Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems Raleigh, NC July 7-10, 1992

Learning to Avoid Obstacles through Reinforcement: Noise-tolerance, Generalization and Dynamic Capabilities

José del R. MILLÁN
Institute for System Engineering and Informatics
Commission of the European Communities. Joint Research Centre
TP 361. 21020 ISPRA (VA). ITALY
e-mail: j_millan@jrc.it

Carme TORRAS
Institut de Cibernètica (CSIC-UPC)
Diagonal, 647. 08028 BARCELONA. SPAIN
e-mail: torras@ic.upc.es

Abstract—**We have argued elsewhere that the robot path finding problem should be solved in two stages, the former involving *symbolic planning* and the latter relying on *subsymbolic obstacle-avoidance* capabilities. This paper focusses on the second stage which is characterized by (*i*) a continuous set of robot configurations and of robot actions, (*ii*) a partially unknown and dynamic environment, and (*iii*) a need of coping with unexpected events and of making real-time decisions. We present a reinforcement connectionist system able to find and learn the suitable situation-action rules so as to generate feasible paths characterized by strong performance demands for a mobile robot in a 2D environment, while appropriately dealing with the three problem characteristics above. The codification scheme adopted and the algorithm used to discover stable solution paths not only lead to very quick learning, but are also responsible for three additional positive features of the path-finder reported in this paper, namely its noise tolerance, its generalization capabilities and its ability to cope with dynamic environments.**

0-7803-0737-2/92$03.00 1992©IEEE

I. INTRODUCTION

We have argued in [1, 2] that the *robot path finding* problem should be solved in two different stages. Firstly, a *coarse-level path* is computed in *physical space* — i.e. disregarding both the shape and the kinematics of the robot. Secondly, this path is refined to obtain a *continuous, obstacle-avoiding trajectory* in *configuration space* —which, roughly speaking, is the space of degrees of freedom of motion.

Having decomposed the robot path finding problem into a two-stage process, it comes out that *planning* methods are appropriate for handling the first stage but are not suitable for dealing with the second stage. The part of the problem tackled at this second stage is characterized by (*i*) a continuous set of robot configurations and of robot actions, (*ii*) a partially unknown and dynamic environment, and (*iii*) a need of coping with unexpected events and of making real-time decisions. In [1, 2] we have explained why subsymbolic techniques are appropriate for handling this second stage of processing, and in [3] we have reviewed previous subsymbolic attempts to tackling the problem. Furthermore, some directions on how to interface a symbolic path-planner with a subsymbolic system of the kind described in this paper were provided also in [2, 3].

The work reported here represents a progress in building a subsymbolic system which deals adequately with all the above-mentioned charateristics of the problem. We present a *reinforcement connectionist system* able to find and learn the suitable *situation-action rules* so as to generate feasible paths for a mobile robot in a 2D environment. The criterion used for evaluating the quality of a path is a compromise between minimizing path length and maximizing the distance to the obstacles.

Discovering suitable situation-action rules by using only a reinforcement signal is a very general approach whose

simplest formulation could be characterized as a *weak search method.* This means that reinforcement methods have theoretically limited learning abilities; i.e. they might require heavy learning phases and they might be unable to capture complex features of the problem. These theoretical limitations can be overcome if domain-specific heuristics are incorporated into the basic reinforcement-based search method [4]. The *codification scheme* adopted in the present work and the *algorithm used to discover stable solution paths* are instances of such heuristics for the path finding domain. The latter allows to learn the necessary situation-action rules in a *very reduced time* and to deal with *continuous-valued actions.* The former, besides contributing to solving the problem in a short time, is responsible for the *generalization* skills, for satisfying the *strong performance demands* concerning path length and clearance and, partially, for the ability to cope with *dynamic environments exhibited by the path-finder.* One additional feature of the path-finder reported in this paper is its *noise tolerance.*

II. REINFORCEMENT LEARNING

It is our believe that the most suitable connectionist approach to learning situation-action rules is *reinforcement learning.* Our statement is mainly supported by the following fact. Unlike agents built through supervised learning, reinforcement-based agents can adapt autonomously to new environments since they do not require a teacher.

