Available online at www.sciencedirect.com

SCIENCE DIRECT®

Robotics and Autonomous Systems 54 (2006) 84–94

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

Navigation with memory in a partially observable environment

A. Montesanto[1], G. Tascini*, P. Puliti[2], P. Baldassarri[1]

Dipartimento di Elettronica, Intelligenza Artificiale e Telecomunicazioni, Università Politecnica delle Marche, Ancona, Italy

Received 25 February 2004; received in revised form 22 September 2005; accepted 28 September 2005
Available online 22 November 2005

Abstract

The paper presents an architecture that allows the reactive visual navigation via an unsupervised reinforcement learning. This objective is reached using Q-learning and a hierarchical approach to the developed architecture. Using these techniques requires a deviation from the Partially Observable Markov Decision Processes (POMDP) and some innovations: heuristic techniques for generalizing the experience and for treating the partial observability; a technique for the speed adjournment of the Q function; the definition of a special reinforcement policy adequate for learning a complex task without supervision. The result is a satisfactory learning of the navigation assignment in a simulated environment.

Keywords: Hierarchical representation of POMDPs; Task hierarchy; Hierarchical reinforcement learning; Q-learning; Self-localization; Navigation

1. Introduction

The autonomous navigation of a mobile agent is actually the object of a number of research work. The navigation ability in the environment is strictly related to the quality of the agent perceptions. These last depend on the quantity, quality and reliability of sensors and actuators. The robotic navigation based on visual servoing needs proper image processing algorithms, for connecting actions to visual information. Varela [19] asserts that sensorial processes and motor processes evolve together, by undertaking respectively perceptions and actions, that are considered as inseparable, ordered and aimed to the whole cognitive system growing.

The agents, in the physical world, rarely have exact and complete information on the environment state. It will be necessary to select actions with partial uncertainty and often the agents try actions aimed to grow the information and to reach the target in a more efficient manner. The domains in which the actions have probabilistic results and the agent has direct access to the environment, may be formalised as Markov Decision Process (MDP) [8]. MDP play an important role in the current AI research on planning [1,14,4,6], but the assumption of full-observability constitutes a serious obstacle to their application in the real world. Partially Observable Markov Decision Processes (POMPD) generalize MDP structure to the case in which the agent has to take a decision in a partial uncertainty context.

In the real world the information of an agent about its surroundings is necessarily incomplete: sensors are imperfect, objects occlude one another's view, the initial or the current position of the robot may not be precisely known. The theory of Partially Observable Markov Decision Processes (POMDP) models this situation and provides a basis for computing optimal behaviour. A POMDP is a Markov Decision Process (MDP) [2] in which the agent is unable to observe the current state. Instead it makes an observation based on the action and the resulting state. The goal of the agent is to maximize expected reward.

Of course if, as in our system, the memory concerns not only the previous state, the Markovian nature, in a narrow sense, of the process is altered.

Many algorithms are developed to solve the POMPD problem but many techniques are too inefficient for being general solutions. Using reinforcement learning techniques appeared as a satisfactory policy for a POMPD with high state number [9].

* Corresponding author. Tel.: +39 071 2204830; fax: +39 071 2204 835.
E-mail addresses: a.montesanto@univpm.it (A. Montesanto), tascinisas@tin.it (G. Tascini), p.puliti@deit.univpm.it (P. Puliti), p.baldassarri@univpm.it (P. Baldassarri).
[1] Tel.: +39 071 2204 449.
[2] Tel.: +39 071 2204900.

Following the treatment of [4] a POMDP is a tuple ⟨S, A, T, R, O, Ω⟩ where S is a set of states, A a set of actions, and Ω a set of observations. The functions T and R define a MDP with which the agents interact without direct information as to the current state. The transition function, $T : S \times A \rightarrow \Pi(S)$, specifies how the various actions affect the state of the environment. $\Pi(\cdot)$ represents the set of discrete probability distributions over a finite set. The notation $T[s, a, s']$ represents the probability that state s' will result from taking action a in state s.

The reward function $R : S \times A \rightarrow \Re$ specifies the playoffs of the agent. The playoff is a random function of the state and action with $R[s, a]$ representing the expected immediate reward for taking action a from state s.

The function $O : S \times A \rightarrow \Pi(\Omega)$ specifies the observation model. That is $O[s', a, o]$ the probability of observing o in state s' after having taken action a.

