ICANN '93

Proceedings of the International Conference
on Artificial Neural Networks
Amsterdam, The Netherlands
13–16 September 1993

Edited by
Stan Gielen and Bert Kappen

Springer-Verlag
London Berlin Heidelberg New York
Paris Tokyo Hong Kong
Barcelona Budapest

Neural Architecture for Robot Planning

P. Morasso†, V. Sanguineti†, and T. Tsuji‡

† *Department of Informatics, Systems, and Telecommunications, Genoa University, Italy*

‡ *Faculty of Engineering, Hiroshima University, Japan*

1. Introduction

Human motor skills depend, to a large extent, on two factors: (i) the intrinsic compliance of the musculo-skeletal system and (ii) its kinematic redundancy. The former factor allows a smooth modulation of contact forces during manipulation as well as a partial compensation of disturbances during trajectory formation. The latter is necessary for incorporating different task constraints in the same motor plan. In particular, experimental investigations of human arm movements [3] showed that the arm (even of a deafferented monkey, deprived of any kinesthetic/somesthetic feedback) returned toward an intermediate position (between the initial and final one) when the arm was temporarily displaced before the onset of the target. This suggests the hypothesis that the central nervous system plans a movement in terms of a sequence of equilibrium points, or *virtual trajectory* [5] but does not tell us anything about the central process which is responsible for producing it, particularly in the case of a redundant system. The main idea of this paper is that in complex robotic manipulators which, similarly to the human arms, are characterized by mechanical compliance and kinematic redundancy, it is quite useful for the planner to establish an analogy beween the real-time gradient-descent process determined by the mechanical potential field and a similar process associated with the dynamics of a neural computational engine which produces the virtual trajectories. In fact, the reaction of the arm to a contact/disturbance force is a gradient-descent in the elastic potential field of the musculo-skeletal system and this does not imply just a single motion pattern but a family of responses, indexed by the stiffness level of the different muscle groups, thus allowing a run-time adaptation of the *reactive-part* of the plan. Analogously, the run-time adaptation of the *trajectory-formation part* of a plan can be achieved by a gradient-descent process in a computational potential field modulated according to different task constraints. In this paper, we show how artificial potential fields can be expressed by means of a self-organized map [8], which also represents a *forward model* of the manipulator, and how gradient-descent can be performed in real-time on this map.

2. Self-organized forward model of redundant manipulators

Redundant manipulators imply an ill-posed inverse kinematic problem, because the mapping from sensory stimuli to motor variables is one-to-many. Although a *direct inverse modelling* approach can solve the problem on the base of *self-supervised learning* with a suitable training set[1], a better solution breaks down the learning process into two phases [6]: (i) in the first phase, the self-supervised strategy is used in order to learn the motor-to-sensory transformation (a *forward model* of the robot), which is always well defined, irrespectively of the degree of redundancy; (ii) in the second phase, the inverse sensory-to-motor transformation (an *inverse model* of the robot) is trained

[1]Self-supervised learning is a strategy for learning the model of a system or its inverse and it consists of generating pseudo-random input patterns, applying them to the system, measuring the response patterns and using the input-output pairs as training set of the model: either a direct/forward model (capable to predict the response given the stimulus) or an inverse/backward model (capable to estimate the stimulus that would produce the observed response).

by combining two criteria, one which aims at getting a global identity mapping[2] and another one which attempts to minimize some additional cost index. In this way, it is possible to have inverse models which are tuned according to specific task needs and are able to integrate different sensory channels. In the implementation of this concept proposed by [6], the forward model as well as the controller are multi-layer networks trained with the well-known technique of back-propagation. This is simple and efficient but can only work in an off-line manner and thus does not allow run-time task adaptation. The approach proposed here exploits the same forward-modelling concept and two-phase adaptation method but attempts to introduce an on-line element: (i) the forward model of the body is learned off-line as a self-organized map, while (ii) the synthesis of the inverse sensory-motor transformation is performed at run-time exploiting computational features of the self-organized forward model that allow a local computation of the gradient field. In this way, it is possible to take into account task-dependent constraints that are not fixed but are indeed specified on-line.

