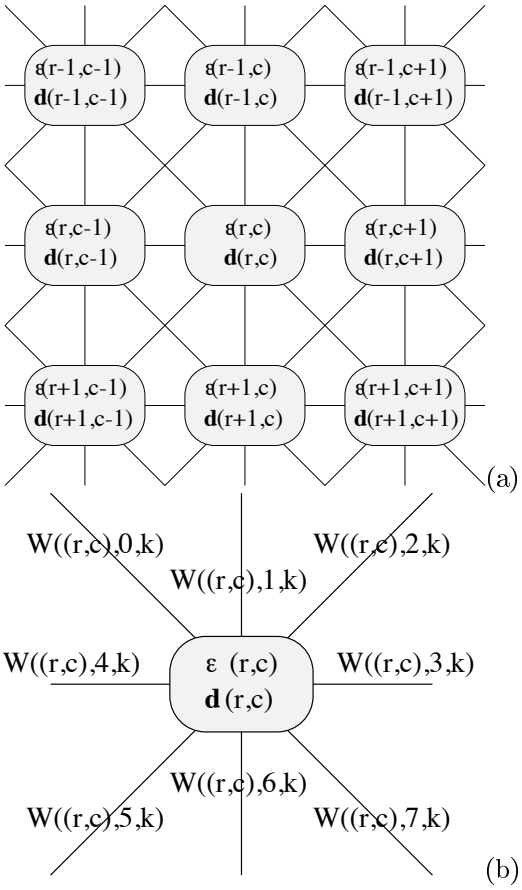


Figure 1: *(a) The eight synapses of the neuron N(r, c) with the number on the synapses at the kth iteration. This order of synapses is just for the convenience of implementation. (b) The neuron N(r, c) contains the information of block matching error $\varepsilon(r,c)$, calculated by using the displacement vector $\mathbf{d}(r,c)$ and eight synapses connected to its neighbors.*

adjust the weights of synapses connecting neuron $\mathrm{N}(r,c)$ and its eight neighboring neurons to reveal the new state of the connection strength.

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 vector $\mathbf{d}(\mathbf{p}_1)$, $\mathbf{d}(\mathbf{p}_2)$ should be different and the matching errors $\epsilon(\mathbf{p}_1, \mathbf{d}(\mathbf{p}_2))$ of neuron $\mathrm{N}(\mathbf{p}_1)$ using the displacement vector $\mathbf{d}(\mathbf{p}_2)$ and $\epsilon(\mathbf{p}_2, \mathbf{d}(\mathbf{p}_1))$ of neuron $\mathrm{N}(\mathbf{p}_2)$ using the displacement vector $\mathbf{d}(\mathbf{p}_1)$ are both larger than $\epsilon(\mathbf{p}_1, \mathbf{d}(\mathbf{p}_1))$ and $\epsilon(\mathbf{p}_2, \mathbf{d}(\mathbf{p}_2))$.

The relationship presented above can be translated into the weight of the synapse. According to the assumption above, if the matching error

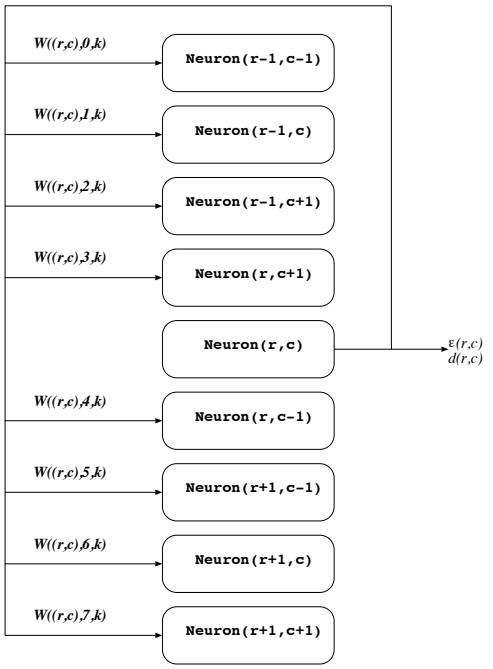


Figure 2: *Single-layer feedback neural network model for the feedback connection of Neuron(r,c) only.*

