Multiscale Pyramid Representation: Stage I responses were generated for one spatial scale only. The single-scale computations of Stages II and III specified above were replicated for four different scales of increasing size. These were obtained by doubling the standard deviations of the elongated Gaussian filters from one scale to the other.

After the generation of each complex cell map, one for each scale, they were downsampled forming a pyramid structure [2]. Pyramid level 0 did not involve any downsampling (and thus had the same resolution as the original image), while levels 1–3 were downsampled by factors of 2, 4, and 8, respectively (in both the x and y axes). Hence, from each image, a four-level complex cell pyramid representation was constructed.

References

[1] R. Brunelli and T. Poggio, "Face recognition: Features versus templates," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, pp. 1042–1052, 1993.

[2] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code," *IEEE Trans. Commun.*, vol. 31, pp. 532–540, 1983.

[3] G. Carpenter, S. Grossberg, and D. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759–771, 1991.

[4] S. Edelman, N. Intrator, and T. Poggio, "Complex cells and object recognition," 1997. Available http://barus.physics.brown.edu/people/nin/research.html

[5] S. Grossberg and L. Pessoa, "Texture segregation, surface representation, and figure-ground separation," *Vision Res.*, vol. 38, pp. 2657–2684, 1998.

[6] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Trans. Neural Networks*, vol. 8, pp. 98–113, 1997.

[7] B. Mel, "SEEMORE: Combining color, shape, and texture histogramming in a neurally-inspired approach to visual object recognition," *Neural Computa.*, vol. 9, pp. 777–804, 1997.

[8] L. Pessoa, E. Mingolla, and H. Neumann, "A contrast- and luminance-driven multiscale network model of brightness perception," *Vision Res.*, vol. 35, pp. 2201–2223, 1995.

[9] R. Rao and D. Ballard, "An active vision architecture based on iconic representations," *Artificial Intell.*, vol. 78, pp. 461–505, 1995.

[10] M. Turk and A. Pentland, "Face recognition using eigenfaces," *J. Cognitive Neurosci.*, vol. 3, pp. 71–86, 1991.

An Adaptable Time-Delay Neural-Network Algorithm for Image Sequence Analysis

Christian Wöhler and Joachim K. Anlauf

***Abstract*—In this letter we present an algorithm based on a time-delay neural network with spatio-temporal receptive fields and adaptable time delays for image sequence analysis. Our main result is that tedious manual adaptation of the temporal size of the receptive fields can be avoided by employing a novel method to adapt the corresponding time delay and related network structure parameters during the training process.**

***Index Terms*—Adaptable time delay, image sequence analysis, receptive field, time-delay neural network.**

I. Introduction

In this letter we present a neural-network algorithm for image sequence analysis based on a time-delay neural network (TDNN) architecture with spatio-temporal receptive fields and adaptable time delays. The aim is to classify objects on temporal sequences of grayscale images and to estimate their motion behavior.

The general TDNN concept is well known from applications in the field of speech recognition (see, e.g., [7]). A temporal backpropagation algorithm for fixed-time delays is proposed in [8]. A TDNN architecture with spatio-temporal receptive fields for estimating image-pattern shape and motion has been proposed in [9]. It has successfully been applied to pedestrian recognition [4], [10] and detection of passing vehicles [11]. In this algorithm, however, the values of the time delays correspond to multiples of the time step between two subsequent images of the sequence. The temporal length of the receptive fields has to be determined by rather tedious manual adaption, given a certain training and test set.

In [1] the Tempo 2 algorithm is proposed that adapts time-delay values in a TDNN by supervised learning of input patterns defined at discrete time steps. In this algorithm, however, an input window of an appropriately smooth shape, e.g., Gaussian, is required; this function is not a result of the training process but has to be defined in a rather heuristic manner. The continuous-time temporal backpropagation algorithm [2] is restricted to continuous time signals such that it is not suitable for processing image sequences, where each image represents the scene at a discrete time step, e.g., multiples of 40 ms for PAL/CCIR video.

Our idea is to extend the TDNN concept described in [9] toward an adaptable time-delay neural network (ATDNN) architecture with spatio-temporal receptive fields to process discrete-time input signals, especially image sequences. The main advantage of the ATDNN algorithm is rather practical as it helps to avoid laborious handtuning of network structure parameters.

II. The ATDNN Algorithm

A. The Network Structure

The architecture of the ATDNN is shown in Fig. 1. For clarity, the two spatial directions x and y are compressed into one in this

Manuscript received March 29, 1999; revised July 2, 1999.

C. Wöhler is with DaimlerChrysler Research and Technology, D-89081 Ulm, Germany.

J. K. Anlauf is with the Rheinische Friedrich-Wilhelms-Universität, Institut für Informatik II, D-53117 Bonn, Germany.

Publisher Item Identifier S 1045-9227(99)08825-6.

