A reinforcement learning approach to obstacle avoidance of mobile robots

Kristijan Maček
University of Zagreb
Faculty of Electrical Engineering and Computing
Department of Control and Computer Engineering in Automation
kristijan.macek@fer.hr

Ivan Petrović
University of Zagreb
Faculty of Electrical Engineering and Computing
Department of Control and Computer Engineering in Automation
ivan.petrovic@fer.hr

Nedjeljko Perić
University of Zagreb
Faculty of Electrical Engineering and Computing
Department of Control and Computer Engineering in Automation
nedjeljko.peric@fer.hr

Abstract: One of the basic issues in navigation of autonomous mobile robots is the obstacle avoidance task that is commonly achieved using reactive control paradigm where a local mapping from perceived states to actions is acquired. A control strategy with learning capabilities in an unknown environment can be obtained using reinforcement learning where the learning agent is given only sparse reward information. This credit assignment problem includes both temporal and structural aspects. While the temporal credit assignment problem is solved using core elements of reinforcement learning agent, solution of the structural credit assignment problem requires an appropriate internal state space representation of the environment. In this paper a discrete coding of the input space using a neural network structure is presented as opposed to the commonly used continuous internal representation. This enables a faster and more efficient convergence of the reinforcement learning process.

1 Introduction

One of the basic issues in navigation of autonomous mobile robots is the obstacle avoidance capability, which fits into the path planning problem. When a complete knowledge of the environment is assumed global path planning techniques can be applied [1], [2]. However, efficiency of such techniques decreases rapidly in more complex and unstructured environments since considerable modeling is needed. Therefore, local path planning techniques, which rely on on-line sensory information of the mobile robots, may prove more adequate in achieving the task of obstacle avoidance.

Among these, the reactive control paradigm is commonly used, where a mapping from perceived states to actions is made. Acquiring the state-action pairs is especially interesting using approaches where learning capabilities apply, such as fuzzy logic and neural networks. Nevertheless, learning the fuzzy logic rule base or providing the necessary target patterns in the supervised neural network learning may be a tedious and difficult task.

A plausible solution is the reinforcement learning approach, where only sparse information is given to the learning agent in form of a scalar reward signal beside the current sensory information [3]. A continuous internal state representation of the input space may be based on fuzzy logic rules [4], [5], [6] or neural network structures such as MLP neural networks [7]. Since the agent must develop an appropriate state-action strategy for itself, a laborious learning phase may occur. Therefore, special attention must be paid to the convergence rate of a particular reinforcement learning approach.

In this paper, convergence rate increase is obtained by reduction of the continuous internal state representation to a set of discrete states thereby extracting the relevant features of the input state space. Moreover, by allowing only a single discrete state to represent the input space at any given time, the discrete states are contrasted which results in individual credit assignment and more precise computation. Similar approaches can be found in [8] and [9], but our approach provides generally more advantageous reinforcement learning scheme.

2 Reinforcement learning

The basic reinforcement learning framework consists of a learning agent that observes the current state of the environment $\vec{x}_t$ and aims to find an appropriate action a_t pertaining to a policy π that is being developed. As a measure of success of agent's interaction with the environment the agent is given an external reward (penalty) $r = r_t$ in time instance t which is defined by the designer and describes the overall goal of the agent. For obstacle avoidance purposes $r = -1$ upon collision with an object and $r = 0$ otherwise. In order to develop a consistent policy, the reward function must be fixed.

The aim of the intelligent agent is to maximize the expected sum of discounted external rewards r for all future instances:

$$E\left(\sum_{\tau=t}^{\infty}\gamma^{\tau-t}r(\tau)\right),\quad 0<\gamma<1, \tag{1}$$

where coefficient γ determines how "far-sighted" the agent should be.

Moreover, each state $\vec{x}$ of the system can be associated with a measure of desirability $V_\pi(\vec{x})$ which represents the expected sum of discounted rewards r for

future instances when system is found in state $\vec{x}_t$ at time t and follows a fixed policy π thereafter:

$$V_\pi\left(\vec{x}\right) = E\left\{\sum_{\tau=t}^{\infty}\gamma^{\tau-t}r\left(\tau\right)\middle|\vec{x}_t=\vec{x}\right\},\ 0<\gamma<1. \qquad (2)$$

A higher state desirability $V_\pi\left(\vec{x}\right)$ implies a greater expected total reward sum.

