Learning Inverse Kinematics: Reduced Sampling Through Decomposition Into Virtual Robots

Vicente Ruiz de Angulo and Carme Torras

***Abstract*—We propose a technique to speedup the learning of the inverse kinematics of a robot manipulator by decomposing it into two or more virtual robot arms. Unlike previous decomposition approaches, this one does not place any requirement on the robot architecture, and thus, it is completely general. Parametrized self-organizing maps are particularly adequate for this type of learning, and permit comparing results directly obtained and through the decomposition. Experimentation shows that time reductions of up to two orders of magnitude are easily attained.**

***Index Terms*—Function approximation, learning inverse kinematics, parametrized self-organizing maps (PSOMs), robot kinematics.**

I. Introduction

NEURAL networks have proved useful for learning the inverse kinematics of robot manipulators, either lacking a well-defined model or needing online recalibration while functioning. Recently, the development of humanoid robots has raised the interest in this topic. Due to the many degrees of freedom involved, the learning task is usually restrained to a specific trajectory, and a fixed cost function is used to resolve redundancy. Following the trend of using localized representations, some researchers [10], [11], have applied a supervised algorithm—locally weighted projection regression—in this context, with promising results. Nevertheless, the task of learning the complete kinematics of general mechanisms remains challenging due to the large number of training samples (i.e., robot movements) required to attain an acceptable precision [2, ch. 7], [3].

Several attempts have been made at reducing the number of required samples, among them the use of hierarchical networks [4], [13], the learning of only the deviations from the nominal kinematics [5], and the use of a continuous representation by associating a basis function to each knot [12].

In [6] and [7], we proposed a practical trick that can be used in combination with all the methods above. It consists in decomposing the learning of the inverse kinematics into several independent and much simpler learning tasks. This was done at the expense of sacrificing generality: The procedure works only for some robot models subject to certain types of deformations. Specifically, the procedure assumes that the last three robot joints cross at a point, a condition satisfied by some classic robot architectures.

Here, we present another decomposition technique for learning inverse kinematics that is not limited by the above assumption. While being more general, it still retains the main advantage of the trick above: The input dimensionality of each of the tasks resulting from the decomposition is half that of the original one. Thus, for a given desired accuracy, if the number of training samples required to learn inverse kinematics directly is $O(q^n)$, through the decomposition it reduces to $O(q^{n/2})$, where n is the number of robot joints, and q is the number of sample points along each joint dimension. This yields an enormous reduction in the number of samples required for high-precision applications.

This paper is structured as follows. In the next section, we describe the proposed decomposition of the inverse kinematics mapping. Section III explains how the workings of two networks encoding the kinematics of the component virtual robots can be combined to provide the inverse kinematics of the original robot. Section IV introduces parametrized self-organizing maps (PSOMs). The following two sections are devoted to the training scheme and the way to retrieve the kinematics from the component PSOMs, respectively. In Section VII, some illustrative experimental results of learning with and without the decomposition are presented, permitting to quantify the savings obtained. Finally, some conclusions and prospects for future work are put forth in Section VIII.

II. Kinematics Decomposition

The technique described here is based on the idea of decomposing the kinematics of a serial manipulator into those of several "virtual robots." The advantage of the approach is that, since the component robots are much simpler than the original one, the learning of their inverse kinematics requires less sampling points to be acquired.

We will explain the technique using only two virtual robots (see Fig. 1). The extension to more virtual robots is straightforward. Let $\boldsymbol{\theta} = (\theta_1, \theta_2 \dots \theta_n)$ and $T_1, T_2, \dots, T_n$ be the joint angles and the associated transformation matrices, respectively, of the real robot. Therefore, the transformation matrix $\mathcal{F}$ associated to the end-effector reference frame is $\mathcal{F} = T_1 T_2 \dots T_n$. To decompose the real robot into the two virtual robots, we select

Manuscript received August 16, 2007; revised February 15, 2008 and May 13, 2008. First published October 10, 2008; current version published November 20, 2008. This work was supported in part by the Generalitat de Catalunya under the consolidated Robotics group, by the Spanish Ministry of Science and Education under Project DPI2007-60858, by the "Comunitat de Treball dels Pirineus" under Project 2006ITT-10004, and by the European Commission under Project PACO-PLUS, CogSys Integrated Project FP6-IST-4-27657. An earlier version of this paper was presented at IWANN-2005 [8]. This paper was recommended by Associate Editor H. Qiao.