Simply stated, a reinforcement task is that of learning to associate with each stimulus $\mathbf{X}$ the action $\mathbf{Y}$ that maximizes reinforcement z —either present, future or cumulative. The reinforcement is a performance feedback signal, that in our case is calculated by the system itself.

The path-finder is made of two elements, namely the *step generator* and the *critic*, and interacts with its environment as depicted in Fig. 1. At each time t, the path-finder perceives the stimulus $\mathbf{X}(t)$, which is fed to both the step generator and the critic. The step generator produces instantaneously an action $\mathbf{Y}(t)$. At time $t+1$, the path-finder perceives a new stimulus $\mathbf{X}(t+1)$ and computes the *environmental reinforcement signal* $z(t+1)$, i.e. the appropriateness of the action $\mathbf{Y}(t)$ for the stimulus $\mathbf{X}(t)$. Nevertheless, the step generator does not use directly the environmental reinforcement signal for learning. Instead, it uses the *heuristic reinforcement signal* $h(t+1)$ elaborated by the critic.

The reason why the critic is needed is that the value of $z(t)$ is only meaningful when compared to past values of z associated to actions taken in response to same —or similar— situation $\mathbf{X}(t)$.

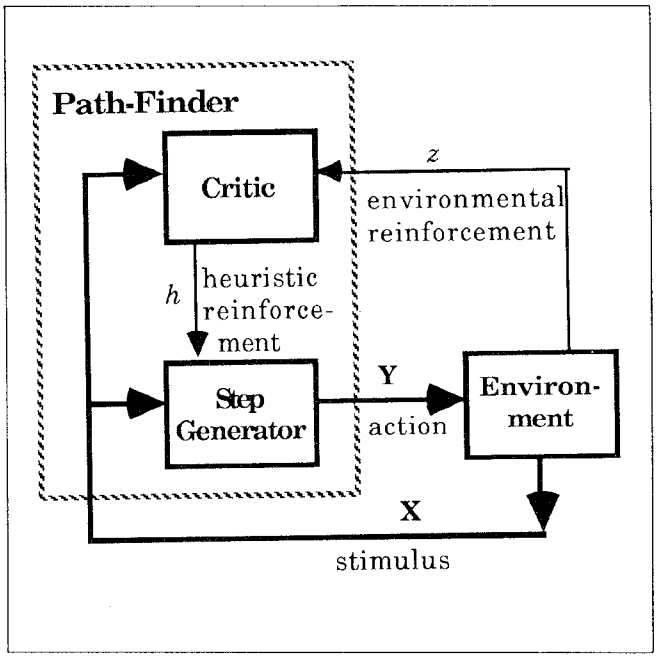

Fig. 1. A connectionist reinforcement path-finder.

III. CODIFICATION SCHEME

The *input* to the path-finder consists of an *attraction force* and several *repulsion forces* exerted simultaneously by the goal configuration and the obstacles, respectively, on the current configuration. The robot does not perceive all the obstacles at the same level, but devotes much more attention to its immediate surroundings.

Let the *shortest path line (SPL)* and the *shortest path vector (SPV)* be, respectively, the line and the vector that connect the current and the goal robot configurations. The direction of the attraction force is that of the SPV, its intensity being an inverse exponential function of the distance between the current and goal configurations.

Each repulsion force represents the resistance of an obstacle to the fact that the robot follow the SPV. Each such force has the direction of a bisector of the SPL and its perpendicular, starting at the current configuration and heading the opposite quadrant to that where the obstacle lies. Because the directions of the repulsion forces are specified with respect to the direction of the attraction force, they are implicit in the codification and therefore are not included in the input to the system. The intensity of each repulsion force depends on three factors. The first factor is aimed at avoiding obstacles in the proximity of the SPV. The second allows to avoid obstacles near to the robot current configuration. The third ensures that, in the case that the SPV intersects an obstacle, the next

robot movement is the more distant from the SPV, the deeper is the penetration into the obstacle.