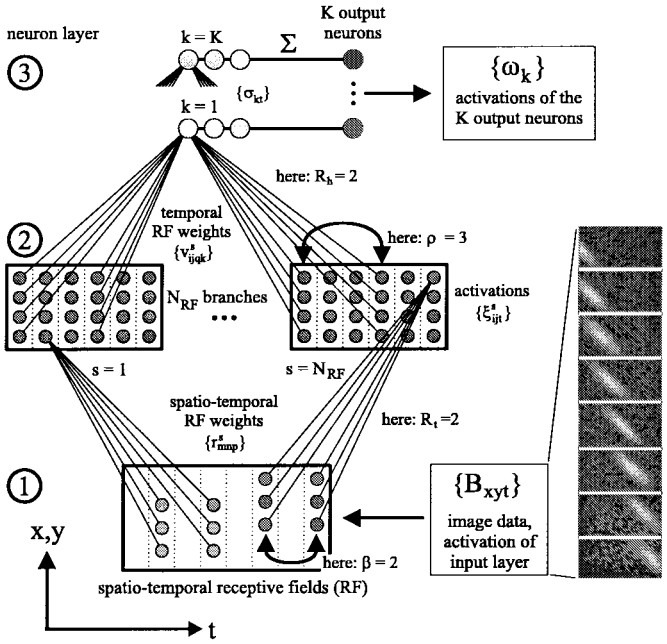

Fig. 1. Architecture of the ATDNN with spatio-temporal receptive fields. In this example, the time-delay parameter β between neuron layers 1 and 2 is $\beta = 2$, the one between layers 2 and 3 is $\rho = 3$. The size of the example input image sequence is $32 \times 16 \times 8$ pixels.

figure. The three-dimensional input layer consists of $S_x^{(1)} \times S_y^{(1)} \times S_t^{(1)}$ neurons with activations corresponding to the grayscale pixel values of the input images—the index (1) refers to the first neuron layer. In the ATDNN architecture, a neuron of a higher layer does not receive input from all neurons of the underlying layer but only from a limited region of it, which is called its *receptive field*. While classical TDNN's [5] only possess temporal receptive fields, this concept is extended toward spatio-temporal receptive fields in [9] and in the ATDNN architecture. This means that each neuron in the second layer can "see" a small three-dimensional part of the input layer only, covering a region of $R_x \times R_y \times T_{\text{eff}}^{(1)}$ pixels with $T_{\text{eff}}^{(1)} = 1 + (R_t - 1)\beta$ and R_t as the number of weight sets that belong to the same time slot, respectively, and β as the time-delay parameter. The distance of the centers of two neighboring receptive fields in the three different dimensions is given by D_x, D_y, and D_t, where we constantly take $D_t = 1$. The spatio-temporal receptive field concept takes into account the spatial and temporal locality of the object and motion features, which is in strong contrast to the standard multi layer perceptron (MLP) architecture the structure of which is invariant with respect to permutations of the input neurons.

The ATDNN is composed of N_{RF} different branches—as shown in Fig. 1, a "branch" consists of a three-dimensional layer of neurons (in Fig. 1, this is denoted by neuron layer 2) "looking at" the underlying input sequence by means of their receptive fields. As we follow the *shared weights* principle inside each branch the same set of weight factors $\{r_{mnp}^s\}$ is assigned to each layer 2 neuron of the branch numbered by s, with m and n as the spatial and p as the temporal coordinate inside the weight configuration of the receptive field. It is $1 \le s \le N_{\text{RF}}$, $1 \le m \le R_x$, $1 \le n \le R_y$, and $1 \le p \le R_t$. The weight configurations of the spatio-temporal receptive fields act as spatio-temporal filters, which means that in each network branch one distinct spatio-temporal feature is extracted from the input sequence.

In principle, the state of the ATDNN is defined only for integer values of the time-delay parameters ρ and β as shown in Fig. 1. As we will show later on how to take into account real-valued time-delay parameters, we will first define the activations for the example integer-valued combination $(\lceil\rho\rceil, \lceil\beta\rceil)$. In the following, $\lfloor x \rfloor$ denotes the largest integer that is less than or equal to the real number x, $\lceil x \rceil$ the smallest integer that is greater than or equal to x. Analogous expressions of the activation values of the network depending on the combinations $(\lfloor\rho\rfloor, \lceil\beta\rceil)$, $(\lceil\rho\rceil, \lfloor\beta\rfloor)$, and $(\lfloor\rho\rfloor, \lfloor\beta\rfloor)$, which we will need in the following section, are obtained in a straightforward manner.

The activation $\xi_{ijt}^s(\lceil\beta\rceil)$ of the layer 2 neuron at position (i, j, t) in branch s for the integer-valued time-delay parameter $\lceil\beta\rceil$ is calculated according to

$$\xi_{ijt}^s(\lceil\beta\rceil) = g_2\left(\sum_{p=1}^{R_t}\sum_{n=1}^{R_y}\sum_{m=1}^{R_x} r_{mnp}^s \times B_{D_x(i-1)+m, D_y(j-1)+n, t+(p-1)\lceil\beta\rceil} - \theta^s\right) \quad (1)$$

with $g_2(x) = \tanh(x)$ as a sigmoidal activation function, θ^s as the respective threshold value, and $\{B_{xyt}\}$ as the greyscale pixel values of the input image sequence. One should keep in mind that the dependence of the activations on the input data $\{B_{xyt}\}$ is omitted in (1) and the following equations.

