Using PSOMs to Learn Inverse Kinematics Through Virtual Decomposition of the Robot

Vicente Ruiz de Angulo and Carme Torras

Institut de Robòtica i Informàtica Industrial (CSIC-UPC)
Llorens i Artigas 4-6, 08028-Barcelona, Spain.
{ruiz, torras}@iri.upc.es,
WWW home page: www-iri.upc.es

Abstract. We propose a technique to speed up 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 (PSOM) are particularly adequate for this type of learning, and permit comparing results obtained directly and through the decomposition. Experimentation shows that time reductions of up to two orders of magnitude are easily attained.

1 Introduction

Neural networks have proved useful for learning the inverse kinematics of robot manipulators, either lacking a well-defined model or needing on-line recalibration while functioning. The main shortcoming is the large number of training samples (i.e., robot movements) required to attain an acceptable precision [2, 3].

Several attempts have been made at reducing the number of required samples, among them the use of hierarchical networks [4, 10], 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 [9].

In [6, 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 $O(n^d)$, through the decomposition it reduces to $O(n^{d/2})$. This yields an enormous reduction in the number of samples required for high-precision applications.

The paper is structured as follows. In the next section we describe the proposed decomposition of the inverse kinematics mapping. Section 3 explains how the workings of two parameterized self-organizing maps (PSOMs) encoding the kinematics of the component virtual robots can be combined to provide the inverse kinematics of the original robot. The following two sections are devoted to the training scheme and the way to retrieve the kinematics from the component PSOMs, respectively. In Section 6, 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 7.

2 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 straighforward. Let $\theta = (\theta_1, \theta_2 ... \theta_n)$ and $T_1, T_2, ..., T_n$ be the joints and the associated transformation matrices, respectively, of the real robot. To connect the two robots we select a reference frame $\mathcal{A}$ rigidly linked to A_k, the reference frame of joint k. Thus, $\mathcal{A} = A_k A_c$, where A_c is a constant matrix. It can be, for example, a reference frame centered on any point I of link k. Ideally $k = n/2$.

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

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 $\theta = (\zeta, \mu)$ we mean

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

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

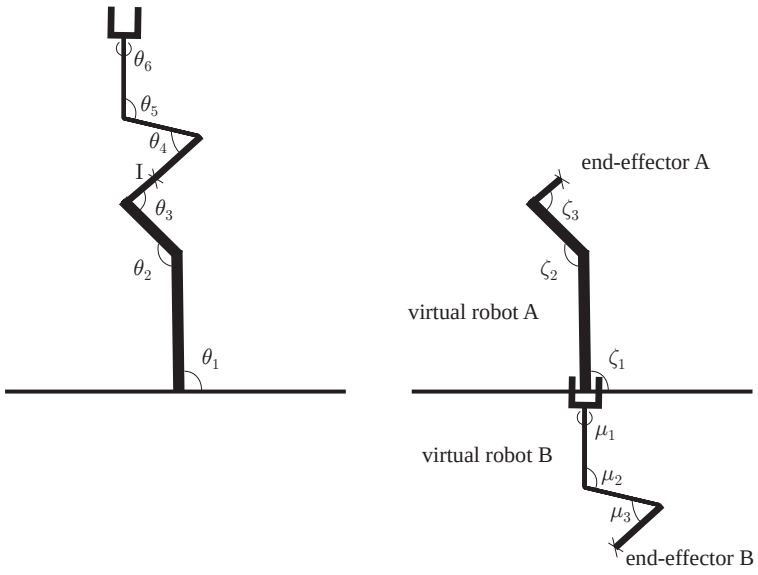

Fig. 1. Decomposing the robot manipulator (left) into two virtual robot arms (right).

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

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

where $TR(X, \Omega)$ is the matrix transformation yielding a translation X and an orientation Ω.

3 Kinematics Composition

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

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

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

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

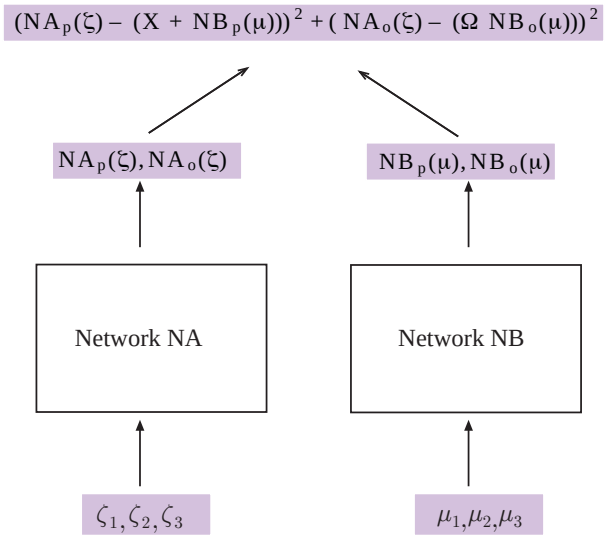

Fig. 2. The workings of the two networks are linked through the cost function at the top.

