Available online at www.sciencedirect.com
SCIENCE DIRECT®
Robotics and Autonomous Systems 48 (2004) 93–110
Robotics and Autonomous Systems
www.elsevier.com/locate/robot

Adaptive neurofuzzy control of a robotic gripper with on-line machine learning

J.A. Domínguez-López, R.I. Damper*, R.M. Crowder, C.J. Harris

School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK

Received 29 October 2003; received in revised form 28 May 2004; accepted 11 June 2004

Available online 5 August 2004

Abstract

Pre-programming complex robotic systems to operate in unstructured environments is extremely difficult because of the programmer's inability to predict future operating conditions in the face of unforeseen environmental conditions, mechanical wear of parts, etc. The solution to this problem is for the robot controller to learn on-line about its own capabilities and limitations when interacting with its environment. At the present state of technology, this poses a challenge to existing machine learning methods. We study this problem using a simple two-fingered gripper which learns to grasp an object with appropriate force, without slip while minimising chances of damage to the object. Three machine learning methods are used to produce a neurofuzzy controller for the gripper. These are off-line supervised neurofuzzy learning and two on-line methods, namely unsupervised reinforcement learning and an unsupervised/supervised hybrid. With the two on-line methods, we demonstrate that the controller can learn through interaction with its environment to overcome simulated failure of its sensors. Further, the hybrid is shown to out perform reinforcement learning alone in terms of faster adaptation to the changing circumstances of sensor failure. The hybrid learning scheme allows us to make best use of such pre-labeled datasets as might exist and to remember effectively good control actions discovered by reinforcement learning.

Keywords: Robotics; Grasping; Neurofuzzy systems; Intelligent control; On-line learning

1. Introduction

In the future, advanced robotic systems will play an increasingly important role in a wide spectrum of applications, in fields as diverse as manufacturing, domestic, medical, and aerospace. One key aspect of these advanced systems is the provision of an end effector that is capable of achieving considerable gripping dexterity in unstructured environments [1]. It has been argued that without this provision, the full realisation of advanced robotic handling will not be possible. We cannot hope to program a robotic manipulator or gripper to anticipate all the real-world circumstances that it will ever encounter. A dexterous gripper needs to

* Corresponding author. Tel.: +44 1073 594577; fax: +44 1073 594498.
E-mail address: rid@ecs.soton.ac.uk (R.I. Damper).

0921-8890/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2004.06.001

display dynamic adaptation to novel and unforeseen situations, mechanical wear, component failures, changes in the environment, etc. Accordingly, it is vital that the robot controller is able to learn from its perception and experience of the environment [2], a requirement which is firmly in the domain of intelligent control.

Research in intelligent control has attracted considerable interest in recent years [3–6]. It is not a single, cohesive theory or methodology ([7], pp. 7–8), but rather a collection of complementary 'soft computing' techniques within a framework of machine learning [8]. These techniques are data-driven (i.e., they learn from example data) and aim to deal appropriately with uncertainty, imprecision and/or minor faults, non-linearities and presence or absence of prior knowledge, all of which figure prominently in our application. Several attempts have been made to combine methodologies to provide a better framework for intelligent control, of which the most successful has probably been that of neurofuzzy modeling [3,4,7], which combines neural network methods with fuzzy logic. In particular, neurofuzzy techniques offer learning capabilities with transparent knowledge representation. They have been extensively researched and developed in our laboratory. Hence, neurofuzzy control is the method of choice in this study.

In previous work [9], we showed that a neurofuzzy controller trained with the well-known back-propagation algorithm [10,11] allowed a robotic gripper to perform satisfactory grasping without slip or damage to the gripped object. Moreover, the learning gave advantages over a manually-designed fuzzy logic controller: It had a faster control action and was easier to modify and maintain. In this previous work, however, the training was supervised with the training data produced off-line. There are occasions when input-output knowledge for supervised training is hard to obtain or is not available at all. When the environment changes, a new input-output dataset must be obtained and the system retrained. Because training was off-line, the system misses the opportunity for self-retuning and reorganisation, so as to adapt automatically to environmental changes. Ideally, therefore, learning in this application should be on-line and unsupervised. This does not, however, mean that there might not be a secondary role for supervised learning, as we will show later.

Reinforcement learning [12] is the natural framework for the solution of the on-line learning problem. The reinforcement learning (RL) paradigm encompasses a broad range of techniques with the common feature that a goal-oriented system adapts its on-line behaviour in such a way as to maximise some 'reward' signal derived from its environment. Because the reward reflects the system's interaction with its environment, learning can be unsupervised, without manual intervention to indicate correct versus erroneous behaviour. Negative rewards, i.e., 'punishments', can also be conveniently incorporated into the paradigm. In view of its generality and on-line nature, RL has found wide application in robotics and autonomous system studies (e.g., [13–16]). It is proposed as the basis of the training method for the neurofuzzy controller in this paper, to overcome the shortcomings of our earlier supervised (back-propagation) training, namely its off-line nature and the difficulty of obtaining complete and consistently-labeled training data in advance.

The remainder of this paper is structured as follows. We next detail (Section 2) the hardware of the end effector that forms the focus of this work, and its control software. In Section 3, we outline the neurofuzzy control methods which are at the heart of our learning systems. Section 4 describes some previous work on supervised learning in this particular application, which we use as a baseline against which to assess our on-line learning methods, as well as forming part of the supervised/unsupervised hybrid which we have studied in this work. Section 5 introduces the basic ideas of reinforcement learning before we describe our particular realisation in Section 6. We then describe our novel hybrid in Section 7. Results obtained using the three systems (supervised learning, reinforcement learning and hybrid) are detailed in Section 8 and Section 9 concludes.

2. End effector hardware and software

The work reported in this paper was undertaken on a simple, low-cost, two-finger end effector (Fig. 1). This had just one degree of freedom; the fingers could either close or open. It was fitted with a slip sensor [17] and force sensors. The slip sensor is located on one finger and is based on a rolling contact principle. Slip induces rotation of a stainless steel roller on a spring mounting, which is sensed by an optical shaft encoder. The slip sensor has an operational range of 0 to 80 mm s^{-1} and

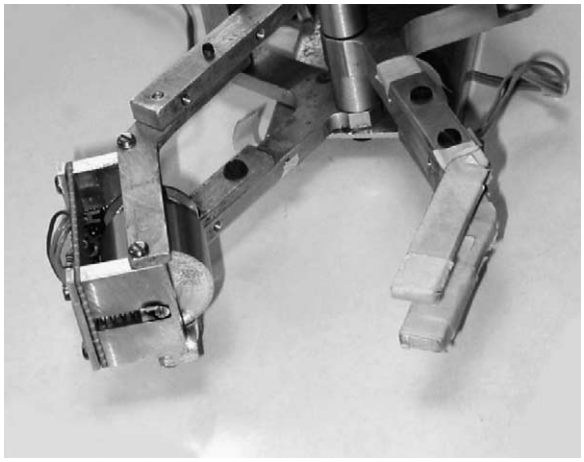

Fig. 1. The experimental, two-fingered end effector. The slip sensor appears on the left of the picture, attached to one of the fingers. The force sensors (strain gauges) are attached approximately midway along the finger shown on the right of the picture.

sensitivity of 0.5 mm s^{-1}. The applied force is measured using a strain gauge bridge on the other finger. The force sensors have a range of 0 to 2500 mN, with a resolution of 2 mN. Control of the end effector was achieved using a personal computer (Pentium 75 MHz, 64 MB RAM) fitted with a high-speed analog input/output card (Eagle Technology PC30GAS4). Control software ran under the MS-DOS operating system. The sampling period for the system was set conservatively to 17 ms to allow adequate time for all processing to be completed between consecutive samples.

Fig. 2 shows the end effector gripping a metal can. The roller of the slip sensor is clearly visible, together with its spring mounting which prevents the sensor from seizing during gripping. This metal can was found to be an effective vehicle for our experiments, as the total weight of the gripped object could be conveniently manipulated by placing various weights in it (see Section 8).

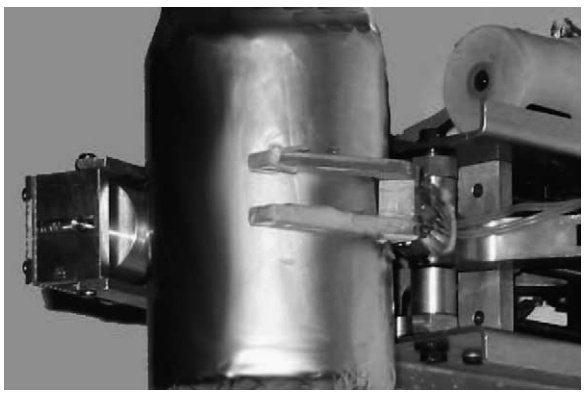

Fig. 2. The end effector gripping a metal can.

