Visual approach skill for a mobile robot using learning and fusion of simple skills

M.J.L. Boada*, R. Barber, M.A. Salichs
System Engineering and Automation Division, Carlos III University, Madrid Spain

Received 30 August 2001; received in revised form 30 September 2001; accepted 30 November 2001

Abstract

This paper presents a reinforcement learning algorithm which allows a robot, with a single camera mounted on a pan tilt platform, to learn simple skills such as watch and orientation and to obtain the complex skill called approach combining the previously learned ones. The reinforcement signal the robot receives is a real continuous value so it is not necessary to estimate an expected reward. Skills are implemented with a generic structure which permits complex skill creation from sequencing, output addition and data flow of available simple skills. © 2002 Published by Elsevier Science B.V.

Keywords: Mobile robots; Reinforcement learning; Visual approach; Control architecture; Neural networks; Adaptive behavior

1. Introduction

An autonomous mobile robot must be able to modify or adapt its skills in order to react adequately in complex, unknown and dynamic environments. A good method for achieving this goal is reinforcement learning [11] because a complete knowledge of the environment is not necessary and also permits the robot to learn on-line.

The more complex a task performed by a robot, the slower the learning, because the number of states increases so that it makes it difficult to find the best action. The task decomposition in simpler sub-tasks permits an improvement of the learning because each skill learns in a subset of possible states, so that the search space is reduced.

The current tendency is to define basic robot behaviors, which are combined to execute more complex tasks. Thus, Brooks [6] proposes a hierarchical decomposition based on behaviors. Gachet et al. [8] define a set of elemental or primitive behaviors. The robot learns to merge these behaviors adequately in order to execute more complex tasks called emergent behaviors. Becker et al. [3] define a flexible library of robot skills that can be easily recombined to obtain a variety of useful behaviors.

Several authors have proposed different methods to learn not only simple behaviors, but also how to combine them to obtain more complex ones. So, Michaud and Mataric [12] provide a memory-based approach to dynamically adapt the behaviors selection according to the history of their use. Ryan and Pendrith [15] introduce the RL-TOPs architecture which allows the automatic construction of appropriate hierarchies of learned behaviors and provides an approach to re-use learned behavior for solving another task. This improves the learning time. Hasegawa and Fukuda [10] propose a hierarchical behavior controller and a
learning algorithm to obtain more complex behaviors combining previously obtained behaviors.

Our research is based on the work done by Barber [2]. He defines an architecture consisting of a deliberative and an automatic level. The deliberative level is formed by the reasoning skills which require a high computing time. The automatic level is formed by skills which interact with the sensors and the actuators. One of the objectives of our research is to develop a reinforcement learning algorithm which allows a mobile robot equipped with a vision system formed by a single CCD camera mounted on a pan tilt platform to learn simple skills such as watch and orientation. The vision system is the only mean by which the robot is able to obtain information from the environment. In the proposed learning algorithm, the robot receives a real continuous reinforcement every time it performs an action. Once the robot has learned the previous simple skills, it performs the complex skill called approach coordinating them. This paper also presents the generic skills’ structure in the AD architecture and three different methods to obtain complex skills.

2. AD architecture

The control architecture used in this work is the AD architecture [2]. The motivation of the AD architecture is the human being reasoning capacity and the actuation capacity. According to the theories of modern psychology of Shiffrin and Schneider [16,17], there are two mechanisms for processing information: reflexive processes and the automatic ones. Therefore, two mental levels of activity can be differentiated in human beings: the deliberative level and the automatic one. These two levels are related to how the reasoning capacity and the actuation capacity are distributed. The three layer architectures developed by Firby [7], Gat [8] and Bonasso co-workers [4] are an example of the current trend to incorporate deliberation and reactivity in the same architecture. These architectures can be considered as a precedent of the AD architecture.

According to the criteria exposed, only two levels can be established, as shown in Fig. 1. The deliberative level is associated with the reflexive processes. The automatic level is the one associated to automatic processes. The deliberative level will be composed by those processes which require a long time of calculation as a consequence of reasoning. The path planner, the environment Modelling and the task supervisor are skills which are found in this level. The automatic level is formed by the skills which interact with the sensors and the actuators and which require minimum time to process the information they work with. Among them are modules which provide the sensorial information and the action modules upon the different mechanical elements of the robot can be found. Both levels present the same characteristic: they are formed by skills. Skills consist of the different reasoning capacities or the sensorial and motor capacities of the system. Those skills are activated by execution orders produced by other skills or by a sequencer. These skills return data and events to the skills or sequencers which have activated them. Those skills are the base of the AD architecture.

