Investment Learning with hierarchical PSOM

Jörg Walter and **Helge Ritter**
Department of Information Science
University of Bielefeld, D-33615 Bielefeld,
Email: {walter,helge}@techfak.uni-bielefeld.de

June 8, 1995

Abstract

The recently introduced *"Parameterized Self-Organizing Maps"* ("PSOM") shows excellent function mapping capabilities after learning of a remarkable *small* set of training data. This is a very important feature in fields where the acquisition of training data is costly, for example in robotics. As a first demonstration, we compare results for the task of kinematic mapping of a 3-DOF robot finger, obtained by a PSOM and a standard backprop network.

A new way of structuring learning becomes feasible: following the idea of interpolating basis mappings learned for a small set of special circumstances, we decompose learning into two phases:
(i) In the first *investment learning* phase we pre-train a hierarchical PSOM-network with a set of basis mappings, each capturing a prototypical situation or system context.
(ii) Then in the second phase, the mapping "skill" adapts very *rapidly*, when the system context changes to new, unknown situations.

In this paper we demonstrate the potential of this approach for the task of a 3D visuo-motor map for a Puma robot and two independent movable cameras. This includes the forward and backward robot kinematics in 3D Cartesian end effector coordinates, the 2D+2D retina coordinates and also the 6D joint angles. After the *investment phase* the correct transformation can be learned in a new camera set-up with a *single* observation.

1 Introduction

Most current applications of neural network learning algorithms suffer from a large number of required training examples. This may not be a problem when data are abundant, but in many application domains, such as e.g. robotics, training examples are costly and the benefits of learning can only be exploited when significant progress can be made within a very small number of learning examples. That such performance can be achieved within the paradigm of statistical learning and on the basis of the widely adopted multilayer-percectrons seems unlikely in view of the massive research carried out along these lines during the last few years.

In the present contribution, we propose in Sec. 3 an hierarchical structured learning approach which can be applied to many learning tasks that require system identification from a limited set of observations. The idea build on the recently introduced *"Parameterized Self-Organizing Maps"* ("PSOM") [10, 11, 14], whose strength is learning maps from a small number training data. Before giving a short mathematical description in Sec. 2, we'd like to demonstrate this briefly by an example:

Fig.1 depict the PSOM application to kinematic of robot finger with 3 degrees-of-freedom (DOF). The mechanical design roughly allows the mobility of the human index finger, scaled up by about 10 % (details can be found in [13].)

Due to the limited space, we describe this task here as a abstract, non-linear mapping, and try to visualize the essentials. The learning goal is the proper correspondence of a 3D rectangular grid (joint angles) and the "banana" like grid structure (1b) (finger tip in Cartesian space; confirm this workspace with your finger!) on the basis of a 3×3×3 training set rendered in (1c). The resulting (inverse kinematic) mapping when calling for a grid test set (1b) is the almost rectangular structured (of the resulting joint angle) set (1d). The interesting measure for the mapping performance is here (the Cartesian positioning error) the deviation of the back-projected result (1e) from the original desired set (1b).

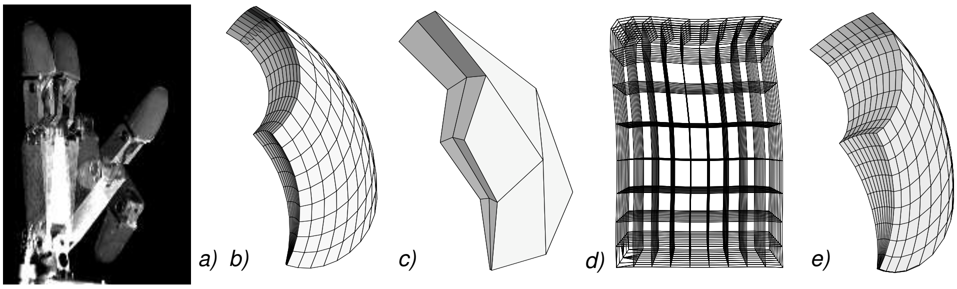

Figure 1: Kinematic mapping example of a 3 DOF robot finger. *(a)* Stroboscopic image of one finger in a sequence of extreme joint positions. *(b)* Perspectives of the workspace envelope, tracing out a cubical 10×10×10 grid in the full joint angle space $\vec{\theta}$. *(c)* A 3×3×3 grid, serving a training set. (Following the connecting lines allows one to verify that the "banana" really possesses a cubical grid structure.) *(d)* PSOM inverse kinematic result using the grid test set displayed in (b). *(e)* True work space after back transforming set (d). We measure the mean Cartesian deviation (b-e) here as 1.6 mm or **1.0 %** of the maximum workspace length of 160 mm.