Since the external reward signal r is not informative enough and can be delayed for a long time [10], the internal reinforcement signal r^* is derived that represents the immediate reward given to the system in terms of correctness of the actions executed so far. It also represents the prediction error of the total reward sum between two successive steps:

$$r^*\left(t\right) = r\left(t\right)+\gamma V\left(\vec{x}_t\right)-V\left(\vec{x}_{t-1}\right). \qquad (3)$$

If $r^*\left(t\right)>0$, the system performed better than expected in the last time step, if $r^*\left(t\right)<0$, the system performed worse than expected. Therefore, the internal reinforcement signal $r^*(t)$ gives a measure of quality of the last action a_{t-1} taken and may be used as a learning signal for parameter update of the learning system.

A generic expression for updating of the k-th parameter p_k of the learning system can be formulated as:

$$p_k(t) = p_k(t-1)+\alpha r^*\left(t\right)e_k\left(t-1\right), \qquad (4)$$

where α is the learning rate and e_k eligibility measure of how a certain parameter p_k influenced the evaluation of state desirability and the action choice in previous time steps. Since prediction error r^* is only known in time step t, eligibility measure of parameter p_k is taken from time step t-1 since parameter p_k influenced the current prediction of total discounted reward sum $r\left(t\right)+\gamma V\left(\vec{x}_t\right)$.

When considering the credit given to the learning agent for its actions, eligibility measures involve both structural and temporal credit assignment problems. The structural credit assignment depends on structure of the learning agent and the temporal credit assignment involves measuring credit (or blame) of a certain sequence of actions to the overall performance of the learning system. Depending on structure of the learning agent and specific reinforcement learning scheme used, the eligibility measure e_k may acquire different forms.

Basically, the current state of the system is due not only to last action taken but also due to other actions taken in past. If a certain parameter p_k of the learning system is involved in current state evaluation and consequentially in action selection, its eligibility measure e_k is updated according to structural credit. Thereafter, if not involved in further state-action assessments in the future its eligibility measure e_k typically decays exponentially:

$$e_k\left(t\right) = \lambda e_k\left(t-1\right),\ 0\le\lambda\le 1. \qquad (5)$$

If $\lambda = 0$, only the last state-action assessment is considered relevant to the current state of the system, however if $\lambda = 1$, all past assessments are considered equally relevant.

If the desirability of a state is associated with a certain action $Q\left(\vec{x}_t, a_t\right)$, expression (3) for the prediction error becomes:

$$r^*\left(t\right) = r\left(t\right)+\gamma Q\left(\vec{x}_t, a_t\right)-Q\left(\vec{x}_{t-1}, a_{t-1}\right). \qquad (6)$$

Based on theory of dynamic programming one can distinguish between two basic reinforcement learning approaches where: 1.) state evaluations $V\left(\vec{x}_t\right)$ or 2.) state-action evaluations $Q\left(\vec{x}_t, a_t\right)$ are calculated [11]. In 1.) policy is derived indirectly through state evaluations $V\left(\vec{x}_t\right)$ so that if $V\left(\vec{x}\right)$ for all possible states $\vec{x}$ are maximized the optimal policy is achieved. In 2.) separate state-action evaluations $Q\left(\vec{x}_t, a_t\right)$ are updated basically when a certain action a_t is taken at time t, thus when all $Q\left(\vec{x}, a\right)$ are maximized, the optimal policy is to choose actions with maximal $Q\left(\vec{x}, a\right)$ values.

3 Controller design

3.1 Input state-space discretisation

As outlined briefly in the previous section, two aspects are particularly important to an autonomous agent using reinforcement learning, namely, the temporal and the structural credit assignment problem. The temporal credit assignment problem is related to rewarding a particular action sequence in time and is solved using core elements of any reinforcement learning agent such as state desirability estimates, the internal reinforcement signal and eligibility measures.

However, the structural credit assignment problem that is related to the internal representation of environment of the learning agent is yet a difficult task. Since reinforcement learning methods are iterative procedures, an inadequate internal representation of the input space (the environment) may result in a very slow convergence rate of the learning process because the credit or blame for certain sequence of actions may be distributed among many internal regions of state space.

Approaches such as fuzzy logic controllers or multi-layer perception neural networks involve a continuous internal representation of the input state space. This may

result in a large number of fuzzy rules to be adjusted simultaneously or in an on-line back-propagation learning of neural networks with a slow convergence rate.

To enhance individual structural credit assignment, the continuous internal representation of the input space may be replaced by a set of discrete states. A possible approach is to use Kohonen neural network structure with the winner-take-all unsupervised learning rule [12].

Learning is based on clustering of input data in order to group similar inputs and separate dissimilar ones. It is given that the number of clusters is N, input state vector is $\vec{x} = [x_1 x_2 ... x_L]'$ and the set of weight vectors is $\{\vec{w}_1, \vec{w}_2, ..., \vec{w}_j, ..., \vec{w}_N\}$, where $\vec{w}_j$ connects input vector $\vec{x}$ to cluster *j*. Then the winning neuron *k*, representing the discrete state *k*, is selected by similarity matching:

$$k : \left\| \vec{x} - \vec{w}_k \right\| \le \left\| \vec{x} - \vec{w}_j \right\| \ , \ \forall j \ . \tag{7}$$

The update rule for the winning vector $\vec{w}_k$ is:

$$\vec{w}_k(t+1) = \vec{w}_k(t) + \eta(\vec{x}(t) - \vec{w}_k(t)) \ . \tag{8}$$

Since the weight vector $\vec{w}_k$ closest to input vector $\vec{x}$ is determined only by the angle spanned between these two vectors, both input vector $\vec{x}$ and the weight vector $\vec{w}_k$ must be normalized, the former before feedforward pass and the later after updating. Thus, all weight vectors $\vec{w}_j$ are spread along the unit circle.

Activation function of the neurons representing clusters is not important since the output of the *j*-th neuron equals $y_j = 0, \forall j \neq k$ and $y_j = 1, j = k$.

3.2 Action (policy) learning

In addition to discrete internal state representation, a discrete set of *M* actions $\{a_1, a_2, ..., a_M\}$ was chosen where each state-action (cluster-action) pair in time *t* is associated with a desirability measure $Q(\vec{x}_t, a_t)$. For environment state $\vec{x}_t$, *k*-th neuron is the winning one and desirability measures for each action are $Q(\vec{x}_t, a_1) = \mu_{k1}$, $Q(\vec{x}_t, a_2) = \mu_{k2}, ...,\ Q(\vec{x}_t, a_M) = \mu_{kM}$. The action chosen at time *t* is the one with maximum desirability measure $Q(\vec{x}_t, a_t)$, which leads to the "greedy" policy. The internal reinforcement signal r^* (prediction error) is according to (6):

$$r^*(t) = r(t) + \gamma \max_{a_t} Q(\vec{x}_t, a_t) - Q(\vec{x}_{t-1}, a_{t-1}) \ . \tag{9}$$

The eligibility measure e_{ji} for all weight vectors $\vec{\mu}_j = [\mu_{j1} \mu_{j2} ... \mu_{jM}]'$ is as follows:

$$e_{ji}(t) = \begin{cases} 1 & j = k, a_t = a_i \\ \lambda e_{ji}(t-1) & else \end{cases} , \begin{matrix} 1 \le j \le N \\ 1 \le i \le M \end{matrix} \ . \tag{10}$$

Thus, only the eligibility measure e_{ki} of the *k*-th winning neuron where action a_i is chosen at time *t* is set to 1, whereas all other eligibility measures e_{ji} are decayed according to their impact in the past.

The update rule for the state-action estimates μ_{ji} is according to (4) derived as:

$$\mu_{ji}(t) = \mu_{ji}(t-1) + \alpha r^*(t) e_{ji}(t-1) \ , \quad \begin{matrix} 1 \le j \le N \\ 1 \le i \le M \end{matrix} \ . \tag{11}$$

Initially, all eligibility measures are set to zero.

Similar controller architecture and the input state space discrete coding was elaborated in [6]. But there reinforcement learning was based on state evaluations $V(\vec{x}_t)$ and we based it on state-action evaluations $Q(\vec{x}_t, a_t)$, which is generally a preferable solution in terms of convergence rate [13]. The controller structure is depicted in Fig.1.

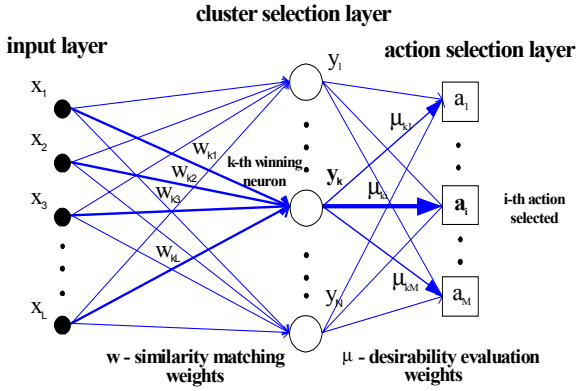

Fig.1: The neural network controller structure.