$\epsilon(\mathbf{p}_1, \mathbf{d}(\mathbf{p}_2))$ is large, then $\mathbf{p}_1$, $\mathbf{p}_2$ should belong to different objects and the weight of the synapse between neurons $\mathrm{N}(\mathbf{p}_1)$ and $\mathrm{N}(\mathbf{p}_2)$ should be small. On the contrary, if $\epsilon(\mathbf{p}_1, \mathbf{d}(\mathbf{p}_2))$ is small, $\mathbf{d}(\mathbf{p}_1)$ 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 $\mathrm{N}(\mathbf{p}_1)$ and $\mathrm{N}(\mathbf{p}_2)$ should be large. The ith goal weight $W_g(\mathbf{p}_1, i, k)$ of neuron $\mathrm{N}(\mathbf{p}_1)$ at the kth iteration is defined as the reciprocal of exponential error $\epsilon(\mathbf{p}_1, \mathbf{d}(\mathbf{p}_2))$ scaled by λ:

$$W_g(\mathbf{p}_1, i, k) = \frac{1}{e^{\lambda \times \epsilon(\mathbf{p}_1, \mathbf{d}(neighbor_i(\mathbf{p}_1)))}} \qquad (1)$$

where $\mathbf{p}_2 = neighbor_i(\mathbf{p}_1)$ is the ith neighboring neuron of $\mathbf{p}_1$, $0 \le i \le 7$, as Figure 1 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 $\epsilon(\mathbf{p}_1, \mathbf{d}(\mathbf{p}_2))$ approaches 0 to represent that $\mathrm{N}(\mathbf{p}_1)$ and $\mathrm{N}(\mathbf{p}_2)$ belong to the same object because $\mathbf{d}(\mathbf{p}_1)$ is similar to $\mathbf{d}(\mathbf{p}_2)$. In this case, a smaller value of λ is used to speed up the increase of W_g. Weight W_g will decrease to 0 if $\epsilon(\mathbf{p}_1, \mathbf{d}(\mathbf{p}_2))$ is large, which means the displacement vectors $\mathbf{d}(\mathbf{p}_1)$ and $\mathbf{d}(\mathbf{p}_2)$ are different and that $\mathrm{N}(\mathbf{p}_1)$ and $\mathrm{N}(\mathbf{p}_2)$ may belong to different objects. In this case, a larger value of λ is used to speed up the decrease of W_g. From our experimental experience, we choose 0.0003 and 0.003 to be the value of λ in both cases. The weight of synapse stands for the strength of the connection of pixels and neurons.

The connection relationship of pixels is first learned iteratively each time a displacement vector is estimated and the matching error is calculated. After all the pixels are processed, the connection information embedded in the neural network is sustained and learned progressively from the consecutive images. At the kth iteration, the ith weight $W(\mathbf{p}, i, k)$ of neuron $\mathrm{N}(\mathbf{p})$ is learned by using an adaptive algorithm:

$$W(\mathbf{p}, i, k) = r \times W_g(\mathbf{p}, i, k) + (1 - r) \times W(\mathbf{p}, i, k-1) \qquad (2)$$

where k is the number of iteration and i is one direction and $0 \le i \le 7$, as Figure 1 shows. Consider pixel $\mathbf{p}$ at iteration $k-1$: using the learning rate $r, 0 \le r \le 1$, we want to upgrade the weight of the i-th synapse of the neuron $\mathrm{N}(\mathbf{p})$ at iteration k toward W_g based on the weight of the i-th synapse of the neuron $\mathrm{N}(\mathbf{p})$ 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 have a bouncing effect. If r is 0, $W(\mathbf{p}, i, k)$ will stay constant on the initial value, $W(\mathbf{p}, i, 0)$. From our experiments, 0.4 is chosen to be a stable learning factor.