The authors are with the Institut de Robòtica i Informàtica Industrial (CSIC-UPC), 08028 Barcelona, Spain (e-mail: ruiz@iri.upc.edu; torras@iri.upc.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2008.928232

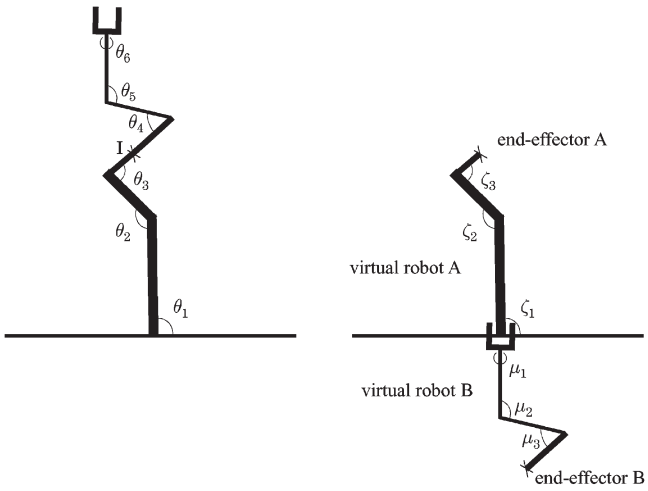


Fig. 1. Decomposing the (left) robot manipulator into (right) two virtual robot arms.

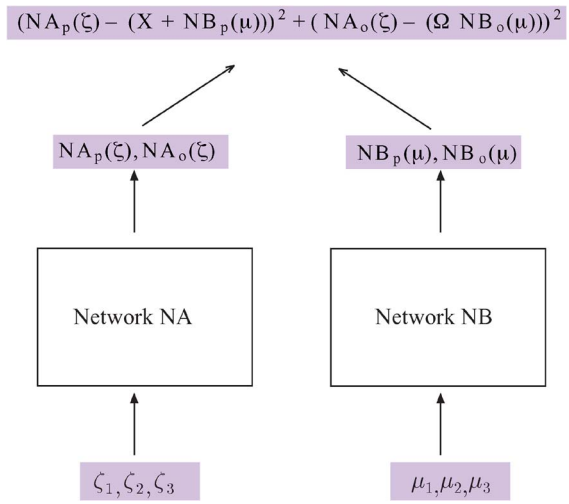


Fig. 2. Workings of the two networks are linked through the cost function at the top.

a point I of joint k. Thus, the reference frame $\mathcal{F}_I$ associated to I is rigidly linked to the reference frame of joint k and $\mathcal{F}_I = T_1 T_2 \ldots T_k T_c$, where T_c is a constant matrix. Ideally, $k = n/2$, as we will justify at the end of this section.

The first virtual robot, or robot A, has k joints $\boldsymbol{\zeta} = (\zeta_1, \zeta_2, \ldots, \zeta_k)$, and their associated transformation matrices are $T_1, T_2, \ldots T_{k-1}, T_k T_c$. The second robot, or robot B, is composed of $n - k$ joints $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_{n-k})$ with associated reference matrices $T_n^{-1}, T_{n-1}^{-1}, \ldots T_{k+2}^{-1}, T_{k+1}^{-1} T_c$.

We could consider that we have virtually broken the original robot into two pieces, exactly at point I of link k. Robot A is the first piece of the robot and has its end-effector at the extreme of the broken link. Robot B is the other piece of the original robot, the base of robot B being the original end-effector (translated and rotated to the origin of coordinates), and the end-effector of robot B, the extreme of the other half of the broken link. The second robot can also be seen as the remaining of the original robot, inverted and translated to the reference frame.

By $\theta = (\boldsymbol{\zeta}, \boldsymbol{\mu})$, we mean

$$\theta_i = \zeta_i \qquad \forall i = 1, \ldots, k \tag{1}$$

$$\theta_i = \mu_{n-i+1} \qquad \forall I = k+1, \ldots, n. \tag{2}$$

We denote $\mathrm{DK_A}(\boldsymbol{\zeta})$ and $\mathrm{DK_B}(\boldsymbol{\mu})$ the direct kinematics of robots A and B, respectively. It is easy to see that $\theta = (\boldsymbol{\zeta}, \boldsymbol{\mu})$ is a valid inverse kinematics solution for a given position X and orientation Ω of the real robot iff

$$\mathrm{DK_A}(\boldsymbol{\zeta}) = \mathrm{TR}(X, \Omega)\mathrm{DK_B}(\boldsymbol{\mu}) \tag{3}$$

where $\mathrm{TR}(X, \Omega)$ is the matrix transformation yielding a translation X and an orientation Ω. Note that the knowledge of the kinematics equations of the original and virtual robots is not required. We can replace $\mathrm{DK_A}$ and $\mathrm{DK_B}$ by black boxes found through learning, and, taking (X, Ω) as input, if equation (3) holds for ($\boldsymbol{\zeta}$, $\boldsymbol{\mu}$), these joint values are a solution of the inverse kinematics for the original whole robot.

Now, we can easily justify the way to choose k. As for the whole robot, we can assume that the number of samples that we need to approximate $\mathrm{DK_A}(\boldsymbol{\zeta})$ and $\mathrm{DK_B}(\boldsymbol{\mu})$ depend on the number of joints of robots A and B, respectively. Therefore, the number of samples needed by the decomposition approach is $q^k + q^{n-k}$, where q is the number of sample points in each joint dimension assuming that the samples are obtained following rectangular grids. The minimum of this quantity as a function of k occurs when $k = n/2$ and increases almost exponentially as k differs from the minimum.

