Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 70 (2007) 2552–2560
NEUROCOMPUTING
www.elsevier.com/locate/neucom

A self-organizing map of sigma–pi units

Cornelius Weber[a,*], Stefan Wermter[b]

[a]Frankfurt Institute for Advanced Studies, Johann Wolfgang Goethe University, 60438 Frankfurt am Main, Germany
[b]Hybrid Intelligent Systems, School of Computing and Technology, University of Sunderland, UK

Received 28 September 2005; received in revised form 3 April 2006; accepted 10 May 2006
Communicated by E.W. Lang
Available online 20 October 2006

Abstract

By frame of reference transformations, an input variable in one coordinate system is transformed into an output variable in a different coordinate system depending on another input variable. If the variables are represented as neural population codes, then a sigma–pi network is a natural way of coding this transformation. By multiplying two inputs it detects coactivations of input units, and by summing over the multiplied inputs, one output unit can respond invariantly to different combinations of coactivated input units. Here, we present a sigma–pi network and a learning algorithm by which the output representation self-organizes to form a topographic map. This network solves the frame of reference transformation problem by unsupervised learning.

Keywords: Frame of reference transformations; Population coding; Invariances

The sensorimotor control by the human body is subject to the complexity of the body geometry. Information extracted from the world by sensors like vision needs to be transformed into a motor-relevant representation. The position of an object as it is perceived by vision on the retina, for example, cannot be directly used to control the arm and the hand for grasping. Instead, the direction in which the head and the eyes are facing needs to be considered, in order to infer an object's position in a body-centered frame of reference which is more suitable for the control of the hand.

In mammals, the posterior parietal cortex (PPC), which lies at a strategic position between the visual and motor cortex, represents objects in different frames of reference. "The encoding of information referenced to the retina (eye-centered) but modulated by eyeposition, called a gain field representation, has proven to be very common throughout parietal and occipital cortex" [14]. PPC neurons have mostly multimodal responses and allow the PPC to carry out computations which transform the location of targets from one frame of reference to another [4,5]. For example, if we progress along the adjacent cortical areas LIP, VIP and area 5 of the macaque, neurons encode locations retinotopically in eye-centered coordinates in the LIP, in eye-and also in head-centered coordinates in the VIP [7] and in hand-centered coordinates in area 5 (in addition to eye-centered representations) [3].

Modeling frame of reference transformations: Neural network models of such frame of reference transformations code the transformed variables by population codes, in order to take into account that in the brain usually many neurons with varying properties respond to a given stimulus. Fig. 1 depicts a situation where three variables μ_x, μ_y and μ_z are each coded on an array of neurons as population vectors $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$, respectively (bold font denotes vector). In the population vectors, neurons are activated under a Gaussian-shaped hill of activation where the position of the hill denotes the variable to be coded. In order to code a two-dimensional vector one would take a two-dimensional sheet of neurons.

The task of the frame of reference transformation is, if μ_x and μ_y are given, to infer μ_z. For example, μ_x may be the location, in the horizontal plane, of an object of interest as

*Corresponding author. Tel.: +49 69 79847536; fax: +49 69 79847611.
E-mail addresses: c.weber@fias.uni-frankfurt.de (C. Weber), stefan.wermter@sunderland.ac.uk (S. Wermter).
URL: http://www.his.sunderland.ac.uk.

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2006.05.014

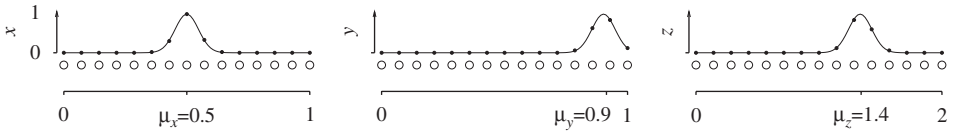

Fig. 1. Taking the sum of two variables using population codes. *Below*: the three variables and their relation $\mu_x + \mu_y = \mu_z$. The extended range of the coordinate system for μ_z accounts for its larger range. *Above*: the neural population codes $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$. Each variable is represented as a neural activation pattern: the dots in the Gaussian-shaped curves represent the activations of the underlying neurons (open circles). The position μ of a hill of neural activation carries the information about the corresponding variable.

perceived visually on the retina, μ_y may be the horizontal position of the eyes and μ_z may then be the head-centered horizontal location of the object. In this case we have the linear relation $\mu_z = \mu_x + \mu_y$. This transformation becomes non-linear between the neural population codes for μ_x, μ_y and μ_z, as we can see in Fig. 1: the neural activations $\boldsymbol{x}$ and $\boldsymbol{y}$ themselves do not simply add up to $\boldsymbol{z}$.

A standard way to achieve this transformation is first to generate the outer product of $\boldsymbol{x}$ and $\boldsymbol{y}$ on a large additional hidden layer of neurons. Then, a projection from this layer to the output layer implements the function $\boldsymbol{z} = f(\boldsymbol{x}, \boldsymbol{y})$ [6,12,16]. The use of the hidden layer as the outer product of input layers allows the hidden code and the connections between the layers to be constructed using algebraic transformations. Another approach has been to learn the transformation by an auto-associative attractor network, using the principle of pattern completion [17]. Without a hidden layer, it exploits the simplicity of the data: the position of the input activation hills causes a broad bias on the output, and a fully recurrent connectivity confines the output to a narrow hill of activation. This network was trained supervised in the sense that the training data was given to all three layers.

