Perception-Action Reasoning in Space for a Robot Arm

Enric Cervera and Angel P. del Pobil

Dept. of Computer Science
Jaume-I University, Campus Penyeta Roja
E-12071 Castelló, Spain
{ecervera, pobil}@inf.uji.es

Keywords: robotics, uncertainty in space, qualitative reasoning, reinforcement learning.

Abstract

Taking robots out of the shop-floor and into service and public-oriented applications brings up several challenges concerning the implementation of real-time and robust systems. In uncertain environments sensors are required to get feedback and detect the actual world state. Perception-action reasoning seems to be the right approach for real-time and robust connections between sensing and action. In order to adapt to new situations, robots must be able to learn from given examples or from their own experience. In addition, taking advantage of the available a-priori task knowledge can speed up the learning task. The proposed sensor-based architecture combines several learning paradigms as well as pre-programmed modules, since experimental evidence suggests that some paradigms are more convenient for learning certain skills. The correspondence between qualitative states and actions is learnt. Programming is used to decrease the complexity of the learning process. A general approach is presented that is a suitable scheme for a wide range of robot situations. Results are provided for a simulated exploration task as well as for a real application of the architecture in dexterous manipulation.

1. Introduction

Taking robots out of the shop-floor and into service and public-oriented applications brings up several challenges concerning the implementation of real-time and robust systems. While modern robots are capable of performing a broad range of tasks, the presence of uncertainty in the motion or in the world model makes current hard-coded robot programs fail miserably. The robot needs to be endowed with sensors for perceiving its environment, adapting to changes, and reducing the uncertainties [1]. Additionally, the robot should be capable of adapting to new situations by learning from given examples or from its own experience. Current learning approaches are often limited to simulated problems or simple real tasks. In order to achieve complex real-world tasks, the system should not ignore the available task knowledge. The objective of this work is to build complex systems using force sensing, task knowledge and learning capabilities inspired on biological systems. The desired architecture should get benefits from both pre-programmed strategies and learning through the integration of several learning paradigms, since experimental evidence suggests that some paradigms are more convenient for certain skills.

2. The Need for Robust Perception-Based Learning

In the absence of feedback from the environment, a robot must rely only on its internal programmed model. The quality of this model is the key factor for the success of the robot operation, but precision tasks are very sensitive to small inaccuracies. For example: if the location of the elements for an insertion task is known with an error greater than the clearance between the elements, the model cannot guarantee that the task will be successfully accomplished.

In a broad sense, the problem is to enable the robot to perform tasks despite its uncertain position relative to external objects. The use of sensing significantly extends the range of possible tasks [2]. Initially, persons rely heavily on their senses, primarily vision and tactile, to accomplish a certain task. However, upon proficiency, the task becomes mechanistic and routine.

A robotic system should increase its skills in a similar way: beginning with simple exploratory actions, and upon completion of successive trials, it should incorporate the learnt knowledge to its strategy, thus becoming more skillful.

A learning approach for robot tasks is proposed in [3]. However, learning from scratch is inefficient in all but the simplest tasks. Advanced control techniques should be considered as primitive elements for such a skilled system:

for example, the compliant motion techniques [4]; or the use of vision in motion control as studied in [5].

3. An Architecture for Perception-Action

The proposed method is a framework which seamlessly integrates different learning paradigms, namely unsupervised learning, inference of Finite State Automata (FSA) through recurrent neural networks and reinforcement learning in a layered architecture, as depicted in Fig. 1.

The goal is to build an autonomous agent which is capable of learning from its own experience. At the beginning its skills are limited, and random exploratory actions must be carried out frequently. After several trials, either successful or not, the agent will be able to learn a policy (correspondence between states and actions) which tends to improve its proficiency.

3.1 Signal processing

The agent gets information from the environment through its sensors. In real applications, sensors are far from perfect: measurements are noisy, and the relevant information must be extracted prior to any other computation. Although it is claimed that neural systems are robust against noise, we argue that signal processing is a well-established discipline that cannot be ignored at this first stage. If the characteristics of the noise are known in advance, efficient filters can be easily designed.

Besides noise filtering, other signal processing operations like scaling or normalization can be performed at this stage.

