Neural Network Approaches for Perception and Action

Helge Ritter

Technical Faculty, Bielefeld University, Bielefeld D-33501, Germany

Abstract. We argue that endowing machines with perception and action raises many problems for which solutions derived from first principles are unfeasable or too costly and that artificial neural networks offer a worthwhile and natural approach towards a solution. As a result, learning methods for creating mappings or dynamical systems that implement a desired functionality replace the implementation of algorithms by programming. We discuss two neural network approaches, the Local Linear Maps (LLMs) and the Parametrized Self-Organizing Maps (PSOMs), that are well suited for the rapid construction of mapping modules and illustrate some of their possibilies with examples from robot vision and manipulator control. We also address some more general issues, such as the need for mechanisms of attention control and for the flexible association of continuous degrees of freedom, together with an ability for handling varying constraints during the selection of actions.

1 Introduction

Neural networks were evolved by nature to enable perception and action. Therefore, artificial neural networks seem as an appropriate approach for building robots that can perceive and that can meaningfully coordinate their actions according to their sensory perception.

Before the advent of artificial neural networks, traditional robotics research had focused on the use of sophisticated, explicit world models, which have to be be built from sensory data and on which then algorithms operate to derive the robot's actions [1]. While this is a clear-cut and very powerful approach whenever it can be carried through, it may be simply too ambitious for many tasks a robot has to carry out. In the domain of robot vision, the emphasis of explicit models has attracted a lot of research with the goal of scene reconstruction. While a 3d representation of its environment undoubtedly is very useful for a robot, many useful behaviors can be carried out with much less information that may be much easier to extract from images than first going through a full geometric representation. This has been demonstrated impressively by the approach of "behavior based robotics", which in its more radical forms even denies the necessity any explicit world models and instead attempts to engineer the behavior of a robot from a suitably interacting set of simple sensory reflexes or reactive behaviors, possibly augmented by some simple memory scheme (for a collection of representative papers, see, e.g. [15]). In addition, by viewing vision as intrinsically embedded in a

perception-action cycle, many vision tasks become significantly changed and often easier, due to the availability of more data and the possibility to actively control them.

If we abandon the traditional, algorithm-oriented approach and instead use neural networks as our main tool, several new issues arise. First, there is a shift from constructing algorithms to the construction of *mappings* and of *dynamical systems* that solve a particular task or that contribute to a desired behavior.

At the lowest level, we need mappings that can form representations from the multitude of sensory signals that must be processed by any perceiving robot. In biological brains, the bulk of these operations are carried out in the primary sensory cortices. These are organized topographically, forming dimension-reduced representations of the sensory signals in the form of activity patterns on two-dimensional arrays of feature selective neurons. The spatial structure of these topographically organized "feature maps" can be modeled [20,3]. to a surprising extent with Kohonen's Self-Organizing Map ("SOM") algorithm ([12,25]; for brief surveys see, e.g., [11,28]) Using this approach, it has become possible not only to model many intriguing properties of topographic brain maps, but also to create artificial feature maps for a variety of computational tasks in pattern recognition, data analysis and visualization, process control, data mining and robotics ([12] gives comprehensive references).

Originally, SOMs were developed to model some initial steps in the organization of the perceptual apparatus in the brain, namely the adaptive creation of dimension-reduced mappings of various sensory spaces. However, it is equally possible to include in the mapped stimuli also features that are related to action, such as parameters of motor commands. This leads to "sensory-motor-maps" that link perception with action [23]. Such maps are known to exist at least in some places in the brain, the best-known example being the collicular map where maps of retinal and of auditory space are in register with a topographic map of motor signals required to perform a saccade to a stimulus that elicited activity in one of the sensory maps.

Very similar maps can be created with the SOM algorithm [25] and work along these lines has shown that sensori-motor SOMs can, e.g., be used to learn hand-eye coordination for robots [24].

Unlike traditional algorithms, most knowledge that is represented in neural networks is encoded in a non-symbolic form, distributed over a large number of "synaptic weights". As a consequence, both our burden and our freedom to specify a desired behavior in terms of a symbolic program is greatly diminished. Programming largely becomes replaced by *learning algorithms* for constructing the necessary mappings or the desired dynamical behavior.

On the one side, this is an appealing feature, since it allows us to create systems that require only a very modest amount of explicit knowledge. Most information can be provided in *implicit* form, by providing a sufficient num-

ber of training examples. On the other hand, we then have to face a very important new issue, namely the task to develop methods for the *rapid construction of neural modules* from limited sets of examples. This is particularly important in robotics, since robot actions are rather costly and, say, a few ten thousand training examples are in most cases a limit beyond which robot learning usually becomes unattractive.

Below, we will present two approaches that have both been motivated by the SOM and that adress this issue. The first approach, the *Local Linear Map* (LLM-) networks, improves the learning capabilities of the basic SOM for smooth mappings by associating with each neuron a locally valid, linear mapping [27]. In differential geometric terms, this can be viewed as constructing for a manifold that is approximated by a discrete SOM a cross section of its tangent bundle.

The second approach, the *Parametrized Self-Organizing Map* [26,32], goes one step further and replaces the discrete representation of the SOM by a differentiable manifold that is constructed with the help of a set of basis functions. A significant amount of flexibility of this approach comes from a generalization of the discrete SOM bestmatch search into the continuous domain, yielding the functionality of a *continuous associative memory.*

Both approaches provide us with convenient functional building blocks for the construction of larger systems. However, before we can use them to build more comprehensive systems, we must consider as an important third issue the question of a suitable *architecture* that can coordinate the cooperation of multiple modules.

One important principle of coordination apparently is *focal attention.* The ability of focal attention is a characteristic of most living cognitive systems, and its implementation in artificial robots is an essential means for coping with the high demands of processing real world multimodal sensory data in real time. Besides the economic aspect of concentrating and thereby making more efficient use of processing resources, a further important aspect of of focal attention is the ability *to ignore the unimportant.* In the overwhelming majority of tasks, the meaningful coordination of action usually requires the coordination of a comparably small number – if compared to the dimensionality of the sensory input – of degrees of freedom. It is the task of our perception system to extract these few relevant degrees of freedom from the almost infinite dimensional visual, auditory, tactile and proprioceptive sensory inputs and to maintain a stable binding to these relevant degrees of freedom during the execution of a task. This, however, requires also to reliably ignore the influence of the majority of sensory inputs. Below, we will describe one particular approach how visual focal attention can be learnt within a hierarchical system of neural networks by an approach that may be described as "neural zooming". The architecture behind this approach is, however, purely computationally motivated and has not been developed with any deeper cognitive modeling in mind.

2 Local Linear Maps

While the discrete nature of a SOM poses no difficulty for tasks such as classification or data analysis and visualization, it becomes a problem when we want to use SOMs to represent sensory-motor mappings. Here, we usually require smoothness, and, in addition, the underlying spaces are frequently higher than just two-dimensional. While we can increase the degree of smoothness by using more nodes, and it is in principle also straightforward to extend the SOM algorithm to higher-dimensional situations, both measures together very rapidly exhaust any reasonable amount of computational resources (providing only 10 discretization steps for a mapping with 6 DOFs would require already the substantial number of 10^6 nodes!). A simple though effective way to deal with this problem is to attach to each SOM node $\mathbf{r}$ a locally valid linear mapping that is used to compute a correction to the node's output reference vector $\mathbf{w}_\mathbf{r}^{(out)}$ that is linear in the deviation of the input $\mathbf{x}$ from the prototype vector $\mathbf{w}_\mathbf{r}^{(in)}$ of the best-matching node $\mathbf{s}$. We then obtain for the output $\mathbf{y_r}$ computed by a node $\mathbf{r}$ the equation

$$\mathbf{y_r} = \mathbf{w}_\mathbf{r}^{(out)} + \mathbf{A}_r(\mathbf{x} - \mathbf{w}_\mathbf{r}^{(in)}). \tag{1}$$

where $\mathbf{A}$ is a $M \times L$ matrix and L and M denote the dimensionalities of the input and of the output space, resp.

