BULETINUL Universităţii Petrol – Gaze din Ploieşti	Vol. LXI No. 3/2009	261 - 266	Seria Tehnică

Self-programming Autonomous Mobile Robots

Alexandru Cristian Georgescu

Petroleum – Gas University of Ploieşti, 39 Bucureşti Blvd., Ploieşti, ROMÂNIA
e-mail: ageorgescu@upg-ploiesti.ro

Abstract

This paper presents a way of self-rewriting the control program for individual autonomous mobile robots that are performing certain common tasks. Each robot has an artificial intelligence mechanism that is performing changes in its programming according to the changes in the working environment. The main program stored in the robot's microcontroller cannot be changed, yet the only portion that can be modified is the one stored in external EEPROM or Flash memory. Still the modification of the programming is limited. Only certain variables and/or routines can be altered resulting in behavioral changes over time.

Key words: *mobile robots, self-programming, self-adjusting, artificial intelligence*.

Introduction

An autonomous mobile robot is by definition a resource limited machine, regarding all the aspects of its functioning: range, computing power, energy, memory, etc. A control program should therefore optimize each command and each process in order to achieve the lowest rates of power consumption. A normal program should take into consideration all the aspects involved in the robot functioning and give the appropriate commands to the actuators in order to perform specific tasks. But what if the robot is facing a different scenario than the one it was programmed for, then it will have only two possibilities: to adapt to the new conditions based only on the current programming, or to start learning and modify its own programming according to the new requirements.

When dealing with a group of cooperating mobile robots, becoming something close to *an artificial intelligence* can be achieved in many ways:

- using collective cognition and control structures;
- using bio-inspired principles;
- autonomous reconfiguration;
- genetic learning;
- sharing of computational and communication resources;
- using neural networks;
- using certain AI algorithms designed for self-configuration, self-adjustment, self-learning and self-programming.

Therefore in order to have a mobile robot capable of changing its own programming according to the changes in the environment, the main program should allow making modifications, and storing new variables or replacing old ones.

Artificial Intelligence

An intelligent agent is a system that perceives its environment and takes actions which maximize its chances of success. For such a robot to exist it needs to perform the following: reasoning, knowledge, planning, learning, communication, perception and the ability to move or to interact with the environment. Many mobile robots, in order to function properly, already have the capabilities to communicate and to perceive the environment using the sensors. But for a robot to be able to learn or to plan future operations it must have software that allows self-changing or self-adjusting.

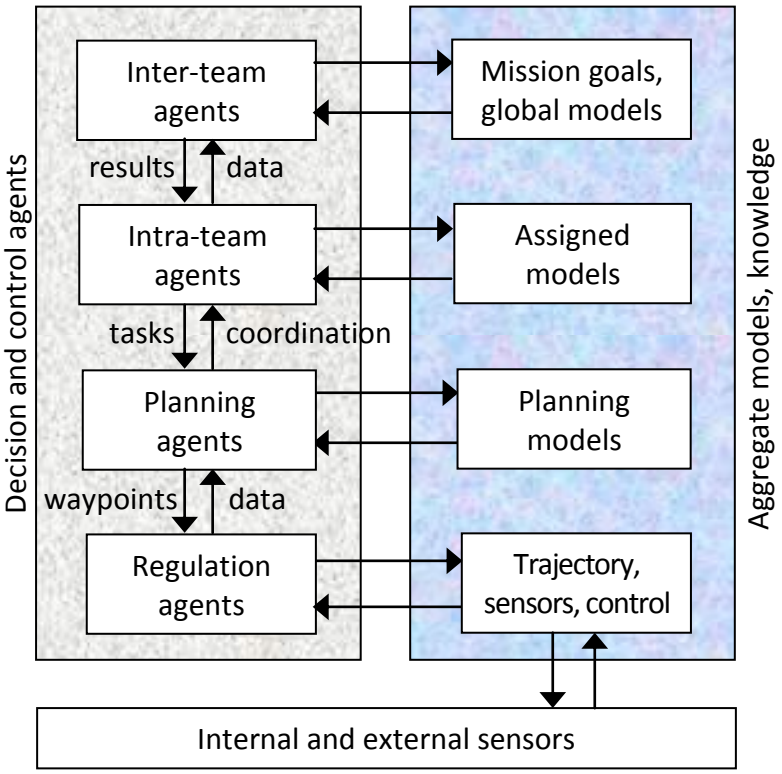

Fig. 1. General architecture for robot cooperative control.

Figure 1 illustrates a general architecture for cooperative control and resource allocation among multiple robots [9]. There are a lot of sequences, presented here as agents, that need to be implemented in the control algorithms of each robot, in order to have a cooperating environment where each individual has its own part in the network.

The inter-team agents have the role of configuring teams of robots and providing them with their goals [9]. They are also responsible for passing along the knowledge and decisions taken by some individuals in the network.

Each artificial intelligence mechanism that controls a robot will be based on predefined models and patterns and each new encountered situation will be treated using comparisons with previously stored models. Only when this method fails the robot will start making new strategies based on previous experiences, or on the experience of other robots in the network that are facing the same problems.

The learning mechanism used by the robots is reinforcement learning which allows an individual to solve different tasks by representing them as trial and error processes where single reinforcement signals indicate the level of success for each performed action [3].

An example of such an algorithm is presented in figure 2.

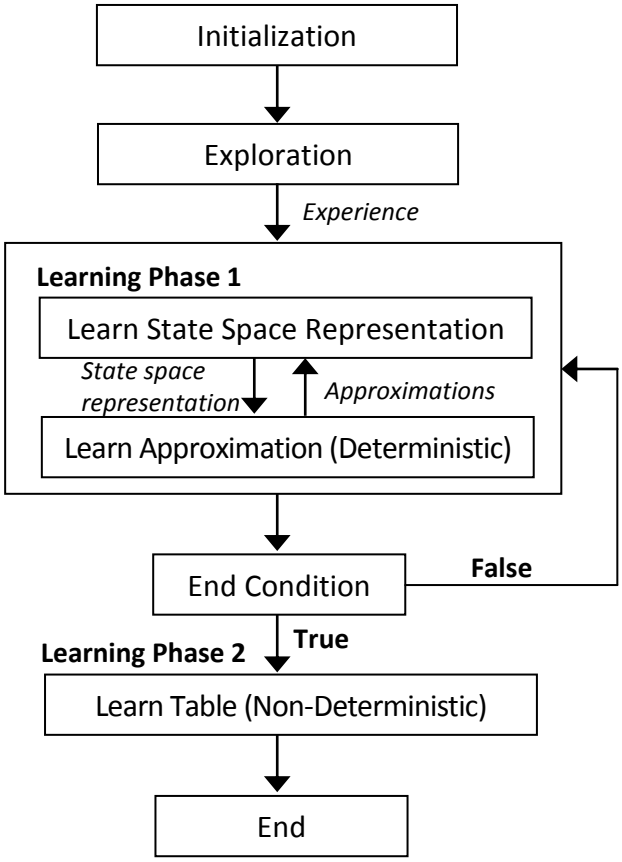

Fig. 2. The general layout of a self-adjusting learning algorithm.

A frequently used technique for learning robots is *Q-learning* which represents a type of *reinforcement learning*. The learning agent is faced with a series of possibilities, each providing a feedback, either a reward (a positive value) or a punishment (a negative value). The goal of the system is to get the a reward as high as possible. The algorithm calculates the *quality* of a state-action ($S \times A$) combination.

$$Q : S \times A \rightarrow R \tag{1}$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t)(1 - \alpha_t(s_t, a_t)) + \alpha_t(s_t, a_t)[r_{t+1} + \gamma \max_a Q(s_{t+1}, a)] \tag{2}$$

Both equations (1) and (2) describe the Q-learning algorithm, where $Q(s_t, a_t)$ represents the old value, $\alpha_t(s_t, a_t)$ the learning rate, r_t the reward, γ a discount factor and $\max_a Q(s_{t+1}, a)$ represents the maximum future value.

Before the beginning of the learning process, Q has a predefined value, given by the designer, but every time the agent is presented with a reward, it recalculates the Q value based on each combination of state and actions.

The main disadvantage for this type of learning algorithm is that it loses its performance while the complexity of the analyzed system increases. And also, while implemented on mobile robots

control systems, it is very difficult to identify important events in the multitude of sensory inputs and to temporarily memorize them in adaptive, dynamic, internal states until the memories can help to compute proper control actions [11].

Planning, learning and reprogramming

A small number of autonomous mobile robots share the same world, W and they all have a number from 1 to n. They are used for transportation of objects of unknown size and weight. Each robot has to learn how to interact with other robots, arrange the group in different formations and calculate the trajectories toward the target object and toward the destination.

A path must be computed for each robot to avoid collisions with obstacles or with other robots and also each robot, A^i, has its associated C-space, C^i and its initial and goal configurations, q^i_{init} and q^i_{goal}, respectively [8].

A state space is defined that considers the configurations of all robots simultaneously,

$$X = C^1 \times C^2 \times ... \times C^n . \tag{3}$$

A state $x \in X$ specifies all robot configurations and may be expressed as $x = \left(q^1, q^2, ..., q^n\right)$. The dimension of X is N, which is $N = \sum_{i=1}^{n} \dim(C^i)$.

There are two sources of obstacle regions in the state space: robot-obstacle and robot-robot.