3.2 Feature extraction

Usually not every component of the signal vector is relevant for the task. The problem is simplified if only those important values are considered in the next levels. However, in the general case it is difficult to state which information is essential and which other signals can be discarded. Principal Component Analysis (PCA) techniques are useful for linear spaces. In a previous work [7, 8] the authors demonstrated the feasibility of unsupervised learning algorithms for extracting relevant feature information from sensor data in robotic manipulation tasks. The dimensionality of the feature vector is considerably lowered, and the computation complexity is decreased.

3.3 Symbolic conversion

The feature vector should be converted into qualitative information before further processing. This is a requirement for the proposed learning algorithm which is suitable for discrete state and action spaces. This

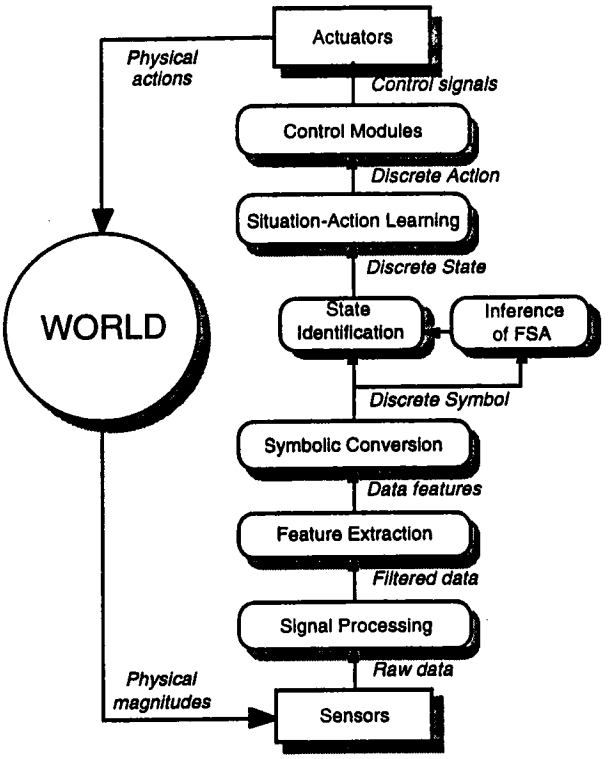

Fig. 1. Overall system architecture.

conversion can be achieved by defining sets in the feature space, and assigning labels to each set. The label of the set which contains the feature vector is chosen. If no task knowledge is available, a random partition can be chosen, or an unsupervised clustering algorithm can be applied [7, 8].

3.4 Inference of FSA through recurrent neural networks

In the simplest case, the qualitative information obtained from the feature vector could be considered as the system state. More generally, the current state is not only identified from the last observed symbol, but from the past history too. If the sequences of symbols are considered strings from an (unknown) regular language, then this behavior can be modeled with a Finite State Automaton. Although the regularity assumption seems to be too restrictive, it should be noted that any language can be approximated at a desired degree with a regular language.

The language alphabet is the finite set of discrete symbols obtained from the feature vector. Since neither the states nor transitions of the automaton are known in advance, an inference process is needed to learn the structure of the automaton from example strings. This process is carried out by means of a recurrent neural network, namely an Elman network trained with the backpropagation algorithm [9]. The FSA is extracted from

the hidden layer of the network through a procedure called dynamic state partitioning [10]. The state of the agent is determined by the state of this inferred automaton upon processing of the qualitative input.

3.5 Learning the correspondence between states and actions

The system generates an output based on its current state. This output can be programmed for a given state, or it can be learned from examples (stored state-action pairs from other executions) or from its own actual experience, by means of a reinforcement signal, which indicates the quality of the actions performed by the robot. This signal needs not be generated externally to the agent; an internal signal which keeps track of the invested effort is also appropriate. A general reinforcement algorithm, namely Q-learning [11], is proposed. This algorithm learns the state-action pairs which maximize a scalar reinforcement signal. Initially, the system randomly explores the action space. As it becomes more experienced, it learns the correct association between each state and the best action for such state.

3.6 Control modules

Only discrete actions are issued by the learning algorithm. This discretization is usually considered a major drawback of this algorithm. However, in our experimental results, it is shown that only eight different actions are sufficient for successfully performing a real insertion task. But these actions are high-level primitives, e.g. compliant motions [4]. Well-known algorithms are used for position control, force control, compliant motion, etc. A separate control law can be used for each distinct qualitative action.