3. Off-line learning of the forward model

The forward model is defined by an input *motor* vector μ, which is the set of joint angles, and an output *sensory* vector β, which stores all the observables dependent upon μ, as the coordinates of the end-effector $\mathbf{x}_{ef}$ in the workspace or visual features of the robot images from a stereo-pair of cameras. We represent such a model $\beta = \beta(\mu)$ by means of a self-organized cortical map [1] which consists of a single layer or *neural field* F of M processing elements (PE's): They operate in parallel on a common input vector $\mu \in \mathbf{M} \subset \mathbf{R}^N$ and their activation function is the normalized Gaussian or *softmax* function

$$U_i(\mu) = \frac{G(\|\mu - \tilde{\mu}_i\|)}{\sum_j G(\|\mu - \tilde{\mu}_j\|)} \tag{1}$$

(the $G(\cdot)$'s are Gaussian functions of equal variance and the norm is L_2) which has been used in the field of regression and classification [7, 4] and is a type of hyper radial basis function [10, 9]. PE_i's have limited receptive fields, centered around *preferred vector prototypes* $\tilde{\mu}_i$'s, where the activation function peaks. The distribution of activities on the field for a given input pattern is also known as coarse or *population code* of that pattern.

Learning is performed by means of self-supervised soft competitive learning:

$$\begin{aligned} \Delta\tilde{\mu}_i &= \eta_1 \, (\mu_k - \tilde{\mu}_i) \, U_i(\mu_k) \\ \Delta\tilde{\beta}_i &= \eta_2 \, (\beta_k - \tilde{\beta}_i) \, U_i(\mu_k) \end{aligned} \tag{2}$$

which is based on a training set of self-generated pseudo-random patterns $(\mu_k, \beta_k : k = 1, 2, ...)$ and carries out a smooth distribution of prototype vectors on the neural field with optimal statistical properties[3]. Moreover, the forward model is approximated by the following formula:

$$\beta = \beta(\mu) \approx \sum_i \tilde{\beta}_i U_i(\mu) \tag{3}$$

which was demonstrated [13] to be a minimum-variance estimator.

As an example, let us consider a simple planar arm with 2 degrees of freedom and let us restrict the sensory part β only to the 2D position vector of the end-effector. While performing a simulation of this model with a cortical map of $M = 200$ processing elements, training was carried out by generating 4000 pseudo-random postures, uniformly distributed in a restricted area of the configuration space. Such a training set was presented 10 times to the network, reducing the learning rates (η_1 and η_2 of Equations 2) from 0.3 to 0.05 and the variance $\sigma_\mu{}^2$ of the softmax

[2]The cascade of the inverse model + forward model should be able to generate a motor command, in response to a given sensory stimulus, in such a way that the predicted sensory consequences of the command can reproduce, as closely as possible, the same stimulus.

[3]The learning rule can be derived by minimizing the cross-entropy between the probability density function of μ and its approximation by means of a Gaussian mixture, with the Gaussian centers in $\tilde{\mu}_i$'s [2].

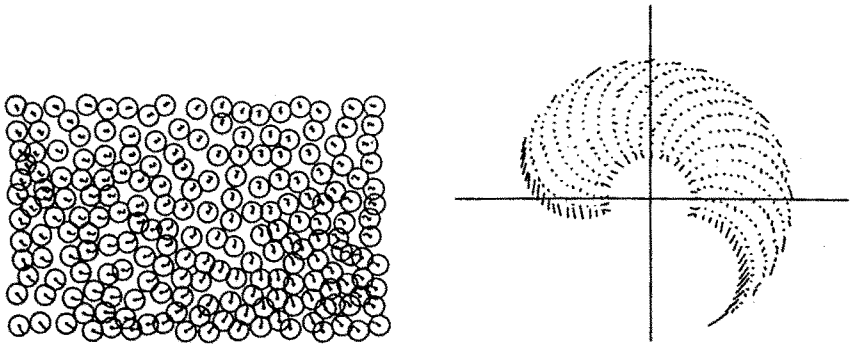

Figure 1: Left: Trained cortical map of a 2 dof arm. Right: Distribution of error vectors on the workspace.

function from 0.2 to 0.004. Figure 1(left) shows the trained cortical map, where each PE is represented by means of a circle containing a small picture of the arm in the learned configuration $(\tilde{\mu}_i, \tilde{\beta}_i)$. For the trained map, we also tested the accuracy of reconstructing β by means of (3) as follows: (i) an arm configuration μ with the corresponding end-effector position β is chosen, (ii) the population code is computed, (iii) the reconstruction function (3) is applied, yielding an estimate $\hat{\beta}$, and (iv) the error vector $e = \beta - \hat{\beta}$ is computed, that is the displacement between the original and the estimated β. Figure 1(right) shows the distribution of these error vectors over the whole workspace for a map of 200 PE's.