For each i where $1 \le i \le n$ the subset of X that corresponds to robot A^i in collision with the obstacle region O is (only robot-obstacle):

$$X^i_{obs} = \{x \in X \mid A^i(q^i) \cap O \neq 0\} . \tag{4}$$

For each pair, A^i and A^j, of robots, the subset of X that corresponds to A^i in collision with A^j is:

$$X^{ij}_{obs} = \{x \in X \mid A^i(q^i) \cap A^j(q^j) \neq 0\} . \tag{5}$$

The resulting obstacle region in X is described by the following equation [8]:

$$X_{obs} = \left(\bigcup_{i=1}^{n} X^i_{obs} \right) \cup \left(\bigcup_{ij \quad i \neq j} X^{ij}_{obs} \right) . \tag{6}$$

The method of planning described above is called *decoupled planning* and it is used as a two step method. The first step is used to create the constraints related only to the robot-obstacle collisions and the second being used to create paths to avoid other robots.

The learning process is limited and consist of two separate processes, one in which pairs of initial and goal configurations (q^i_{init} and q^i_{goal}) together with the calculated trajectories for the respective pairs are analyzed and rated according to the Q-learning method and another in which the solutions with the highest reward r_t rate are stored.

Because the size and weight of objects that are to be transported can be different, certain robot formations and *relative-to-one-another moving patterns* are also being learned for future use in case of similarities.

The reprogramming of each robot is made using *dynamic programming*, a technique that is used to calculate the optimal value for certain given variables and only on certain possible scenarios. It is a five-step procedure:

- characterizes the structure of an optimal solution;
- recursively defines the value of an optimal solution;
- computes the value of an optimal solution from the bottom to the top level;
- constructs an optimal solution from computed information;
- stores the new solution and changes the initial configuration variables and/or routines to match the optimal values.

Limited programming changes

An autonomous mobile robot usually has a very limited storage memory capability and besides that the main program is stored in a protected region of the microcontroller, one that can't be modified while the program is running. Only portions of the code that is stored on external memory can be rewritten. Therefore all the values that need to be modified by *dynamic programming algorithms* have to be stored there.

Conclusions

The main difference between a robot that is capable of learning and one that uses dynamic algorithms to generate the optimal values for its movement trajectories or for its activities is that the learning robot can perform better and faster under previously experienced scenarios.

Besides *reinforcement learning* there are other methods like: *supervised learning*, *unsupervised learning*, *semi-supervised learning*, *transduction*, *learning-to-learn*; and all of these techniques can be implemented to control the movement and actions of an autonomous mobile robot.

This paper describes the methods used by a small group of autonomous mobile robots to reprogram themselves by changing previously stored models with new ones by altering the values of some predefined variables. A more powerful technique could be the use of neural networks and to store the new weights calculated after experiencing new training sets.

References

1. Bergren, C.M. – *Anatomy of a Robot.* McGraw-Hill, New York City, 2003.
2. Borenstein, J, Everett, H.R., Feng, L. – *Where am I? Sensors and Methods for Mobile Robot Positioning.* University of Michigan, 1996.
3. Fernandez, F, Borrajo, D., Parker, L.E. – *A Reinforcement Learning Algorithm in Cooperative Multi-Robot Domains.* Journal of Intelligent and Robotic Systems, Springer, 2005.
4. Fraden, J. – *Handbook of Modern Sensors 3rd Edition.* Springer-Verlag, New York City, 2004.
5. Ge, S., Lewis, F.L. – *Autonomous Mobile Robots – Sensing, Control, Decision Making and Applications.* CRC Press Taylor & Francis, Boca Raton, 2006.
6. Javidi, B. – *Image Recognition and Classification Algorithms, Systems and Applications.* Marcel Dekker, New York City - Basel, 2002.
7. Kennedy, J.R, Russell, C.E., Yuhui, S. – *Swarm Intelligence.* Academic Press, San Diego, 2001.
8. LaValle, S.M. – *Planning Algorithms.* Cambridge University Press, 2006.

9. Murphey, R., Pardalos, P.M. – *Cooperative Control Optimization*. Kluwer Academic Publishers, Dordrecht, 2002.
10. Siegwart, R., Nourbakhsh, I.R. – *Introduction to Autonomous Mobile Robots*. The MIT Press, Cambridge, 2004.
11. Schmidhuber, J. – *Learning Robots*. Cognitive Robotics at Dalle Molle Institute for Artificial Intelligence, 2004.
12. Walter, J. – *Rapid Learning in Robotics*. Technische Fakultät, Universität Bielefeld, 1997.

Roboți mobili autonomi autoprogramabili

Rezumat

Această lucrare prezintă o modalitate de auto-rescriere a programului de comandă pentru roboți mobili autonomi individuali care execută anumite sarcini în comun. Fiecare robot are un mecanism de inteligență artificială care realizează modificări în program în funcție de schimbările apărute în mediul de lucru exterior. Programul principal, stocat în memoria microcontroller-ului din componența robotului nu poate fi modificat, totuși porțiunea care poate fi modificată este stocată pe o memorie externă de tip EEPROM sau Flash. Dar și așa posibilitățile de modificare a programului sunt limitate. Doar anumite variabile și/sau rutine pot fi modificate ducând la o schimbare în viitor a comportamentului robotului.