	ϵ_i	ϵ_f	$n = 3$	$n = 4$	$n = 5$
BP-net 3–50–3	0.02	0.004	72 %	57 %	54 %
BP-net 3–100–3	0.01	0.002	86 %	64 %	51 %
PSOM	–	–	6.2 %	3.7 %	0.4 %

Table 1: Normalized root mean square error (NRMS) for a training set of $n{\times}n{\times}n$ points, obtained by the two best performing standard MLP networks (out of 12 different achitectures, with various (linear decreasing) step size parameter schedules ϵ) 100000 stepest gradient descent steps were performed for the MLP-net and one pass through the set for PSOM network.

Visual inspection of (1e) and (1b) shows a very good approximation of the desired, highly non-linear mapping. In view of the extremely minimal training set of 3×3×3 in (1c), this appears to be a quite remarkable result.

We recently compared another standard network type, the well-known back-prop net, in a one and two hidden layer (of `tanh()` type) configuration (with output layer of linear unit type.) We found that this class of problems is not suitable for the BP-network. Even for larger training set sizes, we didn't succeed in training them to a performance comparable to the PSOM network, Table 1 shows the result of two of the best BP-nets compared to the PSOM.

How does the PSOM work? In the next section we explain the algorithm, albeit in a condensed form. For details see [10, 11, 13].

2 PSOMS

A *Parameterized Self-Organizing Map* is a parametrized, m-dimensional hypersurface $M = \{\mathbf{w}(\mathbf{s}) \in X \subseteq \mathbb{R}^d | \mathbf{s} \in S \subseteq \mathbb{R}^m\}$ that is embedded in some higher-dimensional vector space X. M is used in a very similar way as the standard discrete self-organizing map: given a distance measure $dist(\mathbf{x}, \mathbf{x}')$ and an input vector $\mathbf{x}$, a best-match location $\mathbf{s}^*(\mathbf{x}) \in S$ is determined by minimizing

$$E(\mathbf{s}; \mathbf{x}) = dist(\mathbf{x}, \mathbf{w}(s)). \tag{1}$$

The associated "best-match vector" $\mathbf{w}(\mathbf{s}^*)$ provides the best approximation of input $\mathbf{x}$ in the manifold M. If we require $dist()$ to vary only in a subspace X^{in} of X (i.e., $dist(\mathbf{x}, \mathbf{x}') = dist(\mathbf{Px}, \mathbf{Px}')$, where $\mathbf{P}$ projects into X^{in}), $\mathbf{s}^*(x)$ actually will only depend on $\mathbf{Px}$. Then the projection $y(x) = (\mathbf{1} - \mathbf{P})\mathbf{w}(\mathbf{s}^*(\mathbf{x})) \in X^{\text{out}}$ of $\mathbf{w}(s^*(\mathbf{x}))$ that lies in the orthogonal subspace $Xout$ can be viewed as a (non-linear) *associative completion of a fragmentary input* $\mathbf{x}$ *of which only the part* $\mathbf{Px}$ *is reliable.* It is this associative mapping that we will exploit in applications of the PSOM.

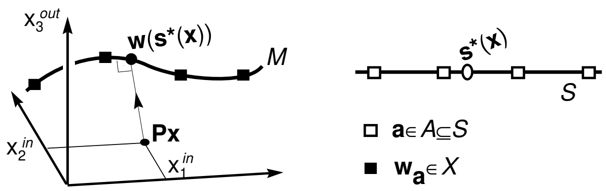

Figure 2: *(left)* Associative completion $\mathbf{w}(\mathbf{s}^*(\mathbf{x}))$ of input $\mathbf{x}$ ($\mathbf{x} \in X = X^{\text{in}} \times X^{\text{out}} = \mathbb{R}^d$ with dependable components $\mathbf{Px} \in X^{\text{in}}$ but errant components in X^{out}) by best match approximation $\mathbf{w}(\mathbf{s}^*(\mathbf{x}))$ on the PSOM-manifold M. Here the $m = 1$ dimensional M is constructed to pass through four data vectors (square marked). *(right:)* The best match parameter $\mathbf{s}^*(\mathbf{x})$ in the parameter manifold S together with the "hyper-lattice" $\mathbf{A}$ of parameter values (indicated by white squares) belonging to the data vectors.

