Robot Arm Control

Carme Torras

Institut de Robòtica i Informàtica Industrial (CSIC-UPC)

Llorens i Artigas 4-6, 08028-Barcelona.

RUNNING HEAD: Robot Arm Control

Correspondence:

Carme Torras

Institut de Robòtica i Informàtica Industrial (CSIC-UPC)

Parc Tecnològic de Barcelona - Edifici U

Llorens i Artigas 4-6, 08028-Barcelona, SPAIN.

Phone: 34-93-401.57.90

Fax: 34-93-401.57.50

e-mail: ctorras@iri.upc.es

INTRODUCTION

A robot is a multifunctional and reprogrammable mechanism able to move in a given environment. Three broad classes of robots can be distinguished on the basis of their mobility: *Robot arms* have a fixed base and their mobility comes from their articulated structure, thus operating on a bounded 3D subspace. *Robot vehicles* move on 2D surfaces by using wheels or other similar continuous traction elements (see ROBOT NAVIGATION). *Walking robots* are designed to move through rough terrains by using articulated legs. Of course, mixed possibilities do also exist like robot arms mounted on wheeled vehicles. This chapter is devoted to robot arms or manipulators, which will be simply called robots hereafter.

Each robot is endowed with a controller that commands its mechanical structure to perform the desired tasks. Controllers are usually *hierarchically* structured from the lowest level of servomotors to the highest levels of trajectory generation and task supervision. The activity taking place at all these levels is conceptually the same: an actual motion (of a single joint, the end-effector or the entire robot) is made to follow as closely as possible a commanded motion through the use of feedback. The difference lies in the coordinate systems used at each level.

At least four coordinate spaces can be distinguished: the taskspace (used to specify tasks, possibly in terms of sensor readings), the workspace (6D Cartesian coordinates defining a position and orientation of the end-effector), the joint space (intrinsic coordinates determining a robot configuration) and the actuator space (in which actual motions are commanded). Since robot control entails transforming a specification in taskspace into actuator commands, it critically depends on accurate mappings between the aforementioned coordinate spaces (see GEOMETRICAL PRINCIPLES IN MOTOR CONTROL).

Neural networks have been used to approximate these mappings when they are difficult or impossible to derive analytically (such as in the case of flexible or redundant robots, or in tasks entailing sensorimotor coordination) and when, because of environmental changes or robot wear-and-tear, the mappings vary in time and the controller needs to adapt on-line to these variations. This chapter discusses four such mappings and the neural models used to implement them adaptively.

NEURAL ADAPTIVITY IN ROBOT CONTROL

A robot, when moving, can be thought of as realizing a mapping from actuator space to joint space, and to the workspace and taskspace as well. These are referred to as *forward mappings*, which are parametrized by the current state of the robot. In the same very general terms, a controller can be viewed as implementing *inverse mappings*, in that, given the current state and a desired output (sensory pattern or robot pose), the controller has to generate the appropriate commands to attain that output.

Control strategies often rely on models of the mappings above. Forward models are used to provide fast internal feedback in order to prevent instabilities in the control loop. Inverse models lie at the core of feedforward control. The learning of such models by means of neural networks is described in SENSORIMOTOR LEARNING, whereas their use inside robot controllers is discussed in ROBOT LEARNING.

For the purpose of this chapter, let us mention that inverse models can be acquired under four schemes, namely *direct inverse modelling*, *feedback-error learning*, *distal supervised learning* and *reinforcement learning* (see SENSORIMOTOR LEARNING for details). These schemes can be applied under both supervised and unsupervised (or self-supervised) training modes and through the use of correlational, reinforcement or error-minimization adaptation procedures (Torras, 1995). This distinction between adaptation procedures is made on the basis of the type of problem information they use (Figure 1).

Correlational procedures use no problem information and their goal is to carry out feature discovery or clustering. In a robot control setting, these procedures are often used to represent a given state space in a compact and topology-preserving manner. Two applications to be described later rely on representations of this type for the robot workspace (see section "Inverse Kinematics") and the space of joint positions, velocities and accelerations (see section "Inverse Dynamics"). The correlational procedures most widely used for robot control are SELF-ORGANIZING FEATURE MAPS, CMAC and ADAPTIVE RESONANCE THEORY.

