A time delay neural network algorithm for estimating image-pattern shape and motion

Christian Wöhler a,*, Joachim K. Anlauf b

a Daimler-Benz Research Centre Ulm, Image Understanding Systems (FT3/AB), Wilhelm-Runge-Str. 11, D-89081 Ulm, Germany
b Rheinische Friedrich-Wilhelms-Universität Institut für Informatik II, Römerstr. 164, D-53117 Bonn, Germany

Received 10 September 1997; received in revised form 16 April 1998; accepted 20 April 1998

Abstract

In this paper we present a novel concept for simultaneous shape estimation and motion analysis based on a feed-forward TDNN architecture with adaptable spatio-temporal receptive fields. On synthetic image sequences displaying elliptic spots of different orientation moving horizontally across the scene at several speeds, this network simultaneously manages to classify the shapes correctly as well as to estimate their speed and motion direction, given various test sets and network parameter settings. A very interesting feature is the property that a network having learned a certain number of shape and motion classes is able to generalize to intermediate shapes and speeds it has never ‘seen’ during training by interpolating between the learned pattern classes. Moreover, the network turns out to be rather robust with respect to random deviations of the actual motion from the trained motion patterns. We furthermore apply the network successfully to a simple example of real-world data. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Shape; Motion; Image sequence; Time delay neural network; Receptive field

1. Introduction

Object recognition and motion analysis are the central topics in the domain of image understanding. Many approaches in the field of motion analysis concentrate on the extraction of the optical flow vector field from an image sequence with the purpose of obstacle detection. Once an obstacle has been detected, the classical approach is to track a characteristic object feature along successive images of the sequence. Most methods of this kind [4,12,19,26,28] involve an identification of the object features in each image by performing an exhaustive search with a template matching algorithm. This scene segmentation often relies on a priori knowledge about the objects to be tracked or other model assumptions.

In this paper, we suggest a novel concept for simultaneous shape estimation and motion analysis based on a time delay neural network (TDNN) with adaptable spatio-temporal receptive fields. TDNN architectures have been successfully used for speech recognition [3,5,18,21,23,24,27,30,31]; a related approach shows that it also has interesting properties concerning motion analysis on image sequences [1]. The method of receptive fields is used in the domain of object recognition on single images, especially for character recognition [2,7–11,20]. Biological aspects of the concept of spatiotemporal receptive fields with respect to motion detection have been described [17,32,33]. In most current applications [22], however, the structure of the receptive fields is imposed a priori instead of being trained in order to introduce model assumptions about the objects to be recognized into the learning algorithm, while we keep them fully adaptable in order to avoid the necessity to use a priori knowledge. Extending the adaptable receptive fields into the time dimension enables us to learn those spatio-temporal features which are optimally designed to describe the characteristics of both the shapes and the motion patterns being examined.

In Section 2 we give a brief description of the network architecture and the applied training algorithm; Section 3 then treats the general properties of the recognition process and the generalization capability of the network. In Section 4 we sum up the results and discuss possible applications of our method to real-world problems.
The receptive fields are adaptable in the sense that they produce one ‘filtered’ version of the input sequence per branch 

2.1. The network architecture

In Fig. 1 the architecture of the time delay network is shown. It receives the pixel values of a sequence of grey-scale images in the form of an xyt volume as an input. This can also be interpreted as feeding one image of the sequence after the other into the network, each delayed by a certain amount of time inside the network. The input image sequence is of the dimension $S^{(1)}_x \times S^{(1)}_y \times S^{(1)}_t$ pixels—the index (1) refers to the first network layer. This first, three-dimensional neuron layer contains the raw pixel values of the image sequence, denoted by $B_{xyt}$ with $1 \leq x \leq S^{(1)}_x$, $1 \leq y \leq S^{(1)}_y$, $1 \leq t \leq S^{(1)}_t$. Already the general feed-forward TDNN architecture is characterized by the fact that it is not fully connected; a neuron of a certain network layer does not receive input from all neurons of the underlying layer, but only from a limited region of it. This region is called the receptive field of the neuron. Furthermore, the TDNN 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 network layer 2) ‘looking at’ the underlying input sequence by means of their respective receptive fields. While classical TDNNs only possess temporal receptive fields, we extend this concept to spatio-temporal receptive fields. This means that a set of weight factors $(r_{mpn})_{ijq}$ is assigned to the layer 2 neuron at position $(ijq)$ in the branch numbered by $s$, where $1 \leq s \leq N_{RF}$, $m$ and $n$ as the spatial and $p$ as the temporal coordinate inside the weight configuration of the receptive field. As we follow the shared weight principle, the sets of weight factors with which the layer 2 neurons are connected inside a certain branch to their respective receptive fields are all identical such that they can simply be denoted by $(r_{mpn}^s)$, independent of the position of the neuron in layer 2. The weight coefficients of the receptive fields act as spatio-temporal filters; consequently, in each of the $N_{RF}$ network branches one distinct spatio-temporal feature is filtered from the input image sequence. The $N_{RF}$ different filtered versions of the input sequence are then displayed by the activations $\hat{g}_{ij}$ of the neurons in layer 2, which are given by

$$
\hat{g}^{ij}_{kl} = g_2 \left( \sum_{p=1}^{R_x} \sum_{m=1}^{R_y} \sum_{n=1}^{R_z} r_{mpn}^s B_{D_x(i-1)+m,D_y(j-1)+n,D_t(t-1)+p} - \theta^s \right),
$$

where $g_2(\cdot)$ is the activation function of the neurons in the second layer. We have chosen the standard sigmoid function $g_2(x) = \tanh(x)$, $\theta^s$ is the threshold value of the activation of the neurons in branch number $s$. The spatio-temporal receptive fields of neighbouring neurons of layer 2 are shifted by $D_x$, $D_y$, and $D_t$ input neurons in the respective directions; $R_x$, $R_y$, and $R_z$ denote the spatial and temporal extensions of the receptive fields.