Mathematically, the mapping $\mathbf{r} \mapsto \mathbf{A_r}$ can be viewed as a "cross section" through a "fiber bundle" with its base manifold given by the manifold that is represented (in discretized form) by the set of reference vectors $\mathbf{w}_\mathbf{r}^{(in)}$ of the SOM. In particular, if $M = L$, this fiber bundle can be identified with the tangent bundle of the SOM manifold. For $M \neq L$, we can represent general fiber bundles, for instance, the fiber bundle of inertia tensor fields on the configuration manifold of a manipulator (note that the "affine" term $\mathbf{w}_\mathbf{r}^{(out)}$ in (1) could be easily absorbed in $\mathbf{A_r}$ by augmenting all input reference vectors $\mathbf{w}^{(in)}$ with an additional, constant component).

To complete our definition, we still have to specify how to use the node outputs $\mathbf{y_r}$. In the simplest case, we just minimally extend the best-match step of the original SOM algorithm and define the output $\mathbf{y}(\mathbf{x})$ of the LLM-network as the response $\mathbf{y_s}(\mathbf{x})$ of the best-match unit $\mathbf{s}$ that satisfies $d_\mathbf{s}(\mathbf{x}) = \min_\mathbf{r} d_\mathbf{r}(\mathbf{x})$, where $d_\mathbf{r}(\mathbf{x})$ denotes some distance function, such as $d_\mathbf{r}(\mathbf{x}) = \|\mathbf{x} - \mathbf{w}_\mathbf{r}^{(in)}\|$ ("winner-take-all"-network). However, this introduces discontinuities at the borders of the Voronoi tesselation cells defined by the $\mathbf{w}_\mathbf{r}^{(in)}$ vectors. Such discontinuities can be avoided by introducing a "soft-max"-function

$$g_\mathbf{s}(\mathbf{x}) = \frac{\exp(-\beta d_\mathbf{s}(\mathbf{x}))}{\sum_\mathbf{r} \exp(-\beta d_\mathbf{r}(\mathbf{x}))} \tag{2}$$

with an "inverse temperature" $\beta > 0$ and blending the contributions of the individual nodes according to

$$\mathbf{y}(\mathbf{x}) = \sum_{\mathbf{r}} g_{\mathbf{r}}(\mathbf{x}) \cdot [\mathbf{w}_{\mathbf{r}}^{(out)} + \mathbf{A}_{\mathbf{r}}(\mathbf{x} - \mathbf{w}_{\mathbf{r}}^{(in)})]. \tag{3}$$

The prototype vectors $\mathbf{w}_{\mathbf{r}}^{(in)}$ can be determined in a variety of ways. Closest in spirit to the original SOM is to use the SOM algorithm with some neighborhood function that is chosen according to the expected topology of the input manifold in order to find a suitable arrangement of the input prototypes $\mathbf{w}_{\mathbf{r}}^{(in)}$. Often, it even suffices to choose a random subset of the training data and, optionally, to refine the positions of the chosen $\mathbf{w}_{\mathbf{r}}^{(in)}$ prototypes by some LBG-type vector quantization procedure (see, e.g., [12]). Together with a gradient based adaptation rule for the output prototypes $\mathbf{w}_{\mathbf{r}}^{(out)}$ and for the Jacobian matrices $\mathbf{A}_{\mathbf{r}}$, this leads to the following adaptation equations for training a LLM network from a set of input-output pairs $(\mathbf{x}, \mathbf{y})$:

$$\Delta \mathbf{w}_{\mathbf{r}}^{(in)} = \epsilon_1(\mathbf{x} - \mathbf{w}_{\mathbf{r}}^{(in)}) \tag{4}$$

$$\Delta \mathbf{w}_{\mathbf{r}}^{(out)} = \epsilon_2(\mathbf{y} - \mathbf{y}^{(net)}(\mathbf{x})) + \mathbf{A}\Delta \mathbf{w}_{\mathbf{r}}^{(in)} \tag{5}$$

$$\Delta \mathbf{A}_{\mathbf{r}} = \epsilon_3(\mathbf{y} - \mathbf{y}^{(net)}(\mathbf{x}))\frac{\mathbf{x}^T}{||\mathbf{x}||^2} \tag{6}$$

Here, the learning rule for output values $\mathbf{w}_{\mathbf{r}}^{(out)}$ differs from the equation for $\mathbf{w}_{\mathbf{r}}^{(in)}$ by the extra term $\mathbf{A}_{\mathbf{r}}\Delta \mathbf{w}_{\mathbf{r}}^{(in)}$ to compensate for the shift $\Delta \mathbf{w}_{\mathbf{r}}^{(in)}$ of the input prototype vectors. The adaptation rule for the Jacobians is the perceptron learning rule, but restricted to those input-output pairs $(\mathbf{x}, \mathbf{y})$ for which $\mathbf{x}$ is in the Voronoi cell of center $\mathbf{w}_{\mathbf{r}}^{(in)}$ (in practice, one should decrease the learning rates $\epsilon_{1...3}$ slowly towards zero, with ϵ_3 decreasing more slowly than the other two in order to leave any fine-tuning to the matrices $\mathbf{A}_{\mathbf{r}}$).

A variant of this scheme weights the adaptation steps (4) by the node activities $g_{\mathbf{r}}(\mathbf{x})$. In this case is important to note that for finite temperature parameter β the positions of the centers to which the $\mathbf{w}_{\mathbf{r}}^{(in)}$ will converge under (4) are then subject to a bifurcation scenario that forces the centers to become clustered. For β below some critical limit β_0, there exists only a single cluster (a weighted average of all input data points) into which all $\mathbf{w}_{\mathbf{r}}^{(in)}$ coalesce. For increasing values of β, this single cluster and its descendants become repeatedly split by a cascade of bifurcation steps. Usually, one is interested in a sufficiently high value of β that admits at least as many clusters as there are nodes. This value is distribution dependent, and it may differ from the value of β that optimizes the blending of the different $\mathbf{y}_{\mathbf{r}}$ contributions according to (3. This may make it necessary to use different values of β for training and for recall.

From a different perspective, a LLM-network can also be viewed as a "gated expert net", with the gating achieved by the SOM layer, and the

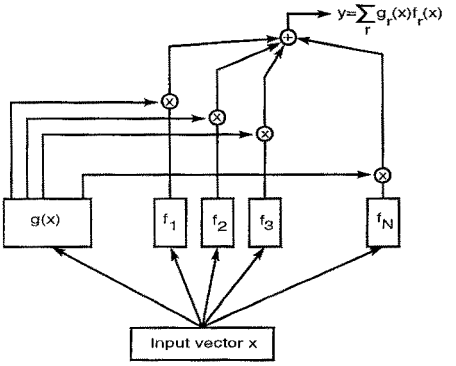


Fig. 1. LLM network as a mixture of "linear experts". The output is a weighted superposition of the outputs of locally valid linear maps $f_1 \ldots f_N$ ("expert nets"). The superposition coefficients $g_r(\mathbf{x})$ are determined by a shared "gating network" $\mathbf{g}(\mathbf{x})$, described by Eq.(2).

expert nets chosen as the local linear mappings defined by (1). This is depicted schematically in Fig.1, showing a number of linear "expert modules" that process a common input vector $\mathbf{x}$ and that contribute to the output according to a weighted superposition with weighting coefficients determined by the gating function $\mathbf{g}(\mathbf{x})$. From this point of view, LLM networks can also be seen as a natural extension of a single linear model, with the nonlinearity nicely "encapsulated" in the gating net equation (2) that determines how strongly the individual local linear maps contribute.

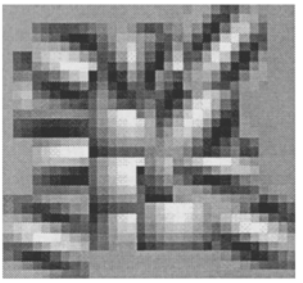
3 Robot Vision with LLM Networks

We have found the approach of LLM networks very suitable to learn both continuous and discrete mappings in various vision tasks, such as pose identification of deformable objects, e.g., hands [16] or object classification [4,5].