The SPL and its perpendicular divide the workspace into four quadrants. Repulsion forces in a quadrant try to deflect the robot move toward the center of the opposite quadrant. Thus all the information related to the repulsion forces can be reduced to four signals: *intensity of the environmental repulsion from each quadrant.*

In short, *the number of input signals* to the path-finder *is independent of the environment*, and these signals are five: the intensity of the attraction force and the four environmental repulsion intensities. Since the output of the path-finder is computed with respect to the direction of the attraction force (see below), this direction needs not be included in the input to the system. Each input signal is codified as a real number in $[0, 1]$.

The *output* of the path-finder represents the step taken by the robot and it is codified as a move in *relative cartesian coordinates* with regard to the SPV. That is, the axis X is the SPL. Both coordinates take real values in $[-1, 1]$. The maximum distance the robot can cover in a single step is limited by the *perception range.* Otherwise, the robot could collide with obstacles "not completely" perceived since the robot concentrates on its immediate neighbourhood.

Two important consequences of the manner in which the output signals are codified and postprocessed are that *the space of attainable configurations is continuous* and that *the system itself decides about the direction and length of each step,* with the only constraint imposed by the perception range.

The *reinforcement signal* is a measure of how good is the answer of the system to a particular stimulus. It is calculated on the basis of the quality of the configuration reached by the robot —a combination of the attraction and repulsion factors— and the way in which this configuration has been reached. It is a real value in $[-1, 1]$.

The determination of both the input and goodness degree is reminiscent of the *potential field approach* to path planning [5].

Three are the benefits of this codification scheme. First, since the goal is codified in the input information and the input is independent of the environment used during the learning phase, it allows to transfer the knowledge acquired for a situation to a different one.

Second, the output postprocessing offers the robot the possibility of coping with dynamic environments. Because the robot does not move beyond its perception range, the step it takes is aimed at avoiding neighbouring obstacles. So, the probability of colliding with a mobile obstacle is low.

Finally, it allows to reduce the complexity of the task to be solved. In the robot path finding problem, the consequences of an action can emerge later in time. Thus, actions must be selected based on both their short- and long-term consequences. Since the environmental reinforcement signal is computed using *global information* —i.e. it is based not on the current robot configuration but on the SPV—, the path-finder gets a measure of the short- and long-term consequences of an action only one time-step after it has been executed. Thus the task is reduced to learn, for each stimulus, to perform the action which maximizes the environmental reinforcement signal.

IV. LEARNING ALGORITHM

Of the two kinds of reinforcement learning algorithms proposed in the literature, namely *associative reward-penalty*, A_{R-P}, [6] and *associative search*, *AS*, [7, 8, 9], we adopt the second one because it fits best the specifications of the input, output and reinforcement signals we have.

A central issue for any reinforcement system is to explore alternative actions for the same stimulus. *Stochastic units* provide this source of variation. But the stochastic behavior of the path-finder should tend to be *deterministic* with learning. Otherwise, the path-finder could not generate *stable solution paths* after it eventually discovers them. By using a suitable algorithm for transforming the stochastic units into deterministic ones [10] the path-finder not only can discover the suitable situation-action rules, but also does this in a very short time.

A. The Basic AS Algorithm

Continuous stochastic units compute their output in three steps [11], as expressed in Equations (1) through (5). The only difference with respect to Gullapalli's formulation is the way in which the variance is computed.

Since the signals we are interested in are continuous, a separate control of the location being sought (*mean*) and the breadth of the search around that location (*variance*) is needed. The first step is to determine the value of these parameters. The mean μ should be an estimation of the optimal output. A simple way is to let $\mu_i(t)$ equal a weighted sum of the inputs $s_j(t)$ to the unit i plus a threshold $\theta_i(t)$:

$$\mu_i(t) = \sum_{j=1}^{n} \left(w_{ij}(t) s_j(t)\right) + \theta_i(t). \qquad (1)$$

The standard deviation σ should be small if the expected output of the step generator is close to the optimal, and it should be high in the opposite case. Since the heuristic reinforcement signal provides this comparative measure of the output goodness, $\sigma(t)$ should depend on the *expected heuristic reinforcement*, $\widehat{h}(t)$:

$$\sigma(t) = k_\sigma * \widehat{h}(t), \qquad (2)$$

where k_σ is a constant and $\widehat{h}(t)$ is a trace of the absolute value of past heuristic reinforcement received:

$$\widehat{h}(t) = \xi * \mathrm{abs}\big(h(t)\big) + [1-\xi]\widehat{h}(t-1), \tag{3}$$

with ξ being a constant in $[0,1]$.