The main reason for the use of receptive fields is that relevant spatio-temporal features of the image sequence presented to the network can be extracted while the number of independent weight parameters is kept quite small (see Section 2.2) compared to a fully connected architecture like the multilayer perceptron (MLP) [13,25]. Another reason is that the receptive fields are capable of taking into account the three-dimensionality of the image sequence as well as the locality of certain features, which is not the case for MLP-like architectures as their structure is invariant with respect to permutations of the input neurons.

The receptive fields are adaptable in the sense that they are adjusted during the training process, there is no more changing afterwards with respect to the continuous input after training. They therefore allow an extraction of arbitrary features which are specialized to the kind of shape and motion being regarded since the features to be extracted are learned from the training examples instead of being imposed a priori. Our concept thus helps, for example, to give an
answer to the rather ‘philosophical’ question whether an edge-based or a region-based approach is better suited to a certain problem: the network itself decides during the training process which features have to be chosen.

In order to obtain a spatial and temporal displacement robustness with respect to the detection of certain spatio-temporal features, the step sizes \(D_x, D_y, \) and \(D_t\) are usually chosen such that the receptive fields are overlapping, i.e. \(D_x < R_x, D_y < R_y, \) and \(D_t < R_t\) [11,14].

Each of the \(K\) output units of the network represents one training class. In this context, a training class means a certain shape moving in a certain manner, i.e. one class may, for example, be a circle moving slowly from the right to the left, another class the same circle displaying a fast top-down motion, another class a fast top-down moving rectangle. To each output class and also to each branch of the neurons in the second layer in the respective direction. It is

\[
S_k^{(2)} = 1 + [(S_k^{(1)} - R_k)/D_k],
\]

where \(S_k^{(2)} = 1 + [(S_k^{(1)} - R_k)/D_k]\).

\[
S_k^{(1)} = 1 + [(S_k^{(0)} - R_k)/D_k],
\]

\[
S_k^{(0)} = 1 + [(S_k^{(0)} - R_k)/D_k],
\]

\[
(3)
\]

\[
(4)
\]

with \(S_k^{(3)} = 1 + S_k^{(2)} - R_k\). Assuming that the network has been trained by presenting examples belonging to \(K\) different classes, the output neuron with the highest activation \(\sigma_{t\ell}\) indicates that the input pattern is most similar to the trained patterns of class \(k, \) where \(1 \leq k \leq K\) (Section 2.2). The values of the other neurons, however, may as well contain important information about the input pattern, as we will show in Section 3.3.

In Section 3.3.2, it will be shown how the choice of the network design parameters such as the size of the spatio-temporal receptive fields and the number of network branches can be justified with respect to a specific shape and motion recognition problem.

One could now argue that given a variety of shapes, speeds, and motion directions to be recognized, the number \(K\) of output classes will quickly become very large. Our intended application domain of the TDNN, however, is the recognition of traffic participants, possibly after a rough pre-segmentation, e.g. by stereo vision. Thus only a small number of ‘objects’ have to be distinguished (e.g. cars, pedestrians) which only move in the horizontal direction. Besides, as it will be shown in Section 3.3, the TDNN is capable of determining speed and shape by interpolating between very few trained reference values. The number of output classes will thus remain moderate, i.e. of the order \(O(10),\) in the intended scenario.

The number \(n_w\) of independent weight parameters is given by the expression

\[
n_w = R_xR_yR_tN_{RF_{spatio-temporal RF}} + N_{RF_{thresholds}} + S_k^{(2)}S_k^{(2)}KN_{RF_{temporal RF}}.\]

\[
(5)
\]

In most considered network configurations, we have \(n_w\) of the order \(O(1000)\) for the training examples described in Section 3.1. The number of weight parameters is thus smaller than the number of pixel values of a single input image sequence, which is 4096; this would not be possible with a fully connected standard MLP-like architecture.

The complexity of the presented TDNN algorithm, i.e.
the number \( F \) of floating-point operations needed to perform the recognition of an input pattern, is given by

\[
F = F_{\text{lop}} + F_{\text{tanh}}
\]

where \( F_{\text{lop}} = R_c R_c R_c N_{RF} S_n^{(2)} S_n^{(2)} S_n^{(2)} + S_n^{(2)} S_n^{(2)} R_c N_{RF} S_n^{(2)}, \)
\[
F_{\text{tanh}} = z(N_{RF} S_n^{(2)} S_n^{(2)} S_n^{(2)} + S_n^{(2)}).
\]

Here, \( F_{\text{lop}} \) is the number of additions and multiplications to be performed in the weighted sums, whereas \( F_{\text{tanh}} \) denotes the number of operations for evaluating the nonlinear activation functions; \( z \) is the number of floating point operations needed to perform one evaluation of the function \( \tanh(x) \). \( F \) is always linear in \( N_{RF} \). For \( D_1 = D_2 = R_c / 2, D_i = 1 \), and \( K \) of the order \( O(10) \), we have \( S_n^{(2)} S_n^{(2)} R_c N_{RF} S_n^{(2)} \ll R_c R_c R_c N_{RF} S_n^{(2)} S_n^{(2)} S_n^{(2)} \), \( F \) is therefore independent of \( K \) in this domain, becoming linear in \( K \) for \( K \) of the order \( O(100) \). Furthermore, by regarding the corresponding CPU times it turned out that \( F_{\text{tanh}} \ll F_{\text{lop}} \) in the examined configurations. This shows that despite the high dimension of the input patterns, the complexity of the TDNN algorithm remains moderate and does not 'explode' with a rising number \( K \) of training classes.

