Rapid Learning with Parametrized Self-Organizing Maps

Jörg Walter and Helge Ritter

Department of Information Science
University of Bielefeld, D-33615 Bielefeld, FRG
Email: {walter,helge}@techfak.uni-bielefeld.de

Abstract

The construction of computer vision and robot control algorithms from training data is a challenging application for artificial neural networks. However, many practical applications require an approach that is workable with a small number of data examples. In this contribution, we describe results on the use of "Parametrized Self-organizing Maps" ("PSOMs") with this goal in mind. We report results that demonstrate that a small number of labeled training images is sufficient to construct PSOMs to identify the position of finger tips in images of 3D-hand shapes to within an accuracy of only a few pixel locations.

Further we present a framework of hierarchical PSOMs that allows rapid *"one-shot-learning"* after acquiring a number of "basis mappings" during a previous *"investment learning stage"*. We demonstrate the potential of this approach with the task of constructing the position-dependent mapping from camera coordinates to the work space coordinates of a Puma robot.

1 Introduction

Learning is one of the capabilities that make artificial neural networks a favorable approach over a variety of more traditional approaches, such as e.g., fuzzy systems [18]. In particular, in the domain of computer vision and robotics, many aspects of a task are costly to model from first principles or even heuristically. Therefore, in these domains efficient learning algorithms can greatly help to overcome the "knowledge acquisition bottleneck" and make it easier to construct more flexible and more robust systems more rapidly.

However, for such applications an important issue is the availability of sufficient training data. Whereas in vision it may be easy to gather large amounts at least of unlabeled data, this is much less so in robotics, where each data item may require a time-consuming action of the robot. If the data have to be labeled, which is usually required by the more powerful supervised learning algorithms, then even in vision the acquisition of larger numbers of training examples can be very costly.

Therefore, to make practical real-world applications in these domains, it is important to go beyond most of the currently popular neural network models that usually require thousands of training examples to provide useful performance, and to develop approaches specifically suited to construct neural systems from small numbers of training examples.

Recently, a computationally motivated variant of the self-organizing map ([5, 4, 14]), the *Parametrized Self-Organizing Map* ("PSOM") was proposed as a potentially useful scheme for learning with particularly small numbers of training examples. The basic idea of a PSOM is to sacrifice the generality of the original SOM by constructing the map manifold from a restricted repertoire of *basis manifolds*. The choice of these basis manifolds can be informed by a-priory knowledge about the task at hand. Consequently, the final map manifold can be described by a small number of parameter values which can be determined from a much smaller number of training examples than for the original SOM.

In [13], the feasibility of this approach was demonstrated for examples from the domain of robotics, among them the learning of the inverse kinematics transform of a full 6-degrees of freedom Puma robot with less than 800 examples and for a 3-DOF manipulator with 27 examples only (in both cases to within percent accuracy).

In this paper, we report more recent results about the use of PSOMs in the context of *visual learning*. Specifically, we discuss the task to learn to extract the positions of object features from monocular gray level images of the objects. As an example, we consider images of an articulated hand of variable shape, taken from various view directions, with the aim to identify the position of the finger tip in these images. As a second demonstration, we describe how one-shot learning in the context of camera-calibration becomes possible within a hierarchy of multiple PSOMs. In this framework, a preparatory or *"investment learning"* stage is used to first construct a number of "basis" PSOMs. Using these pre-learnt mappings then allows a second PSOM in a *rapid learning* phase to construct the required camera mapping on the basis of *one* single training sample.

The plan of the paper is as follows. In Sect. 2, we give an outline of PSOMs. In Sect. 3, we consider their application to vision tasks and report results. In Sect. 4 we describe the hierarchical approach for one-shot learning in the context of the camera calibration task. Finally, Sect. 5 contains the discussion and an outlook on future work.

2 Parametrized Self-organizing Maps

The construction of a good data representation is often the decisive step in the solution of a problem. This is particularly true for learning tasks, where the capability of *generalizing* from a limited set of examples to novel instances forms the central aim. To support this objective, a representation must serve two usually conflicting goals: it must provide a representation of the data that maintains the *similarity relationships* among data elements as faithfully as possible. Yet, it also should provide a *compression* onto the essential variables and project out spurious or unimportant information.

The *Karhunen-Loeve transform* or *principal component analysis* ("PCA", [3]) is the well-known answer that results when looking for an optimal *linear* solution to these requirements. Its basic idea is to construct a low-dimensional, linear subspace that is oriented in such a way in the original space that it captures as much of the data variation as possible. The wide-spread use of this method indicates the importance of constructing good data representations.

The Kohonen *self-organizing map* ("SOM") can be viewed as a non-linear extension of principal component analysis [5, 4, 14]. It replaces the linear subspace of PCA by a *nonlinear manifold* that can represent even very non-linear data distributions. The manifold is constructed by an iterative learning procedure and can be viewed as a non-linear, "topology-conserving map" of the original data space. The great success of this method results from its capability to produce representations that yield a very good compromise between efficient data compression and a sufficient faithful preservation of the underlying "data topology" to support good generalization.

In the basic SOM-algorithm, the nonlinear map manifold is represented by a discrete approximation, using a (m dimensional) lattice $\mathbf{A}$ of formal neurons each labeled by an index $\mathbf{a} \in \mathbf{A}$, and attached reference vectors $\mathbf{w_a}$ [5, 4, 14], as illustrated in Fig. 1. The response of a SOM to an input vector $\mathbf{x}$ is determined by the reference vector $\mathbf{w_{a^*}}$ of the discrete "best-match" node $\mathbf{a}^*$, where $\mathbf{a}^*$ is defined by $\mathbf{a}^* = \text{argmin} \|\mathbf{w_{a'}} - \mathbf{x}\|$, where $\mathbf{a}' \in \mathbf{A}$. Originally, this approach has been motivated by the desire to model the self-organization of topographic maps in populations of discrete neurons in the brain, and it also has the virtue of being very general. However, the discrete nature of the standard SOM can be a limitation when the construction of smooth, higher-dimensional map manifolds is desired. Since the number of nodes grows exponentially with the number of map dimensions, manageably sized lattices with, say, more than three dimensions (for an application of a 3d-SOM, see e.g. [14]) admit only very few nodes along each axis direction and can, therefore, be not sufficiently smooth for many purposes where continuity is very important, as e.g. in control tasks or in robotics.

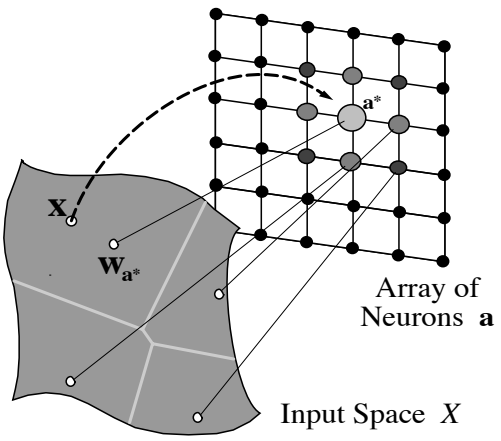

Figure 1: The Kohonen, or "Self-Organizing Map" ("SOM") can be viewed as an array (here 2D, $m = 2$) of formal neurons. Each neuron has a reference vector $\mathbf{w_a}$ attached, tessellating the embedding input space X in discrete patches (Voronoi cells).

A *parametrized self-organizing map* ("PSOM") generalizes this scheme of a discrete lattice $\mathbf{A}$ to a parametrized "mapping manifold" $S \subset \mathbb{R}^m$ on which the map coordinates $\mathbf{s}$ may vary continuously. As a consequence, the discrete assignment of reference vectors $\mathbf{w_a}$ to lattice points $\mathbf{a}$ is replaced by a continuous, vector-valued function $\mathbf{w}(\cdot) : \mathbf{s} \mapsto \mathbf{w}(\mathbf{s}) \in M \subset X$, where $\mathbf{s}$ varies continuously over a subset $S \in \mathbb{R}^m$. Similar to $\mathbf{w_a}$, $\mathbf{w}(\mathbf{s})$ takes its value in an embedding space $X \subseteq \mathbb{R}^d$, from which the input vectors $\mathbf{x}$ are also drawn. The response of the PSOM is determined by the value $\mathbf{w}(\mathbf{s}^*)$, again taken at some best-match location $\mathbf{s}^*$, which, unlike the best-match location $\mathbf{a}^* \in S$ in the SOM case, must now be found in the continous mapping manifold S defined by Eq. 1 below.

Figures 2a-b illustrate the training set-up of a PSOM that represents a $m = 2$ dimensional mapping manifold S. The image of S under $\mathbf{w}()$ is a manifold $M \subset X \subseteq \mathbb{R}^3$,

which we will denote as "embedded manifold". For this pedagogic example, we choose M as a tilted plane of which 9 sample points $\mathbf{w}_1, \ldots, \mathbf{w}_9$ are assumed to be given. To construct the PSOM requires assignment of the location $\mathbf{a}$ in the mapping manifold S to each given training point ($\mathbf{w}_1 = \mathbf{w}_{\mathbf{a}_{11}}; \ldots; \mathbf{w}_9 = \mathbf{w}_{\mathbf{a}_{33}}$), and thereby specifying a topological organization, as explained below.

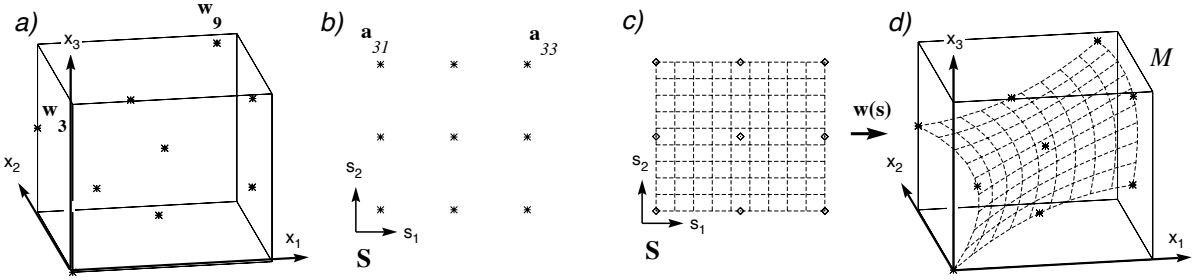

Figure 2: a–d: Illustration of the training setup for a 3×3 PSOM for the simple task of a linear mapping defined by a tilted plane in 3D. *(a)* The nine training points are roughly topologically ordered, as shown by the tick marks in a orthonormal projection of the 3D embedding space X and *(b)* their corresponding auxiliary mapping coordinates $\mathbf{a} \in \mathbf{A}$ in the mapping manifold S, lying on a rectangular grid. By the virtue of Eq. 2, S spans an $m = 2$ dimensional manifold M in the embedding space X, visualized in *(c–d)*. Note, the training set does not necessarily lie on any exact grid in X. The cube is drawn for visual guidance only.

The best-match location $\mathbf{s}^*$ in the mapping manifold S it then found, in analogy to the SOM algorithm, by minimizing a distance function $dist(\cdot)$

$$\mathbf{s}^* = \text{argmin } dist\,(\mathbf{w}(\mathbf{s}), \mathbf{x})\,. \tag{1}$$

How can the required smooth manifold $\mathbf{w}(\mathbf{s})$ be actually constructed? In principle, there are several ways, but a particular close analogy to the standard SOM results when we take

$$\mathbf{w}(\mathbf{s}) = \sum_{\mathbf{a}\in\mathbf{A}} H(\mathbf{a}, \mathbf{s})\, \mathbf{w}_\mathbf{a}\,. \tag{2}$$

This means that we need a *"basis function"* $H(\mathbf{a}, \mathbf{s})$ for each formal neuron (in the following also called "knot"), weighting the contribution of its reference vector ("training point") $\mathbf{w}_\mathbf{a}$ depending on the location $\mathbf{s}$ relative to the knot position $\mathbf{a}$ in the mapping manifold S, and possibly also the relative positions to all the other knots (however, in our notation we drop the dependency $H(\mathbf{a}, \mathbf{s}) = H(\mathbf{a}, \mathbf{s}; \mathbf{A})$ on the latter.)

Specifying for each training vector a knot location $\mathbf{a} \in \mathbf{A}$ introduces a topological order between the training points $\mathbf{w}_\mathbf{a}$: training vectors assigned to knots $\mathbf{a}$ and $\mathbf{a}'$, that are neighbors in the lattice $\mathbf{A}$, are perceived to have the specific neighborhood relation. This allows the PSOM to draw extra curvature information from the training set, information which is not available within other techniques, such as the Radial Basis approach (e.g. [10, 1]).

The topological organization of the given data points is crucial for a good generalization behavior. For a general data set the topological ordering of its points may be quite irregular and a set of suitable basis functions $H(\mathbf{a}, \mathbf{s})$ difficult to construct.

A suitable set of basis functions can be constructed in many ways but must meet two conditions: (i) The hyper-surface M shall pass through all desired support points. At those points, only the local knot contributes (which weight one.) This is expressed by the *ortho-normality* condition:

$$H(\mathbf{a}_i, \mathbf{a}_j) = \delta_{ij} \; ; \quad \forall \, \mathbf{a}_i, \mathbf{a}_j \in \mathbf{A}. \tag{3}$$

(ii) Consider the task of mapping a constant function $\mathbf{x} = \mathbf{w_a}$. Obviously the sum in Eq. 2 should be constant as well, which means, the sum of all contribution weights should be one. This is the *division of unity* condition:

$$\sum_{\mathbf{a} \in \mathbf{A}} H(\mathbf{a}, \mathbf{s}) = 1, \; \forall \mathbf{s}. \tag{4}$$

A simple construction of basis functions $H(\mathbf{a}, \mathbf{s})$ becomes possible when the topology of the given points is sufficiently regular. A particularly convenient situation arises for the case of a multidimensional rectangular grid. In this case, the set of functions $H(\mathbf{a}, \mathbf{s})$ can be constructed from products of one-dimensional Lagrange interpolation polynomials, as described in more detail in the Appendix.

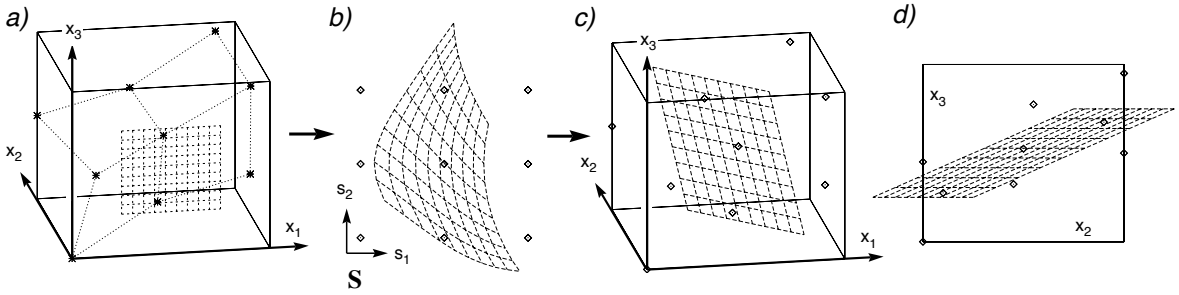

Figure 3: a–d: PSOM recall procedure for a rectangular spaced set of 10×10 (x_1, x_3) tuples to (x_2, x_3), together with the original training set: *(a)* the input space in the $x_2 = 0$ plane, *(b)* the resulting (Eq. 1) mapping coordinates $\mathbf{s}^* \in S$, *(c)* the completed data set in X, *(d)* the desired output space projection (looking down x_1).

When M has been specified, the PSOM is used in an analogous fashion like the SOM: given an input vector $\mathbf{x}$, (i) first find the best-match position $\mathbf{s}^*$ on the mapping manifold S by minimizing a distance $dist(\mathbf{x}, \mathbf{w}(\mathbf{s}))$, and then (ii) use the surface point $\mathbf{w}(\mathbf{s}^*)$ as the output of the PSOM in response to the input $\mathbf{x}$ ("recall", using Eq. 2.) To build an input-output mapping, the standard SOM is often extended by attaching a second vector $\mathbf{w}^{out}$ to each formal neuron. Here, we generalize this and view the embedding space $X \subset \mathbb{R}^d$ as the Cartesian product of some "input" subspace X^{in} and some "output" subspace X^{out}. Then,

$\mathbf{w}(\mathbf{s}^*)$ can be viewed as an *associative completion* of the input space component of $\mathbf{w}(\mathbf{s}^*)$, if the distance function $dist(\cdot)$ (in Eq. 1) is chosen as the Euclidean norm applied only to the input components of $\mathbf{x}$ (belonging to X^{in}), Thus, the function $dist(\cdot)$ actually selects the input subspace X^{in}, since for the determination of $\mathbf{s}^*$ (Eq. 1, and as a consequence, of $\mathbf{w}(\mathbf{s}^*)$) only those components of $\mathbf{x}$ matter, that are regarded in the distance metric $dist(\cdot)$. The mathematical formulation is the definition of a diagonal projection matrix $\mathbf{P} = \text{diag}(p_1, p_2, \ldots, p_d)$ with diagonal element $p_k = 1,\ \forall k \in I$, and all other elements are zero. The set I is the subset of components of X ($\{1, 2, \ldots, d\}$) belonging to the desired X^{in}. Then, the distance function can be written

$$dist(\mathbf{x}, \mathbf{x}')^2 = (\mathbf{x} - \mathbf{x}')^T \, \mathbf{P} \, (\mathbf{x} - \mathbf{x}') = \sum_{k \in I} p_k \, (x_k - x'_k)^2. \tag{5}$$

As an important feature, $dist(\cdot)$ can be changed on demand, allowing e.g. to reverse the mapping direction using the same PSOM (or switch to an alternative input space) as illustrated below.

Returning to Eq. 1, we see that the discrete best-match search in the standard SOM is now replaced by solving the continuous minimization problem for the determination of $\mathbf{s}^*$. This increased cost for using the PSOM is the price to pay for its very direct and simple construction. A simple approach to solve Eq. 1 is to initially ($t = 0$) find the (SOM-like) discrete best-match knot $\mathbf{s}_{t=0} = \mathbf{a}^*$ in the knot set $\mathbf{A}$ associated with the given data points, followed by an iterative gradient descent for minimizing $dist(\mathbf{w}(\mathbf{s}), \mathbf{x})$ (Eq. 1):

$$\mathbf{s}_{t+1} = \mathbf{s}_t + \gamma \, \mathbf{J}(\mathbf{s})^T \, \mathbf{P} \, [\mathbf{x} - \mathbf{w}(\mathbf{s}_t)]. \tag{6}$$

Here, $\gamma > 0$ is a step size parameter, $\mathbf{J}(\mathbf{s})$ is the Jacobian $\partial \mathbf{w}(\mathbf{s}) / \partial \mathbf{s}$ evaluated at the current point, using Eq. 2. This is a very general and simple scheme that works satisfyingly in many cases. However, the proper choice of the learning parameter γ is somewhat critical. Too small values lead to unnecessarily many iteration steps, too large values can lead to divergence. However, the special quadratic form of the minimized distance measure allows to make use of more robust and efficient, second order methods. Thus, our current implementation uses the computationally more efficient Levenberg-Marquardt-Algorithm that finds the best match coordinate vector $\mathbf{s}^*$ safely within a couple of iterations, see Appendix and [12].

Fig. 3b shows the result of the "best-match projection" $\mathbf{x} \mapsto \mathbf{s}^*(\mathbf{x})$ into the manifold M, when $\mathbf{x}$ varies over a regular 10×10 grid in the plane $x_2 = 0$ and x_1, x_3 is selected as the input space (by defining $dist(\cdot)$ appropriately). Fig. 3c–d shows a rendering of the associated "completions" $\mathbf{w}(\mathbf{s}^*(\mathbf{x}))$ which form a grid in X.

Equation (6) can be viewed as a *network dynamics* for a recurrent network, however, with node activities represented parametrically by the map coordinate vector $\mathbf{s}$. This network bears interesting similarities to the standard SOM, in particular, one can consider the vectorial coefficients $\mathbf{w_a}$ as the reference vectors of an underlying, "coarse", standard SOM.

The naming of the PSOM has its historical roots in the derivation from the SOMs. The ability of *self-organizing* a training set is supplied by an pre-structuring SOM by sharing

the reference vectors. Even if we hesitate renaming the method, we introduced the PSOM here independently operating from a SOM. This is useful in all applications where the required topological order can be gained by structured sampling of the training data – often without any extra effort.

A particularly important property of the network dynamics Eq. 6 is that it enables the PSOM *to complete partial data vectors* (of course, the same feature still holds if the gradient dynamics is replaced by a more efficient computation scheme, such as the mentioned Levenberg-Marquardt-algorithm). This occurs, when $|I| < d$, i.e., I is a proper subset of the set of all dimensions of X. In this case, $\dot{dist}()$ (Eq. 2) is taken only in the subspace X^{in} of specified data components, and the specified components of the input vector *act as a constraint* for the determination of a map location $\mathbf{s}^*$, from which then the remaining data components are obtained by virtue of the associated vector $\mathbf{w}(\mathbf{s}^*)$. Usually, if at least m components of $\mathbf{x}$ are specified, this constraint will be sufficient to constrain the set of compatible map locations to a single point, and the associated vector $\mathbf{w}(\mathbf{s}^*)$ can be considered as the completion of the partial input[1] $\mathbf{s}$.

This property is particularly useful if the map manifold represents the *graph of a function*, since it then allows to arbitrarily split the d-dimensional variable set of X into a m-dimensional subset of "independent" ("input") and $d-m$ remaining, "dependent" ("output") variables, the values of which are obtained from the value of $\mathbf{w}(\mathbf{s})$ at the fixed point $\mathbf{s}^*$ of Eq. 6.

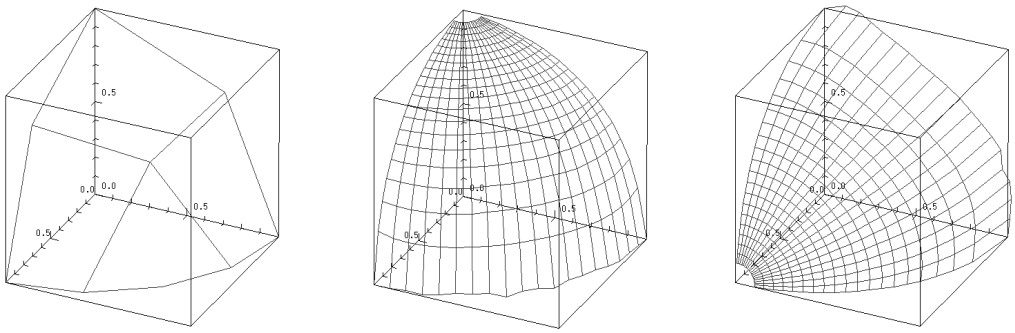

Figure 4: *Left:* Reference vectors of a 3×3 SOM, shared as training vectors by a 3×3 PSOM, representing one octant of the unit sphere surface ($x_1, x_2, x_3 > 0$). *Center:* A mapping $x_3 = x_3(x_1, x_2)$ obtained from the PSOM with $\mathbf{P}$ =diag$(1, 1, 0)$, *right:* same PSOM, but used for mapping $x_1 = x_1(x_2, x_3)$ by choosing $\mathbf{P}$ =diag$(0, 1, 1)$.

As a simple example, consider a 2-dimensional data manifold in $\mathbb{R}^3$ that is given by the portion of the unit sphere in the octant $x_i > 0$ ($i = 1, 2, 3$). Fig. 4, *left*, shows a SOM, providing a discrete approximation to this manifold with a 3×3-mesh. While the number of nodes could be easily increased to obtain a better approximation for the

[1]more precisely, in the non-degenerate case m specified values will constrain the possible solutions $\mathbf{s}$ to a zero-dimensional subset of the map manifold. This subset may contain one or several discrete alternatives.

two-dimensional manifold of this example, this remedy becomes impractical for higher dimensional manifolds and the coarse approximation that results from having only three nodes along each manifold dimension is typical for these cases. However, we can use the nine reference vectors, together with the neighborhood information from the 3×3 SOM to construct a PSOM that provides a much better, fully continuous representation of the underlying manifold.

Fig. 4 demonstrates the 3×3 PSOM working in two different mapping "directions". This flexibility in associative completion of alternative input spaces X^{in} is useful in many contexts. For instance, in robotics a positioning constraint can be formulated in joint, Cartesian or, more general, in mixed variables (e.g. position and some wrist joint angles), and one may need to know the respective complementary coordinate representation, requiring the direct and the inverse kinematics in the first two cases, and a mixed transform in the third case. If one knows the required cases beforehand, one can construct a separate mapping for each. However, it is clearly more desirable to work with a single network that can complete different sets of missing variable values from different sets of known values.

3 PSOMs for Vision Learning

So far, we have been investigating PSOMs for learning tasks in the domain of robotics, since the approach works best for smooth mappings, and if the number of independent degrees of freedom is not larger than about, say, six. In vision, one usually has extremely high-dimensional data, and in many cases, it is doubtful to what extent smoothness assumptions are valid.

Still, there are many situations in which one would like to compute from an image some low-dimensional parameter vector, such as a set of parameters describing location, orientation or shape of an object, or properties of the ambient illumination etc. If the image conditions are suitably restricted, the input images may be samples that are represented as vectors in a very high dimensional vector space, but that are concentrated on a much lower dimensional sub-manifold, the dimensionality of which is given by the independently varying parameters of the image ensemble.

A frequently occurring task of this kind is to *identify and mark a particular part* of an object in an image. For instance, in face recognition it is important to identify the locations of salient facial features, such as eyes or the tip of the nose. Another interesting task is to identify the location of the limb joints of human beings for analysis of body gestures. In the following, we want to report recent results from a third application domain, the identification of finger tip locations in images of human hands. This would constitute a useful preprocessing step for inferring 3D-hand postures from images, and could help to enhance the accuracy and robustness of other, more direct approaches for this task that are based on LLM-networks [9].