Error-minimization procedures require complete target information —in the form of input/output pairs— and their goal is to build a mapping from inputs to outputs that generalizes properly. These procedures are the most widely used in applications and, among others, the LMS rule, backpropagation (see PERCEPTRONS, ADALINES AND BACKPROPAGATION), locally weighted projection regression and conjugate gradient optimization have been applied to robot control, as described below.

Reinforcement-based procedures lie between both extremes. They make use only of a reward/penalty signal to build a mapping that maximizes reward (see REINFORCEMENT LEARNING). These procedures have been applied for learning sensorimotor maps as described later.

INVERSE KINEMATICS

The inverse kinematics mapping is that providing joint coordinates as a function of the position and orientation of the robot end-effector, thus relating the workspace to the joint space. The use of neural networks to learn this mapping is of particular interest when a precise model of some joints is lacking or when, due to the operation conditions of the robot (in space, underwater, etc.), it is hardly possible to recalibrate it.

Feedforward networks using backpropagation have been extensively tested in this context, under both the *direct inverse modelling* and the *distal supervised learning* approaches (Jordan and Rumelhart, 1992), leading to the conclusion that a coarse mapping can be obtained quickly, but an accurate representation of the true mapping is often not feasible or extremely difficult. The reason for this seems to be the *global* effect that every connection weight has on the final approximation obtained (Kröse and van der Smagt, 1993).

A way to avoid this global effect is using local representations, so that every part of the network is responsible for a small subspace of the total input space. One such representation is the 3D SELF-ORGANIZING FEATURE MAP (SOM) used by Ritter et al. (1992) to encode the robot workspace. This is combined with the LMS rule to learn the inverse kinematics of a robot arm with three degrees-of-freedom (dof) under a *direct inverse modelling* approach. The inputs to each neuron are the coordinates of the desired end-effector position, and the outputs (after correct learning) are the joint angles and the Jacobian corresponding to that position. Thus, this model provides a discrete encoding of the inverse kinematics mapping augmented with a linear approximation at each sample point that permits interpolating joint angles with higher precision. The network has been shown to self-organize into a reasonable representation of the workspace in about 30.000 learning cycles. This should be taken as an experimental demonstration of the powerful learning capabilities of this model, because the conditions in which it was made to operate are the worst possible ones: no a priori knowledge of the robot kinematics, random weight initialization, and random sampling of the workspace during training.

This basic model has been extended in three directions to cope with higher-dof robots. First, a hierarchical version, consisting of a 3D SOM whose nodes have associated a 2D SOM each, was applied to a 5-dof robot. The 3D net encodes the workspace as before, while each 2D subnet approximates the end-effector orientation space at the corresponding position (Ritter et al. 1992).

Ruiz de Angulo and Torras (1997) have adapted this hierarchical model to suit a practical setting. Thus, instead of learning the kinematics from scratch, only the deviations from the nominal kinematics embedded in the original robot controller are learnt. This, together with informed initialization and sampling, as well as several modifications in the learning algorithm aimed at improving the cooperation between neurons, lead to a speed-up of two orders of magnitude with respect to the original model. Thus, when applied to the self-calibration of a 6-dof robot installed in a space-station mock-up, 95% of the decalibration was corrected with the first 25 movements, this percentage raising to 98% after 100 movements. Moreover, other desirable features in stand-alone applications, such as parameter stability, are guaranteed.

The third extension relies on the generalization of SOMs to Parametrized SOMs (called PSOMs). The idea is to turn the discrete representation into a continuous one by associating a basis function to each neuron, so that a parametrized mapping manifold is obtained. Moreover, PSOMs make no distinction between inputs and outputs, thus encoding bidirectional mappings. The PSOM reduces considerably the number of training samples required to attain a given precision as compared to the SOM (Walter and Ritter, 1996), allowing the learning of the full inverse kinematics of a 6-dof robot with less than 800 movements.

Recently, the development of humanoid robots has raised the interest in learning inverse kinematics. Due to the many dof's involved, the aim is no longer learning the mapping for the whole workspace, but focussed on a specific trajectory. Following the trend of using localized representations, D'Souza et al. (2001) have applied a supervised algorithm –locally weighted projection regression– in this context, with promising results.

INVERSE DYNAMICS

When the robot dynamics needs to be taken into account, the control learning problem becomes more involved (see GEOMETRIC PRINCIPLES IN MOTOR CONTROL). An inverse dynamics mapping relating end-effector accelerations to the required joint forces and torques needs now to be considered.