3. Neurofuzzy control

There are many factors in the control of a gripper, its specific task and the environment in which it operates that are "qualitative and fuzzy" in nature, and cannot easily be expressed in precise quantitative terms [18]. In a complex environment, it is difficult or impossible to anticipate all the conditions that will be met in operation. Hence, we are exploring the use of 'soft' controllers in preference to classical, 'hard' algorithmic control (e.g., feedforward control, model-referenced adaptive control) which require a full, quantitative specification of system dynamics, operating environment, etc. Soft methods are characterised by a focus on "learning from experimental data and ... transferring human knowledge into analytical methods" ([8], p. xi).

3.1. Advantages of neurofuzzy control

Neurofuzzy techniques have been successful because they combine the well-established modeling and learning capabilities of neural networks with the transparent knowledge representation of fuzzy systems, so satisfying both of the above complementary criteria. Neurofuzzy systems embody rules which can either be learned 'from scratch', used as a way of implanting prior knowledge, and/or improved through experience and learning. The set of rules maps to a multilayer feedforward neural network in which the connection strengths correspond to rule confidences. The system can then be trained by adapting the rule confidences which changes the strength with which a rule fires. A trained neurofuzzy model is able to describe well the input-output mapping for any arbitrary non-linear function. In many applications, a neurofuzzy system can outperform both a neural network controller and/or a fuzzy logic controller [19].

Hence, a strong feature of our work is that our neurofuzzy systems learn from the environment to improve

their performance. According to Haykin ([6], p. 50) neural network learning takes place "through an interactive process of adjustments applied to ... synaptic weights and bias levels ... The type of learning is determined by the manner in which the changes in the parameters take place". For our purposes, the two main distinctions between types of learning are:

(1) supervised versus unsupervised, and
(2) off-line versus on-line.

We believe that on-line, unsupervised learning is most appropriate for our problem. Supervised learning requires that we know the correct system action for a set of complete and representative inputs. This can be difficult to achieve in this application. We will, however, use the much commoner and simpler off-line supervised learning as a benchmark against which to assess our on-line method. Furthermore, there may be situations in which we do have labeled training data available, and we might then seek to use these data to our advantage. We will also describe a hybrid of supervised and unsupervised methods that we believe has benefits over the use of either approach alone.

3.2. Implementation of neurofuzzy network

Fig. 3 shows the architecture of the neurofuzzy network that controls the gripper. It has two inputs, the gripped object's slip rate (ideally zero) and the applied force (ideally minimal), and one output, the motor voltage. There are 5 fuzzy membership values for slip and 4 for force, giving 20 hidden units ('fuzzy neurons') each corresponding to a rule. The units used in a neurofuzzy network are, of course, different from the McCulloch-Pitts type neurons [20] used in neural networks in general. Fuzzy neurons are conceptually simple logic elements that perform fuzzy logic operations on their inputs, as opposed to the usual summing and non-linear thresholding done by the activation functions of McCulloch-Pitts neurons. There is no methodology to set the number of memberships [21]; generally the larger the number, the smoother the behavior will be but the more complex will be the controller. The values of 5 and 4 represent a good compromise between these

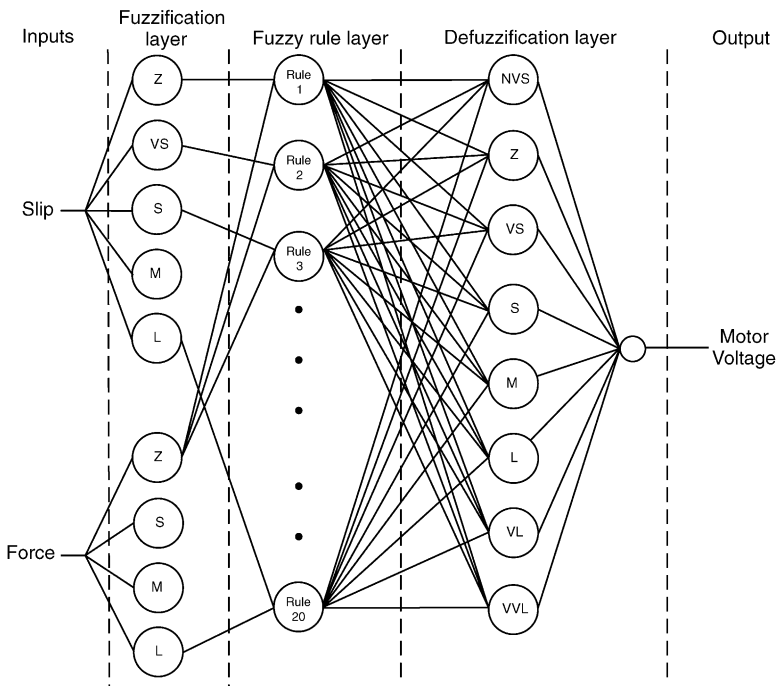

Fig. 3. Structure of the neurofuzzy network used to control the gripper. Connections between the fuzzification layer and the rule layer have fixed (unity) weight. Connections between the rule layer and the defuzzification layer have their weights adjusted during training.

competing constraints. Appropriate ranges for slip and applied force were determined experimentally. Triangular membership functions were chosen for all fuzzy variables for simplicity and economy. The well-known centre of sums (CoS) method was chosen for defuzzification for its simplicity.

The linguistic variables used for the term sets are simply value magnitude components: Zero (Z), Very Small (VS), Small (S), Medium (M) and Large (L) for the fuzzy set slip while for the applied force they are Z, S, M and L. The output fuzzy set (motor voltage) has the set members Negative Very Small (NVS), Z, Very Small (VS), S, M, L, Very Large (VL) and Very Very Large (VVL). This set has more members so as to have a smoother output.

4. Supervised learning

In general, we believe that on-line unsupervised learning is most appropriate for this work. However, to provide a benchmark for assessing unsupervised learning and also because it is a component of the hybrid system to be described later, we first consider simpler and better understood off-line supervised learning. In supervised learning, the system is instructed by a 'teacher' which has knowledge of the environment. This knowledge is represented by a set of input-output examples—the training dataset. The task of the learning algorithm is to adjust the system parameters in response to the given inputs so as to reproduce the desired output ([6] p. 63). To achieve high system performance, the training data must be consistent and complete. This is an obvious shortcoming of the supervised learning approach for our problem: In the real environment, we will not be able to anticipate all conditions which will be met, so it is not possible to guarantee completeness.

To collect training data, a simple C program was written to control the gripper manually using several keys of the computer keyboard. Six keys were designated to define coarse applied motor voltages: −5, 0, 1.5, 2.7, 3.9 and 5 V respectively. Another two keys allowed fine adjustment—increasing or decreasing the applied voltage by increments of 0.1 V. The 'teacher' (the first author) employed his judgement and observation of the current object status (e.g.. gripped satisfactorily, crushed, slipping) to increase/decrease the applied voltage via the keyboard so that the gripper grasped the object correctly. That is, the 'teacher' tried to achieve, as far as possible, fast, stable gripping with minimum finger force, and without allowing the object to fall. The program recorded the readings from the force and slip sensors as well as the applied motor voltage, each separated by the sampling period of 17 ms, to form the dataset for training. An extra *stop* key was included to allow the operator to pause the data collection and consider subsequent actions. This prevented the collection of excess training data, making the training dataset more compact.

Several trials were carried out. Each trial started with the gripper completely opened. The object used was an empty metal can (Fig. 2). During the trials, the weight of the object was modified (both within and between trials) by adding or removing weights of various value up to a maximum of approximately 40 g. There was no very strict scheme for the addition or subtraction of weights: We merely tried to cover a reasonable range of different values. Also, several disturbances of different magnitude were applied on the object to induce slip (i.e., the can was manually displaced by the 'teacher'). Each trial concluded once the object was gripped satisfactorily. From the total number of trials, a subset of 15 were selected and concatenated to form the training dataset. These 15 were chosen on the basis that the 'teacher' judged good manual control to have been achieved. After concatenation, the total dataset consisted of 8200 samples.

It should be obvious from this description that the process of collecting training data is imperfect. It is subjective and relies on the 'teacher' being able to anticipate the full variety of situations which will be met in practice. For this reason, we believe that any practical system needs to be based primarily on unsupervised learning. However, supervised training can form a useful benchmark against which to assess a system employing unsupervised training. Further, there may be occasions when a labeled dataset is available and (because learning from supervised data is a much easier problem generally leading to better performance) it makes good sense to attempt to use it. This is the rationale behind the hybrid system to be described later.