Currently, we are working with a restricted ensemble of hand postures. The main degree of freedom of a hand is its degree of "closure", therefore, for the initial experiments we worked with an image set comprising grips in which all fingers are flexed by about the same

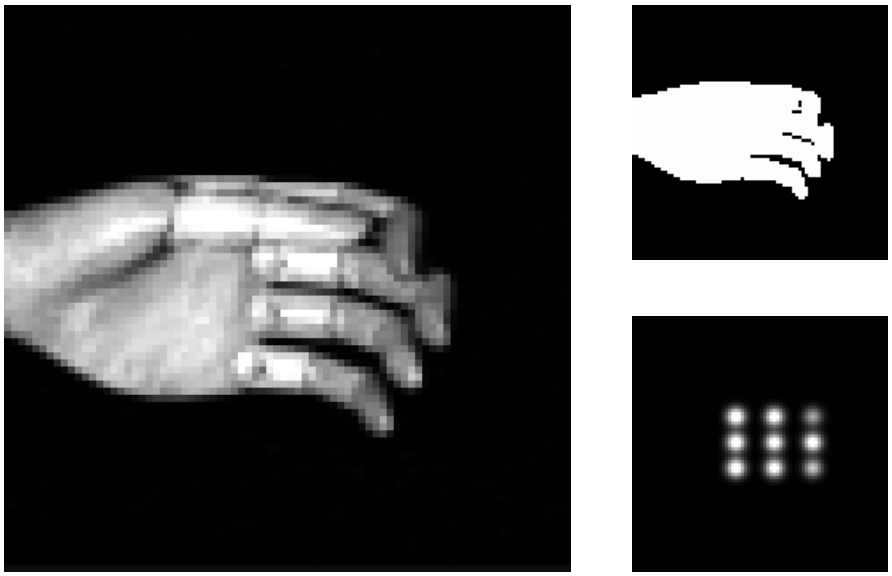

Figure 5: *Left,(a):* Typical input image. *Upper Right,(b):* after thresholding and binarization. *Lower Right,(c):* position of 3×3 array of Gaussian masks (the displayed width is the actual width, reduced by a factor of four in order to better depict the position arrangement)

amount, varying from fully flexed to fully extended. In addition, we consider rotation of the hand about its arm axis. These two basic degrees of freedom yield a two-dimensional image ensemble (i.e., for the dimension m of the map manifold we have $m = 2$). The objective is to construct a PSOM that maps a monocular image from this ensemble to the 2D-position of the index finger tip in the image.

In order to have reproducible conditions, the images were generated with the aid of an adjustable wooden hand replica in front of a black background (for the required segmentation to achieve such condition for more realistic backgrounds, see e.g. [6, 7]). A typical image (80×80 pixel resolution) is shown in Fig. 5a. From the monochrome pixel image, we generated a 9-dimensional feature vector first by thresholding and binarizing the pixel values (threshold = 20, 8-bit intensity values), and then by computing as image features the scalar product of the resulting binarized images (shown in Fig. 5b) with a grid of 9 Gaussians at the vertices of a 3×3 lattice centered on the hand (Fig. 5c). The choice of this preprocessing method is partly heuristically motivated (the binarization makes the feature vector more insensitive to variations of the illumination,) and partly based on good results achieved with a similar method in the context of the recognition of hand postures [7, 9].

To apply the PSOM-approach to this task requires a set of labeled training data (i.e., images with known 2D-index finger tip coordinates) that result from sampling the parameter space of the continuous image ensemble on a 2D-lattice. In the present case, we chose the subset of images obtained when viewing each of four discrete hand postures

(fully closed, fully opened and two intermediate postures) from one of seven view directions (corresponding to rotations in 30°-steps about the arm axis) spanning the full 180^0-range. This yields the very manageable number of 28 images in total, for which the location of the index finger tip was identified and marked by a human observer.

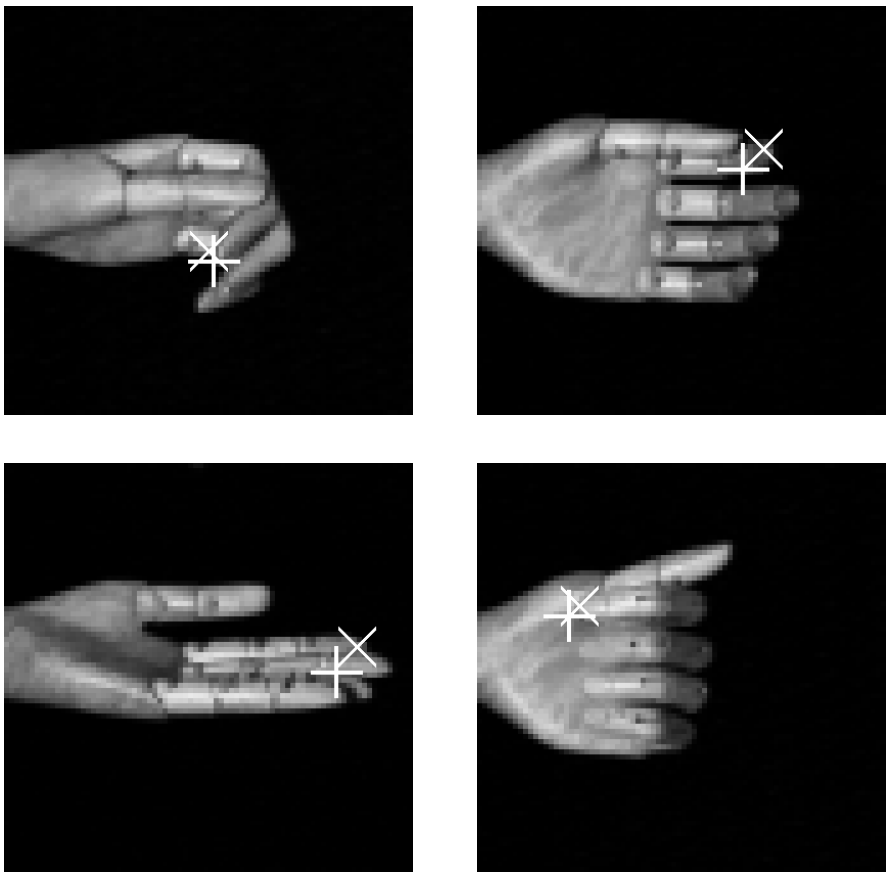

Figure 6: Some examples of hand images with correct (cross-mark) and predicted (plus-mark) finger tip positions. Upper left image shows average case, the remaining three pictures show the three worst cases in the test set. The NRMS positioning error for the marker point was 0.11 for horizontal, 0.23 for vertical position coordinate.

Ideally, the dependency of the x- and y-coordinates of the finger tip should be smooth functions of the resulting 9 image features. For real images, various sources of noise (surface inhomogeneities, small specular reflections, noise in the imaging system, limited accuracy in the labeling process) lead to considerable deviations from this expectation and make the corresponding interpolation task for the network much harder than it would be were the expectation of smoothness fulfilled. Although the thresholding and the subsequent binarization help to reduce the influence of these effects, if compared with computing the feature vector directly from the raw images, the resulting mapping still turns out to be

very noisy. To give an impression of the degree of noise, Fig. 7 shows the dependence of horizontal (x-) finger tip location (plotted vertically) on two elements of the 9d-feature vector (plotted in the horizontal $xy-$plane). The resulting mesh surface is a projection of the full 2d-map-manifold that is embedded in the space X, which here is of dimensionality 11 (nine dimensional input features space $X^{\rm in}$, and a two dimensional output space $X^{\rm out} = (x, y)^T$ for position.) As can be seen, the underlying "surface" does not appear very smooth and is disrupted by considerable "wrinkles".

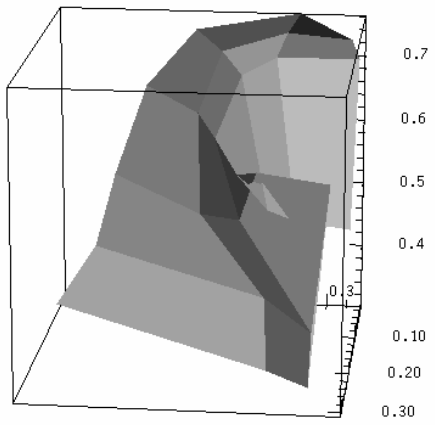

Figure 7: Dependence of vertical index finger position on two of the nine input features, illustrating the very limited degree of smoothness of the mapping from feature to position space.