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 especially 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 $TR(X, \Omega)$ 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 becomes $(NA(\zeta) - NB'(\mu))^2$, whose derivatives are more easily obtained.

A Parametrized Self-Organized Map (PSOM) [9] 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.

4 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 ma-

trix M. Our method requires also knowing the position and orientation of the point I, encapsulated in the transformation matrix M_I.

Every time the robot performs a movement (even during working operation), a sampling point for each of the virtual robots can be obtained. The learning amounts to supplying virtual robot A with a sampling point consisting of an input $\zeta_i = \theta_i, i = 1...k$ and an output M_I. For robot B the sampling point has as input $\mu_i = \theta_{n-i+1}, i = 1...n-k$ and as output $M^{-1}M_I$. 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.

When using PSOMs to learn the kinematics of the virtual robots, the movements are generated following a regular grid in the space of joint angles.

5 Computing Kinematics with PSOMs

Once trained, a PSOM works by putting some constraints on a subset of the variables of the system (input or output), for example fixing them to a desired value. 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 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. To obtain the inverse of the real robot using the PSOMs for the virtual robots, we first transform the points stored in NB with the desired pose, as explained in Section 3. Afterwards, 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 $(A_{0)}, B_{0)})$ be this closest pair. A minimization step is then carried out in NA with $B_{0)}$ as target pose, and another step is done in NB with $A_{0)}$ as desired pose. The result of the step in NA and NB are two points whose pose components are $A_{1)}$, and $B_{1)}$, respectively. These points will be the starting point for the following steps in which the desired poses for NA and NB will be $B_{1)}$ and $A_{1)}$, respectively. More iterations are performed in the same way, until $A_{i)}$ and $B_{i)}$ are closer than a certain threshold. Then we extract the inverse kinematics of the real robot by concatenating the joint components of the last obtained points.

6 Experiments

The experiments have been executed in a new general simulation environment developed at our institute, which allows the visualization of any serial manipulator. 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. 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. The representation of pose orientation has been thoroughly studied and different alternatives have been compared experimentally [8]. 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 a value of 0). Therefore, it is safer to use 6 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 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 1. The units are millimeters and radians.

Table 1. Classic algorithm.

number of movements	position mean error	position stdev. error	orientation mean error	orientation stdev. error
64	476	229	0.927	0.635
729	46	21	0.101	0.049
4096	11	17	0.012	0.027

Table 2. Decomposition algorithm.

number of movements	position mean error	position stdev. error	orientation mean error	orientation stdev. error
8	377	236	0.770	0.653
27	48	42	0.092	0.045
64	10	35	0.016	0.059
125	3.6	27	0.005	0.049
216	2.1	8.3	0.002	0.011
343	1.6	6.4	0.002	0.014
512	0.9	2.9	0.002	0.021

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 knot of robot A and another knot of robot B. Table 2 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 .01 radians with the classic procedure.

7 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 doesn't 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.

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 6 to 3, it is still affordable to include the load change as an extra variable and still have quick learning.

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 risk zones only after learning has been successfully completed.

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 composed robot. Instead, inaccuracies in the orientation components of the subrobots result in orientation inaccuracies of the same order in the composed 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 (especially in memory-based models) and should influence positively generalization. But continuity (two close orientations in the representation space should also be close as regards to robot motion) has a more radical influence in 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 happens 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 correspond necessarily to points inside the representation space, but this does not seem a big problem in practice.

Acknowledgements: This work was supported by the I+D project DPI 2004–07358 of the Spanish Ministry of Education.

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: An Introduction to Neural Networks (5th edition), Chapter **7**: "Robot Control". University of 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. on Neural Networks **1** (1990) 131–136
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 Torras C.: Self-calibration of a space robot. IEEE Trans. on Neural Networks (1997) **8** 951-963
6. V. Ruiz de Angulo and C. Torras: Learning inverse kinematics via cross-point function decomposition. Proc. Intl. Conf. on Artificial Neural Networks (ICANN-02), Lecture Notes in Computer Science (2002) **2415** 856-861
7. V. Ruiz de Angulo and C. Torras: Speeding up the learning of robot kinematics through function decomposition. IEEE Trans. on Neural Networks (to appear)
8. D. Saune Sánchez: Recalibración de un brazo robot mediante técnicas de descomposición de la cinemática. Proyecto final de carrera, Departament LSI, Universitat Politècnica de Catalunya (2003)
9. J. Walter and H. Ritter: Rapid learning with parametrized self-organizing maps. Neurocomputing (1996) **12** 131-153
10. J. Walter and K.J. Schulten: Implementation of self-organizing neural networks for visuo-motor control of an industrial arm. IEEE Trans. on Neural Networks (1993) **4**