III. Kinematics Composition

The approach consists in creating two neural networks NA and NB approximating the functions $\mathrm{DK_A}(\boldsymbol{\zeta})$ and $\mathrm{DK_B}(\boldsymbol{\mu})$ (see Fig. 2). When a desired pose (X, Ω) is given, a search in the inputs of the two networks is carried out to find values of $\boldsymbol{\zeta}$ and $\boldsymbol{\mu}$ satisfying (3) as much as possible. This can be done by imposing a common cost function to be minimized in (ζ, μ) such as

$$\|\mathrm{NA}(\boldsymbol{\zeta}) - \mathrm{TR}(X, \Omega)\mathrm{NB}(\boldsymbol{\mu})\|^2 \tag{4}$$

or by decomposing the output of the networks into two components: position ($\mathrm{NA_p}$, $\mathrm{NB_p}$) and orientation ($\mathrm{NA_o}$, $\mathrm{NB_o}$), and then minimizing

$$\|\mathrm{NA_p}(\boldsymbol{\zeta}) - (X + \mathrm{NB_p}(\boldsymbol{\mu}))\|^2 + \|\mathrm{NA_o}(\boldsymbol{\zeta}) - (\Omega\ \mathrm{NB_o}(\boldsymbol{\mu}))\|^2. \tag{5}$$

The dimension of $\mathrm{NA_p}$ and $\mathrm{NB_p}$ is, of course, 3. The dimension of $\mathrm{NA_o}(\boldsymbol{\zeta})$ and $\Omega\ \mathrm{NB_o}(\boldsymbol{\mu})$ depends on the representation chosen for orientations. In our case, this representation (explained in Section VII) is of dimension 6. If NA and NB approximate reasonably well the kinematics of robots A and B, respectively, for a reachable pose (X, Ω), the minimum of (5) will be zero. However, under a pragmatic point of view, even in this case, a numerical algorithm will output some point slightly different from the minimum, with a value not exactly zero. This means that the output of the minimization will depend somewhat on the units used for position coordinates (our representation of orientation is unit free, since it is composed

of elements of the rotation matrix). In our case, "meter" units yield an equilibrated balance in the minimization of position and orientation, but a weighting constant can be introduced to correct or manipulate this balance.

To facilitate the search, we require the output of the neural networks to be differentiable with respect to the input. We consider that the memory-based neural networks are particularly well suited to our application, since they use stored function points to build the approximation of the function. On the one hand, they allow a quick search among the stored points to find a good starting point for continuous minimization. On the other hand, we can apply $\mathrm{TR}(X, \Omega)$ to the stored points of network NB, so that the whole approximation of the function gets translated and rotated, becoming NB'. In this way, the function to be minimized (4) becomes $\|\mathrm{NA}(\boldsymbol{\zeta}) - \mathrm{NB}'(\boldsymbol{\mu})\|^2$, whose derivatives are more easily obtained.

A PSOM [12] is the type of network better suited to our requisites. It approximates a function using a regular grid of sampled points. Because of its excellent interpolation capabilities, the required number of points is very small. Of particular interest to us is that PSOMs treat input and output variables in the same way. This means that it is as natural to ask which output corresponds to a given input as asking which input correspond to a given output. Therefore, our search in the input variables is naturally addressed and embedded in the framework of these networks.

IV. PSOM

We give a brief overview of the main PSOM concepts introduced in [12]. A PSOM is basically a continuous extension of the Kohonen self-organized maps (SOM). A SOM is a neural network producing low-dimensional representations of the training samples while preserving the topology of the data space. The units are arranged in an m-dimensional grid. Usually, m is 2, and, in any case, smaller than the dimensionality of the data space d. Each input is connected to all units. Attached to every unit, there is a reference weight vector of the same dimensionality as the data. The output of the network to a new input is the attached vector closest to the data. The SOM learning algorithm modifies these attached vectors to become good representatives of the training data. Besides, the learning algorithm arranges the units so that two units topologically close in the grid tend to have also similar attached vectors. Thus, a SOM induces a discrete topology-preserving mapping from grid coordinates to data space.