Since the cerebellum is known to be involved in the production and learning of smooth movements, several cerebellar models have been proposed and applied to control robot arms. The pioneer such model was the Cerebellar Model Articulation Controller (CMAC) developed by Albus in 1975 (see CEREBELLUM AND MOTOR CONTROL), but today the debate is still open as to what model best captures the functionality of the cerebellum and whether any such model can constitute a practical option to control robots (van der Smagt and Bullock, 1997). A point of agreement is that the cerebellum constructs an inverse dynamics model as it learns. Thus, cerebellar models have been used for this purpose inside robot controllers.

Miller, Hewes et al. (1990) combined the table look-up facilities provided by CMAC with an error-correction scheme similar to the LMS rule to accomplish the dynamic control of a 5-dof robot. The idea underlying this combination is similar to that of enlarging SOMs with the LMS rule, as described in the preceding section. Here, CMAC is used to represent the state space in a compact and localized manner, as there SOMs were used to cover the robot workspace. To teach the robot to follow a given trajectory, successive points along it are supplied to both the neural network and a fixed-gain controller and then their responses are added up to command the robot. Therefore, the neural network acts as a feedforward component. After each cycle, the actual command given to the robot together with its current state are used as an input-output pair to train the neural network following a *direct inverse modelling* approach. As learning progresses, the CMAC network approximates the inverse dynamics mapping and, consequently, the effect of the fixed-gain controller tends to zero. The network converges to a low error (between 1 and 2 position encoder units) within 10 trials, provided enough weight vectors are used.

The same trajectory learning task above was tackled by Miyamoto, Kawato, et al. (1988) by using a *feedback-error learning* approach. They used directly as error signal the output of the feedback controller, which can be interpreted as a local linearization of the inverse dynamics mapping if the learning rate is sufficiently small. This error measure is less accurate than that used by Miller, Hewes, et al., but has the advantage of being directly available in the control loop, thus avoiding the computation of the current state of the robot required in the direct inverse modelling approach. The authors report that, after training the robot to follow a trajectory lasting 6 seconds for 300 trials, the average feedback torque decreased from a few hundreds to just a few units, demonstrating that the neural network had taken over control from the fixed-gain controller. Moreover, the mean square error in the joint angles decreased steadily 1.5 orders of magnitude.

FORCE-MOTOR MAPPING

For tasks entailing the achievement of a goal using sensory feedback, even programming in taskspace coordinates can be very complex. An example is the insertion of components with small clearance, since devising a detailed force-control strategy that performs correctly in all possible situations, and subject to real-world conditions of uncertainty and noise, is extremely difficult. What is needed to accomplish this type of tasks is an appropriate *sensorimotor mapping* relating sensory patterns to actuator commands. A relay through the intermediate workspace and joint space representations may or may not be required (see LIMB GEOMETRY: NEURAL CONTROL).

Gullapalli, Barto, and Grupen (1994) have used an associative reinforcement learning system to learn active compliant control for peg-in-hole insertion using a 6-dof robot (see REINFORCEMENT LEARNING IN MOTOR CONTROL). The system takes the sensed peg positions and forces, as well as the previous position command, as inputs, and produces a new position command as output. Thus, eighteen real values are entered into a network with two hidden layers of backpropagation units, and six real values are produced by its output layer of stochastic reinforcement-learning units. The reinforcement signal depends on the discrepancy between the sensed and the desired position of the peg, with a penalty term being activated whenever the sensed forces on the peg exceed a preset maximum. The training runs start with the peg at a random position and orientation with respect to the hole, and end when either the peg is successfully inserted or 100 time steps have elapsed. Experimental results show that, after 150 trials, the robot is consistently able to complete the insertion. Moreover, the time to insertion decreases continuously from 100 to 45 time steps over the subsequent 500 training runs.

VISUOMOTOR MAPPINGS

Depending on the task to be performed and the camera-robot arrangement, visuomotor mappings take different forms. Thus, in eye-hand coordination, where cameras external to the robot are used to monitor the pose (position and orientation) of its end-effector, a mapping from the camera coordinates of a desired end-effector pose to the joint angles that permit attaining that pose is sought. This mapping is closely related to the inverse kinematics one, especially if the camera coordinates of selected points in the end-effector uniquely characterize its pose. Therefore, the same models used to learn inverse kinematics have been applied in this context (Ritter et al. 1992).