2.1. Deliberative level

In this level we find modules that require reasoning or decision capacity. Those modules do not produce immediate responses. They need to process the information they work with. Those modules will form the deliberative skills, and they will be activated by a sequencer, that will be in charge of managing the correct performance of these skills. Fig. 2 shows this level.

This level is formed by a series of skills named deliberative skills, a long-term memory where information is obtained as well as a sequencer that activates and deactivates the deliberative skills:

- **Deliberative skills.** These are each of the capacities of reasoning and learning which the autonomous system has. Examples of these skills are the planners and the relocalization systems. These skills require...
Fig. 2. Deliberative level.

Fig. 3. Automatic level.
the need for a long period of calculation. They activate and deactivate one by one, with concurrency not being possible.

- **Long-term memory.** It contains information which can be considered more stable through time, i.e., it does not depend on the robot’s state. This type of memory is only accessible by the deliberative skills, which can perform reasoning upon that information, modifying the information when necessary. The a priori information, such as the maps, and the information originated from the reasoning or learning of the different deliberative skills, will be included in this memory.

- **Sequencer.** The sequencer is in charge of managing the deliberative skills, giving the execution order to each one when necessary. This sequencer is given a priori and it is the one which defines the system behavior. The sequencer decides which skills should be active depending on their sequence and on the skills events of this level as well as those of the automatic level.

### 2.2. Automatic level

In this level, there exists low level control modules which act directly upon the actuators, as well as the modules that collect data from the different sensors of the system. Fig. 3 shows the different elements that form the automatic level:

- **Automatic skills.** These are the sensorial and motor capacities of the system. The automatic skills’ response is faster than the deliberative skills’ response.
- **Reflex actions.** These are involuntary and priority responses to a stimulus. The intensity and duration of the response is governed by the intensity and duration of the stimulus.

### 3. Automatic skills

Automatic skills are in charge of processing sensorial information and/or executing actions upon the robot actuators. Bonasso et al. [5] define skills as the robot’s connection with the world. For Chatila and co-workers [1], skills are all built-in robot action and perception capacities.

Skills are classified as perceptive and sensorimotor. Perceptive skills interpret the information perceived from the sensors, perceptive skills or sensorimotor skills. Sensorimotor skills perceive information from the sensors, perceptive skills or sensorimotor skills and on the basis of that perform an action upon the actuators. Automatic skills can be combined to obtain more complex ones.

#### 3.1. Skill’s structure

Skills are server/client modules. Each module contains an active object, an event manager object and data objects. Objects are separate units of software with an identity, interfaces and state. The active object has its own thread of control and it is in charge of processing. The processing results are stored in the data objects. These objects contain different data structures depending on type of data stored but the interfaces are similar. During the processing, the active object can generate events. Events are sent to the event manager object which is in charge of notifying them of skills which have registered on it. In order to communicate among objects of the same module or different modules Corba is used.

Skills can be activated or deactivated by skills situated in the same level or in higher levels. During the activation, some parameters can be sent to the activated skill. When a skill is activated, it connects to data objects of other skills or to sensors’ servers as required by the skill. Then, it processes the received input information, and finally, it stores the output results in its data objects. When the skill is sensorimotor, it can connect to actuators’ servers in order to send them movement commands.

A skill can send a report about its state while it is active or when it is deactivated. For example, the skill called go_to_goal can inform on whether the robot has achieved the goal or not. When this skill is deactivated it might inform about the error between the current robot position and the goal.

Skills must define an interface independent from the robot’s physical characteristics to allow, on one hand, the communication among skills and, on the other hand, the software portability from one robot to another robot.

Fig. 4 shows the skill’s structure. The black squares represent the data objects where the output results are
stored. The ellipse represents the active object. If a skill does not have the white circle this means it is always active. The white square represents the events manager object. Solid, dashed and dotted arrows represent the data flow, the performance flow and the events flow, respectively.

3.2. Complex skills

Skills can be combined to obtain complex skills and these skills can also be recursively combined to obtain more complex skills. This presents the following advantages:

- Re-using of software. A skill can be used for different complex skills.
- Reducing the programming complexity. The problem is divided into smaller and simpler problems.
- Improving the learning rate. Each skill is learned in a subset of possible states, so that the search space is reduced.

3.3. Methods for generating complex skills

We propose three different methods to generate complex skills: sequencing, fusion and data flow.

- **Sequencing**. The sequencer is responsible for deciding what skills have to be activated in each moment and has to avoid activating at the same time skills which act upon the same actuator, see Fig. 5.