Because it is not our primary focus, we have used the commonest and best-understood method for off-line supervised training, namely error back-propagation [10,11]. Using this algorithm, the weight correction

$\Delta w_{ij}(n)$, applied to a rule weight w_{ij} between the ith rule unit and the jth unit in the defuzzification layer at iteration n, is given by the delta rule:

$$\Delta w_{ij}(n) = \eta \delta_j(n) x_j(n) + m\, \Delta w_{ij}(n-1)$$

where η is the learning rate, $\delta_j(n)$ is the local gradient, m is the momentum, and $x_j(n)$ is the input signal of neuron j in the defuzzification layer. In our previous work [9] and here, we have used a learning rate and momentum of 0.5. The stopping criterion for learning was that the average squared error per epoch reduced to less than $0.2V^2$. That is,

$$\sum_{i=1}^{N} |d_i - V_i|^2 < 0.2, \quad N = 8200$$

where d_i is the desired output voltage for the ith input (force-slip pair) according to the supervised training dataset and V_i is the actual output voltage.

In previous work [9], we have shown that back-propagation learning, used to train the neurofuzzy controller described in Fig. 3, produces reasonable results. That is, the system did indeed learn to grasp the object in the way specified by the training dataset.

5. Reinforcement learning

Reinforcement learning (RL) is a broad class of machine learning techniques in which the 'teacher' is replaced by an evaluative signal (or signals) derived from the environment. Hence, it is eminently suitable for on-line robot learning applications in which continuous interaction with the environment is all-important. Furthermore, RL is fully automatic, doing away with the need for intervention of an expert (the 'teacher') to provide a suitable training dataset. The evaluative (or reinforcement) signals do not specify what the correct answer should be, since this is unknown. Rather, they specify whether the system output is right or wrong, as well as (possibly) providing a scalar indication of the degree of correctness—see later. This evaluative signal is often couched in terms of a reward or a punishment, for being right or wrong respectively. Because there is no explicit provision of a correct answer by a 'teacher', we take RL to be distinct from supervised learning—although some researchers do treat it as supervised "because the network does, after all, get some feedback from the environment" ([22], p. 188). In many formulations of the RL problem, the system is supposed to decide the best action to select based on its current state. When this step is repeated, as occurs in continuous interaction with the environment, we have a Markov decision process (MDP) ([12], p. 66). A finite MDP is defined by its (finite) state and action sets and by the one-step dynamics of the environment.

The basic idea of the machine learning approach, and what distinguishes it from classical control, is that the detailed design of the controller is replaced by the much simpler task of specifying the desired dynamics of the gripper controller in a suitable high-level form—namely an MDP transition graph as illustrated in Fig. 4. This is where we specify what are good and bad outcomes of interaction with the environment, and how they should be rewarded or punished. For our problem, the state set is defined as $S = \{S_0, S_1, S_2, S_3\}$ = {`not_touching, slipping, crushing, OK`}, and the action set is A = {`grip, release`}. The large open circles denote state nodes and the small filled circles denote action nodes issociated with those states. Each arc is labeled with an ordered pair consisting; of the transition probability of moving from state S to state S' with associated action A, and the expected reward for that transition. Punishments are implemented as negative rewards.

The states `not_touching` and `slipping` are easily detected by the (absence of) output of the force sensor and output of the slip sensor, respectively. Crushing is considerably more difficult to detect in a truly unstructured environment, as it requires some prior knowledge of the object. Consider the difference in failure mode between an egg and a thin walled can—the former is "explosive", the latter will crumple (i.e., the gripper will close, but force is still present) after a period of increasing force with no change of gripper position. Hence, we believe at this stage that any form of crush detection will be have to be ad hoc and specialised to the particular object or objects to be handled. So here, we have constrained the study to gripping metal cans. That is, the latter description can be used. To allow the study to proceed, and recognizing that the focus of this work is intelligent control rather than crush detection, we adopted the following solution. In the absence of automatic crush detection, the first author monitored the system operation and indicated (by pressing a button) when crushing occurred.

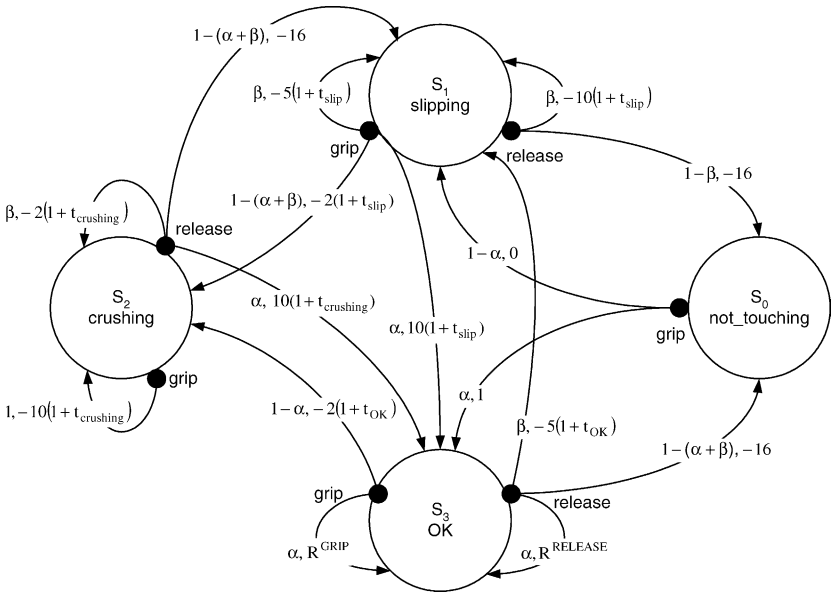

Fig. 4. Transition graph specifying the dynamics of the finite MDP. The large open circles denote state nodes and the small closed circles denote action nodes. Arcs are labeled with an ordered pair consisting of a transition probability and a reward. Punishments are implemented as negative regards. In this work, we simplify the MDP by making all transition probabilities filiations of just two parameters, α and β.

Crushing was considered to happen when the can was visibly deformed by the end effector. When crushing ceased, the button was again pressed to indicate this. Crushing was found not to occur in normal system operation, but could happen during sensor failure (see Section 8.2) or retraining after sensor failure. In future work, we will attempt a more general solution to the problem of automatic crush detection, which will dispense with the need for manual intervention.

In this work, we have used 2 transition-probability parameters α and β. Note that at least two parameters are required because the maximum outdegree of an action node is 3, and 1 degree of freedom is fixed because the transition probabilities must sum to 1. We choose to use 2 parameters (rather than 3) to ease the empirical task of finding appropriate values for them. We set α relatively high to denote a transition probability into the desired (`OK`) state, and β is set relatively low. Generally, β denotes a transition probability into the undesired states (`slipping` and/or `crushing`). There are, however, two exceptions to this generalisation set by the interdependencies between parameters due to the need for the transition probabilities to sum to 1. Thus, when the outdegree of an action node is 2, the transition probability into `slipping` and/or `crushing` states may be $(1 - \alpha)$, which therefore needs to be approximately equal to β. An example is the (undesirable) transition into the `crushing` state from the `OK` state by the grip action. When the transition is from an action node whose outdegree is 3 the transition probability into `slipping` and/or `crushing` states will be $1 - (\alpha + \beta)$, again to keep the sum of the transition probabilities 1. An example occurs for the release action in the `crushing` state leading to `slipping`. So further requirements are that $\alpha + \beta < 1$ in order that these transition probabilities do not vanish, and $1 - (\alpha + \beta) \sim \beta$ to keep transition probabilities into undesired states commensurate. Obviously, we cannot simultaneously satisfy $1 - \alpha = \beta$ and $1 - \alpha = 2\beta$, but it is not essential that these equalities are exact, only that the relevant transition probabilities are commensurate for similar outcomes. In this work, we have set $\alpha = 0.85$ and $\beta = 0.12$ by trial and error.

Rewards and punishments are either fixed, or made to depend upon the time spent in the originating state for that transition. Rewards are attached to transitions into the OK state. Punishments are attached to transitions into the undesirable (`slipping`, `crushing`

and `not_touching`) states. Time-dependent rewards have the effect of increasing the reward for moving to the `OK` state according to the length of time spent in an undesirable state, and conversely for punishments. Both time-dependent and fixed rewards/punishments have been set empirically. Transitions with a −16 punishment are included in the MDP specification for completeness; it is envisaged that they should never occur in the trained network. When the object is being satisfactorily gripped, i.e., we remain in the `OK` state for t_{OK}, the rewards for the two associated actions of `grip` and `release` are:

$$R^{GRIP} = R^{RELEASE} = k\frac{2500 - F}{2500}(1 + t_{OK})$$

where F is applied force and k is a constant empirically set equal to 20 in this work. Its purpose is to make the rewards for remaining in the OK state commensurate with those for other actions and consequences. With this scheme, the rate of increase of the reward decreases as the applied force approaches its upper limit of 2500 (see Section 2), corresponding physically to 2500 mN.