4 Simulation results

The experiments were carried out in the Saphira Pioneer simulation environment for the Pioneer DX2 mobile robot platform with front array of 8 sonars oriented at $\pm 90^\circ, \pm 50^\circ, \pm 30^\circ, \pm 10^\circ$ relative to the longitudinal vehicle axes. Sonar measurements d_i $(1 \le i \le 8)$ were coarse coded as:

$$x_i = 1.0 - 2.0 \frac{d_i - RANGE\_MIN}{RANGE\_MAX - RANGE\_MIN}. \quad (12)$$

where *RANGE_MAX*, *RANGE_MIN* denote maximal and minimal range of sonars, respectively, giving a nominal $[-1,\ 1]$ sonar range. *RANGE_MAX*, *RANGE_MIN* may be chosen arbitrarily (yet taking into account the physical sonar constraints) and define the active sensor region, in our case also the active learning region. It is chosen in our case for *RANGE_MIN*=15*cm* and *RANGE_MAX*=250*cm*. The coarse coded sonar measurement vector $\vec{x} = [x_1 x_2 ... x_8]'$ is normalized and given as input vector to the controller as described in the previous section (see Figure 1).

The action set chosen was {"move forward a distance *d*", "turn left a heading θ and move forward a distance *d*", "turn right a heading θ and move forward a distance *d*"}, where $d = 10cm$ and $\theta = 30^\circ$ (action set number *M*=3). It can be seen from the action set that the robot is on constant forward move. There are two main reasons for this: firstly, by random exploration and obstacle avoidance the mobile agent can build a map of initially unknown environment (thus potentially fulfilling other important tasks of mobile robot navigation), secondly, the mobile agent is prevented from being stuck in a local minimum, such as turning on the same spot. Moreover, instead of choosing a constant forward velocity a constant forward distance action was chosen which enables deriving direct mapping from states to actions regardless of the vehicle dynamics.

The simulated experiment is performed in the following manner: when in active sonar region, depending on the current sonar readings and state-action evaluations, an appropriate action is chosen. Controller parameters are updated thereafter. As stated earlier, the external reward signal r is 0 for all states in the active region. Upon collision with an object (as detected by additional bumper sensors) or upon entering the "dangerous zone" of 15*cm* to the closest object (as read by sonars) the mobile agent receives a negative external penalty $r = -1$. Robot agent is than considered to be in failure state and a trial restart must be performed. The robot is moved back to the "save zone" using very simple if-then logic (i.e. if obstacle is in front then move backward). Therefore, no external trial restart is required which is advantageous to some previous approaches. Upon trial restart all eligibility traces of the action selection layer of the controller are reset to zero.

Initial positions of weight vectors $\{\vec{w}_1, \vec{w}_2, ..., \vec{w}_j, ..., \vec{w}_N\}$ of the controller are uniformly random distributed along the unit circle, whereas weight vectors $\vec{\mu}_j$ are initialised randomly in interval $[-0.1,\ 0.1]$. The corresponding learning rates are set to $\eta = 0.01$ and $\alpha = 0.05$, respectively. Discount factor for future predictions is $\gamma = 0.99$ and the decay rate is $\lambda = 0.05$.

The learning algorithm was verified in corridor-like environments designed in the Saphira world simulator. The mobile robot was allowed to randomly explore the environment and thus acquired robot paths are depicted in Figures 2 to 5.

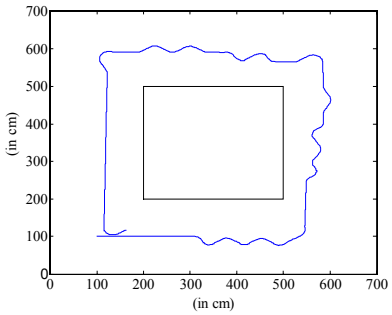

Fig.2: A robot path with 20 clusters internal representation.

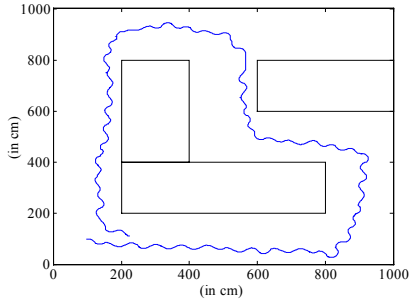

Fig.3: A robot path with 30 clusters internal representation.

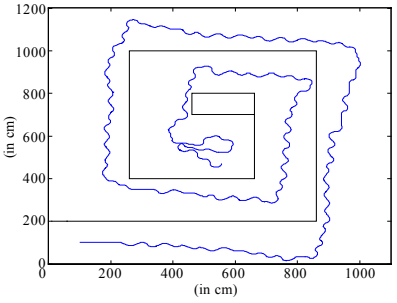

Fig.4: A robot path with 30 clusters internal representation.