In both cases, a very important first step is a suitable preprocessing of the input image to obtain a manageably sized feature vector which also should already have some basic invariance properties, such as (at least approximate) translational invariance to facilitate the subsequent classification or identification task.

The first preprocessing operation usually is a figure ground segmentation, which can be color based (for the use of LLM-networks to learn a classifier for color-based segmentation, see e.g., [14]). For non-overlapping objects, this results in a number of "blobs" that can then be taken as the centers of an equal number of regions of interest (ROIs) which then are processed separately.

Within each ROI we choose a "fixation point" which becomes the center of a "jet" of Gabor filters that are convolved with the image intensity to obtain a low-dimensional feature vector. This choice of filter functions is motivated by the response properties of cells in the primary visual cortex, which can often be reasonably well described by Gabor filter profiles [2]. In the simplest case, the fixation point is taken as the centroid of the input intensity. For a more sophisticated scheme, based on the evaluation of local symmetries, see



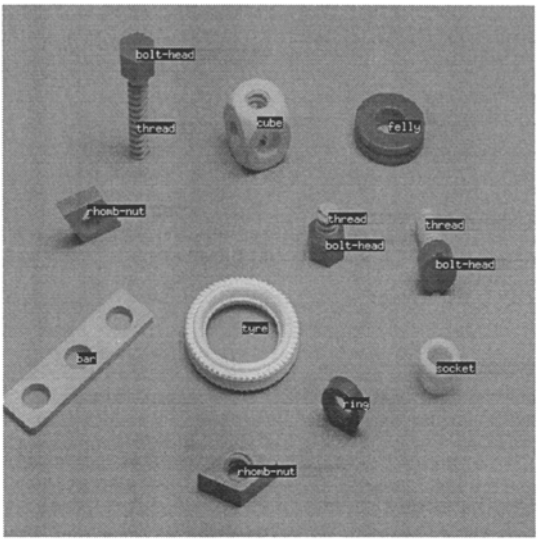


Fig. 2. *(a,top):* optimized Gabor filter mask. *(b,right):* some wooden toy pieces with their classification by the LLM-network (taken from ref. [4]).

e.g. [7]. Often, it is also advantageous to equalize the intensity histogram by a logarithmic intensity transformation, optionally followed by a convolution with a laplacian to suppress the DC component of the image.

While this scheme works already sufficiently well for many tasks, we can still obtain some further improvement by optimizing the chosen Gabor filter masks for an improved separation of the different object classes in feature space. We have investigated this approach for the task of classifying a set of wooden toy pieces in the context of a robot vision task [4]. This leads to optimized spatial filter functions such as the filter mask that is shown in Fig.2a, which usually show no longer any obvious symmetries. It also turns out that the parameter space in which the optimization takes place exhibits numerous local minima that correspond to filter sets that look strikingly different to the human eye (this, however, should not be too surprising, since a similar phenomenon also occurs when correlation filters belonging to different eigenfunctions, but the same eigenvalue, of a correlation matrix are linearly combined).

Figure 2b shows a selection of the toy objects that have been classified with this approach. The training set contains for each of the objects multiple views, so that recognition is also only weakly view-dependent. The same approach can be extended to the recognition of entire aggregates and to the recognition of parts within aggregates. This requires, however, to train a specialized recognition network for each aggregate, and also for the recognition of constituents within an aggregate. Figures 3 and 4 gives some impression of the degree of complexity that still is manageable within such a "holistic" approach. An attractive feature of this method is its speed. We have implemented the preprocessing steps on a MaxVideo200 Datacube. The re-

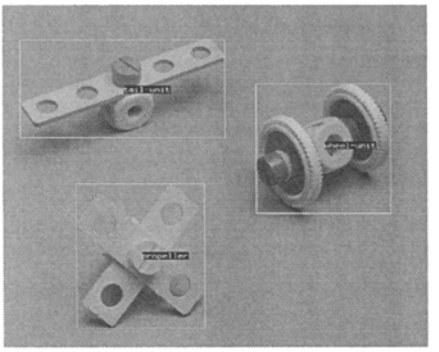


Fig. 3. Holistic identification of toy aggregates by specialized LLM-nets.

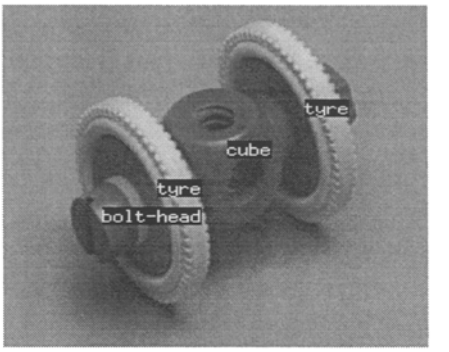


Fig. 4. Identification of individual constituents within an aggregate.

maining classification can then be done at about 10 frames per second on a workstation.

Of course, such "holistic" recognition approach cannot scale up to the recognition of more complex objects that are composed of many different parts. Therefore, we pursue the strategy to use the holistic object recognition networks a "holistic preprocessors" for a subsequent semantic network, which then draws upon symbolic world knowledge in order to guide a decomposition of more complex objects by positioning ROIs on subregions where particular, holistically recognizable object parts are expected. Work along these lines is currently under way, for some initial results, see e.g. [6].

4 Focal Attention by "Neural Zooming"

The previous section indicated already the importance for some control mechanism to guide a decomposition of complex objects into simpler constituents. An important role in this regard is played by *focal attention*, and semantic networks [18] appear to be very well suited for implementing a knowledge-driven, top-down attention control mechanism that can assist a collection of more specialized, neural object recognizers in decomposing object aggregates into more easily recognizable constituents [17].

However, one can also devise mechanisms that are based on simpler, and more general principles and that may even be implemented within an entirely neural architecture. One such approach is based on the observation that an object often can be quite easily identified when a particular set of characteristic object "landmarks" has been found. The detection of any individual landmark (such as a finger tip when a hand shape is to be recognized) depends on both local and global information, the latter providing the context for the correct interpretation (e.g., whether the same local pattern belongs to the middle or to the index finger) of the former.

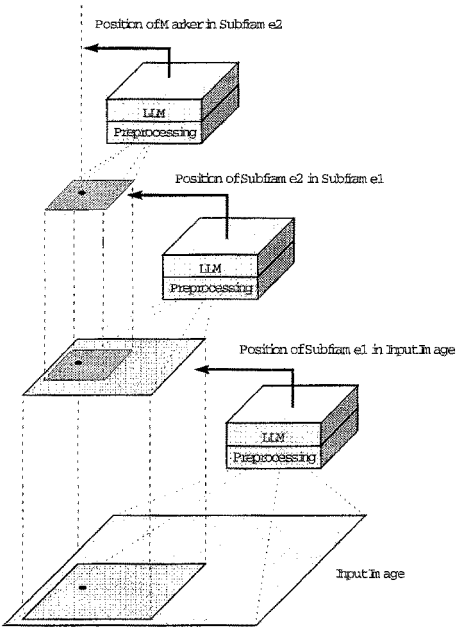


Fig. 5. Hierarchical architecture of LLM-networks to successively locate and "zoom in" on target feature in an image. Each network determines a smaller image region that then serves as input for the next network. The final network outputs the position estimate for the target feature.

The proper weighting of global and local context can be achieved within a hierarchical architecture of recognition networks, operating on a nested sequence of subimages whose spatial resolution and diameter vary inversely. Fig. 5 shows such architecture for the case of three hierarchically arranged LLM networks. The first network processes a coarse view of the entire image and passes to the second network a decision about the position of a smaller image rectangle which is then processed by the next network in the hierarchy, using a correspondingly higher spatial resolution. This step can be repeated several times, until the last network in the hierarchy, analyzing only a comparably small surround of the putative landmark location, makes a final decision of the precise landmark location in the image.