The neurons of layers 2 and 3 are connected in a similar manner; in the spatial directions x and y, however, these higher neuron layers are fully connected, with the receptive field and shared weights structure remaining for the temporal direction t only. The weight configuration is denoted by $\{v_{ijqk}^s\}$. To each branch s and each output class k, $1 \le k \le K$, one such temporal receptive field is assigned, which produces the activations

$$\sigma_{kt}(\lceil\rho\rceil, \lceil\beta\rceil) = g_3\left(\sum_{s=1}^{N_{\text{RF}}}\sum_{q=1}^{R_h}\sum_{j=1}^{S_y^{(2)}}\sum_{i=1}^{S_x^{(2)}} v_{ijqk}^s \times \xi_{i,j,t+(q-1)\lceil\rho\rceil}^s(\lceil\beta\rceil)\right) \quad (2)$$

$g_3(x) = \tanh(x)$, in neuron layer 3. Each temporal receptive field thus covers a temporal range of $T_{\text{eff}}^{(2)} = 1 + (R_h - 1)\rho$ time steps in neuron layer 2 with R_h as the number of weight sets that belong to the same time slot, respectively, and ρ as the time-delay parameter. The K actual output neurons of the ATDNN then perform a classwise temporal integration of the activations of neuron layer 3, resulting in the output activations

$$\omega_k(\lceil\rho\rceil, \lceil\beta\rceil) = \frac{1}{S_t^{(3)}(\lceil\rho\rceil, \lceil\beta\rceil)} \sum_{t=1}^{S_t^{(3)}(\lceil\rho\rceil, \lceil\beta\rceil)} \sigma_{kt}(\lceil\rho\rceil, \lceil\beta\rceil) \quad (3)$$

with $S_t^{(3)}(\lceil\rho\rceil, \lceil\beta\rceil) = S_t^{(1)} - (R_t - 1)\lceil\beta\rceil - (R_h - 1)\lceil\rho\rceil$ as the number of layer 3 neurons per class. As the value $S_t^{(3)}(\lceil\rho\rceil, \lceil\beta\rceil)$ may change under variation of ρ or β, the sum in (3) is correspondingly normalized. The output neuron with the highest activation denotes the class to which the input pattern belongs.

B. The Training Algorithm

As our adaptation procedure for ρ and β is based on a gradient descent method, we need to define a network output $\omega_k(\rho, \beta)$ for

real-valued ρ and β, which is achieved by bilinear interpolation

$$\begin{aligned}\omega_k(\rho,\beta) = &\,\text{frac}(\rho)\text{frac}(\beta)\omega_k(\lceil\rho\rceil,\lceil\beta\rceil)\\ &+(1-\text{frac}(\rho))\text{frac}(\beta)\omega_k(\lfloor\rho\rfloor,\lceil\beta\rceil)\\ &+\text{frac}(\rho)(1-\text{frac}(\beta))\omega_k(\lceil\rho\rceil,\lfloor\beta\rfloor)\\ &+(1-\text{frac}(\rho))(1-\text{frac}(\beta))\omega_k(\lfloor\rho\rfloor,\lfloor\beta\rfloor)\quad\text{with}\\ \text{frac}(x) = &\,x-\lfloor x\rfloor. \end{aligned}\tag{4}$$

Further time-delay parameters may easily be introduced into this interpolation concept.

Interpolation (4) denotes a weighted sum of the output of four different network configurations in which the receptive fields contain different parts of the input image and neuron layer 2, respectively. This is only sensible if subsequent images of the input sequence are in a way similar, i.e., if temporal changes on the images appear continuously. This should always be the case if continuous scenes, e.g., displaying moving objects or distinct motion patterns are examined. It will be, however, impossible to employ the proposed interpolation technique for image sequences in which the content of an image of the sequence extremely differs within a time step.

The time-delay parameters ρ and β can be trained using an online gradient descent rule. For each training step one image sequence is chosen at random from the training set. The quadratic error measure for an input pattern of class c is defined by

$$\epsilon(\rho,\beta) = \tfrac{1}{2}\sum_{k=1}^{K}(\omega_k(\rho,\beta)-\tau_k)^2 \quad \text{with} \tag{5}$$

$$\tau_k = \begin{cases} A & \text{if } k = c \\ 0 & \text{if } k \neq c \end{cases} \tag{6}$$

where the constant A is slightly lower than one.

The learning rules for the time-delay parameters ρ and β are calculated by deriving the error measure $\epsilon(\rho,\beta)$ with respect to them, which leads to

$$\begin{aligned}\Delta\rho = -\eta_\rho &\sum_{k=1}^{K}(\omega_k(\rho,\beta)-\tau_k)\\ &\times[\text{frac}(\beta)(\omega_k(\lceil\rho\rceil,\lceil\beta\rceil)-\omega_k(\lfloor\rho\rfloor,\lceil\beta\rceil))\\ &\quad+(1-\text{frac}(\beta))(\omega_k(\lceil\rho\rceil,\lfloor\beta\rfloor)-\omega_k(\lfloor\rho\rfloor,\lfloor\beta\rfloor))]\end{aligned}\tag{7}$$