2.2. The training algorithm

To keep our model simple, we use a gradient descent learning rule to train the network. This is equivalent to the well-known backpropagation algorithm [13,25]. The error measure \( \epsilon \) that denotes the deviations of the values \( \omega_k \) of the output neurons from the desired outputs \( \tau_k \) is chosen to be

\[
\epsilon = \frac{1}{2} \sum_{k=1}^{K} (\omega_k - \tau_k)^2 = \frac{1}{2} \sum_{k=1}^{K} \Delta^2_k \text{ with } \Delta_k = \omega_k - \tau_k,
\]

which is just the usual quadratic error measure used in most neural network training scenarios. For a presented pattern of class \( c \), the desired output is set to \( \tau_c = \mathbf{A}, \tau_k = 0 \) for \( k \neq c \) with \( A \) close to 1.

In each training step, one image sequence is chosen at random from the training set consisting of \( P \) different examples. The weights are adapted according to the gradient of the error measure \( \epsilon \) with respect to the weight parameters \( r_{\text{map}} \) and \( v_{\text{ijk}} \), and thresholds \( \theta^a \).

Eq. (7) together with Eqs. (2) and (4) yields the variation \( \Delta r_{\text{map}} \) of weight parameter \( r_{\text{map}} \) in a straightforward manner:

\[
\Delta r_{\text{map}} = -\eta_r \frac{\partial \epsilon}{\partial r_{\text{map}}} = -\eta_r \Delta_k \sum_{i=1}^{S_n^{(1)}} g_j^j(h_{ik}) \xi_{\text{map}}^c + c - 1.
\]

In Ref. [34], it is recommended to choose the learning rate inversely proportional to the fan-in of the neurons to which the corresponding weights are connected, i.e. \( \eta_r = C_r / (S_n^{(2)} S_n^{(2)} R_c) \). The constant \( C_r \) has to be chosen small enough such that convergence of the training error \( \epsilon \) is achieved; a value of \( C_r \) of the order \( O(10^{-2}) \) turned out to be a good choice.

An analogous expression follows for the variation \( \Delta v_{\text{ijk}} \) of receptive field weight parameter \( v_{\text{ijk}} \):

\[
\Delta v_{\text{ijk}} = -\eta_v \frac{\partial \epsilon}{\partial v_{\text{ijk}}} = -\eta_v \sum_{k=1}^{K} \Delta_k \sum_{i=1}^{P} g_j^j(h_{ik}) \sum_{q=1}^{Q} \sum_{j=1}^{J} \sum_{l=1}^{L} v_{\text{ijk}} \times g_j^j(h_{ijq} - \theta^s) B_{D_1(i-1)+a.D_2(j-1)+b.D_3(l-1)+c.D_4(q-1)+d}.
\]

The learning rate \( \eta_r \) was chosen to be proportional to the weight rate \( \eta_v \) of the above weight layer as suggested in Ref. [34] for MLP-like architectures. It was also set inversely proportional to the fan-in of a layer 2 neuron, i.e. \( \eta_r = C_v / (R_c R_c R_c) \) with \( C_v \) of the order \( O(10^{-3}) \). Setting \( \eta_r = 0 \) would in principle allow to force the receptive fields to extract distinct features by initializing the weight parameters \( r_{\text{map}} \) in a corresponding manner (e.g. as Sobel edge detectors), leaving them unchanged during the training process.

For the threshold \( \theta^s \), one obtains the expression

\[
\Delta \theta^s = -\eta_v \frac{\partial \epsilon}{\partial \theta^s} = \eta_v \sum_{k=1}^{K} \Delta_k \sum_{i=1}^{P} g_j^j(h_{ik}) \sum_{q=1}^{Q} \sum_{j=1}^{J} \sum_{l=1}^{L} v_{\text{ijk}} g_j^j(h_{ijq} - \theta^s),
\]

where \( \eta_v = \eta_r \) turns out to be a sensible choice for the corresponding learning rate \( \eta_r \). The weight parameters and threshold values are initialized by small positive and negative random numbers of the order \( O(10^{-5}) \) in order to break the symmetry of the weights [13].

3. The properties of the shape estimation and motion analysis process

3.1. The training sequences

We first concentrate on synthetic image data as this helps to understand the general properties of the network in a quite illustrative manner. An example of a real-world application of our method will be given in Section 3.4. The size of a single image is \( S_n^{(1)} = 32 \) by \( S_n^{(1)} = 16 \) pixels, one sequence consists of \( S_n^{(1)} = 8 \) such images. The sequences in the training set contain gaussian elliptic spots in two different orientations, which means there are two shape classes. The corresponding widths of the gaussian profiles are \( \sigma_{\text{map}} \), eight pixels in the direction of the major axis and...
3.2. The role of the different neuron layers in the recognition process

In this section, we will look right into the TDNN in order to explain which task is carried out by which neuron layer by regarding the activations of different neuron layers under presentation of certain input image sequences as well as the learned weight configurations. For clarity, we restrict ourselves to two shape and two speed classes, i.e. \( K = 4 \) training classes altogether, as shown in Fig. 3 on the upper left. This reduced set of training classes corresponds to classes \( k = 1 \ldots 4 \) of Fig. 2.