In this way, the initial networks in the cascade can first eliminate peripheral visual input and provide increasingly focused "regions of attention" for their successor networks. Since the feature vectors in the smaller input regions of the successor networks are computed with correspondingly downscaled versions of the gabor wavelets used in the first processing stage, these networks automatically become trained, and subsequently operate, at a finer spatial scale. This allows the final network to achieve a high spatial resolution

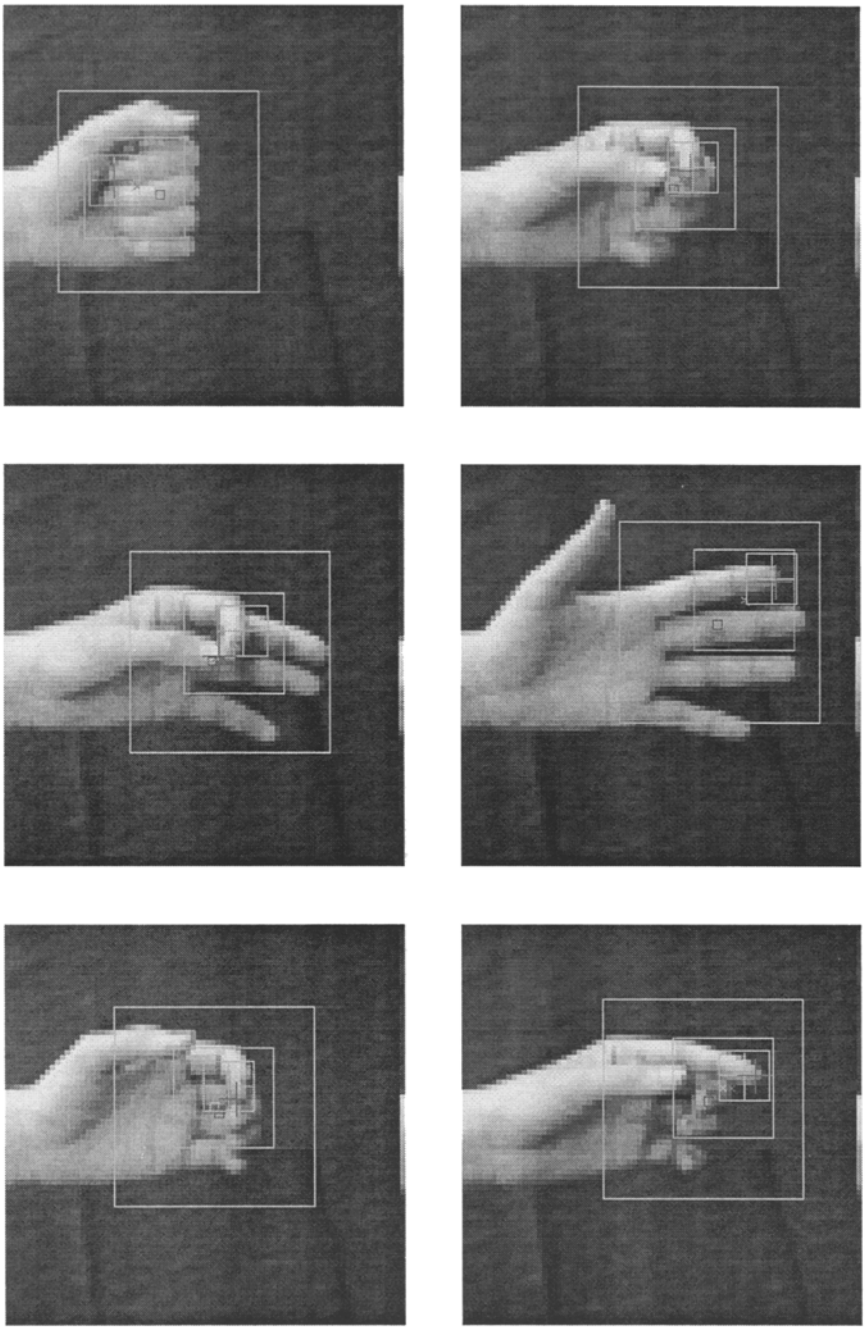


Fig. 6. Detection of index finger tip in hand images, using "neural zooming" with the hierarchical architecture depicted in Fig.5. Innermost bright plus mark indicates position estimate by final LLM network in the hierarchy, dark plus mark indicates correct target position.

while still more global visual information can be taken into account by the processing of the initial networks in the hierarchy.

This approach might be termed "neural zooming", since each network that is lower in the hierarchy determines a smaller subarea in its own image subregion into which then the successor network in the hierarchy can "zoom in".

Training of the network hierarchy can be achieved by providing training images with labeled landmark points (suitable target values for the centers of the intermediate image subregions can be obtained by simple interpolation between the final target point and, e.g., the image center).

We have been investigating this approach for the identification of finger tip locations in images of human hands. Fig.6 shows a number of recognition examples, obtained for identifying the location of the index finger tip for various hand postures before a black background (to simplify preprocessing, which was similar as explained in the previous section).

We used a three-stage recognition hierarchy (as depicted in Fig.5, with a monochrome input image of 80×80 pixels and successive processing stages of 40×40 and 20×20 pixels size, respectively). A feature vector was computed for each image, using a set of 36 gabor wavelets, centered at the points of a quadratic 3×3 lattice (using for each point the same spatial resolution and 4 different orientations, separated by angles of 45^0), covering a central region of the image area.

Using a set of only 50 training images, we can locate the index finger tip in a similar test set to mostly within a few pixels deviation from its true location. The depicted recognition examples are some typical cases from an independent test set of 50 further images (showing the same hand and using the same illumination conditions; at present, we did not yet provide any measures for normalizing illumination or the size of the hand). We are currently extending this approach for recognizing multiple finger tips simultaneously, including a more robust preprocessing and segmentation stage and considering the evaluation of context information that becomes available when several finger tip positions are known or when continuous motion sequences are considered. For a fuller account with more quantitative results see [19].

5 Parametrized Self-Organizing Maps

The LLM networks were motivated by the desire to overcome the discrete nature of the SOM manifold and to realize a smooth input-output mapping, even in higher-dimensional spaces. However, instead of achieving smoothness by attaching tangent planes to a coarsely discretized manifold, we also can try to construct the entire manifold in a parametrized form, using a set of topologically ordered reference vectors of a coarse SOM as support points. This is the basic idea behind the *Parametrized Self-Organizing Map* ("PSOM"). Its construction proceeds in two steps. The first step has to provide a set

of topologically ordered reference vectors $\mathbf{w_r}$, $\mathbf{r} \in \tilde{A}$, through which the desired manifold shall pass. As in the SOM, $\tilde{A}$ is usually some cartesian grid of the same dimensionality as the intrinsic dimensionality of the desired PSOM manifold S. Each grid point of A has "attached" a reference vector $\mathbf{w_r}$ that represents a discretization point on S, with the understanding that the labeling is such that neighboring grid points $\mathbf{r}, \mathbf{r}'$ belong to points $\mathbf{w_r}, \mathbf{w_{r'}}$ that are "close" to each other on the manifold (this is what we want to understand by "topologically ordered reference vectors"; at present, a mathematically rigorous definition encounters the same problems as the definition of "topological order" for higher dimensional SOMs; currently, no unversally accepted definition is available for dimensions larger than one). S itself is represented as an embedding of a d-dimensional hypersurface in a D-dimensional cartesian space and is given explicitly by a vector-valued function

$$\mathbf{w}(\mathbf{s}) = \sum_{\mathbf{r} \in \tilde{A}} H(\mathbf{r}, \mathbf{s}) \mathbf{w_r}. \tag{7}$$

Here, $H(\mathbf{r}, \mathbf{s})$ is a set of scalar valued basis functions (one for each $\mathbf{r} \in \tilde{A}$) that must obey the two conditions

$$H(\mathbf{r}, \mathbf{r}') = \delta_{\mathbf{r},\mathbf{r}'} \text{ for } \mathbf{r}, \mathbf{r}' \in A \text{ and} \tag{8}$$

$$\sum_{\mathbf{r} \in A} H(\mathbf{r}, \mathbf{s}) = 1 \tag{9}$$

The first of these two conditions ensures that the manifold defined by (7) indeed passes through the given "prototypes" $\mathbf{w_r}$. The second condition ensures that the shape of S is not altered under a global translation $\mathbf{w_r} \mapsto \mathbf{w_r} + \mathbf{t}$ of all prototypes.