A PSOM turns this mapping into a continuous one. The mapping in this model goes from continuous grid coordinates $S \subset \mathbb{R}^m$ to data space, $\mathbf{w}(\mathbf{s}) : S \rightarrow \mathbb{R}^d$, which will be referred to as the interpolated manifold. The data space includes all components irrespective of which ones a usual supervised neural network model would consider as input data or output data for the mapping to be learned. The response of the network to a new data vector $\mathbf{x}$ is $\mathbf{w}(\mathbf{s}^*)$ s.t. $\mathbf{s}^* \in S$, the closest point to $\mathbf{x}$ in the interpolated manifold, i.e.,

$$\mathbf{s}^* = \arg\min_{\mathbf{s}} \ \mathrm{dist}\left(\mathbf{w}(\mathbf{s}), \mathbf{x}\right). \tag{6}$$

The distance function is defined as

$$\mathrm{dist}(\mathbf{x}, \mathbf{x}')^2 = \sum p_j \left(x_j - x'_j\right)^2. \tag{7}$$

Setting some of the weights p_js to zero, $\mathbf{w}(\mathbf{s}^*)$ becomes an associative completion of the data whose weights are not zero. Therefore, a partial data vector (missing the components whose $p_j = 0$) could be considered as an input, and the components of $\mathbf{w}(\mathbf{s}^*)$ corresponding to null p_js could be considered as an output. This means that we can select arbitrarily the components of the data that we consider input or output. $\mathbf{w}(\mathbf{s})$ is chosen to have the form of a sum of products of reference vectors in $\mathbb{R}^d$ and their associated basis functions

$$\mathbf{w}(\mathbf{s}) = \sum_{\mathbf{a} \in \mathbf{A}} H(\mathbf{a}, \mathbf{s})\ \mathbf{w}_{\mathbf{a}} \tag{8}$$

where $\mathbf{A}$ is the set of grid coordinates of all the units of the PSOM. The basis functions $H(\mathbf{a}, \mathbf{s})$ determine the degree of contribution of the reference vector $\mathbf{w}_{\mathbf{a}}$ based on the distance between the grid coordinates $\mathbf{a}$ and $\mathbf{s}$. The reference vectors are, as a matter of fact, the training data. The basis functions $H(\mathbf{a}, \mathbf{s})$ are Lagrange polynomials chosen to interpolate exactly the reference vectors. To allow the interpolation, the reference vectors must constitute an m-dimensional rectangular hyperlattice, resulting from the Cartesian product of m 1-D point sets.

V. Learning the Inverse Kinematics of the Virtual Robots

Usual inverse kinematics learning requires the capability to observe the position and orientation of the robot end-effector, represented by the transformation matrix $\mathcal{F}$, as defined in Section II. Our method requires also knowing the position and orientation of the point I, encapsulated in the transformation matrix $\mathcal{F}_I$. Thus, our method needs to acquire twice the number of sensory measures required by the classic procedure, which can be obtained using the same sensory system already in place.

Every time the robot performs a movement (even during working operation), a sample point for each of the virtual robots can be obtained. The learning amounts to supplying virtual robot A with a sample point consisting of an input $\zeta_i = \theta_i$, $i = 1, \ldots, k$ and an output $\mathcal{F}_I$. For robot B, the sampling point has as input $\mu_i = \theta_{n-i+1}$, $i = 1, \ldots, n-k$ and as output $\mathcal{F}^{-1}\mathcal{F}_I$. We could understand this as moving the whole robot B "freezed" in its current configuration to the place it is supposed to be, before extracting its kinematics sample point.

VI. Computing Kinematics With PSOMs

When using PSOMs to learn the kinematics of robots, the movements to obtain the data vectors are generated following a regular grid in the space of joint angles. The training data vectors (as well as the data interpolated manifold and the query vector $\mathbf{x}$) have as components all the joint values and all pose values.

Once trained, a PSOM works by putting some constraints on a subset of the variables of the system (input or output), e.g., fixing them to a desired value. This is achieved by setting the corresponding p_j's in (7) to 1. These variables will be

Algorithm 1: PSOMInverseKinematics algorithm.

Input: A Desired Pose, $DP \equiv (DP_1, \ldots, DP_t)$
A PSOM, N, with a set of training vectors $\{\mathbf{w_a}\ s.t.\ \mathbf{a} \in \mathbf{A}\}$ following a grid in the n joint angles of the robot. Its distance weight vector, current point in S, and current point in the interpolated manifold are $N.\mathbf{p}$, $N.\mathbf{s}$, $N.\mathbf{w}(\mathbf{s})$, respectively
Output: A vector of n joint values that approximately generates the robot pose DP

$\mathbf{x} \leftarrow (DP_1 \ldots DP_t, dummy, \ldots, dummy)$; /* the last n components can take any value */
$N.\mathbf{p} \leftarrow (1, .^{t)}., 1, 0, .^{n)}., 0)$ /* the last n components corresponding to the robot joint values are zero */
$N.\mathbf{s} \leftarrow \mathbf{a}$ s.t. $\mathbf{w_a}$ minimizes $dist(\mathbf{w_a}, \mathbf{x})$
while *not Minimum*($N.\mathbf{s}$) **do**
  $\Delta\mathbf{s} \leftarrow$ MinimizationStep($dist(N.\mathbf{w}(\mathbf{s}), \mathbf{x})$)
  $N.\mathbf{s} \leftarrow N.\mathbf{s} + \Delta\mathbf{s}$
return Last n components of $N.\mathbf{w}(\mathbf{s})$

considered as input. The other p_j's are set to zero, and their corresponding variables are considered as output. Therefore, by manipulating the p_j's, one can obtain the forward or the inverse kinematics at will. The system then carries out a quick optimization aimed to find a point of the approximated input–output manifold satisfying the constraints or, if impossible, the closest one to satisfying them. The starting point of the process is the stored point that best satisfies the constraints. From it, an iterative minimization procedure is launched, which finishes in a few steps.