and

$$\begin{aligned}\Delta\beta = -\eta_\beta &\sum_{k=1}^{K}(\omega_k(\rho,\beta)-\tau_k)\\ &\times[\text{frac}(\rho)(\omega_k(\lceil\rho\rceil,\lceil\beta\rceil)-\omega_k(\lceil\rho\rceil,\lfloor\beta\rfloor))\\ &\quad+(1-\text{frac}(\rho))(\omega_k(\lfloor\rho\rfloor,\lceil\beta\rceil)-\omega_k(\lfloor\rho\rfloor,\lfloor\beta\rfloor))]\end{aligned}\tag{8}$$

with η_ρ and η_β as learning rates. These expressions follow from the relations

$$\frac{\partial}{\partial\rho}\,\text{frac}(\rho) = \frac{\partial}{\partial\beta}\,\text{frac}(\beta) = 1 \tag{9}$$

and

$$\frac{\partial\omega_k(\lceil\rho\rceil,\lceil\beta\rceil)}{\partial\rho} = \frac{\partial\omega_k(\lceil\rho\rceil,\lceil\beta\rceil)}{\partial\beta} = 0 \tag{10}$$

for all noninteger values of ρ and β. There are discontinuities for integer values of ρ and β which, however, turn out to be irrelevant in the training process. Variations of ρ and β during training may change the number of neurons in neuron layers 2 and 3 such that the network adaptively learns part of its own architecture from the examples contained in the training set. Moreover, during training, e.g., the value of β has to be clipped if it becomes inconsistent with the current value of ρ, and vice versa, e.g., if the temporal length of neuron layer 2 becomes smaller than the temporal range covered by the temporal receptive field between neuron layers 2 and 3.

For the weights $\{v^s_{ijqk}\}$ between neuron layers 2 and 3 as well as for the weights $\{r^s_{mnp}\}$ and thresholds $\{\theta^s\}$ between neuron layers 1 and 2 we then have backpropagation-like learning rules

$$\Delta v^s_{abck} = -\eta_v(\omega_k(\rho,\beta)-\tau_k)\frac{\partial\omega_k(\rho,\beta)}{\partial v^s_{abck}} \tag{11}$$

$$\Delta r^s_{abw} = -\eta_r(\omega_k(\rho,\beta)-\tau_k)\frac{\partial\omega_k(\rho,\beta)}{\partial r^s_{abw}} \tag{12}$$

$$\Delta\theta^s = -\eta_\theta(\omega_k(\rho,\beta)-\tau_k)\frac{\partial\omega_k(\rho,\beta)}{\partial\theta^s}. \tag{13}$$

The weight parameters and threshold values are initialized by small positive and negative random numbers of the order $\mathcal{O}(10^{-6})$ in order to break the symmetry (cf. [5]). As in [9], we choose the learning rate η_v inversely proportional to the fan-in of the neurons to which the corresponding weights are connected, i.e., $\eta_v = C_v/(S_x^{(2)}S_y^{(2)}R_h)$. The constant C_v has to be small enough such that convergence of the training error ϵ is achieved; we set $C_v = 2\times10^{-2}$. The learning rate η_r was set proportional to η_v as suggested in [5] for feedforward architectures. It was also set inversely proportional to the fan-in of a layer 2 neuron, i.e., $\eta_r = C_r\eta_v/(R_xR_yR_t)$ with $C_r = 5\times10^{-4}$. Finally we set $\eta_\theta = \eta_v$ and $\eta_\rho = \eta_\beta = C_\rho\eta_v$ with the constant $C_\rho = 25$.

This training algorithm is rather time-consuming; one training run of the kind described in Section III takes about 100 h on a standard Pentium II PC. For acceleration, however, we are currently working on a parallel implementation on the SIMD extension of the Pentium III processor. Due to the high computational complexity it is impossible on standard hardware to use sequences of full-resolution video frames as training examples. We are thus restricted to rather small images as regarded in Section III such that in practical applications, a preliminary segmentation stage that yields the approximate position of the object to be classified is still needed. Furthermore, for the sake of acceptable training times we will restrict the parameters R_t and R_h in Section III to the values $R_t = R_h = 2$.

III. Applications

A. Simple Synthetic Image Sequences

In this section we will examine the general properties of the ATDNN algorithm. We use the same synthetic image sequence data as in [9] in order to have full control of shape, speed, and initial position of the object. The size of a single image is $S_x^{(1)} = 32$ by $S_y^{(1)} = 16$ pixels, one sequence consists of $S_t^{(1)} = 8$ such images. They display Gaussian elliptic blobs in two different orientations, which results in two shape classes. These blobs move horizontally at five speeds $v_0 = -4, -2, 0, +2$, and $+4$ pixels per frame, where negative speeds indicate a motion from the right to the left and positive ones a motion from the left to the right. This gives $K = 10$ training classes altogether; a typical representative of each class is shown in Fig. 2. As we intend to regard "realistic" synthetic image sequences, size and speed of the blobs randomly change slightly from frame to frame such that the center positions jitter around the positions given by motion at constant speed. The average blob speed and shape in a sequence, however, are imposed to be exactly equal to the selected values. The starting point x_0 of the blob is chosen at random from the interval $[0, S_x^{(1)} - S_t^{(1)}|v_0|]$ for $v_0 \geq 0$, from $[S_t^{(1)}|v_0|, S_x^{(1)}]$ for $v_0 \leq 0$, such that the blob is still apparent on the last image of the sequence. This implies significant differences especially between the sequences belonging to the classes with low speeds $|v_0|$.