As long as the receptive fields are of a small extension in space and time (e.g. \( R_x = R_y = R_t = 3 \) pixels, Fig. 3a), they essentially perform a weighted sum of, in this case, \( R_t = 3 \) subsequent input images at each time step, leading to activation patterns in neuron layer 2 in which both the shape and the motion are still visible. Deviations of the shape positions from a strictly linear motion are smoothed. The temporal receptive fields between neuron layer 2 and 3 of length \( R_t = 3 \) are therefore formed as detectors for both shape and motion, as it is shown in Fig. 3a: they extract the speed information from three subsequent filtered frames in neuron layer 2. The spatio-temporal receptive fields perform a preprocessing step adapted to the special sort of pattern being regarded, while the shape and speed recognition itself is carried out in the higher neuron layers. In the plots in Fig. 3 of the temporal receptive field weights \( v_{ijk}^{t} \) between neuron layer 2 and 3, one can clearly see a ‘prototype’ of the shape and motion pattern of each class which they detect in the activation patterns of neuron layer 2, respectively, especially when they are large in the spatial directions. In this context, ‘prototype’ means that the weight configuration \( v_{ijk}^{t} \) corresponding to class \( k \) in network branch \( s \) looks similar to a typical activation pattern \( \{ r_{ijk} \} \) which is produced in neuron layer 2 under presentation of a pattern belonging to class \( k \). Although different initial positions \( x_0 \) of the shape lead to different activation patterns in neuron layer 2, the use of temporal receptive fields between neuron layers 2 and 3 which are shorter than the filtered input sequence leads to a high displacement tolerance. The initial position \( x_0 \) of the shape and therefore the time at which it appears at a certain position barely affects the correct recognition of its speed.

If the temporal size of the spatio-temporal receptive fields becomes larger and comparable to the length of the sequence (e.g. \( R_x = R_y = 5 \), \( R_t = 7 \) or \( R_t = 7 \) pixels), they actually measure the speed, especially if the network parameters are chosen such that there are too few neurons in the second layer to detect it, as is the case in Fig. 3b, where the original image sequence is transformed into a two-dimensional activation pattern \( \{ r_{ijk} \} \) with no temporal extension in the second neuron layer; these activation patterns, however, are clearly distinguishable for shapes of different speed. The weights \( v_{ijk}^{t} \) between neuron layer 2 and 3 then extract the shape...
Spatially large receptive fields $r_{sp}$, e.g. of size $R_x = R_y = 15$, $R_t = 3$ pixels detect complete shapes, indicating as well the direction of motion, as it becomes obvious in Fig. 3c. They produce a high activation in neuron layer 2 at the presence of one of the learned shapes. The second neuron layer $j_{st}$ then consists of a sequence of three activation patterns of 3 by 1 neurons, respectively; the motion of the shape is transformed into a ‘moving’ activation of a single neuron at a certain speed displayed by the $j_{st}$ values, which is detected by the weights $v_{sijk}$ between neuron layer 2 and 3. In this parameter setting, they fully connect neuron layer 2 and 3, as otherwise they would be too short in the temporal direction to perform a speed estimation. The shape estimation is thus completely carried out by the spatio-temporal receptive fields, while the higher neuron layers analyse the motion.

In Fig. 4 the development of this kind of large spatio-temporal receptive field during the training process is illustrated. It is clearly visible how the receptive fields first specialize on circular blobs; after that, they become more and more able to distinguish between the two shape classes. This symmetry breaking is caused by the random fluctuations of the initial weight values.

Fig. 3. Visualization of the inner state of the TDNN after presenting the examples shown in Fig. 2. In each case, the network has $N_{RF} = 2$ branches. The left half of the figure represents the first, and the right half the second network branch. On the upper left, representatives of the four image sequence classes are shown for comparison. The first line of each of the parts (a), (b), and (c) shows the weights of the spatio-temporal receptive fields, sliced in the $xy$ plane, i.e. time slot by time slot, presented side by side. In the next line, the activation patterns of the second neuron layer, i.e. examples of input sequences of each class ‘filtered’ by the spatio-temporal receptive fields, are displayed. In the third line, the purely temporal receptive fields between neuron layer 2 and 3 are shown. A medium grey corresponds to zero, darker grey levels to negative, lighter grey levels to positive values. The scale factor is adapted to the dynamic range of the respective part of the figure. For further explanation, see text.

Fig. 4. Development of the spatio-temporal receptive fields during the training process for a network with large spatio-temporal receptive fields. The training set contains 2000 examples belonging to the $K = 4$ classes of Fig. 3. As the network possesses two branches $s$, there are also two different sets of spatio-temporal receptive fields. The learning rates are $\eta_v = 6.10 \times 10^{-5}$, $\eta_r = 4.88 \times 10^{-9}$, and $\eta_t = 3.30 \times 10^{-7}$, the receptive field configuration is displayed after (from the left) 4, 10, 15, 40, and 99 epochs, with one epoch corresponding to 2000 examples presented to the network. The sliced representation of the receptive fields is the same as in Fig. 3. It is clearly visible how the receptive fields first extract circular blobs, their shape gradually adapting to the orientations of the two trained objects.

and recognize its speed from those two-dimensional activation patterns.

Spatially large receptive fields $r_{sp}$, e.g. of size $R_x = R_y = 15$, $R_t = 3$ pixels detect complete shapes, indicating as well the direction of motion, as it becomes obvious in Fig. 3c. They produce a high activation in neuron layer 2 at the presence of one of the learned shapes. The second neuron layer $j_{st}$ then consists of a sequence of three activation patterns of 3 by 1 neurons, respectively; the motion of the shape is transformed into a ‘moving’ activation of a single neuron at a certain speed displayed by the $j_{st}$ values, which is detected by the weights $v_{sijk}$ between neuron layer 2 and 3. In this parameter setting, they fully connect neuron layer 2 and 3, as otherwise they would be too short in the temporal direction to perform a speed estimation. The shape estimation is thus completely carried out by the spatio-temporal receptive fields, while the higher neuron layers analyse the motion.

In Fig. 4 the development of this kind of large spatio-temporal receptive field during the training process is illustrated. It is clearly visible how the receptive fields first specialize on circular blobs; after that, they become more and more able to distinguish between the two shape classes. This symmetry breaking is caused by the random fluctuations of the initial weight values.