6. Actor-critic method

Actor-critic methods are temporal-difference (or TD) learning methods [23]. TD methods are based on learning the difference(s) between temporally-successive predictions. In the RL situation, we predict reward or punishment and the goal of learning is to make the current prediction (for the current input) more closely match the next prediction, at the next time step (here 17 ms). Actor-critic methods have a separate memory structure to represent the policy (i.e., the mapping between states and actions) explicitly and independent of the *value function* (see Fig. 5). The latter is the system component which estimates or derives an "internal", or TD-error. signal to drive the learning process. The policy structure is known as the *actor* because it selects actions, and the value function is known as the *critic* because it criticises the actions made by the actor. Actor-critic methods have two significant advantages for our purposes: They require minimal computation to select actions and are able to learn an explicitly stochastic policy, allowing the optimal probability of selecting appropriate actions to be learned ([12], p. 153).

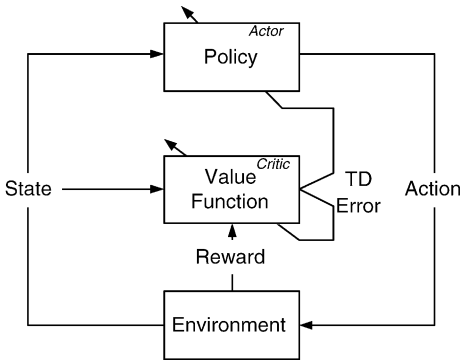

Fig. 5. Block diagram of actor-critic reinforcement learning system (after Sutton and Barto [12], Fig. 6.15).

In this work, we have used the Generalised Approximate Reasoning-based Intelligent Control (GARIC) architecture of Berenji and Khedkar [24] as the basis of our system, primarily because of the facility it gives to combine actor-critic and neurofuzzy methods. It has been widely employed in similar works on intelligent control (e.g., [25–27]). GARIC consists of two neural networks: the Action Selection Network (ASN) operating as the actor and the Action Evaluation Network (AEN) which criticises the actions made by the ASN. Outputs from these two neural networks feed into a Stochastic Action Modifier (SAM), as shown in Fig. 6.

6.1. Action Selection Network

The Action Selection Network, or actor, is a neurofuzzy controller with structure as shown in Fig. 3. The choice of a neurofuzzy network gives transparency to the system. That is. the behaviour of the physical system is described by a set of easily-interpreted linguistic rules. These can then be used to understand and validate the process under control, or to modify it.

The ASN output is $f(t)$, which determines the 'provisional' voltage to be applied to the gripper motor. By 'provisional', we mean that this value is subject to random modification by the Stochastic Action Module as described immediately below. The updating of the ASN weights (i.e., the rule confidence vector of connection weights between rule and denazification layers as in Fig. 3) is in a direction that increases the future reward, $v(t)$, predicted by the AEN. Hence, to effect gradient

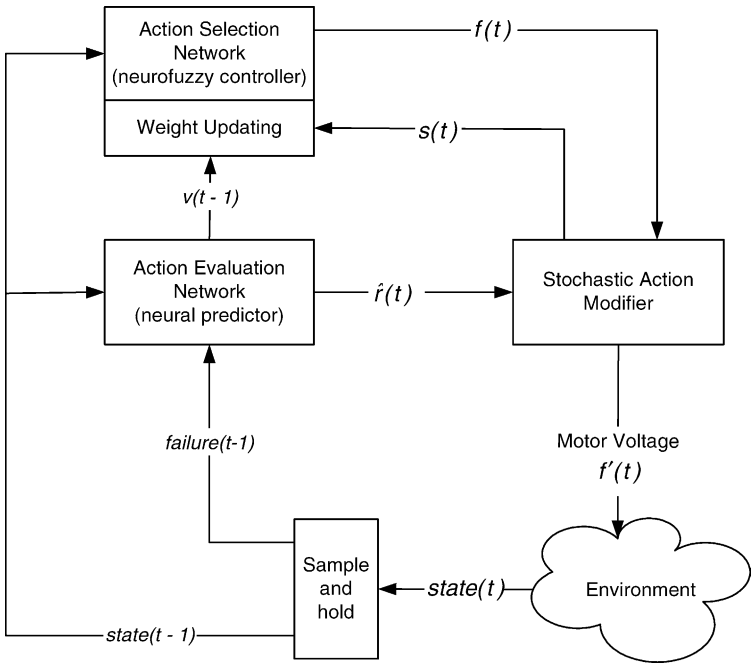

Fig. 6. Block diagram of the actor-critic system using the GARIC architecture (after Berenji and Khedkar [24], Fig. 3j).

descent optimisation, the weight update should be proportional to $\partial v(t)/\partial w_{ij}(t) > 0$, giving:

$$\Delta\omega_{ij}(n) = \eta \sum_{k=0}^{n} m^{n-k} \hat{r}(k) s(k) \frac{\partial v(k)}{\partial \omega_{ij}(n)} \quad (1)$$

where k is an index that runs from the initial time 0 up to the current time index n ([6], p. 170), $\hat{r}(t)$ is the 'internal' reinforcement signal (i.e., the output of the AEN, so called because it is internal to GARIC), $s(t)$ is an output of the SAM, and η and m are the learning and momentum constants which were empirically set to 0.75. Eq. (1) is similar to Eq. (29) of Berenji and Khedkar [24], except that they modify the antecedent and consequent membership functions whereas here we are updating the rule confidence vector.

6.2. Stochastic Action Modifier

The Stochastic Action Modifier (SAM) gives a stochastic deviation to the output of the ASN, to provide better exploration of the state space and a better generalisation ability [24,28]. The deviation, which is used as a learning factor for the ASN, is:

$$s(t) = \frac{f'(t) - f(t)}{e^{-\hat{r}(t-1)}}$$

where $f(t)$ is the output of the ASN and $f'(t)$ is a Gaussian random variable of mean $f(t)$ and standard deviation $e^{-\hat{r}(t-1)}$

6.3. Action Evaluation Network

The Action Evaluation Network (AEN), or critic, is shown in detail in Fig. 7. It is a neural predictor, producing the necessary predictions for TD-learning. Its input variables are the normalised measurements of the slip rate, the applied force to the object and the motor voltage. The AEN produces a prediction of future reinforcement, $v(t)$, which is then combined with the reward signal from the environment, $r(t)$, to produce the 'internal' reinforcement signal, $\hat{r}(t)$. The combination rules are [24]:

$$\hat{r}(t) = \begin{cases} 0, & \text{start state} \\ r(t) - v(t-1), & \text{failure state} \\ r(t) - \gamma v(t) - v(t-1), & \text{otherwise} \end{cases} \quad (2)$$

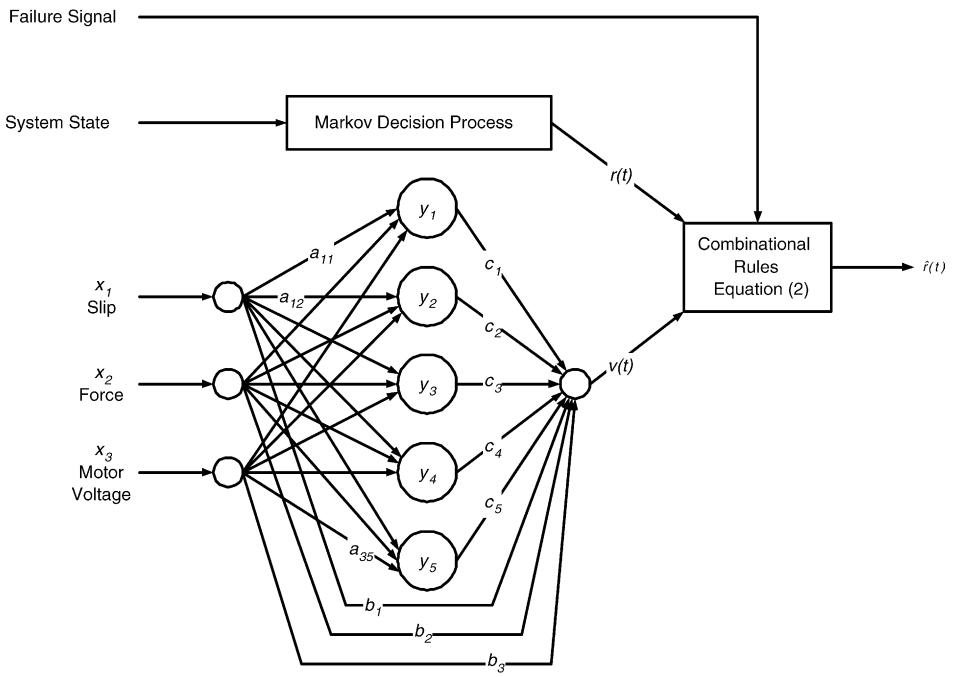

Fig. 7. The Action Evaluation Network (or critic) Li this work consists of a neural network predicting future reward, $v(t)$, which is combined with the reward signal, $r(t)$, from the environment to produce an 'internal' reward signal, $\hat{r}(t)$.

where γ, set to 0.47 in this work, is a discount rate used to control the balance between long-term and short-term consequences of the system's actions ([6], p. 606). Here, 'start' state means the state encountered on power-up, and 'failure' state means that the object has been dropped. This latter, binary state is inferred to have occurred when the force signal falls to zero from a positive value (i.e., we are not in the start start). A further failure mode exists in which the object is being deformed by excessive force, the `crushing` state. However, we are unable at present to detect this state automatically for a range of objects so that this is handled by manual intervention (see Section 5) at the present stage of the work.