2.2 Modified Error Function of Block Matching

Block matching methods use some criteria such as minimizing a sum of squared or absolute intensity differences or maximizing an intensity cross-correlation to match blocks in the next image for each block in the current image. The displacement between these two matched blocks in the current and next images is the displacement vector of the matched block in the current image.

As an example, in Figure 3, we use the criterion of minimizing a 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 by the two surrounding blocks with candidate displacement vector $\mathbf{d}(\mathbf{p}_{33})$. The matching error according to the criterion of minimizing a sum of squared intensity differences is:

$$\varepsilon(\mathbf{p}_{33}, \mathbf{d}(\mathbf{p}_{33})) = \sum_{\mathbf{p} \in block(\mathbf{p}_{33})} (I_1(\mathbf{p}) - I_2(\mathbf{p} + \mathbf{d}(\mathbf{p}_{33})))^2 \qquad (3)$$

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 $\mathbf{p}_{41}, \mathbf{p}_{42}, .., \mathbf{p}_{55}$ are inside the disappearing region in image I_1; pixels $\mathbf{p}_{41} + \mathbf{d}(\mathbf{p}_{33}), \mathbf{p}_{42} + \mathbf{d}(\mathbf{p}_{33}), .., \mathbf{p}_{55} + \mathbf{d}(\mathbf{p}_{33})$ 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 men-

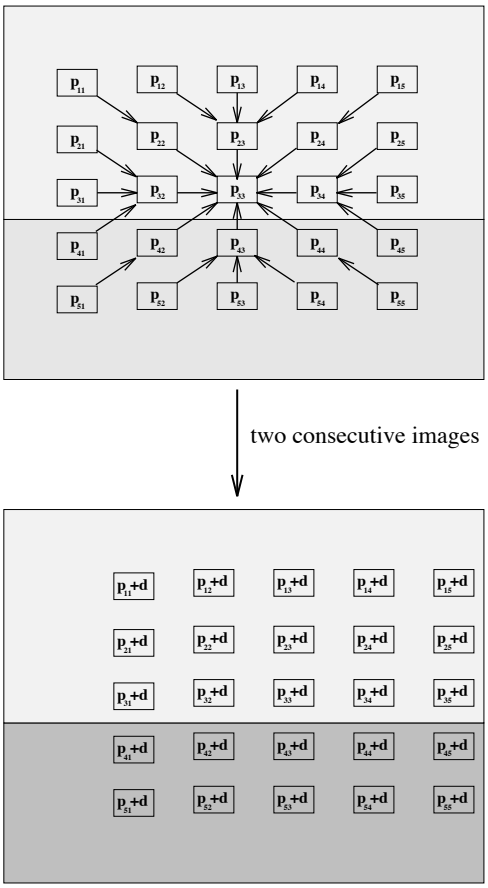


Figure 3: *Block matching error of pixel* $\mathbf{p}_{33}$ *in image* I_1 *and pixel* $\mathbf{p}_{33}+\mathbf{d}(\mathbf{p}_{33})$ *in image* I_2 *where* $\mathbf{d}$ *is* $\mathbf{d}(\mathbf{p}_{33})$.

tioned above, one approach is previously to divide the images into regions containing only one object and to use the matching block bounded by regions in order to exclude the disturbance of appearing and disappearing regions. As Section 2.1 mentions, our approach embeds the segmentation information in the neural network, we can exploit it from the weights of the synapses.

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

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

$$\epsilon(\mathbf{p}_{33},\ \mathbf{d}(\mathbf{q})) = \sum_{\mathbf{p}\in block(\mathbf{p}_{33})} \big((I_1(\mathbf{p}) - I_2(\mathbf{p}+\mathbf{d}(\mathbf{q})))^2 \times W_{path}(\mathbf{p},\mathbf{p}_{33})\big) \tag{4}$$

where $\mathbf{q}$ is one of eight neighbors of $\mathbf{p}_{33}$. Weight $W_{path}(\mathbf{p},\mathbf{p}_{33})$ 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 error.