Equation (7) is the continuous counterpart of the discrete SOM. In order to make the generalization into the continuous domain complete, we also have to specify an analogue of the SOM best-match search. As in the case of the SOM, we require here, too, minimization of some distance measure $d(\mathbf{x}, \mathbf{w}(\mathbf{s}))$ between input $\mathbf{x}$ and a general prototype $\mathbf{w}(\mathbf{s})$ on the PSOM, i.e., we have to find the "best match location" $\mathbf{s}(\mathbf{x})$ according to

$$\mathbf{s}(\mathbf{x}) = \arg\min_{\mathbf{s}} \sum_{i=1}^{D} c_i \cdot (x_i - w_i(\mathbf{s}))^2. \tag{10}$$

Here, we have specialized the distance measure $d(\mathbf{x}, \mathbf{w})$ to a weighted euclidean distance with weighting coefficients $c_i \geq 0$. In contrast to the standard SOM, where the determination of the discrete best match location $\mathbf{s}$ requires only a discrete search, we arrive in the case of a PSOM at a non-linear least-squares problem in continuous variables $\mathbf{s}$. While in principle we might use simple gradient descent to obtain a local solution, a more efficient method for this type of problem is the Levenberg-Marquardt algorithm [22] (simple gradient descent turns out to require a rather delicate step-size control).

The output of the PSOM is represented by the "best match prototype" $\mathbf{w}(\mathbf{s}(\mathbf{x}))$, which is the continuous equivalent of the discrete best match prototype $\mathbf{w_s}$ in the discrete SOM. If the distance could be reduced to zero (and all c_i are positive), we have the rather uninteresting result that $\mathbf{w}(\mathbf{s}(\mathbf{x})) = \mathbf{x}$. A much more interesting situation arises, when only a subset (say, k) of the c_i are positive and the rest vanishes. In this case, minimization of the weighted distance $d(\mathbf{x}, \mathbf{w}(\mathbf{s}))$ is unaffected by those components of $\mathbf{x}$ that belong to dimensions i for which $c_i = 0$. In other words, we now can accept as input an only *partially specified input vector* $\mathbf{x}$, provided that the index set $\tilde{I}(\mathbf{x})$ of its specified components comprises at least all those dimensions for which $c_i > 0$. If k (the number of nonvanishing c_i's) equals the intrinsic dimensionality d of the manifold S or is larger, specification of $\mathbf{x}$ will (except for singular cases) uniquely specify a best match point $\mathbf{w}(\mathbf{s}(\mathbf{x}))$ on S. The mapping $\mathbf{x} \mapsto \mathbf{w}(\mathbf{s}(\mathbf{x}))$ then becomes a non-linear projection from the embedding space $I\!R^D$ into S, and the image $\mathbf{w}(\mathbf{s}(\mathbf{x}))$ of any partially specified input $\mathbf{x}$ (such "partially specified inputs" actually being linear subspaces) under this mapping can be viewed as an *associative completion* of a fragmentary input.

Therefore, the PSOM acts as an *associative memory* that can complete fragmentary input vectors that are only specified along a subset $\tilde{I}(\mathbf{x}) \subset \{1 \ldots D\}$ of all embedding dimensions. All that then is required is to choose the weighting coefficients c_i according to the simple rule

$$c_i = \begin{cases} 1, & \text{if } i \in \tilde{I}(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

and to use the bestmatch step of the PSOM to obtain values for the missing components of $\mathbf{x}$.

The PSOM has some interesting similarities with the well-known spin-glass type associative memory (see, e.g. [8]). In both cases, retrieval occurs by offering a fragmentary input vector as initial condition, letting some dynamics (in the case of the PSOM implemented by the iterative Levenberg-Marquard algorithm) do its "associative completion". However, an important difference is the dimensionality of the attractors: a spin-glass type attractor network usually contains a large number of discrete point attractors (as a consequence of the symmetry of the coupling coefficients), while a PSOM contains a single, *continuous attractor manifold* of some intrinsic dimensionality $d > 0$. This is precisely what we need in many situations in robotics: here, we usually want to represent smooth relationships between continuous valued degrees of freedom instead of isolated and discrete patterns. Therefore, PSOMs are very well suited for robotics (continuous attractors can in principle also be obtained with recurrent multilayer perceptrons; however, these are very difficult to train [for a survey of algorithms, see, e.g., [21]], while the PSOM manifold can be rather directly constructed by simply specifying a reasonably dense set of prototype vectors $\mathbf{w_r}$ that sufficiently constrain the shape of the desired manifold).

So far, we did not specify a particular set of basis functions $H(\mathbf{r}, \mathbf{s})$. The two conditions (8,9) do not yet specify them uniquely. A particularly simple choice becomes possible if we assume that the sampling grid $\tilde{A}$ for the prototype vectors $\mathbf{w_r}$ is a (possibly distorted) d-dimensional cartesian lattice (cf. (7)). Then we can always construct a set of basis functions with the required properties (8,9) as products of one-dimensional Lagrange polynomials in the single axis variables s_i. For technical details, see [32,31] and also Eq.(12) below.

6 Robot Control with PSOMs

Many of the properties of PSOMs are particularly convenient in robotics, where different coordinate systems abound and training data are limited, since each training sample usually requires a rather time-consuming movement of the robot.

We first illustrate a typical use of PSOMs with the example of learning the inverse kinematics for a six-axis PUMA robot arm (for a fuller account of this work, see [26]). Simplified versions of this task have been considered earlier in the literature, for instance, [33,24].

The configuration space of a six-axis robot is a six-dimensional manifold. We can use a PSOM to represent this manifold in any higher-dimensional embedding space and we can choose for the coordinates of this space any set of coordinates that we find useful or convenient to relate with the arm configurations. A straightforward choice is a 15-dimensional embedding space, spanned by the set of the six joint angles $\theta_1 \ldots \theta_6$, the cartesian world coordinates (x, y, z) of the end effector, together with its approach vector (a_x, a_y, a_z) and the normal vector (n_x, n_y, n_z) to its "palm". Note that the last nine coordinates are just a different coordinate system to specify an arm posture, and that we have taken the freedom to specify manipulator orientation (3 DOFs) with the non-minimal, but more convenient set of six coordinates $a_x \ldots n_z$. We could easily extend this choice of an embedding space beyond $D = 15$ by including further coordinates, such as homogeneous transforms, Euler angles or polar coordinates. Moreover, some of these additional coordinates could encode information beyond the mere location of the end effector, e.g., we might include the configuration dependent arm Jacobian, the inertia tensor or homeogeneous transforms describing the location of individual arm segments in world coordinates, or even relative to other arm segments. Extending the embedding space in this way increases the range of parameter associations that can be formed after the PSOM has been constructed (for instance, we might then retrieve the joint angles that produce a given matrix value of the inertia tensor). Note that this increases the computational cost only linearly in the number of additionally included parameters, whereas the main cost factor, the *intrinsic* dimensionality of the PSOM manifold remains unchanged at its value of $d = 6$.

Having set up our embedding space, we have to decide on a suitable grid $\tilde{A}$ for the prototypes $\mathbf{w_r}$ that we use to define the PSOM manifold. In the present case, the most parsimonious choice is a cartesian $3 \times 3 \times 3 \times 3 \times 3 \times 3$-grid[1], requiring us to provide $729 = 3^6$ training prototypes $\mathbf{w_r}$ to construct the expression (7) for $\mathbf{w_r}$. The simplest way to generate these training vectors is to identify $\tilde{A}$ with a suitable hypercubical volume in the joint variables $\theta_1 \ldots \theta_6$ of the robot and to compute the remaining cartesian coordinates with the forward kinematics of the arm. The only remaining step is some choice for the basis functions $H(\mathbf{r}, \mathbf{s})$. In the present case we choose them as products $\prod_{i=1}^{6} H_i^{r_i}(\theta_i, s_i)$ of the simple one-dimensional Lagrange polynomials

$$H_i^j(\theta_i, s_i) = \prod_{k,k \neq j} \frac{s_i - \theta_i^k}{\theta_i^j - \theta_i^k} \tag{12}$$

along the six coordinate axes of our hypercubical joint space volume. As support points $\theta_i^{j=0,1,2}$ along axes $i = 1 \ldots 6$ we choose $\theta_i^0 = \theta_i^* - 45^0$, $\theta_i^1 = \theta_i^*$ and $\theta_i^2 = \theta_i^* + 45^0$, where θ_i^* denotes the angle of joint i when the robot is in its "home position".