This network fine-tunes the rule weight vector using a gradient descent algorithm. As the hidden layer activation function is a sigmoidal function the formulae for updating the AEN weights are [24]:

$$a_{ij}(t) = a_{ij}(t-1) + \beta_1 \hat{r}(t) y_j(t-1)(1 - y_j(t-1)) \; sgn(c_j(t-1)) x_i(t-1)$$

$$b_i(t) = b_i(t-1) + \beta_2 \hat{r}(t) x_i(t-1)$$

$$c_j(t) = c_j(t-1) + \beta_2 \hat{r}(t) y_j(t-1)$$

for $i = 1, 2, 3$ and $j = 1, \ldots, 5$. The learning rate parameters β_1 and β_2 were set empirically to 0.68 and 0.45, respectively. The signum function, *sgn*(), takes the value 1 when its argument is positive and the value 0 otherwise.

7. Hybrid learning

Thus far, our reinforcement learning scheme follows the GARIC methodology [24] quite closely. In our terms, GARIC is an unsupervised learning system. However, supervised and unsupervised learning each have their own characteristic advantages and disadvantages as detailed earlier. For instance, supervised learning generally leads to better performance because the learning task is simpler and better constrained, but it can be problematic to obtain a labeled dataset that covers all situations that will be met in practice. For instance, if there is a sensor failure, the real operating conditions will diverge from those implicit in the training data, with the consequent danger that the system becomes uncontrollable. Hence, we believe that there

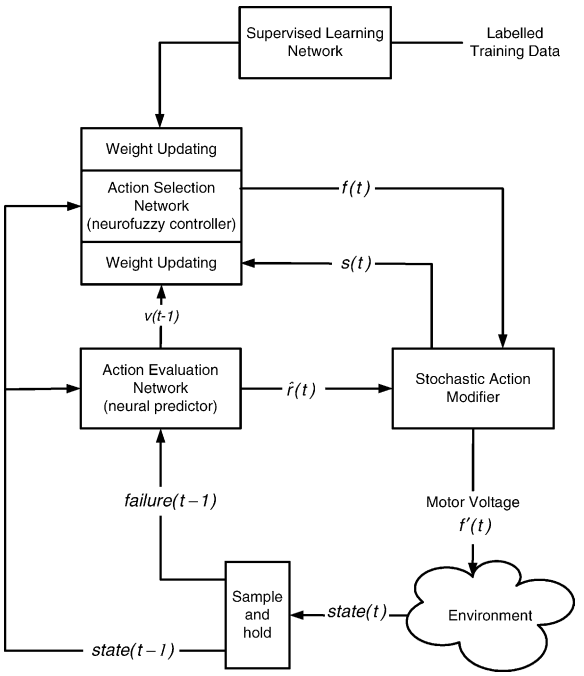

Fig. 8. Hybrid unsupervised/supervised system in which a Supervised Learning Network (SLN) is added to the basic GARIC architecture. The SLN is originally trained on pre-labeled data, but 'good' control actions discovered by reinforcement learning are added into the training dataset as learning proceeds.

are virtues to a hybrid of supervised and unsupervised learning. For our application, we see unsupervised RL as the 'core' of the system because of its advantages for on-line adaptive control, with a supervised learning module 'fine tuning' the overall operation.

The proposed hybrid system is depicted in Fig. 8, in which an extra component, the Supervised Learning Network (SLN), is added to the GARIC architecture of Fig. 6. In this scheme, the ASN always has priority over the SLN. The SLN has the form of the neurofuzzy controller in Fig. 3. It is trained by (supervised) back-propagation, initially using the dataset collected as described in Section 4. However, as and when RL discovers a 'good' action, this is added to the training set for background supervised learning. Specifically, when t_{OK} reaches 3 s, it is assumed that gripping is successful; input-output data recorded over this interval are then concatenated onto the labeled training set. In this way, we hope to ensure that such good actions do not get 'forgotten' as on-line learning proceeds.

With the datasets used here, and with experiments running for approximately 15 min, (re)training of the SLN took about 5–10 s. This was done in the 'background' (i.e., without stopping overall system execution). When such retraining is complete, the SLN then sends a weight updating signal to the ASN. Corresponding ASN and SLN weights are now compared and any ASN weight which is within 5% of the corresponding SLN weight is overwritten by the latter value. More precisely:

$$\text{if } (w_i^{\mathrm{ASN}} > 0.95 w_i^{\mathrm{ASN}}) \wedge (w_i^{\mathrm{ASN}} < 1.05 w_i^{\mathrm{ASN}}) \text{ then } w_i^{\mathrm{ASN}} \leftarrow w_i^{\mathrm{ASN}} \ \forall \ i$$

where i counts over all corresponding ASN and SLN weights.

Although the kind of 'soft' approaches to intelligent control that we have used here allow us to tackle difficult problems in robotics and manipulation, they have the major disadvantage that the lack of an underlying mathematical model means that we cannot derive stability criteria in advance. Obviously, overwriting the ASN weights in the way just described carries the clear risk of introducing instability into the hybrid system. The 5% criterion is an attempt to minimise the problem of instability while still allowing the supervised training data to have an effect. The value of 5% was set empirically, although the system was not especially sensitive to this value. For instance, a series of tests was undertaken with the criterion set to 10% and the system still maintained correct operation.

8. Experiments

Experiments were carried out with a range of weights in the metal can (Fig. 2), different from the ones used in collecting the labeled training data (see Section 4). This was intended to test the ability of neurofuzzy control to maintain correct operation robustly in the face of conditions not previously encountered.

To recap, three experimental conditions were studied:

(i) off-line supervised learning with back-propagation training;
(ii) on-line reinforcement learning;
(iii) hybrid unsupervised/supervised learning.

In each case, by virtue of using a neurofuzzy methodology, we have the possibility either to learn 'from scratch' or to seed learning with prior knowledge about good actions. An example of such prior knowledge would be the manually-written fuzzy rules described by Dubey, Crowder and Chappell [29]. Since we have previously shown [9] that rules learned from scratch by back-propagation are superior to the Dubey et al. rules, we actually proceed as follows. In (i), we learn 'from scratch' by back-propagation. This is repeated 15 times from different initial, random weight settings; it was found that there was very little difference in the solutions as far as qualitative behaviour is concerned. In (ii), reinforcement learning is seeded with the rules from (i), to see if RL can improve on back-propagation. In the hybrid (iii), RL is again seeded with the rules from (i). As stated in the previous section, however, as and when RL discovers a good action, this is added to the training set for background supervised learning, which can in turn modify the weights of the ASN (i.e., the 'actor' in the RL actor-critic framework).

8.1. Learned rules

Typical rule-base and confidences obtained after training are presented in Table 1. In the table, each rule has three confidence values corresponding to conditions (i), (ii) and (iii) above. Since reinforcement learning is on-line, the notion of a stopping criterion does not apply. In principle, learning carries on forever. To assess results, however, we decided to stop learning after approximately 15 min of training, during which time the rule confidences had stabilised.

We choose to show typical results (from 15 runs) because the precise findings depend on factors like the initial start points for the weights (rule confidences), the action of the Stochastic Action Modifier in the reinforcement and hybrid learning systems, the precise weights in the metal can, and the length of time that the system runs for. Nonetheless, there was found to be very little obvious difference in the learned solutions so some very useful generalisations can be drawn.