Fig. 5. Activation of several output neurons with respect to patterns displaying an eccentricity of $e_l = \pm 2$, moving at speeds $v_0 = -4.0, -3.5, -3.0, \ldots, +4.0$. Typical representatives are shown above. The training classes are explained in Fig. 2. The test set contains 340 examples, 20 of each speed value. The activation profiles are centred at the speeds represented by the respective output neurons, decreasing for slightly deviating speed and reaching zero for the speeds represented by the ‘neighbouring’ neurons, respectively.
The output of the network does not reveal the time at which a certain pattern appeared in the input image sequence. The temporal integration of the activations of neuron layer 3 that the output neurons perform leads to a high activation of output neuron $k$ if one of the learned shape and motion patterns has been apparent at any time in the image sequence. To restore the time information of the occurrence of the corresponding pattern, regard the activations $s_{kt}$ of neuron layer 3: if activation $s_{kt}$ is maximal within class $k$, the pattern appeared most distinctly at the time $t_0$.

In all our experiments, the threshold values $\theta^+$ of the activations of the second layer neurons turned out to be limited to the order $O(10^{-2})$ at all stages of the training process; all obtained results would therefore stay the same if the thresholds were fixed to zero.

3.3. Generalization and robustness

In this section we will examine the generalization of the network, i.e. its response to pattern classes not included in the training set, in order to see if its output still contains useful information. We now return to the complete set of $K = 10$ training classes mentioned in Section 3.1.

3.3.1. Qualitative behaviour of the network with respect to unknown pattern types

Fig. 5 displays the response of the five output neurons representing one of the pattern shapes contained in the training set, moving at 17 equidistant speeds $v_0$ from the interval $-4 < v_0 < +4$ pixels per frame. The activation is maximum for the speed of the patterns represented by the respective training class, decreasing symmetrically for higher or lower speeds. The actual pattern speed can be interpolated by means of the activation values; in Fig. 5 this is, for example, suggested by the fact that the activations of output neurons $k = 5$ (trained speed: $v_0^{(5)} = -4$) and $k = 6$ (trained speed: $v_0^{(6)} = -2$) are almost identical for the average speed $(v_0^{(5)} + v_0^{(6)})/2 = -3$.

A similar behaviour of the activations of the output neurons can be observed when regarding elliptic shapes with various eccentricities different from those contained in the training set (Fig. 6). To obtain a symmetric eccentricity measure, we define

$$e_1 = \log_2 \frac{a}{b}$$

(11)

with $a$ and $b$ as the principal axes of the elliptic shape (Fig. 6). Training classes $k = 1, 2, 5, 6, 9$ thus belong to $e_1 = +2$, classes $k = 3, 4, 7, 8, 10$ to $e_1 = -2$ (see Fig. 2). In Fig. 6, the pattern speed is $v_0 = +4$, i.e. one of the trained speeds. The network achieves interpolation between the different pattern shapes by adjusting the activation levels of output neurons 1 and 3 according to the similarity of the presented shape to the trained ones.

These results show qualitatively that the network is able
to generalize from quite a sparse training set to examples it has never ‘seen’ during training. As a general observation, it can be stated that the width of the activation profile adapts to the width of the intervals between the trained speeds and eccentricities such that, for example, the activation of neuron \( k \) representing patterns of a certain speed \( v_0 \) is maximum for the speed \( v_0 = v_0(1) \); decreasing to about half the maximum value for \( v_0 = (v_0(1) + v_0(0)) / 2 \), where \( v_0(1) \) is the ‘neighbouring’ trained speed, reaching zero for \( v_0 = v_0(0) \). The activations show the same behaviour for eccentricities deviating from the trained values. In all examined examples, the ‘tuning curves’ \( \omega_s(v_0) \) and \( \omega_s(e_l) \) displayed gaussian-like shapes. The average width of the \( v_0 \)-dependent profiles in Fig. 5 obtained by a gaussian fit, i.e. their full width at half maximum, is \( \sigma_{\text{FWHM}} = 1.82 \pm 0.14 \); this is approximately the difference between two neighbouring trained speed values, which corresponds to 2. The widths of the tuning curves belonging to \( v_0 = 0 \) are systematically slightly larger than those belonging to \( v_0 \neq 0 \); the situation shown in Fig. 5 is nearly identical for any other examined network configuration. This behaviour is analogous for the case of intermediate eccentricities.

To interpret the network output with respect to examples displaying pattern speeds and eccentricities not contained in the training set, we performed some quantitative measurements of arbitrary combinations of intermediate shapes and speeds, which will be described in the following section.

### 3.3.2. Quantitative measurements of intermediate speeds and shapes as a base for network design decisions

In this section, we develop a method for measuring quantitatively intermediate speeds and shapes not contained in the training set. The results will be used for obtaining network parameter configurations which are adapted to perform a specific task, i.e. in our example, speed or shape estimation.

The network has been trained such that if an example pattern of training class \( c \) displaying one of the trained speeds and eccentricities is presented to it, output neuron \( c \) shows an activation level of \( \omega_c = 1 \), while we have \( \omega_k = 0 \) for \( k \neq c \). If, however, an example pattern displaying a speed and/or shape which is not contained in the training set is presented, several neurons will be significantly activated. To determine the neurons the activations of which are to be used for the measurement, we computed for a test set containing \( P_0 = 1000 \) patterns belonging to the \( k = 10 \) training classes the standard deviation \( \sigma_k \) of the activation of those output neurons that should be equal to zero in the ideal case:

\[
\sigma_k = \sqrt{\frac{1}{P_0(K-1)} \sum_{\mu=1}^{P_0} \sum_{k \neq \mu} \omega_k^2((B^{\mu,c}_0))}.
\] (12)

The example pattern \( (B^{\mu,c}_0) \) is the \( \mu \)th example in the test set and belongs to training class \( c \). The value of \( \sigma_k \) has to be determined separately for every network configuration. For the examined network parameter settings, we obtained values of \( \sigma_0 \) between 0.08 and 0.13.