The question which we seek to answer is: How can a frame of reference transformation, or rather the layer representing the transformed variable, self-organize? In the example above, if $\boldsymbol{x}$ codes for a retinal location of an object, we may assume this to be represented topographically in some area of the visual system. Likewise, the position of the eyes may be read from the eye muscles and coded in $\boldsymbol{y}$. However, the position of the object in a head-centered frame of reference is not given by any sensors. Instead, it can be inferred from $\boldsymbol{x}$ and $\boldsymbol{y}$, so the representation $\boldsymbol{z}$ should self-organize given only these two inputs. For learning, we can exploit a characteristic property of the data: $\boldsymbol{z}$ is more constant than $\boldsymbol{x}$ and $\boldsymbol{y}$. For example, if only the gaze direction is changed by a mere eye-movement, then altered $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{y}}$ still correspond to a constant head-centered object position coded by $\boldsymbol{z}$.

A topographic mapping between the head-centered object position and its representation $\boldsymbol{z}$ would be desirable for clarity. Furthermore, two PPC areas in humans (possibly homologues of monkey LIP) have recently been reported to represent delayed saccadic motor responses in a topographic fashion, demonstrating that "topographic maps tile the cortex continuously from V1 well into PPC" [13].

Self-organization is important in the context of the cerebral cortex, in order to explain structures in a multitude of cortical areas. In the context of frame of reference transformations, an adaptive scheme explains how sensory-motor coordination can cope with adaptations during evolution and ontogenesis. In contrary to previous work that tackles frame of reference transformations via geometrically constructed weights [6,12,16] or via supervised learning [17] we show for the first time that this problem can be solved by unsupervised, or self-organized, learning.

Related work on sigma-pi units: Sigma–pi units have received early interest for explaining features of cortical physiology more easily than classical units (e.g. [11]). As a proposed task area, multiplicative attentional control can dynamically route information from a region of interest within the visual field to a higher area [1]. In our notation, a visual signal $\boldsymbol{x}$ coding object features would be routed by a control signal $\boldsymbol{y}$ conveying the retinal object position to a signal $\boldsymbol{z}$ that retains the features of $\boldsymbol{x}$ but is centered on its layer. Thereby, $\boldsymbol{z}$ is invariant to shifts of $\boldsymbol{x}$, while $\boldsymbol{y}$ codes for the shift.

Another idea for sigma–pi weights encoding invariances can be realized if both inputs that are multiplied originate from the same visual input layer [2]. Shift invariance is expressed in a weight constraint which enforces a weight connecting one pixel pair to be the same as a weight to another pixel pair, if the constellations of both pairs are similar except for a shift in position. While the information about the original object position is lost, a welcome effect is that this "weight sharing" constraint reduces the number of network parameters. A biologically more acceptable way of keeping the number of weights low is a framework that lets so-called cluster weights grow according to need [15].

In another sigma–pi framework, two sets of activations develop jointly from natural image patches as input [8]. One coding features ("content") and the other coding transformation ("style"). The number of hidden units to be activated depends on the number of features present in the model, which requires an iterative procedure to obtain the hidden code, and a number of parameters to be set to control the activation statistics. In contrast, our scenario of estimating one position of an object of interest in a given frame of reference allows the setting of the hidden code to a Gaussian-shaped hill of activation where only its position has to be estimated from the data. Therefore, we can use a simple non-iterative algorithm based on the SOM

algorithm [9] which also allows for larger displacements of the stimuli.

To our knowledge, this is the first formation of a self-organizing map by sigma–pi units. Given the growing interest in sigma–pi networks due to the increase of computational power and given the popularity of SOMs due to their simplicity, we expect frequent use of their combination in the future.

Outline of the paper: In Section 1 we will present the architecture and in Section 2 we will describe the learning algorithm. Section 3 will show results of coordinate transformations by the model, including a transformation that is non-linear in the coded variables and transformations along two-dimensional maps. We also show that after learning with blob-like activations, the network can also convey more complex activation patterns. In Section 4 we will discuss the results.

1. Model architecture

The network architecture is schematically displayed in Fig. 2. With the number of units in the corresponding layers being N_x, N_y and N_z, the total number of possible sigma–pi connections $\{w_{ijk}\}$ is $N_x \cdot N_y \cdot N_z$. This is in the order of, but still less than, the case of the basis function networks [6,12,16]. A unit i on the top layer is activated by the input vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ via the relation

$$a_i = \sum_{j,k} w_{ijk} x_j y_k. \tag{1}$$

Hence, a sigma–pi weight w_{ijk} is effective, if unit j of the input vector $\boldsymbol{x}$ is coactivated with unit k of $\boldsymbol{y}$, implementing a logical AND relation. Fig. 2(b) demonstrates how such a sigma–pi network can solve a relation like $\mu_x + \mu_y = \mu_z$. We assume that one of the three input units is active in each input area, and only the highlighted connections of the top middle neuron are non-zero. In order to activate this unit the active units in x and y must correspond to μ_x and μ_y values which fulfill $\mu_x + \mu_y = 1$. The unit thus responds to a constant sum of the input variables μ_x and μ_y.

2. Algorithm

2.1. General idea

For simplicity of notation we will term the output quantity the "sum location", envisaging the relation $\mu_x + \mu_y = \mu_z$ as a paramount example. For a given sum location μ_z, there are many possible pairs of inputs (μ_x, μ_y) which lead to the same sum. Therefore, learning is about generating responses that are invariant to variations of input pairs which belong to the same sum location.