As stated earlier, one of the virtues of neurofuzzy methods is that the learned rules are transparent so that it should be fairly obvious to the reader what these mean

Table 1
Typical rule-base and rule confidences obtained after training

Voltage	Fingertip force			
	Z	S	M	L
Slip				
Z	L (0.0, 0.1, 0.0) VL (0.1, 0.6, 0.05) VVL (0.9, 0.3, 0.95)	S (0.05, 0.1, 0.0) M (0.1, 0.4, 0.5) L (0.8, 0.5, 0.5) VL (0.05, 0.0, 0.0)	NVS (0.2, 0.4, 0.3) Z (0.8, 0.6, 0.7)	NVS (0.9, 0.8, 0.8) Z (0.1, 0.2, 0.2)
VS	L (0.2, 0.2, 0.0) VL (0.7, 0.8, 0.6) VVL (0.1, 0.0, 0.4)	S (0.3, 0.2, 0.2) M (0.6, 0.6, 0.7) L (0.1, 0.2, 0.1)	Z (0.1, 0.2, 0.0) VS (0.9, 0.5, 0.6) S (0.0, 0.3, 0.4)	NVS (0.0, 0.2, 0.3) Z (0.75, 0.7, 0.6) VS (0.25, 0.1, 0.2)
S	M (0.2, 0.1, 0.2) L (0.8, 0.6, 0.4) VL (0.0, 0.3, 0.4)	M (0.25, 0.3, 0.2) L (0.65, 0.7, 0.7) VL (0.1, 0.0, 0.1)	S (0.4, 0.3, 0.4) M (0.6, 0.7, 0.6)	VS (0.4, 0.5, 0.4) S (0.6, 0.5, 0.6)
M	L (0.08, 0.1, 0.2) VL (0.9, 0.7, 0.4) WL (0.02, 0.2, 0.4)	L (0.2, 0.3, 0.2) VL (0.8, 0.7, 0.8)	M (0.3, 0.4, 0.2) L (0.7, 0.6, 0.6) VL (0.0, 0.0, 0.2)	S (0.3, 0.4, 0.1) M (0.7, 0.6, 0.7) L (0.0, 0.0, 0.2)
L	VL (0.1, 0.3, 0.0) VVL (0.9, 0.7, 1.0)	L (0.1, 0.2, 0.2) VL (0.9, 0.8, 0.8)	L (0.8, 0.7, 0.6) VL (0.2, 0.3, 0.4)	S (0.0, 0.1, 0.0) M (0.9, 0.8, 0.85) L (0.1, 0.1, 0.15)

Rule confidences are shown in brackets in the following order: (i) weights after off-line supervised training; (ii) weights found from on-line reinforcement learning while interacting with the environment; and (iii) weights found from hybrid unsupervised/supervised learning.

and how they relate to control of the object. For example, if the slip is large and the fingertip force is small, it means that we are in danger of dropping the object and the force must be increased rapidly by increasing the motor voltage. The two fuzzy rules inferred in this case (by reinforcement and hybrid learning) are:

$$R_{18,6}: \text{ if (slip is L)} \wedge \text{(force is S) then voltage is L } (c_{18,6} = 0.2)$$

$$R_{18,7}: \text{ if (slip is L)} \wedge \text{(force is S) then voltage is VL } (c_{18,7} = 0.8)$$

where R_{ij} is the rule label, subscript i (=18) here indicates the combination of antecedents (i.e., slip L and force S corresponds to the 18th cell of Table 1 counting left-to-right from top-to-bottom), subscript j (6 or 7) indicates that the consequent is the 6th or 7th membership (L or VL) in the output fuzzy set (motor voltage) as specified in Section 3.2, and c_{ij} denotes confidence in the rule.

The strength of each of these rules is determined by the grade of membership for slip and force in the fuzzy sets (L and S, respectively). The final motor voltage will depend on how these and other relevant rules (i.e., those for which there is also a non-zero grade of membership for slip and force) interact at the defuzzification layer (Fig. 3).

In the case that there are just two memberships to the fuzzy rule consequent, all three learning schemes give very similar results. This is to be expected in view of the restricted range of possible actions (just two). However, when there are more than two rule consequents, a more variable pattern emerges. Sometimes the hybrid learning produces a result more like the supervised learning result, sometimes it is more like the reinforcement learning result, sometimes it is intermediate between the two, and sometimes it rather unlike either. Our best interpretation of this is that the hybrid system is doing something distinctly different to the other two learning systems. The question then arises: Is the hybrid *better* in some quantifiable sense than the 'pure' reinforcement learning system alone?

8.2. Experiments with sensor failure

To answer this question, we have chosen to examine operation of the various systems under conditions of simulated (force) sensor failures. Several different sensor failures were studied—various offsets to the force sensor, to the slip sensor and to both. Results were qualitatively similar in all cases, so here we choose to describe just one case, namely when failure of the force sensor was simulated by increasing the force measure by an offset of 0.2 N. Performance was investigated by manually introducing several disturbances of various intensities acting on the object (i.e., as in Section 4).

8.2.1. Supervised learning system

The solid line of Fig. 9 shows typical performance of the supervised learning system under normal conditions; the dashed line shows operation when a sensor failure is introduced at about 5.5 s. In this and the following tests, the experimenter must attempt to reproduce the same pattern of manual disturbance inducing slip at different times so that different conditions can be compared. This is clearly not possible to do precisely. To allow easy comparison of these slightly different experimental conditions, we have aligned plots on the major induced disturbance, somewhat arbitrarily fixed at 3 s.

Although very precise comparison is not possible, a general pattern of results is clear to see. In normal operation, the figure shows that the system is able to grasp the object (metal can) properly despite the manually-induced disturbances, which can be clearly seen as transients or 'spikes' in the figure. On the other hand, following a sensor failure, the system is unable to adapt automatically and drops the object. The loss of control following a sensor failure can be clearly seen in the period after 5.5 s. The object slip increases rapidly (Fig. 9(a)) while at the same time the end effector motor voltage saturates (Fig. 9(b)) but the resultant force (Fig. 9(c)) drops to zero because the object is no longer being gripped—it has been dropped.

8.2.2. Reinforcement learning system

Fig. 10 shows typical performance of the system trained with 'pure' reinforcement learning. To simulate continuous on-line learning but in a way which allows comparison of results as training proceeds, we broke each complete RL trial into a series of 'interactions'. After each such interaction, lasting approximately 6 s, the rule-base and rule confidence vector obtained were used in the neurofuzzy system to con-

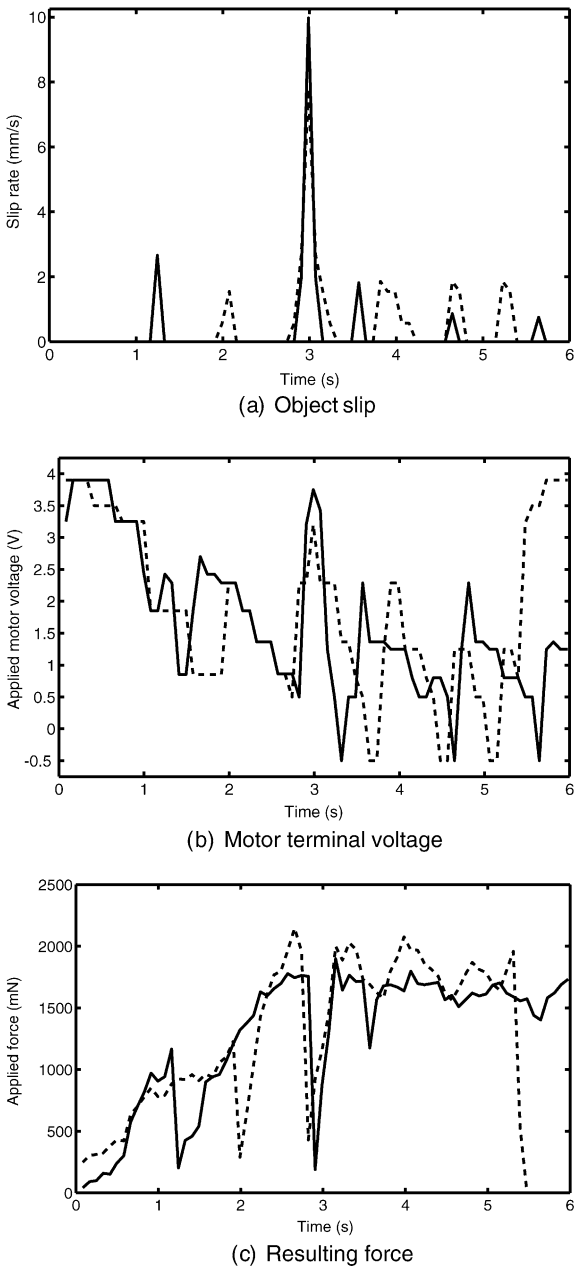

Fig. 9. Typical performance with supervised learning, under normal conditions (—) and with sensor failure at about 5.5 s (- - -): (a) shows the slip initially induced by manual displacement of the object (a metal can), (b) shows the control action (applied motor voltage) which attempts to retain appropriate grasp of the object, (c) shows the resulting force applied to the object. Note that the manually induced slip is not exactly the same in the two cases because it is not possible for the experimenter to reproduce this exactly.