M is constructed as a manifold that passes through a given set D of data examples (Fig. 2 depicts the situation schematically). To this end, we assign to each data sample a point $\mathbf{a} \in S$ and denote the associated data sample by $\mathbf{w_a}$. The set $\mathbf{A}$ of the assigned parameter values should provide a good discrete "model" of the topology of our data set (Fig. 2 right). The assignment between data vectors and points $\mathbf{a}$ must be made in a topology preserving fashion to ensure good interpolation by the manifold M that is obtained by the following steps.

For each element $\mathbf{a} \in \mathbf{A}$, we construct a "basis function" $H(\cdot, \mathbf{a}) : S \mapsto \mathbb{R}$ that obeys $H(\mathbf{a}, \mathbf{a}) = 1$ and vanishes on all other points of A (we will mainly be concerned with the case of $\mathbf{A}$ being a m-dimensional hyper-lattice; in this case, the functions $H(\cdot, \mathbf{a})$ can be constructed as Lagrange interpolation polynomials, see [14]). Then,

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

defines a manifold M that passes through all data examples. Minimizing E in Eq. 1 can be done by some iterative procedure, such as gradient descent or – preferably – the Levenberg-Marquardt algorithm. This makes M into the attractor manifold of a (discrete time) dynamical system. Since M contains the data set D, any at least m-dimensional "fragment" of a data example $\mathbf{w} \in D$ will be attracted to the correct completion $\mathbf{w}$. Any other inputs will be attracted to some interpolating manifold point.

This approach is in many ways the continuous analog of the standard discrete self-organizing map. Particularly attractive features are (i) that the construction of the map manifold is direct from a *small set* of training vectors, without any need for time consuming adaptation sequences, (ii) the capability of associative completion, which allows to freely redefine variables as inputs or outputs (by changing $\mathbf{P}$ on demand), and (iii) the possibility of having *attractor manifolds* instead of just attractor points.

3 Hierarchical PSOMs: Structuring Learning

If one wants to learn with extremely few examples, one inevitably faces a dilemma: one 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 learned. 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 two stages. *(i)* The earlier stage is considered as an *"investment stage"* that may be slow and that may require a larger number of examples. It has the task to pre-structure the system in such a way that in the later stage, *(ii)* the now specialised 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.

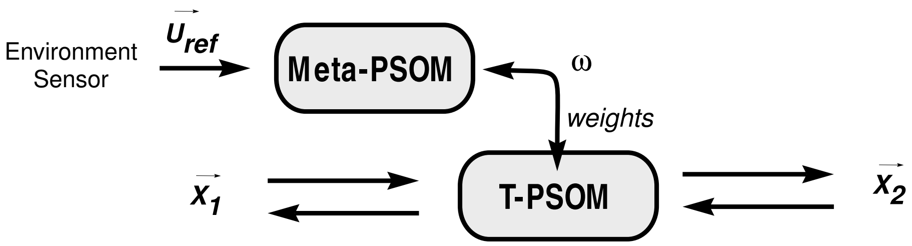

Figure 3: The transforming "T-PSOM" maps between input and output spaces (changing direction on demand). In a particular environmental context, the correct transformation is learned, and encoded in the internal parameter or weight set ω. Together with an characteristic environment observation $\vec{u}_{\rm ref}$, the weight set ω is employed as a training vector for the second level "Meta-PSOM". After learning a structured set of mappings, the Meta-PSOM is able to generalizing the mapping for a new environment. When encountering any change, the environment observation $\vec{u}_{\rm ref}$ gives input to the Meta-PSOM and determines the new weight set ω for the basis T-PSOM.

