Towards autonomous adaptive behavior in a bio-inspired CNN-controlled robot

P. Arena [§], L. Fortuna [§], M. Frasca [§], L. Patané [§], M. Pavone [*]

[*] Scuola Superiore di Catania, Via S. Paolo 73, 95123 Catania, Italy

[§]Dipartimento di Ingegneria Elettrica Elettronica e dei Sistemi
Università degli Studi di Catania
Viale A. Doria, 6 - 95125 Catania, Italy

Abstract— **This paper describes a general approach for the unsupervised learning of behaviors in a behavior-based robot. The key idea is to formalize a behavior produced by a Motor Map driven by an adaptive reward function. Aim of the adaptive reward function is to select the most significant sensory inputs and to use them in the best way. The greatest challenge is to keep small the search space. Motor Map learning relies on the classical Kohonen algorithm, while the structure of the reward function is learnt through a non-associative reinforcement learning algorithm. Simulation results on a six legged biologically-inspired robot confirm the suitability of the approach. This methodology allows the human designer to easily embody all the *a priori* knowledge on the robot controller, while providing at the same time a high degree of adaptability and robustness against the sensory malfunctioning.**

I. Introduction

Autonomy is the key concept in many robotics applications such as humanitarian demining, volcano and space exploration, pipe inspection or sewer maintenance. The major difficulty encountered in the design of autonomous mobile robots is the extreme variability of the robot environment; thus, a certain degree of adaptability is needed: *intelligent* autonomous robots should be able to extract information from the environment and use their built-in knowledge in order to learn to effectively act and adapt within that environment. How to confer adaptability to a robot? Basically, according to [1], four classes of control can be distinguished: reactive control, deliberative control, hybrid control and behavior-based control.

Reactive control tightly couples sensory inputs and effector outputs, to allow the robot to quickly respond to changing and unstructured environments, according to a biological "stimulus-response" scheme. Its limitations, however, include the fact that the robot does not have memory, internal representations of the world and the ability to learn over time. In *deliberative control*, the robot uses all of the available sensory information and all of the internally stored knowledge to reason about what actions to take, where reasoning is typically in the form of a planning algorithm. Unfortunately, this approach can rapidly become computationally prohibitive. *Hybrid control* combines the real-time response of reactivity with the rationality and optimality of deliberation. As a result, the control system contains two different components, the reactive and the deliberative ones. The interaction between them require an intermediary, whose design is typically the greatest challenge of hybrid systems. Finally, a *behavior-based system* is based on a representational substrate, the *behaviors*, which are observable patterns of activity emerging from interactions between the robot and its environment. Like hybrid systems, behavior-based systems may have different layers, but the layers do not differ drastically in terms of time-scale and representation used [1].

Behavior-based systems have been praised for their robustness and simplicity of construction [2] and seem to be a really suitable approach to autonomous robotics.

Basically, behavior-based systems are composed of a collection of "behavior producing" modules that map environment states into low-level actions, and a coordination mechanism that decides, based on the state of the environment, which behavior has to be executed. The task of programming in each individual behavior remains the burden of a human designer since it requires a deep knowledge of the interactions between a particular robot and its application to the environment, typically not available if an exploration mission is considered [2].

In this paper, we focus on the development of a learning algorithm that allows a mobile legged robot, controlled by a CNN-CentralPattern Generator, to learn single behaviors in a initially unknown environment. The key idea is to formalize a behavior as a Motor Map driven by an adaptive reward function. This approach allows the human designer to easily embody all its *a priori* knowledge on the robot controller, while providing at the same time a high degree of adaptability.

Simulation results on a biologically-inspired legged robot confirm the effectiveness of the approach in a complex case, dealing with a bio-inspired robot endowed with a large number of degree of freedom, controlled, at the low level, by a CNN acting and a Central Pattern Generator.

II. Behavior formalization

Different approaches have been proposed to implement a behavior, like stimulus-response diagrams, mathematical functional and finite state machines. Basically, a behavior is such any mapping from possible stimuli to possible responses. The difference with a reactive control is that behaviors do not require a tight coupling between stimuli and actions in the form of a *look up* table.

Since an adaptive mapping between stimuli and actions is needed, we chose to formalize a behavior as a Motor Map. For example, the behavior “Avoid obstacle” is translated into a Motor Map whose input is the obstacle distance and whose outputs are the motor commands that allow the whole structure to perform the behavior considered. In this way it is possible to learn a behavior through the classical Motor Map (MM) learning algorithm. Motor Map Behaviors (MMB) are thus adaptive and self organizing.