For PSOMs trained on the kinematics of a robot, to get the inverse kinematics (Algorithm 1), we simply fix the position and orientation variables, and we let the minimization get the point in the interpolating surface with the desired pose values, the remaining components of the point are taken to be the result, i.e., the joint values yielding the desired pose.

To obtain the inverse kinematics of the real whole robot using the PSOMs for the virtual component robots (Algorithm 2), we first transform the points stored in NB with the desired pose to become NB$'$, as explained in Section III. Afterward, we look for a good starting point for the minimization by finding the closest pair (in pose space) between the points stored in NA and the transformed points in NB$'$. Let $(\mathrm{NA}.\mathbf{w}(\mathbf{s}_0), \mathrm{NB}'.\mathbf{w}(\mathbf{s}_0))$ be this closest pair, which we will denote $(A_{0)}, B'_{0)})$ for short. Likewise, $A_{i)}$ and $B'_{i)}$ will denote points in the interpolated manifold (containing pose coordinates and joint values) obtained in intermediate minimization stages for networks NA and NB$'$, respectively. A minimization step is then carried out in NA with $B'_{0)}$ as target pose, and another step is performed in NB$'$ with $A_{0)}$ as desired pose. The result of these steps in NA and NB$'$ are two points $A_{1)}$, and $B'_{1)}$, respectively. These points will be the starting ones for the following steps in which the desired poses for NA and NB$'$ will be $B'_{1)}$ and $A_{1)}$, respectively. More iterations will be performed in the same way, until the pose components of $A_{i)}$ and $B'_{i)}$ are closer than a certain threshold. Thus, the function being minimized is (5) or, equivalently, $\mathrm{dist}(\mathrm{NA}.\mathbf{w}(\mathbf{s}), \mathrm{NB}'.\mathbf{w}(\mathbf{s}))$. If the desired pose cannot be reached, the above termination criterion may never be satisfied. An alternative termination criterion, which is the one we have implemented, is that the norm of the gradient of the function being minimized is below a certain threshold. Then, we extract the inverse kinematics of the real whole robot by concatenating the joint components of the last obtained points.

Algorithm 2: CompositeInverseKinematics algorithm.

Input: A Desired Pose, $DP \equiv (DP_1, \ldots, DP_t)$, whose matrix transformation representation is $TR(X, \Omega)$
A PSOM, NA, with a set of training vectors $\{\mathbf{w_a}\ s.t.\ \mathbf{a} \in \mathbf{A}\}$ following a grid in the k joint angles of virtual robot A. Its distance weight vector, current point in S, and current point in the interpolated manifold are $NA.\mathbf{p}$, $NA.\mathbf{s}$, $NA.\mathbf{w}(\mathbf{s})$, respectively
A PSOM, NB, with a set of training vectors $\{\mathbf{w_b}\ s.t.\ \mathbf{b} \in \mathbf{B}\}$ following a grid in the $n-k$ joint angles of virtual robot B.
Output: A joint vector that approximately generates the pose DP in the real robot