2.3 Flood Fill Algorithm

In the standard block matching method, the dense displacement field is obtained from 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 parallelly performed on a dense mesh of computing neurons and synapses, On traditional computers, parallel processing is often simulated with asynchronous sequential computations. Instead of watching all the neurons and synapses to know 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 while processing the queue will be appended behind the queue, eliminating the need to watch all the neurons and synapses.

Unlike the circular chicken-or-the-egg problem, the iterative displacement field estimation and segmentation requires an initial value. When matching any two consecutive images of an image sequence, we assume all pixels are stationary, that is, their displacements are all zero, and the initial matching error using zero displacements are calculated. Besides, 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 $\epsilon(\mathbf{p}_1, \mathbf{p}_2-\mathbf{p}_1)$. 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 $\epsilon(\mathbf{p}', \mathbf{d}(\mathbf{p}_1))$ and mod-

ify the weight of the synapse between $\mathbf{p}$ and $\mathbf{p}'$ by displacement $\mathbf{d}(\mathbf{p})$.

Step 4: If $\min_{-1\le i\le 1,-1\le j\le 1} \epsilon(\mathbf{p}', \mathbf{d}(\mathbf{p}_1) + (i,j))$ is less than the old 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 the object but the pixels across the occluding boundary have dissimilar displacements.

3 Image Segmentation

In this work, image segmentation is defined as partitioning each image into a set of moving objects and background. In Sections 3.1 and 3.2, two segmentation algorithms are presented. In fact, they are actually the post-processings of segmentation. Recall that 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 partitions from the neural network.

3.1 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 to neighboring neurons. If the weight of some synapse is large compared with the local neighboring weights, which are ranged as the 72 weights of synapses of the neuron and its eight neighboring neurons, the neurons connected by this synapse are aggregated together. Because the large weight of some synapse means 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 described below:

Input: The neural network.

Output: The 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 $(0 \le n \le 7)$ 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 the 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.

3.2 Boundary Segmentation

Instead of aggregating neurons as regions, the segmentation algorithm presented in this section draw 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 neurons.

If the weights of the three or four synapses between the four neighboring neurons are all smaller than some threshold value, we draw a boundary there. This method cannot get the 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 from our experiments.

4 Results of Experiments

In this section, we will demonstrate the experimental results using these methods. The image sequence is a cube sequence consisting a magic cube on a round plate, as shown in Figure 4. The round plate is rotating counterclockwise with the magic cube. The displacement fields of this sequence are shown in Figure 5 without showing the zero displacement vectors. We can see there is a region of zero displacement vectors inside the round plate due to the little information contained by the plain white illumination of this region. Three major partitions and the boundary of the first two images are shown in Figure 6 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, rotating.

Computing the neural network by software is very time-consuming without special hardware assistance. We ran our algorithms on Sparc station 10 model 41 and it took 30 minutes to process each consecutive pair of images of size 512 by 512.