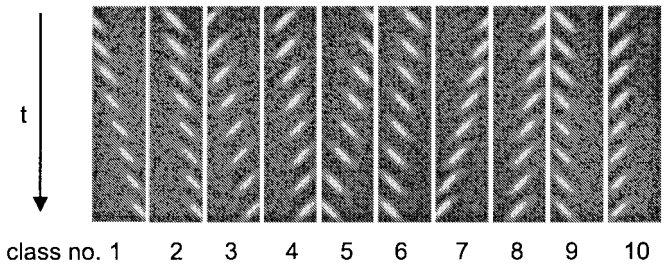

Fig. 2. Typical representatives of the $K = 10$ training classes. The frames of a sequence are displayed sequentially from top to bottom, respectively.

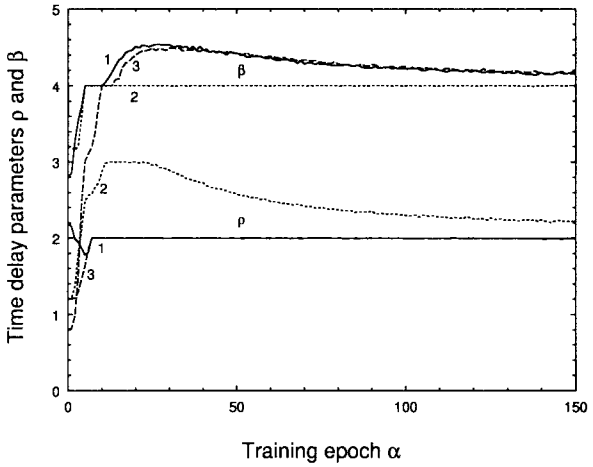

Fig. 3. Development of the time-delay parameters ρ and β during three training runs with $R_x = R_y = 7$, $R_t = 2$, $N_{\mathrm{RF}} = 2$, $R_h = 2$, $D_x = D_y = 4$, and different initial ρ and β values. Training run 1 is represented by solid curves, training run 2 by dotted curves, and training run 3 by dashed curves.

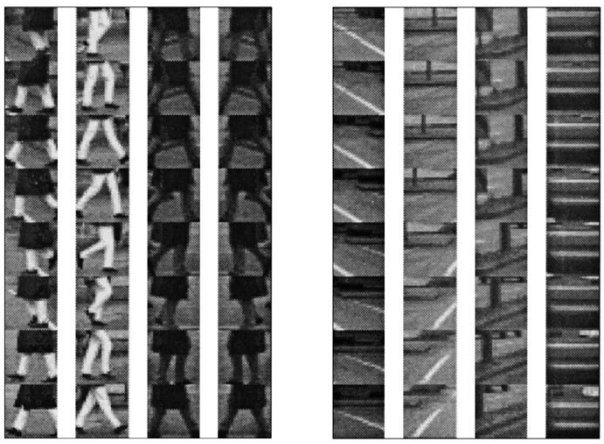

Fig. 4. Typical pedestrian (left) and garbage patterns (right).

The training set we use in our experiments consists of 5000 examples, 500 of each class. During training, we calculate the average network error on a test set of 1000 independent examples, 100 of each class. Due to the shared weights concept, the ATDNN possesses only 530 weight parameters to be adapted during training in the configuration that will be used in the following. This is significantly smaller than the number of training examples such that no overtraining effects could be observed. Consequently, we do not employ techniques like, e.g., cross-validation to determine the end of the training process. Instead, we terminate the training process when the decrease per training step of the error on the test set becomes

TABLE I
RESULTS OF PEDESTRIAN RECOGNITION BY ATDNN. THE FALSE POSITIVE RATE AMOUNTS TO 1%

run	$T_{\mathrm{eff}}^{(1)}$	$T_{\mathrm{eff}}^{(2)}$	error [%]
1	5.2	3.0	14.7
2	1.0	1.0	18.1
3	5.1	3.0	14.4
4	5.2	1.0	12.3
fixed time delays (cf. [9])	5.0	3.0	15.5

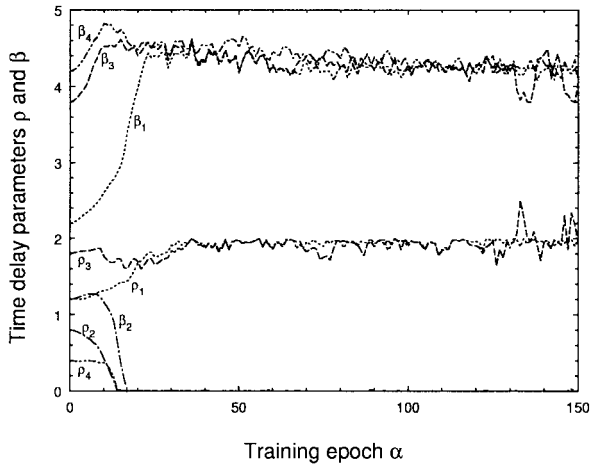

Fig. 5. Development of the time-delay parameters ρ and β during four training runs with $R_x = R_y = 9$, $R_t = 2$, $N_{\mathrm{RF}} = 2$, $R_h = 2$, $D_x = D_y = 5$, and different initial ρ and β values in the scenario of pedestrian recognition. The indices of ρ and β denote the training run, respectively.

smaller than a given threshold value, which was chosen to be of the order $\mathcal{O}(10^{-7})$. It is furthermore possible to reduce the size of the training set from 5000 toward 1000 examples without any noticeable change of the behavior of ρ and β and the error rates.

In [9], the network parameters $R_x = R_y = 7$, $R_t = 5$, $N_{\mathrm{RF}} = 2$, $R_h = 3$, and $D_x = D_y = 4$ have been determined for optimal blob eccentricity and speed measurement. In this model, all time-delay parameters are fixed to one, leading to receptive fields covering a temporal range of $T_{\mathrm{eff}}^{(1)} = 5$ time steps in the first and $T_{\mathrm{eff}}^{(2)} = 3$ in the second neuron layer, respectively. In Fig. 3, we present the results of three ATDNN training runs with the training data described above. We set $R_t = R_h = 2$ as the temporal ranges to be covered by the receptive fields are now given by the adaptable time-delay parameters ρ and β; the other network parameters remain unchanged. In all three training runs of Fig. 3 the time-delay parameters converge toward $\rho \approx 2$ and $\beta \approx 4$. Depending on the initial values ρ_0 and β_0 of the time-delay parameters, which are $\rho_0 = 2.2$, $\beta_0 = 2.8$ in run 1, $\rho_0 = 1.2$, $\beta_0 = 3.2$ in run 2, and $\rho_0 = 0.8$, $\beta_0 = 1.2$ in run 3, either time-delay parameter ρ (runs 1 and 3) or β (run 2) has to be clipped. In run 2, even both ρ and β are clipped temporarily for training epochs 10–25. Apart from clipping effects, Fig. 3 shows that the final ρ and β values do not significantly depend on their initial values ρ_0 and β_0.

The temporal ranges covered by the receptive fields and corresponding to the trained time-delay parameters are thus $T_{\mathrm{eff}}^{(1)} = 1 + (R_t - 1)\beta \approx 5$ and $T_{\mathrm{eff}}^{(2)} = 1 + (R_h - 1)\rho \approx 3$. This corresponds well with the temporal ranges determined in [9] obtained by manual adaptation.

We obtained an error rate on the test set of 1.1% for the TDNN configuration with fixed time-delay parameters taken from [9]. We

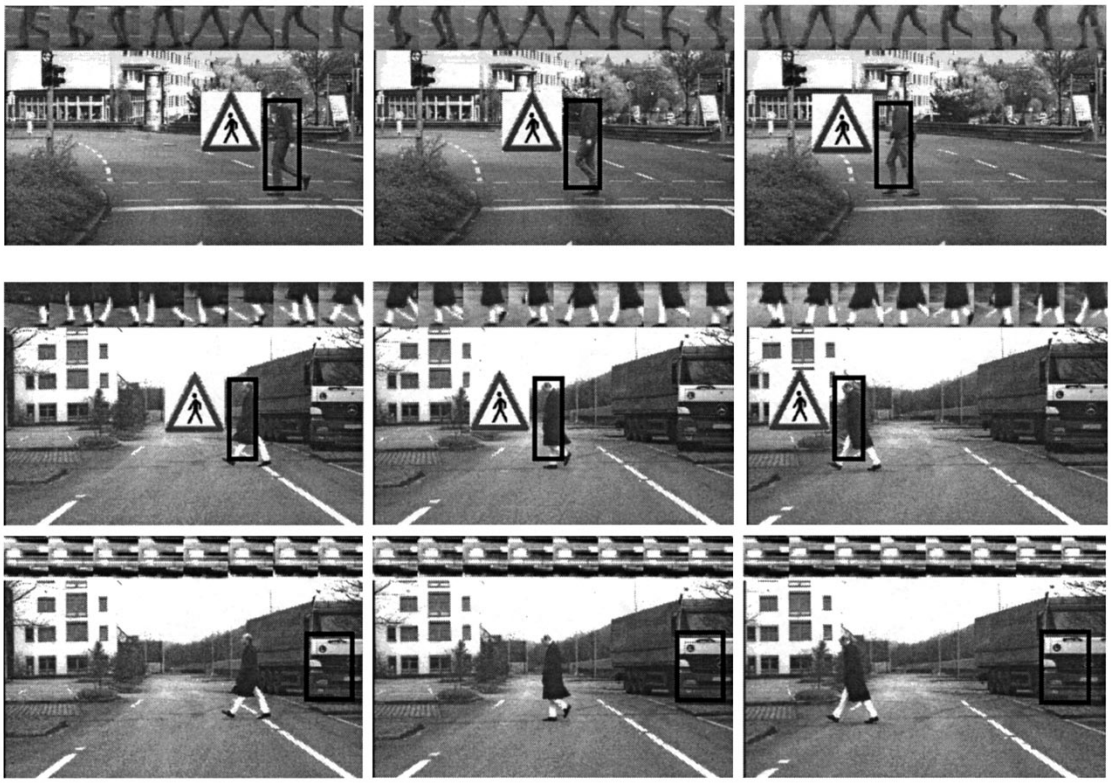

Fig. 6. Example image sequences displaying pedestrians in the urban traffic environment. Only the left stereo image of each acquired image pair is shown, respectively. The black bounding boxes have been determined by the stereo algorithm. In the upper part of the image, the input of the ATDNN, i.e., the scaled ROI on the current and on the seven preceeding images is shown. In the second sequence, the stereo algorithm detects two objects simultaneously, a pedestrian (above), and a truck front (below) correctly classified as a garbage pattern.