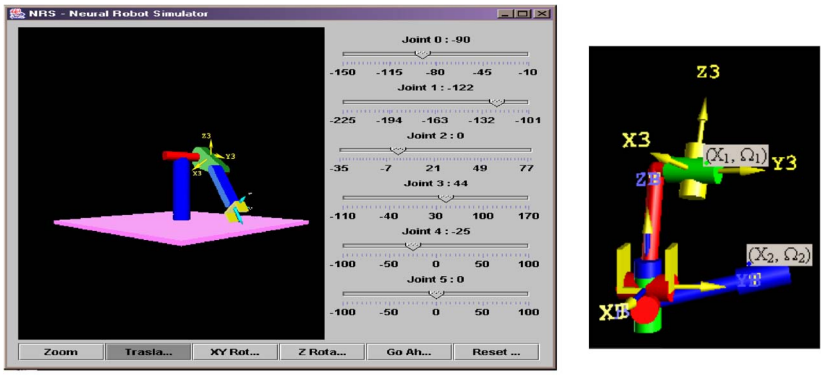
Generate a new PSOM: $NB' \leftarrow NB$
foreach $\mathbf{b}' \in \mathbf{B}'$ **do**
  $\mathbf{w_{b'}} \leftarrow$ Apply transformation $TR(X, \Omega)$ to the pose components of $\mathbf{w_{b'}}$
$NA.\mathbf{p} \leftarrow (1, .^{t)}., 1, 0, .^{k)}., 0)$
$NB'.\mathbf{p} \leftarrow (1, .^{t)}., 1, 0, .^{n-k)}., 0)$
$(NA.\mathbf{s}, NB'.\mathbf{s}) \leftarrow (\mathbf{a} \in \mathbf{A}, \mathbf{b}' \in \mathbf{B}')$ s.t. $\mathbf{w_a}, \mathbf{w_{b'}}$ minimizes $dist(\mathbf{w_a}, \mathbf{w_{b'}})$
while *not Minimum*($NA.\mathbf{s}, NB'.\mathbf{s}$) **do**
  /* Equivalent to a usual minimization step of NA with $NB'.\mathbf{w}(\mathbf{s})$ as desired value */
  $\Delta\mathbf{s}_A \leftarrow$ MinimizationStep($dist(NA.\mathbf{w}(\mathbf{s}), NB'.\mathbf{w}(\mathbf{s}))$) on $NA.\mathbf{s}$
  $NA.\mathbf{s} \leftarrow NA.\mathbf{s} + \Delta\mathbf{s}_A$
  /* Equivalent to a usual minimization step of NB' with $NA.\mathbf{w}(\mathbf{s})$ as desired value */
  $\Delta\mathbf{s}_B \leftarrow$ MinimizationStep($dist(NA.\mathbf{w}(\mathbf{s}), NB'.\mathbf{w}(\mathbf{s}))$) on $NB'.\mathbf{s}$
  $NB'.\mathbf{s} \leftarrow NB'.\mathbf{s} + \Delta\mathbf{s}_B$
return Last k components of $NA.\mathbf{w}(\mathbf{s})$ and last $n-k$ components of $NB'.\mathbf{w}(\mathbf{s})$

VII. Experimental Results

The experiments have been performed in a new general simulation environment developed at our institute, which allows the visualization of any serial manipulator (Fig. 3). The only input needed for the simulator is a Denavit–Hartenberg table, from which the graphical model is created using a uniform link and joint representation.

We used a PSOM variant known as LPSOM, the L standing for Local. This model builds a PSOM extracting for each query a subgrid of the sampling grid, which is centered on the closest point to the query. We have used subgrids with four points in each dimension. The representation of pose orientation has thoroughly been studied, and different alternatives have experimentally been compared [9]. There exist many possible representations, but none is completely satisfactory. For example, the Euler representation is very compact, but lacks continuity. This drawback affects also other in principle good candidate representations such as quaternions. The classical 3×3 rotation matrix is continuous but not compact. The solution was to select a subset of elements of the standard rotation matrix that determine it. The five elements in the last column and row are good in general, although not perfect because the matrix is not determined in one point (when the common element of the last row and last column takes the value 0). Therefore, it is safer to use six elements, the last two columns of the rotation matrix, which completely determine it.

We have chosen the well-known PUMA robot to validate our technique. The experiments were carried out using a very large workspace, allowing ranges for the six joints [1] as follows: [−150,−35], [−215,−100], [−35,80], [−110,5], [−100,15], [−100,15]. We trained one LPSOM in a classical way, by generating samples of the kinematics of the robot in a regular grid in the joint space covering the workspace above. Then,



(a) (b)

Fig. 3. (a) View of the neural robot simulator showing the 3-D representation of a PUMA-like robot arm. (b) Two virtual subrobots in which the robot in (a) is decomposed. (X_1, Ω_1) and (X_2, Ω_2) are the position and orientation of subrobot A and subrobot B, respectively.

TABLE I
CLASSIC ALGORITHM

number of movements	position mean error	position stdev. error	orientation mean error	orientation stdev. error
64	476	229	53	36
729	46	21	5.8	2.81
4096	11	17	0.69	1.55

TABLE II
DECOMPOSITION ALGORITHM

number of movements	position mean error	position stdev. error	orientation mean error	orientation stdev. error
8	377	236	44	37
27	48	42	5.27	2.58
64	10	35	0.92	3.38
125	3.6	27	0.29	2.80
216	2.1	8.3	0.11	0.63
343	1.6	6.4	0.11	0.80
512	0.9	2.9	0.11	1.20

we moved the robot to the different configurations represented in the grid to obtain the associated positions and orientations. Thus, each knot in the grid requires one movement. The results are shown in Table I. The units are millimeters and degrees.

In the experiment to test our decomposition approach, we used two smaller PSOMs, one for each of the two virtual robots A and B. The corresponding regular grids were also generated. In this case, with only one movement of the robot, we get the required information for one unit in the grid associated to robot A and for another unit in the grid associated to robot B. Table II shows the results. The comparison of both tables reveals that, for the only number of points in common (64), the averages in position and orientation are around 50 times more precise for the decomposition algorithm. We note also that the limits of physical accuracy of the manipulator are approximately reached with 512 movements with the decomposition algorithm, whereas it was impossible with our computer memory resources (allowing grids of up to 262 144 points) to reach precisions under 1 mm and 0.6° with the classic procedure.