Fig.7 shows the accuracy of the inverse kinematics mapping that is obtained from the resulting PSOM by depicting the robot's positioning error with an error cross for a set of 200 randomly chosen target positions in its workspace. The resulting dimensionless NRMS positioning error is 0.06, which compares very favorably with the earlier approach of [33] (these authors obtain a similar accuracy with a specially structured multilayer perceptron, but for the simplified task that remains when manipulator orientation is neglected; moreover, they use several thousands of training examples instead of just 729).

Considering the 3d-positioning task alone and staying with the 3 discretization points per manifold dimension reduces the required number of training vectors to only 27 and the mean root square positioning error to 2.6 cm (corresponding to a NMRSE of 0.05). By using five instead of three nodes per dimension (requiring 125 data samples), the error can be reduced to 5mm (NMRSE of 0.0097). By making use of the further improvements considered in [30], the error can be reduced still further. For a more recent discussion, including extensions to combining several PSOMs, see, e.g., [29,31].

This example indicates that PSOMs compare very favorably with other neural network approaches when it comes to construct smooth and highly non-linear mappings for robot control from limited numbers of data examples. Here we only stress that this comes with the additional benefit of a great freedom in the choice of input and output variables: even without extending the embedding space as indicated at the outset, we could as well use the

[1] a corresponding $2 \times 2 \times 2 \times 2 \times 2 \times 2$ grid would appear as even more parsimonious, but the resulting basis functions would then be linear in each single coordinate and we would lose the ability to obtain a good approximation for functions that possess a local extremum in the interior of their domain.

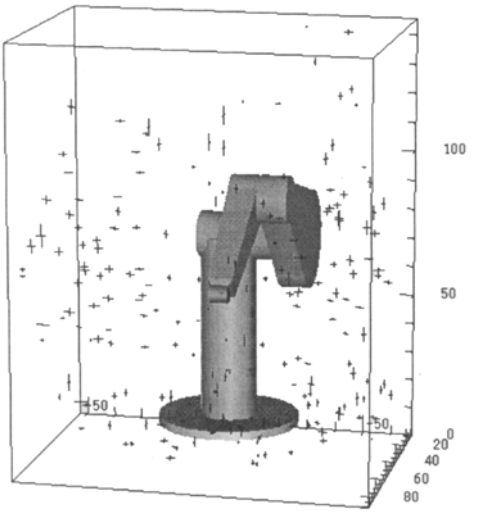


Fig. 7. Positioning errors of six-axis PUMA robot arm when the inverse kinematics transform is computed with a six-dimensional PSOM with 3^6 nodes, representing the configuration manifold of the manipulator embedded in a 15-dimensional $\boldsymbol{r}, \boldsymbol{a}, \boldsymbol{n}, \boldsymbol{\theta}$-space (for details, see text).

present PSOM to find an arm configuration for which, say, target values for three arm angles and three cartesian parameters are given.

Instead of supporting this assertion with additional simulation results, we illustrate as a further capability the use of a PSOM to flexibly satisfy different constraints to a manipulator with excess degrees of freedom. This is a frequent situation in robotics, and the use of the associative completion capability of the PSOM to solve this type of task was first pointed out in [31].

A common way to determine a robot configuration in the presence of excess degrees of freedom is the formulation of one or several additional cost functions that express additional constraints that are to be met by the manipulator. For instance, if the cost function is a weighted sum of squared arm velocities, we arrive a the well-known pseudo-inverse control for redundant manipulators (for a review, see e.g., [10]). By including each desired cost function in the set of embedding coordinates of the PSOM manifold we can use the PSOM to *associate manipulator configurations with desired values of particular cost functions.* For each query, we can thus "activate" a different cost function, simply by setting its associated distance weighting coefficient c_i to a positive value. If we choose this value to be large we make the associated constraint mandatory. If we specify only a small value, we establish

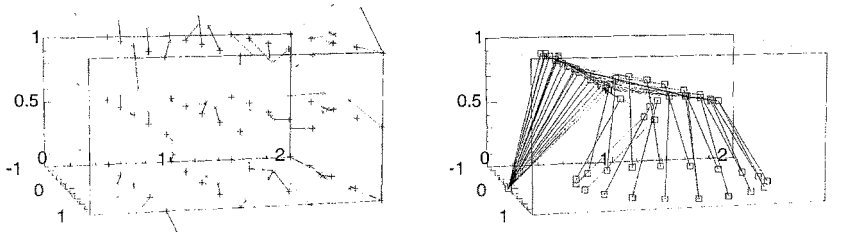


Fig. 8. *(a,left)* Positioning errors (magnified by a factor of 100 to become visible) for a redundant 4-link manipulator, evaluated at a cartesian grid of $3 \times 4 \times 5$ test target points it its workspace (+ and x-marks indicate target and actual positions, resp., with the deviation of the latter from the former magnified by a factor of 100). *(b,right)* Stroboscopic rendering of the arm when traversing a circular path in the ground plane, resolving the redundancy with an additional constraint feature maximizing the similarity of the last (distal) two joint angles.

a "soft constraint" that can be violated if it conflicts with other constraints that have larger c_i coefficients.

To illustrate these possibilities with a concrete example, let us consider a robot arm based at the coordinate origin and composed of three coplanar links in a vertical plane that can be rotated about the vertical z-axis (angle θ_0). The base link of the robot can be rotated about an axis passing through the origin and perpendicularly to the plane of the arm (θ_1), with the remaining two distal arm segments attached via two further revolute joints (angles θ_2 and θ_3, resp.) (see Fig.8b). The entire arm thus has four DOFs and, therefore, one excess degree of freedom with respect to the task of positioning its end effector at a specified location. For the construction of a PSOM for the configuration manifold of this arm, we choose as the lattice $\tilde{\mathbf{A}}$ a $5 \times 5 \times 5 \times 5$ grid covering the joint angle hypercube $[0^0, 180^0] \times [0^0, 120^0] \times [-120^0, 0^0] \times [-120^0, 0^0]$ for $\theta_{0\ldots3}$, resp. (segment lengths were 1.0, 0.8 and 0.7 from base to distal end). For the embedding space, we use the four joint angles and the three cartesian end effector coordinates. To later allow resolving the arm redundancy in different ways, we augment the embedding space by three further coordinate parameters, namely the variance of the last two joint angles, the angle of the middle arm segment against the horizontal, and the angle of the most distal arm segment with the vertical. This yields a 10-dimensional embedding space. If we activate the first constraint by specifying a target value of 0 and its associated weighting coefficient as 0.01 ("soft minimization of joint angle variance"), we obtain for the inverse kinematics (i.e., $(x, y, z) \mapsto (\theta_0 \ldots \theta_3)$) a positioning error of less than 0.008 for all indicated test target points within the work space depicted in Fig.8a (NOTE: to make the positioning errors visible, the length all error vectors depicted in Fig.8a has been scaled by a factor of 100). Fig.8b illustrates how in this case the redundancy is resolved when the end effector traverses a circular path in the ground plane. We can now readily change this behavior by activating a different constraint. Activating

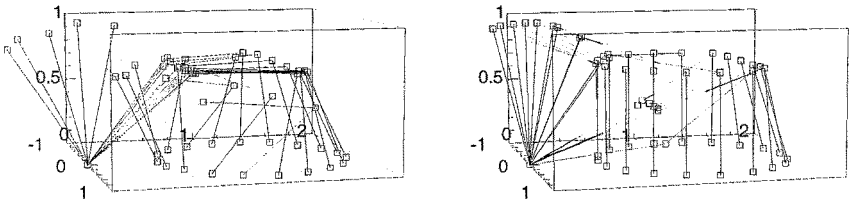


Fig. 9. Tracing of a horizontal circle with the same arm and the same PSOM mapping shown already in Fig.8, but activating different constraint features. *(a,left:* require middle arm segment to stay horizontal. Note that for the given geometry this constraint cannot be fulfilled for some target points on the path near the base the robot. Here, the PSOM still correctly positions the manipulator and selects arm configurations that try to violate the constraint only weakly. *(b,right):* Similar as *b)*, but now with a constraint specifying a vertical direction for the last arm segment. This time, the given constraint cannot be fulfilled for the most distal points of the circle, but the positioning again is correct along the entire path.

the constraint for the direction of the middle arm segment and specifying a target value of 0 (horizontal direction) results in a different sequence of arm configurations depicted in Fig.9a. As can be seen, the traversed path has remained the same, but the middle arm segment is now kept horizontal for all those target points for which this constraint can be met. For some points close to the base of the robot, meeting this constraint is impossible. Having, however, specified only a "soft constraint", allows the PSOM to violate it without (significantly) sacrificing accuracy in the tracking of the target trajectory. Finally, Fig.9b shows the result when instead the constraint for the vertical direction of the distal arm segment is activated, with a similar "graceful degradation" for some points, this time along the most distal points on the path.