now have 1.2% for training run 1, 1.1% for run 2, and 1.2% for run 3. The errors always occur mainly between classes 9 and 10, the members of which are most variable with respect to the initial position of the blob.

B. Recognition of Pedestrians

In this section we will examine a rather complex application: the recognition of pedestrians on image sequences based on the characteristic criss-cross motion of their legs. The aim is to distinguish between pedestrian and nonpedestrian ("garbage") patterns. This module is part of the prototype of a driver assistance system for the inner city environment [3]. The objects in the scene are segmented by a real-time stereo vision algorithm described in detail in [3] and [10] which also works in the presence of a moving background. After detection of an object, we crop the lower half of the region of interest (ROI) delivered by the stereo algorithm, which contains the pedestrian's legs, and normalize it to a size of 24 $\times$ 24 pixels. Every third acquired pair of PAL/CCIR video images is evaluated by the stereo algorithm, which corresponds to a time step of 120 ms between two subsequent frames of the sequence. Eight subsequent normalized ROI's are combined into an image sequence covering a temporal range thus approximately corresponding to one walking step. After each stereo detection procedure, the batch of images is shifted backward by one image, discarding the "oldest" image while placing the new image at the first position.

In Fig. 4 typical representatives of the $K = 2$ training classes "pedestrian pattern" and "garbage pattern" are shown. Our training set contains 3926 pedestrian and 4426 garbage patterns, the test set consists of 1000 pedestrian and 1200 garbage patterns. In [10], the TDNN parameters $R_x = R_y = 9$, $R_t = 5$, $N_{\mathrm{RF}} = 2$, $R_h = 3$, and $D_x = D_y = 5$ have been determined for an optimal recognition performance. With the ATDNN algorithm, we performed four training runs altogether. We always set $R_t = R_h = 2$. Depending on the initial values of the time-delay parameters, we obtain three configurations of temporal extensions $T_{\mathrm{eff}}^{(1)}$ and $T_{\mathrm{eff}}^{(2)}$ of the receptive fields (cf., Table I), among which training runs 1 and 3 yield approximately the same configuration as obtained in [10] by manual adaptation. The behavior of the time-delay parameters, which are by definition positive numbers, during training run 2 and 4 is dominated by clipping effects as both ρ and β in run 2 and ρ in run 4 have to be limited correspondingly.

The network output is evaluated such that if $q\omega_1 > \omega_2$, the object is classified as a pedestrian. By varying the parameter q we adjust the system to the permissible false positive rate, i.e., the rate of garbage patterns erroneously classified as pedestrians. In the driver assistance system, 1% false positives are tolerable from the practical point of view. The corresponding recognition rates in Table I show that the performance of the ATDNN in all training runs except run 2 is slightly higher than that of the TDNN configuration obtained in [10]. These results mean in practice that in scenes of a length of some seconds, pedestrians are recognized with a high stability. There may be drop-outs if the contrast between the legs of the pedestrian and the background is too low; the recognition result, however, is strongly confirmed if a certain object detected by the stereo algorithm has been determined to represent a pedestrian at several subsequent time steps. Examples of combined pedestrian detection and recognition results are shown in Fig. 6.

IV. Summary and Conclusion

We have presented an algorithm for image sequence analysis based on a TDNN with spatio-temporal receptive fields and adaptable time delays. In this ATDNN algorithm, the time-delay parameters of the

network are learned from the training examples instead of being determined by manual adaptation. As the number of neurons in the higher neuron layers is automatically adapted accordingly, the network indeed learns part of its own architecture.

In experiments with synthetic image sequences we have shown that during training the ATDNN algorithm converges toward a configuration that is equivalent to the one previously obtained by manual adaptation in [9]. The corresponding error rates on the test set are nearly identical as well. The algorithm was furthermore successfully applied to the complex problem of pedestrian recognition.

The main advantage of the presented method of training the time-delay parameters is rather practical as it helps to avoid laborious handtuning of network structure parameters. The "optimal" time-delay parameters, however, are not necessarily integer numbers. It depends on the application and on the permissible computational cost whether to use directly the real values of ρ and β yielded by the ATDNN algorithm or the nearest integer values in order to finish the training process with fixed time delays as described in [9]. As it comes out in the pedestrian recognition scenario, using an ATDNN with trained noninteger time delays, i.e., the weighted average of a set of four ATDNN's with integer-valued time delays as given by (4), may yield a slightly higher performance than using one ATDNN with integer-valued time delays alone (cf. Table I).