Since the robot is supposed to explore dynamic, hazardous and unknown environments, the definition of the Motor Map reward function for each behavior could represent a challenging task. Let us consider, for example, the “Pursuing target” behavior and let us suppose that the robot measures the intensity of two different chemicals: sensor performance could be not known in advance for the target environment; moreover, in case of sensor malfunctioning, the reward function could take into account a wrong signal with detrimental consequences for the behavior execution. As a consequence, the need of an adaptive reward arises.

We therefore introduce a high level learning system for the reward function of a MMB. To the best of our knowledge, it is the first time that an adaptive reward is considered. Aim of the adaptive reward function is to select the most significant sensory inputs and to use them in the best way. The greatest challenge is to keep small the search space.

In synthesis, we model a behavior as a Motor Map with adaptive reward function; Motor Map learning algorithm is the classical Kohonen algorithm, while the reward function learning algorithm is the subject of the next section.

III. Reinforcement learning of the MMB reward function

A. Reward function set

Let us consider a behavior p [1]; to this behavior we associate a set S of basic reward units r_j. The S set induces a general reward function R:

$$R = \sum_{j}^{N} -\alpha_j r_j^2 \tag{1}$$

where N is the total number of basic units and α_j is a binary weight (*i.e.* α_j can take values 0 or 1).

Eq. 1 represents a set of reward functions. A particular reward function is obtained by setting the α_j weights to the values 0 or 1; we will refer to a vector like $(1, 0, 1, 1 \ldots 0)$ as a reward function instantiation k and we will denote it as $\bar{\alpha}_k$.

One rule to follow, in order to avoid contradictory reward functions, is the exclusivity rule: *in a reward function instantiation the direct basic reward unit* (r_j) *and the inverse basic reward unit* ($1/r_j$) *can not be present at the same time.*

[1]For sake of clarity, henceforth we will suppress the index p.

Once the R set is formed, the leaning algorithm has to determine the optimal reward function instantiation, *i.e.* the reward function that best fits the environment.

S set definition could appear a difficult task. Nevertheless, all sensory information available to the robot are known *a priori*; what is unknown and has to be learnt is how to use all sensory inputs relevant for the behavior considered. Therefore the definition of basic reward units is indeed simple and can be made in the design phase.

The key point is that the designer, based on the fact that he knows which are the available information (the signals coming from the sensors placed on the robot), can easily decide generality and potentiality of the reward function set R; the reward function learning system will find during the mission the best choice for the unknown environment. Therefore, in the design stage, the designer has to accomplish two competing tasks:

- placing all his available *a priori* information in order to minimize the search space;
- guaranteeing behavior adaptability.

B. Reward function learning algorithm

The learning of the reward function can be achieved with a *non-associative reinforcement* learning, since the reward function form does not depend on the particular learning example, *i.e.* it does not make any difference if for the learning of the reward function we consider, referring for example to the “Climbing obstacle” behavior, an obstacle with height x instead of an obstacle with height $x + \Delta x$.

Fig. 1 shows the basic components of a non-associative reinforcement learning. The learning system’s actions influence

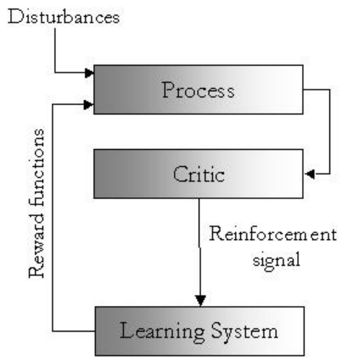

Fig. 1. Non-associative reinforcement learning [3].

the dynamics of some process, which might also be influenced by random or unknown factors (labelled “disturbances” in Fig. 1). A critic sends the learning system a reinforcement signal whose value at any time is a measure of the “goodness” of the current process behavior. Using this information, the learning system updates its action-generation rule, generates another action, and the process repeats [3]. In our framework, the action is the reward function form and the signal critic is the evaluation of behavior execution.

In order to test a specific reward function form, the underlying Motor Map has to be trained. Thus, at each MMB reward function learning step, the reward function learning system has to:

- select a reward function form;
- train the Motor Map with the selected reward function;
- evaluate the critic signal;
- update reward-generation rule.

When a reward function appears to be appropriate, the reward function learning stops and will start again only if a failure happens.

At first glance, Motor Map training constitutes a prohibitive obstacle, since it makes MMB reward function learning phase excessively long. Indeed, learning is very fast: it is enough to train the Motor Map *just in one input configuration*. This happens coherently with the fact that a non associative learning is used, therefore a reward function instantiation $\bar{\alpha}_k$ can be tested referring just to a particular learning example.

Suppose now that, on each trial, the learning system selects an instantiation $\bar{\alpha}_k$ from the set R of the M possible instantiations according to a probability vector $(p_1(t), \dots, p_M(t))$, where $p_k(t) = \Pr(\bar{\alpha}(t) = \bar{\alpha}_k(t))$, *i.e.* the probability that at trial t the actual reward function instantiation is the k^{th} one.

The proposed learning algorithm performs the following linear reward-penalty (L_{R-P}) method [3]: if instantiation k is chosen on trial t and the critic's feedback is "success", then $p_k(t)$ is increased and the probabilities of the other instantiations are decreased; whereas if the critic indicates "failure", then $p_k(t)$ is decreased and the probabilities of the other instantiations are appropriately adjusted.

In detail, if $\bar{\alpha}(t) = \bar{\alpha}_k(t)$ and, after Motor Map training, the critic says "success", then

$$\begin{aligned} p_k(t+1) &= p_k(t) + \beta(1 - p_k(t)) \\ p_l(t+1) &= (1-\beta)p_l(t), \quad l \neq k \end{aligned} \tag{2}$$

If $\bar{\alpha}(t) = \bar{\alpha}_k(t)$ and, after Motor Map training, the critic says "failure", then

$$\begin{aligned} p_k(t+1) &= (1-\gamma)p_k(t) \\ p_l(t+1) &= \frac{\gamma}{M-1} + (1-\gamma)p_l(t), \quad l \neq k \end{aligned} \tag{3}$$

where $0 < \beta < 1$ and $0 \leq \gamma < 1$.

In this way, after a transient, the most suitable instantiations are detected.

C. Remarks

1) Uncertainty: Uncertainty plays a key role in non-associative reinforcement learning through the selection of the reward function instantiation. For example, if the critic evaluated actions deterministically, then the problem would be a much simpler optimization problem.

2) Critic: The critic is an abstract entity that evaluates the learning system's actions. The critic does not need to have direct access to the actions or have any knowledge about the interior workings of the process influenced by those actions. The critic has just to measure the "goodness" of the current behavior according to some criteria; clearly, critic criteria depend on the specific robot behavior currently under learning.

3) Memory: The vector probability represents a kind of memory of past actions; if the best reward function changes structure over the time, for example due to sensory malfunctioning or different environmental conditions, the new learning phase is greatly facilitated by the memory of previous trials.

IV. Simulation set-up

We tested the effectiveness of the approach by considering the control of a six legged robot. The dynamic model of the robot was built in a C++ environment based on `DynaMechs` library [4]. The control system architecture has a two-levels hierarchical organization, as suggested by biological results discussed in [5]. The low level control is based on a CNN-based Central Pattern Generator (CPG), discussed in details in [6]; the CPG provides the basic rhythmic signals needed for locomotion. The high level is aimed to handle complex tasks. We model the high level as a set of Motor Map behaviors. In particular we focused on the learning of the "Pursuing target" behavior.

We assume that there are two chemicals originating at the target location. Let us suppose that the following information are available to the robot:

- chemical 1 intensity I_1;
- chemical 2 intensity I_2.

The first step is to associate to the "Pursuing target" behavior a Motor Map. The Pursuing Motor Map has:

- 12 neurons;
- 2 inputs (I_1 and I_2 intensity);
- 1 output (yaw angle).

In order to simplify the learning phase, we considered a winner-take-all strategy by selecting unitary neighborhood functions. The learning threshold value is $a = 0.01$, so that, after the learning phase, a residual plasticity for a later re-adaptation is guaranteed. The learning rate is $\eta = 0.5$ as a trade-off between speed and accuracy of learning.

The second step requires the definition of the S set. We define the S set for the Pursuing Motor Map as:

$$S_{climbing} = \{I_1, I_2, 1/I_1, 1/I_2\} \tag{4}$$

From the S set we can easily derive the corresponding R set, taking into account the exclusivity rule. Considering a instantiation vector notation, the R set has the following elements: $(1,0,0,0)$, $(0,0,1,0)$, $(0,1,0,0)$, $(0,0,0,1)$, $(0,0,1,1)$, $(0,1,1,0)$, $(1,0,0,1)$, $(1,1,0,0)$.

The final step is to define the critic signal: trivially, the critic for the Pursuing Motor Map indicates a "success" if, after an evaluation time t_{max}, the distance from the target is below a threshold $d_{treshold}$.

The simulation is articulated as follows: at the beginning the robot is placed in a starting position; an instantiation k is selected and the Motor Map is congruently trained for a fixed number of epochs ($N = 100$) with the same particular learning example (as a consequence of the non-associative learning of the reward function). Each epoch lasts four times the leg cycle time, *i.e.* in one epoch the robot performs four steps. At the end of each epoch the robot is placed again at its starting position. After 100 epoches the reinforcement signal is evaluated. At this time thus the effectiveness of the given reward is evaluated. In practice, this time is needed to evaluate how the given reward function is performing: to do this, a complete learning phase of the Motor Map (which requires 100 steps) should be accomplished. The value of the reinforcement signal is then used to update the probability vector according to equations (2) and (3). Then a new reward function instantiation is randomly chosen according to its associated probability as expressed by the probability vector. At each cycle of 100 epochs different robot initial conditions are chosen.