4. Simulation of a sensor-based spatial exploration task

A simplified simulation of a real problem is carried out to test the learning capabilities of the proposed architecture. The task is to attain a goal guided only by sensory information. This sensor signal is related to the region where the agent is, but this relationship is unknown, and the agent's position is unknown too. The agent can move from a cell in a grid world to any of its eight neighbors with a single step. The task is simplified at the sensor and actuator levels, but the agent must learn an unknown and possibly complex relationship between states and actions. Starting from a random location, the agent must find its way to the goal (at the center of the scenario) guided only by its sensory information (see Fig. 2). The sensor measures a single discrete signal with a finite number of different values (A to F in the first scenario or A to D in the second one).

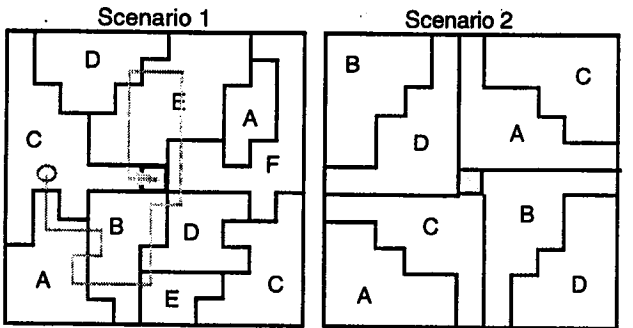

Fig 2. Sensor maps for two different scenarios.

Each location is associated to a static sensor measurement, but two distinct isolated locations may produce the same sensor value. Thus, the sensor measurement is ambiguous and the real state of the system cannot be directly obtained from the current sensor signal. Despite this ambiguity, the learning algorithm is able to find the best action (the one which leads to the goal in the minimum number of steps on average) for those states which are not replicated, in the first scenario. No best action is learnt for the rest of the states; the reason is that the agent is *confused* and cannot guess an optimal policy.

In the second scenario, configuration of the regions is even more ambiguous, and no optimal action can be learnt by using only the current symbol information. However, the agent can infer some underlying structure of both scenarios by means of randomly walking and building strings of sensor measurements. In Fig 2, such a trajectory is depicted, which produces the string CAAABBAABBE DBBDFFEEEEDEEEC and ends in the goal region.

Sets of such strings are used to infer the structure of the automata by means of an inference process, which is carried out by an Elman recurrent neural network with a hidden recurrent layer of two units. The automata are extracted through dynamic state partitioning, and reduced to its equivalent minimal state representation using a standard minimization algorithm. The resulting automata are used to identify the current state based on the current and previous sensor readings.

Results of the simulations are shown in Fig. 3. The number of steps vs. number of trials are depicted in the plots, using a moving average window of 25 consecutive values. Experiments consisted of 1500 trials, each of them beginning at a random location and finishing either when the goal is reached or the agent performs 100 steps without attaining the goal. Three independent runs are depicted in each plot. The performance of the system is compared for the two approaches, using only symbol information, and using the inferred automaton, on the two scenarios. The use of the automata leads to an improvement in the number of trials needed to converge and the final number of steps needed to attain the goal. In fact, the system that uses only symbols is unstable in scenario 2.

where $B_\varepsilon(\mathbf{x})$ is the sphere of radius ε centered at $\mathbf{x}$. One should note that the clearance (difference between hole and peg width) is much smaller than the uncertainty radius, but the hole depth is much greater than this radius. Consequently, the location of the hole cannot be detected due to uncertainties in the coordinates (x, y), but the insertion of the peg can be detected with the value of the z coordinate. The key problem is how to move towards the goal with position uncertainty. A random walk is one possible choice [6, 12], but a sensor-guided strategy is likely to be more efficient. Physical studies suggest that torque signal provides enough information about the direction to the hole [13]. However, when using real sensors, analytical solutions are quite difficult to derive. Thus, a learning scheme will be introduced at this stage.

Based on all the previous task knowledge, our proposed architecture is now applicable to this problem. Each independent module is described in the following.

5.1 Sensor signal processing

The system inputs are the six force and torque signals from the wrist sensor, and the joint angles which are measured by internal encoders.

After visual inspection of the signal spectra, we decided to use a 2nd order digital Butterworth filter with cutoff frequency 4 Hz (sampling rate is 140 Hz).

The encoder values are used to get the joint angles, and the kinematic equations allow to calculate the Cartesian coordinates of the end-effector.

5.2 Feature extraction and symbolic conversion

Task knowledge suggests that only the uncertainty along x and y axes is important in our setup. Contact information is extracted from the reaction force. Depending on the point of contact, a different torque will be sensed, but the z component of the torque vector is always small (since the normal force to the surface is nearly parallel to the z axis). This is confirmed by visual inspection of the force profiles obtained during different trials.

Despite the fact that the real task is inherently continuous, we argue that qualitative information is sufficient to achieve the goal successfully and reliably. The current state is derived from position and force sensing. Qualitative position is obtained by thresholding the current relative position with respect to the nominal goal. Two thresholds are used for X and Y (namely -1mm and +1mm), thus partitioning the space into 9 regions (negative, zero and positive for each coordinate). Qualitative torques are obtained in the same way from the torque space. Thresholds for X and Y torques are -1 and +1 Kgf-mm. Similarly to position space, torque space is divided into nine regions: positive, zero or negative for each coordinate.

Contact is detected by a single threshold in Z force, currently set to -0.1 kgf.

All threshold values are selected on an ad hoc basis, but consistently with the signal value distributions.

In our experiments we found that torque information is relevant only when position information is uncertain, namely when both X and Y values are close to zero. Thus only 17 task states from the 81 possible are used. Eight of them are based on the qualitative position and do not depend on the torque values. The remaining nine states are defined by the qualitative value of the torques *when* the qualitative location values of X and Y are zero.

5.3 State Identification and automata inference

In the current implementation, the state is defined by the current qualitative values, and no past history is considered. However, one should note that the inference process of the simulated example is equally feasible, and it will be included in future implementations. Since the state set is finite, the automaton can be discovered by a recurrent neural network. The results obtained with the simple system are encouraging, but for pegs with complex shapes it is likely than the inclusion of the automaton will improve the performance.

5.4 Action selection and control modules

The action space is discretized. For example, the exploratory motions consist of fixed steps in eight different direction of the XY-plane.

These motions are hybrid, with some degrees of freedom (XY) being position-controlled and other degrees (Z) being force-controlled, as stated in [4]. The complexity of the motion is transferred to the control modules, and the learning process is simplified.

5.5 Learning subsystem

When the peg is contacting with the surface, the qualitative features described in 5.2 are used to move the peg towards the hole. The relationship amongst states and actions is learned by means of a reinforcement algorithm, namely Q-learning [11]. A discrete number of actions is chosen, i.e. different directions of motion. Each pair (*State, Action*) is given a numerical value (zero, initially). The learning algorithm updates these values according to the reinforcement signal obtained when each action is executed in a given state. A simple action-penalty representation is used [15]. The agent is penalized with the same value for every action that it executes. The method converges to the values which minimize the accumulated penalization (which will correspond to the minimum insertion time). Initially, the agent chooses actions

where $B_\varepsilon(\mathbf{x})$ is the sphere of radius ε centered at $\mathbf{x}$. One should note that the clearance (difference between hole and peg width) is much smaller than the uncertainty radius, but the hole depth is much greater than this radius. Consequently, the location of the hole cannot be detected due to uncertainties in the coordinates (x, y), but the insertion of the peg can be detected with the value of the z coordinate. The key problem is how to move towards the goal with position uncertainty. A random walk is one possible choice [6, 12], but a sensor-guided strategy is likely to be more efficient. Physical studies suggest that torque signal provides enough information about the direction to the hole [13]. However, when using real sensors, analytical solutions are quite difficult to derive. Thus, a learning scheme will be introduced at this stage.

Based on all the previous task knowledge, our proposed architecture is now applicable to this problem. Each independent module is described in the following.

5.1 Sensor signal processing

The system inputs are the six force and torque signals from the wrist sensor, and the joint angles which are measured by internal encoders.

After visual inspection of the signal spectra, we decided to use a 2nd order digital Butterworth filter with cutoff frequency 4 Hz (sampling rate is 140 Hz).

The encoder values are used to get the joint angles, and the kinematic equations allow to calculate the Cartesian coordinates of the end-effector.

5.2 Feature extraction and symbolic conversion

Task knowledge suggests that only the uncertainty along x and y axes is important in our setup. Contact information is extracted from the reaction force. Depending on the point of contact, a different torque will be sensed, but the z component of the torque vector is always small (since the normal force to the surface is nearly parallel to the z axis). This is confirmed by visual inspection of the force profiles obtained during different trials.

Despite the fact that the real task is inherently continuous, we argue that qualitative information is sufficient to achieve the goal successfully and reliably. The current state is derived from position and force sensing. Qualitative position is obtained by thresholding the current relative position with respect to the nominal goal. Two thresholds are used for X and Y (namely -1mm and +1mm), thus partitioning the space into 9 regions (negative, zero and positive for each coordinate). Qualitative torques are obtained in the same way from the torque space. Thresholds for X and Y torques are -1 and +1 Kgf-mm. Similarly to position space, torque space is divided into nine regions: positive, zero or negative for each coordinate.

Contact is detected by a single threshold in Z force, currently set to -0.1 kgf.

All threshold values are selected on an ad hoc basis, but consistently with the signal value distributions.

In our experiments we found that torque information is relevant only when position information is uncertain, namely when both X and Y values are close to zero. Thus only 17 task states from the 81 possible are used. Eight of them are based on the qualitative position and do not depend on the torque values. The remaining nine states are defined by the qualitative value of the torques *when* the qualitative location values of X and Y are zero.

5.3 State Identification and automata inference

In the current implementation, the state is defined by the current qualitative values, and no past history is considered. However, one should note that the inference process of the simulated example is equally feasible, and it will be included in future implementations. Since the state set is finite, the automaton can be discovered by a recurrent neural network. The results obtained with the simple system are encouraging, but for pegs with complex shapes it is likely than the inclusion of the automaton will improve the performance.

5.4 Action selection and control modules

The action space is discretized. For example, the exploratory motions consist of fixed steps in eight different direction of the XY-plane.

These motions are hybrid, with some degrees of freedom (XY) being position-controlled and other degrees (Z) being force-controlled, as stated in [4]. The complexity of the motion is transferred to the control modules, and the learning process is simplified.

5.5 Learning subsystem

When the peg is contacting with the surface, the qualitative features described in 5.2 are used to move the peg towards the hole. The relationship amongst states and actions is learned by means of a reinforcement algorithm, namely Q-learning [11]. A discrete number of actions is chosen, i.e. different directions of motion. Each pair (*State*, *Action*) is given a numerical value (zero, initially). The learning algorithm updates these values according to the reinforcement signal obtained when each action is executed in a given state. A simple action-penalty representation is used [15]. The agent is penalized with the same value for every action that it executes. The method converges to the values which minimize the accumulated penalization (which will correspond to the minimum insertion time). Initially, the agent chooses actions

randomly. After learning, the action with higher value for the current state is chosen.

6. Experimental results

The system has been implemented in a Zebra Zero robot arm with a wrist-mounted force sensor. The task is the insertion of a cylindrical peg into a hole. The peg is 29 mm in diameter, while the hole is chamferless and 29.15 mm in diameter. The clearance between the peg and the hole is 0.075, thus the clearance ratio is 0.005. The peg has to be inserted to a depth of 10 mm into the hole. In order to apply our system to other shapes of the peg, we are working on methods for exporting the learned knowledge (automaton, correspondence between states and actions) from simple tasks to more complicated ones.

The controller is trained in a sequence of trials, each of which starts at a random position within the uncertainty ball (its radius is 3mm). The exploration area is a 5 mm square, centered at the real starting position.

The discrete *qualitative* values of the sensed location and forces are used to determine the state of the system (see 5.2). In this first implementation no automaton is inferred thus the recurrent network is not used, but the robot is still able to improve its skills. Further tests which include the inference process are needed to test if the knowledge of this structure leads to a further significant improvement.

A threshold in F_z is used to detect a surface contact (-0.10 Kgf). The learning algorithm uses only 17 different states. The action space is discretized and only eight directions of motion are allowed. In [16] it is suggested that qualitative reasoning may be an effective means by which humans understand the sensory force information they receive during a manipulation task.