4. Run-time gradient descent

A potential field $\varepsilon = \varepsilon(\mu)$, as any smooth function of μ, can be represented by means of a distributed representation which uses the same population code and the same interpolation mechanism of the forward model:

$$\varepsilon(\mu) \approx \sum_i \tilde{\varepsilon}_i U_i(\mu) \tag{4}$$

where the $\tilde{\varepsilon}_i's$ are samples of the potential field which are assigned to each processing element in relation to its preferred sensory-motor pattern $(\tilde{\mu}_i, \tilde{\beta}_i)$.

The gradient-descent strategy requires to integrate the equation

$$\dot{\mu} = -\gamma \nabla \varepsilon(\mu) \tag{5}$$

and this can be implemented in the cortical map by using the following result:

$$\dot{\mu} = \gamma \sum_i (\mu - \tilde{\mu}_i) \tilde{\varepsilon}_i U_i(\mu) \tag{6}$$

which can be derived by writing the equation for the gradient vector $\nabla\varepsilon(\mu) \approx \sum_i \tilde{\varepsilon}_i \partial U_i(\mu)/\partial\mu$ and then taking into account the structure of the softmax function (with variance σ^2):

$$\frac{\partial U_i(\mu)}{\partial \mu} = -\frac{1}{\sigma^2}\left(\sum_j \tilde{\mu}_j U_j - \tilde{\mu}_i\right) U_i(\mu) \approx -\frac{1}{\sigma^2}(\mu - \tilde{\mu}_i)\, U_i(\mu) \tag{7}$$

The block diagram of figure 2 summarizes the simple circuitry which carries out the local computations outlined above, particularly as regards equations (3) and (6) and allows the cortical map to carry out gradient-descent. It can work in parallel and can support real-time provided that gradient-descent is allowed to operate near equilibrium. This is a key computational constraint which can be satisfied by a strategy of *local-incremental gradient-descent*, i.e. by a time-varying

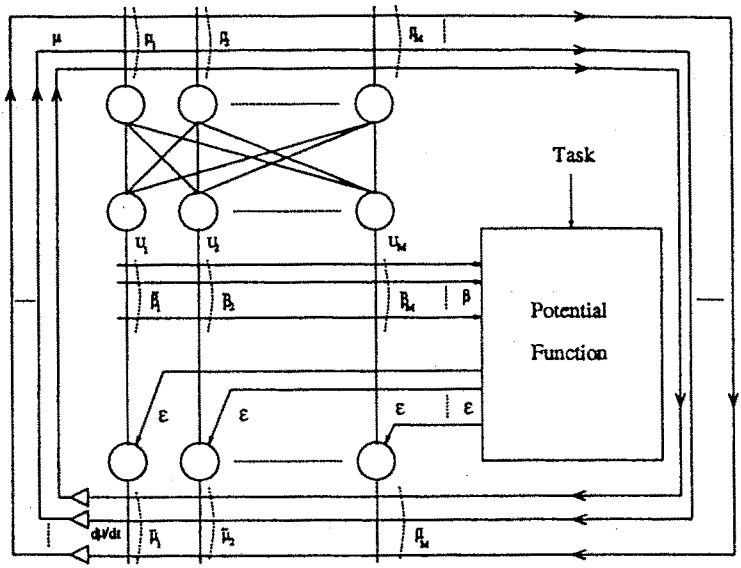

Figure 2: Block diagram of the cortical map gradient-descent network. The neurons in the top bank have a Gaussian activation, in the middle bank perform lateral gating inhibition, and in the bottom bank are simply multiplicative. The triangles are a bank of integrators.