The significant output neurons that contain information about the properties of the presented example are those whose activations significantly exceed the ‘noise level’ \( \sigma_0 \). We therefore regard the \( S \) output neurons with activations \( \omega_s \geq 3\sigma_0, 1 \leq s \leq S \), as relevant for the measurement of pattern speed and shape. Typical values for \( S \) lie between 1 and 4. As an illustration, activations resulting from a test set of ‘intermediate’ examples with \( v_0 = 3 \) and \( e_l = 0 \) are shown in Fig. 7. The measured speed or shape value is then defined as a weighted average of the trained values represented by the significant output neurons:

\[
v_0^{\text{meas}} = \sum_{s=1}^{S} \omega_s v_s^{(s)} / \sum_{s=1}^{S} \omega_s, \quad e_l^{\text{meas}} = \sum_{s=1}^{S} \omega_s e_s^{(s)} / \sum_{s=1}^{S} \omega_s.
\] (13)

In the rarely observed case \( S = 0 \), i.e. no activation larger than \( 3\sigma_0 \), we simply take the speed and shape given by the neuron with the highest activation.

To keep the computation simple, we use the activations themselves as weights in the sum. Eq. (13) is rather heuristic as the activations \( \omega_s \) are actually no probabilities or probability densities. It mirrors, however, the assumption that the similarity of the speed and shape values of the observed example to those values represented by a certain output neuron increases with the activation of this neuron.
at the results obtained in the following, Eq. (13) appears to be capable of yielding an acceptable interpolation beha-

For a systematic examination, we regard the speed interval $2 \leq v_0 \leq 4$ at a stepsize of $Dv_0 \approx 0.1$ and
the shape interval $2 \leq e_l \leq 2$ at a stepsize of $De_l \approx 0.2$, resulting in 1701 different speed/shape combinations.
For each combination, speed and shape were measured for $P_1 = 100$ examples, as well as the average quadratic devia-
tions from the respective true values $v_0^{\text{true}}$ and $e_l^{\text{true}}$. We defined these quadratic errors as

\[ \sigma^2_{v_0} = \frac{1}{P_1} \sum_{\mu=1}^{P_1} (v_0^{\text{meas},\mu} - v_0^{\text{true}})^2, \]

\[ \sigma^2_{e_l} = \frac{1}{P_1} \sum_{\mu=1}^{P_1} (e_l^{\text{meas},\mu} - e_l^{\text{true}})^2. \]  

(14)

In Fig. 8, an example result of the measurement of speed and shape over the complete range is given. The deviation pattern is symmetric with respect to $v_0 = 0$ and $e_l = 0$ as the corresponding distribution of examples in the training set is symmetric as well. The error of the speed measurement is minimal for the trained speeds $v_0 = -4, -2, 0, +2$, and $+4$ as near these interpolation points the activations of only those neurons representing the corresponding speed are large while all other activations lie well below $3\sigma_{v_0}$. Moreover, the error is systematically larger for low values of $|v_0|$ (i.e. $|v_0| < 2$) than for larger ones (i.e. $2 < |v_0| < 4$) due to the high variation of the starting point $x_0$ of the pattern in the sequence at low speeds $|v_0|$ (see Section 3.1). The same is true for the error of the eccentricity measurement; in addition to that, for $|v_0| > 2$ we have a small error not only for the trained eccentricities $e_l = \pm 2$ but also for $e_l = 0$, i.e. circular shapes. For $|v_0| < 2$, high eccentricities are systematically estimated too low. The orientation of the elliptic shape, i.e. the sign of $e_l^{\text{true}}$, is nevertheless correctly determined even for $v_0 = 0$, where the $x$ position of the shape is completely arbitrary. As a whole, this example shows that the network achieves an estimation of arbitrary combinations of pattern speed and shape within the trained intervals.

Fig. 9 illustrates the robustness of the network output with
respect to variations of the initial shape position \(x_0\) at low speeds \(|v_0|\). The examples shown are correctly classified with most network configurations taken into account, especially the one mentioned in Figs. 5–7, despite the significantly different \(x_0\) value within each of the triples, respectively. In this context ‘correctly classified’ means that for an example of training class \(c\) we have \(\omega_{c} = \max \{ \{ \omega_{k} \}_{k=1}^{x} \}\). In the special cases \(k = 9\) and \(k = 10\), i.e. \(v_0 = 0\), both activations \(\omega_{0}\) and \(\omega_{10}\) are significant, and even though the ‘correct’ activation is always larger, the contribution of the second-largest activation leads to a slight underestimation of the eccentricity, as is shown in Fig. 8.

If there are, however, patterns presented to the network displaying speeds and/or eccentricities outside the trained range, i.e. \(|v_0| > 4\) or \(|e| > 2\), the activation of the output neurons representing the extreme trained values \(|v_0| = 4\) or \(|e| = 2\) decrease well below 1 while those of other neurons slightly rise above zero such that wrong speed and eccentricity values are calculated which lie inside the trained range. This makes clear that the network is limited to interpolation within the trained range; the extreme trained values of \(|v_0|\) and \(|e|\) therefore have to limit the complete range of speeds and eccentricities to be recognized by the network.

To obtain a measure for the performance of the network with respect to speed or shape, we define the overall deviations

\[
\sigma_{v_{0}}^{\text{ave}} = \sqrt{\langle \sigma_{v_{0}}^{2} \rangle}, \quad \sigma_{e_{i}}^{\text{ave}} = \sqrt{\langle \sigma_{e_{i}}^{2} \rangle},
\]

where \(\langle \ldots \rangle\) denotes the average over the complete regarded speed and eccentricity interval. These overall deviations can be used to justify certain design decisions, i.e. as a base for finding the set of free network parameters like receptive field size, number of branches, etc., that is optimally suited for a certain recognition problem, which in our example consists of a determination of either pattern speed or shape. In principle, there are eight network parameters that can be chosen freely. However, for reasons of computation time, it was not possible to vary all of them over a wide range in order to determine the position of optimal shape and speed measurement in this eight-dimensional parameter space as one training process takes about 72 hours on a Sun Ultra-Sparc workstation, so first of all we restricted ourselves to quadratic spatio-temporal receptive fields, i.e. \(R_e = R_s\) and \(D_s = D_t\). This can be justified by the fact that the extension of the trained shapes is identical in the \(x\) and the \(y\) dimension.

Furthermore, we determined that the performance of speed and shape recognition is generally better for \(N_{RF} = 2\) network branches than for \(N_{RF} = 1\), but does not increase any more for \(N_{RF} = 3\). We therefore equipped the network constantly with \(N_{RF} = 2\) branches.

The length \(R_s\) of the temporal receptive field in layer 2 was chosen to be quite small, \(R_s = 3\), to ensure a maximum robustness with respect to variations of the initial position—larger temporal receptive field would possibly prevent a large correlation level between the corresponding weights and the filtered versions of the image sequence in neuron layer 2 in the case of either very small or very large initial position values \(x_0\) even though pattern speed and shape of the input example are ‘correct’. To make this assumption plausible, we have looked at the example \(R_s = R_s = 9, R_s = 5, N_{RF} = 2, D_s = D_t = 5, D_s = 1\) with \(R_s = 3\) and \(R_s = 4\), observing that the performance decreases with increasing extension \(R_s\) of the temporal receptive field.

The mutual overlap of the spatio-temporal receptive fields in layer 1 should in principle be large in order to extract as much information as possible from the input image sequence. Less overlap just means less information in neuron layer 2. This assumption was in fact taken into account for the overlap \(D_s\) in the temporal direction which was chosen to be \(D_s = 1\) as the image sequence is quite short compared to its extension in the \(x\) and \(y\) direction. In the spatial dimensions, however, we set the respective overlap to \(D_s = D_t = R_s/2\), which seems to be enough to extract the most relevant information without requiring too much computation time. Note that \(D_s = D_t = 1\) would produce training cycles of several months on standard hardware. For the special case of spatio-temporal receptive fields of size \(R_s = R_s = 9, R_s = 5\) with spatial overlaps \(D_s = D_t = 5\) we proved this assumption by showing that choosing \(D_s = D_t = 3\) gave no performance improvement in terms of speed and shape measurement error, while with \(D_s = D_t = 9\) (no overlap) the performance decreased strongly.

The remaining free network parameters are thus the sizes of the spatio-temporal receptive fields in all directions, i.e. \(R_s, R_e, \) and \(R_s\). In order to obtain their optimal values with respect to speed and shape determination, we calculated \(\sigma_{v_{0}}^{\text{ave}}\) and \(\sigma_{e_{i}}^{\text{ave}}\) for various settings. The results are shown in Fig. 10. The first result is that the performance is best for \(R_s = 5\) for both speed and shape measurement. For the example of \(R_s = R_s = 9\), we have shown that further enlargement of the receptive fields in the temporal dimension \((R_s = 7)\) is of no use concerning the network performance. The optimal spatial extension is thus \(7 \times 7 \times 5\) neurons for speed and \(3 \times 3 \times 5\) neurons for shape measurement. Of course, these results are specific for the problem to be solved by the TDNN and the kind of examples presented.

---

Fig. 9. Example triples of several training classes representing low speeds, i.e. \(v_0 = \pm 2\) and \(v_0 = 0\), each triple displaying significantly different initial positions \(x_0\). The example image sequences are taken from the test set. Despite the strong positional variance, all examples shown are classified correctly.
to it during the training process. In the case of hard network decisions, e.g. a differentiation between an ‘object’ and a ‘garbage’ class, an optimization criterion might be to maximize the correct recognition rate of the ‘object’ class, given a certain permissible rate of incorrectly classified ‘garbage’ patterns.

3.3.3. Robustness of the network

We furthermore regarded the robustness of the network output with respect to a more or less significant random jitter of the shape position with respect to the position determined by assuming a motion at constant speed. The jittered shape position \((x(t),y(t))\) is thus given by

\[
x(t) = x_0 + v_0 t + U_x[-J,J],
\]

\[
y(t) = S_y(t)/2 + U_y[-J,J]
\]

with \(U_x[\ldots]\) and \(U_y[\ldots]\) being random numbers from the respective interval. This can as well be regarded as adding noise to the constant speed. As in Section 3.1, the average speed in a sequence is imposed to be exactly equal to \(v_0\). For all examples of the previous sections, a jitter parameter \(J_{\text{train}} = J_{\text{test}} = 0.5\) was chosen for the training as well as for the test set. We will now examine the dependence of

![Figure 10](image1.png)

**Fig. 10.** Results of the network performance with respect to the determination of pattern speed and shape for several values of the extensions \(R_x\) and \(R_t\) of the spatio-temporal receptive fields. Curve 1: error \(\sigma_{\text{ave}}^\text{vel}\) of the speed measurement, \(R_x = R_t = 3\). Curve 2: error \(\sigma_{\text{ave}}^\text{ecc}\) of the eccentricity measurement, \(R_x = 3\). Curve 3: \(\sigma_{\text{ave}}^\text{vel}\) for \(R_x = R_t = 5\). Curve 4: \(\sigma_{\text{ave}}^\text{ecc}\) for \(R_x = R_t = 5\). In each case, a clear minimum of the respective error value is visible. The optimal temporal extension of the spatio-temporal receptive fields is \(7 \times 7 \times 5\) neurons with respect to speed and \(3 \times 3 \times 5\) neurons with respect to eccentricity determination.