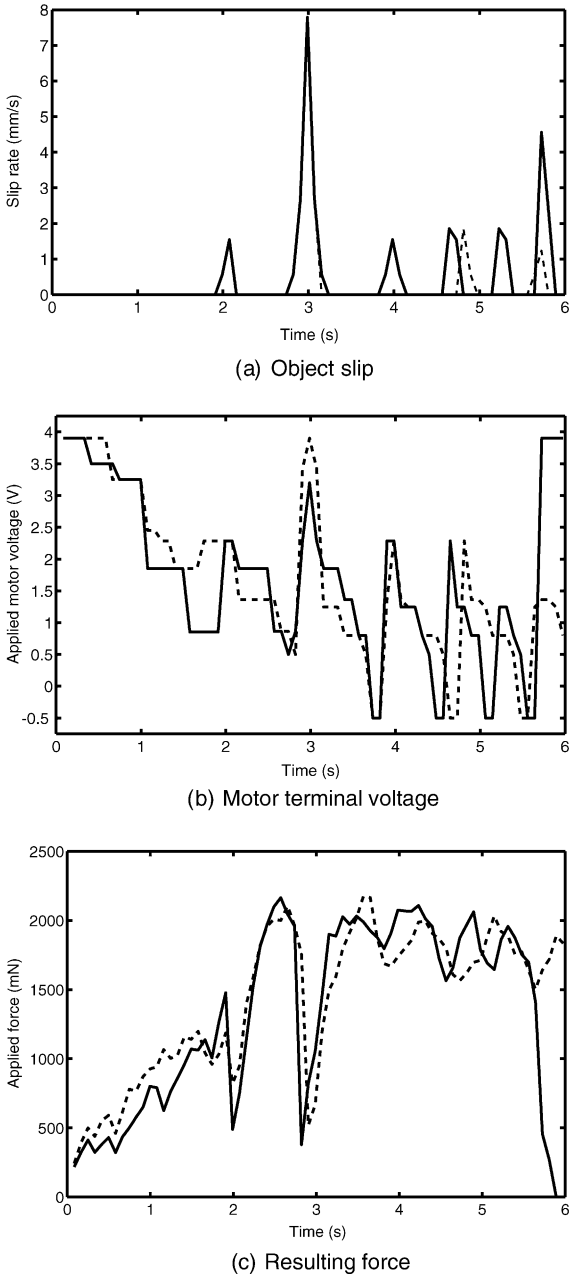

Fig. 10. Typical performance with reinforcement learning, during the first interaction (—) and the fifth interaction (- - -) after sensor failure: (a) shows the slip initially induced by manual displacement of the object, (b) shows the control action (applied motor voltage), (c) shows the resulting force applied to the object.

trol the end effector. This rule-base and rule confidence vector were then used as the start point for reinforcement learning for the next interaction. As before, during each interaction, the experimenter attempted to maintain a consistent pattern of disturbance, although this could not be done precisely. Simulated sensor failures were introduced at approximately 5.5 s during the first interaction.

In Fig. 10, the solid line shows results during the first interaction and the dashed line shows results during the fifth interaction after a sensor failure. (Note that the first interaction after a sensor failure is actually the second interaction in absolute terms.) As can be seen, during the first interaction following a failure, the object is dropped just before 6 s. There is a rapid fall off of resulting force (Fig. 10(b)) while the control action (end effector motor voltage) saturates (Fig. 10(c)). The control action is ineffective because the object is no longer present, having been dropped. By the fifth interaction after a failure, however, an appropriate control strategy has been learned. Effective force is applied to the object using a moderate motor voltage.

This result clearly demonstrates the effectiveness of on-line reinforcement learning. Even in the presence of sensor failure—simulated by adding a large offset to the force sensor reading so that the end effector is not applying as much force as it 'thinks'—the system is able to retain good grip in response to manually-induced slip after a relatively short time.

8.2.3. *Hybrid learning system*

Fig. 11 shows the performance of the hybrid trained system during the first interaction after a failure (—) and compares it with the performance of the system trained with supervised learning (- - -). Note that the latter result is identical to that shown by the full line in Fig. 9. It is clear that the hybrid system succeeds in gripping the object where the system trained on supervised learning fails. Remarkably, this success follows very quickly after experiencing the sensor failure, whereas it took some 5 or 6 interactions for the reinforcement learning system depicted in Fig. 10 to learn an appropriate control strategy in these circumstances.

As the results presented thus far are typical rather than exhaustive, it is worth asking if this apparent superiority of the hybrid system is real and consistent. To answer this question, we plot in Fig. 12 the number of control failures (either dropping the object or crushing

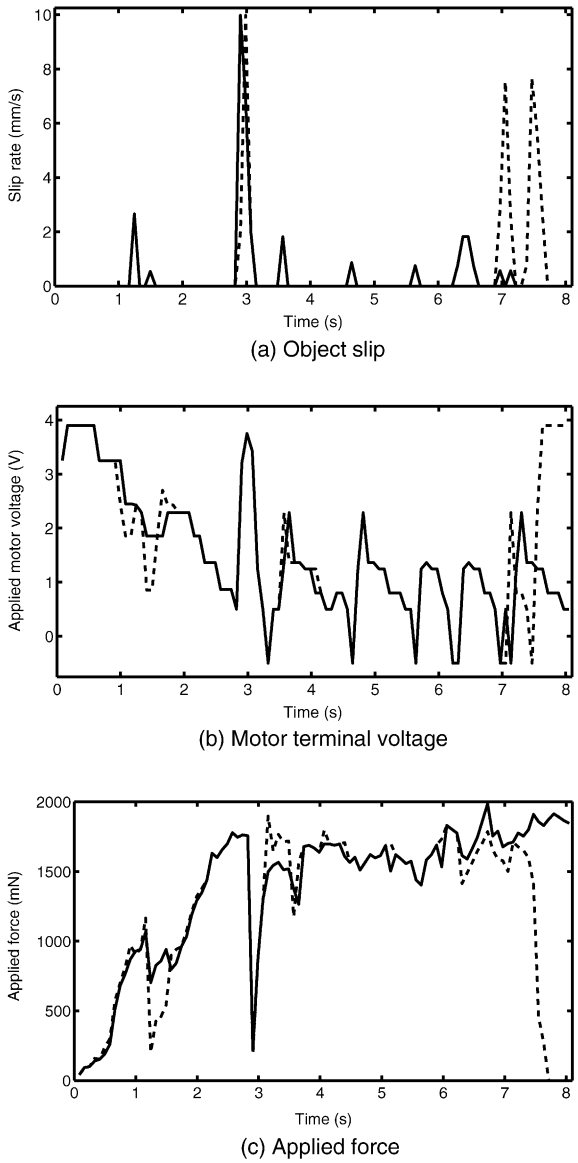

Fig. 11. Comparison of typical results of hybrid learning (—) and supervised learning (- - -) during the first interaction after a sensor failure: (a) shows the slip initially induced by manual displacement of the object, (b) shows the control action (applied motor voltage), (c) shows the resulting force applied to the object.

it) in ten repeated experiments, with both reinforcement learning (—) and hybrid learning (- - -). Both learning systems are able to reduce control failures to zero but this takes longer for the reinforcement learning system. It is not until the sixth interaction after a sensor failure

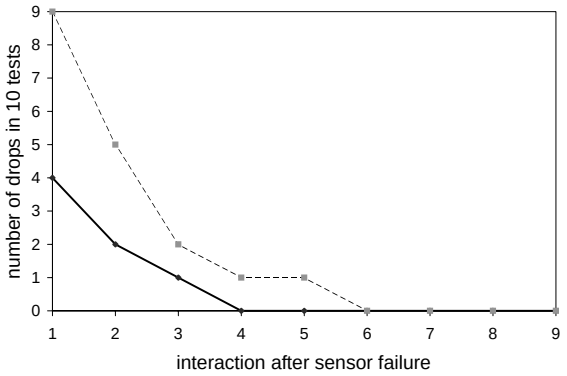

Fig. 12. Number of object drops after a simulated force sensor failure: Reinforcement learning (—); hybrid learning (- - -). There were 10 tests in all. The figure shows the number of drops following the first test.

that the RL system adapts completely to the change. The hybrid learning system manages this in fewer interactions and displays far fewer control failures in total (7 cf. 18).

9. Conclusions

Many applications of robotic end effectors require optimal object grasping, preventing object motion relative to the end effector and avoiding crushing the object by excess force. This must all be achieved in the face of possible disturbance forces acting on the object. The variety of objects and working conditions that might be met in practice make it impossible to foresee all the situations the system might encounter. Hence, some form of learning or adaptation to changing circumstances is necessary. Ideally, therefore, the system should learn on-line without a requirement for supervision, since it is difficult or impossible to collect a dataset for supervised training which anticipates all future requirements.

In certain cases, however, some amount of useful labeled training data may exist and it then makes good sense to try to use it. In this work, we have used supervised training of a neurofuzzy controller and shown that the trained system operates well provided there are no unexpected changes in the environment or the system's interface with the environment. In the case of simulated force sensor failure, however, its accuracy drops dramatically. Using reinforcement learning to train the neurofuzzy system allows on-line adaptation to such changes and so led to successful control in situations where the supervised learning system failed (dropping the object to be gripped). It is important to recognize that this is not a reflection of any inherent inferiority of the supervised learning per se but is rather a result of the difficulty of collecting complete labeled training data in an unstructured environment. This, of course, is the whole reason for using on-line reinforcement learning.