The peg approaches the surface until a contact is detected. Next, a sequence of compliant motion steps across the surface is executed. The *xy*-direction of each step is determined by the action, which is chosen by the so-called Boltzmann exploration strategy. The probability of selecting an action a in state s is:

$$p(s,a) = \frac{e^{\frac{Q(s,a)}{T}}}{\sum_a e^{\frac{Q(s,a)}{T}}}$$

where T is a positive constant value, which controls the degree of randomness and is often referred to as *temperature*. Its value is gradually decayed from an initial fixed value. When it is close to zero, exploration is turned off and the best action is always selected.

The z-motion is determined by a damping controller. Each motion command lasts for 14 control steps (i.e. ten motions per second at a sampling rate of 140 Hz).

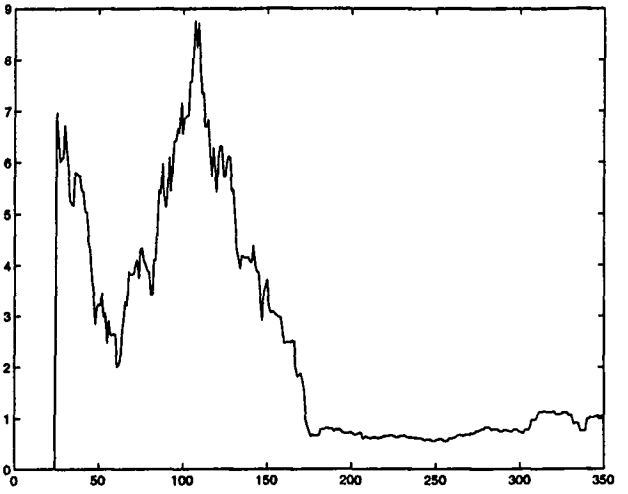

Figure 5. Smoothed exploration time taken on 350 consecutive trials of the insertion task.

After the execution of each step, the learning algorithm updates the Q-value associated with the starting state i and executed action a as follows:

$$Q_{t+1}(i,a) = (1-\alpha)Q_t(i,a) + \alpha\left(r + \gamma \cdot V_{Q_t}(j)\right)$$

where j is the successor state reached, r is the immediate cost or *reinforcement* value, α is the learning rate, and γ is a discount factor (γ=1.0 is used here). $V_{Q_t}(j)$ is the value of state j, i.e. the largest value of its actions. In the *action-penalty representation*, the system is penalized for every action it executes (r = -1 always). Since reinforcement-learning methods determine policies that maximize the total reward, the system will learn to insert the peg in the minimum possible number of steps. Each trial ends up when the hole is detected or it is aborted after a time limit (20 seconds). This detection is achieved by a threshold in F_z (loss of contact). A motion along the z-axis is performed until a contact in F_z is detected again. Despite the low clearance, no problems of jamming or wedging arose in the experiments, possibly due to the passive compliance of the gripper and the inherent noise of the process.

Experimental results are shown in figure 5. The critical phase is the surface-compliant motion towards the hole. The system must learn to find the hole based on sensory information. The exploration time of 350 consecutive trials is shown. The smoothed curve was obtained by filtering the data using a moving-average window of 25 consecutive values. The learning algorithm is executed since the beginning. Temperature is gradually decreased from the initial high value, thus turning down exploration.

After 150 trials, the exploration time is approximately constant and it has been considerably

improved over the values at the first steps. These results are better than those presented in [3] for a similar task, although the setup is different.

Experimental results show that the system learns incrementally from an initially random strategy, by improving its performance based only on its own experience. A significant amount of task knowledge is embedded in the system architecture, which simplifies the learning problem. However, the imprecision, noise and inherent difficulties of a real robot are dealt with a discrete learning algorithm. A dramatic reduction of insertion time is achieved in only two hundred trials.