As a second step, the unit calculates its *activation level* $a_i(t)$ which is a *normally distributed random variable*:

$$a_i(t) = N\big(\mu_i(t), \sigma(t)\big). \tag{4}$$

Finally, the unit computes its output $s_i(t)$:

$$s_i(t) = f\big(a_i(t)\big) = \frac{2}{1+e^{-\beta a_i(t)}} - 1, \tag{5}$$

where β is a constant in $[0,1]$.

A deterministic unit computes its output as a weighted sum of its input:

$$s_i(t) = f\left(\sum_{j=1}^{n}\big(w_{ij}(t)s_j(t)\big) + \theta_i(t)\right), \tag{6}$$

where $f(\cdot)$ is the same function as in Equation (5).

In the AS family of algorithms, the weights are modified according to the following general expression:

$$\Delta w_{ij}(t) = \alpha h(t) e_{ij}(t-1), \tag{7}$$

where α is the *learning rate* and e_{ij} is the *eligibility factor* of w_{ij}. The eligibility factor of a given weight is a measure of how influential that weight was in choosing the action. Each particular version of this general algorithm differs from the others in the way h and e_{ij} are calculated.

In the experiments reported here, we use the version of the AS algorithm that best suits the robot path finding problem. This version has been identified after a comparative study on twenty five versions of the basic AS algorithm [12]:

$$h(t) = z(t) - \widehat{z}(t-1), \tag{8}$$

$$e_{ij}(t) = \frac{\partial ln N}{\partial w_{ij}}(t) = s_j(t)\frac{a_i(t)-\mu_i(t)}{\sigma^2(t)}, \tag{9a}$$

$$e_{ij}(t) = s_i(t) * s_j(t), \tag{9b}$$

where $\widehat{z}$ is the *expected primary reinforcement* calculated by the critic. In order to undertake this prediction task, the critic is built as a second network out of deterministic units, and since it is provided with input/output pairs, a *supervised algorithm* is used. e_{ij} is calculated in two different manners according to the kind of units of the step generator. If the units are stochastic, it is computed in such a manner that the learning rule corresponds to a *gradient ascent* mechanism on the expected environmental reinforcement [9]. If the units are deterministic, it is the *Hebbian rule*.

B. Discovering Stable Solutions

It has been stated above that in order to obtain stable solution paths, stochastic units should become deterministic as learning proceeds. The way in which σ is computed guarantees that the breadth of search will diminish asymptotically to zero as the path-finder learns. But this is not acceptable to solve the problem efficiently.

A mechanism for accelerating the generation of stable solution paths is the following. When the path-finder discovers, after a certain experience with the environment, *acceptable* paths, it might be more suitable to search the solution paths in a more "controlled" manner by *transforming the stochastic units of the step generator into deterministic units.* Nevertheless, as explained in [12], the process for discovering a stable solutions requires further refinements.

V. SYSTEM PERFORMANCE

As it is usual in the reinforcement-learning paradigm, the path-finder consists of a step-generator and a critic. It has been implemented in Common Lisp on a VAX station 2000. The *step-generator* we have used to carry out the simulations has three layers of units, all the units in a layer being connected to all the units in all the layers above. The hidden layer is made of four units. As usual, the functionality of an input layer consists simply in forwarding the signal it receives. The hidden and output units of the step-generator are stochastic units that become deterministic as learning proceeds. The step-generator is built in two phases [12]. The *critic* has a hidden layer of four deterministic units and, obviously, one output deterministic unit.