We then described a hybrid of supervised and reinforcement learning which was intended to make best use of pre-labeled training data should they exist. The supervised learning scheme ran "in the background" and its training dataset was added to if reinforcement learning discovered a 'good' control action. It was only allowed to overwrite the results of reinforcement learning if the neurofuzzy network weights were not too dissimilar. We reasoned that this would allow the hybrid system to "ignore" the labeled data if they were inconsistent with currently-encountered conditions, and also represented an attempt to maintain stability in the absence of a precise mathematical model of the system. This simple hybrid performed surprisingly well, recovering from simulated sensor failure much faster than the 'pure' reinforcement learning system. Our present work is extending these ideas to the on-line control of an end effector with more degrees of freedom and more sensors than the simple two-fingered gripper used here, to see what limits the well-known "curse of dimensionality" (frequently cited as a drawback of neurofuzzy methods) might place on practical exploitation of our findings.

Acknowledgements

J.A. Domínguez-Lòpez wishes to thank the National Council for Science and Technology of Mexico for its sponsorship.

References

[1] A. Bicchi, Hands for dextrous manipulation and robust grasping: a difficult road toward simplicity, IEEE Trans. Robot. Autom. 16 (6) (2000) 652–662.

[2] A.M. Okamura, N. Smaby, M.R. Cutkosky, An overview of dextrous manipulators, in: Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, 2000, pp. 171–180.
[3] M. Brown, C.J. Harris, Neurofuzzy Adaptive Modelling and Control, Prentice Hall, Hemel Hempstead, UK, 1994.
[4] J-S.R. Jang, C-T. Sun, E. Mizutani, Neuro-Fuzzy Modeling and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice-Hall, Englewood Cliffs, NJ, 1997.
[5] M.M. Gupta, N.K. Sinha, Soft Computing and Intelligent Systems: Theory and Applications, Elsevier, Amsterdam, The Netherlands, 1999.
[6] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice Hall, Upper Saddle River, NJ, 1999.
[7] C. Harris, X. Hong, Q. Gan, Adaptive Modelling, Estimation and Fusion from Data: A Neurofuzzy Approach, Berlin and Heidelberg, Germany, Springer-Verlag, 2002.
[8] V. Kecman, Learning and Soft Computing, MIT Press/Bradford Books, Cambridge, MA, 2001.
[9] J.A. Domínguez-López, R.I. Damper, R.M. Crowder, C.J. Harris, Optimal object grasping using fuzzy logic, in: Proceedings of the IEEE International Conference on Robotics Vision Information and Signal Processing (ROVISP2003), Penang, Malaysia, 2003, pp. 367–372.
[10] D.E. Rumelhart, G.E. Hinton, R. Williams, Learning representations by back-propagating errors, Nature 323 (9) (1986) 533–536.
[11] Y. Chauvin, D. Rumelhart (Eds.), Backpropagation: Theories, Architectures and Applications, Lawrence Erlbaum Associates, Hillsdale, NJ, 1995.
[12] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, 2000.
[13] C.J.C.H. Watkins, Learning from Delayed Rewards. Ph.D. thesis, University of Cambridge, UK, 1989.
[14] L.P. Kaelbling, Foundations of learning in autonomous systems, Robot. Autonom. Syst. 8 (1991) 131–144.
[15] R.S. Sutton, Reinforcement learning architectures for animats, in: J.-A. Meyer, S.W. Wilson (Eds.), From Animals to Animats: Proceedings of the 1st International Conference on Simulation of Adaptive Behavior, Bradford Books/MIT Press, Cambridge, MA, 1991, pp. 288–296.
[16] C.J.C.H. Watkins, P. Dayan, Q-learning, Machine Learn. 8 (1992) 279–292.
[17] R.M. Crowder, Sensors: Touch, force, and torque, in: R.L. Shell, E.L. Hall (Eds.), Handbook of Industrial Automation, New York, NY, Marcel Dekker, 2000, pp. 377–392 (Chapter 5.1).
[18] C.W. de Silva, Applications of fuzzy logic in the control of robotic manipulators, Fuzzy Sets Systems 70 (2–3) (1995) 223–234.
[19] W. Pedrycz, Computational Intelligence: An Introduction, Boca Raton, FL, CRC Press, 1998.
[20] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (1943) 115–133.
[21] D. Driankov, H. Hellendoorn, M. Reinfrank, An Introduction to Fuzzy Control, Springer-Verlag, London, UK, 1993.
[22] J. Hertz, A. Krogh, R.G. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, Reading, MA, 1991.
[23] R.S. Sutton, Learning to predict by the methods of temporal differences, Machine Learn. 3 (1) (1988) 9–44.
[24] H. Berenji, P. Khedkar, Learning and tuning fuzzy logic controllers through reinforcements, IEEE Trans. Neural Networks 3 (5) (1992) 724–740.
[25] R.M. Kandadai, J.M. Tien, A knowledge-base generating hierarchical fuzzy-neural controller, IEEE Trans. Neural Networks 8 (6) (1997) 1531–1541.
[26] A. Chatovich, S. Oktug, G. Dundar, Hierarchical neuro-fuzzy call admission controller for ATM networks, Comput. Commun. 24 (11) (2001) 1031–1044.
[27] S. Tzafestas, E. Tzafestas, Computational intelligence techniques for short-term electric load forecasting, J. Intell. Robot. Syst. 31 (1–3) (2001) 7–68.
[28] E. Bingham, Reinforcement learning in neurofuzzy traffic signal control, Eur. J. Operat. Res. 131 (2) (2001) 232–241.
[29] V.N. Dubey, R.M. Crowder, P.H. Chappell, Optimal object grasp using tactile sensors and fuzzy logic, Robotica 17 (6) (1999) 685–693.

Jorge Domínguez-López was born in Mexico City. He obtained his BEng in Electronics and Communications with honours in 2000 from the Universidad Iberoamericana, Mexico, and his PhD in intelligent control for robotics in 2004 from the University of Southampton. Jorge Axel is now working at the Mathematical Research Centre (CIMAT) in Guanajuato. Mexico, where he is continuing his research in robotics and artificial intelligence.

Robert Damper obtained his MSc in biophysics in 1973 and PhD in electrical engineering in 1979, both from the University of London. He also holds the Diploma of Imperial College, London, in electrical engineering. He was appointed Lecturer in electronics at the University of Southampton, UK, in 1980. Senior Lecturer in 1989, Reader in 1999 and Professor in 2003. His research interests include speech science and speech technology, signal and pattern processing, neural computing and mobile robotics. Prof. Damper has published more than 250 research articles and authored the undergraduate text Introduction to Discrete-Time Signals and Systems. He is past Chairman of the IEEE Signal Processing Chapter of the UK and Republic of Ireland Section (1992–1999), a Chartered Engineer and a Chartered Physicist, a Fellow of the UK Institution of Electrical Engineers and of the UK Institute of Physics.

Richard Crowder is currently Senior Lecturer in Robotics and Control in the School of Electronics and Computer Science (ECS), University of Southampton, UK. He joined the Intelligence, Agents. Multimedia Group in ECS from the Department of Electrical Engineering, to develop his interests in advanced robotics and the industrial applications of knowledge technology. He obtained his PhD in Electrical Engineering from the University of Leicester in the UK in 1977. Prior to joining the University of Southampton he worked in the machine tool industry, specialising in high-performance systems for the aerospace industry. His research interests include biologically-inspired robotics systems, with particular reference to manipulation. Richard has authored approximately 80 refereed journal and conference papers, and has supervised a number of successful PhD candidates. He is a Chartered Engineer and a Member of the UK Institution of Electrical Engineers.

Chris Harris received his BSc degree from the University of Leicester, UK in 1967, his MA degree from the University of Oxford, UK, in 1972 and PhD and DSc degrees from the University of Southampton. UK, in 1976 and 2002, respectively. He has previously held appointments at the University of Hull, UMIST, and the Universities of Oxford and Cranfield, as well as being employed by the UK Ministry of Defence. He was elected Emeritus Professor at the University of Southampton in 2002. His research interests are in intelligent and adaptive systems theory and its application to intelligent autonomous systems, management infrastructures, intelligent control and estimation of dynamic processes, and in multisensor data fusion and systems integration. He has authored or coauthored 12 books and over 350 research papers. Prof. Harris is the Associate Editor of numerous international journals. He was elected to the Royal Academy of Engineering in 1996 and was awarded the Senior Achievement Medal of the UK Institution of Electrical Engineers in 1998 for his work on autonomous systems. He received the Institutions's highest international award, the Faraday Medal, in 2001 for his work on intelligent control and neurofuzzy systems.