To construct the PSOM, we used a subset 16 images of the image ensemble by keeping the images seen from the two view directions at the ends ($\pm 90°$) of the full orientation range, plus the eight pictures belonging to view directions of $\pm 30°$. For subsequent testing, we used the 12 images from the remaining three view directions of $0°$ and $\pm 60°$. I.e., both training and testing ensemble consisted of image views that were multiples of $60°$ apart, and the directions of the test images are midway between the directions of the training images.

Even with the very small training set of only 16 images, the resulting PSOM achieved a NRMS-error of 0.11 for the x-coordinate, and of 0.23 for the y-coordinate of the finger tip position (corresponding to absolute RMS-errors of about 2.0 and 2.4 pixels in the 80×80 image, respectively). To give a visual impression of this accuracy, Fig. 6 shows the correct (cross mark) and the predicted (plus mark) finger tip positions for a typical average case (upper left image), together with the three worst cases in the test set (remaining images).

4 Hierarchical Learning with multiple PSOMS

If one wants to learn with extremely few examples, one inevitably faces a dilemma: on the one hand, with few examples one can only determine a rather small number of adaptable

parameters and, as a consequence, the learning system must be either very simple, or, and this is the usually relevant alternative, it must have a structure that is already well-matched to the task to be learnt. On the other hand, however, having to painstakingly pre-structure a system by hand is precisely what one wants to avoid when using a learning approach.

Is is possible to find a workable compromise that can cope with this dilemma, i.e., that somehow allows the structuring of a system without having to put in too much by hand?

One way to approach a solution is to *split learning* into several stages. The earlier stage is considered as *"investment stage"* that may be slow and has the task to pre-structure the system in such a way that the later stage, operating on a now specialized system, can learn *fast* and with extremely few examples.

Of course, this does not bring about the "miracle" to learn with an unstructured system *and* a small number of examples. However, going through a maybe longer pre-structuring stage once and arriving at a pre-structured system which then can be rapidly adapted to specific situations within a narrower (as a result of the pre-structuring phase) domain may be a useful compromise. It also seems to implement an important principle, namely *(i)* investing some time (the earlier stages) into *learning of how to learn* (*investment stage*) and thereby preparing the ground for *(ii) rapid learning* during the later stages.

To implement this hierarchical learning approach we will use the two-level PSOM arrangement depicted in Fig. 8. We consider in the following system "skills" in the form of function mappings or transforms which are dependent on some *context*, i.e. on the state of the system or system environment. Within a fixed context, these transforms are carried out by the lower level "T-PSOM" ("T" indicates "transforming"). For instance, a particular "skill" might be represented by a bi-directional mapping between two "task variable" sets $\vec{x}_1$ and $\vec{x}_2$. The context dependence comes in through the upper level "Meta"-PSOM. This PSOM maps the current context into the *weight set* $\vec{\omega}$ that defines the mapping of the T-PSOM.

To construct this Meta-PSOM requires to find a set of input-output pairs $(\vec{u}_{ref}, \vec{\omega})$ where $\vec{u}_{ref}$ varies over a set of "prototypical context situations", and $\vec{\omega}$ is the weight pattern for the T-PSOM that represents the required "skill" when the context is characterized by $\vec{u}_{ref}$. Thus, we are led very naturally to the construction of a set of prototypical basis mappings, one for each of the prototypical context situations that were chosen for the training points of the Meta-PSOM. The learning of these mappings constitutes the "investment learning" stage.

Once this investment learning phase is completed, the Meta-PSOM allows us to adjust the T-PSOM immediately to a new context since it directly maps contexts into *mappings* between the task variables $\vec{x}_1$ and $\vec{x}_2$. In the absence of the Meta-PSOM, any such adjustment would demand potentially time-consuming re-learning. With the Meta-PSOM, this re-learning becomes tremendously accelerated. The task of learning a mapping in – or adapting to – a new, unknown environment now takes the form of an *immediate* Meta-PSOM mapping that in effect *interpolates in the space of the previously acquired basis mappings* and thereby provides us with the equivalent of an extremely rapid learning ability.

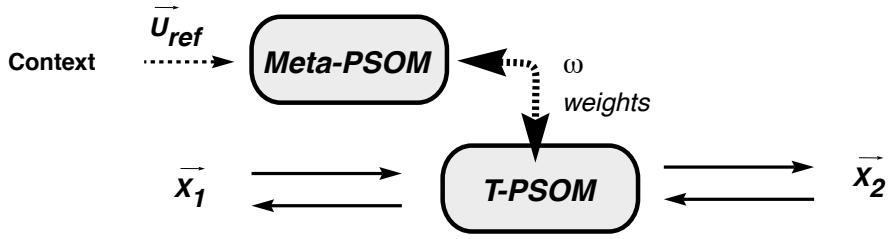

Figure 8: The transforming "T-PSOM" maps between input and output spaces (allowing to switch the mapping direction whenever need arises). In a particular environmental context, the correct transformation is learned, and encoded in the internal parameter or weight set $\vec{\omega}$ of the T-PSOM. Together with some observation $\vec{u}_{ref}$ that characterizes the current context, the weight set $\vec{\omega}$ is employed as a training point $(\vec{u}_{ref}, \vec{\omega})$ for the second level "Meta-PSOM". After a sufficient number of such training points, the Meta-PSOM is able to generalize the mapping $\vec{u}_{ref} \mapsto \vec{\omega}$ to a new environment. When encountering any change, the context observation $\vec{u}_{ref}$ gives input to the Meta-PSOM and determines the new weight set $\vec{\omega}$ for the base T-PSOM.

In the following, we illustrate this approach more concretely with an example situation, involving two learning stages and a robot (Puma 560) that is monitored by a camera, see Fig. 9. The robot is positioned behind a table and the entire scene is displayed on a monitor. By a mouse click, a user can select on the monitor some target point of the displayed table area. The goal is to move the robot end effector to the indicated position on the table. This requires to compute a transformation $T : \vec{x} \leftrightarrow \vec{u}$ between coordinates on the monitor (or "camera retina" coordinates) and corresponding world coordinates $\vec{x}$ in the frame of reference of the robot. This transformation depends on many factors. The learning task (for the later stage) is to rapidly relearn this transformation whenever the camera has been repositioned.

In this example, the T-PSOM has to represent the transformation $T : \vec{x} \leftrightarrow \vec{u}$ with the camera position as the additional context. To apply the previous scheme, we must first learn ("investment stage") the mapping T for a set of prototypical contexts, i.e., camera positionings.

To keep the number of prototype contexts manageable, we eliminate some DOFs of the camera by requiring fixed focal length, camera tripod height, and roll angle. To constrain the elevation and azimuth viewing angle, we choose one fixed land mark, or "fixation point" ξ_{fix} in the scene which we require to be held at a constant location in the camera image. This leaves two degrees of freedom for the camera repositioning. I.e., the set of contexts on which T will depend is a two-parameter family.

For the present investigation, we chose from this set 9 different camera positionings, arranged in the shape of a 3×3 grid (Fig. 9). For each of these nine contexts, the associated mapping $T \equiv T_j$, $(j = 1, 2 \ldots 9)$ is learned by a T-PSOM by visiting a rectangular grid set of end effector positions ξ_i (here we visit a 3×3 grid in $\vec{x}$ of size $30 \times 30\,\mathrm{cm}^2$) jointly with the location in camera retina coordinates (2D) $\vec{u}_i$. This yields the tuples $(\vec{x}_i, \vec{u}_i)$ as the training vectors $\mathbf{w}_{\mathbf{a}_i}$ for the construction of a weight set $\vec{\omega}_j$ (valid for context j) for the

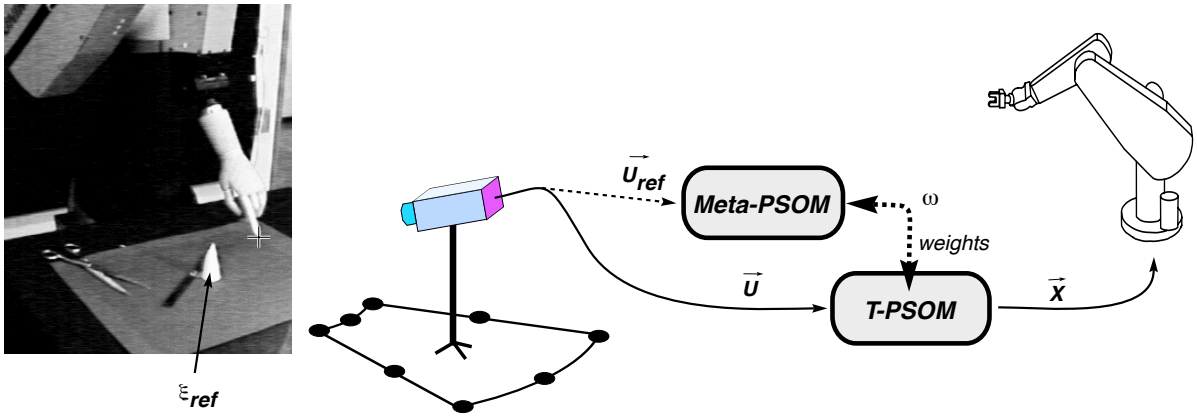

Figure 9: Rapid learning of the 2D visuo-motor coordination for a camera in changing locations. The basis T-PSOM is capable of mapping to (and from) the Cartesian robot world coordinates $\vec{x}$, and the location of the end-effector (here the wooden hand replica) in camera coordinates $\vec{u}$ (see cross mark.) In the pre-training phase, nine basis mappings are learned in prototypical camera locations (chosen to lie on the depicted grid.) Each mapping gets encoded in the weight parameters $\vec{\omega}$ of the T-PSOM and serves then, together with the system context observation $\vec{u}_{ref}$ (here, e.g. the cone tip), as a training vector for the Meta-PSOM.