To test the scalability of the proposed approach, we have enlarged the workspace up to 9 times the preceding one: [−150,15], [−215,−50], [−35,130], [−110,55], [−100,65], [−100,65]. Fig. 4 permits comparing the results obtained for the two workspaces under the two approaches. While the scalability of the classic approach is already quite good that of the decomposition approach is remarkable.

VIII. CONCLUDING REMARKS AND FUTURE WORK

The purpose of this paper is to propose a technique to learn inverse kinematics (IK) with a reasonable number of movements when high accuracy is required.

Unlike our previous work on IK learning through function decomposition [6], [7], the technique here proposed does not place any restriction on the type of robot architecture to which it can be applied. The kinematics of any serial manipulator undergoing whatever deformation can be learned with this technique. However, a new "sensorial" requisite must be fulfilled: the reference frame attached to a point in an intermediate link of the robot must be known using some sensing system. We think that this is not a shortcoming, since learning IK with any procedure requires anyway a sensorial system to determine the position and orientation of the gripper. Notice that by duplicating the number of sensory inputs, we obtain training time reductions of up to two orders of magnitude.

One of the most promising applications of our technique is to flexible robots. Since it reduces the dimensionality of the functions to be learned from six to three, it is still affordable to include the load change as an extra variable and still have quick learning.

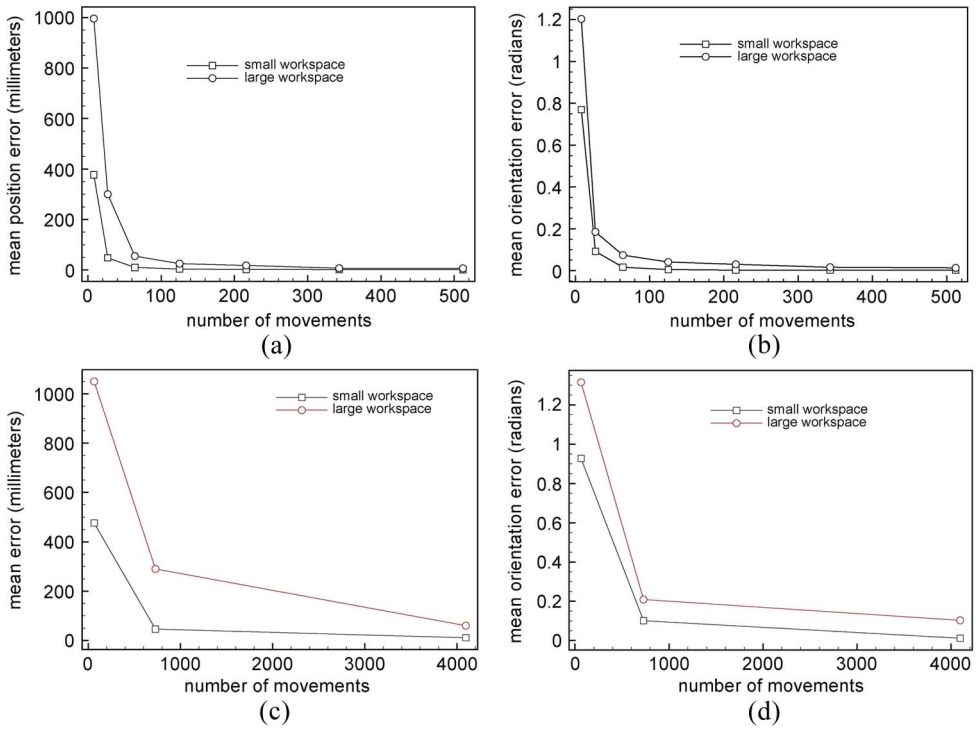


Fig. 4. Testing the scalability of (a) and (b) the proposed and (c) and (d) the classic approaches. The graphs on the left display position errors and those on the right, orientation errors.

In addition to learning efficiency, our technique has other advantages over classic IK learning. For instance, it allows the robot to learn to move in the complete workspace without actually moving everywhere, and to approach risky zones only after learning has been successfully completed. Moreover, we have shown that the proposed technique scales well to workspace enlargement.

Among the tasks left for future work, we can mention testing the extension of this framework to more than two virtual robots. Also, we think that appropriately weighting the learning of the position and orientation of the two virtual robots can further improve the results. The inaccuracies in the interpolated position of the virtual subrobots are simply added (vectorially) in the composite robot. Instead, inaccuracies in the orientation components of the subrobots result in orientation inaccuracies of the same order in the composite robot, but also add a possibly large error component in position.

