Run-Time Robot Planning

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

† Department of Informatics, Systems, and Telecommunications, Genoa University, Italy
‡ Faculty of Engineering, Hiroshima University, Japan

1 Introduction

Robot planning models based on real or artificial potential fields are a powerful computational metaphor [5, 11, 10, 3]. In particular, we developed a neural network architecture [7] which learns a forward model [4] of a redundant manipulator (via self-supervised training) as a map of normalized radial basis neurons and inverts the model by means of run-time gradient descent of a task-related potential field. In this paper, we propose a distributed model for the computation of the field, which is consistent with the model-inversion map, and we discuss the problem of self-synchronization between the gradient-descent process and a process for the generation of virtual trajectories of the end-effector.

2 Run-time inversion of a self-organized forward model

Let a redundant manipulator be described by a vector of joint angles $q \in Q \subset R^n$, an end-effector vector $x \in X \subset R^6$ (with $n > 6$), and the corresponding forward kinematic model $x = x(q)$. We approximate such a model with a single-layer map or neural field $F$ of $M$ processing elements ($PE_i, i = 1, 2 \ldots M$) which operate in parallel receiving the common input vector $q$ and reacting with a normalized Gaussian or softmax activation function [6, 2]:

$$U_i(x) = \frac{G(||x - \bar{x}_i||)}{\sum_j G(||x - \bar{x}_j||)}$$  \hspace{1cm} (1)

(The $G(\cdot)$'s are Gaussian functions of equal variance and the norm is $L_2$). $PE_i$'s have limited receptive fields, centered around preferred vector prototypes $\bar{x}_i$'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:

$$\Delta \bar{x}_i = \eta (q - \bar{q}_i) U_i(q)$$
$$\Delta \bar{q}_i = \eta (x - \bar{x}_i) U_i(q)$$  \hspace{1cm} (2)

which is based on self-generated pseudo-random patterns $(x, q)$ and carries out a smooth distribution of prototype vectors on the neural field with optimal statistical properties. The forward model is then approximated by the following formula:

$$x = x(q) \approx \sum_i \bar{x}_i U_i(q)$$  \hspace{1cm} (3)

which was demonstrated [9] to be a minimum-variance estimator. This kind of estimator is also applicable to any smooth function of $x$ and in particular to an artificial potential field $\varepsilon = \varepsilon(q)$ which represents the task constraints:

$$\varepsilon(q) \approx \sum_i \bar{\varepsilon}_i U_i(q)$$  \hspace{1cm} (4)

\footnote{The learning rule can be derived by minimizing the cross-entropy between the probability density function of $X$ and its approximation by means of a Gaussian mixture, with the Gaussian centers in $\bar{x}_i$'s [1].}
Figure 1: Block diagram of the gradient-descent network.

where the $\xi_i$'s are samples of the potential field which are assigned to each $PE$ in relation to its preferred sensory-motor pattern $(\tilde{x}_i, \tilde{q}_i)$. Model inversion via gradient-descent exploits the following result [7]:

$$\nabla \epsilon(q) \approx - \sum_i (q - \tilde{q}_i) \xi_i U_i(q)$$  \hspace{1cm} (5)

thus yielding an explicit local dynamic equation for each $PE$ in the map:

$$\dot{q} = \alpha \sum_i (q - \tilde{q}_i) \xi_i U_i(q)$$  \hspace{1cm} (6)

The block diagram of figure 1 summarizes the simple feedback which allows the cortical map to carry out gradient-descent. In order to support run-time planning, the computational mechanism described above must be complemented by two additional mechanisms which are described in the two following sections: (i) a distributed mechanism for the generation of the potential field and (ii) a synchronizable mechanism for the generation of virtual targets.