T-PSOM in Fig. 8.

Each T_j (the T-PSOM in Fig. 8, equipped with weight set $\vec{\omega}_j$) solves the mapping task only for the camera position, for which T_j was learned. Thus there is not yet any particular advantage to other, more specialized methods for camera calibration [2]. However, the important point is, that we now can employ the Meta-PSOM to rapidly map a new camera position into the associated transform T by *interpolating* in the space of the previously constructed basis mappings T_j.

To this end, the Meta-PSOM must map the current context into a suitable combination of the weight patterns of the previously acquired basis mappings T_j (indicated by the dashed arrows in Fig. 8). We already know the required outputs $\vec{\omega}_j$ of the Meta-PSOM for those contexts that were defined by the camera positions used earlier for the construction of the basis mappings T_j during the investment phase.

For the still missing associated inputs, we can choose any observable quantity that carries enough information to uniquely identify the current context. Since we restricted the allowable contexts to a two-parameter family of camera positionings, the observation of the camera image coordinates $\vec{u}_{ref}$ of a single reference point ξ_{ref} fulfils that requirement[2].

The constructed input-output tuples $(\vec{u}_{ref,j}, \vec{\omega}_j)$, $j \in \{1, \ldots, 9\}$, serve as the training vectors for the construction of the Meta-PSOM in Fig. 8 such that each $\vec{u}_{ref}$ observation that pertains to an intermediate camera positioning becomes mapped into a weight vector $\vec{\omega}$ that, when used in the base T-PSOM, yields a suitably *interpolated mapping* in the space

[2]for good position sensitivity, ξ_{ref} should be not too close to ξ_{fix}

spanned by the basis mappings T_j.

This enables in the following *one shot learning* for new, unknown camera places. On the basis of *one single* observation $\vec{u}_{ref}$, the Meta-PSOM provides the weight pattern $\vec{\omega}$ that, when used in the T-PSOM in Fig. 8, provides the desired transformation T for the chosen camera position. Moreover (by using different projection matrices $\mathbf{P}$), the T-PSOM can be used for various mapping directions, formally:

$$\vec{x}(\vec{u}) = F^{u\mapsto x}_{T-PSOM}(\vec{u};\quad \vec{\omega}(\vec{u}_{ref})) \tag{7}$$

$$\vec{u}(\vec{x}) = F^{x\mapsto u}_{T-PSOM}(\vec{x};\quad \vec{\omega}(\vec{u}_{ref})) \tag{8}$$

$$\vec{\omega}(\vec{u}_{ref}) = F^{u\mapsto \vec{\omega}}_{Meta-PSOM}(\vec{u}_{ref};\quad \vec{\omega}_{Meta}) \tag{9}$$

Table 1 shows the experimental results averaged over 100 random locations ξ (from within the range of the training set) seen from 10 different camera locations, from within the 3×3 roughly radial grid of the training positions, located in a normal distance of about 65–165 cm (to work space center, about 80 cm above table, total range of about 95–195 cm), covering a 50° sector. For identification of the positions ξ in image coordinates, a tiny light source was installed at the manipulator tip and a simple procedure automatized the finding of $\vec{u}$ with about ±1 pixel accuracy. For the achieved precision is important that all learned T_j share the same set of robot positions ξ_i, and that the training sets (for the T-PSOM and the Meta-PSOM) are topologically ordered, here as two 3×3 grids. It is not important to have an alignment of this set to any exact rectangular grid in e.g. world coordinates, as demonstrated with the radial grid of camera training positions (Fig. 9.)

	Direct trained T-PSOM		T-PSOM with Meta-PSOM	
pixel $\vec{u} \mapsto \vec{x}$ Cartesian $\Delta\vec{x}$	2.2 mm	2.1 %	3.8 mm	3.6 %
Cartesian $\vec{x} \mapsto \vec{u}$ pixel error	1.2 pix	1.6 %	2.2 pix	2.8 %

Table 1: Mean Euclidean deviation (mm or pixel) and normalized root mean square error (NRMS) in % for 1000 points total in comparison of a direct trained T-PSOM and the described hierarchical PSOM-network, in the rapid learning mode with one observation.

These data demonstrate, that the hierarchical learning scheme does not fully achieve the accuracy of a straightforward, full retraining of the T-PSOM after each camera relocation. This is not surprising, since in the hierarchical scheme there is necessarily some loss of accuracy as a result of the interpolation in the weight space of the T-PSOM. However, the possibility to achieve the still very good accuracy of the hierarchical approach with only a single observation per camera relocation is extremely attractive and may often by far outweight the still moderate decrease that is visible in Table 1.

5 Discussion

For the flexible construction of more complex neural systems, the availability of network modules that can be rapidly constructed from data examples is an important prerequisite. PSOMs can fill this need, whenever smooth mappings need to be implemented on the basis of a small number of data examples.

Like many other neural learning algorithms, the merits of PSOMs can only be judged from studying them in the context of concrete tasks. In this paper, we reported results for a rather generic type of problem frequently occurring in vision, namely to mark the location of an object part in an image. As a concrete and non-trivial example, we chose the task of marking the index finger tip position in images of human hand shapes. The results indicate, that PSOMs may provide a promising approach for the rapid construction of vision modules for such tasks on the basis of rather small numbers of training examples. Thus, they may offer a useful building block also for hybrid vision systems (see, e.g. [7, 6]), where the detection of object part locations is an important step for invoking higher-level semantic information for scene analysis.

A second contribution of the paper is a hierarchical learning scheme that allows a very flexible combination of several PSOMs in such a way that the learning process itself can be structured into a *(i)* longer, preparatory or *"investment learning"* stage that serves to specialize the system for a certain range of operating contexts, and *(ii)* a subsequent, very *rapid learning* stage in which the system can adapt to changing contexts within an extremely small number of learning steps.

The key idea for this approach is to use the investment learning stage for the acquisition of a number of "basis mappings" that solve the task for a sufficient number of prototypical contexts. The weight sets found for these basis mappings are then used for the construction of a second, "Meta" PSOM that uses context information to interpolate in the weight sets of the previously acquired basis mappings.

We have demonstrated the effectiveness of this approach for the task of camera calibration of an industrial Puma 560-Robot, where the "contexts" are different camera positionings. Under suitable conditions, the proposed learning scheme leads to the ability of "one-shot" learning from a single observation of a calibration point whenever the camera has been moved to a new location. This compares favorably to earlier approaches that required a longer re-learning phase in that case [17]. A recent extension of the new approach to the binocular case will be reported elsewhere [15].

Two features that distinguish PSOMs from the majority of other neural learning algorithms are particularly important for the practical feasibility of the hierarchical learning scheme.

First, the mapping implemented by a PSOM is always exact on the examples that were used for its construction. This is very convenient when mappings shall be constructed rapidly and without the worry about convergence issues that plague many other learning algorithms. However, one major restriction is the requirement that the data examples for constructing the PSOM must be "topologically ordered", i.e., one must be able to associate with each data sample $\mathbf{w}$ a point $\mathbf{a}$ in some discrete lattice in a topology-preserving

fashion (otherwise the interpolation by the hyper-surface will lead to unacceptably large generalization errors). However, in many situations the generation of the training points can be controlled by the user, and the required topological organization can be easily assured.

The second important feature is the possibility to use the same PSOM mapping in a "multi-way"-fashion, i.e., the decision of whether a feature of the data vector is used as input or computed as output need not stay fixed, even after the PSOM has been constructed. The only condition to be met is that a sufficiently large number of features is declared as input (via the index set I) such that their values can specify the values for the remaining output features uniquely.

This multiway-mapping capability of a PSOM leads to a considerable flexibility in all its applications. For instance, in the hierachical PSOM arrangement of Fig. 8, the Meta-PSOM actually maps contexts into a continuous familiy of multiway mappings operating at the level of the task variables $\vec{x}_{1,2}$. This already provides an enormous flexibility, since both the roles of the task variables $\vec{x}_1$ and $\vec{x}_2$ as input or output can be flexibly interchanged and even different partitionings into an input and output set can be realized, a possibility not indicated in Fig. 8.