Concrete, we consider specialized mappings or "skills", which are dependent on the state of the system or system environment. Pre-structuring the system is

achieved by learning a set of basis mappings, each in a prototypical system context or environment state (investment phase.) This imposes a strong need for an efficient learning tool – efficient in particular with respect to the number of required training data points.
The PSOM networks appears as a very attractive solution: Fig. 3 shows a hierarchical arrangement of two PSOM. The basis task of mapping from (possibly various) input to output spaces is learned – and performed, by the "Transformation-PSOM" ("T-PSOM").

During the first learning stage, the *investment learning* phase, a set of basis mappings T_j or context dependent "skills" is constructed in the "T-PSOM", which gets encoded in the internal parameter or "weight" set ω. The second level PSOM is responsible for learning these entire parameter sets ω of the first level T-PSOM, therefore called "Meta-PSOM".

The context situations are chosen such that the associated basis mappings capture already a significant amount of the underlying model structure, while still being sufficiently general to capture the variations with respect to which system environment identification is desired. The system context is characterized by a suitable environment observation, denoted $\vec{u}_{ref}$, see Fig.3.

For each of the prototypical system environment situations j the constructed T-PSOM weight set ω_j serves together with the environment observation $\vec{u}_{ref,j}$ as a high dimensional training data vector for the second level Meta-PSOM.

Rapid learning is the return on investment in the longer pre-training phase. The task of learning of an unknown system environment situation now takes the form of an *immediate* Meta-PSOM mapping. Then the Meta-PSOM maps the new system context observation $\vec{u}_{ref,new}$ into the parameter set ω_{new} for the T-PSOM, encoding the desired mapping T_{new}.

4 Rapid Visuo-motor Coordination Learning

In the following, we demonstrate the potential of hierarchical PSOMs with the task of fast learning of 3D visuo-motor maps for a robot manipulator seen by a pair of movable cameras. The Puma robot is positioned behind a table and the entire scene is displayed on two windows on a computer monitor. By mouse-click, a user can, for example, select on the monitor one point and the position on an (in the other window) appearing line and the goal is to move the robot end effector tip to the indicated world position, see Fig. 4. This requires to compute the transformation T between pixel coordinates $\vec{u} = (\vec{u}^L, \vec{u}^R)$ on the monitor images and corresponding world coordinates $\vec{x}$ in the robot reference frame – or alternatively – the corresponding six robot joint angles $\vec{\theta}$ (6 DOF). Here we demonstrate an integrated solution, solving both simultaneously.

The T-PSOM learns each individual basis mapping T_j by visiting a rectangular grid set of end effector positions ξ_i (here we visit a 3×3×3 grid in $\vec{x}$ of size $40 \times 40 \times 30\,\mathrm{cm}^3$) jointly with the joint angle tuple $\vec{\theta}_j$ and the location in camera retina coordinates (2D in each camera) $\vec{u}_j^L, \vec{u}_j^R$. Thus the training vectors $\mathbf{w}_{\mathbf{a}_i}$ for the construction of the T-PSOM are the tuples $(\vec{x}_i, \vec{\theta}_i, \vec{u}_i^L, \vec{u}_i^R)$.

However, each T_j solves the mapping task only for the current camera position pair, for which T_j was learned. Thus there is not yet any particular advantage to other, specialized methods for camera calibration [4]. The important point is, that we now employ the Meta-PSOM to *interpolate* in the space of mappings.

To keep the number of prototype mappings manageable, we reduce some DOF of the cameras by requiring fixed focal length, camera tripod height, and twist joint. To constrain the elevation and azimuth viewing angle, we choose one land mark, or "fixation point" ξ_{fix} somewhere centered in the region of interest. After