3 Network model for the generation of the potential field

The use of potential fields is a powerful technique for representing task constraints of different nature and defined in different coordinate frames. In particular, we hypothesize a finite repertoire of general-purpose cost functions $\epsilon_1 = \epsilon_1(q), \ldots, \epsilon_N = \epsilon_N(q)$) which operate either as attractors (for task-components which assign a credit proportional to the distance from desired states) or repulsors (for task-components which assign a penalty proportional to the distance from dangerous states). For example, an attractive target-potential can be defined as follows: $\epsilon^{\text{at}}(q) = \frac{1}{2} \| x_T - x(q) \|^2$, where $x_T$ is the target position. A repulsive obstacle potential can be written as: $\epsilon^{\text{ob}}(q) = f(|| x_{ob} - x(q) ||)$, where $x_{ob}$ is the obstacle point which is closest to the end-effector and $f(\cdot)$ is a monotonic decreasing function. Then, we can exploit the additivity of potential fields in order to integrate the different task-components:

$$\epsilon = \epsilon(q) = \sum_i g_i \epsilon_i(q)$$  \hspace{1cm} (7)

where the $g_i$'s are relative gain coefficients which can be set according to a high-level attentional module.

We propose a distributed mechanism which implements the global field concept formulated above (see fig. 2 for a block diagram). It consists of a number of potential-networks, one for each potential function of the repertoire, which have the same size of the gradient-descent network. The generic $PE$ (element $i$ of network $k$) has two types of vector inputs: (i) the pair of prototype vectors $(\tilde{g}_i, \tilde{x}_i)$ which come from the
corresponding PE of the gradient-descent network and (ii) a task-specific vector $x_k$, which is common to all the PE's of the same network and is coming, as the attention coefficients, from the high-level part of the planner. The output is just a scalar ($\xi$) which estimates the credit/penalty assigned to the input pattern according to specific task-component. For example, in the case of the target-potential, considered above, $x^{st} = x_T$ and the activation function of each PE is simply a Euclidean distance between $x_T$ and $x_k$. The outputs of the homologous PE's of the different potential-networks are added and the global potential value is fed back to the gradient-descent network.

4 A synchronizable mechanism for the generation of virtual targets

Real-time gradient-descent requires that the potential field is incrementally updated in order to always keep the gradient-descent mechanism operating near equilibrium (local-incremental gradient-descent). This can be obtained by a target generation mechanism that smoothly shifts a virtual target $x_v = x_v(t)$ from the initial hand position $x_0$ to the terminal target position $x_T$ thus shaping, via the corresponding potential network, a target-potential field whose equilibrium state smoothly shifts the position of the end-effector along the target path. The other, overlapped potential fields introduce a sort of bias which, for the same target motion, determines task-consistent arm-motions in the null-space of the forward kinematic function.

The general requirement for a target-generation mechanism is to produce smooth trajectories such as trajectories with a bell-shaped velocity profile which are known to (approximately) minimize jerk. In another paper [8] we proposed a model of this kind which is based on a time base generator expressed as a non-linear dynamical system of the following type:

$$\xi = \gamma (\xi(1 - \xi))$$

where $\xi$ is a normalized scalar variable, the exponent $t$ must be less than 1 in order to guarantee a finite duration, and the coefficient $\gamma$ is proportional to the peak velocity. The trajectory of the virtual target is derived from the time base generator with a simple linear operator:

$$x_v = x_v(t) = x_0 + (x_T - x_0) \xi(t)$$

Let us suppose that a nominal value $\gamma = \hat{\gamma}$ is chosen according to a desired duration of the movement. Then the time base generator can be started, generating a time-varying potential field $x^{st} = x^{st}(t)$ which excites the gradient-descent network. In general, we wish that this network tracks as precisely as possible the virtual target, i.e. we wish that at any time-instant $x^{st}(t)$ is as small as possible. On the other hand, if the motion of the target potential is sufficiently slow (i.e. if $\gamma$ is sufficiently small) then it is always possible to obtain any kind of positional precision. Thus, we have two contrasting requirements: timing precision.
Figure 3: Simulated trajectories for different values of the gradient gain $\alpha$

(measured by $\gamma - \hat{\gamma}$) vs spatial precision (measured by $\varepsilon^{st}$). The synchronization problem can then be formulated as a trade-off between the two requirements. In analytic terms, this is a very complex problem. We simply performed a preliminary simulation study, examining the effect of different values of the gradient gain $\alpha$ on the tracking error, without any synchronization mechanism, for a planar arm with 2 degrees of freedom. The results are reported in Fig. 3. We are currently investigating a synchronization strategy based on modulating the speed factor $\gamma - \hat{\gamma}$ as a function the tracking error $\varepsilon^{st}$.

References