An open issue common to all the approaches to IK learning is the representation of orientation. We think that the goodness of representations for orientation can be evaluated with respect to three criteria: 1) compactness; 2) continuity; and 3) whether interpolated representations are proper representations. Compactness saves memory (particularly in memory-based models) and should influence positively generalization. However, continuity (two close orientations in the representation space should also be close regarding robot motion) has a more radical influence on the quality of the interpolation. Finally, it is desirable that every interpolated representation corresponds to a true orientation. Otherwise, one has the problem of how to map interpolated values onto the representation space, as it occurs with rotation matrices. By choosing as representation six elements of the rotation matrix, we have given priority to the continuity criterion, while trying to maximize compactness. Interpolated representations do not necessarily correspond to points inside the representation space, but this does not seem a big problem in practice.

REFERENCES

[1] K. S. Fu, R. C. González, and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*. New York: McGraw-Hill, 1987.
[2] B. J. A. Kröse and P. P. van der Smagt, "Robot control," in *An Introduction to Neural Networks*, 5th ed. Amsterdam, The Netherlands: Univ. Amsterdam, 1993.
[3] T. M. Martinetz, H. J. Ritter, and K. J. Schulten, "Three-dimensional neural net for learning visuomotor coordination of a robot arm," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 131–136, Mar. 1990.
[4] H. Ritter, T. Martinetz, and K. J. Schulten, *Neural Computation and Self-Organizing Maps*. New York: Addison-Wesley, 1992.
[5] V. Ruiz de Angulo and C. Torras, "Self-calibration of a space robot," *IEEE Trans. Neural Netw.*, vol. 8, no. 4, pp. 951–963, Jul. 1997.
[6] V. Ruiz de Angulo and C. Torras, "Learning inverse kinematics via cross-point function decomposition," in *Proc. ICANN*, 2002, vol. 2415, pp. 856–861.
[7] V. Ruiz de Angulo and C. Torras, "Speeding up the learning of robot kinematics through function decomposition," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1504–1512, Nov. 2005.
[8] V. Ruiz de Angulo and C. Torras, "Using PSOMs to learn inverse kinematics through virtual decomposition of the robot," in *Proc. 8th IWANN*, 2005, vol. 3512, pp. 701–708.
[9] D. Saune Sánchez, "Recalibracion de un brazo robot mediante tecnicas de descomposicion de la cinematica," in *Proyectonal de Carrera*. Barcelona, Spain: Dept. LSI, Univ. Politècnica de Catalunya, 2003.
[10] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *Proc. IEEE/RSJ Int. Conf. IROS*, 2001, vol. I, pp. 203–298.
[11] S. Vijayakumar, A. D'Souza, T. Shibata, J. Conradt, and S. Schaal, "Statistical learning for humanoid robots," *Auton. Robots*, vol. 12, no. 1, pp. 55–69, Jan. 2002.
[12] J. Walter and H. Ritter, "Rapid learning with parametrized self-organizing maps," *Neurocomputing*, vol. 12, pp. 131–153, 1996.
[13] J. Walter and K. J. Schulten, "Implementation of self-organizing neural networks for visuo-motor control of an industrial robot," *IEEE Trans. Neural Netw.*, vol. 4, no. 1, pp. 86–96, Jan. 1993.

Vicente Ruiz de Angulo was born in Miranda de Ebro, Burgos, Spain. He received the B.Sc. and Ph.D. degrees in computer science from the Universidad del País Vasco, Bilbao, Spain.

During the academic year 1988–1989, he was an Assistant Professor with the Universitat Politècnica de Catalunya, Barcelona, Spain. In 1990, he was with the Neural Network Laboratory, Joint Research Center of the European Union, Ispra, Italy. From 1995 to 1996, he was with the Institut de Cibernètica, Barcelona, participating in the ESPRIT project entitled "Robot Control Based on Neural Network Systems" (CONNY). He also spent six months with the Istituto Dalle Molle di Studi Sull' Inteligenza Artificiale di Lugano, working in applications of neural networks to robotics. Since 1996, he has been with the Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Barcelona. His interests in neural networks include fault tolerance, noisy and missing data processing, and their application to robotics and computer vision.

Carme Torras received the M.Sc. degree in mathematics from the Universitat de Barcelona, Barcelona, Spain, the M.Sc. degree in computer science from the University of Massachusetts, Amherst, and the Ph.D. degree in computer science from the Universitat Politècnica de Catalunya, Barcelona.

She is currently with the Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Barcelona. She is a Research Professor with the Spanish Scientific Research Council, Spain. She has published four books and more than 100 papers in the areas of robot kinematics, geometric reasoning, computer vision, and neurocomputing. She has been Local Project Leader of several European projects, such as "Planning RObot Motion" (PROMotion), "Robot Control based on Neural Network Systems" (CONNY), "Self-organization and Analogical Modelling using Subsymbolic Computing" (SUBSYM), "Behavioural Learning: Sensing and Acting" (B-LEARN), and the ongoing 6th framework IP project "Perception, Action and COgnition through Learning of Object-Action Complexes" (PACO-PLUS).