In our algorithm the ATDNN output for real-valued time-delay parameters ρ and β is obtained by interpolation between several ATDNN outputs for integer-valued time-delay parameters. An alternative way might be to employ an interpolation technique for moving images, e.g., a corresponding straightforward extension of the RBF network algorithm proposed in [6], yielding interpolated pixel values for each desired intermediate, noninteger time value. The computation of the ATDNN output and its derivatives could then easily be based on these interpolated pixel values.

REFERENCES

[1] U. Bodenhausen and A. Waibel, "The tempo 2 algorithm: Adjusting time-delays by supervised learning," *Advances in Neural Information Processing Systems 3*, R. P. Lippmann, J. E. Moody, and D. S. Touretzky, Eds. San Mateo, CA: Morgan Kaufmann, 1991, pp. 155–161.

[2] S. P. Day and M. R. Davenport, "Continuous-time temporal backpropagation with adaptable time delays," *IEEE Trans. Neural Networks*, vol. 4, pp. 348–354, 1993.

[3] U. Franke, D. Gavrila, S. Görzig, F. Lindner, F. Paetzold, and C. Wöhler, "Autonomous driving goes downtown," *IEEE Intell. Syst.*, vol. 13, pp. 40–48, 1998.

[4] B. Heisele and C. Wöhler, "Motion-based recognition of pedestrians," in *Int. Conf. Pattern Recognition*, Brisbane, Australia, 1998, pp. 1325–1330.

[5] J. A. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Redwood City, CA: Addison-Wesley, 1991.

[6] T. Sigitani, Y. Iiguni, and H. Maeda, "Image interpolation for progressive transmission by using radial basis function networks," *IEEE Trans. Neural Networks*, vol. 10, pp. 381–390, 1999.

[7] A. Waibel, T. Hanazawa, G. Hinton, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 328–339, 1989.

[8] E. A. Wan, "Temporal backpropagation for FIR neural networks," in *IEEE Int. Joint Conf. Neural Networks*, San Diego, CA, 1990, pp. 575–580.

[9] C. Wöhler, and J. K. Anlauf, "A time-delay neural-network algorithm for estimating image-pattern shape and motion," *Image Vision Comput. J.*, vol. 17, pp. 281–294, 1999.

[10] C. Wöhler, J. K. Anlauf, T. Pörtner, and U. Franke, "A time-delay neural-network algorithm for real-time pedestrian recognition," in *IEEE Int. Conf. Intell. Vehicles*, Stuttgart, Germany, 1998, pp. 247–252.

[11] C. Wöhler, J. Schürmann, and J. K. Anlauf, "Segmentation-free detection of overtaking vehicles with a two-stage time-delay neural-network classifier," in *European Symp. Artificial Neural Networks*, Bruges, Belgium, 1999, pp. 301–306.

Shape Recovery from Shading by a New Neural-Based Reflectance Model

Siu-Yeung Cho and Tommy W. S. Chow

***Abstract*—In this paper, we present a neural-based reflectance model of which the physical parameters of the reflectivity under different lighting conditions are interpreted by the network weights. The idea of our method is to optimize a proper reflectance model by an effective learning algorithm and to recover the object surface by a simple shape from shading recursive algorithm with this resulting model. Experimental results, including synthetic and real images, were performed to demonstrate the performance of the proposed method for practical applications.**

***Index Terms*— Heuristic learning algorithm, neural-based reflectance model, shape from shading.**

I. INTRODUCTION

The success of shape from shading (SFS) depends on two major elements, namely the research of a better reflectance model and the development of an effective SFS algorithm. The importance of developing a better reflectance model lies in the fact that whether or not the models describe the surface shape accurately from the image intensity. Since the reflectance model is nonlinear in terms of the surface gradients, certain restrictive assumptions have to be made. First, surface reflection is generally assumed to be diffuse reflection. Second, a distant point light source is assumed to be known. Third, images are formed by orthographic projection. Based on these assumptions, a simple Lambertian model was established to describe the relationship between the surface normal and the light source direction [1]. As conventional SFS algorithms [2], [3] do not use generalized models, apparent distortions on reconstructed surfaces often result in many practical applications. In order to make the conventional SFS algorithms more practical sophisticated non-Lambertian models were studied by different researchers. Healy and Binford [4] employed the Torrance–Sparrow model assuming that a surface is composed of small randomly oriented mirror-like facets. A generalized reflectance model was proposed in [5] that depends on not only gradient variable but also variables of the position and the distance of light source. The surface depths are directly obtained but the computation scheme is highly complex. Most recently, a novel approach in using multilayer neural networks was proposed to tackle the SFS problem [6]. Under the framework of neural-networks learning, the network is capable of estimating the depth without going through complicated mathematical computation, but this approach is still under the restrictive condition of the Lambertian model. Obviously, this restrictive condition makes the algorithm not applicable to many practical situations in which knowledge on the illumination is not available. Perceivably, there is a need to establish a generalized model to relax some of these restrictive

Manuscript received April 2, 1999; revised August 14, 1999.

The authors are with the Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong.

Publisher Item Identifier S 1045-9227(99)09112-2.