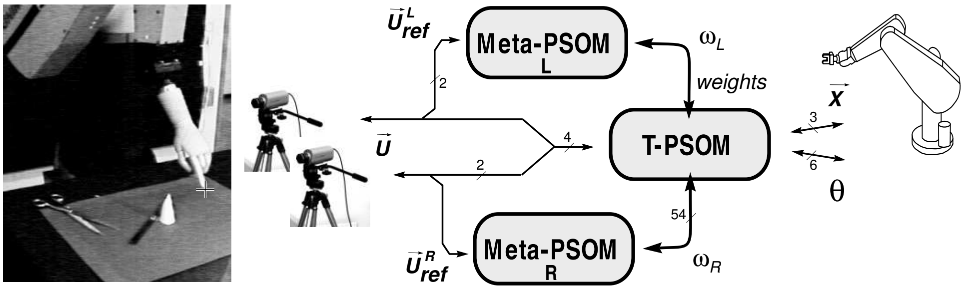

Figure 4: Rapid learning of the 3D visuo-motor coordination for two camera in changing locations. The basis T-PSOM ($m = 3$) is capable of mapping to and from three coordinate systems: the Cartesian robot world coordinates (Puma manipulator), the robot joint angles (6-DOF), and the location of the end-effector (here the wooden hand replica) in coordinates of the two camera retinas (see left camera view with cross mark). Since the left and right camera change tripod place independently, the weight set of T-PSOM is split, and parts ω_L, ω_R are learned in separate Meta-PSOMs.

repositioning any camera, its viewing angle must be re-adjusted, to keep this fixation point visible in a constant image position, serving at the same time the need of a fully visible region of interest. These practical instructions achieve the reduction of free parameters per camera to the 2D lateral position, which can now be sufficiently determined by *one* extra observation of a chosen auxiliary world reference point ξ_{ref}. We denote the pixel coordinates, where ξ_{ref} becomes visible, by $\vec{u}_{ref} = (\vec{u}^L_{ref}, \vec{u}^R_{ref})$. By reuse of the cameras as "environment sensor", $\vec{u}_{ref}$ does implicitly encode the camera positions.

In the investing pre-training phase, nine mapping T_j are learned by the T-PSOM, each camera varying place on a 3×3 grid, sharing the set of visited robot positions ξ_i. Since the left and right camera change tripod place independently, the weight set of T-PSOM is split, and only parts are learned in two separate Meta-PSOMs. I.e. each training vector j for the left camera Meta-PSOM consists of ($\vec{u}^L_{ref}$ and T-PSOM weight set part $\omega_L = (\vec{u}^L_1, \ldots, \vec{u}^L_{27})$, analogous for the right Meta-PSOM.

This enables in the following the *rapid learning* phase for new, unknown camera places. On the basis of *one single* observation $\vec{u}_{ref}$, the desired transformation T is constructed. As visualized in Fig. 4, $\vec{u}_{ref}$ serves as the input to the second level Meta-PSOMs. There output are interpolations in-between previously learned parameter sets and they project directly into the weight set of the basis level T-PSOM.

After the rapid learning step the weight set the transforming T-PSOM can map various directions (by using different projection matrices $\mathbf{P}$), e.g.:

$$\vec{x}(\vec{u}) = F^{u\mapsto x}_{T-PSOM}(\vec{u};\quad \omega_L(\vec{u}^L_{ref}), \omega_R(\vec{u}^R_{ref})) \tag{3}$$

$$\vec{\theta}(\vec{u}) = F^{u\mapsto \theta}_{T-PSOM}(\vec{u};\quad \omega_L(\vec{u}^L_{ref}), \omega_R(\vec{u}^R_{ref})) \tag{4}$$

$$\vec{u}(\vec{x}) = F^{x\mapsto u}_{T-PSOM}(\vec{x};\quad \omega_L(\vec{u}^L_{ref}), \omega_R(\vec{u}^R_{ref})) \tag{5}$$

$$\omega_L(\vec{u}^L_{ref}) = F^{u\mapsto \omega}_{Meta-PSOM,L}(\vec{u}^L_{ref};\quad \Omega_L); \quad \text{analog } \omega_R(\vec{u}^R_{ref}) \tag{6}$$

Table 2 shows the experimental results averaged over 100 random locations ξ (from within the range of the training set) seen in 10 different camera set-ups, from within the 3×3 square grid of the training positions, located in a normal distance of about 125 cm (center to work space center, 1 m^2, total range of about 55–210 cm), covering a disparity angle range of 25°–150°. 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 ± 0.8 pixel accuracy. For the achieved precision is important to share the same set of robot positions ξ_i, and that the sets are topologically ordered, here as a 3×3×3 (i) and two 3×3 (j) grids. It is not important to have an alignment of this set to any exact rectangular grid in space, e.g., some radial grid of camera training positions is perfectly ok.