In order to generate these invariances, we will supply the learning algorithm with sets of input pairs that shall lead to identical output activations. However, we must not externally set the activations of the output units when using a self-organizing algorithm. For conceptual clarity, for each learning step such a set will consist of just two pairs of inputs, (μ_x, μ_y) and $(\bar{\mu}_x, \bar{\mu}_y)$. The first pair will be used to obtain the map unit activations and thereby the post-synaptic learning term. A winner-finding step and a Gaussian-profile activation function implement lateral competition and topographic relations between neighboring units. The second pair of inputs will be used in the pre-synaptic learning term to form a difference between the data and the weights to be trained.

2.2. Algorithm details

We choose an on-line learning algorithm where an incremental weight change is made during presentation of the data. The full procedure of one iteration is displayed in the algorithm box, Fig. 3.

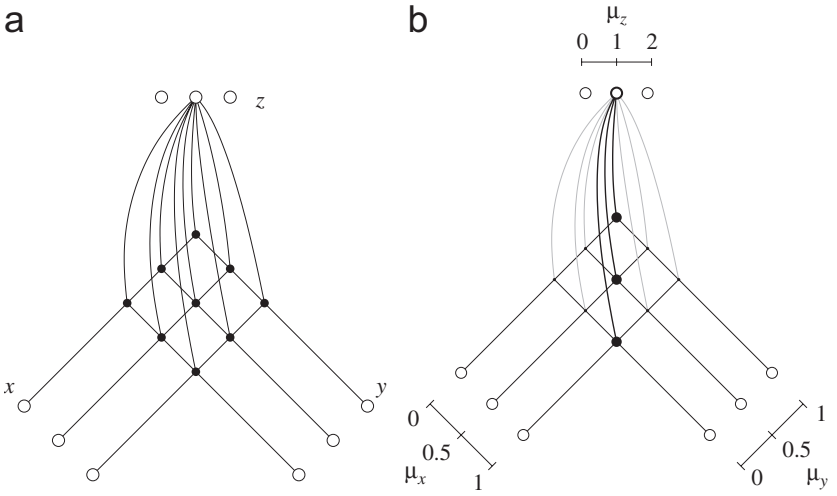

Fig. 2. (a) Model architecture. A filled circle represents a multiplication of the two inputs in the x and y layer. The results of the multiplications are summed up over the curved connections which have a weight assigned. Only three units are displayed in each area, and only the connections to one of the units in the z layer are drawn. (b) Concept of a solution for the relation $\mu_z = \mu_x + \mu_y$.

1. Choose a random sum location μ_z
2. (a) Sample randomly input locations (μ_x, μ_y) with $\mu_x + \mu_y = \mu_z$ and produce the corresponding population code $(\boldsymbol{x}, \boldsymbol{y})$
   (b) Activate all map units $\{i\}$: $a_i = \sum_{j,k} w_{ijk} x_j y_k$
   (c) Find the winning unit: $m = \mathrm{argmax}_i(\boldsymbol{a} * G)(i)$
   (d) Obtain activations: $z_i(\boldsymbol{x}, \boldsymbol{y}) = G(|i-m|, \sigma)$
   (e) Decrease the neighborhood interaction width σ
3. Sample randomly input locations $(\bar{\mu}_x, \bar{\mu}_y)$ with $\bar{\mu}_x + \bar{\mu}_y = \mu_z$ and produce the corresponding population code $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$
4. Change weights incrementally: $\Delta \boldsymbol{w}_{ijk} = \epsilon\, z_i(\boldsymbol{x}, \boldsymbol{y})\, (\bar{x}_j \bar{y}_k - \boldsymbol{w}_{ijk})$

Fig. 3. One iteration of the learning algorithm. See text for details.

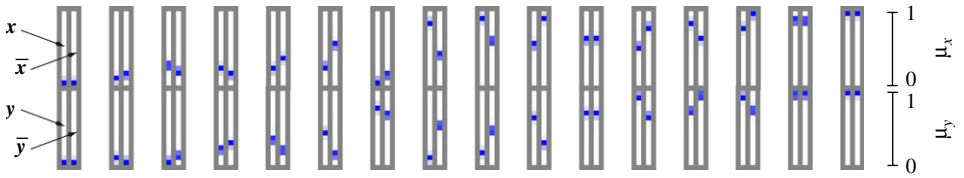

Fig. 4. Examples of training data. Top row: $\boldsymbol{x}$; bottom row, the corresponding $\boldsymbol{y}$. The data are sorted so that $\mu_z = \mu_x + \mu_y$ increases linearly from the leftmost frame to the rightmost frame. Active units are dark. In each frame, the two pairs of data, $(\boldsymbol{x}, \boldsymbol{y})$ and $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$, are shown next to each other, both belonging to the same μ_z. Scales on the right indicate the corresponding μ_x and μ_y positions.

Briefly, in Step 1, one sum location μ_z is defined, corresponding to which in Steps 2 and 3, two different pairs of input locations, (μ_x, μ_y) and $(\bar{\mu}_x, \bar{\mu}_y)$, are chosen. Both yield the same sum location μ_z, but are otherwise independent. For example, this may be when looking at a static object before and after a gaze change—then the body-centered position, here μ_z remains constant. The algorithm produces *invariant representations*, because in Step 4, the two different input pairs are combined into a single learning step. This combination is asymmetrical, using a pre- and a post-synaptic term, while an explicit output is not given to the algorithm. The algorithm produces a *topographic map* representation, because the post-synaptic term computed in Step 2(d) is a winner-take-all-like activation with excitatory, Gaussian surround.

In detail, for each iteration, the sum location is chosen in Step 1 and two different pairs of input locations are chosen in Steps 2 and 3 of the algorithm. For the first pair, (μ_x, μ_y), the corresponding population codes $(\boldsymbol{x}, \boldsymbol{y})$ are computed in Step 2(a) as input to the network. Examples of input data for the one-dimensional case are shown in Fig. 4.

The activations of the map units are given in Step 2(b) by the sigma–pi input of the data (cf. Eq. (1)). In Step 2(c) the winning unit is found by a weighted winner scheme, where the map activations are convolved with the Gaussian:

$$(\boldsymbol{a} * G)(i) := \sum_n a_n \cdot G(|i-n|, \sigma)$$

$$\text{where } G(|i-n|, \sigma) = \mathcal{N} \cdot \mathrm{e}^{-(i-n)^2/2\sigma^2}.$$

The factor $\mathcal{N}$ normalizes the sum of all map unit activations to 1. The most active unit from the result of this convolution is chosen as the winner. Directly taking the winning unit from $\boldsymbol{a}$ leads to similar results, but the weighted winner scheme, advised in [10], may lead to the reduction of noise, in particular at boundaries.

Step 2(d) computes the network output $\boldsymbol{z}$. A Gaussian activation function $G(|i-m|, \sigma)$ centered around the winning unit m implements competition as well as neighborhood interaction. The interaction width σ is the same as in Step 2(c) and is decreased in Step 2(e) over the course of learning.

Step 3 is similar to Step 2(a), but with another randomly chosen pair of locations $(\bar{\mu}_x, \bar{\mu}_y)$ which also sums up to μ_z. Step 4 is the incremental learning rule with learning rate ε. It is similar to a Hebbian rule in which the post-synaptic activation z_i is a Gaussian neighborhood function and the pre-synaptic activations are given by the difference between a data point and the weight vector.

Note that the sum location μ_z is only used to generate the network input $(\boldsymbol{x}, \boldsymbol{y})$ but not for the output $\boldsymbol{z}$. The network's representation of the sum location which can be read from the position of the activation hill $\boldsymbol{z}(\boldsymbol{x}, \boldsymbol{y})$ is therefore self-organized.

2.3. Data and network parameters

For the one-dimensional case, samples of the randomly generated input data are shown in Fig. 4. One value of μ_z is used to generate two independent pairs of positions, (μ_x, μ_y) and $(\bar{\mu}_x, \bar{\mu}_y)$. As indicated in Fig. 1, the value of μ_z spans from 0 to 2, since μ_x and μ_y both span from 0 to 1.

We set each of the two network input layers to a size of 15 units, as well as the output layer. The network weights

were initialized with random values in the interval $[-0.1, 0.1]$. The 300,000 learning iterations (Fig. 3) were made with a constant learning rate of $\varepsilon = 0.01$.

The width σ of the neighborhood activation function was reduced linearly from 8 units distance at the beginning of learning to 3.5 units distance at iteration 33,000, followed by a slower linear reduction to $\sigma = 0$ at iteration 200,000. Fig. 5 illustrates this reduction of σ. A faster decrease of σ in the early part would possibly lead to a disrupted global topography while a faster decrease in the latter part may lead to topology defects at the borders. At iteration 200,000, learning has essentially converged, but we continued training until iteration 300,000 while no relevant changes being visible, in order to make sure that any effects of the neighborhood interaction when $\sigma > 0$ are eliminated.

For the two-dimensional case, we set all three network layers to a size of 15×15 units. This leads to a total of $15^6 = 11{,}390{,}625$ connections. Near the end of training, we have cut small weights which were smaller than 0.1 times the maximum weight of a map unit. This led to about 5% of all weights remaining, thus saving computation time and memory, while not leading to a noticeable decrease in performance. Two-dimensional data samples can be seen in Fig. 9(a). In order to assist learning, also the width of the Gaussian-shaped hills of the data was changed. During an initial learning period it was set large, $\sigma^{\text{data}} = 1.5$, to assist the forming of global topography, then it was set to $\sigma^{\text{data}} = 0.5$, as in Fig. 9(a), in order to produce a sharpened mapping.

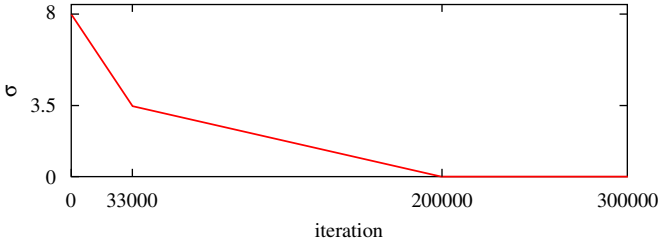

Fig. 5. Decay of the neighborhood interaction width σ over training.

3. Results

3.1. One-dimensional maps

Simple case: Fig. 6 shows the resulting connections of trained networks. In Fig. 6(a) the weights of each unit fall onto a diagonal line in input space along which the sum $\mu_x + \mu_y$ is a constant. This constant decreases linearly from left to right, indicating an "inverted" polarity of the map. Different initial random values of the weights can lead to another polarity. Test transformations of this network are displayed in Fig. 7(b) in response to the input shown in Fig. 7(a). The map units activations $\boldsymbol{a}$ are also displayed (cf. Step 2(b) of the algorithm, Fig. 3). These activations are already well focused.

Non-linear relation: Fig. 6(b) shows weights which have been trained on data that satisfy the non-linear relation $\mu_x + \mu_y^2 = \mu_z$. The plots below of the function $\mu_y = \sqrt{\mu_z - \mu_x}$ display a good match of the weight structure with this function. The polarity is here such that μ_z increases from left to right. Weights are weaker where this function has a large (negative) slope. This is because the density of the μ_y occurrences in these areas was lower during training, since μ_z was homogeneously sampled. Corresponding map activations are shown in Fig. 7(c).

Note that single network inputs (μ_x, μ_y) are similar to those used for the linear relation, since a pair which satisfies $\mu_x + \mu_y^2 = \mu_z$ for some μ_z will also satisfy $\mu_x + \mu_y = \mu_z'$ for another μ_z' and vice versa, as μ_x and μ_y are taken from the interval [0,1] and thus μ_z from [0,2]. The difference to the linear relation lies in the choice of the

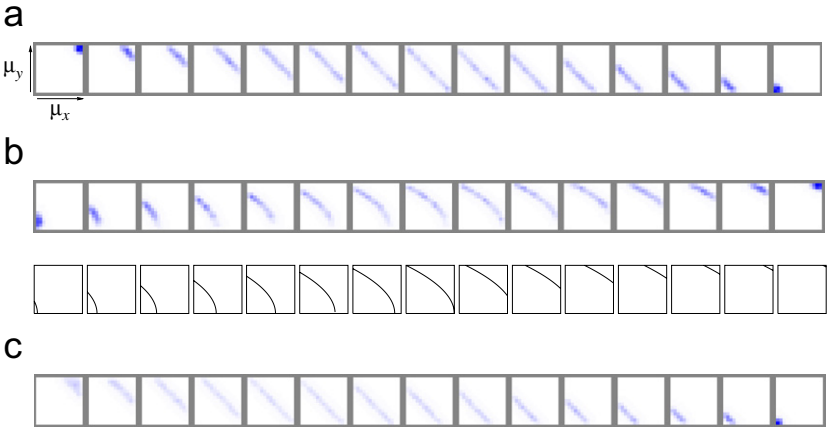

Fig. 6. Resulting weights of the one-dimensional transformation. Each square denotes the connections of a sum neuron received from the inputs. The input axes are indicated on the leftmost unit in (a). Dark blue represents strong connection weights. In (a), the data obeyed the relation $\mu_x + \mu_y = \mu_z$ where the μ_z values were uniformly distributed. In (b), the data obeyed the non-linear relation $\mu_x + \mu_y^2 = \mu_z$. Below, the corresponding relation $\mu_y = \sqrt{\mu_z - \mu_x}$ is displayed with μ_z progressing linearly from the leftmost to the rightmost unit. (c) is similar to (a), but the density of the μ_z was subject to a linear gradient which preferred small sum values.

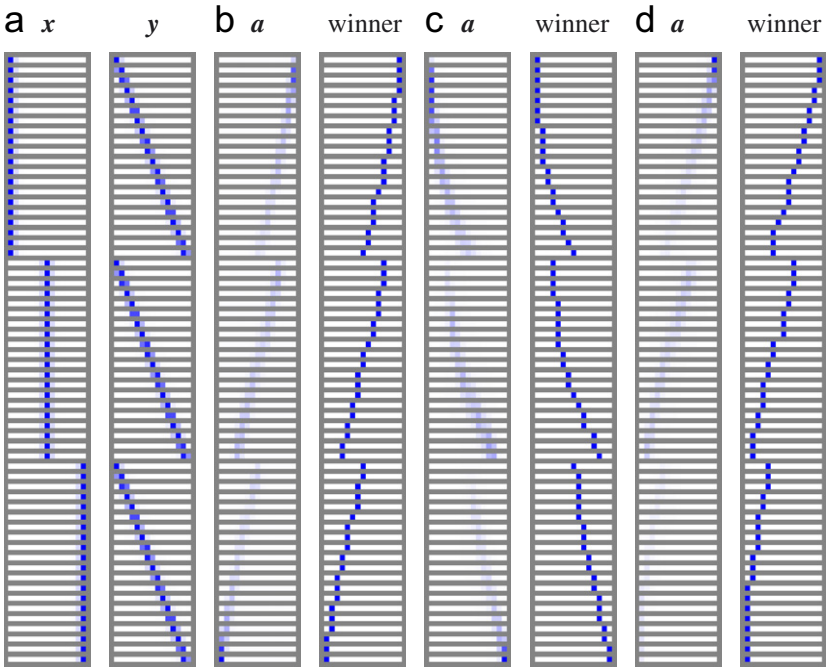

Fig. 7. Test transformations. Each thin horizontal line shows a neural activation vector. (a) shows the $\boldsymbol{x}$ and $\boldsymbol{y}$ input vectors, where locations μ_x and μ_y have been varied systematically from top to bottom. (b)–(d) show output units activation vectors $\boldsymbol{a}_z$ and the corresponding winning units in response to the $\boldsymbol{x}$ and $\boldsymbol{y}$ inputs in the corresponding lines in (a) for three different networks. In (b) the network of Fig. 6 (a) is used (linear relation), in (c) the network of Fig. 6 (b) (non-linear relation) and in (d) the network of Fig. 6(c) is used (linear relation with inhomogeneous data density).

second pair $(\bar{\mu}_x, \bar{\mu}_y)$ and therefore the relation between such two pairs of inputs.

Non-constant data density: Fig. 6(c) shows weights which have been trained on data that were distributed with a non-constant density. The μ_z values were sampled under a wedge-shaped distribution which favored small values. Again the linear relation $\mu_x + \mu_y = \mu_z$ was given. Accordingly, the weights of individual units are as in Fig. 6(a), but now more units represent the more frequent smaller values. More precisely, the 10 rightmost units represent values $\mu_z < 1$ and only the five leftmost units represent $\mu_z > 1$. A unit representing $\mu_x = 1$ and $\mu_y = 1$ is more or less missing. Accordingly, the mapping is less precise at the leftmost units. This can also be seen below in Fig. 7(d) where the last eight μ_x, μ_y combinations activate the same (leftmost) map unit. On the other hand, high-density regions in data space are finer resolved by more units of the network, so in the upper region of Fig. 7(d), more units are recruited than in a corresponding region in Fig. 7(b).

3.2. Two-dimensional maps

Because of the practical importance, we have also implemented transformations of two-dimensional codes. Now $\boldsymbol{\mu}_x$, $\boldsymbol{\mu}_y$ and $\boldsymbol{\mu}_z$ are two-component vectors denoting a point on a plane, and we have tested the vector summation $\boldsymbol{\mu}_x + \boldsymbol{\mu}_y = \boldsymbol{\mu}_z$. Practically, $\boldsymbol{\mu}_x$ may be a retinal-centered position of a seen object and $\boldsymbol{\mu}_y$ the eye position in terms of horizontal and vertical angle of gaze direction. Then $\boldsymbol{\mu}_z$ would be the head-centered object position.

The high-dimensional network weights are difficult to visualize. Topography cannot be displayed as usual for SOM networks, by displaying the map units' weights in the input space, because of high-dimensional population coding, and even the lower-dimensional $(\boldsymbol{\mu}_x, \boldsymbol{\mu}_y)$ space is a product of two two-dimensional variables. The network should (i) map inputs that correspond to different sum locations to different regions on the map, (ii) map groups of inputs which correspond to the same sum location to a narrowly confined region on the map, and (iii) should establish a topography between the map and the $\boldsymbol{\mu}_z$ space. These properties are demonstrated in Fig. 8 for the two-dimensional variable transformation.

The large grid connects groups of map locations, where the corresponding sum locations $\boldsymbol{\mu}_z$ have been taken from a square grid. For each sum location, 50 random input pairs $(\boldsymbol{\mu}_x, \boldsymbol{\mu}_y)$ that lead to that sum location were given to the network as input, and the map locations of maximum activation were assessed. These 50 output locations were joined by one line of a random color in order to display the spread of the output. This process was repeated for 10×10 different sum locations located on a grid in $\boldsymbol{\mu}_z$ space. The locations of maximum activation were determined by a weighted winner scheme, by convolving the map activations $\boldsymbol{a}$ with a Gaussian ($\sigma = 1$), as in Step 2(c) of the algorithm. This allowed a near continuous estimate of the locations rather than being confined to the grid units if simply the winning unit was taken. It can be seen that the locations of maximum activation varied by a maximum distance of around one grid unit separation for each sum

location, and that these 10×10 different regions do not overlap. Furthermore, the map establishes topography with respect to the $\boldsymbol{\mu}_z$ space, even though it has only received input drawn from the $(\boldsymbol{\mu}_x, \boldsymbol{\mu}_y)$ space.

Some example transformations are demonstrated in Fig. 9. In Fig. 9(a) the three input pairs represent locations $(\boldsymbol{\mu}_x, \boldsymbol{\mu}_y)$ which lead to the same given sum $\boldsymbol{\mu}_z$. These data are three of the 50 samples used to produce one of the 10×10 connected groups in Fig. 8. Despite the large variation of the inputs, the network correctly responds in an invariant fashion to these transformations.

In Fig. 9(b) it can be seen that a more complex activation pattern on one input layer will be reflected in the "inner activation" vector $\boldsymbol{a}$ on the map. This is because of the linear relation of $\boldsymbol{a}$ to either input, given that the other input is fixed. This linear relation could also be used to convey the amplitudes of the signals. However, the output is sensitive to noise in either input, since both inputs are multiplied (bilinear relation) [15].

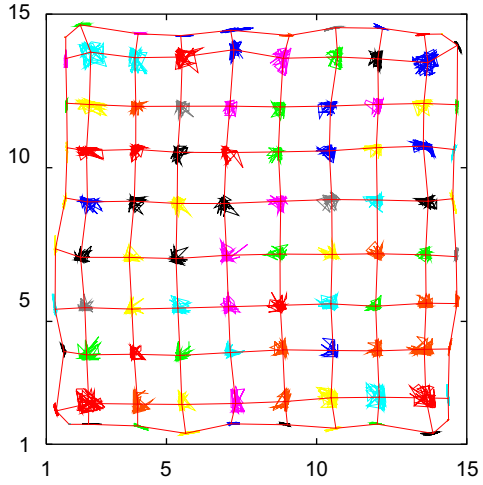

Fig. 8. A plot on the precision and the topography of the two-dimensional mapping. The 10×10 sum locations were chosen on a square grid in $\boldsymbol{\mu}_z$ space. For each sum location, 50 pairs of $(\boldsymbol{\mu}_x, \boldsymbol{\mu}_y)$ were sampled randomly given that they lead to the chosen sum. The 50 map locations of maximal activation are connected by one line of a random color, for each given sum location, to visualize the spread of the mapping. The centers of these 10×10 groups are connected by a grid, to visualize the topographic mapping. Labels on the axes delineate map unit space.

4. Discussion

Based on our recent approach of a neural frame of reference transformation which was trained by supervised learning [17], we intend to use the model presented in this paper in the context of a neurally controlled robot docking maneuver. The supervised system has been tested on a robot simulator, and Fig. 10 explains the geometry on our PeopleBot robot.