The multiway mapping ability can even be used *across levels in the hierarchy*. Imagine, for instance, that a set of training pairs that pertain to some unknown context $\vec{u}_{ref}$ has been used to obtain a new weight set $\vec{\omega}$ for the lower level T-PSOM. Then, operating the top level Meta-PSOM in the inverse mapping direction $\vec{\omega} \mapsto \vec{u}_{ref}$ will result in an *identification of the current context observable* on the basis of a number of input-output pairs at the task level. Clearly, such flexibility in combining information, together with the very rapid construction of PSOMs opens up extremely interesting possibilities.

Besides the detection of object part locations and camera calibration, there are also many other tasks for which the PSOM approach can offer significant promise. Recently, we have studied the task of learning a highly non-linear kinematics transform of an articulated robot finger from a small number of training postures [16]. In earlier work, we demonstrated the use of a PSOM in an eye-tracking system to compensate distortions that arise when the subject wears spectacles. The quick adaptation of the PSOM to the optical characteristics of the user's spectacles allowed a three-fold increase in the accuracy of gaze-tracking with this system [11].

These results are very encouraging for studying the use of PSOMs in other, similar domains, such as the compensation of lighting variability in computer vision, or the learning of various other kinematics and dynamics transforms in robotics. Work along these lines, together with the development of still more efficient variants of PSOMs through improved knot placing schemes ("Chebyshev PSOMs") and the use of PSOMs in conjunction with local approximation techniques ("local PSOMs") [16] is currently under way.

Appendix A: Choice of Basis Functions

A favorable choice for $H(\mathbf{a}, \mathbf{s})$ is the multidimensional extension of the well known Lagrange polynomial. In one dimension (m=d=1) a is from a set $A = \{a_i \mid i = 1, 2, \ldots n\}$ of discrete

values and Eq. 2 can be written as the Lagrange polynomial, interpolating through n support points (a_i, w_i):

$$w(s) = l_1(s, A)w_1 + l_2(s, A)w_2 + \cdots + l_n(s, A)w_n = \sum_{k=1}^{n} l_k(s, A)\ w_k \tag{10}$$

where the Lagrange factors $l_i(s, A)$ are given by

$$l_i(s, A) = \prod_{j=1,\ j\neq i}^{n} \frac{s - a_j}{a_i - a_j}. \tag{11}$$

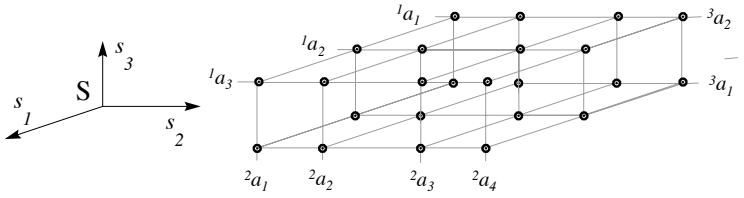

Figure 10: Example of a $m = 3$ dimensional mapping manifold S with a 3×4×2 knot set $\mathbf{A} \subset S$ in orthonormal projection. Note the rectangular structure and the non equidistant spacing of the ${}^{2}a_i$.

If $m > 1$, Eq. 10 and 11 can be generalized in a straightforward fashion, provided the set of knot points form a m-dimensional hyper-lattice in S, i.e. is the Cartesian product of m one-dimensional knot point sets $\mathbf{A} = A_1 \times A_2 \times \cdots \times A_m$, with $A_\nu = \{{}^{\nu}a_1, {}^{\nu}a_2, \ldots, {}^{\nu}a_{n_\nu}\}$. Here, ${}^{\nu}a_i$ denotes the i-th value that can be taken by the ν-th component ($\nu \in \{1, \ldots, m\}$) of $\mathbf{a}$ along the ν-th dimension of $\mathbf{s} = ({}^{1}s, {}^{2}s, \ldots, {}^{m}s)^T \in S$. Fig. 10 illustrates an example with $n_1 = 3$, $n_2 = 4$, and $n_3 = 2$.

With this notation, we can write $H(\mathbf{a}, \mathbf{s})$ as

$$H(\mathbf{a}, \mathbf{s}) = \prod_{\nu=1}^{m} l_{i_\nu}({}^{\nu}s, A_\nu) \tag{12}$$

using the one-dimensional Lagrange factors and the notation $\mathbf{a_i} = ({}^{1}a_{i_1}, {}^{2}a_{i_2}, \ldots, {}^{m}a_{i_m})^T \in \mathbf{A}$. In Fig. 11 some basis functions of a, here, $m = 2$ dimensional PSOM with equidistantly chosen knot spacing, are rendered. Note, that the PSOM algorithm is invariant to offsetting each S axes (together with the iterative best-match finding procedure, described below, it becomes also invariant to rescaling.) Recent comparative results using (for $n > 3$) a knot spacing derived from the Chebyshev polynomial are reported elsewhere [16].

The first derivative of (2) turns out to be surprisingly simple, if we write the product rule in the form

$$\frac{\partial}{\partial x} f_1(x) f_2(x) \cdots f_n(x) = f_1(x) f_2(x) \cdots f_n(x) \sum_{k=1}^{n} \frac{\frac{\partial f_k}{\partial x}(x)}{f_k(x)}, \qquad \forall f_k(x) \neq 0. \tag{13}$$

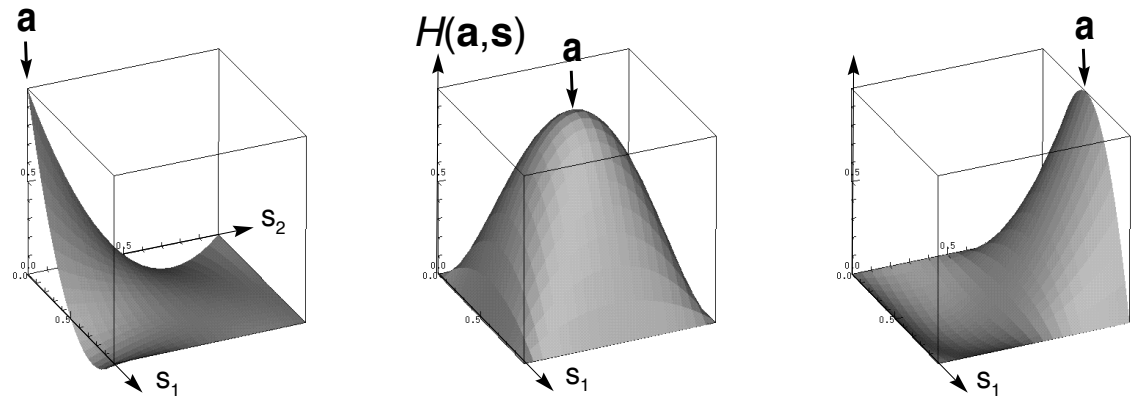

Figure 11: Three of the nine basis functions $H(\mathbf{a},\mathbf{s})$ for a 3×3 PSOM with equidistant knot spacing $A_1 = A_2 = \{0, \frac{1}{2}, 1\}$ *(left:)* $H((0,0),\mathbf{s})$; *(middle:)* $H((\frac{1}{2},\frac{1}{2}),\mathbf{s})$; *(right:)* $H((\frac{1}{2},1),\mathbf{s})$. The remaining six basis functions are obtained by 90° rotations around $\mathbf{s} = (\frac{1}{2},\frac{1}{2})$.

With the intermediate result

$$\frac{\partial}{\partial\, {}^{\mu}s} l_{i_\nu}({}^{\nu}s, A_\nu) = \delta_{\mu\nu} l_{i_\nu}({}^{\nu}s, A_\nu) \sum_{j=1,\ j\neq i_\nu}^{n_\nu} \frac{1}{{}^{\nu}s - {}^{\nu}a_j} \qquad \text{and} \tag{14}$$

$$\frac{\partial}{\partial({}^{\mu}s)} H(\mathbf{a},\mathbf{s}) = H(\mathbf{a},\mathbf{s}) \sum_{\nu=1}^{m} \frac{\frac{\partial}{\partial({}^{\mu}s)} l_{i_\nu}({}^{\nu}s, A_\nu)}{l_{i_\nu}({}^{\nu}s, A_\nu)} = H(\mathbf{a},\mathbf{s}) \sum_{j=1,\ j\neq i_\mu}^{n_\mu} \frac{1}{{}^{\mu}s - {}^{\mu}a_j} \tag{15}$$

the row vector of the required Jacobian matrix is

$$\frac{\partial \mathbf{w}(\mathbf{s})}{\partial({}^{\mu}s)} = \sum_{\mathbf{a}} \mathbf{w}_{\mathbf{a}}\, H(\mathbf{a},\mathbf{s}) \sum_{j=1,\ j\neq i_\mu}^{n_\mu} \frac{1}{{}^{\mu}s - {}^{\mu}a_j}. \tag{16}$$

This is correct for all $\mathbf{s}$ staying away from the dashed grid structure in Fig. 10; there, one (or more) of the product functions f_k (l_i) becomes zero. Although the vanishing denominator is canceled by a corresponding zero in the pre-factor (the derivative of a polynomial is well-behaved everywhere,) the numerical implementation of the algorithm requires special attention in the vicinity of a diminishing denominator in Eq. 14. One approach is, to treat the smallest term $\|{}^{\nu}s - {}^{\nu}a_j\|$ in each S -axis direction ν *always* in a special manner, as shown below.