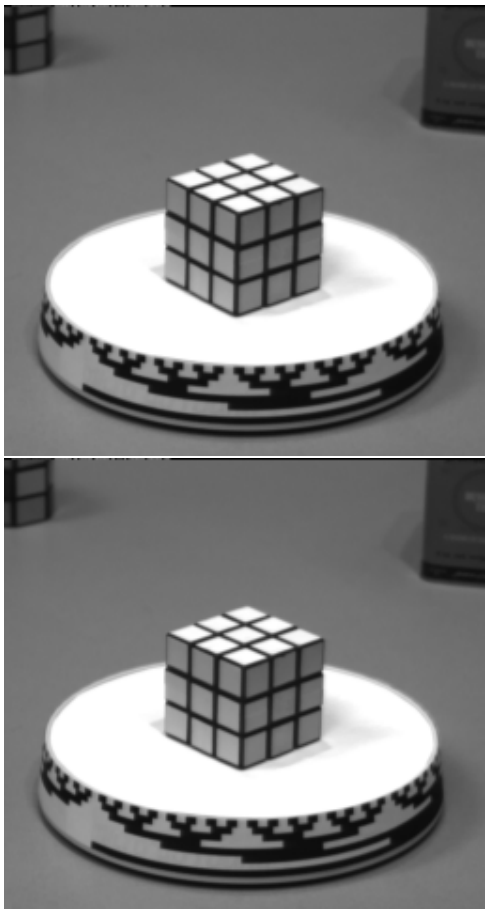


Figure 4: Consecutive images of rotating plate with a cube on it.

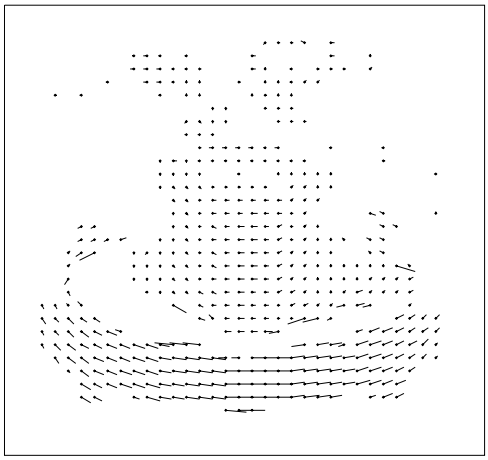
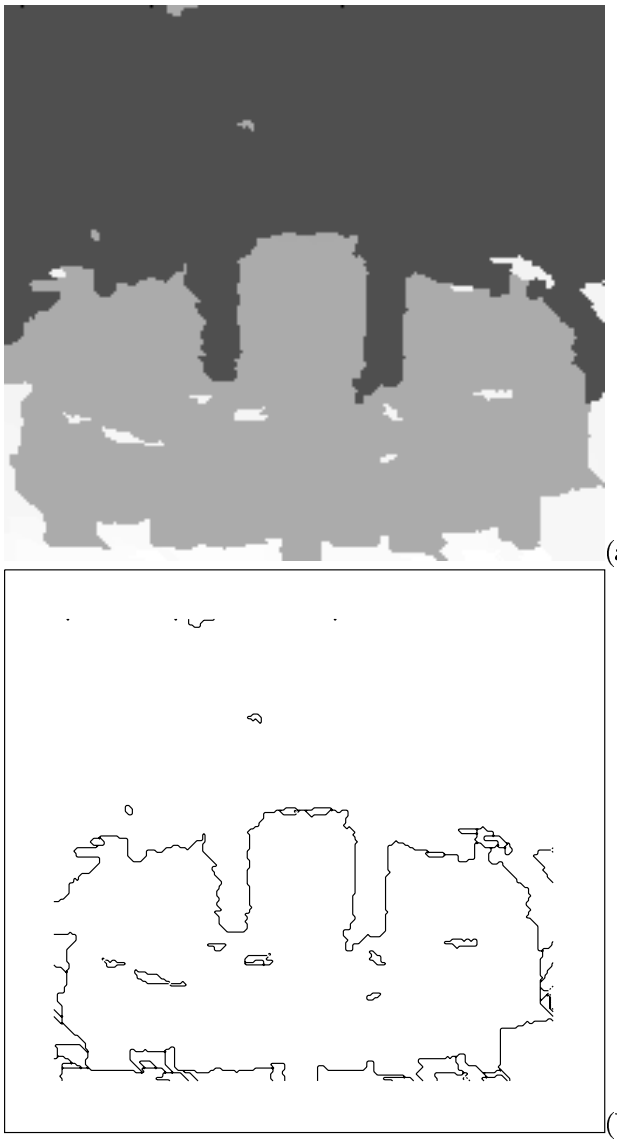


Figure 5: Displacement field between the consecutive images in Figure 4.