7. Conclusions and future work

An architecture that incorporates a seamless integration of different learning paradigms has been introduced. Sensor processing, learning from experience and qualitative knowledge are the key elements of the system. The modular structure provides a clean integration of the different paradigms. The goal applications are those tasks which cannot be fully programmed due to uncertainties and incomplete knowledge. The proposed scheme also differs from other learning approaches in that it clearly states the difference between previous knowledge (programmed) and learned knowledge (association amongst states and actions). The qualitative treatment of information makes it suitable for the analysis of system behavior, knowledge extraction and generalization to other more complex tasks. Future research includes the study of the convergence and stability of the algorithm, the application to other non-cylindrical shapes, considering uncertainty in peg-orientation and using more degrees of freedom.

Acknowledgments

The authors wish to thank Toni Castellanos and Asun Castaño for their helpful discussions about grammatical inference and recurrent neural networks. Luis Amable helped us with the FSA software. Elman networks have been trained with the Stuttgart Neural Network Simulator. FSAs were manipulated through the FSA Utilities Toolbox by Gertjan van Noord. This paper describes research done in the Robotic Intelligence Laboratory. Support for this laboratory is provided in part by the CICYT under project TAP95-0710, by the Generalitat Valenciana under project GV-2214/94, by Fundació Caixa-Castelló under P1A94-22, and by a scholarship of the FPI Program of the Spanish Department of Education and Science.

References

[1] Gottschlich, S., C. Ramos, and D. Lyons (1994). Assembly and Task Planning: A Taxonomy. *IEEE Robotics & Automation Magazine*, 4-12.

[2] Lozano-Perez, T., M. Mason, and R. Taylor (1984). Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, **3**, 3-24.

[3] Gullapalli, V., J. Franklin, and H. Benbrahim (1994). Acquiring Robot Skills via Reinforcement Learning. *IEEE Control Systems*, **1**, 13-24.

[4] Mason, M. (1981). Compliance and force control for computer controlled manipulators. *IEEE Trans. Systems, Man and Cybernetics*, **11**, 418-432.

[5] Walter, J., and K. Schulten (1993). Implementation of self-organizing neural networks for visuo-motor control of an industrial robot. *IEEE Trans. Neural Networks*, **4**, 86-95.

[6] Hongler, M., F. Badano, M. Betemps, and A. Jutard (1995). A random exploration approach for automatic chamferless insertion. *International Journal of Robotics Research*, **14**, 161-173.

[7] Cervera, E., A. P. del Pobil, E. Marta, and M. A. Serna (1995). A Sensor-based approach for motion in contact in task planning. In: *Proc. 1995 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS'95), vol. 2*, 468-473. IEEE Computer Society Press.

[8] Cervera, E., A. P. del Pobil, E. Marta, and M. A. Serna (1996). Perception-Based Learning for Motion in Contact in Task Planning. *Journal of Intelligent and Robotic Systems*, **17**, 283-308.

[9] Elman, J. L. (1990). Finding Structure in Time. Cognitive Science, 2, 4, 279-311.

[10] Giles, C. L., Miller, C. B., Chen, D., Chen, H. H., Sun, G. Z., and Lee, Y. C. (1992). Learning and extracting finite state automata with second-order recurrent neural networks, *Neural Computation*, **4**, 393-405.

[11] Watkins, C.J.C.H. and Dayan, P. (1982). Q-Learning, *Machine Learning*, **8**, 279-292.

[12] Erdmann, M., (1992). Randomization in Robot Tasks. *The International Journal of Robotics Research*, **11**, 5, 399-436.

[13] Nevins, J., D. E. Whitney, et al. (1975). Exploratory research in industrial modular assembly. C.S. Draper Laboratory, Cambridge, Mass., R-921.

[14] Cervera, E. and del Pobil, A.P. (1997). Force-Based Robot Control for Dexterous Tasks. In: *Proc. IFAC-IFIP-IMACS Conference on Control of Industrial Systems*, Belfort, France. Edited by: L. Grujic, P. Borne and M. Fernet, Elsevier Science (in press).

[15] Koenig, S., and R. G. Simmons (1996). The Effect of Representation and Knowledge on Goal-Directed Exploration with Reinforcement Learning Algorithms. *Machine Learning*, **22**, 227-250.

[16] McCarragher, B. J. (1994). Force Sensing from Human Demonstration Using a Hybrid Dynamical Model and Qualitative Reasoning. In: *Proc. IEEE Int. Conf. on Rob. and Aut.*, 557-563.