The overall neural system which controls a robot to pick up an object will consist of three parts: (i) a visual system provides the horizontal and vertical coordinates $(x^{\mathrm{h}}, x^{\mathrm{v}})$ of an object of interest within the camera image. (ii) This location as well as the camera pan and tilt direction $(y^{\mathrm{p}}, y^{\mathrm{t}})$ is input to the sigma–pi SOM, which computes motor-relevant coordinates that correlate with distance and angle $(z^{\mathrm{d}}, z^{\Theta})$ of the object in the robot-body-centered frame of reference. (iii) With such a body-centered position as input, a reinforcement-trained network directs the robot until the object appears at a rewarded position at the robot's grippers.

The advantage of self-organized as opposed to supervised learning of the frame of reference transformation is that the body-centered location need not be included in the training data. Moreover, no geometrical calculations need to be done as is the case with hard-wired methods. This is important since the body-centered location is not given directly in humans or monkeys and in a robot would need costly measuring or retrieval by a simulator.

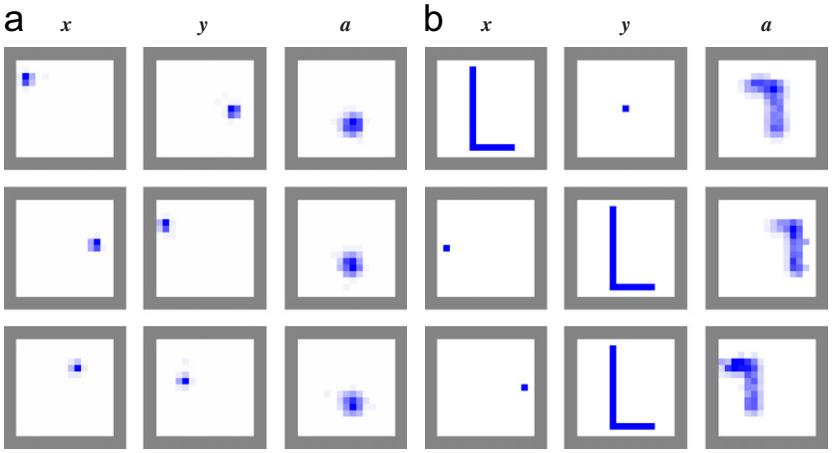

Fig. 9. Test transformations of the two-dimensional network. Samples of inputs $\boldsymbol{x}$ and $\boldsymbol{y}$ are given to the network as shown in the first two columns of each part of the figure, and the corresponding network response $\boldsymbol{a}$ is shown in the third columns. (a) Three random input pairs are given under the constraint that they belong to the same sum value $\boldsymbol{\mu}_z$. The network response $\boldsymbol{a}$ is almost identical for all three input pairs. (b) When a more complex "L"-shaped activation pattern is given to either input, a similar activation pattern emerges on the sum area. It can be seen that the map polarity is rotated by 180°.

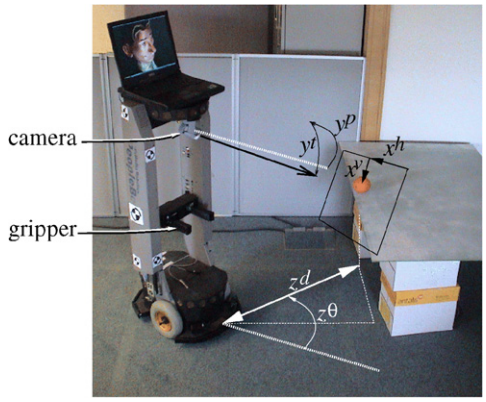

Fig. 10. The PeopleBot robot shown can pan/tilt its camera, hence image coordinates do not directly unveil an object's location in real space. Instead, a coordinate transformation has to be made in the following way. Sensory inputs are the horizontal and vertical coordinates (x^h, x^v) of an object of interest within the camera image, and the pan and tilt direction (y^p, y^t) of the robot camera. The coordinate transformation must compute object coordinates which can be used to pick up the object. These can be the distance and angle (z^d, z^Θ) of the object to the robot in a fixed robot-body-centered frame of reference. (In order to have a uniquely defined mapping, here the assumption is made that the object is always at the same elevation from the floor.)

The self-organized map output is a suitable state space representation for the reinforcement learner, because different views of the object—by varying only the camera pan/tilt angle—will yield a constant representation, as long as the body-centered position of the target object remains constant (in analogy to Fig. 9(a)). On the other hand, varying the body-centered positions by moving the robot will yield varying output activations, because the SOM units strive to cover the whole input space. The reinforcement learner will learn to reach the goal region of the state space based on a reward that can be administered based on external factors. The particular reinforcement algorithm used earlier [18] takes advantage of topography, but does not make assumptions on the state space such as a certain polarity of the topographic map.

The use of sigma–pi neurons is in the nature of the given problem, but may overestimate the computational capabilities of cortical neurons. It is therefore worthwhile attempting such a self-organized mapping with simpler, connectionist neurons in future.

Novelty of contribution to SOMs: The Kohonen algorithm has many applications such as data analysis and visualization, classification of images and acoustic patterns, financial analysis, traveling salesman problem, adaptive robot control and the modeling of biological phenomena [9]. The use of sigma–pi-type neuronal units with SOMs is entirely novel and may spawn new areas of application for SOMs. We have chosen these kinds of units in order to account for the specific data structure which allows to associate a given input in the x layer with a variety of inputs in the y layer. Our data are relevant for an agent acting in the physical environment such as a human or a humanoid robot, but applicable data structure may exist also in other domains. The learning algorithm has been modified from the original Kohonen rule to account for this qualitative increase in data complexity by pairing a map activation vector with two data presentations instead of just one as for the standard Kohonen learning rule. As a consequence, for each input unit on the x layer, a complete map to the y layer emerges from learning, and vice versa.