(a)

(b)

Figure 6: Segmentation result of the first two consecutive images produced by (a) region growing segmentation algorithm and (b) 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 displacement field and segment image simultaneously in an iterative manner. This neural network is trained by and maintains the segmentation information during the estimation of the displacement field. The enhanced block matching will use the segmentation information embedded in the neural network to estimate accurate displacement vectors. We applied our algorithms to various indoor and outdoor real-world image sequences; experiment results show that the displacement vectors and object boundaries are accurate even near the occluding boundary. Indeed, using synapse weight and displacement field from previous images as initial values improves displacement field estimation and image segmentation.

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

The problem of choosing the size of the matching block still remains unsolved. Some works use variable size to adapt to different environments. If the variation of intensities distributed in a region is large, the information contained in this region is large. Only a small size is needed to match two blocks well. On the contrary, there is little information contained in the plain intensity region. Thus, large block size is preferred in order to contain as much information as possible. One way to estimate the amount of information contained in a region is to use the concept of entropy. If $P(i)$ denotes the probability of the appearance of intensity i, the information contained by the intensity i is $\log \frac{1}{P(i)}$. The size of the matching blocks is chosen when the total information $\sum_{i \in block} \log \frac{1}{P(i)}$ is large enough.

Acknowledgement

This research work was supported by National Science Council of Taiwan, under Grants NSC 85-2212-E-002-077 and NSC 86-2212-E-002-025, by EeRise Corporation, by Institute of Information Industry under Grant III 84-TC-0422, and by Mechanical Industry Research Laboratories, Industrial Technology Research Institute under Grant MIRL 863K67BN2.

References

[1] M. S. Costa, R. M. Haralick, and L. G. Shapiro, (1990), "Optimal Affine Invariant Point Matching," *Proc. Intl. Conf. on Pattern Recognition*, Atlantic City, pp. 233–236.

[2] F. Dufaux and M. Kunt, (1992), "Multigrid Block Matching Motion Estimation with an Adaptive Local Mesh Refinement," *Proceedings of Visual Communications and Image Processing*, Boston, pp. 97–109.

[3] C. S. Fuh and P. Maragos, (1989), "Region-Based Optical Flow Estimation," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, San Diego, pp. 130–135.

[4] C. S. Fuh and P. Maragos, (1990), "Application of Mathematical Morphology to Motion Image Analysis," *Proc. Electronic Imaging EAST Conference*, Boston, pp. 261–264.

[5] C. S. Fuh and P. Maragos, (1991a), "Affine Models for Image Matching and Motion Detection," *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, Toronto, Canada, pp. 2409–2412.

[6] C. S. Fuh and P. Maragos, (1991b), "Motion Displacement Estimation Using an Affine Model for Image Matching," *Optical Engineering*, vol. 30, pp. 881–887.

[7] C. S. Fuh and P. Maragos, and L. Vincent, (1991c), "Region-Based Approaches to Visual Motion Correspondence," *Technical Report 91–98, Harvard Robotics Laboratory.*

[8] B. K. P. Horn and B. G. Schunck, (1981), "Determining Optical Flow," *Artificial Intelligence*, vol. 17, pp. 185–203.

[9] M. Irani, B. Rousso, and S. Peleg, (1994), "Computing Occluding and Transparent Motions," *International Journal of Computer Vision*, vol. 12, pp. 5–16.

[10] C. Schnörr, (1992), "Computation of Discontinuous Optical Flow by Domain Decomposition and Shape Optimization," *International Journal of Computer Vision*, vol. 8, pp. 153–165.

[11] V. Seferidis and M. Chanbari, (1992), "Generalized Block Matching Motion Estimation," *Proceedings of Visual Communications and Image Processing*, Boston, pp. 110–119.

[12] W. B. Thompson, K. M. Mutch, and V. A. Berzins, (1985), "Dynamic Occlusion Analysis in Optical Flow Fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7, pp. 374–383.