Therefore, if we have some foresight which cost functions will be useful, we can augment our embedding space in advance, enabling us to construct very flexibly configurable "mapping modules" which have as their simultaneous benefits that we can (*i*) construct them readily from examples, (*ii*) apply them to flexibly associate partial state specifications, without being rigidly committed to particular mapping directions and (*iii*) use them as dynamically reconfigurable optimizers for a previously selected set of cost functions that have become associated with the stored configuration manifold.

7 Recognition of Hand Postures

The flexible mapping capabilities of PSOMs are also useful in other domains. We conclude this paper with a brief report of ongoing work in applying PSOMs to the identification of object pose from images. In Sec.4 we presented an approach to identify finger tip locations in video images. A natural

next step is to use this information to estimate the three-dimensional posture of the hand. Since a human hand has a large number of degrees of freedom, an accurate estimate will probably require the inclusion of a number of further "landmark points" on the hand; for the time being we bypass this issue by restricting the set of allowed hand postures to a subset that comprises those finger joint angle configurations that can be expressed as a linear superposition of a small number (currently three) of basis postures (such as a "fist", the open hand and a "precision grip", in which only the index finger and the thumb are flexed to touch each other at their tips). If we allow in addition some rotation about the arm axis and about a second axis that is perpendicular to the arm and to the line of sight (we left out rotations about the line of sight itself, since these correspond to a simple image rotation, while the other two change the appearance of the image itself) we end up with a five-dimensional configuration manifold for the described, restricted set of hand postures.

Our goal is to estimate postures from this set on the basis of a number of observed finger tip locations. To study how well this task can be solved with PSOMs, we are currently using a (simplified) articulated hand model that can be rendered on a computer screen. Using this model, we can generate training examples for the "forward transform", i.e, the mapping from joint coordinates to projected finger tip coordinates in the image. Presently, we have a 13-dimensional embedding space (five parameters to specify a hand posture and its orientation [subject to the restrictions explained previously], and eight parameters to specify the 2d-locations of four finger tips) for a configuration manifold of intrinsic dimensionality $d = 5$. For the construction of the PSOM we use 243 training points, obtained by computing the projected finger tip positions for all points of a $3 \times 3 \times 3 \times 3 \times 3$ grid of the configuration/orientation parameters of the synthetic hand model. Estimation of the hand posture that is associated with an observed set of finger tip locations can then be achieved by using the PSOM in the reverse mapping direction. Fig.10 gives some impression of the identification accuracy that can be obtained with this approach. The cross marks indicate observed finger tip locations (these were generated by randomly selecting a hand posture from the range of posture parameters that is covered by the training grid). The depicted hand posture is the estimate obtained by the PSOM, with its finger tip positions surrounded by the gray squares. As can be seen, the estimated postures very accurately fit the observed finger tip locations. The average accuracy is in the range of a few pixel positions. There is, however, as small fraction of cases for which the PSOM responds with a grossly incorrect posture (an example being given by the last picture in Fig.10). We attribute the occurence of these outliers to the occurrence of a local minimum for the best match search. Fortunately, the percentage of these cases is only small (about 3%) and they can probably be eliminated rather easily, e.g., by training several PSOMs with different data sets, making it less likely that more than a single PSOM gets simultaneously trapped in one of these rare local minima.

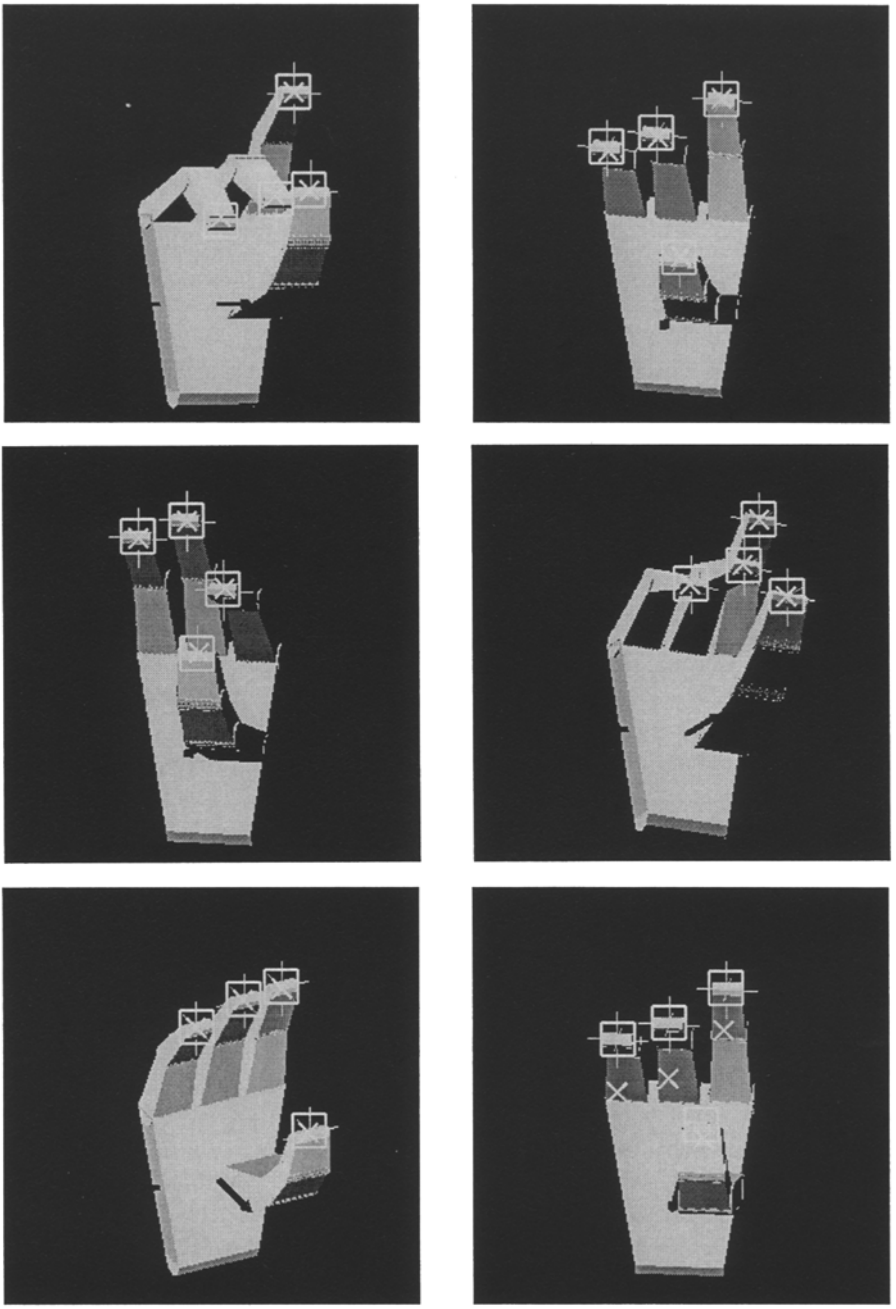


Fig. 10. Hand postures, estimated from finger tip locations (cross marks), using a PSOM network trained with 243 examples of the forward transform (posture parameters $\mapsto$ finger tip locations). The first five images illustrate the typical performance of the PSOM. However, a small percentage of cases results in grossly wrong responses, such as depicted in the lower right image.