Figs. 2 and 3 show the behavior of the path-finder once the learning phase is finished. Fig. 2 depicts the paths generated by the path-finder from every starting configuration in the training set. Obstacles are shown as circles and every initial configuration and the goal are represented by a triangle and a square, respectively. In Fig. 3, instances of the situation-action rules discovered by the path-finder are illustrated; for every starting configuration considered —little circles— the move to be taken by the robot is shown. The path-finder is able to produce stable collision-free paths from almost all the initial configurations in the training set. The number of steps required to reach this state of the path-finder has been 77315. The only situations not properly handled by the path-finder are a subset of the most difficult ones, that is those in which an obstacle lies in between the

goal and the current robot configuration and is very close to this configuration. These situations may be handled by getting appropriate guidance from a symbolic path-planner [2].

Until now, we have assumed that the robot can perceive the workspace perfectly. Nevertheless, a robot navigating in a real environment is subject to noisy and inaccurate measurements. To test the ability of our path-finder to work under real conditions, a 20% of white noise is added to the sensory input. Fig. 4 depicts the resulting behavior. Comparing this figure with Fig. 3 it is evident that the path-finder exhibits a *large noise tolerance*, since both "maps" are very similar.

Fig. 3 illustrates some of the generalization abilities of the path-finder. It tackles many more situations than those perceived during the learning phase. In addition, the path-finder is also able to navigate in workspaces different from that used during learning. Figs. 5 through 7 show the behavior of the path-finder when only the goal is changed, more obstacles are added to the original workspace, and both the goal and the number and location of obstacles have changed. The results of these three experiments show that the *situation-action rules learned are workspace-independent.*

Finally, Figs. 8 and 9 show how the *path-finder copes with dynamic environments.* If the robot has taken one or more steps toward the goal and either the goal (Fig. 8) or the obstacles (Fig. 9) move, the path-finder is still able to generate feasible paths. In the first case, the goal is moving toward the northeast and the path-finder tracks it. In the second case, the obstacles are moving toward the northwest, therefore approaching the goal, and the path-finder avoids them. This ability could be enhanced if a more powerful perception module were incorporated into the system. This module should predict where the goal and the obstacles would be placed at each time step.

ACKNOWLEDGMENT

Carme Torras acknowledges partial support from the ESPRIT Basic Research Action number 3234.

REFERENCES

[1] C. Torras, "Motion planning and control: Symbolic and neural levels of computation," *Proc. 3rd COGNITIVA Conference*, pp. 207–218, 1990.

[2] J. del R. Millán and C. Torras, "Learning to avoid obstacles through reinforcement," in *Machine Learning: Proceedings of the Eighth International Workshop*, L. Birnbaum and G. Collins, Eds. San Mateo, CA: Morgan Kaufmann, 1991, pp. 298–302.

[3] J. del R. Millán and C. Torras, "Connectionist approaches to robot path finding," in *Progress in Neural Networks Series,* vol. 3. O.M. Omidvar, Ed. Norwood, NJ: Ablex, in press.

[4] P. Langley, "Learning to search: From weak methods to domain-specific heuristics," *Cognitive Science*, vol. 9, pp. 217–260, 1985.

[5] O. Khatib, "Real time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, pp. 90–98, 1986.

[6] A. G. Barto and P. Anandan, "Pattern-recognizing stochastic learning automata," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, pp. 360–374, 1985.

[7] A. G. Barto, R. S. Sutton, R.S., and P. S. Brouwer, "Associative search network: A reinforcement learning associative memory," *Biological Cybernetics*, vol. 40, pp. 201–211, 1981.

[8] R. S. Sutton, "Temporal credit assignment in reinforcement learning," Ph.D. Thesis, Dept. of Computer and Information Science, University of Massachusetts, Amherst, 1984.

[9] R. J. Williams, "Reinforcement-learning connectionist systems," Technical Report NU-CCS-87-3, College of Computer Science, Northeastern University, Boston, 1987.

[10] J. del R. Millán and C. Torras, "Reinforcement learning: Discovering stable solutions in the robot path finding domain," *Proc. 9th European Conference on Artificial Intelligence*, pp. 219–221, 1990.

[11] V. Gullapalli, "A stochastic algorithm for learning real-valued functions via reinforcement feedback," Technical Report COINS-88-91, Dept. of Computer and Information Science, University of Massachusetts, Amherst, 1988.