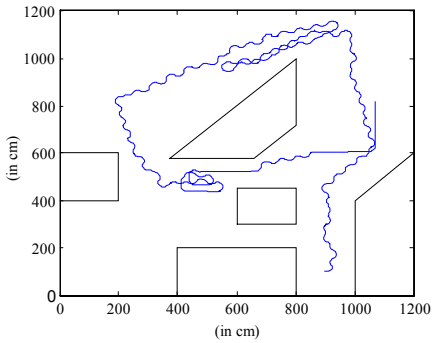

Fig.5: A robot path with 30 clusters internal representation.

To verify the learning algorithm the greedy policy was applied, as stated earlier. Essentially, the learning agent chooses the action with maximum desirability, which is considered satisfactory thereafter. This may result in sub optimal solutions (a pending motion in case when a straightforward motion is optimal). Since the primary task of obstacle avoidance is satisfied, a further improvement of optimality of solution should include a form of stochastic action exploration and a longer action sequence history. These aspects are to be included in further elaboration.

Moreover, to achieve the global path planning navigation a goal seeking behavior must be included and coordinated with the local obstacle avoidance task. If the action set is chosen in such a way that robot is on a constant forward move a map building, path tracking functionality may be included, converting an initially unknown environment into a known one, where global path planning techniques may be applied thereafter. However, these aspects are beyond the scope of this paper.

5 Conclusions

A reinforcement learning approach for the obstacle avoidance task of navigation of mobile robots was developed. In general, reinforcement learning methods present a suitable solution to developing control strategies in an unknown environment. The learning agent may receive only sparse reward signals or credits for the actions taken, therefore internal state and action evaluation is required. In terms of credit assignment problem internal representation of the input state space is particularly important.

A clustered discrete coding of the input state space was developed using Kohonen neural network structure as opposed to continuous internal state representation. This enabled individual state-action credit assignment, giving a more precise evaluation computation and a faster convergence rate.

The learning algorithm was verified in simulation environment where the mobile robot performed obstacle avoidance capability, which was developed by using initially unknown control strategy.

In perspective, optimality of the solution obtained should be taken into account as well as an adaptive neural network structure which could reduce the size of relevant feature representation to a minimum required to fulfill a given task such as obstacle avoidance.

6 References

[1] J.T. Schwartz, M.Shirir: "A survey of motion planning and related geometric algorithm", *Artif. Intell. J.*, vol. 37, pp. 157-169, 1988.

[2] O. Khatib: "Real-time obstacle avoidance for manipulators and mobile robots", *Int. J. Robot. Res.*, vol.5. no. 1., pp. 90-98, 1986.

[3] A.G. Barto, R.S. Sutton, and C.W. Anderson: "Neuronlike adaptive elements that can solve difficult learning control problems", *IEEE Trans. Syst. Man. Cybern.,* vol. SMC-13, no. 5, pp. 834-847, 1983.

[4] C.T. Lee, C.S.G. Lee: "Reinforcement structure/parameter learning for neural-network-based fuzzy logic control system", *IEEE Trans. on Fuzzy Systems,* vol. 2, no. 1, pp. 46-63, 1994.

[5] H. Beom, H. Cho: "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning", *IEEE Trans. Syst. Man. Cybern., vol. SMC-25,* no. 3, pp. 464-477, 1995.

[6] N.H.C. Yung, C. Ye: "An intelligent mobile vehicle navigator based on fuzzy logic and reinforcement learning", *IEEE Trans. Syst. Man. Cybern., vol. SMC-29,* no. 2, pp. 314-321, 1999.

[7] G.A. Rummery: *Problem solving with reinforcement learning*, PhD Thesis, Cambridge University Engineering department, University of Cambridge, 1995.

[8] B.J.A. Krose, J.W.M van Dam: "Learning to avoid collisions: a reinforcement learning paradigm for mobile robot navigation", *Proceedings of IFAC/IFIP/IMACS Symposium on Artificial Intelligence in Real-Time Control,* pp. 295-30, 1992.

[9] A.H. Fagg, D. Lotspeich, and G.A. Bekey: "A Reinforcement-Learning Approach to Reactive Control Policy Design for Autonomous Robots", *Proc. of the 1994 IEEE International Conference on Robotics and Automation,* vol. 1, pp. 39-44, San Diego, CA, May 8-13, 1994.

[10] R.S. Sutton: "Learning to predict by the methods of temporal differences", *Machine Learning 3,* pp. 9-44, 1988.

[11] R.S. Sutton: "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming", *in Seventh International Conference on Machine Learning,* pp. 216-226, 1990.

[12] T. Kohonen: *Self-organization and associative memory,* Springer Verlag, 1984.

[13] C.J.C.H. Watkins, P. Dayan: "Technical Note: Q-learning", *Machine Learning,* vol. 8, pp. 279-292, 1989.