8 Discussion

The complexities involved in linking perception with action challenge our traditional ways of constructing systems in a purely analytic way, working from basic principles upwards to cleanly specified algorithms that can deal transparently with all the complexities of real world tasks. While such an approach will always remain desirable, it is most likely unfeasable for the majority of tasks a robot has to solve. Therefore, we must develop alternative methods, with an emphasis on the adaptive construction of mappings and of dynamical systems that can achieve a desired behavior. Neural networks offer a natural framework for meeting this challenge, and we have identified as some of the important questions that have to be solved along this way the need for algorithms that allow the *rapid construction of mapping modules* from limited amounts of data, the development of strategies for *attention control*, both for the efficient allocation of processing resources as well as to support a tight man-machine cooperation, and finally, the necessity for exploring ways to implement additional functional building blocks, such as *continuous associative memories* with an intrinsic "multiway-mapping" capability in conjunction with an ability to deal with dynamically changing and possibly conflicting *constraints.*

We have presented examples of concrete approaches that attempt to address some of these issues, including *LLM* and *PSOM networks* for the rapid construction of mappings and continuous associative memories, a hierarchical architecture to implement a simple type of focal attention that can be learned from examples, and work showing how these approaches can be used in the context of various robot tasks, such as object classification, the identification of hand postures of a human user or the control of robot manipulators with the inclusion of excess degrees of freedom.

In all cases, the emphasis was on learning approaches with an ability to extract the necessary world knowledge in implicit form from modest numbers of training examples. Undoubtedly, there is still a long way to go until robots even remotely reach the learning competence that we routinely unfold when we learn to carry out a new task. However, compared with the long history of biological neural nets, artificial neural networks are still in their infancy and the rapid progress in their development encourages us to expect significant further progress ahead.

Acknowledgement

Part of this work has been supported by the German Research foundation (DFG) in the SFB 360 project.

References

1. Brady M. (ed.): Robotics Science. MIT Press, Cambridge Massachusetts (1989)

2. Daugman, J.G.: Two-dimensional spectral analysis of cortical receptive field profiles, Vision Research 20 (1980) 847-856
3. Erwin E., Obermayer K., Schulten K.: Models of Orientation and Ocular Dominance Columns in the Visual Cortex: A Critical Comparison. Neural Computation 7 (1995) 425-468
4. Heidemann G., Ritter H.: A Neural 3-D Object Recognition Architecture Using Optimized Gabor Filters. Proc. 13th Int. Conf. Patt. Recog., Vol. IV, (1996) 70-74, IEEE Computer Society Press.
5. Heidemann G., Ritter H.: A neural recognition architecture for composed objects. DAGM Symposium Mustererkennung (B. Jähne, P. Geißler, H. Haußecker, F. Hering eds.), (1996) 475-482, Springer Verlag Berlin Heidelberg.
6. Heidemann, G., Kummert F., Ritter H., Sagerer G.: A Hybrid Object Recognition Architecture, ICANN 96, Springer Lecture Notes in Computer Science 1112, (1996) 305-310. Springer Berlin Heidelberg.
7. Heidemann, G., Nattkemper T., Menkhaus G., Ritter H.: Blicksteuerung durch präattentive Fokussierungspunkte. Proceedings in Artificial Intelligence (B. Mertsching ed.) (1996) 109-116 Infix Verlag, St. Augustin (in German).
8. van Hemmen, J.L., Kühn R.: Collective Phenomena in Neural Networks. In: Models of Neural Networks (E. Domany et al. eds.) 1-106, Springer Berlin Heidelberg 1991.
9. Hunt K.J., Sbarbaro D., Zbikowski R., Gawthrop P.J.: Neural Networks for Control Systems: A Survey. Automatica 28 (1992) 1083ff.
10. Klein C.A., Huang C-H.: Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators. IEEE Trans. Sys. Man and Cybern. SMC-13, (1983) 245-250.
11. Kohonen, T.: The Self-Organizing Map, Proc. IEEE 78 (1990) 1464-1480
12. Kohonen, T.: Self-Organizing Maps, Springer Series in Information Sciences, Springer Berlin Heidelberg 1997.
13. Littmann E., Drees A., Ritter H.: Neural Recognition of Human Pointing Gestures in Real Images. Neural Processing Letters (1996) 61-71, Kluwer Academic Publishers.
14. Littmann E., Ritter H.: Adaptive Color Segmentation – A comparison of Neural and Statistical Methods. IEEE Trans. Neural Networks 8 (1997) 175-185
15. Maes P. (ed.): Designing Autonomous Agents (Special Issue of Robotics and Autonomous Systems), MIT Press Cambridge (1990)
16. Meyering A., Ritter H.: Learning 3D-Shape Perception with Local Linear Maps. Proc. Int. Joint Conf. on Neural Networks (1992) IV:432-436, Baltimore
17. Moratz R., Heidemann G., Posch S., Ritter H., Sagerer G.: Representing Procedural Knowledge for Semantic Networks using Neural Nets. Proc. 9th Scand. Conf. Image Anal. (1995) 819-828 Uppsala.
18. Niemann H., Sagerer G., Schröder S., Kummert F.: ERNEST: A Semantic Network for Pattern Understanding. IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (1990) 883-905
Nölker, C., Ritter, H.:
19. Nölker, C., Ritter, H.: Detektion von Fingerspitzen in Videobildern DAGM Mustererkennung (F. Wahl et al. ed.) (1997) (accepted) Springer Verlag Berlin Heidelberg.
20. Obermayer K., Ritter H., Schulten K.: A Model for the Development of the Spatial Structure of Retinotopic Maps and Orientation Columns. IEICE Trans. Fundamentals E75-A, (1992) 537-545.

21. Pearlmutter B.: Gradient calculation for dynamic recurrent neural networks: a survey. IEEE Trans. Neural Networks 6 (1995) 1212-1228
22. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in C, Cambridge University Press (1990).
23. Ritter H., Schulten K.: Extending Kohonen's Self-Organizing Mapping Algorithm to Learn Ballistic Movements. In: Neural Computers (Eckmiller R., v.d.Malsburg Ch. eds.) (1987) 393-406 Springer Heidelberg
24. Ritter, H., Martinetz, T., Schulten, K.: Topology Conserving Maps for Learning Visuomotor-Coordination. Neural Networks 2 (1989) 159-168.
25. Ritter, H., Martinetz, T., Schulten, K.: Neural Computation and Self-organizing Maps, Addison-Wesley, Reading, MA. (1992)
26. Ritter, H. Parametrized Self-Organizing Maps. ICANN 93 Proceedings (S.Gielen and B.Kappen eds.), (1993) 268-273, Springer Berlin Heidelberg.
27. Ritter H.: Learning with the Self-Organizing Map. In: Artificial Neural Networks (T. Kohonen et al. ed.) (1991) 379-384.
28. Ritter H.: Self-Organizing Feature Maps: Kohonen Maps. In: The Handbook of Brain Theory and Neural Networks (M.A. Arbib ed.) 846-851 MIT Press, Cambridge (1995)
29. Walter J., Ritter H.: Investment Learning with Hierarchical PSOM. In: Advances in Neural Information Processing Systems 8 (NIPS*95) (D. Touretzky, M. Mozer and M. Hasselmo eds.) (1996) 570-576, MIT Press
30. Walter J., Ritter H.: Local PSOMs and Chebychev PSOMs – Improving the Parametrized Self-Organizing Maps. ICANN95 Conf. Proceedings (1995) 95-102
31. Walter J.: Rapid Learning in Robotics. Cuvillier Verlag Göttingen 1996.
32. Walter, J., Ritter, H., Rapid Learning with Parametrized Self-organizing Maps. Neurocomputing 12, (1996) 131-153
33. Yeung D., Bekey G.: On Reducing Learning Time in Context Dependent Mappings, IEEE Transactions on Neural Networks 4 (1993) 31-42