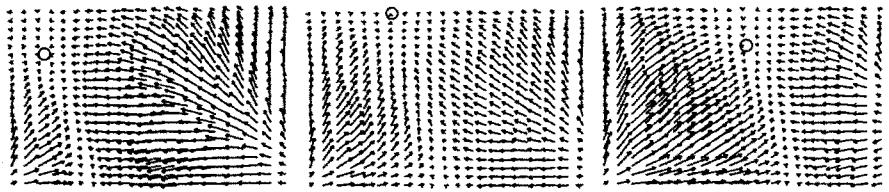

Figure 3: Three time frames of the target-distance gradient field snapped when the virtual target is at the initial, intermediate, and final position.

potential field which varies in such a way that its minimum stays close to the current position. It is important not only for implementing the real-time constraint but also for avoiding a typical pathology of gradient-descent mechanisms, i.e. getting trapped into local minima. In particular, this can be obtained by a target generation mechanism that smoothly shifts a virtual target $\mathbf{x}_v = \mathbf{x}_v(t)$ from the initial hand position $\mathbf{x}_0$ to the real target position $\mathbf{x}_T$ and a target potential function which simply measures the Euclidean distance between the end-effector and virtual-target positions: $\varepsilon^{tgt} = \varepsilon^{tgt}(\mu, t) = \frac{1}{2} \| \mathbf{x}_v(t) - \mathbf{x}_{ef}(\mu) \|^2$. A model of the target generation mechanism is discussed elsewhere in the book [12]. Figure 3 shows the evolution over time of the field for a simple targeting movement.

5. Integration of different task constraints

The use of potential fields is a powerful technique for representing task constraints of different nature and defined in different coordinate frames. We previously considered a field for the representation of targets which operates as an attractor on the cortical representation of the end-effector. This is effectively a kinematic inversion mechanism and it operates equally well with redundant and

non-redundant systems, implicitly computing an inverse matrix: the inverse Jacobian of the direct kinematic function, for non-redundant systems, or the Moore-Penrose pseudo-inverse matrix, for redundant systems. One of the nice features of performing the inversion via gradient-descent in a cortical map is its computational robustness, even in the vicinity of kinematic singularities, such as the boundary of the workspace, which tend to make unstable conventional inversion methods. The robustness comes from two elements: (i) the nature of the cortical map organization and training limits the domain of the generable motor patterns, whichever is the synergy formation mechanism, to a smooth area covering the set of training patterns; (ii) the gradient-descent mechanism implies that the flow of motor patterns is, in any case, a smooth path in the cortical map. For example, if the target x_T is outside the workspace, then the generated virtual trajectory *hits* the boundary of the workspace and then smoothly *slides* on it until it reaches the point closest to the target.

In fact, potential fields are general purpose tools for the specification of tasks, namely for representing both the *attraction* to a desirable state and the *repulsion* from undesirable or dangerous states. For example, a task component which aims at staying away from a dangerous joint configuration $\hat{\mu}$ can be represented by means of a repulsive field of the type $\varepsilon^{rep} = \varepsilon^{rep}(\mu) = g(\| \mu - \hat{\mu} \|^2)$, where $g(.)$ is a suitable monotonically decreasing function.

Attractive or repulsive fields can be computed from measures performed in different coordinate frames and the crucial point is that the fields can be superimposed on the same cortical map provided that the forward kinematic model embedded in the map allows to perform such measurements. The additivity of the task-related fields is the fundamental concept in our model for the integration of different task constraints as well as for the composition of complex tasks that attempt to reach several objectives at the same time, exploiting kinematic redundancy. An example is the task of keeping the end-effector as parallel as possible to its initial orientation while it is following an assigned path. This can be solved by defining, in addition to the *target-potential* above, a *parallelism-potential*[4] $\varepsilon^{par} = \varepsilon^{par}(\mu) = \frac{1}{2} \| f(\mu) - f(\mu(t_0)) \|^2$, and by carrying out a gradient-descent in the combined field: [5]

$$\varepsilon = \varepsilon(\mu) = k_1 \varepsilon^{tgt}(\mu) + k_2 \varepsilon^{par}(\mu) \tag{8}$$

where k_1 and k_2 weight the relative contributions of the two fields. This problem was simulated in relation with the cortical map of a 3 degree-of-freedom planar arm (map size: 900 $PE's$) and then we performed two experiments of reaching: in the first one, only the target reaching task was specified and the potential function ε contained the pure target component, computed in the end-effector space; in the second experiment, the initial configuration as well as the final target were the same, but we added the parallelism-potential with a small relative weight (0.1). Figure 4 (left) shows a simulation run of the first experiment: the end-effector follows the planned straight trajectory, smoothly changing its absolute orientation. Figure 4 (right) shows the simulation of the second experiment. As is apparent, the trajectory of the end-effector is maintained and its orientation is significantly more stable than in the previous case. A further example is given by obstacle avoidance. Let us suppose that while tracking a virtual target in a cluttered environment a robot might hit an obstacle unless it exploits its redundant degrees of freedom. In this case there should be a repulsive *obstacle-potential* $\varepsilon^{obs} = \varepsilon^{obs}(\mu)$ in addition to the attractive *target-potential*.