A camera mounted on a robot arm is used in tasks such as visual positioning and object tracking. The goal of these tasks is to move the camera so that the image captured matches a given reference pattern. The target is thus no longer a position in space but a desired image pattern, and the desired visuomotor mapping needs to relate offsets w.r.t. that pattern with appropriate movements to cancel them. In visual positioning, the scene is assumed to be static and the main issue is to attain high precision. Applications include inspection and grasping of parts that cannot be precisely placed (e.g., in underwater or space settings). The aim of object tracking is to maintain a moving object within the field of view, speed being here the critical parameter instead of precision.

The classical way of tackling these tasks consists of defining a set of image features and then deriving an interaction matrix relating 2D shifts of these features in the image to 3D movements of the camera (Samson, LeBorgne, and Espiau, 1990). Note that the visuomotor mapping can be implemented with or without a relay through the workspace, depending on how the movements of the camera are commanded.

In the case of visual positioning, Wells, Venaille, and Torras (1996) have used backpropagation to learn the interaction matrix. The training procedure consists of moving the camera from the reference position to random positions and then using the displacement in image features together with the motion performed as input-output pairs. The system thus follows a *direct inverse mod-*

elling approach. In operation, the robot is commanded to execute the inverse of the motion that the network has associated to the given input. The key option in this work is the use of global image descriptors, which permits avoiding the costly matching of local geometric features in the current and reference images. By using a statistical measure of variable interdependence (the mutual information criterion), sets of global descriptors as variant as possible with each robot dof are selected from a battery of features, including geometric moments, eigenvectors, pose-image covariance vectors and local feature analysis vectors (Wells and Torras, 1998). The results obtained with a 6-dof show that, after 10.000 learning cycles, translation and rotation errors are lower than $2mm$ and 0.1 degrees, respectively.

Concerning object tracking, Schram, van der Linden, et al. (1996) have used a feedforward network together with a conjugate gradient learning algorithm to make a camera track a cart moving arbitrarily on a table. A visuomotor mapping relating the current and past visual coordinates of the cart with joint displacements is built on-line as the robot moves. Only two robot dofs need to be controlled, and thus the network has two outputs, while several numbers of inputs have been tried. The tracking performance is shown to improve as more previous positions of both the cart and the robot are used, attaining an average lag of only $8mm$ in the case of seven inputs.

DISCUSSION

After surveying several robot neurocontrol applications entailing the learning of various mappings (Table 1), we have extracted some guidelines.

In the case of *mappings that can be easily sampled*, it seems sufficient to apply a direct inverse modelling approach combined with an error-minimization learning procedure. Some simple inverse kinematics mappings and visuomotor mappings used for visual positioning have been learned in this way. If the *input space is complex*, then many researchers have resorted to a combination of correlational rules for the efficient coding of that space, with error-minimization procedures to build the appropriate association with the outputs. The use of self-organizing feature maps to encode the robot workspace or the sensor space, as well as the application of CMAC to the coding of the robot dynamics state space, fall into this category. When *the task is specified as a goal to be attained* using sensory feedback, without making explicit the movements necessary to attain it, the only possibility is to resort to reinforcement learning procedures, which depend just on the availability of a measure of success rather than an error measure.

The number of learning cycles required ranges widely in the applications described, depending on the complexity of the mapping to be learned as well as on the accuracy required. Only 10 trials are needed to get a useful mapping in the case of inverse dynamics using CMAC. The explanation is that only a very coarse mapping is needed, since the neural controller is used as a feedforward component in combination with a fixed-gain feedback controller. The number of trials raises to a few hundreds in the case of force-motor mappings for insertion of components. 100 learning cycles suffice to correct the distortions in the inverse kinematics mapping resulting from robot wear-and-tear, while this number raises to near 1.000 when the full mapping has to be learned from scratch. And the progression continues to up to 10.000 trials when the inputs are not spatial coordinates but global descriptors extracted from images. Of course, some of these figures might be considerably lowered in the future; especially the last one if more efficient codings of the input space are found.

REFERENCES

D'Souza, A., Vijayakumar, S., and Schaal, S., 2001, Learning inverse kinematics, IEEE/RSJ Conf. Intel. Robots and Systems, Maui, Hawaii, USA, pp. 298-303.

Gullapalli, V., Barto, A.G., and Grupen, R., 1994, Learning admittance mappings for force-guided assembly, in Proc. IEEE Intl. Conf. on Robotics and Automation, Los Alamitos, CA: IEEE Computer Society Press, pp. 2633-2638.