![Figure 11](image2.png)

**Fig. 11.** Above: typical representatives of the \(K = 10\) shape and motion classes in a test set with strongly \((J_{\text{test}} = 4.0)\) jittered shapes. Below: dependence of the error \(\sigma_{\text{ave}}^\text{vel}\) of speed estimation on the strength \(J_{\text{test}}\) of the jittering parameter of the test set. The calculations were performed for two different training sets with \(J_{\text{test}} = 0.5\) and 4.0 for the network configurations listed in the table.
the network error on test sets with higher values of \( J_{\text{test}} \) using two different training sets with \( J_{\text{train}} = 0.5 \) and 4.0 for three different network configurations. In Fig. 11, the error \( \sigma_{\text{ave}} \) of speed estimation depending on the jitter parameter \( J_{\text{test}} \) of the test set is depicted for three different network configurations and the jitter parameters \( J_{\text{train}} \) of the two different training sets. The performance of networks trained with \( J_{\text{train}} = 0.5 \) (curves 1, 2, 3) is better than that of networks trained with \( J_{\text{train}} = 4 \) (curves 4, 5, 6) for small jitter parameters of the test set, i.e. \( J_{\text{test}} < 2 \), becoming comparable for \( J_{\text{train}} = 2 \). Up to \( J_{\text{test}} = 4 \), the performance of networks trained with \( J_{\text{train}} = 4 \) is barely decreasing compared to those trained with \( J_{\text{train}} = 0.5 \). The best performance in the domain of strongly jittered test patterns (\( J_{\text{test}} > 3 \)) is obtained with small spatio-temporal receptive fields and thus a large second neuron layer (Fig. 11, curve 4).

The corresponding behaviour of the error \( \sigma_{\text{ave}} \) of eccentricity estimation is very similar; the best performance is as well yielded by the network configuration 4, strongly decreasing for larger spatio-temporal receptive fields.

The network performance with respect to speed estimation barely decreases within a wide range of jitter parameters \( J_{\text{test}} \). This shows the remarkable robustness of the network for shapes that jitter strongly around the position determined by motion at constant speed.

### 3.4. Application to real world data—a toy example

In order to show that the network is capable of learning real-world image data, we produced image sequences displaying two objects—a teapot and a bowl—moving horizontally on a conveyor belt at five different speeds
\(v_0 = -4, -2, 0, +2, \) and \(+4\) pixels per frame (Fig. 12). The training scenario is the same as with the synthetic examples in the foregoing sections. We originally recorded 40 examples of each class at a resolution of \(360 \times 180\) pixels per frame. We then cropped at random a region of \(320 \times 160\) pixels from each image and downscaled it by a factor of 10; this random cropping was repeated 10 times for each image sequence, resulting in 4000 different image sequences of size \(32 \times 16\) pixels and 8 frames per sequence. We used 3000 examples for the training set and 1000 for the test set. The network successfully separates the \(K = 10\) shape and motion classes (Fig. 12); the average rate of correct recognition is 96.4\% on the test set. This toy example makes it plausible that the network is capable of performing shape and speed recognition not only on synthetically produced image data.

4. Summary and conclusion

In this paper, we have presented a novel concept for simultaneous shape estimation and motion analysis based on a TDNN architecture with adaptable spatio-temporal receptive fields. This feed-forward network is trained by a gradient descent training algorithm. On synthetic images displaying gaussian spots of different orientations moving horizontally at several speeds in both directions, this network manages to classify the shape correctly as well as to estimate its speed, given various test sets and network parameter settings. Especially, a network having learned \(K\) shape and motion classes allows it to interpolate shapes and speeds it has never ‘seen’ during training based on the mutual relations of the activations displayed by the \(K\) output neurons. The average errors of this interpolation procedure are used as a measure for the network performance with respect to the specific tasks of speed and shape recognition. They serve as a base for a determination of the parameter configurations of the TDNN which are optimally designed for speed and shape estimation. Moreover, the network turns out to be rather robust with respect to shapes that jitter strongly around the position determined by motion at constant speed. We furthermore applied the network successfully to examples of simple real-world data.

The application this method has been designed for is the real-time or near-real-time object recognition and motion analysis in urban traffic scenes, which is one of the principal domains being worked on at the Daimler-Benz Department of Image Understanding. It is intended to apply this method to tasks like warning the driver of approaching cars, pedestrians or bicyclists or recognizing overtaking cars as well as estimating their speed. Eventually, a preliminary detection step, e.g. based on a technique like stereo vision [6] or the colour cluster flow [15], will be useful to limit the image regions possibly containing moving objects of interest. First attempts of pedestrian recognition using the algorithm described in this paper are outlined in Ref. [16].

It is possible to run this method in real-time when regarding image sequences of sizes of the same order of magnitude as those examined in this paper; the time needed to classify an \(32 \times 16 \times 8\) image sequence is between approximately 10 and 30 ms on a Sun Ultra-Sparc workstation, depending on the network parameter setting. Moreover, as the TDNN architecture primarily involves correlation-like computations it is specially suitable for an implementation on fast DSP boards or even in parallel hardware [29]. Finally, it should be possible to enhance the efficiency of the training process by applying more sophisticated training algorithms.

References


B.W. Mel, SEEMORE: combining color, shape, and texture histogramming in a neurally-inspired approach to visual object recognition, Neural Computation, in press.