	Direct trained T-PSOM		T-PSOM with Meta-PSOM	
pixel $\vec{u} \mapsto \vec{x}$ Cartesian $\Delta\vec{x}$	1.4 mm	0.8 %	4.4 mm	2.5 %
Cartesian $\vec{x} \mapsto \vec{u}$ pixel error	1.2 pix	1.0 %	3.3 pix	2.5 %
pixel $\vec{u} \mapsto \vec{\theta}$ commanded $\mapsto \Delta\vec{x}$	3.8 mm	2.3 %	5.4 mm	3.0 %

Table 2: 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.

5 Discussion and Conclusion

PSOM exhibit good generalization capabilities, even for highly non-linear function mappings and with remarkably small training sets. The reason for this is that PSOMs combine the benefits of local models [2, 6, 8, 7, 1] with the ability of self-organizing maps [5, 12] to use topology information as an additional constraint for forming an interpolating hyper-surface through given data points. Due to their continuous formulation of the map manifold M, PSOMs allow to exploit this additional benefit even for data sets with very few elements, i.e., precisely for those cases, where the use of additional information of how to interpolate is particularly valuable.

For the PSOM, the additional topology information is provided with the "model" **A** of the topological neighborhood relationships among the data samples. This "topology model" provides additional curvature information, information which is not available within other techniques, such as e.g. radial basis functions [9, 3].

Since this allows the construction of PSOMs from very small data sets (by a direct, non-iterative process that yields exact values on the samples themselves!), PSOMs offer versatile "building blocks" for modular systems.

A crucial question is how to structure such systems by learning. In the present paper, we demonstrated a hierarchical approach that is motivated by a *decomposition of the learning phase into two different stages:* A longer initial learning phase "invests" effort into a gradual and domain-specific specialization of the system. This *investment learning* does not yet produce the final solution, but instead pre-structures the system such that the subsequently final specialization to a particular solution within the chosen domain can be achieved extremely rapidly.

Technically, the investment learning phase is realized by learning a set of *prototypical basis mappings* (implemented as PSOMs) that attempt to cover the range of tasks in the given domain. The capability for subsequent rapid specialization within the domain is then provided by an additional mapping that maps a set of observations that is sufficient to characterize a task instance into a suitable combination of the previously learned prototypical basis mappings. The construction of this additional mapping again is solved with a PSOM ("Meta"-PSOM) that *interpolates in the space of prototypical basis mappings* that were constructed during the "investment phase".

We demonstrated the potential of this approach with the task of 3D visuo-motor mapping, learnable with a single observation after repositioning a pair of cameras.

When comparing the distance range 0.5–2.1 m of exercised positions to the positioning capabilities after learning by a single observation, the achieved accuracy of 4.4 mm is very satisfying.

The presented arrangement of a basis T-PSOM and two Meta-PSOMs demonstrates further the possibility to split hierarchical learning in independently changing domain sets. When the number of involved free context parameters is growing, this factorization is increasingly crucial to keep the number of pre-trained prototype mappings manageable.

References

[1] C.G. Atkeson. Memory-based approaches to approximating continuous functions. In M. Casdagli and S. Eubank, editors, *Nonlinear Modeling and Forecasting*, volume XII of *SFI Studies in the Sciences of Complexity*, pages 503–521. Addison-Wesley, New York, 1992.

[2] W.S. Cleveland, S.J. Devlin, and E. Grosse. Regression by local fitting: methods, properties, and computational algorithms. *Nonlinear Modeling and Forecasting, Journal of Econometrics*, 37:87–114, 1988.

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

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

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

[6] John Moody and Christian Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294, 1989.

[7] P. Omohundro. Bumptrees for efficient function, constraint, and classification learning. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 693–699. Morgan Kaufman Publishers, San Mateo, CA, 1991.

[8] John Platt. A resource-allocating network for function interpolation. *Neural Computation*, 3:213–255, 1991.

[9] M.J.D. Powell. *Radial basis functions for multivariable interpolation: A review*, pages 143–167. Clarendon Press, Oxford, 1987.

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

[11] Helge Ritter. Parametrized self-organizing maps for vision learning tasks. In *ICANN'94-Proceedings, Italy*, pages 803–810, 1994.

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

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

[14] Jörg Walter and Helge Ritter. Rapid learning with parametrized self-organizing maps. *Neurocomputing, Special Issue*, (submitted), 1995.