The objective of the agent is to generate actions so as to maximize its expected sum of reward. A reward receiving t steps in the future is only worth γ^t as much as it would be if it received them today. The variable γ is a factor strictly between 0 and 1 that controls how much future rewards are worth. The agent should choose actions to make the expected sum of discounted future rewards $E\left\{\sum_{i=0}^{\infty} \gamma^i R[s_i, a_i]\right\}$ as large as possible.

In the particular framework in which is part of our work, i.e. that one of a robotics navigation based on the visual servoing in which is necessary a selflocalization without map and a limited planning, it is worth saying to catch up a bordering zone to that one the agent is in, without hitting the walls. The environment taken into consideration is of the static type and office-like, without landmarks. This environment involves a space of the states which is very large. States indistinguishable between them (perceptual aliasing) even if topologically distant exist and they demand various actions (POMDP). Not having a map on which to base the knowledge, for navigation a reactive approach based on the reinforcement learning and using in particular the Q-learning must be used. A variety of algorithms have been developed in order to resolve the problem of the POMDP [11] and techniques of reinforcement learning have demonstrated a satisfactory politics for a POMDP with a high number of states [9]. The use of the Q-learning involves a serious problem for the time of learning. It is not simple to find an optimal policy and such an approach is very slow in converging towards optimal politics. In fact if it is indicated with $Q^*(s, a)$ the value of the discounted reinforcement attended as a result for an action a in the state s, and $V^*(s)$ like the value of s that assumes the best action, and therefore is indicated $V^*(s) = \max a Q^*(s, a)$. The Q^* (s function, a) can recursively be written like:

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') \max_{a'} Q^*(s', a')$$

in which optimal politics are defined also as: $\pi^*(s) = \arg\max_a Q^*(s, a)$. From the moment that the Q function lets the action be explicit, the values of Q can be estimated on-line, using them in order to define politics, since an action can be chosen considering the maximum value of Q for the running state. The principle of learning of the Q-learning is given by the following equation:

$$Q(s, a) = Q(s, a) + \alpha(R(s, a) - \gamma \max_{a'} Q(s', a') - Q(s, a)).$$

If every action is executed in every state infinitely many times on an infinite cycle and α is “discounted” in an appropriate way, the values of Q will converge with a probability range from 1 to Q^*. When the values of Q are near to converging to their optimal values, it is preferable that the agent acts gradually considering in every situation the action with higher Q value. Therefore, the Q-learning can converge slowly enough to one optimal policy. A solution can be that one to adopt the techniques of reinforcement learning based on the model through the techniques of dynamics programming. Therefore a model based on the last experience is used iteratively in order to improve politics until the convergence to optimal politicy. An example is supplied in the Dyna architecture [18] in which a model of the environment is constructed through the T' and R' functions, that approximate T and R, and stored the behaviour to hold through the Q function. The T and R functions are respectively the function of state transition and the function of the reinforcement. Every n-pla of experience $\langle s, to, r, s \rangle$ is exploited in order to improve the model and in order to improve the behaviour. To every step the model is improved updating the statistics for the state transitions and the reinforcement. Then the behaviour is improved based on the updated T' and R' functions, updating the Q function $k + 1$ times:

$$Q(s_j, a_j) := R'(s_j, a_j) + \gamma \sum_{s'} T'(s_j, a_j, s') \max_{a'} Q(s', a')$$

with j from 0 to k and $(s0, a0) = (s, a)$, while the other k couples are randomly chosen. This updating method is an iterative version of the technique based on the model [11]; carrying out only $k + 1$ updating, the computing necessities are limited. The choice of the action a' to complete in the new state s' is carried out alternating the exploitation of the actual Q function to a different choice, being based on some strategy of exploration.

Regarding the Q-learning, Dyna demands k times more computing for every step, but the convergence can be also of a faster order of magnitude. A defect of the Dyna architecture is in the random choice of the k update of the Q function that they are carrying out to every step: a better choice can carry to a reduction of the computing and to a speed of the learning. Two techniques that use this indication are the prioritized sweeping [13], Queue-Dyna [15] and [3]. In reinforcement learning the great spaces of state involve also memory problems. As an example the Q-learning must store in a table a number of Q values equal to the product of the number of the states multiplied by the number of the actions. Therefore in order to face the problem of the largeness of the space of the states two different approaches can be adopted: the techniques of generalizing and the hierarchical methods. In the first, the agent must be able to transfer the acquired knowledge through the experimentation of a couple (s, a), that it produces