[12] J. del R. Millán and C. Torras, "A reinforcement connectionist approach to robot path finding in non-maze-like environments," *Machine Learning*, in press.

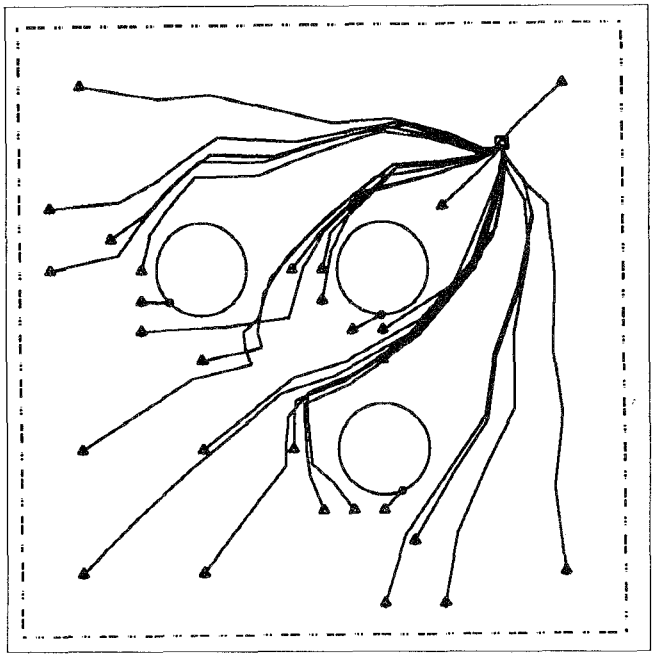

Fig. 2. Behavior of the path-finder: Trajectories from every starting configuration.

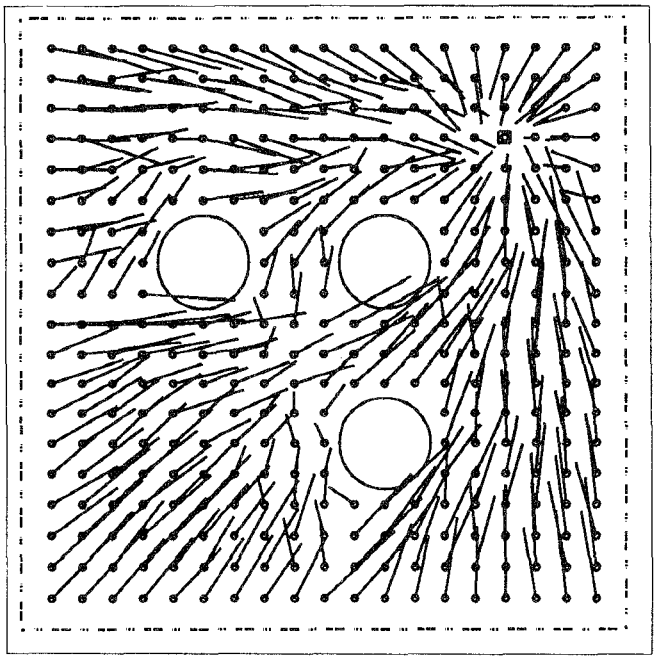

Fig. 4. Noise tolerance exhibited by the path-finder.

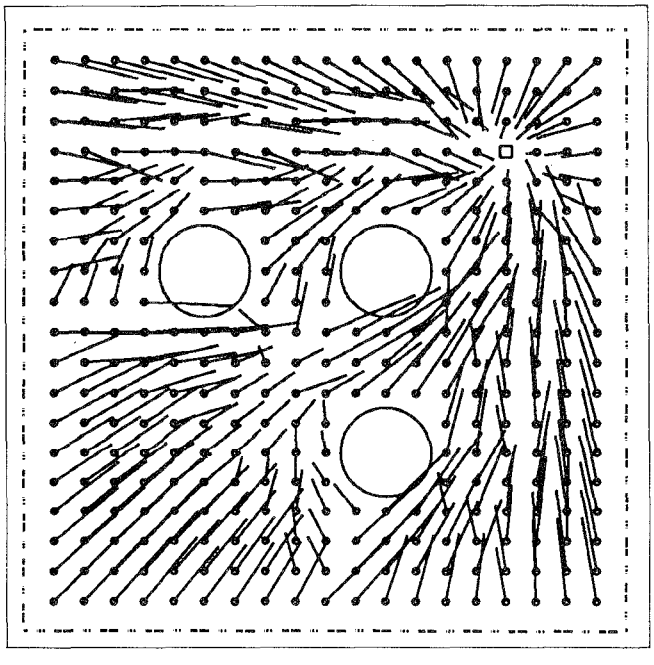

Fig. 3. Behavior of the path-finder: Instances of situation-action rules.

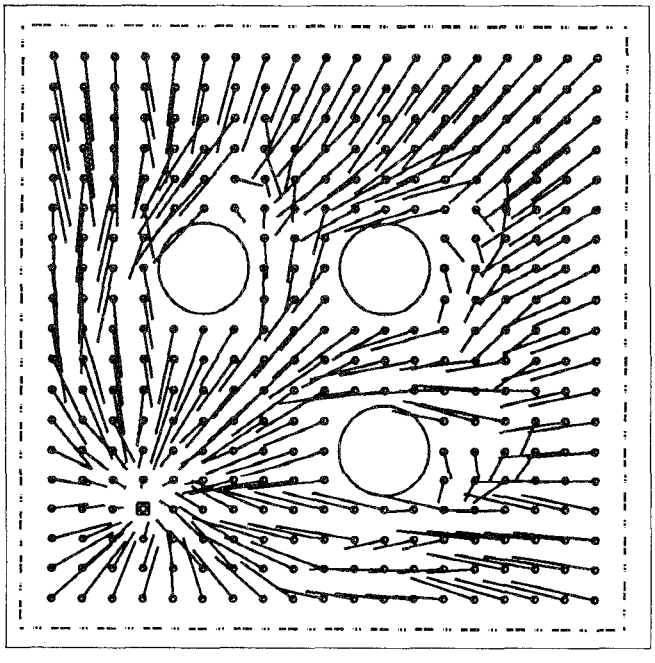

Fig. 5. Generalization abilities: Experiments with a new goal.

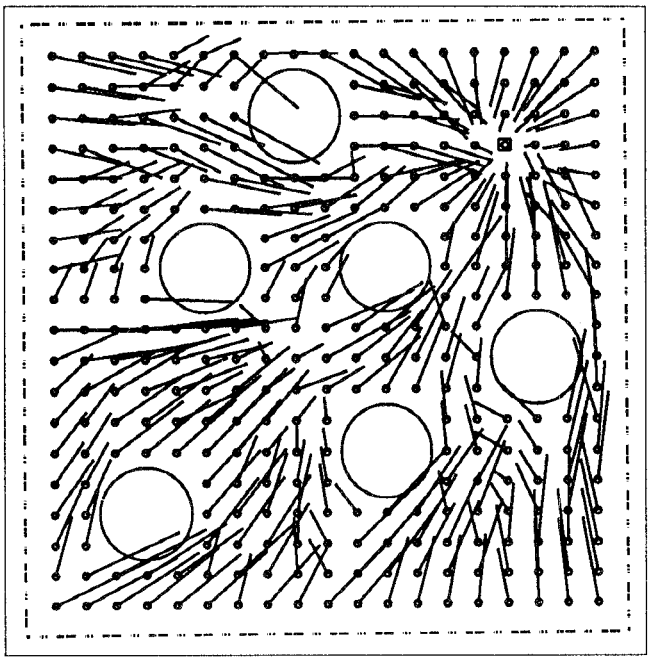

Fig. 6. Generalization abilities: Experiments with more obstacles.

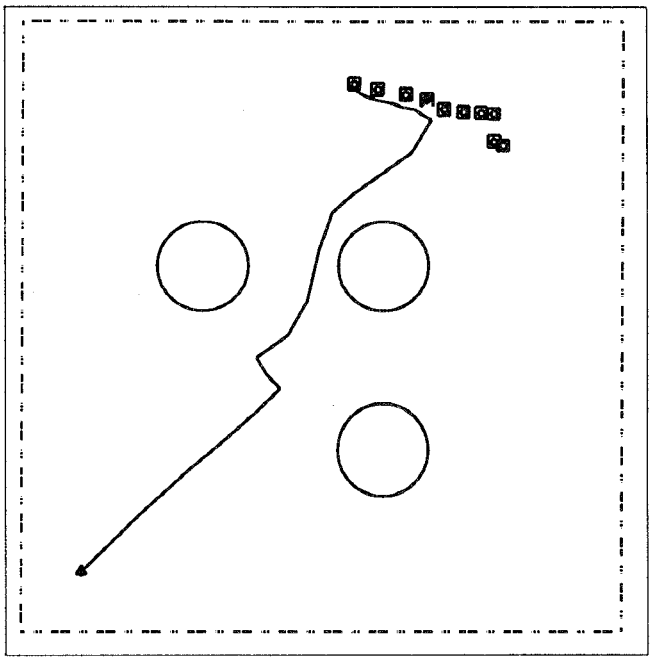

Fig. 8. Coping with a dynamic goal.

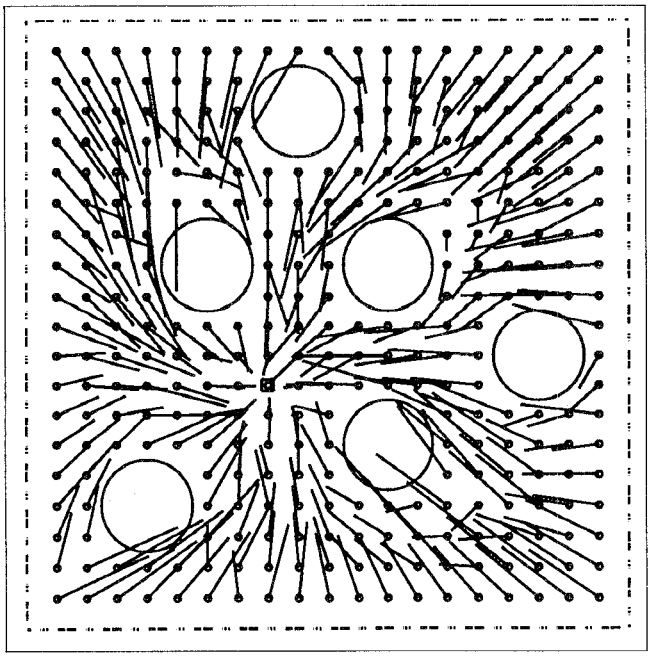

Fig. 7. Generalization abilities: Experiments with a new goal and more obstacles.

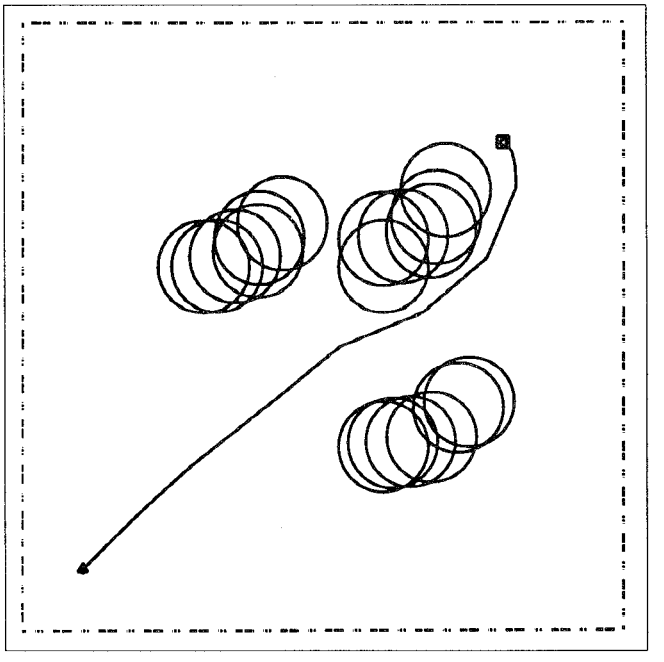

Fig. 9. Coping with dynamic obstacles.