In summary, we have presented a novel self-organizing map of sigma–pi neurons that performs frame of reference transformations which are important to robotic applications as well as to the biological system.

Acknowledgments

This research is part of the MirrorBot project supported by a EU, FET-IST programme, Grant IST-2001-35282, coordinated by Prof. Wermter.

References

[1] C.H. Andersen, D.C. van Essen, B. Olshausen, Directed visual attention and the dynamic control of information flow, in: L. Itti, G. Rees, J. Tsotsos (Eds.), Encyclopedia of Visual Attention, Academic Press, Elsevier, New York, Amserdam, 2004.
[2] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, 1995.
[3] C.A. Buneo, M.R. Jarvis, A.P. Batista, R.A. Andersen, Direct visuomotor transformations for reaching, Nature 416 (2002) 632–636.
[4] Y.E. Cohen, R.A. Andersen, A common reference frame for movement plans in the posterior parietal cortex, Nat. Rev. Neurosci. 3 (2002) 553–562.
[5] J.D. Crawford, W.P. Medendorp, J.J. Marotta, Spatial transformations for eye–hand coordination, J. Neurophysiol. 92 (2004) 10–19.
[6] S. Deneve, P.E. Latham, A. Pouget, Efficient computation and cue integration with noisy population codes, Nat. Neurosci. 4 (8) (2001) 826–831.
[7] J.R. Duhamel, F. Bremmer, S. Benhamed, W. Graf, Spatial invariance of visual receptive fields in parietal cortex neurons, Nature 389 (1997) 845–848.
[8] D.B. Grimes, R.P.N. Rao, Bilinear sparse coding for invariant vision, Neural Comput. 17 (2005) 47–73.
[9] T. Kohonen, Self-Organizing Maps, Springer Series in Information Sciences, third ed., vol. 30, Springer, Berlin, Heidelberg, New York, 2001.
[10] S.P. Luttrell, A bayesian analysis of self-organizing maps, Neural Comput. 6 (1994) 767–794.
[11] B.W. Mel, C. Koch, Sigma–pi learning: on radial basis functions and cortical associative learning, in: D.S. Touretzsky, (Ed.), Advances in Neural Information Processing Systems, vol. 2, 1990, pp. 474–481.
[12] E. Sauser, A. Billard, Three dimensional frames of references transformations using recurrent populations of neurons, Neurocomputing 64 (2005) 5–24.
[13] D. Schluppeck, P. Glimcher, D.J. Heeger, Topographic organization for delayed saccades in human posterior parietal cortex, J. Neurophysiol. 94 (2005) 1372–1384.
[14] L.H. Snyder, Coordinate transformations for eye and arm movements in the brain, Curr. Opinion Neurobiol. 10 (2000) 747–754.
[15] M.W. Spratling, G.M. Hayes, Learning synaptic clusters for non-linear dendritic processing, Neural Process Lett. 11 (2000) 17–27.
[16] A. van Rossum, A. Renart, Computation with populations codes in layered networks of integrate-and-fire neurons, Neurocomputing 58–60 (2004) 265–270.

[17] C. Weber, D. Muse, W. Elshaw, S. Wermter, A camera-direction dependent visual-motor coordinate transformation for a visually guided neural robot, Knowl.-Based Syst. 19 (5) (2006) 348–355.
[18] C. Weber, S. Wermter, A. Zochios, Robot docking with neural vision and reinforcement, Knowl.-Based Syst. 17 (2–4) (2004) 165–172.

Cornelius Weber is a Junior Fellow at the Frankfurt Institute for Advanced Studies in Germany since March 2006. He graduated in physics in Bielefeld, Germany in 1995 and received his PhD in computer science in Berlin in 2000. Then he worked in the group of Alexandre Pouget in Brain and Cognitive Sciences, University of Rochester, USA. From 2002 to 2005 he worked in Hybrid Intelligent Systems at the University of Sunderland, UK throughout the EU-funded MirrorBot project. His research interests are in computational neuroscience, focusing on visual and motor systems, and robotic applications. In December 2003 he won the Machine Intelligence Prize of the British Computer Society in Cambridge, demonstrating the "visually guided grasping robot MIRA". This publication is motivated by extending the robot's grasping range for such a scenario.

Stefan Wermter is professor in Intelligent Systems at the University of Sunderland, UK and is the Director of the Centre for Hybrid Intelligent Systems. His research interests are in intelligent systems, neural networks, cognitive neuroscience, hybrid systems, language processing and learning robots. He has a Diploma from the University of Dortmund, an MSc from the University of Massachusetts and a PhD and Higher Doctorate (Habilitation) from the University of Hamburg, all in computer science. He was a Research Scientist at ICSI, Berkeley in 1997 before accepting the Chair in Intelligent Systems at the University of Sunderland in 1998.

Professor Wermter has written or edited five books and published about 150 articles on this research area, including books like "Hybrid Connectionist Natural Language Processing" or "Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing", "Hybrid Neural Systems", "Emergent Neural Computational Architectures based on Neuroscience" and "Biomimetic Neural Learning for Intelligent Robots".

He is an Associate Editor of the journals "Connection Science", the "International Journal for Hybrid Intelligent Systems" and the "Knowledge and Information Systems". He is on the editorial board of the journals "Neural Networks", "Cognitive Systems Research", "Neural Computing Surveys", "Neural Information Processing" and "Journal of Computational Intelligence". Furthermore, he is leading the EU project MirrorBot on biomimetic multimodal learning in a mirror neuron-based robot and coordinates the EmerNet network on "emerging computational neural architectures based on neuroscience".