After some cycles, the reward function instantiations with better performance are characterized by higher probabilities. Thus, the simulation ends if a $p_k > 0.9$ is found, i.e. if a given reward function instantiation performs better. We set the reward function learning constants to the values $\beta = 0.5$ and $\gamma = 0.6$.

V. Simulation results

Firstly we assume that chemical 1 follows a gaussian distribution centered in the target location, while the chemical 2 is constant, thus not providing any information. The evaluation time is $t_{max} = 25$ s.

It is easy to notice that the suitable reward instantiations are: $(0, 0, 1, 0)$, $(0, 0, 1, 1)$, $(0, 1, 1, 0)$. These instantiations, in fact, induce a strategy that leads to a maximization of the chemical 1 intensity (in fact, if I_1 increases, $1/I_1$ decreases and thus R increases). Congruently, in simulation we obtain that all and only these three instantiations are selected as suitable reward function candidates.

In Fig. 2 the temporal evolution of the probability vector is shown: violet bar refers to the selected instantiation. This shows that at the end of the whole simulation the configuration labelled as $k = 6$, *i.e.* $(0, 1, 1, 0)$, has became prevalent. In fact, p_6 increases each time the reward instantiation $k = 6$ is chosen. When, for instance, at the first simulation cycle, $k = 1$ is chosen, since it does not correspond to a reward function performing well, its probability p_1 is decreased.

In a second experiment we assume that both chemicals spread with the same gaussian distribution, but the detector of second chemical measures with some noise. The evaluation time is again $t_{max} = 25$ s. Simulation results show that the robot is able to learn to disregard the second chemical and to just rely on the maximization of the first chemical by selecting the instantiation 2. This last example shows the effectiveness of the proposed high level control in case of sensory malfunctioning.

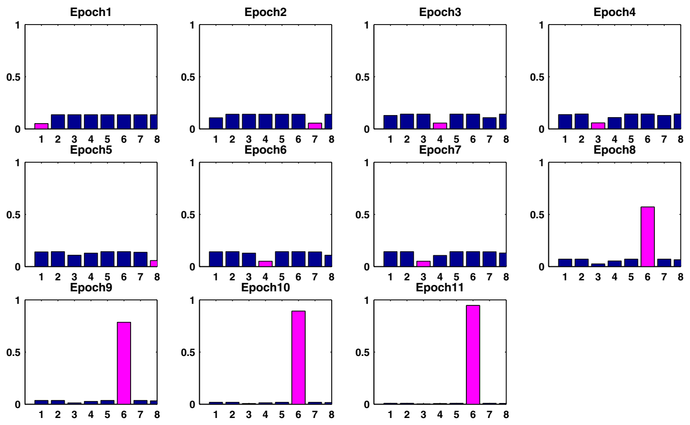

Fig. 2. Evolution of the probability vector.

VI. Conclusion

Adaptation in dynamic and unpredictable environments is a very challenging problem. The goal of our approach is to enable a robot to learn and dynamically utilize the interaction with its environment from self-evaluation of its behavior. We propose a general approach for the unsupervised learning of behaviors in a behavior-based robot. The key idea is to formalize a behavior as a Motor Map driven by an adaptive reward function. The advantages of this methodology are that the human designer can easily embody all its *a priori* knowledge on the robot controller, while providing at the same time a high degree of adaptability and robustness against sensory malfunctioning. Much work remains to be done in the temporal coordination of the behaviors.

Acknowledgment

The authors acknowledge the support of the European Commission under the Project FP6-2003-IST2-004690 SPARK.

References

[1] M.J. Mataric *Learning in behavior-based multi-robot systems: policies, models, and other agents.* Journal of Cognitive System Research, 2(1), pp. 81–93, 2001.
[2] F. Michaud and M.J. Mataric. *Learning from History for Behavior-Based Mobile Robots in Non-Stationary Conditions.* Autonomous Robots, 5, pp. 335–354, 1998.
[3] O. M. Omidvar and D. L. Elliott. *Neural Systems for Control.* Academic Press, 1997.
[4] S. McMillan, D. E. Orin and R. B. McGhee. *A computational framework of underwater robotic vehicle systems.* J. Auton. Robots - Special Issue On Autonomous Underwater Robots, 3, pp. 253–268, 1996.
[5] G. M. Shepherd *Neurobiology.* Oxford Univ. Press, 1997.
[6] M. Frasca, P. Arena, L. Fortuna. *Bio-Inspired Emergent Control Of Locomotion Systems.* World Scientific Series on Nonlinear Science, Series A - Vol. 48, 2004.