In the presentation above we assumed that the samples of the potential functions $\tilde{\varepsilon}_i$'s are somehow "downloaded" on the PE's of the cortical map. The discussion of how this can be done is outside the scope of this paper. A possible model, which is based on a further layer of maps, is discussed in [11].

[4] The function $f(.)$ is the sum of the components of the argument vector, which corresponds, in the case of a planar arm, to the absolute orientation of the end-effector. $\mu(t_0)$ is the initial configuration vector.

[5] We can note that the combined field is "sharper" than the original target field which allows infinite equilibrium configurations if the system is redundant (the null-space of the transformation). The second field reduces the dimensionality of this space, increasing the cost of configurations which do not match that constraint.

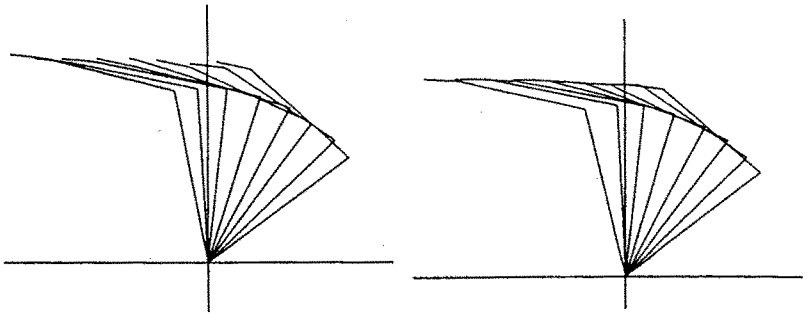

Figure 4: Reaching movements with a planar 3 degree-of-freedom arm. Left: only the target constraint is present. Right: for the same target, a constraint of parallelism is added.

References

[1] S. Amari. Dynamical stability of formation of cortical maps. In M.A. Arbib and S. Amari, editors, *Dynamic interactions in neural networks: models and data*, pages 15–34. Springer-Verlag, Berlin, 1989.

[2] M. Benaim and L. Tomasini. Competitive and Self-Organizing algorithms based on the minimization of an information criterion. In T. Kohonen, K. Makisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, Amsterdam, 1991. North-Holland.

[3] E. Bizzi, N. Accornero, W. Chapple, and N. Hogan. Posture control and trajectory formation during arm movements. *J. Neuroscience*, 4:2738–2744, 1984.

[4] J.S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. Fogelman Soulie and J. Herault, editors, *Neurocomputing*, volume NATO ASI F-68, pages 227–236. Springer Verlag, Berlin, 1989.

[5] N. Hogan. An organizing principle for a class of voluntary movements. *J. Neuroscience*, 4:2745–2754, 1984.

[6] M. I. Jordan and D. E. Rumelhart. Internal world models and supervised learning. In L. Birnbaum and G. Collins, editors, *Machine Learning: Proceedings of the Eighth International Workshop*, San Mateo, CA, 1991. Morgan Kaufmann.

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

[8] P. Morasso and V. Sanguineti. SOBoS – A Self-Organizing Body Schema. In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks*, Amsterdam, 1992. North-Holland.

[9] T. Poggio and F. Girosi. Networks for approximation and learning. *Proc. of the IEEE*, 78:1481–1495, 1990.

[10] M. J. D. Powell. Radial basis functions for multivariable interpolation: a review. In J. C. Mason and M. G. Cox, editors, *Algorithms for Approximation*. Clarendon Press, Oxford, 1987.

[11] V. Sanguineti and P. Morasso. Models of cortical maps. In *5nd Italian Workshop on Parallel Architectures and Neural Networks*, Vietri sul Mare, Italy, 1993. in press.

[12] V. Sanguineti, T. Tsuji, and P. Morasso. A dynamical model for the generation of curved trajectories. In *International Conference on Artficial Neural Networks*, Amsterdam, 1993. in press.

[13] D. F. Specht. Probabilistic neural networks. *Neural Networks*, 3:109–118, 1990.