For an implementation of the algorithm, an important point is the efficient evaluation of the Lagrange factors and there derivatives. Below we give some hints how to improve their computation requiring $O(n)$ operations, instead of $O(n^2)$ operations from the "naive" approach. For the sake of readability we omit from here on the (upper left) S component index ν ($\in \{1, 2, \ldots, m\}$) and reduce the notation of running indexes $1 \leq {}^{\nu}j \leq {}^{\nu}n$ to the short form "j"; extra exclusions $j \neq i$ are written as "$j, -i$".

We denote the index of the closest support point (per axes) with an asterisk

$$i^* := \operatorname*{argmin}_j \|s - a_j\| \tag{17}$$

and rewrite Eq. 11 and (15):

$$
\begin{aligned}
l_i(s,A) &= \begin{cases} Q_{i^*} & \text{if } i = i^* \\ Q_i d_{i^*} & \text{else} \end{cases} \\
\frac{\partial l_i}{\partial s} &= \begin{cases} Q_{i^*} S_{1,i^*} & \text{if } i = i^* \\ Q_i (S_{1,i} d_{i^*} + 1) & \text{else} \end{cases} \\
\frac{\partial^2 l_i}{\partial s^2} &= \begin{cases} Q_{i^*} (S_{1,i^*}^2 - S_{2,i^*}) & \text{if } i = i^* \\ Q_i \left[\left(S_{1,i}^2 - S_{2,i} \right) d_{i^*} + 2 S_{1,i} \right] & \text{else} \end{cases}
\end{aligned} \tag{18}
$$

$$
\begin{aligned}
&\text{using:} && d_i := s - a_i \\
C_i &:= \prod_{j,-i} (a_i - a_j) = const && Q_i := \frac{1}{C_i} \prod_{j,-i,-i^*} d_j \\
S_{1,i} &:= \sum_{j,-i,-i^*} \frac{1}{d_j} && S_{2,i} := \sum_{j,-i,-i^*} \left(\frac{1}{d_j} \right)^2
\end{aligned}
$$

The interim quotients Q_i and sums $S_{1,i}$ and $S_{2,i}$ are efficiently generated by defining the master product Q_{i^*} (and sums $S_{\cdot,i^*}$ respectively) and working out the specific terms via "synthetic division" and "synthetic subtraction" for all $i \neq i^*$

$$
Q_i = \frac{Q_{i^*}}{d_i}, \quad S_{1,i} = S_{1,i^*} - \frac{1}{d_i}, \text{ and } \quad S_{2,i} = S_{2,i^*} - \left(\frac{1}{d_i} \right)^2. \tag{19}
$$

Computing $H(\mathbf{a}, \mathbf{s})$ and its derivatives is now a matter of collecting the proper pre-computed terms. For $m > 1$ and $d > 1$, this formulation is superior to otherwise optimized evaluation schemes, for example the recursive "Neville's algorithm" [12].

Appendix B: Finding the best-match $\mathbf{s}^*$

This section describes, how to solve the iterative minimization problem for solving Eq. 1 efficiently. Eq. 6 approximates gradient decent for the cost function

$$
E(\mathbf{s}) = \tfrac{1}{2} \left[dist\left(\mathbf{x}, \mathbf{w}(\mathbf{s}) \right) \right]^2 = \tfrac{1}{2} \sum_{k=1}^{d} p_k \left[x_k - w_k(\mathbf{s}) \right]^2 \tag{20}
$$

for a given input vector $\mathbf{x}$. Out experience shows that the simple first order gradient descent method given by Eq. 6 is rather sensitive to the proper choice of the step size parameter γ. Therefore we employ the Levenberg-Marquardt algorithm [8, 12] requiring essentially to solve a linear system

$$
\sum_{\nu=1}^{m} \left(\frac{\partial^2}{\partial({}^{\mu}s) \partial({}^{\nu}s)} E + \delta_{\mu\nu} \lambda \right) \Delta s_\nu - \frac{\partial}{\partial({}^{\mu}s)} E = 0. \tag{21}
$$

Since the system can be rather ill-conditioned we employ the "Singular Value Decomposition" algorithm [12] (in double precision) for solving Eq. 21.

Starting with $\mathbf{s}_{t=0} = \mathbf{a}^*$ and $\lambda = 10^{-4}$ the so-called Levenberg-Marquardt parameter λ is adapted dynamically: if the last iteration step resulted in a cost reduction $E(\mathbf{s}_t + \Delta\mathbf{s})$ the step is accepted, $\mathbf{s}_{t+1} = \mathbf{s}_t + \Delta\mathbf{s}$, and λ decreased by a significant factor of, say 10. Otherwise, if the step leads to an increase of cost, then the step will be repeated with $\Delta\mathbf{s}$ based on an augmented λ. The iteration is stopped when the step size drops below a desired threshold.

The Levenberg-Marquardt finds the best-match safely within a couple of iterations, normally more than one order of magnitude faster than the simple steepest gradient algorithm.

References

[1] F. Girosi and T. Poggio. Networks and the best approximation property. *Biol. Cybern.*, 63(3):169–176, 1990.

[2] Gonzalez, Fu, and Lee. *Robotics : Control, Sensing, Vision, and Intelligence*. McGraw-Hill, 1987.

[3] I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New-York, 1986.

[4] T. Kohonen. The self-organizing map. In *Proc. IEEE*, volume 78, pages 1464–1480, 1990.

[5] Teuvo Kohonen. *Self-Organization and Associative Memory*. Springer Series in Information Sciences 8. Springer, Heidelberg, 1984.

[6] F. Kummert, E. Littmann, A. Meyering, S. Posch, H. Ritter, and G. Sagerer. A hybrid approach to signal interpretation using neural and semantic networks. In S.J. Pöppl and H. Handel, editors, *Mustererkennung 1993*, pages 245–252. Springer-Verlag, Berlin, 15. DAGM-Symposium 1993.

[7] F. Kummert, E. Littmann, A. Meyering, S. Posch, H. Ritter, and G. Sagerer. Recognition of 3d-hand orientation from monocular color images by neural semantic networks. *Pattern Recognition and Image Analysis*, 3(3):311–316, 1993.

[8] D. W. Marquardt. *J. Soc Appl. Math.*, 11:431–441, 1963.

[9] A. Meyering and H. Ritter. Learning to recognize 3d-hand postures from perspective pixel images. In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks 2*, pages 821–824, 1992.

[10] John Moody and Christian Darken. Learning with localized receptive fields. In *Proc. Connectionist Models Summer School*, pages 133–143. Morgan Kaufman Publishers, San Mateo, CA, 1988.

[11] M. Pomplun, B. Velichkovsky, and H. Ritter. An artificial neural network for high precision eye movement tracking. In *KI-94: Advances in Artificial Intelligence*, Springer Lecture Notes in Artificial Intelligence 861, pages 63–69. Springer-Verlag, Berlin, 1994.

[12] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Cambridge Univ. Press, 1988.

[13] Helge Ritter. Parametrized self-organizing maps. In S. Gielen and B. Kappen, editors, *ICANN93-Proceedings, Amsterdam*, pages 568–575. Springer Verlag, Berlin, 1993.

[14] Helge Ritter, Thomas Martinetz, and Klaus Schulten. *Neural Computation and Self-organizing Maps*. Addison Wesley, 1992.

[15] Jörg Walter and Helge Ritter. Investment learning with hierarchical PSOM. In *(submitted)*, 1995.

[16] Jörg Walter and Helge Ritter. Local PSOMs and Chebyshev PSOMs – improving the parametrised self-organizing maps. In *ICANN (submitted)*, 1995.

[17] Jörg Walter and Klaus Schulten. Implementation of self-organizing neural networks for visuo-motor control of an industrial robot. *IEEE Transactions in Neural Networks*, 4(1):86–95, 1993.

[18] L. Zadeh. Fuzzy logic, neural networks, and soft computing. *Communications of the ACM*, 37(3):77–86, 1994.