Jordan, M.I., and Rumelhart, D.E., 1992, Forward models: Supervised learning with a distal teacher, Cognitive Science, 16: 307-354.

*Kröse, B.J.A., and van der Smagt, P.P., 1993, An Introduction to Neural Networks, 5th ed., Amsterdam: University of Amsterdam, Chap. 7: Robot control.

Miller, W.T., Hewes, R.P., Glanz, F.H., and Kraft, L.G., 1990, Real-time dynamic control of an industrial manipulator using a neural-network-based learning controller, IEEE Trans. on Robot. Automat., 6(1): 1-9.

Miyamoto, H., Kawato, M., Setoyama, T., and Suzuki, R., 1988, Feedback-error-learning neural network for trajectory control of a robotic manipulator, Neural Networks, 1: 251-265.

*Ritter, H., Martinetz, T., and Schulten, K., 1992, Neural Computation and Self-Organizing Maps, New York: Addison Wesley.

Ruiz de Angulo, V., and Torras, C., 1997, Self-calibration of a space robot, IEEE Trans. on Neural Networks, 8(4): 951-963.

*Samson, C., LeBorgne, M., and Espiau, B., 1990, Robot Control: The Task Function Approach, Oxford Engineering Science Series 22, Oxford: Oxford Science Publications.

Schram, G., van der Linden, F.X., Kröse, B.J.A., and Groen, F.C.A., 1996, Visual tracking of moving objects using a neural network controller, Robotics and Autonomous Systems, 18: 293-299.

*Torras, C., 1995, Robot adaptivity, Robotics and Autonomous Systems, 15: 11-23.

van del Smagt, P. and Bullock, D., Eds., 1997, Can Artificial Cerebellar Models Compete to Control Robots?, Extended abstracts of the NIPS*97 Workshop, DLR Technical Report #515-97-28, German Aerospace Center (DLR Oberpfaffenhofen).

Walter, J., and Ritter, H., 1996, Rapid learning with parametrized self-organizing maps, Neurocomputing, 12: 131-153.

Wells, G., Venaille, C., and Torras, C., 1996, Vision-based robot positioning using neural networks, Image and Vision Computing, 14: 715-732.

Wells, G., and Torras, C., 1998, Selection of image features for robot positioning using mutual information, Proc. IEEE Conf. on Robotics and Automation, Los Alamitos, CA: IEEE Computer Society Press, pp. 2819-2826.

FIGURE AND TABLE CAPTIONS

Figure 1. Procedures for neural adaptivity. See (Torras, 1995) for a detailed explanation.

Table 1. Neuroadaptive procedures used to approximate several robot mappings.

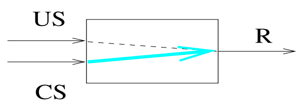
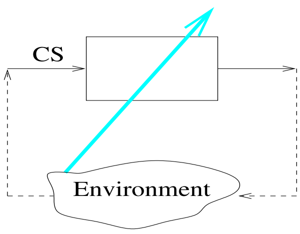
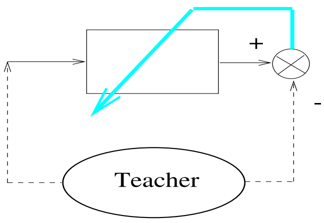
PROCEDURES	DEGREE OF FEEDBACK	DIAGRAM	KEYWORDS
CORRELATIONAL	none		Unsupervised Classical conditioning Open-loop
REINFORCEMENT	qualitative		Trial-and error Instrumental conditioning Optimal control
ERROR-MINIMIZATION	quantitative		Supervised Reference-model control Closed-loop

Figure 1:

MAPPING	CORRELATIONAL + ERROR-MINIMIZATION	ERROR-MINIMIZATION	REINFORCEMENT LEARNING
INVERSE KINEMATICS	**SOM+LMS** [Ritter et al. 92, Ruiz de Angulo & Torras 97] **PSOM+LMS** [Walter & Ritter 96]	**BP** [Jordan & Rumelhart 92, Kröse & van der Smagt 93] **LWPR** [D'Souza et al. 01]	
INVERSE DYNAMICS	**CMAC+LMS** [Miller et al. 90]	**LMS** [Miyamoto et al. 88]	
FORCE-MOTOR			**RL** [Gullapalli et al. 92]
VISUOMOTOR	**SOM+LMS** [Ritter et al. 92]	**BP** [Wells et al. 96] **CG** [Schram et al. 96]	

Table 1: