

# Neural Reinforcement Path Planning for the Miniature Robot Khepera

Samira Sehad and Claude Touzet<sup>†</sup>  
LGI2P/EMA-EERIE, Parc Scientifique G. Besse  
F-30000 Nimes, FRANCE

<sup>†</sup> DIAM-IUSPIM, Domaine universitaire de St Jerome  
F-13397 Marseille Cedex 20, FRANCE  
email : samira@eerie.eerie.fr, diam\_ct@vmes11.u3mrs.fr

## Abstract

In this paper, we present a neural reinforcement path planning for the miniature mobile robot Khepera. The neural reinforcement learning is a self-organizing map implementation of Q-learning. The path planning task for the mobile robot is to move, in a euclidean space, from a given position and direction to a goal position. The Kohonen map is built by exploration and creates a self-organizing representation of the environment. Collisions with obstacles are detected using sensors and then used in the path planning. Results are obtained in the real world.

## 1 Introduction

Several authors [1][2] have worked on the path planning problem using Reinforcement Learning (RL) but generally, their experiments were carried out on simulations, rarely with a mobile robot moving in the real world. As a result, performances of the methods used in those experiments are not known when used in real world situations. In particular, a self-organizing implementation of Q-learning [3] has never been applied to the problem of optimal path planning with changing goals. This paper is organized as follows: In section 2, we describe the miniature robot Khepera. Section 3 defines the path planning problem. Section 4 presents the reader with a self-organizing map implementation of the Q-learning algorithm. We then instantiate the model for path planning in section 5. Experiments and results obtained in the real world are described in section 6. We conclude with a discussion about the limitations of this work and our plans for futur researchs.

## 2 The miniature mobile robot Khepera

Khepera [4] (see Fig [1]) is a miniature robot of circular shape measuring 55mm in diameter, 30mm in height and 86g of weight. It was developed to test mobile robot control algorithms in the real world. Its configuration consists of two wheels and 8 infrared proximity sensors placed around its body (6 on the front, 2 on the back). These sensors are sufficient for simple obstacles and light detection. Sensor values are numerical, ranging from 0 (for a distance  $> 5$  cm) to 1023 (approximately 2cm). Wheels have integer values between  $]-10, 10]$ . Khepera can be used on a desk, connected to a workstation through a wired serial link. This configuration allows an optimal experimental configuration with everything at hand: the robot, the environment and the remote host computer.

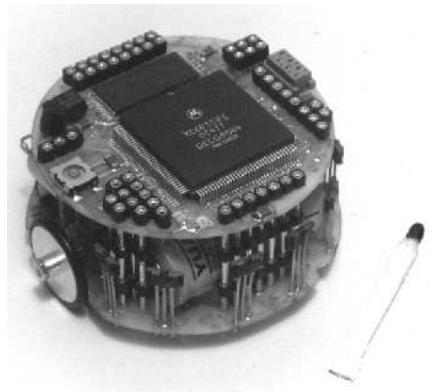


Figure 1: Khepera, the miniature mobile robot

### 3 The path planning problem

Let a robot moving in a Euclidean space  $E$ , be represented as  $R^2$  [5] (see Fig[2]). The problem may be defined as follows: for a given initial position, orientation and a goal position of the robot in  $R^2$ , generate a path  $\tau$  specifying a continuous sequence of positions and orientations of the robot avoiding contact with obstacles, starting at the initial position and orientation, and terminating at the goal position.

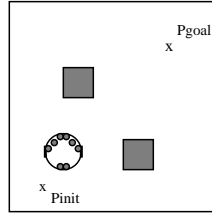


Figure 2: The mobile robot environment

### 4 A self-organizing map for the neural implementation of Q-learning

The self-organizing map [6] is distinguished by the development of a significant spatial organization of the layer. Weights specify clusters that sample the input space such that the point density function of the cluster tends to approximate the probability density function of the input vectors. In addition, the weights are organized such that topologically close neurons are sensitive to inputs that are physically similar. The connections with neighboring neurons allow a continuous mapping of the inputs to be generated. The self-organizing map for neural implementation of Q-learning has been presented in [7].

- **The internal state :** The internal state is composed of the set of weights of the network ( $W$ ). The memory size required by the system to store the knowledge is then defined, a priori, by the number of connections in the network. It is independent of the number of explored situation-action pairs.
- **Evaluation function :** is based on euclidean distances between the stored weights and the input situation.
- **Update function :** The update function is the learning rule of the Kohonen map and the Q-learning rule for updating the utility value associated with a situation-action pair.

### 5 Instantiation of the neural implementation of Q-learning for path planning

We propose to construct a path as the product of successive path segment starting at any initial position and ending at any specified goal position.

A self-organizing map is an associative memory. It is therefore possible to retrieve information using an incomplete probe. For use in path planning (see Fig[3]) the following question should be asked: for a given position and orientation of the robot in the workspace, what is the best rewarding action  $a$  to be taken in order to reach the goal by the shortest path ? The probe is therefore the current position  $p$ , the current robot orientation  $d$ , the goal position  $g$  and the maximum utility value  $q$  ( $= 1$  for example). The activated neuron corresponds to a vector composed of a position, orientation, goal position, action, utility value. The action retrieved from the self-organizing map is the one which offers the best reward in the present situation. A random component is added to the proposed action so as to explore the search space.

- **The internal state :** The internal state is composed of the set of weights ( $W_p, W_d, W_g, W_a, W_q$ ). Each neuron weight of the map stores the vector input ( $p, d, g, a, q$ ).

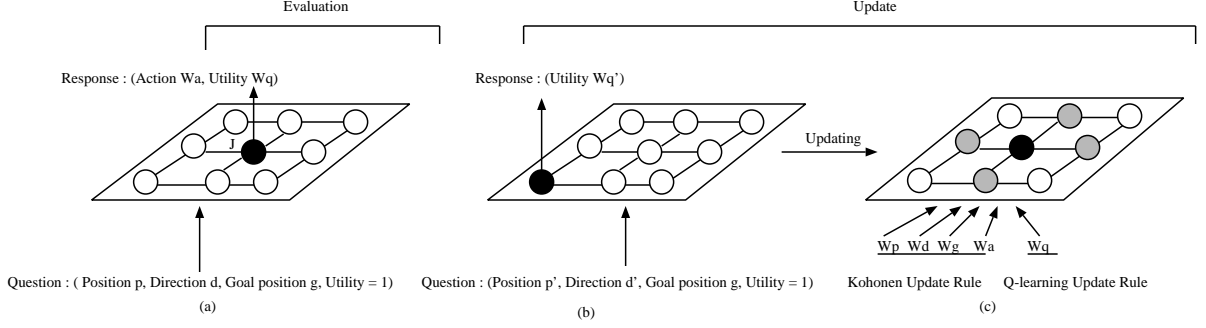


Figure 3: (a) The components  $p$  and  $d$  are the current position and direction,  $g$  is the goal,  $q$  is the utility value. The input vector, composed only of  $(p, d, g, q = 1)$ , allows the neuron  $j$  with the maximal utility value associated with  $p, d$  and  $g$  to be found and the associated action  $a$  to be obtained. (b) The components  $p', d'$  are the position and the direction of the robot after executing the action  $a$  beginning at position  $p$  with a direction  $d$ . The input vector  $(p', d', g, q = 1)$  allows the maximal utility value associated with  $p', d', g$  to be found. (c) Updating of  $W_p, W_d, W_g$  and  $W_a$  according to the Kohonen rule. Updating of  $W_q$  according to the Q-learning rule (for neuron  $j$  and its neighbors).

- **Evaluation function :** The selected neuron is the one with the minimum distance  $d_j$ . We compute the distance  $d_j$  between the input and each neuron  $j$  using  $d_j = (p - W_p)^2 + (d - W_d)^2 + (g - W_g)^2 + (1 - W_q)^2$ , where 1 is the best possible utility that can be associated with the situation constituted by  $p$  and  $d$  for the goal  $g$ . In a position  $p$  and a direction  $d$ , the action  $a$  to be performed by the agents is given by :

In the learning phase :

$a = a_{rand}$  when the robot is moving far from obstacles,  $a_{rand}$  is a random component between  $[-0.2, +0.6]$ .

$a = a_{obstacle-avoidance}$  when the robot is near obstacles,  $a_{obstacle-avoidance}$  is an action given by a Braitenberg's controller [8] or a self-organizing map for obstacle-avoidance [7].

The sums of the front and back sensors are computed. If these sums exceed the thresholds (3000, 1000), the robot is near an obstacle which it must avoid. Thresholds are determined through extensive experiments.

In the test phase :

$a = W_a$ , the weight of the action input of the selected neuron.

- **Update function :** Once a neuron has been selected, its weights and those of its neighboring neurons are modified to make the neurons more responsive to the current input (see Fig[3](b, c)). New weights for the coding ( $W_p$ ), ( $W_d$ ), ( $W_g$ ) and ( $W_a$ ) are given by :

$$W(t + 1) = W(t) + \eta(t) \cdot (X(t) - W(t))$$

where

$\eta(t)$  is a gain term  $0 < \eta(t) < 1$  that decreases in time. It is setting at 0.34 for the winner neuron and 0.30 for its neighbors

$X(t)$  is the input vector to neuron  $j$  at time  $t$ ,

$W(t)$  is the weight vector from the neuron  $j$  at time  $t$ .

( $W_q$ ) is updated by the following equation :

$$W_q(t + 1) = W_q(t) + \beta \cdot (r + \gamma \cdot W_q'(t) - W_q(t))$$

where  $W_q'(t)$  is the best utility associated with the position  $p'$  and the direction  $d'$  after executing  $a$  starting from the position  $p$  with a direction  $d$  (see Fig[3](b)). The input vector  $(p', d', g, q = 1)$  allows the value  $W_q'(t)$  to be found.

$\beta$  and  $\gamma$  are constant parameters setting to 0.6 and 0.4.

## 6 Experiments and results

- **The map:** In our experiments, we used a two-dimensional map with 256 neurons (16x16). The Kohonen map receives an input vector of 9 bits (current position  $(x, y)$ , direction  $(d_1, d_2)$ , goal position  $(x_g, y_g)$ , actions  $(a_1, a_2)$ , utility  $q$ ). Positions are simply pairs of cartesian coordinates. The position and direction of the robot in the workspace are given by a computed odometry. The size of the weights is  $(256 \times 9)$ . They are initialized randomly at the beginning.
- **The reinforcement function:** We compute a Manhattan distance between the goal position and the present position  $(x', y')$  and the last position  $(x, y)$ . If the difference of the two computed distances is greater than 0.02 then  $r = -1$ . This means that the robot is moving away from the goal. If the difference is less than 0.02 then  $r = 1$ . This indicates that the robot is moving towards the goal. 0.02 has been determined through experiments.
- **Experiments and results:** The robot is put in a 30x30 cm maze, at an arbitrary position referenced as the position  $(0, 0)$ . An arbitrary goal in the  $(x, y)$  domain is selected. The learning phase allows Khepera to encounter pairs of different spatial positions and directions and to associate an action and a utility with each pair, relative to a goal position. Obstacle avoidance is made by a Braitenberg controller. After about 1000 learning steps we test the performance of the robot. Khepera is placed again at the position  $(0, 0)$ . When moving, it is attracted by the goal and don't collide obstacles. The robot does not stop exactly at the goal position but continues to move around it. Performances of Khepera are shown by (see Fig [4] (a)(b)) which displays discounted cumulative reward over time. The graphs measure the performance of the algorithm in minimizing errors. We see that the robot reach the goal in about 100 steps.

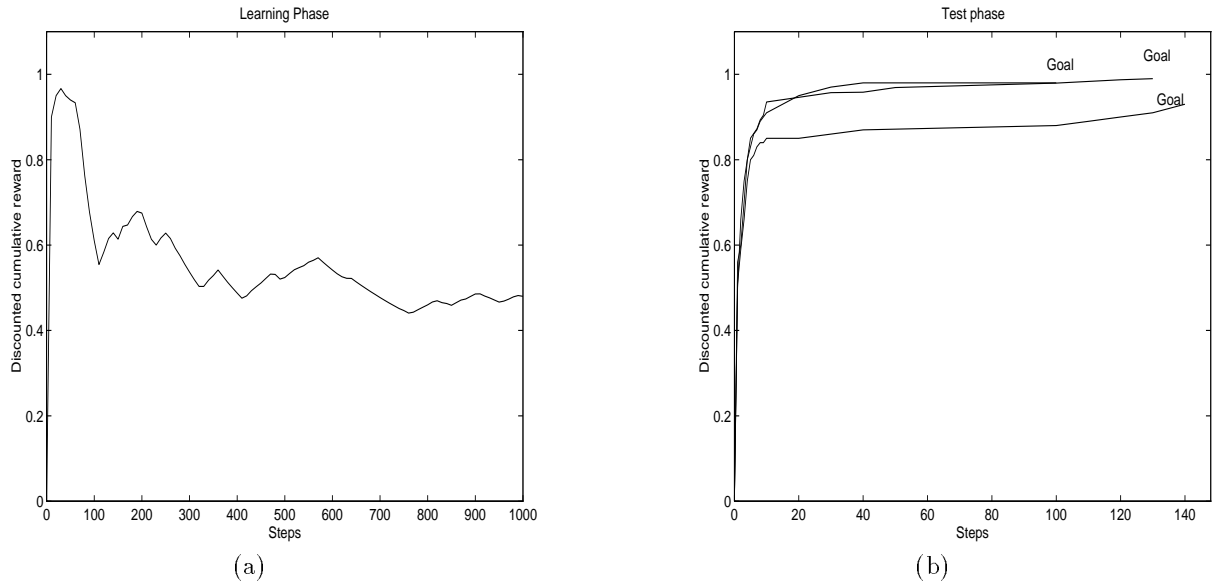


Figure 4 : Discounted cumulative reward over time of several experiments. (a) During the learning phase, the policy is randomly based and the performance is poor. (b) During the test phase, the policy is based on the self-organizing map, the performances is maximal.

The system builds a path in which every action taken by the robot leads to the best future rewards. Positive rewards are given by means of minimizing a Manhattan distance between the current position and the goal position. The generated path takes into account the location of the obstacles encountered during the learning phase. Note that none of the neurons encode an obstacle location. An interesting question is : is the path finally found, after  $\approx 1000$  steps, the one having the lowest cost?

## 7 Conclusion

We have proposed a model for path planning based on a self-organizing implementation of Q-learning. Contrary to a simulation environment, in the experiments described here, we deal with sensor noise and control errors due to a real world environment. A disadvantage of the proposed method is the selection of the size of the Kohonen network. It would be worthwhile to use an approach where the number of neurons can be adapted like RCE or RBF [9]. As shown by the results, this way of using neural RL appears applicable for generating paths on-line when obstacle location are unknown, only sensed during the learning.

## References

- [1] Millan J. and Torras C. A reinforcement connectionist approach to robot path finding in non-maze -like environments. *Machine Learning*, 8(3/4):363-395, 1992.
- [2] Lin L-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3/4):293-321, 1992.
- [3] Watkins C.J.C.H. and Dayan P. Technical note: Q-learning. *Machine Learning*, 8(3/4):279-292, 1992.
- [4] Franzi E. Mondada F. and Ienne P. Mobile robot miniaturisation: A tool for investigation in control algorithms. In *Third International Symposium on Experimental Robotics*, Kyoto, Japan, October 1993.
- [5] Latombe J.C. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
- [6] Kohonen T. *Self-Organisation and Associative Memory*, volume 8. Springer-Verlag, 1984.
- [7] Sehad S. and Touzet C. Self-organizing map for reinforcement learning: Obstacle avoidance with khepera. In *From Perception to Action*, Lausanne, Switserzerland, September 1994. IEEE Computer Society Press.
- [8] Braitenberg V. *Vehicles:Experiments in Synthetic Psychology*. MIT Press, 1986.
- [9] Blayo F. and Verleysen M. Setting initial conditions for the rce model. In *Fist IFIP Workshop WG-I0.6 on Neural Networks*, 1992.