- **Output addition**. Fusion allows combining the outputs of more than one sensorimotor skills that act upon the same actuator at the same time. The resultant movement commands are obtained combining the movement commands of each skill. In this case, the skills are activated at the same time, see Fig. 6.

In the sequencing method, simple skills connect directly to actuators’ servers. In the output addition method simple skills have to store the movement commands in their data objects in order to be able to be used by the complex skill. When simple skills are activated, they do not know if they have to send commands to actuators or store them in their data objects. In order to solve this problem, simple skills check if there is another skill connected to actuators. In the negative case, they connect them and send movement orders. In the affirmative case, they store the commands in their data objects. In the output addition method, first of all, complex skill connects to actuators and then activates simple skills.

- **Data Flow**. A complex skill can also be made of skills which send information one to the other, see Fig. 7. The difference with the above methods is that the complex skill does not have to be responsible for activating all the skills. Simple skills can activate other simple skills.

The three methods are not exclusive; they can occur in the same skill. A generic complex skill must have a structure which allows its generation by one or
Fig. 5. Sequencing method for generating complex skill.

Fig. 6. Output addition method for generating complex skill.

Fig. 7. Data flow method for generating complex skill.
more of the methods described above. The complex skill connects to the actuators servers before activating any simple skill. The movement commands sent by the simpler skills are passed through the complex skill. Fig. 8 show the generic structure with the three possible methods of generating a complex skill.

3.4. Skill learning

An autonomous mobile robot must be able to modify and improve its skills in order to carry out its tasks well in complex, dynamic and unknown environments. In this case, the learning from the experience plays a very important role. A robot should be able to learn simple skills, which outputs it has to generate from the perceived inputs, and it should be able to learn how to combine skills to obtain more complex skills. The work presented in this paper is focused on simple skill learning.

4. Approaching skill description

Approaching a target means moving towards a stationary object [14]. In the process the human performs to execute this skill using visual feedback is first of all to move his eyes and head to center the object in the image and then to align the body with the head while he is moving towards the target. Humans are not able to perform complex skills when they are born, they undergo a development process where they are able to perform more complex skills through the merging or coordination skills which have been learned. According to these ideas, our robot learns independently to maintain the object in the image center and to turn towards the base to align the body with the vision system and finally to execute the approaching skill coordinating the learned skills. Fig. 9 shows the diagram for the approach skill. This skill is generated by the data flow method. As the figure shows, the object center skill is not activated by the complex skill; it is activated by the skill called watch. There is a data flux between one skill to another.

4.1. Watching skill

Watching a target means keeping the eyes on it. The received inputs are the object center coordinates in the image plane and the performed outputs are the pan tilt velocities. The information is not obtained from the camera sensor directly but it is obtained by the skill called object center. This skill receives an image, finds the searched for object in the image, and calculates the object center. This last skill is perceptive because it does not produce any action upon actuators but it only interprets the information obtained from the sensors.

4.2. Robot orientation skill

Orientating the robot means turning the robot’s body to align it with the vision system. The turret is mounted on the robot so the angle formed by the
robot body and the turret coincides with the turret angle. The input is the turret pan angle and the output is the robot angular velocity. The information about the angle is obtained from the encoder sensor placed on the pan tilt platform.

4.3. Object center skill

Object center means searching for an object on the image previously defined. The input is the image recorded with the camera and the output is the object center position on the image in pixels. If the object is not found, the skill sends a report saying the object has not been found.

5. Reinforcement learning

Reinforcement learning is a learning technique based on trial and error. A good performance action provides a reward, increasing the probability of recurrence. A bad performance action provides punishment, decreasing the probability. Reinforcement learning is used when there is not detailed information about the desired output. The system learns the correct mapping from situations to actions maximizing a scalar, called a reinforcement signal, without a priori knowledge of its environment. Another advantage that the reinforcement learning presents is that the system is able to learn on-line, so that the system can adapt to changes produced in the environment.

The external reinforcement signal normally is not enough. In worst cases, it says only if the system is still working or has crashed. In these cases, the reinforcement signal is a scalar value, typically \{0,1\}. 0 means bad performance and 1 means good performance, and/or is delayed in time. To cope with this problem, the reinforcement learning algorithms such as TD(\lambda) [19], and Q-Learning [20], are based on estimating an expected reward.

In our learning algorithm, when the robot performs an action, it receives a reinforcement, real value between 0 and 1. This value shows how well the robot has performed the action. The robot can compare the action result with the last action result performed in the same state, so that it is not necessary to estimate an expected reward.

In our case, both the states space and outputs space are continuous. If the set of actions were discrete, some feasible solution would not have been taken into account.
6. Network architecture

Fig. 10 shows the neural network architecture used. The input layer consists of radial basis function (RBF) nodes. This layer is used to discretize the input space. The activation value for each node depends on the input vector proximity to the center of each node thus, if the activation level is 0 it means that the perceived situation is outside its receptive field. But if it is 1, it means that the perceived situation corresponds to the node center. The output layer consists of linear stochastic units allowing the search for better responses in the action space. Each output unit represents an action. There exists a complete connectivity between the two layers.

6.1. Input layer

The input space is divided into discrete, overlapping regions using RBF nodes. The activation value for each node is

$$i_j = \exp\left(-\left\|\vec{i} - \vec{c}_j\right\|^2 / \sigma_{RBF}^2\right)$$

where \(\vec{i}\) is the input vector, \(\vec{c}_j\) is the center of each node and \(\sigma_{RBF}\) the width of the activation function. The obtained activation values are normalized.

$$i_j^* = \frac{i_j}{\sum_{i=1}^{n_n} i_i}$$

where \(n_n\) is the number of created nodes. The nodes are allocated dynamically where they are necessary [13] maintaining the network structure as small as possible. Each time a situation is presented to the network, the activation value for each node is calculated. If all the values are lower than a threshold, \(a_{\text{min}}\), a new node is created. The center of this new node coincides with the input vector presented to the neural network, \(\vec{c}_i = \vec{i}\).

The connection weights, between the new node and the output layer, are initialized to randomly small values.
6.2. Output layer

The output layer must find the best action for each situation. The recommended action is a weighted sum of the input layer given values

$$ o^i_k = \sum_{j=1}^{n_o} w_{kj} \cdot i^j_j, \quad 1 \leq k \leq n_o, \quad (3) $$

where $n_o$ is the number of output layer nodes. During the learning process, it is necessary to explore for the same situation all the possible actions to discover the best one. The real final action is obtained from a normal distribution centered in the recommended value and with variance $\sigma$

$$ \sigma_k = \sqrt{\text{error}^2 / 2} \quad (4) $$

As the system learns a suitable action for each situation, the value of $\sigma$ is decreased. We state that the system can perform the same action for the learned situation [18].

In order to improve the results, the weights of the output layer are adapted according to

$$ \mu_{kj}(t+1) = \nu \cdot \mu_{kj}(t) + (1 - \nu) \cdot e_{kj}(t) \quad (8) $$

$$ e_{kj}(t) = \frac{o^j_k - o^j_j}{\sigma} \cdot i^j_j \quad (7) $$

$$ \mu_{kj}(t+1) = v \cdot \mu_{kj}(t) + (1 - v) \cdot e_{kj}(t) \quad (8) $$

where $\beta$ is the learning rate, $\mu_{kj}$ is the eligibility trace and $e_{kj}$ is the eligibility of the weight $w_{kj}$, and $v$ is a value in the [0, 1] range. The weight eligibility measures how this weight influences in the action, and the eligibility trace allows rewarding or punishing not only the last action but the previous ones. $r_{kj}$ is obtained from the expression

$$ r_{kj}(t) = \begin{cases} r_{ext}(t) & \text{if } i^j_j \neq 0 \\ r_{kj}(t-1) & \text{otherwise} \end{cases} \quad (9) $$

where $r_{ext}$ is the exterior reinforcement. Actions’ results depend on the activated states, so that only the reinforcement values associated with these states will update.

6.3. Reinforcement signal

The robot receives an external reinforcement signal, $r_{ext}$, after the action is performed, so that the learning rate increases. Its value is included in the [0, 1] range, where 0 means a bad evaluation and 1 means a good evaluation. Our reinforcement signal has the following form:

$$ r_{ext} = \exp^{-k(\text{error}^2 / 2)} \quad (10) $$

For the watching skill the error is

$$ \text{error} = \sqrt{(x_{oc} - x_{ic})^2 + (y_{oc} - y_{ic})^2} \quad \text{in pixels} \quad (11) $$

where $x_{oc}$ and $y_{oc}$ are the object center coordinates in the image plane and $x_{ic}$ and $y_{ic}$ are the image center coordinates.

For robot orientation skill the error is

$$ \text{error} = \text{angle turret - \text{angle robot}} \quad \text{in radians} \quad (12) $$

where $\text{angle turret}$ is the angle formed by the robot body and the turret.

---

Fig. 11. Robot RWI-B21.
7. Experimental results

The experimental results have been carried out on an RWI-B21 mobile robot shown in Fig. 11, equipped with a vision system formed by a single monochrome CCD camera mounted on a pan tilt platform. The robot has two computers. One PC is dedicated to images acquisition and processing. Vision processing hardware is formed by a Matrox card. The other PC is dedicated to running the movement modules of the robot and of the pan tilt platform and to execute the robot skills.

In the watching skill, the robot must learn the mapping from the object center coordinates \((x, y)\) to the turret motors \((\text{pan}, \text{tilt})\). In our experiment, a cycle starts with the target on the image plane at an initial position \((243, 82)\) pixels, and ends when the target comes out of the image or when the target reaches the image center \((0, 0)\) pixels and stays there. The turret pan tilt movements are coupled so that a \(x\)-axis movement involves a \(y\)-axis movement and vice versa. This makes the learning task difficult. Fig. 12 shows the robot performance while learning the watching skill. The plots represent the \(X-Y\) object coordinates on the image plane. As seen in the figure, the robot is improving its performance while it is learning. In the first cycles, the target comes out of the image in a few learning steps, while in cycle 6 the robot is able to center the target on the image rapidly. The robot learns to center the object from different positions on the image without forgetting what it has already learned. The learning parameters values are \(\beta = 0.01, \mu = 0.3, \sigma_{\text{RBF}} = 0.2\) and \(\alpha_{\text{max}} = 0.2\).

Once the robot has achieved a good level of performance in the watching skill, it learns the orientation skill. In this case, the robot must learn the mapping from the turret angle \((\text{pan})\) to the robot angular velocity \((\dot{\theta})\). To align the robot’s body with the turret, maintaining the target in the center image, the robot has to turn an angle. Because the turret is mounted on the robot’s body, the target is displaced on the image. The learned watching skill obliges the turret to turn to center the object so the robot’s body-turret angle decreases. The experimen-
tal results for this skill are shown in Fig. 13. The plots represent the robot’s angle as a function of the number of learning steps. In this case, a cycle starts with the robot’s angle at $-0.61$ radians, and it ends when the body is aligned with the pan tilt platform. As Fig. 13 shows, the number of learning steps is decreased. The learning parameters values are $\beta = 1.0$, $\mu = 0.3$, $\sigma_{RBF} = 0.1$ and $a_{\text{min}} = 0.2$.

Once the robot has learned the above skills, the robot is able to perform the approaching skill by coordinating them. Fig. 14 shows the results. This experiment consist of the robot going towards a goal which is a visual target. First of all, the robot moves the turret to center the target on the image and then the robot moves towards the target. The plot represents the robot’s X-Y positions where the circle is the robot and the X is the goal. The robot starts with the angle at 0 radians and with the position at (0, 0). Position information is obtained from the robot odometry.

8. Summary and conclusions

We have proposed a reinforcement learning algorithm which allows a robot to learn simple skills. The main advantages of this algorithm are that it is not necessary to estimate an expected reward because the
robot receives a real continuous reinforcement signal each time it performs an action and that the learning is on-line so that the system can adapt to changes produced in the environment.

This paper also present a generic structure definition to implement perceptive and sensorimotor skills in a robot. This structure allows the generation of complex skills from three different methods. All skills have the same characteristics; they can be activated by other skills from the same level or from a higher level, the output data can be stored in data objects in order to be used by other skills, and skills notify events to other skills which want to receive notification.

We have applied our algorithm to an autonomous mobile robot, equipped with a vision system, being able to correctly perform the complex skill called approach from other learned simple skills such as watch and orientation.

Acknowledgements

The authors gratefully acknowledge the funds provided by the Spanish Government through the CICYT project TAP1999-214.

References


Maria Jesus Lopez Boada received the Industrial Engineering degree from Carlos III University of Madrid in 1996. Since 1997 she is a research assistant in Systems Engineering and Automation at the University Carlos III of Madrid. Currently she is a PhD candidate working on the development of skills in an autonomous mobile robot and skills learning.
Ramon Barber is a research assistant at the System Engineering and Automation Unit, at the University Carlos III of Madrid, Spain. He received BSc degree in Industrial Engineering from Polytechnical University of Madrid in 1994, and the PhD degree in Industrial Technologies from the University Carlos III. In 2000, he has developed a new control architecture for mobile robots based on topological navigation. His current research area is automatic generation of topological maps. He is a member of the International Federation of Automatic Control (IFAC).

Miguel A. Salichs received the Electrical Engineering and PhD degrees from Polytechnical University of Madrid in 1978 and 1982, respectively. He is currently a Full Professor at Systems Engineering and Automation Unit at the University Carlos III of Madrid. He is Chairman of the Technical Committee on Intelligent Autonomous Vehicles of the International Federation on Automatic Control (IFAC). He has published more than 80 papers on robotics and automation. His primary research interests are mobile robotics, intelligent autonomous systems and service and personal robots.